INFORMATICA, 1998, Vol. 9, No. 2, 161–171

High Speed LMS Adaptive Filtering

Kazys KAZLAUSKAS
Institute of Mathematics and Informatics
Akademijos 4, 2600 Vilnius, Lithuania
e-mail: kazlausk @ktl.mii.lt

Received: April 1998

Abstract. In this paper we show that the least mean square (LMS) algorithm can be speeded up without changing any of its adaptive characteristics. The parallel LMS adaptive filtering algorithm and its modifications are presented. High speed is achieved by increasing the parallelism in the LMS adaptive algorithm through a proper modification of the LMS adaptive algorithm. An iterative procedures for efficient computation of the lower triangular inverse matrix and the input signal covariance matrix are presented.

Key words: parallel, least mean square, adaptive filtering.

1. Introduction

The time-domain least mean square adaptive filter algorithm has been used in a variety of applications as system modeling, noise cancelling, channel equalization, adaptive antennas, and adaptive array processing (Haykin, 1986; Joint, 1981; Special, 1987). In the usual implementation the outputs of a tapped delay line are weighted and summed to produce the filter output. The weights of this transversal filter which determine its impulse response, are updated iteratively based upon the difference between the filter output and a desired input, so as to minimize the mean-square error of the difference. A frequency domain implementation of an adaptive filter, requiring less computation than its time domain counterpart, has been proposed in (Dentino, McCool and Widrow, 1978). In some applications good performance can still be obtained with frequency domain techniques based on circular convolution, e.g., in the detection of narrow-band signals in broad-band noise (Ferrara, 1980).

The realization of adaptive filters at high sampling rates is important in many applications, but is made difficult by recursive nature of algorithms. Adaptive filters based on the transversal filter have an inherent sampling rate limitation for a given speed of hardware due to the feedback of the residual error to the adaptation of the individual stages (Meng and Messerschmit, 1987).

Due to the availability of high speed parallel processors and fast Fourier transform algorithms for convolution, block digital filtering has received great attention (Mikhael and Wu, 1987; 1989; Wang and Wang, 1993). The block processing technique involves the calculation of a block of outputs from a block of inputs. The block least mean square

(BLMS) algorithm in (Clark *et al.*, 1981; 1983) is a gradient algorithm with a fixed convergence factor μ_B. Similar to the LMS algorithm, μ_B controls the convergence speed, accuracy, and stability of the adaptive filter. To obtain a proper convergence factor of the BLMS algorithm, a priori knowledge of the input processes as well as trial and error are necessary. In the case of a correlated input, the BLMS algorithm has a smaller convergence domain than the LMS algorithm, resulting in a slower overall convergence to ensure stability. This is the reason for the well-known fact that the adaptation step must be devided by block length M in a BLMS algorithm.

In this paper, we show that this drawback can be overcome, and that the availability of block processing techniques allows the derivation of computationally efficient parallel algorithms that behave exactly the same as their scalar version. These algorithms are just a rearrangement of the initial LMS equations. We propose a realization of the conventional LMS adaptive filter in the time domain which does converge to the optimum transversal filter solution. The proposed method is intended to be a direct replacement for the LMS adaptive filter, but requires less computation than the time domain realization. The proposed algorithm converges at the same rate as the time domain LMS filter. Convergence at this rate is not always advantageous, e.g., when the correlation matrix of the signals at the filter taps has highly disparate eigenvalues. Nevertheless, the LMS adaptive filter has widely used and is well understood, so that a fast implementation should be of interest.

2. The LMS Adaptive Filter

The LMS adaptive filter (Widrow and Stearns, 1985) is an finite impulse response (FIR) digital filter of order $L-1$ for which the output $\widehat{y}(k)$ at discrete time instant k is given as the convolution sum of the input $x(k)$ and the filter weights $w_i(k)$:

$$\widehat{y}(k) = \sum_{i=0}^{L-1} x(k-i)w_i(k) = X^T(k)W(k) = W^T(k)X(k), \quad k = 0, 1, \ldots \tag{1}$$

The LMS algorithm adjusts the filter weights in accordance with (2):

$$W(k+1) = W(k) + \mu e(k)X(k), \tag{2}$$

where μ is a parameter that controls rate and stability of convergence, and $W(k)$ and $X(k)$ are, respectively, the $L \times 1$ weight vector and the $L \times 1$ input vector:

$$\begin{aligned} W(k) &= [w_0(k), w_1(k), \ldots, w_{L-1}(k)]^T, \\ X(k) &= [x(k), x(k-1), \ldots, x(k-L+1)]^T, \end{aligned} \tag{3}$$

and $e(k)$ is the error at the kth instant given by the difference between the desired output $y(k)$ and the actual output $\widehat{y}(k)$:

$$e(k) = y(k) - X^T(k)W(k). \tag{4}$$

3. The Block LMS Adaptive Filter

Letting $m = 0, 1, \ldots$ denote the block number, and $M = 1, 2, \ldots$ the block length, there are two different vectors of interest for the block adaptive filter. The first is the $L \times 1$ vector $X(k)$ of input stored in the filter's register at time k from (3). The other is the mth input data block designed by the $M \times 1$ vector

$$\bar{x}(m) = \left[x(mM), x(mM+1), \ldots, x(mM+M-1)\right]^T. \tag{5}$$

Similarly, the mth output data block is given by the $M \times 1$ vector $\bar{\hat{y}}(m) = \left[\hat{y}(mM), \hat{y}(mM+1), \ldots, \hat{y}(mM+M-1)\right]^T$, and desired mth output data block is given by the $M \times 1$ vector $\bar{y}(m) = \left[y(mM), \ldots, y(mM+M-1)\right]^T$. Since the filter weights are adjusted on a block by block basis, the weights for the mth block are denoted by the following element vector

$$W(m) = \left[w_0(m), w_1(m), \ldots, w_{L-1}(m)\right]^T, \tag{6}$$

where the subscripts denote the individual filter weights.

The block adaptive filter output is given by the equation

$$\bar{\hat{y}}(m) = \bar{X}(m)W(m), \tag{7}$$

where $\bar{X}(m)$ is an $M \times L$ matrix given by

$$\bar{X}(m) = \begin{bmatrix} x(mM) & x(mM-1) & \ldots & x(mM-L+1) \\ x(mM+1) & x(mM) & \ldots & \cdot \\ \cdot & \cdot & \cdot & \cdot \\ x(mM+M-1) & \ldots & \ldots & x(mM+M-L) \end{bmatrix}. \tag{8}$$

Substituting (6) and (8) into (7) yields

$$\hat{y}(mM+l) = X(mM+l)W(m), \quad l = 0, 1, \ldots, M-1, \tag{9}$$

where $X(mM+l)$ is lth row of the matrix $\bar{X}(m)$.

Equation (9) gives the elements of $\bar{\hat{y}}(m)$, the mth block of the output data, as a linear convolution, which can also be written as

$$\hat{y}(mM+l) = \sum_{i=0}^{L-1} x(mM+l-i)w_i(m), \quad l = 0, 1, \ldots, M-1. \tag{10}$$

The filter weights are updated as follows (Clark *et al.*,1983):

$$W(m+1) = W(m) + \frac{2\mu_\beta}{M}\bar{X}^T(m)\bar{e}(m), \tag{11}$$

where

$$\bar{X}^T(m)\bar{e}(m) = \left[f_0(m), f_1(m), \ldots, f_{L-1}(m)\right]^T, \tag{12}$$

in which $\bar{e}(m)$ is an M element vector of errors given by

$$\bar{e}(m) = \left[e(mM), \ldots, e(mM + M - 1)\right]^T, \tag{13}$$

with $e(k) = y(k) - \widehat{y}(k)$ where $y(k)$ is desired response sample. In block form $\bar{e}(m) = \bar{y}(m) - \bar{\widehat{y}}(m)$. Substituting (8) and (13) into (12) yields the following expression for the ith element of the block mean-square error gradient estimate vector:

$$f_i(m) = X^T(mM - i)\bar{e}(m), \quad i = 0, 1, \ldots, L - 1. \tag{14}$$

Equation (14) represents a correlation between the mth error block and the $(mM - i)$th input data block, as can be seen by writting it as a summation

$$\begin{aligned} f_i(m) &= \sum_{i=mM}^{mM+M-1} x(k-i)e(k) \\ &= \sum_{n=mM-i}^{mM+M-i-1} e(n+i)x(n), \quad i = 0, 1, \ldots, L-1. \end{aligned}$$

It is apparent that the block adaptive algorithm consists of successive operations of block convolution (9), gradient estimation (12), and weight update (11). Note that the most efficient choice of block length is $M = L$.

4. Parallel LMS Adaptive Filter

We propose here an algorithm for realizing adaptive filters that achieve two objectives simultaneously. The first objective is to allow arbitrarily high sampling rates for a given speed hardware, at the expense of additional hardware. The second objective is to not modify the characteristics of the algorithm, and hence not affect the convergence and tracking capabilities.

4.1. *Derivation of the Parallel LMS Adaptive Filter Algorithm*

Consider the LMS equation (2). The solution of the Eq. 2 is (Kazlauskas,1997):

$$W(n) = W(k) + \mu \sum_{j=k}^{n-1} e(j)X(j), \quad k = 0, 1, \ldots, \tag{15}$$

where $W(k)$, $e(k)$, and $X(k)$ are defined in (3) and (4). Substituting $n = mM + M$, and $k = mM$, $m = 0, 1, \ldots$, into (15), we obtain

$$\begin{aligned} W(mM+M) &= W(mM) + \mu \sum_{j=mM}^{mM+M-1} e(j)X(j) \\ &= W(mM) + \mu \sum_{j=1}^{M} e(mM+j-1)X(mM+j-1) \\ &= W(mM) + \mu \bar{X}^T(m)\bar{e}(m), \end{aligned}$$

or

$$\bar{W}(m+1) = \bar{W}(m) + \mu \bar{X}^T(m)\bar{e}(m), \quad m = 0, 1, \ldots, \tag{16}$$

where

$$\bar{W}(m+1) = W(mM+M), \quad \bar{W}(m) = W(mM),$$

$\bar{X}(m)$ is an $M \times L$ matrix as in (8), and $\bar{e}(m) = \big[e(mM), \ldots, e(mM+M-1)\big]^T$.

Vector $\bar{e}(m)$ is formed using values $e(mM), \ldots, e(mM+M-1)$.

Now we show how to compute these values. Substituting (15) into (4), we have

$$e(n) = y(n) - X^T(n)W(k) - \mu X^T(n) \sum_{j=k}^{n-1} e(j)X(j), \quad n = 0, 1, \ldots. \tag{17}$$

Defining $n = mM + i - 1$, and $k = mM$, $m = 0, 1, \ldots$, $i = 1, 2, \ldots, M$, and using them in (17), we get

$$\begin{aligned} e(mM+i-1) &= y(mM+i-1) - X^T(mM+i-1)W(mM) \\ &\quad -\mu X^T(mM+i-1) \sum_{2j=mM}^{mM+i-2} e(j)X(j) \\ &= y(mM+i-1) - X^T(mM+i-1)W(mM) \\ &\quad -\mu X^T(mM+i-1) \sum_{j=1}^{i-1} e(mM+j-1)X(mM+j-1), \end{aligned}$$

or in matrix form:

$$\bar{e}(m) = \bar{y}(m) - \bar{X}(m)\bar{W}(m) - \bar{D}(m)\bar{e}(m). \tag{18}$$

From (18), we obtain

$$\bar{e}(m) = (I_M + \bar{D}(m))^{-1}\big(\bar{y}(m) - \bar{X}(m)\bar{W}(m)\big), \quad m = 0, 1, \ldots, \tag{19}$$

where I_M is the unity matrix,

$$\bar{y}(m) = \left[y(mM), \ldots, y(mM + M - 1)\right]^T,$$
$$\bar{W}(m) = \left[w_0(mM), \ldots, w_{L-1}(mM)\right]^T.$$

$\bar{D}(m)$ is the $M \times M$ matrix defined as

$$\bar{D}(m) = \left[d_{ij}(m)\right], \quad i, j = 1, 2, \ldots, M, \tag{20}$$

in which

$$d_{ij}(m) = 0, \quad \text{if} \quad i \leqslant j,$$
$$d_{ij}(m) = \mu X^T(mM + i - 1)X(mM + j - 1), \quad \text{if} \quad i > j.$$

Parallel LMS adaptive filter algorithm (Egs. 16 and 19) allows us to compute the values of weight vector $\bar{W}(m) = W(mM)$, $m = 0, 1, \ldots$ at the time instants 0, M, $2M$ and so on. In the following section, we shall derive modified parallel adaptive algorithm which allows us to compute M values of $W(k)$ simultaneously.

4.2. *Modified Parallel LMS Adaptive Filter Algorithm*

Defining $n = mM + l$, and $k = mM - M + l$, $m = 0, 1, \ldots$, $l = 1, 2, \ldots, M$, and using them in (15), we obtain

$$\begin{aligned} W(mM + l) &= W(mM - M + l) + \mu \sum_{j=mM-M+l}^{mM+l-1} X(j)e(j) \\ &= W(mM - M + l) + \mu \sum_{j=1}^{M} X(mM - M + l + j - 1) \\ &\quad \times e(mM - M + l + j - 1), \quad l = 1, 2, \ldots, M, \end{aligned}$$

or in matrix form:

$$\bar{W}(m) = \bar{W}(m - 1) + \bar{B}(m)\bar{E}(m), \quad m = 0, 1, \ldots, \tag{21}$$

where the $ML \times 1$ vectors $\bar{W}(m)$ and $\bar{W}(m - 1)$ are defined by

$$\bar{W}(m) = \left[W(mM + 1), \ldots, W(mM + l), \ldots, \ldots, W(mM + M)\right]^T,$$
$$\bar{W}(m - 1) = \left[W((m - 1)M + 1), \ldots, W((m - 1)M + l), \ldots, W(mM)\right]^T,$$

the $ML \times M^2$ matrix $\bar{B}(m)$ is defined by

$$\bar{B}(m) = \text{blockdiag}\{\bar{B}(m, 1), \ldots, \bar{B}(m, l), \ldots, \bar{B}(m, M)\}, \tag{22}$$

in which the $L \times M$ matrix

$$\bar{B}(m,l) = \big[X(mM - M + l), \ldots, X(mM + l - 1)\big],$$

and the $M^2 \times 1$ vector $\bar{E}(m)$ is defined by

$$\begin{aligned}\bar{E}(m) = \big[&e(mM - M + j), \ldots, e(mM - M + l + j - 1), \ldots,\\ &e(mM + j - 1)\big]^T \quad l = 1, 2, \ldots, M,\end{aligned} \tag{23}$$

in which

$$e(mM - M + l + j - 1) = \big[e(mM - M + l), \ldots, e(mM + l - 1)\big]^T,$$
$$j = 1, 2, \ldots, M.$$

Vector $\bar{E}(m)$ in (21) is formed from values $e(mM), \ldots, e(mM + M - 1)$, which we compute using Eq. 19. Modified parallel LMS adaptive algorithm (Eqs. 21 and 23) allows us to compute M weights $W(mM + 1), \ldots, W(mM + M)$ simultaneously. Thus, this algorithm is useful in the case we need to calculate all weights and to ensure high speed adaptive filtering.

4.3. *Modified Parallel LMS Adaptive Filter Algorithm 2*

Defining $n = mM + i$, $k = mM$, $m = 0, 1, \ldots$, $i = 1, 2, \ldots, M$, and using them in (15), we have

$$\begin{aligned}W(mM + i) &= W(mM) + \mu \sum_{j=mM}^{mM+i-1} e(j)X(j)\\ &= W(mM) + \mu \sum_{j=1}^{i} e(mM + j - 1)X(mM + j - 1),\\ &\qquad i = 1, 2, \ldots, M,\end{aligned}$$

or in matrix form:

$$\bar{W}(m) = \bar{W}(mM) + \mu \bar{X}(m)\bar{e}(m), \tag{24}$$

where

$$\begin{aligned}\bar{W}(m) &= \big[W(mM + 1), \ldots, W(mM + M)\big]^T,\\ \bar{W}(mM) &= \big[W(mM), \ldots, W(mM)\big]^T,\\ \bar{e}(m) &= \big[e(mM), \ldots, e(mM + M - 1)\big]^T,\end{aligned}$$

and the $LM \times M$ matrix $\bar{X}(m)$ is defined by

$$\bar{X}(m) = [X_{ij}], \quad i,j = 1,2,\ldots,M, \tag{25}$$

in which

$$X_{ij} = 0, \quad \text{if} \quad i < j,$$
$$X_{ij} = X(mM + j - 1), \quad \text{if} \quad i \geqslant j.$$

Vector $\bar{e}(m) = \left[e(mM),\ldots,e(mM+M-1)\right]^T$ is computed using Eq. 19. Modified parallel LMS adaptive Algorithm 2 calculates all weights in every time instant, and enables us to compute M weights values simultaneously. Thus, it is applicable in the parallel adaptive filtering schemes.

5. Algorithm Realization Considerations

To realize the parallel LMS adaptive algorithms, we need to calculate inverse matrix $\bar{T}^{-1}(m) = (I_M + \bar{D}(m))^{-1}$(see Eq. 19).

Define $\bar{g}(m) = \bar{y}(m) - \bar{X}(m)\bar{W}(m) = \left[g_1(m),\ldots g_M(m)\right]^T$. Then (19) in matrix form:

$$\begin{bmatrix} 1 & & & \\ d_{21} & 1 & 0 & \\ d_{31} & d_{32} & 1 & \\ \cdot & & \cdot & \\ d_{M,1} & \cdot & \cdot\, d_{M,M-1} & 1 \end{bmatrix} \begin{bmatrix} e_1(m) \\ \cdot \\ \cdot \\ \cdot \\ e_M(m) \end{bmatrix} = \begin{bmatrix} g_1(m) \\ \cdot \\ \cdot \\ \cdot \\ g_M(m) \end{bmatrix}, \tag{26}$$

where $\bar{e}(m) = \left[e_1(m),\ldots,e_i(m)\ldots,e_M(m)\right]^T$, in which $e_i(m) = e(mM+i-1)$, $i = 1,2,\ldots,M$ and $d_{ij}(m)$, $i,j = 1,2,\ldots,M$ are as in (20).

From (26), we obtain recurrent equation for calculation $e_i(m)$:

$$e_i(m) = g_i(m) - \sum_{j=1}^{i-1} d_{ij}e_j(m), \quad i = 1,2,\ldots,M. \tag{27}$$

In some practical situations, when we need to calculate all $e_i(m)$, $i = 1,2,\ldots,M$ values simultaneously, the following algorithm for calculation inverse matrix is defined by

PROPOSITION 1. The inverse matrix $\bar{T}^{-1}(m)$ of the $M \times M$ lower triangular matrix $\bar{T}(m) = [d_{ij}(m)]$, $d_{ii}(m) = 1$, $i, j = 1, 2, \ldots, M$ is defined by

$$\bar{T}^{-1}(m) = [t_{ij}(m)] = \begin{cases} 0, & \text{if } i < j, \\ 1, & \text{if } i = j, \\ d_{ij}(m), & \text{if } i = j+1, \\ \det(\bar{D}_{ij}(m) = [d_{kl}(m)], & \\ k = j+1, j+2, \ldots, i, & \\ l = j, j+1, \ldots, i-1), & \text{if } i > j+1. \end{cases} \tag{28}$$

Calculation of the matrix $\bar{D}(m)$.

From (20), it follows for $i > j$, that

$$\begin{aligned} d_{ij}(m) &= \mu X^T(mM+i-1)X(mM+j-1) \\ &= \mu\big[x(mM+i-1)x(mM+j-1) + x(mM+i-2) \\ &\quad \times x(mM+j-2) + \ldots + x(mM+i-L)x(mM+j-L)\big], \end{aligned} \tag{29}$$

and

$$\begin{aligned} d_{i+1,j+1}(m) &= \mu X^T(mM+i)X(mM+j) \\ &= \mu\big[x(mM+i)x(mM+j) + x(mM+i-1)x(mM+j-1) \\ &\quad + \ldots + x(mM+i-L+1)x(mM+j-L+1)\big]. \end{aligned} \tag{30}$$

Thus, using (29) and (30), we obtain

$$\begin{aligned} d_{i+1,j+1}(m) &= d_{ij}(m) + \mu\big[x(mM+i)x(mM+j) \\ &\quad - x(mM+i-L)x(mM+j-L)\big], \end{aligned} \tag{31}$$

for $j = 1, 2, \ldots, M-2$, and $i \geqslant j+1$; $m = 1, 2, \ldots$ Eq. 31 provides the computations to be performed recursively along the subdiagonals of the matrix $\bar{D}(m)$.

Now we derive the recursive expression of the first column $d_{i,1}(mM+M)$ of matrix $\bar{D}(m)$ in terms of the last row $d_{M,M-i+1}(mM)$ of matrix $\bar{D}(m)$ during the previous block. The first column of matrix $\bar{D}(m)$ (20) can be expressed as:

$$\begin{aligned} d_{i,1}(mM+M) &= \mu X^T(mM+M+i-1)X(mM+M) \\ &= \mu\big[x(mM+M+i-1)x(mM+M) \\ &\quad + x(mM+M+i-2)x(mM+M-1) + \ldots \\ &\quad + x(mM+M+i-l-1)x(mM+M-l) + \ldots \\ &\quad + x(mM+M+i-L)x(mM+M-L+1)\big]. \end{aligned} \tag{32}$$

The last row of matrix $\bar{D}(m)$ during the previous block can be expressed as:

$$\begin{aligned} d_{M,M-i+1}(mM) &= \mu X^T(mM+M-1)X(mM+M-i) \\ &= \mu\big[x(mM+M-1)x(mM+M-i) \\ &\quad +x(mM+M-2)x(mM+M-i-1)+\dots \\ &\quad +x(mM+M-l-1)x(mM+M-i-l)+\dots \\ &\quad +x(mM+M-L)x(mM+M-L-i+1)\big]. \end{aligned} \tag{33}$$

Then, it follows from (32) and (33), that

$$\begin{aligned} d_{i,1}(mM+M) &= d_{M,M-i+1}(mM) \\ &\quad +\mu\Bigg[\sum_{l=0}^{i-1} x(mM+M+l)x(mM+M+l-i+1) \\ &\quad -\sum_{l=0}^{i-1} x(mM+M+l-L)x(mM+M+l-L-i+1)\Bigg], \\ &\qquad i=2,3,\dots,M;\ m=0,1,\dots. \end{aligned} \tag{34}$$

So, the matrix is computed as follows:

1. Using the last row $d_{M,M-i+1}(mM)$ of the matrix $\bar{D}(m)$, we compute the first column $d_{i,1}(mM+M)$ of the matrix $\bar{D}(m)$ during the next block (Eq. 34).
2. Using these first column values, we provide the computations recursively along the subdiagonals of the matrix $\bar{D}(m)$ (Eq. 31).

6. Conclusion

We show that the LMS algorithm can be speeded up by the use such block formulation, which allows LMS algorithm to be exactly equivalent to the sample-by-sample LMS. We demonstrate a realization of adaptive filters for which there is no theoretical limit on sampling rate in a given speed of hardware, at the expense of additional hardware and latency. Our realization does not change the adaptive characteristics and hence does not degrade the adaptive filter tracking capability. This technique can be applied to many different algorithms in the LMS family.

References

Clark G.A., S.K. Mitra and S.R. Parker (1981). Block implementation of adaptive digital filters. *IEEE Trans. Circuits Syst.*, **28**, 584–592.

Clark G.A., S.R. Parker and S.K. Mitra (1983). A unified approach to time and frequency domain realization of FIR adaptive digital filters. *IEEE Trans. Acoust., Speech, Signal Processing*, **31**, 1073–1083.

Dentino M., J. McCool and B. Widrow (1978). Adaptive filtering in the frequency domain. *Proc. IEEE*, **66**, 1658–1659.

Ferrara E.R. (1980). Fast implementation of LMS adaptive filters. *IEEE Trans. Acoust., Speech, Signal Processing*, **28**, 474–475.

Haykin S. (1986). *Adaptive Filter Theory. Englewood Cliffs, NJ: Prentice Hall.*

Kazlauskas K.(1997). Pipelined-block models of linear discrete-time systems. *Informatica*, **8**(2), 215–236.

Meng T.H.-Y., and D.G. Messerschmitt (1987). Arbitrarily high sampling rate adaptive filters. *IEEE Trans. on Acoust., Speech, Signal Processing*, **35**, 455–470.

Mikhael W.B., and F.H. Wu (1987). Fast algorithms for block FIR adaptive digital filtering. *IEEE Trans.Circuits Syst.*, **34**, 1152–1160.

Mikhael W.B., and F.H. Wu (1989). A fast block FIR adaptive digital filtering algorithm with individual adaptation of parameters. *IEEE Trans. Circuits Syst.*, **36**, 1–10.

Wang T., and Ch.-L. Wang (1993). On the optimum design of the block adaptive FIR digital filter. *IEEE Trans. on Signal Processing*, **41**, 2131–2140.

Widrow B., and S.D. Stearns (1985). *Adaptive Signal Processing*. Englewood Cliffs, NJ: Prentice Hall.

Joint Special Issue on Adaptive Signal Processing (1981). *IEEE Trans. Accoust., Speech, Signal Processing*, **29**.

Special Issue on Adaptive Systems and Applications (1987). *IEEE Trans. Circuits Syst.*, **34**.

K. Kazlauskas received Ph.D. degree from Kaunas Polytechnic Institute (Kaunas, Lithuania) in 1975. He is a senior researcher of the Process Recognition Department at the Institute of Mathematics and Informatics. His research interests include design of concurrent algorithm and architectures for signal processing, and computer aided design of signal processing systems.

LYGIAGRETUS ADAPTYVUS FILTRAVIMAS

Kazys KAZLAUSKAS

Straipsnyje parodyta, kad mažiausiųjų kvadratų adaptyvūs filtravimo algoritmai gali būti pagreitinti nepakeičiant jų charakteristikų. Nagrinėjamas lygiagretus adaptyvus filtravimo algoritmas ir jo modifikacijos. Šio algoritmo greitį teoriškai galima didinti be galo papildomos procesorinės įrangos ir latentiškumo sąskaita. Aprašyti žemutinės trikampės atvirkštinės matricos ir įėjimo signalo kovariacinės matricos skaičiavimo algoritmai.