Fast Newton Transversal Filters—A New Class of Adaptive Estimation Algorithms

George V. Moustakides and Sergios Theodoridis, *Member, IEEE*

***Abstract*—A new class of adaptive algorithms for the estimation of FIR transversal filters is presented. The main characteristic of this class is the fast computation of the gain vector needed for the adaptation of the transversal filters. The method for deriving these algorithms is based on the assumption that the input signal is autoregressive of order M, where M can be selected to be much smaller than the order of the filter we want to estimate. Under this assumption the covariance matrix of the input signal is estimated by extending in a min-max way the Mth order sample covariance matrix. This estimate can be regarded as a generalization of the diagonal covariance matrix used in LMS and leads to an efficient computation of the gain needed for the adaptation. The new class of algorithms contains the LMS and the fast versions of LS as special cases. The complexity changes linearly with M, starting from the complexity of the LMS (for $M = 0$) and ending to the complexity of the fast versions of LS.**

I. Introduction and Background Material

OPTIMAL estimation of an FIR filter's impulse response is of major importance in many application areas such as automatic control, system identification, channel equalization, interference rejection, echo cancellation, etc. [1], [6], [11], [12]. The task is to estimate the filter's response in such a way so that for a given input signal, its output tracks a desired response signal in an optimal way.

Wiener's approach to the problem is to assume that the input and output signals are jointly stationary with known second-order statistics. The FIR filter is parametrized in its transversal form and optimality is defined as the minimization of the mean-squared error between the actual and the desired output signals. Under this setting the unknown filter parameters satisfy the well-known Wiener equation.

There are very rare cases where the knowledge of the second-order statistics is available. To overcome this lack of information, a simultaneous estimation of the filter coefficients and of the second-order statistics is used. The estimates are based on the available input and output data. In most practical cases, the number of data grows with time, thus the estimates have to change with every new incoming information. Sequential techniques were derived to deal with this situation. These techniques use only the new information in order to correct the estimates of the previous instant. Most sequential algorithms behave the way one hopes they will, that is, when the number of available data tends to infinity, under stationarity and some other weak conditions, their estimates tend to the Wiener solution.

Before presenting the algorithmic class that we will deal with, let us state the estimation problem in a more rigorous way. Suppose we are given sequentially two sequences $\{x(t), y(t)\}$ (input and output). Define $X_N(t) = [x(t) \cdots x(t-N+1)]^T$, we are interested in finding a filter $H_N = [h_0 \cdots h_{N-1}]^T$ such that the sequence $\{w_N(t)\}$ with $w_N(t) = y(t) - H_N^T X_N(t)$ is uncorrelated with $\{X_N(t)\}$ (Wiener problem).

Since from a practical point of view, sequential techniques are more important for solving the problem we just defined, we will concentrate on a very general class of sequential algorithms, a class known as stochastic Newton (SN). To define the class, let $H_N(t)$ denote an estimate of H_N at time instant t, then $H_N(t)$ can be obtained from the estimate of the previous instant $H_N(t-1)$ by the following relations [9, p. 47]:

$$C_N(t) = -\boldsymbol{R}_N^{-1}(t)\, X_N(t) \tag{1a}$$

$$e_N(t) = y(t) - H_N^T(t-1)\, X_N(t) \tag{1b}$$

$$H_N(t) = H_N(t-1) - e_N(t)\, C_N(t) \tag{1c}$$

where $e_N(t)$ is the prior prediction error and $C_N(t)$ the gain vector. Notice that (1b), (1c) are common to all algorithms in this class and are known as the "filtering part" of the algorithm. Notice also that in order to compute this part we have to perform $2N$ (where N is the filter length) operations per time step. This number of operations constitutes a lower bound for the computational complexity of any algorithm in this class. The part that discriminates the algorithms is the definition of the matrix $R_N(t)$ in (1a). This matrix usually estimates the second-order statistics of the input sequence $\{x(t)\}$ and it is obvious that it plays the most important role for the SN class.

The two best known estimation algorithms fall into the SN class. Selecting $\boldsymbol{R}_N(t) = (\sigma_x^2/\delta)\boldsymbol{I}$, with σ_x^2 being an estimate of the input power and δ a small constant known as "step size," results in the LMS algorithm. If $\boldsymbol{R}_N(t)$ is taken as the sample covariance $\Sigma^t X_N(n)\, X_N^T(n)$, the resulting algorithm is the RLS. Both algorithms have been extensively analyzed in the literature theoretically. LMS

Manuscript received March 27, 1989; revised September 6, 1990.

G. V. Moustakides is with the Computer Technology Institute, 26110 Patras, Greece.

S. Theodoridis is with the Department of Computer Engineering, University of Patras, Patra, Rio, Greece, and with the Computer Technology Institute, 26110 Patras, Greece.

IEEE Log Number 9101877.

is characterized by its simplicity, a feature that constitutes this algorithm (and its variants) as a major estimation tool for practical applications nowadays. RLS, on the other hand, has a much more complex form, but from a performance point of view is superior to LMS. RLS is known mostly as the solution to the least squares (LS) problem, a unique feature of this algorithm.

In real-time estimation, two important properties are considered when deciding the usefulness of a sequential algorithm. The first is convergence performance, that is, how fast the estimation tends to the optimum (Wiener solution), and second, the computational complexity expressed in number of operations per time instant. Obviously in practice we would like an algorithm with low complexity and high convergence rate. LMS has complexity equal to $2N$ (the minimum) and very slow convergence, while RLS has high complexity $O(N^2)$ and very fast convergence rate. In order to improve the complexity of RLS, in [8] the very special structure of the LS problem was taken into account and fast LS algorithms were derived that have linear with N complexity. Later faster versions appeared, with the fastest being FAEST and FTF [3], [4], which have complexity equal to $7N$. These algorithms combine the convergence properties of the RLS with linear complexity. Unfortunately, they present stability problems, thus stabilized versions occurred [2], [10], [13] that have complexity from $8N$ to $10N$. For some applications the extra $6N$ to $8N$ operations, with respect to the LMS, we have to pay for the fast LS versions can be prohibitive with today's technology. A notable example is echo cancellation in audioconferencing where N can go up to a few thousands.

RLS and LMS can be regarded as the two extremes in performance and in complexity for the SN class, leaving a notable gap between them. It is exactly this gap we will try to completely cover with the present paper. Specifically we will define a class of algorithms that have complexity $2N + 5M$. By selecting M from 0 to N we can go from the performance and complexity of LMS to the performance and complexity of the fast versions of LS. Since this class has the extra parameter M that we can play with, it might be possible to select an M that will yield an algorithm with low complexity while the performance will remain comparable to the performance of RLS.

The rest of the paper is organized as follows: Section II contains the main result of this paper, namely a method for extending a nonstationary covariance matrix. Based on the results of Section II, in Section III we derive three versions of our class of algorithms. Simulations are presented in Section IV with a discussion on how we can obtain simplified versions of the algorithm. Finally the conclusion is given in Section V.

II. A Min-Max Extension of a Nonstationary Covariance Matrix

This section contains the main result of the paper which we will use in Section III in order to derive our class of algorithms. Before getting into any details let us first discuss some key features of LMS and RLS. As we have seen in Section I, the most important quantity for the SN class is the matrix $\boldsymbol{R}_N(t)$ which estimates the second-order statistics of the input sequence $\{x(t)\}$. For this estimation we have seen that RLS uses the sample covariance matrix. This estimate of the covariance matrix is an estimate one will use in cases where there is absolutely no prior information (or no assumption is made) about the input sequence. If, for example, the input is white noise it is obvious that the sample covariance is a bad estimate. For such a case we expect that with a diagonal matrix we will have a much better estimate. It is exactly this fact that can be used to derive LMS. A way to obtain LMS (from RLS) is to assume that the input sequence can be modeled by white noise, thus $\boldsymbol{R}_N(t)$ can be replaced by a diagonal matrix. It is well known that the two algorithms do not exhibit a significant difference in performance when the input is indeed white noise. Even though LMS models the input as white, it still works for nonwhite sequences given that some weak persistent excitation conditions are satisfied. This is a consequence of the convergence properties of the algorithm [15]. Unfortunately, the convergence rate is reduced if the white noise model does not adequately model the input.

In this paper we will generalize the idea used in LMS. Instead of assuming that the input sequence is white, we will assume that it is autoregressive of order M (AR(M)). Notice that a white noise is just an AR(0) sequence. Also, if we observe in Table I the FAEST algorithm, we can see that it uses predictors of order N for the prediction part. Thus FAEST models the input as an AR(N) sequence. Concluding we have that LMS assumes an AR(0) input and FAEST (and most fast LS versions) an AR(N). By assuming an AR(M) input sequence with $0 \leq M \leq N$, we can see that LMS and the fast versions of LS are the two extremes of our class.

Having decided the input model, it is clear from (1) that the only quantity to be defined is the matrix $\boldsymbol{R}_N(t)$ for a nonstationary AR(M) sequence. We will define the covariance matrix in a somewhat indirect way. We will regard the definition of the matrix as an extension problem, namely, the extension of a covariance matrix which is partially known. Specifically, we want to define a covariance matrix $\boldsymbol{R}_N$ for which we are given the elements of the diagonal and all elements of the M off-diagonals. Such a problem is already solved for the stationary case with the use of the maximum entropy (ME) criterion. It was also shown [14] that the ME criterion is equivalent to assuming that the underlying sequence is AR(M). Here we will try to extend this result to the nonstationary case. We will use a criterion different from the ME in order to avoid risky generalizations of the entropy concept for nonstationary sequences.

Let us denote with $\boldsymbol{R}_{k+1,m}$, where $k > m$, a covariance matrix of size $k + 1$ for which all elements of its diagonal and of the m off-diagonals are known. Let us first consider solving the simplest problem, that of extending a covariance matrix $\boldsymbol{R}_{k+1,k-1}$ when only the upper right corner

TABLE I
THE FAEST ALGORITHM (PREDICTION PART)

Available at time: t	$C_M(t-1)$, $A_M(t-1)$, $B_M(t-1)$, $X_M(t-1)$ $\gamma_M(t-1)$, $\alpha^f_M(t-1)$, $\alpha^b_M(t-1)$			
New information:	$x(t)$			
Time Update of the Gain Vector		MULT	/	DIV
$e^f_M(t) = x(t) - A^T_M(t-1)\,X_M(t-1)$		M	/	
$\epsilon^f_M(t) = e^f_M(t)/\gamma_M(t)$			/	1
$\alpha^f_M(t) = \lambda\alpha^f_M(t-1) + e^f_M(t)\,\epsilon^f_M(t)$		2	/	
$\gamma_{M+1}(t) = \gamma_M(t)\,\alpha^f_M(t)/\lambda\alpha^f_M(t-1)$		2	/	1
$C_{M+1}(t) = \begin{bmatrix} 0 \\ C_M(t-1) \end{bmatrix} - \frac{e^f_M(t)}{\lambda\alpha^f_M(t-1)} \begin{bmatrix} 1 \\ -A_M(t-1) \end{bmatrix}$		$M+1$	/	1
$A_M(t) = A_M(t-1) - \epsilon^f_M(t)\,C_M(t-1)$		M	/	
$e^b_M(t) = -\lambda\alpha^b_M(t-1)\,C^{M+1}_{M+1}(t)$		2	/	
$\theta(t) = 1 + e^b_M(t)\,C^{M+1}_{M+1}(t)/\gamma_{M+1}(t)$		1	/	1
$\gamma_M(t) = \theta_M(t)\,\gamma_{M+1}(t)$		1	/	
$\epsilon^b_M(t) = e^b_M(t)/\gamma_M(t)$			/	1
$\alpha^b_M(t) = \lambda\alpha_M(t-1) + e^b_M(t)\,\epsilon^b_M(t)$		2	/	
$\begin{bmatrix} C_M(t) \\ 0 \end{bmatrix} = C_{M+1}(t) - C^{M+1}_{M+1} \begin{bmatrix} -B_M(t-1) \\ 1 \end{bmatrix}$		M	/	
$B_M(t) = B_M(t-1) - \epsilon^b_M(t)\,C_M(t)$		M	/	
Total Number of Mult/Div		$5M+11$	/	5

and lower left corner element is unknown. Suppose that

$$\boldsymbol{R}_{k+1,k-1}(\hat{r}) = \begin{bmatrix} r_{0,0} & V^T_{k-1} & \hat{r} \\ V_{k-1} & \boldsymbol{R}_{k-1} & W_{k-1} \\ \hat{r} & W^T_{k-1} & r_{k,k} \end{bmatrix} \tag{2}$$

where $\hat{r}$ is the element we want to define and all other entries are assumed known. Since prediction is of crucial importance for estimation we shall define $\hat{r}$ in an optimum way with respect to prediction. Let A_k, B_k be a forward and a backward predictor, respectively. If we call $\alpha^f(A_k, \hat{r})$ and $\alpha^b(B_k, \hat{r})$ the respective powers of the two prediction errors we have

$$\alpha^f(A_k, \hat{r}) = [1 \;\; -A^T_k]\boldsymbol{R}_{k+1,k-1}(\hat{r})[1 \;\; -A^T_k]^T \tag{3a}$$

$$\alpha^b(B_k, \hat{r}) = [-B^T_k \;\; 1]\boldsymbol{R}_{k+1,k-1}(\hat{r})[-B^T_k \;\; 1]^T. \tag{3b}$$

Consider now a game theoretic approach of the extension problem, a game between nature and us. Nature selects $\hat{r}$ while we select the predictors; for each combination our loss is the corresponding α. We try to make our loss as minimal as possible while nature tries to maximize it. We can distinguish two problems which are common in game theory [5, p. 33]:

P1: The Max-Min Problem: Find $\hat{r}^*$, A^*_k, B^*_k such that

$$\alpha^f(A^*_k, \hat{r}^*) = \max_{\hat{r}} \min_{A_k} \alpha^f(A_k, \hat{r}) \tag{4a}$$

$$\alpha^b(B^*_k, \hat{r}^*) = \max_{\hat{r}} \min_{B_k} \alpha^b(B_k, \hat{r}). \tag{4b}$$

The above equations require some comments [5, p. 57]. They basically reveal that if nature were a rational opponent plotting our ruin, she could use a least favorable $\hat{r}$ (the $\hat{r}^*$) that would guarantee her that our loss would be at least the amounts defined in (4) no matter what selection of predictors we might use.

P2: The Min-Max Problem: Find $\hat{r}^*$, A^*_k, B^*_k such that

$$\alpha^f(A^*_k, \hat{r}^*) = \min_{A_k} \max_{\hat{r}} \alpha^f(A_k, \hat{r}) \tag{5a}$$

$$\alpha^b(B^*_k, \hat{r}^*) = \min_{B_k} \max_{\hat{r}} \alpha^b(B_k, \hat{r}). \tag{5b}$$

Here we have a selection of predictors A^*_k, B^*_k that ensure that our expected loss will not exceed the quantities in (5) no matter which $\hat{r}$ nature decides to use.

The value of the max-min problem is known as "the lower value" of the game and the value of the min-max as "the upper value." The lower is always less than or equal to the upper value. When the two values are equal, the common value is called "the value" of the game. It is by no means obvious that a single triple $(\hat{r}^*, A^*_k, B^*_k)$ exists satisfying the optimization problems defined by (4a, b) and (5a, b). With the next theorem we show that such a triple indeed exists by identifying it and by showing that it solves all optimization problems at the same time.

Theorem 1: Let $\boldsymbol{R}_{k+1,k-1}(\hat{r})$ defined in (2) be a matrix with the two principal minors of size k being positive definite matrices, then the triple $\hat{r}^*$, A^*_k, B^*_k defined as

$$\hat{r}^* = V^T_{k-1}\boldsymbol{R}^{-1}_{k-1}W_{k-1} \tag{6a}$$

$$A^*_k = \begin{bmatrix} \boldsymbol{R}_{k-1} & W_{k-1} \\ W^T_{k-1} & r_{k,k} \end{bmatrix}^{-1} \begin{bmatrix} V_{k-1} \\ \hat{r}^* \end{bmatrix} = \begin{bmatrix} \boldsymbol{R}^{-1}_{k-1}V_{k-1} \\ 0 \end{bmatrix} \tag{6b}$$

$$B^*_k = \begin{bmatrix} r_{0,0} & V^T_{k-1} \\ V_{k-1} & \boldsymbol{R}_{k-1} \end{bmatrix}^{-1} \begin{bmatrix} \hat{r}^* \\ W_{k-1} \end{bmatrix} = \begin{bmatrix} 0 \\ \boldsymbol{R}^{-1}_{k-1}W_{k-1} \end{bmatrix} \tag{6c}$$

solves all problems defined in P1 and P2. Also

$$\det\{\boldsymbol{R}_{k+1,k-1}(\hat{r}^*)\} > 0. \tag{7}$$

Proof: The complete proof of this theorem is given in the Appendix.

With Theorem 1, we have shown that if we extend our matrix using $\hat{r}^*$ then this extension is max-min and min-max optimum simultaneously with respect to forward and backward prediction. Equations (6b), (6c) suggest that A^*_k, B^*_k are the optimal predictors under the optimal extension $\hat{r}^*$ of the matrix. Also these predictors of order k are equal to the optimal predictors of order $k-1$ enlarged properly with a zero. It is exactly this property that will reduce the complexity of our algorithms compared to FAEST. Finally, inequality (7) shows that the optimal extension of the matrix is positive definite and thus a legitimate covariance matrix.

Let us now apply Theorem 1 to solve the general extension problem. Consider the following covariance matrix corresponding to a nonstationary case with more than

one off-diagonals to be defined:

$$\boldsymbol{R}_{k+1,m}(t) = \begin{bmatrix} r_0(t) & \cdots & r_m(t) & \hat{r}_{m+1}(t) & \cdots & \hat{r}_k(t) \\ \vdots & r_0(t-1) & \cdots & r_m(t-1) & \hat{r}_{m+1}(t-1) & \vdots \\ r_m(t) & \vdots & \ddots & \ddots & \ddots & \vdots \\ \hat{r}_{m+1}(t) & r_m(t-1) & \ddots & \ddots & \ddots & \vdots \\ \vdots & \hat{r}_{m+1}(t-1) & \ddots & \ddots & \ddots & \vdots \\ \hat{r}_k(t) & \cdots & \cdots & \cdots & \cdots & r_0(t-k) \end{bmatrix} \tag{8}$$

where again all entries with hats are to be defined while all elements in the diagonal and the off-diagonals of order up to m are assumed known. Before proceeding we must make the following assumption.

Assumption A: All principal minors of size $m+1$ are positive definite matrices.

To extend our matrix we will use the same procedure used in ME. We will first define all elements across the $m+1$ off-diagonal then those of the $m+2$, etc. To define the first element of the $m+1$ off-diagonal consider the first principal minor of size $m+2$. This matrix because of assumption A satisfies all conditions of Theorem 1 and has only one unknown element, namely $\hat{r}_{m+1}(t)$. We can thus, with the use of (6a), define $\hat{r}_{m+1}(t)$. To define the second element, we consider the second principal minor of size $m+2$. For this matrix the only unknown element is $\hat{r}_{m+1}(t-1)$ and the matrix again satisfies the conditions of Theorem 1. Proceeding in this manner we can completely define all elements in the $m+1$ off-diagonal. The resulting principal minors of size $m+2$ are all positive definite because of (7). Having completed with the $m+1$ off-diagonal we consider principal minors of size $m+3$ to define the $m+2$ off-diagonal. Continuing in this way we can completely fill the whole matrix. We can also give a recurrent formula for the elements defined in this way. Using (6a) we have for $j > m$

$$\hat{r}_j(t) = [r_1(t) \cdots \hat{r}_{j-1}(t)]\boldsymbol{R}^{-1}_{j-1,m}(t-1) \cdot \begin{bmatrix} \hat{r}_{j-1}(t-1) \\ \vdots \\ r_1(t-j+1) \end{bmatrix}. \tag{9}$$

The whole procedure may seem quite complex but as we will see it will be considerably simplified by considering the inverse of the matrix $\boldsymbol{R}_{k+1,m}(t)$, which is actually involved in the SN algorithmic class. With the next theorem we present three properties that this inverse satisfies, which are the key points in defining the versions of our class of algorithms.

Theorem 2: Let $\boldsymbol{R}_{k+1,m}(t)$ be the matrix defined by (8), (9), with $k > m$ and satisfying assumption A, then the inverse of this matrix satisfies the following properties:

$$\boldsymbol{R}^{-1}_{k+1,m}(t) = \begin{bmatrix} 0 & \boldsymbol{O}_k^T \\ \boldsymbol{O}_k & \boldsymbol{R}^{-1}_{k,m}(t-1) \end{bmatrix} + \frac{1}{\alpha^f_m(t)} \begin{bmatrix} 1 \\ -A_m(t) \\ \boldsymbol{O}_{k-m} \end{bmatrix} \cdot [1 \; -A_m^T(t) \; \boldsymbol{O}^T_{k-m}] \tag{10a}$$

$$= \begin{bmatrix} \boldsymbol{R}^{-1}_{k,m}(t) & \boldsymbol{O}_k \\ \boldsymbol{O}_k^T & 0 \end{bmatrix} + \frac{1}{\alpha^b_m(t-k+m)} \cdot \begin{bmatrix} \boldsymbol{O}_{k-m} \\ -B_m(t-k+m) \\ 1 \end{bmatrix} \cdot [\boldsymbol{O}^T_{k-m} \; -B_m^T(t-k+m) \; 1] \tag{10b}$$

$$\boldsymbol{R}^{-1}_{k+1,m}(t) = \begin{bmatrix} \boldsymbol{O}_{k+1-m,k+1-m} & \boldsymbol{O}_{k+1-m,m} \\ \boldsymbol{O}_{m,k+1-m} & \boldsymbol{R}^{-1}_{m,m}(t-k-1+m) \end{bmatrix} + \sum_{j=0}^{k-m} \frac{1}{\alpha^f_m(t-j)} \begin{bmatrix} \boldsymbol{O}_j \\ 1 \\ -A_m(t-j) \\ \boldsymbol{O}_{k-m-j} \end{bmatrix} \cdot [\boldsymbol{O}_j^T \; 1 \; -A_m^T(t-j) \; \boldsymbol{O}^T_{k-m-j}] \tag{11}$$

$$\boldsymbol{R}^{-1}_{k+1,m}(t) = \begin{bmatrix} \boldsymbol{R}^{-1}_{m,m}(t) & \boldsymbol{O}_{m,k+1-m} \\ \boldsymbol{O}_{k+1-m,m} & \boldsymbol{O}_{k+1-m,k+1-m} \end{bmatrix} + \sum_{j=0}^{k-m} \frac{1}{\alpha^b_m(t-j)} \begin{bmatrix} \boldsymbol{O}_j \\ -B_m(t-j) \\ 1 \\ \boldsymbol{O}_{k-m-j} \end{bmatrix} \cdot [\boldsymbol{O}_j^T \; -B_m^T(t-j) \; 1 \; \boldsymbol{O}^T_{k-m-j}] \tag{12}$$

where $\boldsymbol{O}_i$ is a zero vector of size i and $\boldsymbol{O}_{i,j}$ a zero matrix of dimensions $i \times j$. Also $A_m(t)$, $B_m(t)$ are the optimum

forward and backward predictors of order m and $\alpha_m^f(t)$, $\alpha_m^b(t)$ their corresponding prediction error powers. By taking derivatives in (3a), (3b) with respect to A_k, B_k it can be easily shown that they are given by

$$A_m(t) = \boldsymbol{R}_{m,m}^{-1}(t-1)\begin{bmatrix} r_1(t) \\ \vdots \\ r_m(t) \end{bmatrix} \tag{13a}$$

$$B_m(t) = \boldsymbol{R}_{m,m}^{-1}(t)\begin{bmatrix} r_m(t) \\ \vdots \\ r_1(t-m+1) \end{bmatrix} \tag{13b}$$

$$\alpha_m^f(t) = r_0(t) - [r_1(t) \cdots r_m(t)]A_m(t) \tag{14a}$$

$$\alpha_m^b(t) = r_0(t-m) - [r_m(t) \cdots r_1(t-m+1)]B_m(t). \tag{14b}$$

Proof: The proof of this theorem is also given in the Appendix.

Notice from Theorem 2 that in order to find the inverse of the matrix $\boldsymbol{R}_{k+1,m}(t)$ we need to compute predictors only of order m. Notice also from (11), (12) that the inverse has all off-diagonals of order greater than m equal to zero. This property is characteristic for covariances of AR(m) processes. In other words, the matrix $\boldsymbol{R}_{k+1,m}(t)$ can be regarded as an estimate of the covariance matrix that takes into account the prior knowledge of the order m. In the next section we are going to use the results given in Theorem 2 in order to derive three versions of our new class of algorithms.

III. Derivation of the New Class of Algorithms

In this section we will use the results of Theorem 2 in order to define three efficient schemes for computing the gain vector $C_N(t)$ defined in (1a). Since we would like the class of algorithms we are going to define to contain LMS and FAEST as its members, we will slightly modify the definition of the SN class. This is necessary because FAEST and FTF, in order to achieve the $7N$ complexity, use the dual Kalman gain and not the usual Kalman gain [3], [4]. Thus the general form of our class becomes

$$C_N(t) = -\frac{1}{\lambda}\boldsymbol{R}_N^{-1}(t-1)\,X_N(t) \tag{15a}$$

$$\epsilon_N(t) = y(t) - H_N^T(t)\,X_N(t) \tag{15b}$$

$$H_N(t) = H_N(t-1) - \epsilon_N(t)\,C_N(t) \tag{15c}$$

where $0 < \lambda \leq 1$ is the forgetting factor. The problem of the posterior error $\epsilon_N(t)$ is solved in the same way as in FAEST. From (15c) we can easily express the posterior error in terms of the prior. This gives

$$\epsilon_N(t) = \frac{e(t)}{\gamma_{N,M}(t)} \tag{16}$$

where

$$e(t) = y(t) - H_N^T(t-1)\,X_N(t) \tag{17a}$$

$$\gamma_{N,M}(t) = 1 - C_N^T(t)\,X_N(t). \tag{17b}$$

Notice that $\gamma_{N,M}(t)$ has exactly the same properties as the corresponding variable (power) in FAEST. It is nonnegative and greater than unity. The algorithms which follow give recursions for the gain vector $C_N(t)$ and also for $\gamma_{N,M}(t)$.

Let us now define the matrix $\boldsymbol{R}_N(t)$. In order to apply the theory of the previous section it is clear that we must define this matrix as some $\boldsymbol{R}_{N,M}(t)$ matrix of the form of (8). For the known elements we will use the sample estimates, that is

$$r_i(t-j) = \sum_k^{t-j} \lambda^{t-j-k}x(k)\,x(k-i) \quad \text{for } |i-j| \leq M. \tag{18}$$

In other words, the diagonal and the M off-diagonals coincide with the corresponding elements of the sample covariance matrix used by all LS algorithms. From (18) we conclude that all principal minors of order $M+1$ can be written as $\Sigma_k^{t-j}\,\lambda^{t-j-k}X_{M+1}(k)\,X_{M+1}^T(k)$ and thus under persistent excitation are positive definite. Since assumption A holds, we can use the procedure defined in Section II to fill the rest of the matrix. If we apply the results of Theorem 2 to this matrix, all predictors of order M that are involved in (10)–(12) are *LS optimal predictors* of order M and thus can be computed using any LS algorithm RLS, FAEST, FTF, etc. The three versions of our algorithmic class are as follows.

Version 1: This version uses (10) and yields a FAEST-type adaptation of the gain vector. Applying (10a), (10b) for $k = N$, $m = M$, and $t-1$. After multiplying with $X_{N+1}(t)$ and using (15a) we have

$$C_{N+1}(t) = \begin{bmatrix} 0 \\ C_N(t-1) \end{bmatrix} - \frac{e_M^f(t)}{\lambda\alpha_M^f(t-1)}\begin{bmatrix} 1 \\ -A_M(t-1) \\ O_{N-M} \end{bmatrix}$$

$$= \begin{bmatrix} C_N(t) \\ 0 \end{bmatrix} - \frac{e_M^b(t-N+M)}{\lambda\alpha_M^b(t-N+M-1)} \tag{19a}$$

$$\cdot \begin{bmatrix} O_{N-M} \\ -B_M(t-N+M) \\ 1 \end{bmatrix} \tag{19b}$$

where $e_M^f(t)$, $e_M^b(t)$ are the forward and backward LS predictions errors of order M and $\alpha_M^f(t)$, $\alpha_M^b(t)$ their corresponding powers. From (17b) and using (19a), (19b) we have

$$\gamma_{N,M}(t) = \gamma_{N,M}(t-1) + \frac{[e_M^f(t)]^2}{\lambda\alpha_M^f(t-1)} - \frac{[e_M^b(t-N+M)]^2}{\lambda\alpha_M^b(t-N+M-1)}. \tag{20}$$

Version 2: For this version we use (11) and thus we compute the gain using only forward predictors

$$C_N(t) = \begin{bmatrix} O_{N-M} \\ C_M(t-N+M) \end{bmatrix} - G_N(t) \tag{21}$$

where $G_N(t)$ corresponds to the sum of the forward predictors in (11) and satisfies

$$G_{N+1}(t) = \begin{bmatrix} 0 \\ G_N(t-1) \end{bmatrix} + \frac{e_M^f(t)}{\lambda \alpha_M^f(t-1)} \begin{bmatrix} 1 \\ -A_M(t-1) \\ O_{N-M} \end{bmatrix} \tag{22a}$$

$$= \begin{bmatrix} G_N(t) \\ 0 \end{bmatrix} + \frac{e_M^f(t-N+M)}{\lambda \alpha_M^f(t-N+M-1)} \cdot \begin{bmatrix} O_{N-M} \\ 1 \\ -A_M(t-N+M) \end{bmatrix}. \tag{22b}$$

For $\gamma_{N,M}(t)$ we have

$$\gamma_{N,M}(t) = \gamma_M(t-N+M) + g_N(t) \tag{23}$$

where $g_N(t) = X_N^T(t)\, G_N(t)$ and satisfies

$$g_N(t) = g_N(t-1) + \frac{[e_M^f(t)]^2}{\lambda \alpha_M^f(t-1)} - \frac{[e_M^f(t-N+M)]^2}{\lambda \alpha_M^f(t-N+M-1)}. \tag{24}$$

Version 3: Here we use (12) and thus only backward predictors:

$$C_N(t) = \begin{bmatrix} C_M(t) \\ O_{N-M} \end{bmatrix} - F_N(t) \tag{25}$$

where $F_N(t)$ corresponds to the sum of the backward predictors and satisfies

$$F_{N+1}(t) = \begin{bmatrix} 0 \\ F_N(t-1) \end{bmatrix} + \frac{e_M^b(t)}{\lambda \alpha_M^b(t-1)} \cdot \begin{bmatrix} -B_M(t-1) \\ 1 \\ O_{N-M} \end{bmatrix} \tag{26a}$$

$$= \begin{bmatrix} F_N(t) \\ 0 \end{bmatrix} + \frac{e_M^b(t-N+M)}{\lambda \alpha_M^b(t-N+M-1)} \cdot \begin{bmatrix} O_{N-M} \\ -B_M(t-N+M-1) \\ 1 \end{bmatrix}. \tag{26b}$$

For $\gamma_{N,M}(t)$ we have

$$\gamma_{N,M}(t) = \gamma_M(t) + f_N(t) \tag{27}$$

where $f_N(t) = X_N^T(t)\, F_N(t)$ and satisfies

$$f_N(t) = f_N(t-1) + \frac{[e_M^b(t)]^2}{\lambda \alpha_M^b(t-1)} - \frac{[e_M^b(t-N+M)]^2}{\lambda \alpha_M^b(t-N+M-1)}. \tag{28}$$

These are the three versions of our class. The complete estimation algorithms for the three versions are summarized in Table II.

Comments: Even though it seems that all versions need $2M$ multiplications for the computation of the gain, this is not the case when the predictors are computed via the FAEST algorithm. Notice that the quantities which are necessary for the adaptation of $C_N(t)$ are also computed in FAEST for the adaptation of $C_M(t)$. Thus the only additional operations are $2M$ additions for version 1 and $3M$ additions for versions 2 and 3. Of course, this will be the case if we keep in memory the vectors that will be used after $N - M$ time instants. The memory requirements are $M \times (N - M)$ for versions 1 and 3 and twice as much for version 2 (in the latter we must keep in memory $C_M(t)$ as well). Clearly, if we want to save memory we can always have two FAEST or RLS in parallel to compute the predictors, one applied to samples of the time instant t and another to samples of the time instant $t - N + M$. Such a combination though will double the complexity of the prediction part. If the RLS is used for the prediction part, then versions 2 and 3 are preferable because RLS computes directly $C_M(t)$ and adapts only one filter. Note that the use of RLS may not be prohibitive since in many cases $M \ll N$ and it may be desirable due to its good numerical properties.

We can, without any difficulty, derive similar algorithms for the class defined in (1). The only difference with our class will be the fact that, if the prediction part is implemented with a fast algorithm, this algorithm will have complexity $6M$ instead of $5M$ [4]. In any case, this increase is not important since most fast versions have stability problems. Thus a stabilized version [2], [10], [13] having complexity $6M$ to $8M$, will be preferable.

IV. Simulations

In this section we will present a simulation example for our class and discuss problems encountered in practice. Since our class is a combination of LMS and fast LS algorithms, it has the tendency to diverge for the same reasons that these algorithms diverge. The basic sources of divergence for the fast LS are the initialization method and the accumulation of roundoff errors. For the initialization it was observed that the fast LS algorithms behave much better if a soft constraint initialization is used instead of an exact initialization [4]. It is exactly this initialization that we are going to use in our example for the

TABLE II
THREE VERSIONS OF THE FNTF ALGORITHMS

The Fast Newton Transversal Filter (FNTF) Algorithms

Define:

$$S_{M+1}(t) = \frac{e_M^f(t)}{\lambda \alpha_M^f(t-1)} \begin{bmatrix} 1 \\ -A_M(t-1) \end{bmatrix}$$

$$U_{M+1}(t) = \frac{e_M^b(t)}{\lambda \alpha_M^b(t-1)} \begin{bmatrix} -B_M(t-1) \\ 1 \end{bmatrix}, \quad t^c = t - N + M$$

Version 1. Using Forward and Backward Predictors

Available at time t: $C_N(t-1)$, $\gamma_{N,M}(t-1)$
From any LS algorithm: $S_{M+1}(t)$, $e_M^f(t)$, $U_{M+1}(t^c)$, $e_M^b(t^c)$

$$C_{N+1}(t) = \begin{bmatrix} 0 \\ C_N(t-1) \end{bmatrix} - \begin{bmatrix} S_{M+1}(t) \\ O_{N-M} \end{bmatrix}$$

$$\begin{bmatrix} C_N(t) \\ 0 \end{bmatrix} = C_{N+1}(t) + \begin{bmatrix} O_{N-M} \\ U_{M+1}(t^c) \end{bmatrix}$$

$$\gamma_{N,M}(t) = \gamma_{N,M}(t-1) + S_{M+1}^{\downarrow}(t)\, e_M^f(t) - U_{M+1}^{M+1\downarrow}(t^c)\, e_M^b(t^c)$$

Version 2. Using Only Forward Predictors

Available at time t: $G_N(t-1)$, $g_N(t-1)$
From any LS algorithm: $S_{M+1}(t)$, $e_M^f(t)$, $S_{M+1}(t^c)$, $e_M^f(t^c)$, $C_M(t^c)$, $\gamma_M(t^c)$

$$G_{N+1}(t) = \begin{bmatrix} 0 \\ G_N(t-1) \end{bmatrix} + \begin{bmatrix} S_{M+1}(t) \\ O_{N-M} \end{bmatrix}$$

$$\begin{bmatrix} G_N(t) \\ 0 \end{bmatrix} = G_{N+1}(t) - \begin{bmatrix} O_{N-M} \\ S_{M+1}(t^c) \end{bmatrix}$$

$$C_N(t) = \begin{bmatrix} O_{N-M} \\ C_M(t^c) \end{bmatrix} - G_N(t)$$

$$g_N(t) = g_N(t-1) + S_{M+1}^{\downarrow}(t)\, e_M^f(t) - S_{M+1}^{\downarrow}(t^c)\, e_M^f(t^c)$$

$$\gamma_{N,M}(t) = \gamma_M(t^c) + g_N(t)$$

Version 3. Using only Backward Predictors

Available at time t: $F_N(t-1)$, $f_N(t-1)$
From any LS algorithm: $U_{M+1}(t)$, $e_M^b(t)$, $U_{M+1}(t^c)$, $e_M^b(t^c)$, $C_M(t)$, $\gamma_M(t)$

$$F_{N+1}(t) = \begin{bmatrix} 0 \\ F_N(t-1) \end{bmatrix} + \begin{bmatrix} U_{M+1}(t) \\ O_{N-M} \end{bmatrix}$$

$$\begin{bmatrix} F_N(t) \\ 0 \end{bmatrix} = F_{N+1}(t) - \begin{bmatrix} O_{N-M} \\ U_{M+1}(t^c) \end{bmatrix}$$

$$C_N(t) = \begin{bmatrix} C_M(t) \\ O_{N-M} \end{bmatrix} - F_N(t)$$

$$f_N(t) = f_N(t-1) + U_{M+1}^{M+1\downarrow}(t)\, e_M^b(t) - U_{M+1}^{M+1\downarrow}(t^c)\, e_M^b(t^c)$$

$$\gamma_{N,M}(t) = \gamma_M(t) + f_N(t)$$

Filtering Part (Common to all Versions)

Available at time t: $H_N(t-1)$, $X_N(t-1)$
New information: $y(t)$, $x(t)$

$$e_N(t) = y(t) - H_N^T(t-1)\, X_N(t)$$

$$\epsilon_N(t) = e_N(t)/\gamma_{N,M}(t)$$

$$H_N(t) = H_N(t-1) - \epsilon_N(t)\, C_N(t)$$

prediction part of the FNTF class. The second source, i.e., the accumulation of the roundoff errors can be overcome by using some stabilized version [2], [10], [13] of the fast LS algorithms. Here we are going to use the SFAEST [10]. The LMS, on the other hand, is known from practice to diverge when the step size is larger than necessary, especially during the start up period of the algorithm [6]. A large step size basically means that the gain vector takes large values. A similar problem arises also for our algorithmic class when M takes small values close to zero. In other words, it is possible for the prediction part to work perfectly well but the filtering part to diverge. Because this type of divergence appears mainly during the start up period, it is possible, with proper initial conditions, to force the gain to be small in the beginning; then letting the algorithm itself modify the gain adaptively. This method worked in all simulations we performed. The algorithm was able to successfully follow changes of the statistics, as we shall see, that were quite abrupt (change of the real model from H_N to $-H_N$). To define initial conditions we use the soft constraint method as in [4] assuming that at time $t = -N$ we had an input equal to $\sqrt{\mu}$ and at all other time instants until $t = 0$ the input was zero. We can easily compute all initial values of the necessary quantities because the matrix $R_{N,M}(t)$ is diagonal. Specifically, for the prediction part, the filters $A_M(t)$, $B_M(t)$, $C_M(t)$ are zero for $t \leq 0$ and the respective powers satisfy $\alpha_M^f(t) = \mu\lambda^{N+t}$, $\alpha_M^b(t) = \mu\lambda^{N+t-M}$ and $\gamma_M(t) = 1$ for $t \leq 0$. Finally, $f_N(0) = g_N(0) = 0$, and $\gamma_{N,M}(0) = 1$. Now, in order to have a small gain, we must give μ a large value. The value $\mu \approx \sigma_x^2/(1-\lambda)$, where σ_x^2 is an estimate of the input power, is usually adequate.

The simulation we present here is for a model $N = 100$ (100 random numbers in the interval $[-1\ 1]$). As input we used a speech signal, a part of the phrase "Our method works fine." In Fig. 1 we can see the input and the output. At some point we change the model from H_N to $-H_N$. Finally, to the output we add a 20-dB white Gaussian noise. Since all algorithms are very sensitive to initial conditions and have a very different behavior, in order to make a fair comparison independent of the initial conditions, we decided to let the algorithms under comparison converge first and then check their tracking capability with respect to the same change in the model. In Fig. 2 we plot the estimation error computed by our algorithm for $M = 0, 8, 15, 100$. We have used for all cases $\lambda = 0.96$ and the predictors were computed with SFAEST (complexity $6M$). Since we are interested in the part after the change, we plot only 100 points before the change. Thus the time of change in Fig. 2 is at $t = 100$. Notice that as M increases, the algorithm has an increased tracking capability. For $M = 15$, its behavior is almost identical to the LS optimum $M = 100$. This was expected since speech signals can be adequately modeled by AR(15) to AR(20).

Comments: Even though the input model used to derive the algorithms is AR(M), this does not mean that the algorithms converge only for this type of inputs. This is the same property as LMS which converges for nonwhite

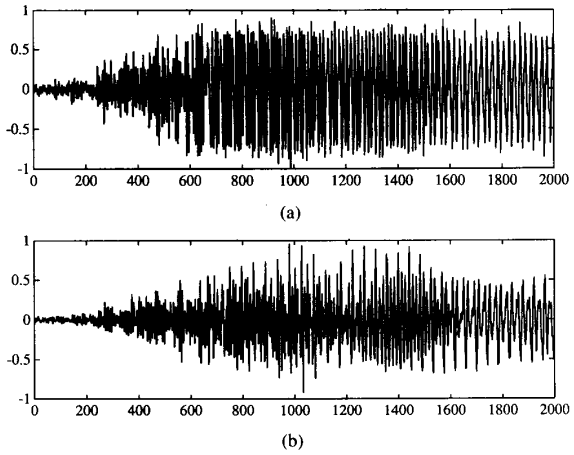


Fig. 1. (a) Input and (b) output of the filter ($N = 100$) used for the simulation.

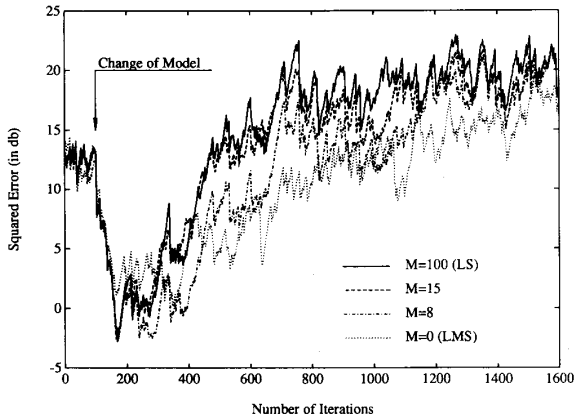


Fig. 2. Performance of the FNTF algorithms for $N = 100$ and $M = 0$, 8, 15, 100.

input sequences as well. One can show using the ODE convergence analysis [9], that the algorithms converge provided that the input is persistently excited of order N. In other words, if the covariance matrix of order N of the input is nonsingular then it can be shown that the algorithms converge regardless of the type of the input. Of course, we expect that their convergence performance will be reduced if the AR(M) does not adequately model the input.

We believe that versions 2 and 3 are more appropriate for deriving simplified algorithms. This is so since the quantities $F_N(t)$ and $G_N(t)$ defined in (22) and (26) will always be a finite sum of terms, because a vector added at some time instant is always subtracted after $N - M$ instants regardless of the way it is computed. This is not the case for version 1 since we add forward, while we subtract backward predictors. Consequently, if the computation is approximate, this might lead to accumulation of terms and eventually to divergence.

V. Conclusion

A new class of algorithms has been derived based on an Mth order AR modeling of the input sequence. The user can select the order M from $M = 0$ up to N, where N is the order of the filter. Depending on the value of M, the complexity and performance of the algorithm varies from that of LMS ($M = 0$) to that of the fast LS schemes ($M = N$). The performance of the algorithm was tested using a speech signal. The simulations showed that the performance was improving with increasing M obtaining almost identical results with LS for $M = 15$ while N was equal to 100. This suggests that in cases where the input signal can be adequately modeled as an AR process of order much less than the order of the filter (such as speech in echo cancellation) this algorithm may offer significant computational savings without sacrificing performance.

Appendix

Proof of Theorem 1: We will prove the theorem only for the forward predictor case since following similar steps it can be shown for the backward. Before proving the theorem let us introduce one more relation which is known as "saddle point" relation in game theory. We will say that a pair $(\hat{r}^*, A_k^*)$ satisfies a saddle point relation if for every predictor A_k and matrix element $\hat{r}$ we have

$$\alpha^f(A_k^*, \hat{r}) \le \alpha^f(A_k^*, \hat{r}^*) \le \alpha^f(A_k, \hat{r}^*). \quad \text{(A.1)}$$

We will first show that if a pair satisfies (A.1) then it also solves the max-min and min-max problems defined by (4a) and (5a). To show (4a) from (A.1), consider the left-hand side (LHS) inequality of (A.1), this yields

$$\min_{A_k} \alpha^f(A_k, \hat{r}) \le \alpha^f(A_k^*, \hat{r}) \le \alpha^f(A_k^*, \hat{r}^*) \quad \text{(A.2)}$$

and thus

$$\max_{\hat{r}} \min_{A_k} \alpha^f(A_k, \hat{r}) \le \alpha^f(A_k^*, \hat{r}^*). \quad \text{(A.3)}$$

Since the right-hand side (RHS) inequality of (A.1) holds for every A_k we have

$$\alpha^f(A_k^*, \hat{r}^*) \le \min_{A_k} \alpha^f(A_k^*, \hat{r}) \le \max_{\hat{r}} \min_{A_k} \alpha^f(A_k, \hat{r}). \quad \text{(A.4)}$$

Combining (A3) and (A4) we conclude that (4a) is true.

To show now (5a), consider the RHS inequality of (A.1). We conclude that

$$\alpha^f(A_k^*, \hat{r}^*) \le \alpha^f(A_k, \hat{r}^*) \le \max_{\hat{r}} \alpha^f(A_k, \hat{r}) \quad \text{(A.5)}$$

and thus

$$\alpha^f(A_k^*, \hat{r}^*) \le \min_{A_k} \max_{\hat{r}} \alpha^f(A_k, \hat{r}). \quad \text{(A.6)}$$

The LHS inequality of (A.1) holds for every $\hat{r}$, thus

$$\alpha^f(A_k^*, \hat{r}^*) \geq \max_{\hat{r}} \alpha^f(A_k^*, \hat{r}) \geq \min_{A_k} \max_{\hat{r}} \alpha^f(A_k, \hat{r}). \tag{A.7}$$

Combination of (A6) and (A7) proves (5a).

Until now we have shown that if a pair $(\hat{r}^*, A_k^*)$ satisfies the saddle point condition defined in (A.1) this is sufficient for solving the min-max and the max-min problems defined with (4a), (5a). Notice also that the two problems have the same value equal to $\alpha^f(A_k^*, \hat{r}^*)$. In order to prove the theorem we conclude that it is sufficient to show that the pair defined by (6a, b) satisfies the saddle point relation (A.1). Consider first the RHS inequality of (A.1), namely, $\alpha^f(A_k^*, \hat{r}^*) \leq \alpha^f(A_k, \hat{r}^*)$. This inequality means that A_k^* is the optimum predictor for $\hat{r} = \hat{r}^*$. Indeed, by differentiating (3a) with respect to A_k we can show that the optimum predictor is given by the LHS equality of (6b). If we apply Shur's inversion formula on the existing partitions we can easily show that the second equality of (6b) is also true. To show now that the RHS inequality in (A.1) is true, that is, $\alpha^f(A_k^*, \hat{r}) \leq \alpha^f(A_k^*, \hat{r}^*)$ we will compute $\alpha^f(A_k^*, \hat{r})$. Substituting (6b) in the definition (3a) yields

$$\alpha^f(A_k^*, \hat{r}) = r_{0,0} - V_{k-1}^T R_{k-1}^{-1} V_{k-1} \tag{A.8}$$

which is independent of $\hat{r}$. In other words, the LHS relation in (A.1) is satisfied as equality. We have thus shown that the quantities defined in (6a, b, c) solve the min-max and the max-min problems. What is left for our proof to be complete is to show inequality (7). Let us compute $\det \{R_{k+1,k-1}(\hat{r})\}$ for an arbitrary $\hat{r}$. Using the property of the determinant for block matrices we obtain

$$\begin{aligned} &\det \{R_{k+1,k-1}(\hat{r})\} \\ &\quad = \det \left\{ \begin{bmatrix} R_{k-1} & W_{k-1} \\ W_{k-1}^T & r_{k,k} \end{bmatrix} \right\} \\ &\qquad \times \left\{ r_{0,0} - [V_{k-1}^T \; \hat{r}] \begin{bmatrix} R_{k-1} & W_{k-1} \\ W_{k-1}^T & r_{k,k} \end{bmatrix}^{-1} \begin{bmatrix} V_{k-1} \\ \hat{r} \end{bmatrix} \right\}. \end{aligned} \tag{A.9}$$

Applying Shur's inversion formula on the last term of (A9), gives

$$\begin{aligned} &\det \{R_{k+1,k-1}(\hat{r})\} \\ &\quad = \det \left\{ \begin{bmatrix} R_{k-1} & W_{k-1} \\ W_{k-1}^T & r_{k,k} \end{bmatrix} \right\} \\ &\qquad \times \left\{ r_{0,0} - V_{k-1}^T R_{k-1}^{-1} V_{k-1} \right. \\ &\qquad \left. - \frac{[\hat{r} - \hat{r}^*]^2}{r_{k,k} - W_{k-1}^T R_{k-1}^{-1} W_{k-1}} \right\}. \end{aligned} \tag{A.10}$$

Since by assumption, the two principal minors of order k are positive definite we conclude from (A10) that $\hat{r}^*$ has the additional property to maximize the determinant of the matrix $R_{k+1,k-1}(\hat{r})$. This last property is actually used for the ME extension of a covariance matrix in the Toeplitz (stationary) case and we see that it carries over to the nonstationary case as well. From (A.10) and the positivity of the minors we conclude that inequality (7) is valid. This concludes the proof of Theorem 1.

Proof of Theorem 2: To prove (10a) we use Shur's inversion formula after forming a lower partition of the matrix $R_{k+1,m}(t)$, thus

$$\begin{aligned} R_{k+1,m}^{-1}(t) &= \begin{bmatrix} 0 & O_k^T \\ O_k & R_{k,m}^{-1}(t-1) \end{bmatrix} \\ &\quad + \frac{1}{\alpha_k^f(t)} \begin{bmatrix} 1 \\ -A_k(t) \end{bmatrix} [1 \; -A_k^T(t)] \end{aligned} \tag{A.11}$$

where $A_k(t)$, $\alpha_k^f(t)$ are the optimum forward predictor of order k and its corresponding error power, respectively. Repeated use of (6b) gives

$$A_k(t) = \begin{bmatrix} A_{k-1}(t) \\ 0 \end{bmatrix} = \cdots = \begin{bmatrix} A_m(t) \\ O_{k-m} \end{bmatrix} \tag{A.12}$$

now using (A12) we have

$$\begin{aligned} \alpha_k^f(t) &= r_{0,0} - A_k^T(t) \, R_{k,m}(t-1) \, A_k(t) \\ &= r_{0,0} - A_m^T(t) \, R_{m,m}(t-1) \, A_m(t) = \alpha_m^f(t). \end{aligned} \tag{A.13}$$

Substituting (A12), (A13) in (A11) proves (10a). Using an upper partitioning in $R_{k+1,m}(t)$ and (6c) proves in the same way (10b). To show now (11) and (12) we repeatedly apply (10a) and (10b) for $k+1, k, \cdots, m+1$. Equations (11) and (12) are not the only relations we can have for the inverse. Equation (11), for example, was the result of repeated application of (10a) only, by mixing (10a) and (10b) we can have several different relations for the inverse that involve in the sum both types of predictors.

References

[1] M. G. Bellanger, *Adaptive Digital Filters and Signal Analysis*. Marcel Dekker, 1987.

[2] J. L. Botto and G. V. Moustakides, "Stabilizing the fast Kalman algorithms," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. 37, pp. 1342-1348, Sept. 1989.

[3] G. Carayannis, D. Manolakis, and N. Kalouptsidis, "A fast sequential algorithm for least squares filtering and prediction," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-31, no. 6, pp. 1394-1402, Dec. 1983.

[4] J. Cioffi and T. Kailath, "Fast, recursive, least squares transversal filters for adaptive processing," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-32, pp. 304-337, Apr. 1984.

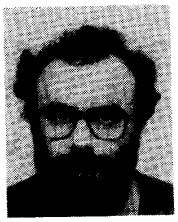
[5] T. S. Ferguson, *Mathematical Statistics, a Decision Theoretic Approach*. New York: Academic, 1967.

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

[7] N. Kalouptsidis, G. Carayannis, D. Manolakis, and E. Koukoutsis, "Efficient recursive in order least squares FIR filtering and prediction," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-33, pp. 1175-1187, Oct. 1985.

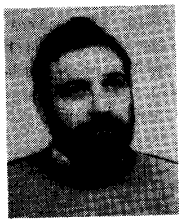
[8] L. Ljung, M. Morf, and D. Falconer, "Fast calculation of gain matrices for recursive estimation schemes," *Int. J. Contr.*, vol. 27, no. 1, pp. 1-9, Jan. 1978.

[9] L. Ljung and T. Söderstrom, *Theory and Practice of Recursive Identification*. Cambridge, MA: M.I.T. Press, 1983.
[10] G. V. Moustakides, "Correcting the instability due to finite precision of the fast Kalman algorithms," *Signal Processing*, vol. 18, pp. 33–42, 1989.
[11] S. J. Orfanidis, *Optimum Signal Processing, an Introduction*, 2nd ed. New York: Macmillan, 1988.
[12] J. G. Proakis and D. Manolakis, *An Introduction to Digital Signal Processing*. New York: Macmillan, 1988.
[13] D. T. M. Slock and T. Kailath, "Numerically stable fast recursive least squares transversal filters," in *Proc. ICASSP 88 Conf.* (New York), Apr. 1988, pp. 1365–1368.
[14] A. Van den Bos, "Alternative interpretation of the maximum entropy spectral analysis," *IEEE Trans. Inform. Theory*, pp. 493–494, July 1971.
[15] B. Widrow and S. P. Sterns, *Adaptive Signal Processing*. Englewood Cliffs, NJ: Prentice-Hall, 1985.



George V. Moustakides was born in Drama, Greece, on April 16, 1955. He received the diploma in electrical engineering from the National Technical University of Athens, Greece, in 1979, the M.Sc. degree in systems engineering from the Moore School of Electrical Engineering, University of Pennsylvania, Philadelphia, in 1980, and the Ph.D. degree in electrical engineering from Princeton University, Princeton, NJ, in 1983.

From 1983 to 1986 he held a research position at the Institut de Researche en Informatique et Systèmes Aléatoires, Rennes, France. From 1986 to 1988 he served in the Greek Air Force. He is currently with the Computer Technology Institute, Patras, Greece. His interests include detection theory, fast estimation algorithms, and theory of optimal stopping times.



Sergios Theodoridis (M'87) was born in Piraeus, Greece, on December 17, 1951. He received the B.Sc. degree in physics with honors from the University of Athens, Athens, Greece, in 1973, and the M.Sc. and Ph.D. degrees in communications and signal processing, both from the University of Birmingham, U.K., in 1975 and 1978, respectively.

From 1978 to 1981 he was a Research Fellow in the Department of Electronics and Electrical Engineering, University of Birmingham. From 1981 to 1983 he was with the Greek Army. Since 1984 he has been with the Department of Computer Engineering, University of Patras, Patras, Greece. His current research interests are focused on the fields of digital signal processing, communications, and VLSI implementation of DSP algorithms.