Simplified Newton-Type Adaptive Estimation Algorithms

Panagiotis P. Mavridis and George V. Moustakides, *Member, IEEE*

***Abstract*—A new adaptive estimation algorithm is presented. It is the result of a combination of the LMS and the fast Newton transversal filters (FNTF) class. The main characteristic of the proposed algorithm is its improved convergence rate as compared to LMS, for cases where it is known that LMS behaves poorly. This improved characteristic is achieved in expense of a slight increase in the computational complexity while the overall algorithmic structure is very simple (LMS type). The proposed algorithm seems also to compare relatively well against RLS and FNTF.**

I. Introduction

THE LMS algorithm is the most widely used, in practice, adaptive estimation algorithm [18]. Its success is due to its simplicity, its low complexity ($2N$ where N is the number of taps to be adjusted) and its robustness. The only characteristic drawback of this algorithm is its low convergence rate, especially when the number of taps N and/or the eigenvalue spread of the input data covariance matrix is large [18]. On the other hand, RLS and its family of fast and stabilized versions [1], [4], [6], [7], [9], [12], [16] is characterized by a fast convergence rate but requires a significantly higher computational complexity [$O(N^2)$ for RLS and $(7\text{–}10)N$ for its fast versions]. Although the complexity of the fast versions is linear with the number of taps N there nevertheless exist applications for which this complexity is restrictive even with today's technology.

The gap between the two algorithms (LMS and RLS), regarding performance and complexity, was filled by the introduction of the fast Newton transversal filters (FNTF) algorithmic class [13], [17]. The FNTF algorithm, instead of modeling the data as white noise (LMS), or as an AR process of order N (fast versions of RLS), models the input data as an AR process of order M. This results in the use of predictors that are of size M instead of size N (with $0 \leq M \leq N$). As M takes values between the two ends (0 and N) the algorithm can achieve performances and complexities that are intermediate of the corresponding performances and complexities of LMS and RLS. Specifically, FNTF by utilizing the fast versions of RLS to solve prediction problems of order M, requires $2N + (5\text{–}8)M$ operations per time step. The usefulness of FNTF is maximized when the input data are indeed (or very close to) an AR process of order M. For such a case the performance of FNTF is similar to the performance of RLS, but with a reduced complexity. Applications where FNTF yields a high gain in complexity without significant sacrifice in performance are Echo Cancellation in Mobile Telephony ($N = 100\text{–}200$, $M = 10\text{–}20$) [14], or Echo Cancellation in Audioconferencing ($N \geq 1000$, $M = 10\text{–}20$). This is because the input data (speech) can be adequately modeled by an AR process of order 10–20, while the size N of the filter is significantly larger.

The advantage of LMS over the fast RLS versions and FNTF, that makes it tractable for most practical problems, is basically its extremely simple structure. During the last years many variants of LMS have been proposed as the Leakage, the Sign Error, the Sign Regressor, and the Sign–Sign LMS [15, p. 84]. All these variants have common aim to achieve comparable or better, if possible, performance than the LMS but by requiring fewer computations per time step. For instance in the Sign Error algorithm, instead of the prediction error, its sign is used thus avoiding N multiplications per time step. Similarly in the Sign Regressor the sign of the regression vector, instead of the original vector, is used thus avoiding again N multiplications. A comparison study of these algorithms can be found in [5], [15] for the i.i.d. input data case.

In this work we present a class of algorithms related to LMS and FNTF. Our aim is to define algorithms by incorporating those elements from FNTF that accelerate convergence, maintaining on the same time the algorithmic structure as simple as possible (similar to LMS). It is thus clear that we do not intend to minimize the computations per time step, but rather to accelerate the convergence. This is achieved in expense of a slight increase in computation. To be more precise, the complexity of the proposed algorithm is $2N+3M$ multiplications and $2N + 4M$ additions per time step. The important thing is that the extra operations, order M, are due to parts of the total algorithm that are LMS like, thus resulting in a very simple overall algorithmic structure. The proposed algorithm can converge significantly faster than LMS in cases of large eigenvalue spread of the input sequence. It is known that for these cases LMS behaves very poorly. The proposed algorithm compares also well with RLS and FNTF while at the same time seems not to have robustness problems due to finite precision as is the case for the fast versions of RLS (FAEST, FTF), and FNTF.

The rest of the paper is outlined as follows. In Section II, the new algorithm is presented. Section III contains a theoretical

Manuscript received January 29, 1994; revised January 20, 1996. The associate editor coordinating the review of this paper and approving it for publication was Fuyun Ling.

The authors are with the Department of Computer Engineering and Informatics, University of Patras, 26500 Patras, Greece, and also with the Computer Technology Institute of Patras, 21100 Patras, Greece.

Publisher Item Identifier S 1053-587X(96)05297-X.

analysis of the proposed algorithm. Based on the theory of Section III, a "fair" comparison of the proposed algorithm with LMS, RLS and FNTF is presented in Section IV. Finally Section V has the concluding remarks.

II. Main Results

Before going to the introduction of the algorithmic class let us define our notation. With lower case letters we denote scalars, with upper case vectors and with bolded upper case matrices, finally superscript "T" denotes transpose. Let us now assume that we are given sequentially two sequences $\{d(n)\}$ and $\{x(n)\}$, where the first signal is the desired response and the second is the input signal. The Newton Algorithm is then a very general adaptive algorithm for identifying an FIR model $W_N(n)$ using the two sequences [10, p. 47]. This algorithm has the following form

$$\begin{aligned} \epsilon(n) &= d(n) - W_N^T(n-1)X_N(n) \\ C_N(n) &= \boldsymbol{Q}_N(n)X_N(n) \\ W_N(n) &= W_N(n-1) + \mu\epsilon(n)C_N(n) \end{aligned} \tag{1}$$

where $X_N(n) = [x(n), \cdots, x(n-N+1)]^T$, $\boldsymbol{Q}_N(n)$ is a matrix that can be properly selected, $C_N(n)$ the corresponding gain vector and μ the (constant) step size of the algorithm.

As we said before the algorithm in (1) is very general. It is easy to see that most known algorithms are special cases of this algorithm corresponding to a specific selection of the matrix $\boldsymbol{Q}_N$ and the step size μ. For example, if $\boldsymbol{Q}_N(n) = \boldsymbol{I}_N$ with $\boldsymbol{I}_N$ the identity matrix the resulting algorithm is the LMS. If $\mu = 1$ and $\boldsymbol{Q}_N(n) = \boldsymbol{R}_N^{-1}(n)$ with $\boldsymbol{R}_N(n)$ the sample covariance matrix of the sequence $x(n)$ then the resulting algorithm is the RLS. In a similar way we can identify other recursive algorithms.

It is now clear that in order to define a new algorithm it is enough to define a new matrix $\boldsymbol{Q}_N$ and an efficient computation scheme for the gain vector C_N. Let us first consider the RLS, that is $\boldsymbol{Q}_N(n) = \boldsymbol{R}_N^{-1}(n)$. From [8, p. 577] we know that we can write

$$\begin{aligned} \boldsymbol{R}_N^{-1}(n) = \sum_{j=0}^{N-1} \frac{1}{\alpha_{N-1-j}^f(n-j)} \begin{bmatrix} 0_j \\ 1 \\ -A_{N-1-j}(n-j) \end{bmatrix} \\ \cdot \begin{bmatrix} 0_j^T & 1 & -A_{N-1-j}^T(n-j) \end{bmatrix} \end{aligned} \tag{2}$$

and

$$\boldsymbol{R}_N^{-1}(n) = \sum_{j=0}^{N-1} \frac{1}{\alpha_j^b(n)} \begin{bmatrix} -B_j(n) \\ 1 \\ 0_{N-j-1} \end{bmatrix} \begin{bmatrix} -B_j^T(n) & 1 & 0_{N-j-1}^T \end{bmatrix} \tag{3}$$

where A_j, B_j and α_j^f, α_j^b are the least squares optimum forward and backward predictors of order j and their corresponding minimum prediction error powers.

In FNTF [13], by assuming that the input sequence is an AR process of order M a different matrix $\boldsymbol{Q}_N$ is proposed. This matrix satisfies relations similar to (2) and (3), only now all optimum predictors of order larger than M are equal to the optimum predictor of order M. The corresponding matrix $\boldsymbol{Q}_N = \boldsymbol{R}_{N,M}^{-1}$ can thus be written as follows:

$$\begin{aligned} \boldsymbol{R}_{N,M}^{-1}(n) = \sum_{j=0}^{N-M-1} \frac{1}{\alpha_M^f(n-j)} \begin{bmatrix} 0_j \\ 1 \\ -A_M(n-j) \\ 0_{N-M-j-1} \end{bmatrix} \\ \cdot \begin{bmatrix} 0_j^T & 1 & -A_M^T(n-j) & 0_{N-M-j-1}^T \end{bmatrix} \\ + \sum_{j=N-M}^{N-1} \frac{1}{\alpha_{N-1-j}^f(n-j)} \\ \cdot \begin{bmatrix} 0_j \\ 1 \\ -A_{N-1-j}(n-j) \end{bmatrix} \\ \cdot \begin{bmatrix} 0_j^T & 1 & -A_{N-1-j}^T(n-j) \end{bmatrix} \end{aligned} \tag{4}$$

$$\begin{aligned} \boldsymbol{R}_{N,M}^{-1}(n) = \sum_{j=0}^{N-M-1} \frac{1}{\alpha_M^b(n-j)} \begin{bmatrix} 0_j \\ -B_M(n-j) \\ 1 \\ 0_{N-M-j-1} \end{bmatrix} \\ \cdot \begin{bmatrix} 0_j^T & -B_M^T(n-j) & 1 & 0_{N-M-j-1}^T \end{bmatrix} \\ + \sum_{j=0}^{M-1} \frac{1}{\alpha_j^b(n)} \begin{bmatrix} -B_j(n) \\ 1 \\ 0_{N-j-1} \end{bmatrix} \\ \cdot \begin{bmatrix} -B_j^T(n) & 1 & 0_{N-j-1}^T \end{bmatrix}. \end{aligned} \tag{5}$$

Both selections ($\boldsymbol{Q}_N = \boldsymbol{R}_N$ and $\boldsymbol{Q}_N = \boldsymbol{R}_{N,M}$) are known to yield algorithms with a high convergence rate.

In this paper our intention is to find a means for accelerating the convergence of LMS while preserving at the same time its very simple structure. A possible way to reach our goal is by properly modifying the matrix $\boldsymbol{Q}_N = \boldsymbol{I}_N$ of the LMS algorithm. We will base our modification on (4), (5) on which we will perform a series of simplifications in order to obtain the desired final matrix.

Since usually $M \ll N$, in both (4) and (5) the first sum contains most information regarding the structure of the inverse of the covariance matrix. As we can see, this sum has a very characteristic property, namely it contains predictors that are of the same order but shifted in time. This is actually the key point utilized by FNTF for achieving its reduced complexity. Here, since we are interested in simplified algorithms, we are going to use the first sum only after making some approximations. Notice that the prediction error powers (α_M^f and α_M^b) are the sums of the squares of the corresponding prediction errors (exponentially weighted) [8]. When the exponential factor is close to unity it is clear that the two powers will change very slowly. Thus our first approximation consists in considering all these quantities equal to some constant k. We can thus define the following matrix

$$\begin{aligned} \boldsymbol{D}_N(n) = k \sum_{j=0}^{N-M-1} \begin{bmatrix} 0_j \\ D_{M+1}(n-j) \\ 0_{N-M-j-1} \end{bmatrix} \\ \cdot \begin{bmatrix} 0_j^T & D_{M+1}^T(n-j) & 0_{N-M-j-1}^T \end{bmatrix} \end{aligned} \tag{6}$$

where the vector $D_{M+1}(n)$ can be either the forward or the backward predictor (including the unities). Notice that the matrix $\boldsymbol{D}_N(n)$ just defined is not of full rank because it is the sum of $N-M$ rank one matrices. There are two directions

one can follow to solve this problem. A first direction would be to find a means to approximate the second sum in (4), (5) which, combined with the matrix $\boldsymbol{D}_N$, will guarantee a full rank matrix. A second and significantly simpler direction is to add the matrix $\boldsymbol{I}_N$ to $\boldsymbol{D}_N$. We followed this last idea. Finally instead of using the matrix $\boldsymbol{D}_N(n)$ to define the matrix $\boldsymbol{Q}_N(n)$ we use $\boldsymbol{D}_N(n-1)$. This corresponds in using the dual Kalman gain in the original RLS and FNTF algorithms. This shift in time was necessary because in the computation of the gain C_N it will result in using prior prediction errors instead of posterior [as is the case with $\boldsymbol{D}_N(n)$]. Therefore we propose the following matrix $\boldsymbol{Q}_N(n)$ in order to define a new algorithm:

$$\boldsymbol{Q}_N(n) = \boldsymbol{I}_N + k \sum_{j=0}^{N-M-1} \begin{bmatrix} 0_j \\ D_{M+1}(n-j-1) \\ 0_{N-M-j-1} \end{bmatrix} \cdot [0_j^T \quad D_{M+1}^T(n-j-1) \quad 0_{N-M-j-1}^T]. \quad (7)$$

The matrix $\boldsymbol{Q}_N$ in (7) can also be regarded as a modification to the corresponding matrix of LMS. Notice that when $k=0$ or $N=M$ this matrix reduces to the one used by LMS.

Having defined $\boldsymbol{Q}_N(n)$ we must propose an efficient scheme for computing the gain $C_N(n)$ of (1). By multiplying $\boldsymbol{Q}_N(n)$ by $X_N(n)$ we obtain

$$C_N(n) = X_N(n) + k \sum_{j=0}^{N-M-1} e_M^d(n-j) \cdot \begin{bmatrix} 0_j \\ D_{M+1}(n-j-1) \\ 0_{N-M-1-j} \end{bmatrix} \quad (8)$$

where D_{M+1}, e_M^d is either the forward or the backward predictor of order M and its corresponding prior prediction error. Because of the shift invariance in both terms of (8) we can efficiently compute $C_N(n)$ as in [13] using a step up–step down process. Specifically we have

$$\begin{bmatrix} C_N(n) \\ * \end{bmatrix} = \begin{bmatrix} x(n) \\ C_N(n-1) \end{bmatrix} + k e_M^d(n) \begin{bmatrix} D_{M+1}(n-1) \\ 0_{N-M} \end{bmatrix} - k e_M^d(n_M) \begin{bmatrix} 0_{N-M} \\ D_{M+1}(n_M-1) \end{bmatrix} \quad (9)$$

where $n_M = n-N+M$ and "*" denotes a "don't care" element.

Comments: Notice that we can apply (9) for computing the gain regardless of the method we use to estimate the predictors. Since in most cases $M \ll N$ we can use LMS for these estimates. Because of the small size of the prediction problem, LMS will quickly converge (compared to the whole algorithm), yielding good estimates for the predictors. This in turn will result in the gain C_N pointing to the right direction, thus accelerating the convergence of the filtering part.

The proposed algorithm is summarized in Table I. In this table, (9) is presented as Version 1. Notice that the total complexity is $2N+3M$ multiplications and $2N+4M$ additions if we store information that will be used at time n_M. Otherwise, we must run two LMS in parallel, one for the time instant n and another for n_M. For this case the complexity becomes $2N+6M$ multiplications and additions. In Section IV, we will see how we can further simplify the algorithm and obtain a final complexity of $2N+3M$ multiplications and additions without the need of storing any information and without sacrificing any performance (Version 2). Notice also that the extra, order M, operations come from an LMS algorithm; thus, the total algorithm has a very simple computational structure.

Regarding the LMS used for the estimation of the predictors, as we can see in Table I, we use a step size μ^f (or μ^b) which is different from the step size μ used in the filtering part. This is natural since the size of the prediction problem is different from the filtering problem (usually much smaller). We can thus use a larger step size for the prediction problem yielding a faster convergence and at the same time a satisfactory steady-state behavior.

III. Theoretical Analysis of the Proposed Algorithm

In this section we will derive the necessary analytic results that will help us perform a "fair" comparison of the proposed algorithm with LMS, RLS, and FNTF. For this reason we are going to follow the method introduced in [3], [5] which consists in selecting the parameters of the algorithms so that the algorithms have the same steady-state performance. The algorithm that converges faster is regarded as the best. Specifically for the proposed algorithm and LMS we will theoretically analyze their convergence rate by studying the behavior of their mean trajectories (ODE method). This analysis will be valid for the "small" step size case.

As steady-state performance measure the covariance of the estimation error is usually proposed [5]. For this matrix there are analytic expressions based on an asymptotic stochastic approximation theory (small μ case). Even though this measure seems the most appropriate for estimation algorithms it has the basic difficulty of being a matrix. Specifically, it is not possible to force all algorithms to have the same asymptotic covariance, necessary requirement to perform a fair comparison, just by selecting the parameters of the algorithms. An alternative performance measure widely used in the literature is the steady-state excess mean square error (EMSE) and this is the one we are going to use in this case. The EMSE is a scalar quantity that enters naturally into the estimation problem and has a practical significance. The only problem with this selection is that the formula we our going to use for estimating its value will be derived based on the Independence Assumption (IA) in contrast to the asymptotic covariance where this assumption is not needed.

To this end, let $\epsilon(n)$ denote the filtering error, then we can divide it into two parts as follows

$$\begin{aligned} \epsilon(n) &= d(n) - W_N^T(n-1)X_N(n) \\ &= [d(n) - W_o^T X_N(n)] \\ &\quad - [W_N^T(n-1) - W_o^T]X_N(n) \end{aligned} \quad (10)$$

where W_o is the optimum Wiener filter. The first term is the minimum filtering error we can achieve (if we have available the statistics of the problem) while the second term is the excess error. Let us call $\sigma_{\min}^2$ the minimum filtering error

TABLE I
LISTING OF THE PROPOSED ALGORITHM

Available from time $n-1$:

$W_N(n-1)$, $C_N(n-1)$, $X_N(n-1)$

$A_M(n-1)$, $B_M(n-1)$, $A_M(n_M-1)$, $B_M(n_M-1)$ $\quad (n_M = n-N+M)$

New Information:

$d(n)$, $x(n)$

Adaptation of Forward or Backward Predictor using LMS:

$$e_M^f(n) = x(n) - A_M^T(n-1)X_M(n-1)$$

$$A_M(n) = A_M(n-1) + \mu^f e_M^f(n) X_M(n-1)$$

$$D_{M+1}(n-1) = \begin{bmatrix} 1 \\ -A_M(n-1) \end{bmatrix}$$

$$e_M^d(n) = e_M^f(n)$$

$$e_M^b(n) = x(n-M) - B_M^T(n-1)X_M(n)$$

$$B_M(n) = B_M(n-1) + \mu^b e_M^b(n) X_M(n)$$

$$D_{M+1}(n-1) = \begin{bmatrix} -B_M(n-1) \\ 1 \end{bmatrix}$$

$$e_M^d(n) = e_M^b(n)$$

Computation of the Gain:

Version 1:

$$\begin{bmatrix} C_N(n) \\ * \end{bmatrix} = \begin{bmatrix} x(n) \\ C_N(n-1) \end{bmatrix} + k e_M^d(n) \begin{bmatrix} D_{M+1}(n-1) \\ 0_{N-M} \end{bmatrix} - k e_M^d(n_M) \begin{bmatrix} 0_{N-M} \\ D_{M+1}(n_M-1) \end{bmatrix}$$

Version 2:

$$\begin{bmatrix} C_N(n) \\ * \end{bmatrix} = \begin{bmatrix} x(n) \\ C_N(n-1) \end{bmatrix} + k e_M^d(n) \begin{bmatrix} D_{M+1}(n-1) \\ 0_{N-M} \end{bmatrix}$$

Filtering Part:

$$\epsilon_N(n) = d(n) - W_N^T(n-1)X_N(n)$$

$$W_N(n) = W_N(n-1) + \mu \epsilon_N(n) C_N(n)$$

variance (the minimum mean square error) and $\sigma_{ex}^2(n)$ the EMSE at time n. To compare now the algorithms we will fix their parameters in order to have the same $\sigma_{ex}^2(\infty)$ and then, either analytically or through simulations, we will observe their relative convergence behavior. It is thus clear that we need to find an expression for the EMSE.

The analysis we are going to use is based on the IA and is valid for "small" step size. Even though the IA is obviously erroneous, it seems that the estimates obtained for the EMSE are relatively accurate when μ is small. This fact was also observed in [11]. In other words, as far as the EMSE is concerned, the IA must be asymptotically ($\mu \to 0$) correct. We can now summarize the following result concerning the asymptotic value of the EMSE.

Theorem 1: Under the IA, the steady-state EMSE of the proposed algorithm satisfies the following property

$$\sigma_{ex}^2(\infty) = \frac{\mu}{2}\sigma_{\min}^2 [\sigma_x^2 N + k\alpha_M^d (N-M)] + o(\mu) \qquad (11)$$

where σ_x^2 is the variance of the input signal $x(n)$, α_M^d is the optimum prediction error power of the predictor D_{M+1} (which is either the forward or the backward predictor of order M including the unities), μ is the step size, k the parameter of the proposed algorithm and $o(\mu)$ denotes a quantity that satisfies $\lim_{\mu \to 0} [o(\mu)/\mu] = 0$.

Proof: The proof is given in the Appendix. ■

As we had mentioned before, we would like to set the parameters of the algorithm in order for $\sigma_{ex}^2(\infty)$ to have some prescribed value. For this reason we will select accordingly the step size μ. Disregarding the term $o(\mu)$ in (11) and solving for μ we have the following equation for the step size:

$$\mu = \frac{2\sigma_{mis}^2}{\sigma_x^2 N + k\alpha_M^d (N-M)} \qquad (12)$$

where

$$\sigma_{mis}^2 = \frac{\sigma_{ex}^2(\infty)}{\sigma_{\min}^2} \qquad (13)$$

is known as the misadjustment. Notice that we can obtain the corresponding estimate for the step size of LMS from (12) by setting $k = 0$, this yields

$$\mu_{LMS} = \frac{\sigma_{mis}^2}{\sigma_x^2 N}. \qquad (14)$$

Summarizing, in order to make our comparisons we will fix the misadjustment and then select the step sizes of the proposed algorithm and LMS through (12) and (14), respectively.

A. Estimate of the Convergence Rate of the Mean Trajectory

In this subsection we will attempt a theoretical analysis of the proposed algorithm aiming also in defining its dependence on the parameter k. We will achieve this by studying the speed of convergence of the mean trajectory using the Ordinary Differential Equation (ODE) method [2, pp. 40–42], [5], [10].

To this end let us consider a general recursive algorithm of the form:

$$\theta(n) = \theta(n-1) - \mu H\Big(\theta(n-1),\, Z(n)\Big) \tag{15}$$

where $\theta(n)$ is the state of the algorithm and $Z(n)$ is the new input data vector at time n. Let us now define

$$h(\theta) = E_\theta\{H\Big(\theta,\, Z(n)\Big)\} \tag{16}$$

where $E_\theta\{\cdot\}$ denotes expectation only with respect to the input data $Z(n)$ thus considering θ as deterministic. The mean trajectory is then defined by the following ODE:

$$\dot{\overline{\theta}}(t) = -h\Big(\overline{\theta}(t)\Big) \tag{17}$$

where the correspondence between continuous and discrete time is

$$t_n = \mu n. \tag{18}$$

Although, in general, we have $\overline{\theta}(t_n) \neq E\{\theta(n)\}$ the name "mean trajectory" for the trajectory of the system in (17) is justified by the fact that $\overline{\theta}(t_n)$ is very close to the real mean trajectory $E\{\theta(n)\}$ and that the approximation becomes better as μ becomes smaller [10].

Let us now consider the mean trajectory of the proposed algorithm where for the prediction part we use the forward predictor (we can apply similar steps for the backward predictor case). Notice that the state of the algorithm involves the combination of the predictor $A_M(n)$ and the filter $W_N(n)$. Thus the mean trajectory satisfies

$$\begin{aligned}
\dot{\overline{A}}_M(t) &= -r^f \boldsymbol{R}_M[\overline{A}_M(t) - A_o] \\
\dot{\overline{W}}_N(t) &= -[\boldsymbol{I}_N + \overline{\boldsymbol{D}}_N(t)]\boldsymbol{R}_N[\overline{W}_N(t) - W_o] \\
\overline{\boldsymbol{D}}_N(t) &= k \sum_{j=0}^{N-M-1} \begin{bmatrix} 0_j \\ 1 \\ -\overline{A}_M(t) \\ 0_{N-M-j-1} \end{bmatrix} \\
&\quad \cdot \begin{bmatrix} 0_j^T & 1 & -\overline{A}_M^T(t) & 0_{N-M-j-1}^T \end{bmatrix}
\end{aligned} \tag{19}$$

where $\boldsymbol{R}_j$ is the covariance matrix, of order j, of the input signal $x(n)$, $r^f = \mu^f/\mu$, and A_o, W_o are the optimum Wiener predictor and filter, respectively. The corresponding mean trajectory for LMS can be obtained by setting $k = 0$ in (19), which yields

$$\dot{\overline{W}}_N^{LMS}(t) = -\boldsymbol{R}_N[\overline{W}_N^{LMS}(t) - W_o]. \tag{20}$$

Our intention now is to find the speed of convergence of the mean trajectory $\overline{W}_N(t_n)$ toward the Wiener solution W_o. This is an easy task for LMS because the ODE in (20) is linear and time invariant. The situation is more difficult for the proposed algorithm because the ODE in (19) is nonlinear. For (19) we will make the following simplification. Specifically, since we are interested in cases where $M \ll N$, we can select μ^f (and thus r^f) so as to have a fast convergence for $\overline{A}_M(t_n)$ toward the Wiener predictor A_o. This will also result in a fast convergence of $\overline{\boldsymbol{D}}_N(t_n)$ to its steady-state value $\boldsymbol{D}_o$ defined as

$$\begin{aligned}
\boldsymbol{D}_o &= k \sum_{j=0}^{N-M-1} \begin{bmatrix} 0_j \\ 1 \\ -A_o \\ 0_{N-M-j-1} \end{bmatrix} \\
&\quad \cdot \begin{bmatrix} 0_j^T & 1 & -A_o^T & 0_{N-M-j-1}^T \end{bmatrix}.
\end{aligned} \tag{21}$$

Replacing $\overline{\boldsymbol{D}}_N(t)$ by $\boldsymbol{D}_o$ in (19) yields the following linear time invariant ODE

$$\dot{\overline{W}}_N(t) = -(\boldsymbol{I}_N + \boldsymbol{D}_o)\boldsymbol{R}_N[\overline{W}_N(t) - W_o]. \tag{22}$$

Writing the ODE in terms of the mean error vector $\overline{U}_N(t) = \overline{W}_N(t) - W_o$ we obtain

$$\dot{\overline{U}}_N(t) = -(\boldsymbol{I}_N + \boldsymbol{D}_o)\boldsymbol{R}_N\overline{U}_N(t). \tag{23}$$

We will thus study the convergence rate of the mean error vector toward zero. Since for systems of the form of (23) the convergence is exponential, we define the convergence rate as the limit

$$CR = -\lim_{n\to\infty} \frac{\log[\|\overline{U}(t_n)\|]}{n}. \tag{24}$$

Using now (18) we have

$$\begin{aligned}
CR &= -\mu \lim_{t_n\to\infty} \frac{\log[\|\overline{U}(t_n)\|]}{t_n} \\
&= -\mu \lim_{t\to\infty} \frac{\log[\|\overline{U}(t)\|]}{t}.
\end{aligned} \tag{25}$$

The next theorem yields the desired convergence rate.

Theorem 2: The convergence rate defined in (24) satisfies the following relation:

$$CR = 2\sigma_{mis}^2 \frac{\lambda_{\min}\{(\boldsymbol{I}_N + \boldsymbol{D}_o)\boldsymbol{R}_N\}}{\sigma_x^2 N + k\alpha_M^d(N-M)} \tag{26}$$

where for $\boldsymbol{A}$ a matrix with real eigenvalues, $\lambda_{\min}\{\boldsymbol{A}\}$, denotes the smallest eigenvalue.

Proof: The proof is given in the Appendix. ■

Using Theorem 2 we can also obtain the corresponding rate for LMS by setting $k = 0$, this yields

$$CR_{\text{LMS}} = 2\sigma_{mis}^2 \frac{\lambda_{\min}\{\boldsymbol{R}_N\}}{\sigma_x^2 N}. \tag{27}$$

Notice that by comparing CR with CR_{LMS} it is possible to compare the two algorithms theoretically. Clearly the algorithm with the larger convergence rate can be considered as better. Unfortunately it was not possible to draw any general conclusions by using (26) and (27). This was due to the fact that $\lambda_{\min}\{(\boldsymbol{I}_N + \boldsymbol{D}_o)\boldsymbol{R}_N\}$ could not be set under a more tractable form. Finally notice that (26) describes also the dependence of the algorithm on the parameter k. We will use this formula in the next section in order to observe the behavior of the algorithm for different values of k and find a means for selecting this parameter properly.

B. Numerical Behavior of the Proposed Algorithm

A very desirable characteristic for fast estimation algorithms is their robustness to finite precision effects. It is known that most fast algorithms as FAEST, FTF, and FNTF present instability under finite precision. This nonrobustness is particularly present in the prediction part of these algorithms and is due to the complicated interrelation between the predictors and the corresponding Kalman gain.

The LMS algorithm, besides its simplicity, is also known for its robustness. Since we use this algorithm in the prediction part of the proposed algorithmic scheme we immediately have robustness in the prediction part, that is, in the part where FAEST, FTF, FNTF are known to fail.

It is possible, using stochastic approximation techniques, to theoretically analyze the algorithm from the point of view of robustness (and show that it is robust). Unfortunately the analysis is quite lengthy and thus we prefer not to include it in the present work. We like only to stress that the analysis is possible because the computation of the gain $C_N(n)$ is open loop, that is, from (9) we can see that the gain is adapted using the predictors but is not consequently used to adapt these predictors for the next time instant (as is the case in FAEST, FTF, and FNTF). Finally, in all simulation we performed we never encountered instability due to finite precision.

IV. Simulation Results

Before proceeding to the simulations let us first make a few more comments. As we said in Section III the complexity of the proposed algorithm is $2N + 3M$ multiplications and $2N + 4M$ additions if we store the information that will be needed at time n_M. Otherwise two LMS are required to run in parallel, one for time n and another for time n_M, increasing the complexity to $2N + 6M$ multiplications and additions.

Let us observe (9) which introduces the extra order M complexity. We can see that the last term which refers to the time instant n_M affects only the last M elements of the gain vector C_N. What is more important, because of the down shift defined by (9), this effect is confined in these last M elements and is not transmitted to any other higher position. This means that the last term of (9) interferes only in the estimates of the last M taps of the filter W_N. Regarding these last M taps there are two facts that drastically limit their contribution in the overall process. First, usually we have $M \ll N$, second the last taps in most practical applications have very small values. In other words if the part of the gain vector that affects the M last filter taps is not the best possible this is not so severe for most practical problems. We can thus discard the last term completely. This will yield the following possible adaptations for $C_N(n)$:

$$\begin{bmatrix} C_N(n) \\ * \end{bmatrix} = \begin{bmatrix} x(n) \\ C_N(n-1) \end{bmatrix} + k e_M^d(n) \begin{bmatrix} D_{M+1}(n-1) \\ 0_{N-M} \end{bmatrix}. \tag{28}$$

Equation (28) is presented in Table I as Version 2. As we will see both versions of the algorithm have almost identical performance.

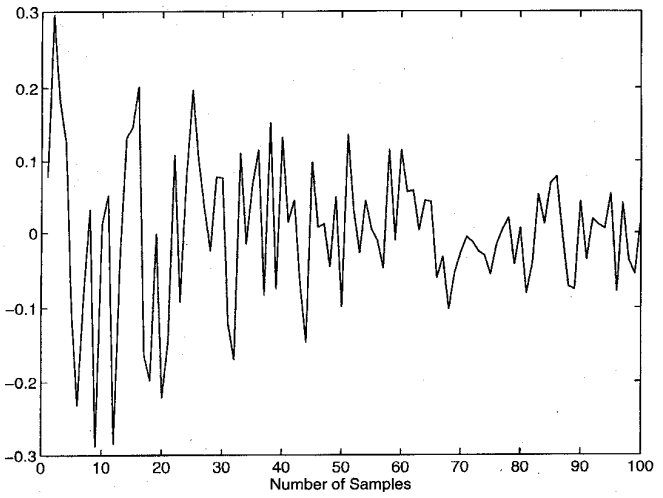


Fig. 1. Unknown FIR system with length $N = 100$.

To test the algorithm we generated an AR process by passing white Gaussian noise, through an all-pole system of order 10 with poles at $0.98\,e^{\pm j0.02\pi}$, $0.96\,e^{\pm j0.21\pi}$, $0.8\,e^{\pm j0.96\pi}$, $0.75\,e^{\pm j0.45\pi}$, and $0.70\,e^{\pm j0.44\pi}$. The signal $x(n)$ was generated by normalizing the AR sequence to be of unit variance. The input sequence $x(n)$ was consequently passed through an FIR filter of length $N = 100$ whose impulse response can be seen in Fig. 1. To the output of the filter a 20-dB white Gaussian noise was added and this resulted in the desired response signal $d(n)$.

For the simulations we used predictors of order $M = 5$ and in the LMS, for the prediction part, a step size $\mu^f = \mu^b = 0.0035$ was selected. The optimum prediction error power, needed for the computation of the step size μ, can be computed as follows. First we obtain the autocorrelation sequence (using for example the backward Levinson [8, pp. 204–207]), then the Wiener solution A_o can be computed yielding also the prediction error power $\alpha_M^d = 0.0019$.

In order to see the dependence of the proposed algorithm on the parameter k we plot the convergence rate of the algorithm as a function of k. We normalize the values of CR so as for $k = 0$ (LMS) the convergence rate is equal to unity. The result can be seen in Fig. 2. We observe that the convergence rate, very abruptly, reaches a maximum value and then decreases very slowly. This indicates that the performance of the algorithm is robust with respect to the value of k. This fact was also observed in the simulations. We can also see from Fig. 2 that for a large region of k values the proposed algorithm is four to six times faster than LMS.

A practical selection for the parameter k yielding most of the time satisfactory results is

$$k \approx \frac{\text{estimate of input power}}{\text{estimate of prediction error power}}. \tag{29}$$

For our simulations, we used $k = 100$ but we obtained similar results with values of $k = 200$, 300, and 400. The step sizes of the proposed algorithm and LMS were selected using (12) and (14) so as to have a misadjustment $\sigma_{mis}^2 = 15$dB resulting in $\mu = 5.36 \times 10^{-4}$ and $\mu_{LMS} = 6.32 \times 10^{-4}$.

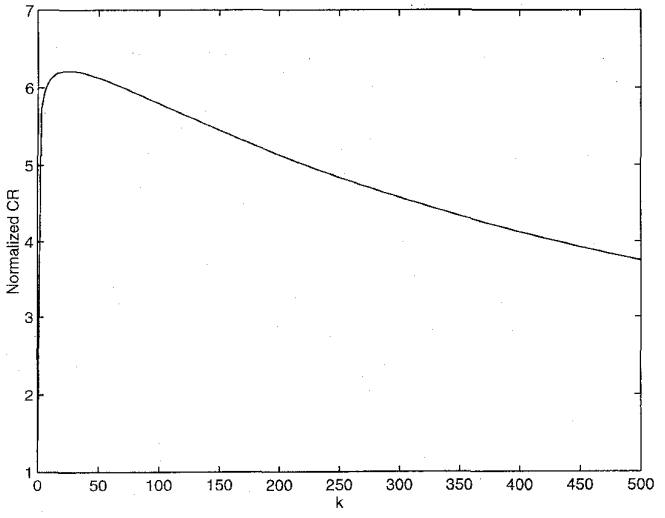


Fig. 2. Convergence rate of the proposed algorithm as a function of the parameter k.

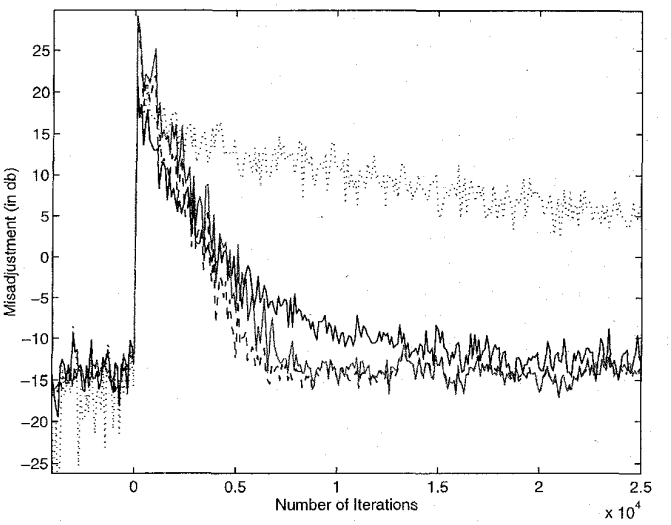


Fig. 3. Performance of the proposed algorithm (solid), RLS (dashed), FNTF (half tone), and LMS (dotted).

In Fig. 3 we compare the proposed algorithm with RLS, FNTF and LMS. For the simulations we first let all algorithms to converge and then we made an abrupt change of the real filter model (from W_o to $-W_o$). We plot the misadjustment of the proposed algorithm (solid), the RLS (dashed), the FNTF (half tone), and the LMS (dotted). Notice that the proposed algorithm converged after 20 000 iterations while LMS needed more than 100 000 iterations, being in agreement with the theoretical results of Fig. 2. On the other hand the algorithm compares well with RLS and FNTF having at the same time a significantly simpler structure and no robustness problems. Finally in Fig. 4 we plot the misadjustment of the two versions of the proposed algorithm (presented in Table I). We can see that their difference in performance is minor. This was typical in all our experiments.

V. Conclusion

We have presented a new adaptive estimation algorithm. The proposed algorithm is characterized by a very simple computational structure similar to the one of LMS but with a significantly higher convergence rate in cases where LMS is known to behave poorly (large eigenvalue spread). The proposed algorithm compares also favorably with other estimation algorithms known to have a fast convergence rate as RLS and FNTF and does not present instability problems due to finite precision as is the case for the fast versions of RLS (FAEST, FTF) and FNTF.

Appendix

Before going to the proof of the two theorems let us first make a definition and prove a number of lemmas. The following definition refers to the ordering of two symmetric matrices. Thus for any two symmetric matrices $\boldsymbol{A}$, $\boldsymbol{B}$ with the same dimensions we will say that $\boldsymbol{A} \geq \boldsymbol{B}$ when the difference $\boldsymbol{A} - \boldsymbol{B}$ is a nonnegative definite matrix.

Lemma 1: Let $\boldsymbol{A}$ and $\boldsymbol{B}$ be two symmetric and positive definite matrices and $b_{\min}$, $b_{\max}$ the smallest and largest eigenvalues of $\boldsymbol{B}$, then

i) $b_{\min}\boldsymbol{I} \leq \boldsymbol{B} \leq b_{\max}\boldsymbol{I}$;
ii) $b_{\min}\,\mathrm{tr}\{\boldsymbol{A}\} \leq \mathrm{tr}\{\boldsymbol{AB}\} \leq b_{\max}\,\mathrm{tr}\{\boldsymbol{A}\}$;

where $\mathrm{tr}\{\boldsymbol{X}\}$ denotes the trace of the matrix $\boldsymbol{X}$.

Proof: For i) we are going to prove only the left-hand side inequality (in the same way we can prove the other). According to the definition we must prove that the difference $\boldsymbol{B} - b_{\min}\boldsymbol{I}$ is nonnegative definite. Since $\boldsymbol{B}$ is symmetric it can be decomposed as $\boldsymbol{B} = \boldsymbol{PDP}^T$ where $\boldsymbol{P}$ is orthonormal and $\boldsymbol{D}$ is diagonal with elements the eigenvalues of $\boldsymbol{B}$ (which are real). Let X be any vector and define $Y = P^T X$, then from the orthonormality of $\boldsymbol{P}$ we have $X^T X = Y^T Y$ and thus

$$\begin{aligned} X^T(\boldsymbol{B} - b_{\min}\boldsymbol{I})X &= X^T\boldsymbol{B}X - b_{\min}X^TX \\ &= Y^T\boldsymbol{D}Y - b_{\min}Y^TY \\ &= y_1^2(b_1 - b_{\min}) + \cdots + y_k^2(b_k - b_{\min}) \\ &\geq 0 \end{aligned} \tag{30}$$

where y_i are the elements of Y and b_i are the elements of $\boldsymbol{D}$. The last inequality is true since by definition $b_{\min}$ is the smallest among all eigenvalues b_i.

To prove ii), notice that since for any two matrices $\boldsymbol{Y}$, $\boldsymbol{Z}$ with appropriate dimensions we have

$$\mathrm{tr}\{\boldsymbol{YZ}\} = \mathrm{tr}\{\boldsymbol{ZY}\}\ , \tag{31}$$

we conclude that

$$\mathrm{tr}\{\boldsymbol{AB}\} = \mathrm{tr}\{\boldsymbol{A}^{1/2}\boldsymbol{B}\boldsymbol{A}^{1/2}\}. \tag{32}$$

From i), by multiplying from right to left with $A^{1/2}$ we conclude that

$$\begin{aligned} b_{\min}\boldsymbol{A} &\leq \boldsymbol{A}^{1/2}\boldsymbol{B}\boldsymbol{A}^{1/2} \\ &\leq b_{\max}\boldsymbol{A}. \end{aligned} \tag{33}$$

Taking traces and using (32) yields the desired relation. ■

Lemma 2: If $\boldsymbol{A}$, $\boldsymbol{B}$ are two symmetric positive definite matrices then the matrices $\boldsymbol{AB}$ and $\boldsymbol{B}^{1/2}\boldsymbol{AB}^{1/2}$ have the same eigenvalues, which are real and positive, also both matrices are diagonalizable.

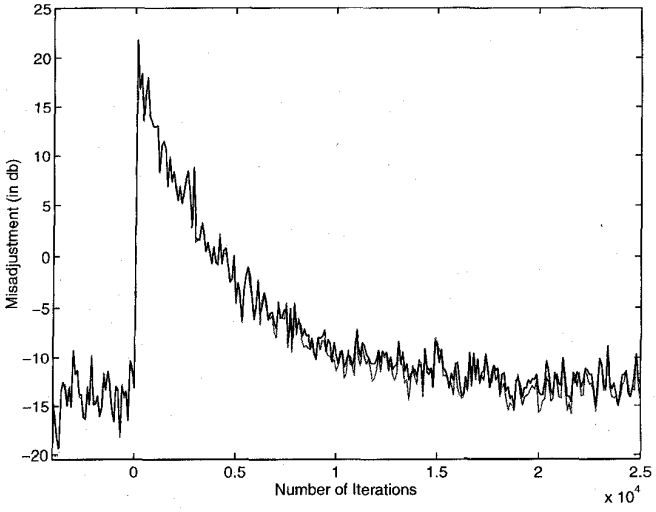


Fig. 4. Performance of the two versions of the proposed algorithm when the forward predictor is used.

Proof: Notice that the matrix $\boldsymbol{B}^{1/2}\boldsymbol{A}\boldsymbol{B}^{1/2}$ is symmetric and positive definite. Thus it is diagonalizable with real and positive eigenvalues. If $\boldsymbol{B}^{1/2}\boldsymbol{A}\boldsymbol{B}^{1/2} = \boldsymbol{P}\boldsymbol{D}\boldsymbol{P}^{-1}$ where $\boldsymbol{D}$ diagonal then we have that $\boldsymbol{A}\boldsymbol{B} = (\boldsymbol{B}^{-1/2}\boldsymbol{P})\boldsymbol{D}(\boldsymbol{B}^{-1/2}\boldsymbol{P})^{-1}$ which means that $\boldsymbol{A}\boldsymbol{B}$ is also diagonalizable to the same diagonal $\boldsymbol{D}$. This concludes the proof. ■

Lemma 3: Let Y be a random vector, Y_o a constant vector and $\mu \ll 1$. If

$$E\{(Y - Y_o)(Y - Y_o)^T\} = O(\mu) \tag{34}$$

then

$$E\{YY^T\} = Y_o Y_o^T + O(\mu^{1/2}) \tag{35}$$

where by $O(x)$ we denote a quantity that satisfies $\|O(x)\| \leq cx$ with c constant independent of x.

Proof: Since for any random vector Z we have that the matrix $E\{(Z - E\{Z\})(Z - E\{Z\})^T\}$ is nonnegative definite (the covariance matrix of Z) we conclude that

$$\begin{aligned} E\{(Z - E\{Z\})(Z - E\{Z\})^T\} &= E\{ZZ^T\} - E\{Z\}E\{Z^T\} \\ &\geq 0 \end{aligned} \tag{36}$$

or equivalently, that $E\{ZZ^T\} \geq E\{Z\}E\{Z^T\}$. We thus conclude that

$$E\{(Y - Y_o)\}E\{(Y - Y_o)^T\} = O(\mu) \tag{37}$$

and consequently

$$E\{(Y - Y_o)\} = O(\mu^{1/2}) \tag{38}$$

Since we can always write

$$\begin{aligned} YY^T = {} & Y_o Y_o^T + Y_o(Y - Y_o)^T + (Y - Y_o)Y_o^T \\ & + (Y - Y_o)(Y - Y_o)^T \end{aligned} \tag{39}$$

taking expectation, using (38) and the hypothesis of the lemma we prove the desired approximation. ■

Proof of Theorem 1: We are interested in obtaining an expression for the EMSE. From (10), if we define $U_N(n) = W_N(n) - W_o$ then we can write

$$\sigma_{ex}^2(n) = E\{[U_N(n-1)^T X_N(n)]^2\}. \tag{40}$$

Using the IA we conclude

$$\begin{aligned} \sigma_{ex}^2(n) &= E\{U_N^T(n-1)\boldsymbol{R}_N U_N(n-1)\} \\ &= \operatorname{tr}\{E\{U_N(n-1)U_N^T(n-1)\}\boldsymbol{R}_N\}. \end{aligned} \tag{41}$$

Notice from (41) that in order to find the EMSE we need to find the covariance matrix of the error vector $U_N(n)$. From now on, for simplicity, let us drop the subscript "N." If we call

$$\boldsymbol{K}(n) = E\{U(n)U^T(n)\} \tag{42}$$

the covariance matrix of $U(n)$, then using (1) we have

$$\begin{aligned} U(n) = {} & [\boldsymbol{I} - \mu\boldsymbol{Q}(n)X(n)X^T(n)]U(n-1) \\ & + \mu\epsilon_o(n)\boldsymbol{Q}(n)X(n) \end{aligned} \tag{43}$$

where $\epsilon_o(n)$ is the estimation error produced by the optimum Wiener filter. In order to find the covariance matrix $\boldsymbol{K}(n)$ we will follow a methodology and hypotheses similar to [8, pp. 315–330]. Assuming independence between $U(n)$, $X(n)$ and $\boldsymbol{Q}(n)$ (independence assumption) and following similar steps we can show that

$$\begin{aligned} \boldsymbol{K}(n+1) = {} & \boldsymbol{K}(n) - \mu[\tilde{\boldsymbol{Q}}(n)\boldsymbol{R}\boldsymbol{K}(n) + \boldsymbol{K}(n)\boldsymbol{R}\tilde{\boldsymbol{Q}}(n)] \\ & + \mu^2 \operatorname{tr}\{\boldsymbol{R}\boldsymbol{K}(n)\}\tilde{\boldsymbol{Q}}(n)\boldsymbol{R}\tilde{\boldsymbol{Q}}(n) \\ & + \mu^2\tilde{\boldsymbol{Q}}(n)\boldsymbol{R}\boldsymbol{K}(n)\boldsymbol{R}\tilde{\boldsymbol{Q}}(n) \\ & + \mu^2\sigma_{\min}^2\tilde{\boldsymbol{Q}}(n)\boldsymbol{R}\tilde{\boldsymbol{Q}}(n) \end{aligned} \tag{44}$$

where $\tilde{\boldsymbol{Q}}(n) = E\{\boldsymbol{Q}(n)\}$ is a symmetric matrix. If we let $n \to \infty$, multiply from the left by $\tilde{\boldsymbol{Q}}^{-1}(\infty)$ and consequently take traces, we obtain

$$\begin{aligned} & 2\operatorname{tr}\{\boldsymbol{R}\boldsymbol{K}\} - \mu\operatorname{tr}\{\boldsymbol{R}\boldsymbol{K}\}\operatorname{tr}\{\boldsymbol{R}\tilde{\boldsymbol{Q}}\} - \mu\operatorname{tr}\{\boldsymbol{R}\boldsymbol{K}\boldsymbol{R}\tilde{\boldsymbol{Q}}\} \\ & \quad = \mu\sigma_{\min}^2\operatorname{tr}\{\boldsymbol{R}\tilde{\boldsymbol{Q}}\}. \end{aligned} \tag{45}$$

Notice that for simplicity we have also dropped the time index from the formulas. The only difficult term in (45) is $\operatorname{tr}\{\boldsymbol{R}\boldsymbol{K}\boldsymbol{R}\tilde{\boldsymbol{Q}}\}$. This term with the help of Lemma 1 can be written as

$$\begin{aligned} \operatorname{tr}\{\boldsymbol{R}\boldsymbol{K}\boldsymbol{R}\tilde{\boldsymbol{Q}}\} &= c\operatorname{tr}\{\boldsymbol{R}\boldsymbol{K}\boldsymbol{R}\} \\ &= c\operatorname{tr}\{\boldsymbol{R}^{1/2}\boldsymbol{K}\boldsymbol{R}^{1/2}\boldsymbol{R}\} \\ &= c'\operatorname{tr}\{\boldsymbol{R}^{1/2}\boldsymbol{K}\boldsymbol{R}^{1/2}\} \\ &= c'\operatorname{tr}\{\boldsymbol{R}\boldsymbol{K}\} \end{aligned} \tag{46}$$

where c, c' can be bounded from above and below by constants. Substituting (46) in (45) and solving for $\operatorname{tr}\{\boldsymbol{R}\boldsymbol{K}\}$ we obtain

$$\begin{aligned} \operatorname{tr}\{\boldsymbol{R}\boldsymbol{K}\} &= \frac{\mu}{2 - \mu c''}\sigma_{\min}^2\operatorname{tr}\{\boldsymbol{R}\tilde{\boldsymbol{Q}}\} \\ &= \frac{\mu}{2}\sigma_{\min}^2\operatorname{tr}\{\boldsymbol{R}\tilde{\boldsymbol{Q}}\} + o(\mu) \end{aligned} \tag{47}$$

where $c'' = c' + \operatorname{tr}\{\boldsymbol{R}\tilde{\boldsymbol{Q}}\}$ which can also be bounded from both sides by constants. We thus conclude

$$\sigma_{ex}^2(\infty) = \frac{\mu}{2}\sigma_{\min}^2\operatorname{tr}\{\boldsymbol{R}E[\boldsymbol{Q}(\infty)]\} + o(\mu). \tag{48}$$

What remains to be specified is the expectation $E\{\boldsymbol{Q}(\infty)\}$. Let us recall the form of the matrix $\boldsymbol{Q}(n)$

$$\boldsymbol{Q}(n) = \boldsymbol{I} + k \sum_{j=0}^{N-M-1} \begin{bmatrix} 0_j \\ D_{M+1}(n-j-1) \\ 0_{N-M-j-1} \end{bmatrix} \cdot [0_j^T \quad D_{M+1}^T(n-j-1) \quad 0_{N-M-j-1}^T]. \quad (49)$$

Notice that the expectation of $\boldsymbol{Q}(n)$ involves expectation of quantities of the form $D(n-j)D^T(n-j)$. The vectors $D(n)$ are predictors (forward or backward) and since they are computed through a regular LMS, we know [18, p. 110] that the covariance of the error $D(n) - D_o$, as $n \to \infty$, is $O(\mu)$, with D_o the corresponding optimum Wiener predictor. Using Lemma 3 we conclude that

$$E\{D(\infty)D^T(\infty)\} = D_o D_o^T + O(\mu^{1/2}). \quad (50)$$

With the help of (31), (49), and (50) we have that

$$\text{tr}\{\boldsymbol{R}E[\boldsymbol{Q}(\infty)]\} = \text{tr}\{\boldsymbol{R}\} + k \sum_{j=0}^{N-M-1} [0_j^T \quad D_o^T \quad 0_{N-M-j-1}^T] \cdot \boldsymbol{R} \begin{bmatrix} 0_j \\ D_o \\ 0_{N-M-j-1} \end{bmatrix} + O(\mu^{1/2}). \quad (51)$$

Since $\boldsymbol{R}$ is Toeplitz all $N-M$ terms in the above sum are equal. Also, since D_o is the optimum Wiener predictor, each term is equal to the optimum prediction error power. Thus

$$\text{tr}\{\boldsymbol{R}E[\boldsymbol{Q}(\infty)]\} = N\sigma_x^2 + k\alpha_M^d(N-M) + O(\mu^{1/2}). \quad (52)$$

Substituting this expression in (48) proves the theorem. ■

Proof of Theorem 2: Notice that the matrices $(\boldsymbol{I}_N + \boldsymbol{D}_o)$, $\boldsymbol{R}_N$ involved in (23) satisfy the assumptions of Lemma 2, this means that the matrix $(\boldsymbol{I}_N + \boldsymbol{D}_o)\boldsymbol{R}_N$ is diagonalizable and has real and positive eigenvalues.

It is known that the convergence of $\|\overline{U}_N(t)\|$ toward zero is governed by the smallest eigenvalue $\rho_{\min} = \lambda_{\min}\{(\boldsymbol{I}_N + \boldsymbol{D}_o)\boldsymbol{R}_N\}$. More precisely since $(\boldsymbol{I}_N + \boldsymbol{D}_o)\boldsymbol{R}_N$ is diagonalizable with real positive eigenvalues then there exist constants c_1, c_2 such that, for t large enough, we have

$$c_1 e^{-\rho_{\min} t} \le \|\overline{U}_N(t)\| \le c_2 e^{-\rho_{\min} t}. \quad (53)$$

Taking the logarithm in (53), dividing by t and taking the limit for $t \to \infty$, we have that the convergence rate is equal to

$$CR = \mu\rho_{\min}. \quad (54)$$

Substituting the estimate of μ defined in (12), we can easily obtain the desired relation. ■

REFERENCES

[1] A. Benallal and A. Gilloire, "A new method to stabilize fast RLS algorithms based on a first order model of the propagation of errors," in *Proc. ICASSP 88 Conf.*, New York, Apr. 1988, pp. 1373–1376.

[2] A. Benveniste, M. Metivier, and P. Priouret, *Adaptive Algorithms and Stochastic Approximations.* Berlin, Germany: Springer-Verlag, 1987.

[3] N. J. Bershad, "Comments on comparison of the convergence of two algorithms for adaptive FIR digital filters," *IEEE Trans. Acoust., Speech, Signal Processing,* vol. ASSP-33, pp. 1604–1606, 1985.

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

[5] J. A. Bucklew, T. G. Kurtz, and W. A. Sethares, "Weak convergence and local stability properties of fixed step size recursive algorithms," *IEEE Trans. Inform. Theory,* vol. IT-39, pp. 966–978, 1993.

[6] 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, pp. 1394–1402, Dec. 1983.

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

[8] S. Haykin, *Adaptive Filter Theory,* 2nd ed. Englewood Cliffs, NJ: Prentice Hall, 1991.

[9] L. Ljung, M. Morf, and D. Falconer, "Fast calculation of gain matrices for recursive estimation schemes," *Int. J. Contr.,* vol. 27, pp. 1–19, Jan. 1978.

[10] L. Ljung and T. Soderstrom, *Theory and Practice of Recursive Identification.* Cambridge, MA: MIT Press, 1983.

[11] I. E. Mazo, "On the independence theory of equalizer convergence," *Bell Syst. Tech. J.,* vol. 58, no. 5, pp. 963–993, May/June 1979.

[12] G. V. Moustakides, "Correcting the instability due to finite precision of the fast Kalman algorithms," *Signal Processing,* vol. 18, pp. 33–42, 1989.

[13] G. V. Moustakides and S. Theodoridis, "Fast Newton transversal filters—A new class of adaptive estimation algorithms," *IEEE Trans. Signal Processing,* vol. 39, pp. 2184–2193, Oct. 1991.

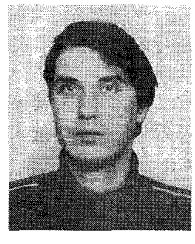
[14] T. Petillon, A. Gilloire, and S. Theodoridis, "The fast Newton transversal filter: An efficient scheme for acoustic echo cancellation in mobile radio," *IEEE Trans. Signal Processing,* vol. 42, pp. 509–518, Mar. 1993.

[15] W. A. Sethares, "The least mean square family," in *Adaptive System Identification and Signal Processing*, N. Kalouptsidis and S. Theodoridis, Eds. Englewood Cliffs, NJ: Prentice Hall, 1993.

[16] D. T. M. Slock and T. Kailath, "Numerically stable fast transversal filters for recursive least squares adaptive filtering," *IEEE Trans. Signal Processing,* vol. 39, pp. 92–114, Jan. 1991.

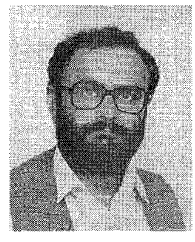
[17] S. Theodoridis, G. V. Moustakides, K. Berberidis, "A fast adaptive multichannel algorithm and application to decision feedback equalization," *IEEE Trans. Signal Processing,* vol. 43, pp. 327–331, Jan. 1995.

[18] B. Widrow and S. D. Stearns, *Adaptive Signal Processing.* Englewood Cliffs, NJ: Prentice Hall, 1985.



Panagiotis P. Mavridis was born in Orestiada, Greece, in 1967. He received the diploma in computer engineering and informatics from the University of Patras, Greece, in 1989, where he has been working toward the Ph.D. degree since 1990.

Since 1995 he has been with the Greek Army for the completion of his military service. His interests include fast algorithms for estimation and signal processing.



George V. Moustakides (M'92) was born in Drama, Greece, in 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 Recherche en Informatique et Systemes Aleatoires (IRISA-INRIA), Rennes, France, and from 1987 to 1990 a research position at the Computer Technology Institute (CTI), Patras, Patras, Greece. Since 1991 he has been an Associate Professor in the Department of Computer Engineering and Informatics, University of Patras, Patras, Greece. His interests include detection of signals, systems monitoring, adaptive estimation algorithms, and biomedical signal processing.