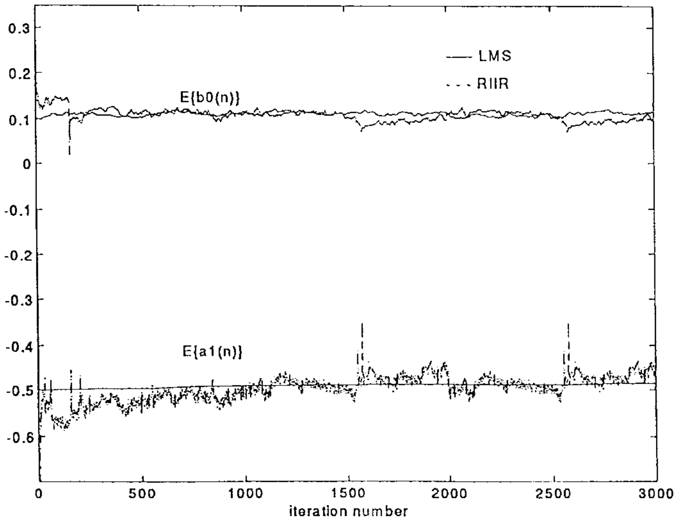

Fig. 5. Comparison of ensemble average of $a_1(n)$ and $b_0(n)$ between the LMS and the RIIR algorithms, for example, three.

two minima appear simultaneously in the instantaneous error surface as data is obtained. With such properties of the transient error surface, convergence of fast OE algorithms to the global minimum when initialized near the local minimum is unpredictable. Instantaneous descent directions around each minimum point to the corresponding minimum. Therefore, when the starting point is near the region of the local minima, the LMS will always converge to the local minimum.

The RIIR and LMS algorithms are applied to the above example. λ is set to 0.99 for the RIIR algorithm. The LMS is used with $\mu_{\text{LMS}} = 0.005$. The algorithms are initialized near the local minimum at $[b_0(0), a_1(0)] = [0.1, -0.5]$. As expected, Fig. 5 demonstrates the local convergence of both algorithms. This behavior is in agreement with our earlier discussion.

IV. Conclusion

In this correspondence, it was demonstrated that when the global minimum of the OE surface is the main attractor point of the transient error surface, convergence to the global minimum is achieved by fast converging algorithms. Properties of the error surface are directly determined by the combination of the input signal statistics, sufficiency/insufficiency of the IIR filter order relative to the unknown system, and the location of the unknown system poles. Possible identification of situations when the global minimum is the unique one at early formation of the error surface is an issue that needs further investigation.

References

[1] S. L. Netto *et al.*, "Adaptive IIR filtering algorithms for system identification: A general framework," *IEEE Trans. Educ.*, vol. 38, pp. 54–66, Feb. 1995.

[2] J. J. Shynk, "Adaptive IIR filtering," *IEEE Acoust., Speech, Signal Processing Mag.*, Apr. 1989.

[3] T. Söderström and P. Stoica, "Some properties of the output error method," *Automatica*, vol. 18, pp. 93–99, 1982.

[4] S. D. Stearns, "Error surfaces of recursive adaptive filters," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-29, pp. 763–766, June 1981.

[5] K. Mayyas and T. Aboulnasr, "A coupled LMS-LS approach to output-error adaptive IIR filtering," in *Proc. 4th ISSPA*, Aug. 1996.

[6] V. L. Stonick, "Time-varying performance surfaces for adaptive IIR filters: Geometric properties and implications for filter stability," *IEEE Trans. Signal Processing*, vol. 43, pp. 29–42, Jan. 1995.

[7] H. Fan, and W. K. Jenkins, "A new adaptive IIR filter," *IEEE Trans. Circuits Syst.*, vol. CAS-33, pp. 939–947, Oct. 1986.

A Family of Adaptive Filter Algorithms with Decorrelating Properties

Markus Rupp

***Abstract*—Although the normalized least mean square (NLMS) algorithm is robust, it suffers from low convergence speed if driven by highly correlated input signals. One method presented to overcome this problem is the Ozeki/Umeda affine projection (AP) algorithm. The algorithm applies update directions that are orthogonal to the last P input vectors and thus allows decorrelation of an AR(P) input process, speeding up the convergence.**

This correspondence presents a simple approach to show this property, which furthermore leads to the construction of new algorithms that can handle other kinds of correlations such as MA and ARMA processes. A statistical analysis is presented for this family of algorithms. Similar to the AP algorithm, these algorithms also suffer a possible increase in the noise energy caused by their pre-whitening filters.

***Index Terms*—Adaptive filter, affine projection, ARMA process, decorrelation, echo cancellation, minimum-norm least-squares solution.**

I. Introduction

Adaptive filters have become standard solutions in many fields in the last decades. The celebrated least-mean-squares (LMS) algorithm [1] and its normalized version (NLMS) is certainly one of the most referenced algorithms for adjusting the filter parameters. Most implementations of adaptive filters use this algorithm because it is robust [2] and of low complexity. However, one of its major drawbacks is its convergence speed that depends on the statistics of the input signal. Signal correlation tends to lower the convergence speed. This has lead to algorithmic modifications that aim to decorrelate the input sequence. Ozeki and Umeda [3] have used an affine projection (AP) to speed up the convergence, a method for which a fast version has also been proposed by Gay [4], [5] for acoustical echo compensation, thus making the complexity tractable.

In [6]–[8], the AP algorithm has been investigated among others, and some fundamental decorrelating properties were found. This correspondence will extend these results for general moving average (MA) and ARMA processes. The following paragraph will provide the necessary notation used in this correspondence. Vector quantities will be denoted by boldface lowercase letters and matrix quantities by boldface capital letters. The algorithms considered are recursive; thus, the new estimate $\hat{\mathbf{w}}(k+1)$ of the M-dimensional unknown plant parameter $\mathbf{w}$ is computed from the old estimate $\hat{\mathbf{w}}(k)$ by a

Manuscript received February 2, 1996; revised September 9, 1997. The associate editor coordinating the review of this paper and approving it for publication was Dr. Stephen M. McLaughlin.

The author is with Lucent Technologies, Wireless Research Laboratory, Holmdel, NJ 07733-0400 USA.

Publisher Item Identifier S 1053-587X(98)02009-1.

correction term. The algorithm update can be written as

$$\hat{\mathbf{w}}(k+1) = \hat{\mathbf{w}}(k) + \alpha\, \frac{\boldsymbol{\phi}(k)}{\boldsymbol{\phi}^T(k)\boldsymbol{\phi}(k)}\, e(k) \tag{1a}$$

$$e(k) = d(k) - \hat{\mathbf{w}}^T(k)\mathbf{u}(k) \tag{1b}$$

where $\mathbf{u}(k) = [u(k), \cdots, u(k-M+1)]$ is a collection of M past input samples $u(k)$. Vector $\boldsymbol{\phi}(k)$ will be denoted as a direction vector since $\boldsymbol{\phi}(k)$ defines the direction of the update. The desired signal $d(k)$ consists of the plant output $\mathbf{w}^T\mathbf{u}(k)$ plus additive noise $v(k)$, i.e., $d(k) = \mathbf{w}^T\mathbf{u}(k) + v(k)$. Defining the weight-error vector $\boldsymbol{\epsilon}(k) = \mathbf{w} - \hat{\mathbf{w}}(k)$ (the difference between the true system parameters and their estimates), (1a) can be written into state-space form

$$\boldsymbol{\epsilon}(k+1) = \boldsymbol{\epsilon}(k) - \alpha\, \frac{\boldsymbol{\phi}(k)}{\boldsymbol{\phi}^T(k)\boldsymbol{\phi}(k)}\, [\mathbf{u}^T(k)\boldsymbol{\epsilon}(k) + v(k)]. \tag{2}$$

For $\boldsymbol{\phi}(k) = \mathbf{u}(k)$, the well-known normalized LMS algorithm (NLMS) is obtained. The step-size α is often used to control the algorithm and improve its adaptation to a changing environment. In the following, we will mainly consider the case $\alpha = 1$ for which fastest convergence is obtained for time-invariant systems. It will be shown that there exist particular choices for $\boldsymbol{\phi}(k)$ that allow rewriting of the update into the symmetrical form

$$\boldsymbol{\epsilon}(k+1) = \boldsymbol{\epsilon}(k) - \frac{\boldsymbol{\phi}(k)}{\boldsymbol{\phi}^T(k)\boldsymbol{\phi}(k)}\, [\boldsymbol{\phi}^T(k)\boldsymbol{\epsilon}(k) + v_f(k)]. \tag{3}$$

These symmetrical algorithms have principally the same behavior as the NLMS; however, there are two major differences.

- The noise will be modified by a linear (time-varying) filter and, thus, can potentially increase.
- The correlation of the elements in $\boldsymbol{\phi}(k)$ can be reduced by choice of $\boldsymbol{\phi}(k)$, and therefore, the convergence speed can be increased.

Note that the opposite effect can also be achieved. Noise with a known correlation can be decorrelated, and therefore, its effect can be lowered by decreasing the convergence speed.

If the input process is correlated, a method as described in [1], [9], and [10] can be used to describe the convergence behavior of a symmetrical scheme by rewriting it into a linear system of the form

$$\mathbf{c}(k+1) = \mathbf{B}\mathbf{c}(k) + \mathbf{h}. \tag{4}$$

The matrix $\mathbf{B}$ and the vector $\mathbf{h}$ describe the evolution of the weight vector variances in $\mathbf{c}(k)$. They consist of the step size and the eigenvalues of the autocorrelation matrix of the input process and may depend on various moments of the input statistics [10]. The largest eigenvalue of the matrix $\mathbf{B}$ determines the slowest mode of convergence, and the vector $\mathbf{h}$, together with $\mathbf{B}$, defines the misadjustment.

Statistical analysis of NLMS has shown for white Gaussian [9] and white spherically invariant random processes (SIRP's) [10] that the algorithm convergence speed v_{NLMS} is given by

$$v_{\text{NLMS}} = -20\,\log_{10}\left[1 - \frac{\alpha(2-\alpha)}{M}\right]\text{dB} \tag{5}$$

per step. The fastest convergence speed is obtained for $\alpha = 1$ (α is the normalized step size). As a rule of thumb, the speed is about 20 dB per $5M$. It is well known that this speed will decrease for correlated signals. The decrease depends on the strength of the correlation, i.e., the largest eigenvalue of $\mathbf{B}$.

II. Underdetermined LS Solution: The AP-Algorithm

In [1] and [14], it is demonstrated that the NLMS algorithm can also be viewed as an underdetermined LS or minimum-norm solution that uses only one observation at every time instant k. The AP algorithm is an extension of this idea. In addition to the instantaneous observation, P past observations are used to calculate the update. If the P past input vectors $\mathbf{u}(k-1) \cdots \mathbf{u}(k-P)$ are collected in the matrix $\mathbf{U}(k)$, which is defined below in (8), the LS problem is

TABLE I
Ozeki Umeda's AP Algorithm

$\mathbf{U}(k)$	$=$	$[\mathbf{u}(k-1), ..., \mathbf{u}(k-P)]$
$\hat{\mathbf{a}}(k)$	$=$	$\left[\mathbf{U}^T(k)\mathbf{U}(k)\right]^{-1}\mathbf{U}^T(k)\mathbf{u}(k)$
$\boldsymbol{\phi}(k)$	$=$	$\mathbf{u}(k) - \mathbf{U}(k)\hat{\mathbf{a}}(k)$
$e(k)$	$=$	$d(k) - \mathbf{u}^T(k)\hat{\mathbf{w}}(k)$
$\hat{\mathbf{w}}(k+1)$	$=$	$\hat{\mathbf{w}}(k) + \dfrac{\boldsymbol{\phi}(k)}{\boldsymbol{\phi}^T(k)\boldsymbol{\phi}(k)}\, e(k)$

$$\varepsilon_d = \min_{\hat{\mathbf{w}}(k+1)} \left\| \begin{bmatrix} \mathbf{u}^T(k) \\ \mathbf{U}^T(k) \end{bmatrix} \hat{\mathbf{w}}(k+1) - \mathbf{d}(k) \right\|_2^2. \tag{6}$$

$\mathbf{d}(k)$ is a $P+1$ vector of desired values from k to $k-P$. Note that the minimum in (6) is always zero, as long as $P \le M$, where M is the dimension of the vector $\mathbf{d}(k)$. Thus, a manifold of solutions $\hat{\mathbf{w}}(k+1)$ is possible. The one solution of interest minimizes the norm between the previous solution and the new one, i.e., the solution that also minimizes

$$\varepsilon_w = \min_{\hat{\mathbf{w}}(k+1)} \|\hat{\mathbf{w}}(k+1) - \hat{\mathbf{w}}(k)\|_2^2. \tag{7}$$

It can be shown that minimizing (7) with (6) as a linear constraint $\varepsilon_w + \lambda\varepsilon_d$ leads to the minimum-norm solution of an underdetermined LS problem (sliding rectangular window) of order M with only $P < M$ observations. The matrix $\mathbf{U}^T(k)\mathbf{U}(k)$ is assumed to be of rank P. A small positive regularization constant δ can be added on the main diagonal [15] to prevent undesired behavior if the rank is lower than P. The AP algorithm is given in Table I.

Assume $\mathbf{u}(k)$ is given by an AR vector process of the form

$$\mathbf{u}(k) = \sum_{i=1}^{P} a_i\mathbf{u}(k-i) + \mathbf{z}(k) = \mathbf{U}(k)\mathbf{a} + \mathbf{z}(k). \tag{8}$$

The matrix $\mathbf{U}(k)$ is a collection of P of these past vectors $\mathbf{U}(k) = [\mathbf{u}(k-1) \cdots \mathbf{u}(k-P)]$, and $\mathbf{z}(k)$ is a vector with samples from the white random process, i.e., $\mathbf{z}^T(k) = [z(k), z(k-1), \cdots, z(k-M+1)]$. An estimate $\hat{\mathbf{a}}(k)$ of the a parameters can be found from the least squares (LS) problem

$$\min_{\hat{\mathbf{a}}(k)} \|\mathbf{u}(k) - \mathbf{U}(k)\hat{\mathbf{a}}(k)\|_2^2. \tag{9}$$

This LS solution can be recognized in the second line of the AP algorithm. Thus, in case the order P of the algorithm is at least of the order of the AR process, the direction vector $\boldsymbol{\phi}(k)$ is statistically white, i.e., $\boldsymbol{\phi}(k) = \mathbf{z}(k)$. $[\mathbf{u}(k), \mathbf{U}(k)]$ can be multiplied from the left, yielding

$$[\mathbf{u}(k), \mathbf{u}(k-1), \mathbf{u}(k-2), \cdots, \mathbf{u}(k-P)]^T\, \frac{\boldsymbol{\phi}(k)}{\boldsymbol{\phi}^T(k)\boldsymbol{\phi}(k)} = (1, 0, \cdots, 0)^T. \tag{10}$$

Equation (10) gives us the two conditions

$$\mathbf{U}^T(k)\boldsymbol{\phi}(k) = \mathbf{0}, \quad \text{and} \quad \boldsymbol{\phi}^T(k)\mathbf{u}(k) = \boldsymbol{\phi}^T(k)\boldsymbol{\phi}(k). \tag{11a}$$

Thus, if $u(k)$ is an AR process of order P, $\hat{\mathbf{a}}(k)$ is an estimate of its AR coefficients, and $\boldsymbol{\phi}(k) = \hat{\mathbf{z}}(k)$, i.e., $\boldsymbol{\phi}(k)$ is a vector whose elements are estimates of a white random process. If $\boldsymbol{\phi}(k)$, $\mathbf{u}(k)$ and $\mathbf{U}(k)$ are multiplied in the state space update equation, the properties

$$\boldsymbol{\phi}^T(k)\boldsymbol{\epsilon}(k+1) = \boldsymbol{\phi}(k)^T\boldsymbol{\epsilon}(k) - \mathbf{u}^T(k)\boldsymbol{\epsilon}(k) - v(k) \quad (12a)$$

$$\mathbf{u}^T(k)\boldsymbol{\epsilon}(k+1) = -v(k) \quad (12b)$$

$$\mathbf{U}^T(k)\boldsymbol{\epsilon}(k+1) = \mathbf{U}^T(k)\boldsymbol{\epsilon}(k). \quad (12c)$$

are obtained. Note that (12b) is recursively used in (12c) as

$$\mathbf{U}^T(k)\boldsymbol{\epsilon}(k+1) = -[v(k-1),\, v(k-2),\, \cdots,\, v(k-P)]^T. \quad (12d)$$

If the update equation is rewritten, $\mathbf{u}(k)$ can be substituted for in terms of $\boldsymbol{\phi}(k)$ and $\mathbf{U}(k)$, finally obtaining

$$\boldsymbol{\epsilon}(k+1) = \boldsymbol{\epsilon}(k) - \frac{\boldsymbol{\phi}(k)\boldsymbol{\phi}^T(k)}{\boldsymbol{\phi}^T(k)\boldsymbol{\phi}(k)}\boldsymbol{\epsilon}(k) - \frac{\boldsymbol{\phi}(k)}{\boldsymbol{\phi}^T(k)\boldsymbol{\phi}(k)}v_a(k) \quad (13)$$

with the new filtered noise sequence

$$v_a(k) = v(k) - \sum_{i=1}^{P}\hat{\mathbf{a}}_i(k)v(k-i) \quad (14)$$

where $\hat{a}_i(k)$ is the elements of the vector $\hat{\mathbf{a}}(k)$. This leads to the following conclusions.

- The algorithm runs with a decorrelated direction vector $\boldsymbol{\phi}(k)$ rather than the original correlated $\mathbf{u}(k)$, thus increasing the convergence rate.
- The noise sequence $v(k)$ is filtered through a filter with coefficients $\hat{\mathbf{a}}(k)$ close to the optimal solution $\hat{\mathbf{a}}$. If the noise is white, the new noise sequence $v_a(k)$ has a variance that is larger by $1 + \mathbf{a}^T\mathbf{a}$.
- For $\alpha \neq 1$, the algorithm is of filtered-error type (see [13]).

III. Decorrelating MA and ARMA Processes

A. Orthogonal Direction Algorithm

Similar solutions as in the previous section for AR processes can be found for MA and ARMA vector process. For example, an MA(Q) vector process can be described as

$$\mathbf{u}(k) = \sum_{i=1}^{Q} b_i\mathbf{z}(k-i) + \mathbf{z}(k) = \mathbf{Z}(k)\mathbf{b} + \mathbf{z}(k). \quad (15a)$$

$\mathbf{Z}(k)$ is a matrix composed of the past Q white random vectors $\mathbf{z}(k-1)\cdots\mathbf{z}(k-Q)$. Assuming the past values in $\mathbf{Z}(k)$ are known, the parameter set $\mathbf{b}$ can be computed by the LS method above, and $\mathbf{u}(k)$ can be decorrelated. Since $\mathbf{Z}(k)$ is not known exactly, it can be estimated by, say, $\hat{\mathbf{Z}}(k)$

$$\hat{\mathbf{z}}(k) = \mathbf{u}(k) - \hat{\mathbf{Z}}(k)\hat{\mathbf{b}}(k). \quad (15b)$$

$\hat{\mathbf{Z}}(k)$ is a collection of the Q past estimates $\hat{\mathbf{z}}(k-1)\cdots\hat{\mathbf{z}}(k-Q)$. Thus, the unknown parameters $\hat{\mathbf{b}}(k)$ can be calculated recursively by alternating step (15b) and (15c)

$$\hat{\mathbf{b}}(k) = [\hat{\mathbf{Z}}^T(k)\hat{\mathbf{Z}}(k)]^{-1}\hat{\mathbf{Z}}^T(k)\mathbf{u}(k). \quad (15c)$$

Assume that the update direction is chosen as

$$\boldsymbol{\phi}(k) = \hat{\mathbf{z}}(k) = \mathbf{u}(k) - \hat{\mathbf{Z}}(k)\hat{\mathbf{b}}(k) \quad (16a)$$

where $\hat{\mathbf{Z}}(k) = [\boldsymbol{\phi}(k-1),\, \cdots,\, \boldsymbol{\phi}(k-Q)]$. The $(M \times Q)$ matrix $\hat{\mathbf{Z}}(k)$ has the orthogonalizing property

$$\boldsymbol{\phi}^T(k)\hat{\mathbf{Z}}(k) = \mathbf{0}. \quad (16b)$$

As in (11a), multiplying (16a) with $\boldsymbol{\phi}^T(k)$ from the right leads to $\boldsymbol{\phi}^T(k)\mathbf{u}(k) = \boldsymbol{\phi}^T(k)\boldsymbol{\phi}(k)$. Since the update direction at time instant k is orthogonal to the past Q update directions, the algorithm will be denoted the *orthogonal direction (OD)* algorithm (see Table II).

TABLE II
OD Algorithm

$\hat{\mathbf{Z}}(k)$	$=$	$[\boldsymbol{\phi}(k-1),\ldots,\boldsymbol{\phi}(k-Q)],\ \hat{\mathbf{Z}}(0) = [\mathbf{0},\ldots,\mathbf{0}]$
$\hat{\mathbf{b}}(k)$	$=$	$\left[\hat{\mathbf{Z}}^T(k)\hat{\mathbf{Z}}(k)\right]^{-1}\hat{\mathbf{Z}}^T(k)\mathbf{u}(k)$
$\boldsymbol{\phi}(k)$	$=$	$\mathbf{u}(k) - \hat{\mathbf{Z}}(k)\hat{\mathbf{b}}(k)$
$e(k)$	$=$	$d(k) - \mathbf{u}^T(k)\hat{\mathbf{w}}(k)$
$\hat{\mathbf{w}}(k+1)$	$=$	$\hat{\mathbf{w}}(k) + \dfrac{\boldsymbol{\phi}(k)}{\boldsymbol{\phi}^T(k)\boldsymbol{\phi}(k)}e(k)$

The OD algorithm has two practical advantages as compared with the AP algorithm.

1) First, the matrix inverse, which is used for calculating the decorrelation coefficients, does not have to be calculated explicitly. Since the directions $\boldsymbol{\phi}(k-1),\, \cdots,\, \boldsymbol{\phi}(k-Q)$ are orthogonal, the inverse can be given directly in form of a diagonal matrix.

$$[\hat{\mathbf{Z}}^T(k)\hat{\mathbf{Z}}(k)]^{-1} = \mathrm{diag}[\|\boldsymbol{\phi}(k-1)\|_2^{-2},\, \cdots,\, \|\boldsymbol{\phi}(k-Q)\|_2^{-2}].$$

Therefore, if $\boldsymbol{\phi}(k)$ does not tend to zero, the inverse always exists. Thus, no problems occur from nonpersistent excitation, as in the AP algorithm. To ensure that $\|\boldsymbol{\phi}(k)\|_2^2$ is nonzero, a small positive constant can be added to the diagonal elements of the inverse matrix.

2) The orthogonal directions allow the appliction of a step-size $\alpha < 1$ without losing its fundamental orthogonal properties.

To derive the decorrelating properties of the algorithm, multiply the update equation by $\boldsymbol{\phi}(k)$, $\mathbf{u}(k)$ and $\hat{\mathbf{Z}}(k)$ and obtain, respectively

$$\boldsymbol{\phi}^T(k)\boldsymbol{\epsilon}(k+1) = \boldsymbol{\phi}^T(k)\boldsymbol{\epsilon}(k) - \mathbf{u}^T(k)\boldsymbol{\epsilon}(k) - v(k) \quad (17a)$$

$$\mathbf{u}^T(k)\boldsymbol{\epsilon}(k+1) = -v(k) \quad (17b)$$

$$\hat{\mathbf{Z}}^T(k)\boldsymbol{\epsilon}(k+1) = \hat{\mathbf{Z}}^T(k)\boldsymbol{\epsilon}(k). \quad (17c)$$

As in the AP algorithm, the past samples of $v(k)$ do not occur directly in the update equation but only in the filtered values. The general noise term for $Q \geq 1$ can be written as

$$v_b(k) = v(k) - \sum_{i=1}^{Q}\hat{b}_i(k)v_b(k-i). \quad (18)$$

The general description can be derived in vector form. If $\boldsymbol{\epsilon}(k+1)$ is multiplied by $\hat{\mathbf{Z}}(k+1)$ and the properties (17a)–(17c) are used, we have (19a), shown at the bottom of the page. Let $\hat{\mathbf{Z}}^T(k)\boldsymbol{\epsilon}(k) \triangleq -\mathbf{v}_b(k-1)$. Then, the vector equation (19b), shown at the bottom of the page, is obtained with the $(Q \times 1)$ column

$$\hat{\mathbf{Z}}^T(k+1)\boldsymbol{\epsilon}(k+1) = -\begin{bmatrix} v(k) + \hat{\mathbf{b}}^T(k)\hat{\mathbf{Z}}^T(k)\boldsymbol{\epsilon}(k) \\ v(k-1) + \hat{\mathbf{b}}^T(k-1)\hat{\mathbf{Z}}^T(k-1)\boldsymbol{\epsilon}(k-1) \\ \vdots \\ v(k-Q+1) + \hat{\mathbf{b}}^T(k-Q+1)\hat{\mathbf{Z}}^T(k-Q+1)\boldsymbol{\epsilon}(k-Q+1) \end{bmatrix}. \quad (19a)$$

$$\mathbf{v}_b^T(k) = \mathbf{v}^T(k) - [\hat{\mathbf{b}}^T(k)\mathbf{v}_b(k-1),\, \hat{\mathbf{b}}^T(k-1)\mathbf{v}_b(k-2),\, \cdots,\, \hat{\mathbf{b}}^T(k-Q+1)\mathbf{v}_b(k-Q)]. \quad (19b)$$

TABLE III
ARMA DECORRELATING ALGORITHM

$$\begin{aligned}
\mathbf{U}(k) &= [\mathbf{u}(k-1), ..., \mathbf{u}(k-P)] \\
\hat{\mathbf{Z}}(k) &= [\boldsymbol{\phi}(k-1), ..., \boldsymbol{\phi}(k-Q)],\ \ \hat{\mathbf{Z}}(0) = [\mathbf{0}, ..., \mathbf{0}] \\
\begin{pmatrix} \hat{\mathbf{a}}(k) \\ \hat{\mathbf{b}}(k) \end{pmatrix} &= \left[\left(\mathbf{U}(k)\,\hat{\mathbf{Z}}(k)\right)^T \left(\mathbf{U}(k)\,\hat{\mathbf{Z}}(k)\right)\right]^{-1} \left(\mathbf{U}(k)\,\hat{\mathbf{Z}}(k)\right)^T \mathbf{u}(k) \\
\dot{\boldsymbol{\phi}}(k) &= \mathbf{u}(k) - \mathbf{U}(k)\hat{\mathbf{a}}(k) - \hat{\mathbf{Z}}(k)\hat{\mathbf{b}}(k) \\
e(k) &= d(k) - \mathbf{u}^T(k)\hat{\mathbf{w}}(k) \\
\hat{\mathbf{w}}(k+1) &= \hat{\mathbf{w}}(k) + \frac{\boldsymbol{\phi}(k)}{\boldsymbol{\phi}^T(k)\boldsymbol{\phi}(k)}\, e(k)
\end{aligned}$$

vectors $\mathbf{v}_b(k)$ and $\mathbf{v}(k)$. The first line of (19b) equals (18). Now, the update equations can be written as

$$\boldsymbol{\epsilon}(k+1) = \boldsymbol{\epsilon}(k) - \frac{\boldsymbol{\phi}(k)\boldsymbol{\phi}^T(k)}{\boldsymbol{\phi}^T(k)\boldsymbol{\phi}(k)}\,\boldsymbol{\epsilon}(k) - \frac{\boldsymbol{\phi}(k)}{\boldsymbol{\phi}^T(k)\boldsymbol{\phi}(k)}\, v_b(k). \tag{20}$$

Note that if the decorrelation filter length Q exceeds the weight-vector length M, the algorithm can be interpreted as an orthogonalized least squares algorithm. This case is simply a Gram–Smith procedure and has been successfully applied in neural networks [16].

In conclusion, the OD algorithm, which is analogous to the AP algorithm, is a system identification procedure that decorrelates the input sequence. If the estimates $\hat{\mathbf{b}}(k)$ are close to the minimum-phase solution $\mathbf{b}$, a white noise input power is increased by

$$\frac{1}{2\pi}\int_{-\pi}^{\pi} \left| \frac{1}{1+\sum_{i=1}^{Q} b_i\, e^{-j\Omega\, i}} \right|^2 d\Omega.$$

Note also that the noise filter now is of the all-pole type. Thus, only the minimum phase solution of the filter $1 - B(q^{-1})$ leads to a stable filter for the noise sequence. Simulations show that the algorithm always takes the stable solution no matter what $\mathbf{b}$ is chosen. However, the possibility of achieving an instable filter solution remains an open question.

B. An ARMA Process Decorrelating Algorithm

Thus far, the principle of decorrelating AR and MA processes has been demonstrated for an adaptive algorithm. It is relatively straightforward to develop an algorithm that decorrelates an ARMA(P, Q) random process. For this, assume the input sequence is of the form

$$u(k) = \sum_{i=1}^{P} a_i\, u(k-i) + \sum_{i=1}^{Q} b_i\, z(k-i) + z(k)$$

or, in equivalent vector form

$$\mathbf{u}(k) = \mathbf{U}(k)\mathbf{a} + \mathbf{Z}(k)\mathbf{b} + \mathbf{z}(k).$$

An estimate for the white random vector $\mathbf{z}(k)$ is then given by

$$\hat{\mathbf{z}}(k) = \mathbf{u}(k) - \mathbf{U}(k)\hat{\mathbf{a}}(k) - \hat{\mathbf{Z}}(k)\hat{\mathbf{b}}(k). \tag{21a}$$

$\hat{\mathbf{a}}(k)$ and $\hat{\mathbf{b}}(k)$ are obtained from the LS solution

$$\begin{bmatrix} \hat{\mathbf{a}}(k) \\ \hat{\mathbf{b}}(k) \end{bmatrix} = \{[\mathbf{U}(k)\,\hat{\mathbf{Z}}(k)]^T [\mathbf{U}(k)\,\hat{\mathbf{Z}}(k)]\}^{-1} [\mathbf{U}(k)\,\hat{\mathbf{Z}}(k)]^T \mathbf{u}(k). \tag{21b}$$

Again, using the estimated white vector process $\hat{\mathbf{z}}(k)$ as direction vector $\boldsymbol{\phi}(k) = \hat{\mathbf{z}}(k)$, the algorithm is listed in Table III.

The ARMA process can be estimated by the same method. A compound matrix is obtained, consisting of observed input vectors $\mathbf{u}(k)$ and estimated decorrelated vectors $\hat{\mathbf{z}}(k)$. This procedure is called the least-squares modified Yule–Walker method [11], [12]. The initial choice of $\hat{\mathbf{z}}(-1) \cdots \hat{\mathbf{z}}(k-Q)$ will be discussed in the following paragraph. Assuming the estimates $\hat{\mathbf{a}}(k)$ and $\hat{\mathbf{b}}(k)$ are close to the true solution $\mathbf{a}$ and $\mathbf{b}$, respectively, the noise sequence is filtered by a transfer function given by

$$F(q^{-1}) = \frac{1 - \sum_{i=1}^{P} a_i q^{-i}}{1 + \sum_{i=1}^{Q} b_i q^{-i}}. \tag{22}$$

C. Initialization

Thus far, all start-up values were assumed set to zero at time $k = 0$. However, the ARMA process initialization causes some difficulties. These difficulties are demonstrated by an example. Assume that $P = Q = 1$. Set the first iteration to

$$\boldsymbol{\phi}(0) = \mathbf{u}(0).$$

since $\mathbf{u}(-1) = \mathbf{0}$ as well as $\boldsymbol{\phi}(-1) = \mathbf{0}$. A problem occurs at time instant $k = 1$

$$\begin{aligned}
\boldsymbol{\phi}(1) &= \mathbf{u}(1) - \hat{a}(1)\mathbf{u}(0) - \hat{b}(1)\boldsymbol{\phi}(0) \\
&= \mathbf{u}(1) - \hat{a}(1)\mathbf{u}(0) - \hat{b}(1)\mathbf{u}(0).
\end{aligned}$$

There does not exist a unique solution for choosing $\hat{a}(1)$ and $\hat{b}(1)$. For example, choose $\hat{b}(1) = 0$, obtaining

$$\hat{a}(1) = \frac{\mathbf{u}^T(1)\mathbf{u}(0)}{\mathbf{u}^T(0)\mathbf{u}(0)}.$$

In the next step

$$\begin{aligned}
\boldsymbol{\phi}(2) &= \mathbf{u}(2) - \mathbf{a}(2)\mathbf{u}(1) - \hat{b}(2)\boldsymbol{\phi}(1) \\
&= \mathbf{u}(2) - \hat{a}(2)\mathbf{u}(1) - \hat{b}(2)[\mathbf{u}(1) - \hat{a}(1)\mathbf{u}(0)]
\end{aligned}$$

and a unique solution for $\hat{a}(2), \hat{b}(2)$ generally exists. Thus, the initialization problem can be solved by setting $\hat{\mathbf{b}}(k) = \mathbf{0}$ for the first P steps until the $\hat{\mathbf{a}}(k)$ coefficients are initialized. Subsequently, at each step, add one of the Q coefficients of $\hat{\mathbf{b}}(k)$ to the initialization process until at time $k = Q + P$, all of the coefficients are finally in use.

IV. CONCLUSION

A new family of adaptive filters with desirable decorrelation properties has been proposed. Due to their equivalence to symmetrical-form algorithms, their convergence properties can be computed similar to those of the LMS algorithm. The improvement in convergence speed comes along with the possibility of increasing the additive noise variance. Simulations validated the correctness of the properties. Due to the limitations of a correspondence, they will be reported elsewhere.

REFERENCES

[1] S. Haykin, *Adaptive Filter Theory*, 3rd ed. Englewood Cliffs, NJ: Prentice-Hall, 1996.

[2] A. H. Sayed and M. Rupp, "Error energy bounds for adaptive gradient algorithms," *IEEE Trans. Signal Processing*, vol. 44, pp. 1982–1989, Aug. 1996.

[3] K. Ozeki and T. Umeda, "An adaptive filtering algorithm using orthogonal projection to an affine subspace and its properties," *Electron. Commun. Jpn.*, vol. 67-A, no. 5, pp. 19–27, 1984.

[4] S. L. Gay, "A fast converging, low complexity adaptive filtering algorithm," in *Proc. IEEE Acoust., Speech, Signal Process. Workshop Appl. Signal Process. Audio Acoust.*, New Paltz, NY, Oct. 1993.

[5] S. L. Gay and S. Tavathia, "The fast affine projection algorithm," in *Proc. ICASSP*, May 1995, pp. 3023–3026.

[6] D. Slock, "The block underdetermined covariance (BUC) fast transversal filter (FTF) algorithm for adaptive filtering," in *Proc. 26th. Asilomar Conf.*, Pacific Grove, CA, 1992, pp. 550–554.

[7] ——, "Underdetermined growing and sliding covariance fast transversal filter RLS algorithms," in *Proc. EUSIPCO*, Brussels, Belgium, 1992, pp. 1169–1172.

[8] B. Baykal, D. Tarikulo, and A. Constantinides, "Asymptotic analysis of the underdetermined RLS algorithm," in *Proc. EUSIPCO*, Trieste, Italy, Sept. 1996, pp. 1397–1400.

[9] N. J. Bershad, "Analysis of the normalized LMS algorithm with Gaussian inputs," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-34, pp. 793–806, Aug. 1986.

[10] M. Rupp, "The behavior of LMS and NLMS algorithms in the presence of spherically invariant processes," *IEEE Trans. Signal Processing*, vol. 41, pp. 1149–1160, Mar. 1993.

[11] J. A. Cadzow, "ARMA spectral estimation: An efficient closed-form procedure," in *Proc. RADC Spectrum Estimation Workshop*, Rome, NY, Oct. 1979, pp. 81–97.

[12] J. G. Proakis and D. G. Manolakis, *Introduction to Digital Signal Processing.* New York: Macmillan, 1988.

[13] M. Rupp and A. H. Sayed, "A time-domain feedback analysis of filtered-error adaptive gradient algorithms," *IEEE Trans. Signal Processing* vol. 44, pp. 1428–1440, June 1996.

[14] G. C. Goodwin and K. S. Sin, *Adaptive Filtering, Prediction and Control.* Englewood Cliffs, NJ: Prentice-Hall, 1984.

[15] S. G. Kratzer and D. R. Morgan, "The partial rank algorithm for adaptive beamforming," in *Proc. SPIE Real-Time Signal Processing VIII*, pp. 9–14, 1985

[16] S. Chen, S. McLaughlin, and B. Mulgrew, "Complex-valued radial basis function network, Part I: Network architecture and learning algorithms," *Signal Process.*, vol. 35, no. 1, pp. 19–31, Jan. 1994.

A Pipelined LMS Adaptive FIR Filter Architecture Without Adaptation Delay

Scott C. Douglas, Quanhong Zhu, and Kent F. Smith

Abstract— **Past methods for mapping the least-mean-square (LMS) adaptive finite-impulse-response (FIR) filter onto parallel and pipelined architectures either introduce delays in the coefficient updates or have excessive hardware requirements. In this correspondence, we describe a hardware-efficient pipelined architecture for the LMS adaptive FIR filter that produces the same output and error signals as would be produced by the standard LMS adaptive filter architecture without adaptation delays. Unlike existing architectures for delayless LMS adaptation, the new architecture's throughput is independent of the filter length.**

I. Introduction

Advancements in VLSI technology have spurred efforts to map complex algorithms onto regular architectures with computations that can be parallelized and/or pipelined [1]. Although it is relatively straightforward to synthesize high-speed architectures for feedforward-only signal processing structures such as finite impulse response (FIR) filters, it is considerably more difficult to synthesize similar architectures for feedback structures [2], [3]. A case in point is the least mean square (LMS) adaptive FIR filter [4], in which the error of the adaptive filter is used to adjust the filter coefficients in real time. The need for a high-speed architecture for this ubiquitous signal processing system has long been recognized as it is used in numerous high-data-rate systems in communications [5]–[7].

Without exception, previously proposed methods to parallelize and/or pipeline the LMS adaptive FIR filter introduce either delays in the coefficient updates [3], [8]–[12] or a large hardware overhead [2]. In [8]–[11], each coefficient of the system receives an equal delay in the coefficient update, resulting in the delayed LMS adaptive filter. The delayed LMS algorithm is generalized in [3] to allow different coefficient delays for both the filter output computation and the calculation of the coefficient updates. Alternate algorithms employing a transposed form of the FIR filter structure are proposed and described in [12] in which the update delay is different for each LMS adaptive filter coefficient. Analyses of these delayed adaptation algorithms indicate that the performance of these systems is inferior to that of the LMS adaptive filter in some cases, and the performance loss is particularly severe as the amount of delay is increased [3], [8]–[10], [12]. This loss in performance is problematic for adaptive FIR filters with hundreds of coefficients as the update delay usually increases in proportion to the number of filter coefficients.

A second difficulty in employing LMS adaptive filters with delayed updates is in choosing the step size to obtain fast and accurate adaptation behavior. The best step size choice for these algorithms is a complicated function of the input statistics and the delays within the adaptation loop [9], [10]. Step-size normalization for the delayed LMS adaptive filter has been proposed [13]; however, the performance of the resulting system still depends on the input signal correlation statistics and the adaptation delays. Clearly, it would be desirable to obtain a low-complexity pipelined architecture whose coefficient updates contain no adaptation delay so that normalized LMS adaptation or other variable step-size strategies can be reliably employed.

Recently, a method has been introduced for correcting the error produced by the delayed LMS adaptive filter so that standard LMS or normalized LMS adaptation can be employed [13], [14]. In this method, a correction term is computed using products of past corrected errors and certain input signal correlation terms that can be computed recursively. Extensions of the method to allow delayless filtered-X LMS adaptation for adaptive control applications are presented in [15]. It is not clear, however, how the computation of the correction term in these methods can be paralleled and/or pipelined. No high-speed architecture employing these methods has been presented.

In this paper, we describe a pipelined implementation of the LMS adaptive FIR filter. To develop this implementation, we first describe a transpose-form version of the delayed LMS adaptive filter. Then, we describe a transpose-form architecture that computes the correction term used in [14] to calculate the delayless LMS error signal from the delayed LMS error signal. Combining these two structures produces the new LMS adaptive filter architecture. This system can be implemented using regularly connected processing modules, making it amenable to VLSI implementation, and the overall complexity is linear in the number of filter coefficients if μ is chosen to be a power of two. The new architecture's throughput is always greater than that of existing delayless LMS adaptive filter architectures. While its output latency is equal to the FIR filter length, its throughput does not depend on the filter length to first order. Simulations verify the computational equivalence of the new architecture to that of the standard LMS adaptive filter.

Manuscript received September 17, 1996; revised April 25, 1997. This work was supported in part by the Federal Bureau of Investigation under Contract JFBI95017 and by the National Science Foundation under Grant MIP-9501680. The associate editor coordinating the review of this paper and approving it for publication was Dr. Sergios Theodoridis.

The authors are with the Department of Electrical Engineering, University of Utah, Salt Lake City, UT 84112 USA.

Publisher Item Identifier S 1053-587X(98)02206-5.