Extended Fast Fixed-Order RLS Adaptive Filters

Ricardo Merched, *Student Member, IEEE,* and Ali H. Sayed, *Fellow, IEEE*

***Abstract*—The existing derivations of conventional fast RLS adaptive filters are intrinsically dependent on the shift structure in the input regression vectors. This structure arises when a tapped-delay line (FIR) filter is used as a modeling filter. In this paper, we show, unlike what original derivations may suggest, that fast fixed-order RLS adaptive algorithms are not limited to FIR filter structures. We show that fast recursions in both explicit and array forms exist for more general data structures, such as orthonormally based models. One of the benefits of working with orthonormal bases is that fewer parameters can be used to model long impulse responses.**

***Index Terms*—Fixed-order filter, Laguerre network, orthonormal model, regularized least-squares, RLS algorithm.**

I. Introduction

FAST recursive least squares (RLS) adaptive filtering algorithms represent an attractive way to compute the least squares solution of growing length data efficiently. Whereas conventional RLS requires $\mathcal{O}(M^2)$ computations per sample, where M is the filter order, its fast versions require only $\mathcal{O}(M)$ operations. Examples of such fast schemes include the fast *a posteriori* error sequential technique (FAEST) [1], the fast transversal filter algorithm (FTF) [2], and least-squares lattice algorithms [5]–[7]. The latter class of algorithms deal with order-recursive structures, whereas the first two examples (FTF and FAEST) deal with fixed-order structures; both FTF and FAEST can also be viewed as special cases of a general fast estimation algorithm for state-space models known as the (extended) Chandrasekhar recursions [8], [9].

The low complexity that is achieved by these algorithms is a direct consequence of the shift structure that is characteristic of regression vectors in FIR adaptive implementations. This fact is evident in the conventional derivations of fast adaptive algorithms, all of which rely heavily on the shift structure. The arguments in [8] and [9], however, have shown that fast RLS algorithms can still be derived for certain more general structures in the regression vectors other than the shift structure. In this paper, we show that input regression vectors that arise from more general networks, such as orthonormal filters, satisfy the structural conditions required in [9], and, therefore, that fast fixed-order RLS algorithms can be derived in this context (in [10] and [11], we have shown that order-recursive (lattice) structures can also be developed for Laguerre networks).

There are several important reasons to consider orthonormal basis functions instead of the usual FIR implementations. First, the use of orthonormal models to describe the dynamical behavior of a wide class of systems has been studied extensively in many recent works on system identification and control [13]–[16]. Second, the orthonormality property of such structures offers many benefits in estimation problems, including better numerical conditioning of the data. Third, one of the primary motivations in using IIR basis functions for adaptive filtering is the fact that it requires fewer parameters to model systems with long impulse responses.

In echo cancellation applications, for example, a long FIR filter may be necessary to model the echo path, and adaptive IIR techniques have been proposed as possible alternatives (e.g., [17] and [18]). These techniques are nevertheless known to face stability problems due to the arbitrary pole locations during filter operation. Adaptive filters based on orthonormal models can offer an attractive alternative since, in this case, the location of the poles is fixed. Such schemes have already been suggested for echo cancellation and equalization applications [22], [23]. However, these earlier contributions rely on a slow RLS-type form for adaptive algorithm that requires $\mathcal{O}(M^2)$ operations.

In this paper, we show that fast $\mathcal{O}(M)$ fixed-order adaptive filters for general orthonormal models can be derived in the least-squares domain, thus leading to fast RLS Laguerre adaptive schemes. One of the advantages of using a least-squares-based adaptive scheme is that these tend to exhibit faster convergence rates and smaller misadjustments than gradient-based adaptive schemes.

II. RLS Algorithm

We start by reviewing the standard least-squares problem and its recursive solution. Given a column vector $y_N \in \mathbb{C}^{N+1}$ and a data matrix $H_N \in \mathbb{C}^{(N+1)\times M}$, the exponentially weighted least squares problem seeks the column vector $w \in \mathbb{C}^M$ that solves

$$\min_w [\lambda^{N+1} w^* \Pi^{-1} w + (y_N - H_N w)^* W_N (y_N - H_N w)] \quad (1)$$

where Π is a positive-definite regularization matrix, and

$$W_N = (\lambda^N \oplus \lambda^{N-1} \oplus \cdots \oplus \lambda \oplus 1)$$

is a weighting matrix that is defined in terms of a forgetting factor $\lambda, 0 \ll \lambda < 1$. The symbol * denotes complex conjugate transposition.

Manuscript received October 2, 2000; revised September 11, 2001. This work was supported in part by the National Science Foundation under Awards CCR-9732376 and ECS-9820765 and by UC Core Project CR 98-19. The work of R. Merched was also supported by a fellowship from CAPES, Brazil. The associate editor coordinating the review of this paper and approving it for publication was Dr. Xiang-Gen Xia.

The authors are with the Department of Electrical Engineering, University of California, Los Angeles, CA 90095 USA.

Publisher Item Identifier S 1053-587X(01)10485-X.

The individual entries of the measurement vector y_N will be denoted by $\{d(i)\}$, and the individual rows of the data matrix H_N will be denoted by $\{u_i\}$, i.e.,

$$y_N = \begin{bmatrix} d(0) \\ d(1) \\ \vdots \\ d(N) \end{bmatrix}, \quad H_N = \begin{bmatrix} u_0 \\ u_1 \\ \vdots \\ u_N \end{bmatrix}.$$

Let w_N denote the optimal solution of (1). It is given by

$$w_N \triangleq P_N H_N^* W_N y_N \tag{2}$$

in terms of the coefficient matrix

$$P_N = (\lambda^{N+1}\Pi^{-1} + H_N^* W_N H_N)^{-1}. \tag{3}$$

Let $\hat{y}_N$ denote the vector

$$\hat{y}_N \triangleq H_N w_N \tag{4}$$

which we refer to as the *regularized* projection (or simply projection) of y_N onto the range space of H_N, which is written as $\mathcal{R}(H_N)$.

Define further two estimation error vectors: the *a posteriori* error vector

$$e_N = y_N - H_N w_N$$

and the *a priori* error vector

$$\epsilon_N = y_N - H_N w_{N-1}$$

where w_{N-1} is the solution to a least-squares problem similar to (1) with data up to time $N-1$ (and with $\lambda^{N+1}\Pi^{-1}$ replaced by $\lambda^N\Pi^{-1}$)

$$\min_w [\lambda^N w^* \Pi^{-1} w + (y_{N-1} - H_{N-1}w)^* \times W_{N-1}(y_{N-1} - H_{N-1}w)]. \tag{5}$$

The minimum cost of (1) is denoted by $\xi(N)$, and it is given by[1]

$$\xi(N) = y_N^* W_N e_N. \tag{6}$$

The last entries of e_N and ϵ_N are called the *a posteriori* and *a priori* estimation errors at time N, and they are given by

$$e(N) = d(N) - u_N w_N, \quad \epsilon(N) = d(N) - u_N w_{N-1}.$$

These are related by a conversion factor, i.e.,

$$e(N) = \gamma(N)\epsilon(N) \tag{7}$$

where

$$\gamma(N) = 1 - u_N P_N u_N^*. \tag{8}$$

[1]We may note that in the absence of regularization $(\Pi^{-1} = 0)$, the expression for the minimum cost can also be expressed in the equivalent form $\xi(N) = e_N^* W_N e_N$. This follows from the orthogonality property $\hat{y}_N^* W_N e_N = 0$. However, when regularization is present, we need to use (6) instead. This fact distinguishes the derivations we will provide for the updates of the minimum costs of the so-called forward and backward prediction problems from those that assume no regularization. More on this later.

The well-known RLS algorithm allows us to update w_N recursively as follows:

$$w_{N+1} = w_N + g_{N+1}[d(N+1) - u_{N+1}w_N] \tag{9}$$
$$g_{N+1} = \lambda^{-1}P_N u_{N+1}^* \gamma(N+1) \tag{10}$$
$$\gamma^{-1}(N+1) = 1 + \lambda^{-1}u_{N+1}P_N u_{N+1}^* \tag{11}$$
$$P_{N+1} = \lambda^{-1}P_N - g_{N+1}\gamma^{-1}(N+1)g_{N+1}^* \tag{12}$$

where $w_{-1} = 0$, and $P_{-1} = \Pi$. It also holds that $g_{N+1} = P_{N+1}u_{N+1}^*$.[2]

The computation of the gain vector g_{N+1} in the above solution relies on the propagation of the (Riccati) variable P_N. This method of computation requires $\mathcal{O}(M^2)$ operations per iteration. Fast $\mathcal{O}(M)$ RLS schemes, on the other hand, avoid the propagation of P_N and evaluate the necessary gain vector in a more efficient manner. It turns out that the choice of Π plays a crucial role in the derivation of such fast schemes, as we now explain.

Assume for the time being that there exists a square matrix Ψ that relates two successive regression vectors as

$$u_N = u_{N+1}\Psi. \tag{13}$$

Using (10), we can then relate two successive gain vectors $\{g_N, g_{N+1}\}$ as follows. Define the normalized gain vector

$$k_N = g_N\gamma^{-1}(N). \tag{14}$$

Then

$$\Psi k_N = \lambda^{-1}\Psi P_{N-1}u_N^* = \lambda^{-1}\Psi P_{N-1}\Psi^* u_{N+1}^*.$$

Subtracting this equality from $k_{N+1} = \lambda^{-1}P_N u_{N+1}^*$, we find that

$$k_{N+1} = \Psi k_N + \lambda^{-1}(P_N - \Psi P_{N-1}\Psi^*)u_{N+1}^*. \tag{15}$$

This equation shows that in order to update the normalized gain vector from time N to time $N+1$, it is *not* necessary to evaluate the individual $\{P_N, P_{N-1}\}$ but only the difference

$$\nabla_{\{P_N,\Psi\}} = P_N - \Psi P_{N-1}\Psi^*. \tag{16}$$

It turns out that such differences can be updated quickly for certain choices of Π and Ψ (as shown in [8] and [9] in a more general state-space context), and this fact can be used to derive fast RLS schemes for input data vectors with or without shift structure, as we will explain.

III. Brief Overview of IIR and Orthonormal Basis Structures

The objective in this section is to motivate the use of orthonormal filter structures. Thus, consider a stable transfer function

$$G(z) = \frac{B(z)}{A(z)} = \frac{\sum_{k=0}^{Q} b_k z^{-k}}{1 - \sum_{k=1}^{P} a_k z^{-k}}$$

[2]We remark that without the factor λ^{N+1} in the cost (1), the above RLS recursions would not correspond to an exact least-squares solution.

and assume that we want to estimate $\{a_k, b_k\}$ from noisy observations of the output signal, say, $\{d(N)\}$, in response to an input sequence $\{s(N)\}$. Using the z-transform notation, we can write

$$o(z) = \frac{B(z)}{A(z)} s(z)$$

or, equivalently

$$o(N) = \sum_{k=1}^{P} a_k o(N-k) + \sum_{k=0}^{Q} b_k s(N-k)$$

so that the available measurements satisfy

$$d(N) = \sum_{k=1}^{P} a_k o(N-k) + \sum_{k=0}^{Q} b_k s(N-k) + v(N). \quad (17)$$

There are two common techniques for estimating the parameters $\{a_k, b_k\}$ from $\{d(N), s(N)\}$:

A. Equation-Error Method

Here, we replace the $\{o(N-k)\}$ on the right-hand side of (17) by $\{d(N-k) - v(N-k)\}$ so that

$$d(N) = \sum_{k=1}^{P} a_k d(N-k) + \sum_{k=0}^{Q} b_k s(N-k) + v'(N) \quad (18)$$

where $v'(N)$ is a colored noise that is obtained by filtering $v(N)$ through $A(z)$. If we ignore the dependency of the spectrum of $v'(N)$ on the $\{a_k\}$, then model (18) can be assumed to be *linear* in the parameters $\{a_k, b_k\}$, which can be estimated via any linear estimation procedure (e.g., LMS) to find

$$\hat{d}(N) = \sum_{k=1}^{P} \hat{a}_k d(N-k) + \sum_{k=0}^{Q} \hat{b}_k s(n-k). \quad (19)$$

A major problem with this formulation is that the estimates $\{\hat{a}_k, \hat{b}_k\}$ become biased due to the color of the noise $v'(N)$ and its dependency on $\{a_k\}$.

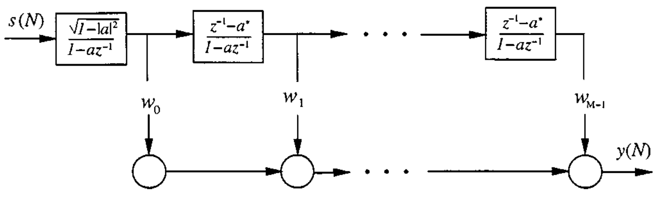
B. Output-Error Method

Here, we replace the $\{d(N-k)\}$ on the right-hand side of (19) by $\{\hat{d}(N-k)\}$ so that the estimate $\hat{d}(N)$ is now computed via

$$\hat{d}(N) = \sum_{k=1}^{P} \hat{a}_k \hat{d}(N-k) + \sum_{k=0}^{Q} \hat{b}_k s(N-k)$$

where the $\{\hat{a}_k, \hat{b}_k\}$ are still updated via an instantaneous-gradient algorithm. One of the main drawbacks of this formulation is the existence of multiple local minima in the corresponding mean-squared-error surface. This implies that an adaptive solution can be trapped before achieving "optimality." An alternative method to overcome the problem of local minima is to employ the so-called Steiglitz–McBride (SM) error formulation [19], which in many cases presents global convergence.

Nevertheless, regardless of which error formulation is used, there are still major disadvantages in considering an adaptive IIR solution for practical purposes. These include



s(N)
$\frac{\sqrt{1-|a|^2}}{1-az^{-1}}$
$\frac{z^{-1}-a^*}{1-az^{-1}}$
$\frac{z^{-1}-a^*}{1-az^{-1}}$
w_0
w_1
w_{M-1}
y(N)

Fig. 1. Transversal Laguerre network.

- the necessity to monitor the poles since they can become unstable during the process of convergence of the algorithm;
- the convergence can be very slow compared with an adaptive FIR implementation.

These two facts constitute a major impairment facing the employment of adaptive IIR filters in practice.

C. A Priori Information on System Dynamics

One way to overcome these difficulties is to choose a fixed IIR model structure of the form

$$y(z) = \left(\sum_{k=0}^{M-1} w_k \mathcal{B}_k(z) \right) s(z) \quad (20)$$

where

$\{w_k\}$	parameters to be estimated;
$\{\mathcal{B}_k\}$	rational transfer functions;
M	model order.

Note that in this formulation, the output signal does not depend on its past values. Moreover, in this case, the estimates $\{w_k\}$ will not be biased by measurement noise.

A useful modeling structure is the so-called (complex) Laguerre model, where the $\{\mathcal{B}_k\}$ are chosen as the orthonormal set

$$\mathcal{B}_k(z) = \left(\frac{\sqrt{1-|a|^2}}{1-az^{-1}} \right) \left(\frac{z^{-1}-a^*}{1-az^{-1}} \right)^k, \quad |a| < 1. \quad (21)$$

Fig. 1 shows the corresponding network. Laguerre functions have been studied extensively in recent works on system identification and control [13], [16] and have also been suggested for echo cancellation purposes [22].

In this paper, we consider an extention of the Laguerre model that allows to incorporate prior information on a *variety of poles*, say, $\{a_0, a_1, \ldots, a_{M-1}\}$. That is, we consider

$$\mathcal{B}_k(z) = \frac{\sqrt{1-|a_k|^2}}{1-a_k z^{-1}} \prod_{i=0}^{k-1} \frac{z^{-1}-a_i^*}{1-a_i z^{-1}}. \quad (22)$$

This orthonormal network is illustrated in Fig. 2. This model was proposed in [21] in the context of system identification, and it preserves orthonormality of the $\mathcal{B}_k(z)$. Note that the Laguerre model is obtained by setting all the poles to $a_k = a$. There are three main reasons for considering the use of such orthonormal structures in adaptive filtering.

1) When the system to be modeled has poles, an adaptive FIR filter can exibit poor performance in comparison with an adaptive structure based on rational functions.

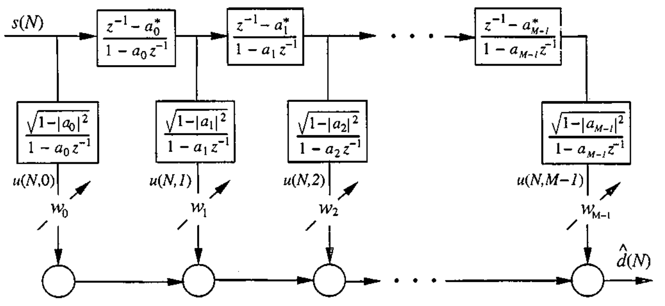

s(N)
$\frac{z^{-1}-a_0^*}{1-a_0z^{-1}}$
$\frac{z^{-1}-a_1^*}{1-a_1z^{-1}}$
$\frac{z^{-1}-a_{M-1}^*}{1-a_{M-1}z^{-1}}$
$\frac{\sqrt{1-|a_0|^2}}{1-a_0z^{-1}}$
$\frac{\sqrt{1-|a_1|^2}}{1-a_1z^{-1}}$
$\frac{\sqrt{1-|a_2|^2}}{1-a_2z^{-1}}$
$\frac{\sqrt{1-|a_{M-1}|^2}}{1-a_{M-1}z^{-1}}$
u(N,0)
u(N,1)
u(N,2)
u(N,M−1)
w_0
w_1
w_2
w_{M-1}
$\hat{d}(N)$

Fig. 2. Transversal orthonormal structure for adaptive filtering.

2) Unlike the conventional IIR adaptive methods, which present serious problems of stability, local minima, and slow convergence, the use of prior information offers a stable and global solution, due to fixed poles location.
3) Orthonormality guarantees good numerical conditioning for the underlying estimation problem, in contrast with other equivalent system descriptions. That is, note that one could have chosen to represent the transfer function $G(z)$ according to a partial-fraction description or a fixed-denominator model. However, such representations can be numerically ill-conditioned in comparison with the orthonormal structure. Moreover, the statistical properties of the regressors in a nonorthonormal model could lead to data covariance matrices with large eigenvalue spread, and the training of the coefficients $\{w_k\}$ could be adversely affected.

IV. Fast Array Algorithm for Orthonormal Filter Structures

We now discuss how a fast least-squares adaptive filter can be derived for trainning the structure of Fig. 2.

Thus, consider the orthonormal filter structure of Fig. 2 with transfer function [from $s(N)$ to $\hat{d}(N)$] given by

$$G(z) = \sum_{i=0}^{M-1} w_i \frac{\sqrt{1-|a_i|^2}}{1-a_iz^{-1}} \prod_{k=0}^{i} \frac{z^{-1}-a_k^*}{1-a_kz^{-1}}, \quad |a_k|<1. \tag{23}$$

The input to the orthonormal filter at time N is denoted by $s(N)$, and the coefficients that combine the outputs of the successive lowpass sections are denoted by $\{w_i\}$. Let

$$u_N \triangleq [u(N,0) \quad u(N,1) \quad \cdots \quad u(N,M-1)] \tag{24}$$

denote the regression vector at time N. Observe that the individual entries of u_N are not time-delayed versions of each other. However, as we now verify, two successive regression vectors $\{u_N, u_{N+1}\}$ satisfy a relation similar to (but not exactly of the same form as) (13) for some Ψ. (Later, we will show that we can handle the slight discrepancy in the relation by properly defining extended vectors.)

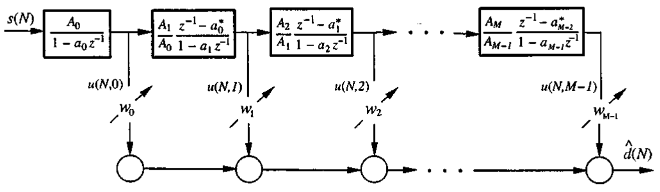

s(N)
$\frac{A_0}{1-a_0z^{-1}}$
$\frac{A_1}{A_0}\frac{z^{-1}-a_0^*}{1-a_1z^{-1}}$
$\frac{A_2}{A_1}\frac{z^{-1}-a_1^*}{1-a_2z^{-1}}$
$\frac{A_M}{A_{M-1}}\frac{z^{-1}-a_{M-2}^*}{1-a_{M-1}z^{-1}}$
u(N,0)
u(N,1)
u(N,2)
u(N,M−1)
w_0
w_1
w_2
w_{M-1}
$\hat{d}(N)$

Fig. 3. Equivalent orthonormal structure.

To see this, consider the equivalent orthonormal network depicted in Fig. 3. Then

$$\begin{aligned} u(N+1,0) &= a_0u(N,0) + A_0s(N) \\ u(N+1,i) &= a_iu(N,i) \\ &\quad + \frac{A_i}{A_{i-1}}[u(N,i-1) - a_{i-1}^*u(N+1,i-1)] \end{aligned} \tag{25}$$

where $A_i \triangleq \sqrt{1-|a_i|^2}$.

Using (25), we can easily relate all the entries of two successive regression vectors u_N and u_{N+1} by writing their corresponding difference equations (for example, the following relations hold for the first three entries of u_N):

$$\begin{aligned} u(N+1,0) &= a_0u(N,0) + A_0s(N) \\ u(N+1,1) &= -a_0^*\frac{A_1}{A_0}u(N+1,0) + \frac{A_1}{A_0}u(N,0) \\ &\quad + a_1u(N,1) \\ u(N+1,2) &= (a_0a_1)^*\frac{A_2}{A_0}u(N+1,0) \\ &\quad - a_1^*\frac{A_2}{A_0}u(N,0) - A_2A_1u(N,1) + a_2u(N,2). \end{aligned}$$

In matrix form, we can express these relations as

$$\begin{aligned} u_{M+1,N} &\triangleq [u(N+1,0) \quad u_N] = [u_{N+1} \quad u(N,M-1)]\Psi \\ &\triangleq \bar{u}_{M+1,N+1}\Psi \end{aligned} \tag{26}$$

where Ψ is the $(M+1)\times(M+1)$ matrix (say, for $M=5$)

$$\Psi = \begin{bmatrix} 1 & a_0^* & 0 & 0 & 0 & 0 \\ 0 & A_1A_0 & a_1^* & 0 & 0 & 0 \\ 0 & -a_1A_2A_0 & A_2A_1 & a_2^* & 0 & 0 \\ 0 & a_1a_2A_3A_0 & -a_2A_3A_1 & A_3A_2 & a_3^* & 0 \\ 0 & \frac{-a_1a_2a_3A_0}{A_4} & \frac{a_2a_3A_1}{A_4} & \frac{-a_3A_2}{A_4} & \frac{A_3}{A_4} & 0 \\ 0 & \frac{a_1a_2a_3a_4A_0}{A_4} & \frac{-a_2a_3a_4A_1}{A_4} & \frac{a_3a_4A_2}{A_4} & \frac{-a_4A_3}{A_4} & 1 \end{bmatrix} \tag{27}$$

and

$$\bar{u}_{M+1,N+1} = [u_{N+1} \quad u(N,M-1)].$$

Except for the additional terms $u(N+1,0)$ and $u(N,M-1)$, (26) is of the same form as (13). This slight difference in the nature of the relations can be handled by properly defining extended quantities.

Thus, note that using (10) and (14), we can write

$$\begin{bmatrix} k_{N+1} \\ 0 \end{bmatrix} = \lambda^{-1}\begin{bmatrix} P_N & 0 \\ 0 & 0 \end{bmatrix}\begin{bmatrix} u_{N+1}^* \\ u^*(N,M-1) \end{bmatrix}. \tag{28}$$

Likewise

$$\Psi\begin{bmatrix}0\\k_N\end{bmatrix}=\lambda^{-1}\Psi\begin{bmatrix}0&0\\0&P_{N-1}\end{bmatrix}\begin{bmatrix}u^*(N+1,0)\\u_N^*\end{bmatrix}$$
$$=\lambda^{-1}\Psi\begin{bmatrix}0&0\\0&P_{N-1}\end{bmatrix}\Psi^*\begin{bmatrix}u_{N+1}^*\\u^*(N,M-1)\end{bmatrix}\quad(29)$$

where the second equality follows from (26). Subtracting (28) from (29), we obtain

$$\begin{bmatrix}k_{N+1}\\0\end{bmatrix}-\Psi\begin{bmatrix}0\\k_N\end{bmatrix}=\frac{1}{\lambda}\left(\begin{bmatrix}P_N&0\\0&0\end{bmatrix}-\Psi\begin{bmatrix}0&0\\0&P_{N-1}\end{bmatrix}\Psi^*\right)\times\begin{bmatrix}u_{N+1}^*\\u^*(N,M-1)\end{bmatrix}.\quad(30)$$

This relation shows that in order to update the scaled gain vector from k_N to k_{N+1}, we only need to know how to update the difference

$$\nabla_{\{P_N,\Psi\}}\triangleq\begin{bmatrix}P_N&0\\0&0\end{bmatrix}-\Psi\begin{bmatrix}0&0\\0&P_{N-1}\end{bmatrix}\Psi^*.\quad(31)$$

We now verify that it is possible to update the differences $\nabla_{\{P_N,\Psi\}}$ efficiently for all N.

For this purpose, we first note that for the case of prewindowed input data [i.e., $s(N)=0$ for $N\le 0$ and zero initial conditions], we obtain at the initial time instant $N=0$

$$\nabla_{\{P_0,\Psi\}}=\begin{bmatrix}\lambda^{-1}\Pi&0\\0&0\end{bmatrix}-\Psi\begin{bmatrix}0&0\\0&\Pi\end{bmatrix}\Psi^*.\quad(32)$$

Assume that Π is chosen such that the above difference has low rank (say r, where r is independent of M—see Section V-C). We can then factor $\nabla_{\{P_0,\Psi\}}$ as

$$\nabla_{\{P_0,\Psi\}}=L_0JL_0^*\quad(33)$$

where L_0 is $(M+1)\times r$, and J is an $r\times r$ signature matrix with as many ± 1s as $\nabla_{\{P_0,\Psi\}}$ has positive or negative eigenvalues. We will show that when this holds, successive differences $\nabla_{\{P_N,\Psi\}}$ for $N>0$ will not exceed r and, more importantly, that the inertia of all these successive differences can also be taken as J. In other words, by forcing the initial difference (32) to have low rank and a certain inertia, we end up forcing all successive differences to have a similar property. This fact is essential to the derivation of a fast algorithm.

A. Fast Array Algorithm

To establish the above claims we proceed by induction. Assume that the difference $\nabla_{\{P_N,\Psi\}}$ at time N can be factored as $\nabla_{\{P_N,\Psi\}}=L_NJL_N^*$ for some $(M+1)\times r$ matrix L_N. Define further, for compactness of notation, the extended quantities

$$\tilde{k}_N^l\triangleq\begin{bmatrix}0\\g_N\gamma_N^{-1/2}(N)\end{bmatrix},\quad\tilde{k}_N^u\triangleq\begin{bmatrix}g_N\gamma_N^{-1/2}(N)\\0\end{bmatrix}$$
$$P_N^l\triangleq\begin{bmatrix}0&0\\0&P_N\end{bmatrix}\quad\text{and}\quad P_{N+1}^u\triangleq\begin{bmatrix}P_{N+1}&0\\0&0\end{bmatrix}.$$

Now, implement a $(1\oplus J)$-unitary transformation matrix Θ_N, i.e., Θ_N satisfies $\Theta_N(1\oplus J)\Theta_N^*=(1\oplus J)$ that transforms the following prearray to the form

$$\begin{bmatrix}\gamma^{-1/2}(N)&\frac{1}{\sqrt{\lambda}}\bar{u}_{N+1}L_N\\\Psi\tilde{k}_N^l&\frac{1}{\sqrt{\lambda}}L_N\end{bmatrix}\Theta_N=\begin{bmatrix}s&0\\m&C\end{bmatrix}\quad(34)$$

where

s	positive scalar;
m	column vector;
C	matrix.

Then, we claim that we can make the identifications

$$s=\gamma^{-1/2}(N+1),\quad m=\tilde{k}_{N+1}^u,\quad C=L_{N+1}$$

and show that $\nabla_{\{P_{N+1},\Psi\}}=CJC^*$. (This last equation means that the inertia of $\nabla_{\{P_{N+1},\Psi\}}$ can be taken as J and that the above array algorithm provides the desired low rank factor L_{N+1} as well.)

To determine the $\{s,m,C\}$ as above, we proceed as follows. Using the $(1\oplus J)$-unitarity of Θ_N, we obtain from (34) that the following equality holds:

$$\begin{bmatrix}\gamma^{-1/2}(N)&\frac{1}{\sqrt{\lambda}}\bar{u}_{N+1}L_N\\\Psi\tilde{k}_N^l&\frac{1}{\sqrt{\lambda}}L_N\end{bmatrix}\times\begin{bmatrix}1&\\&J\end{bmatrix}\begin{bmatrix}\gamma^{-1/2}(N)&\frac{1}{\sqrt{\lambda}}\bar{u}_{N+1}L_N\\\Psi\tilde{k}_N^l&\frac{1}{\sqrt{\lambda}}L_N\end{bmatrix}^*=\begin{bmatrix}s&0\\m&C\end{bmatrix}\begin{bmatrix}1&\\&J\end{bmatrix}\begin{bmatrix}s&0\\m&C\end{bmatrix}^*.\quad(35)$$

Equating the (1, 1) entries on both sides of this equality, we find that s should satisfy

$$\begin{aligned}|s|^2&=\gamma^{-1}(N)+\lambda^{-1}\bar{u}_{N+1}L_NJL_N^*\bar{u}_{N+1}^*\\&=\gamma^{-1}(N)+\lambda^{-1}\bar{u}_{N+1}\left(P_N^u-\Psi P_{N-1}^l\Psi^*\right)\bar{u}_{N+1}^*\\&=\gamma^{-1}(N)-\lambda^{-1}\bar{u}_NP_{N-1}^l\bar{u}_N^*+\lambda^{-1}\bar{u}_{N+1}P_N^u\bar{u}_{N+1}^*\\&\overset{(11)}{=}1+\lambda^{-1}\bar{u}_{N+1}P_N^u\bar{u}_{N+1}^*\\&=\gamma^{-1}(N+1).\end{aligned}\quad(36)$$

This allows us to identify s as $s=\gamma^{-1/2}(N+1)$. Likewise, equating the (2, 1) block entries on both sides of (34), we obtain

$$\begin{aligned}s^*m&=\Psi\tilde{k}_N^l\gamma^{-1/2}(N)+\lambda^{-1}L_NJL_N^*\bar{u}_{N+1}^*\\&=\Psi\begin{bmatrix}0\\k_N\end{bmatrix}+\lambda^{-1}\left(P_N^u-\Psi P_{N-1}^l\Psi^*\right)\bar{u}_{N+1}^*.\end{aligned}$$

Comparing with (30) and using the value of s, we conclude that we can make the identification $m=\tilde{k}_{N+1}^u$.

Finally, equating the (2, 2) entries on both sides of (35), we find that C should satisfy

$$\begin{aligned}CJC^*&\\&=\Psi\tilde{k}_N^l\tilde{k}_N^*\Psi^*+\lambda^{-1}L_NJL_N^*-mm^*\\&=\Psi\tilde{k}_N^l\tilde{k}_N^{l*}\Psi^*+\lambda^{-1}L_NJL_N^*-\tilde{k}_{N+1}^u\tilde{k}_{N+1}^{u*}\\&=\Psi\tilde{k}_N^l\tilde{k}_N^{l*}\Psi^*+\lambda^{-1}\left(P_N^u-\Psi P_{N-1}^l\Psi^*\right)-\tilde{k}_{N+1}^u\tilde{k}_{N+1}^{u*}\\&=\left[\lambda^{-1}P_N^u-\tilde{k}_{N+1}^u\tilde{k}_{N+1}^{u*}\right]-\Psi\left[\lambda^{-1}P_{N-1}^l-\tilde{k}_N^l\tilde{k}_N^{l*}\right]\Psi^*\\&=P_{N+1}^u-\Psi P_N^l\Psi^*\end{aligned}\quad(37)$$

which allows us to identify C as L_{N+1} and to take the inertia of the difference $\nabla_{\{P_{N+1},\Psi\}}$ as J. The resulting fast algorithm can be summarized as follows.

(Fast Array RLS) *Consider input regression vectors u_N arising from the orthonormal filter structure of Fig. 2 as in (24). The solution to the minimization problem (1) can be recursively computed as follows. Start with $w_{-1} = 0, \gamma^{-1/2}(0) = 1, k_0 = 0, L_0$ and J from the factorization (32) and (33), and repeat for each $N > 0$*

$$\begin{bmatrix} \gamma^{-1/2}(N) & \frac{1}{\sqrt{\lambda}}[u_{N+1} \quad u(N, M-1)]L_N \\ \Psi\begin{bmatrix} 0 \\ g_N\gamma^{-1/2}(N)\end{bmatrix} & \frac{1}{\sqrt{\lambda}}L_N \end{bmatrix}\Theta_N = \begin{bmatrix} \gamma^{-1/2}(N+1) & 0 \\ \begin{bmatrix} g_{N+1}\gamma^{-1/2}(N+1) \\ 0 \end{bmatrix} & L_{N+1}\end{bmatrix}$$

where Θ_N is a $(1 \ominus J)$-unitary matrix that produces the zero entries in the above post-array, and L_N is $(M+1)\times r$. The transformation matrix Θ_N and the multiplication by the matrix Ψ defined in (27) can be implemented according to the procedure described in Sections IV-B and VI. Moreover

$$w_{N+1} = w_N + g_{N+1}[d(N+1) - u_{N+1}w_N].$$

In Section V-C, we discuss the choice of Π and the values of L_0 and J. More specifically Π^{-1} is chosen as the unique positive-definite solution of the Lyapunov equation

$$\lambda\Pi^{-1} - \bar{T}\Pi^{-1}\bar{T}^* = \mu\bar{v}\bar{v}^*$$

as defined later in (77). Then, as described by (79)–(83) and (85)

$$L_0 = \begin{bmatrix} 1 & -q_0 \\ 0 & 1\end{bmatrix}\begin{bmatrix}\sqrt{\lambda/\mu} & \\ & \frac{1}{\sqrt{\varsigma_M^b(0)}}\end{bmatrix}$$

and $J = (1 \ominus -1)$.

Note that when $a = 0$, we have $\Psi = I$, and the algorithm collapses to the fast array RLS algorithm for the usual FIR input data structure; see, e.g., [26] and [27]. In Section VI, we will further exploit the structure of Ψ to show that the matrix product $\Psi[\begin{smallmatrix} 0 \\ g_N\gamma^{-1/2}(N)\end{smallmatrix}]$ can be obtained efficiently in $\mathcal{O}(2M)$ operations.

B. Some Implementation Issues

Although, from a theoretical point of view, any $(1 \oplus J)$-unitary matrix Θ_N that produces the zero entries in the first row of the post-array in (34) will do, we have noticed that different implementations lead to different numerical behavior. To see this, consider for simplicity the case $M = 3$ and $J = (1 \oplus -1)$. Then, the pre- and post-arrays will be of the generic forms

$$\begin{bmatrix} \times & \times & \times \\ \times & \times & \times \\ \times & \times & \times \\ \times & \times & \times \\ \times & \times & \times \end{bmatrix}\Theta = \begin{bmatrix} \times & 0 & 0 \\ \times & \times & \times \\ \times & \times & \times \\ \times & \times & \times \\ 0 & \times & \times \end{bmatrix}.$$

In order to create the zero pattern in the first row of the post-array, the $(1 \oplus J)$-unitary rotation Θ can be constructed based solely on the first row of the pre array, which means that only the information that is needed to update $\gamma^{-1}(N)$ to $\gamma^{-1}(N+1)$ is required to determine Θ. In other words, no information from the other equations is used to update the rest of the entries of the array. We have observed in simulations (see Section VIII) that even for $\lambda = 1$, this type of construction can cause the algorithm to diverge in finite precision.

To improve the numerical behavior of the array algorithm, we propose to construct Θ as follows. First, create a zero entry in the first row of the post-array by means of a circular (Givens) rotation, using the entries (0, 0) and (0, 1) of the pre-array (as indicated by the arrows)

$$\begin{array}{c} \downarrow \quad \downarrow \qquad\quad \\ \begin{bmatrix} \otimes & \otimes & \times \\ \times & \times & \times \\ \times & \times & \times \\ \times & \times & \times \\ \times & \times & \times \end{bmatrix} \end{array} \xrightarrow{\text{Givens}} \begin{bmatrix} \times' & \boxed{0} & \times \\ \times' & \times' & \times \\ \times' & \times' & \times \\ \times' & \times' & \times \\ \times' & \times' & \times \end{bmatrix}.$$

Now, note that the additional hyperbolic rotation that is needed to zero out the remaining entry in the first row of the post-array also results in a zero entry in the $(M+1, 0)$ position of the post-array. Rather than determining this hyperbolic rotation by using the entries (0, 0) and (0, 2) of the *first* row of the pre-array, we propose instead to determine the hyperbolic rotation by using the entries $(M+1, 0)$ and $(M+1, 2)$ of the *last* row of the pre-array. That is

$$\begin{array}{c} \begin{bmatrix} \times & \times & \times \\ \times & \times & \times \\ \times & \times & \times \\ \times & \times & \times \\ \otimes & \times & \otimes \end{bmatrix} \\ \uparrow \qquad\quad \uparrow \end{array} \xrightarrow{\text{Hyperbolic}} \begin{bmatrix} \times & 0 & 0 \\ \times & \times & \times \\ \times & \times & \times \\ \times & \times & \times \\ \boxed{0} & \times & \times \end{bmatrix}.$$

This choice seems to be more reasonable since in this case, the rotation matrix Θ is determined by using all the equations that constitute the algorithm. We have verified by simulations (in Matlab precision) that this method of constructing Θ is more reliable in terms of numerical errors (see Fig. 11).

It is important to clarify that the method or even the type of rotations used in these recursions are not entirely responsable for the numerical behavior of the fast RLS algorithm. In Section VII, we will invoke the concepts of *backward consistency* and *minimality* and comment on some stability issues for such fast RLS recursions, as done in [28]–[30].

V. Extended Fast Transversal Filter

The recursions of Section IV provide a fast algorithm in array form. Its cost is $\mathcal{O}(10M)$ per iteration plus the additional cost for implementing the rotations. An alternative description, which is often more efficient and relies on a set of explicit equations, can be given. This description is obtained by employing an alternative factorization for (31) that is motivated by introducing the so-called forward and backward prediction problems. These problems will further allow us to provide an interpretation for the columns of L_N in terms of forward and backward prediction filters.

Before proceeding, we should remark that since, in the remainder of this paper, we need to deal with order-recursive relations, it becomes important to explicitly indicate the size of all quantities involved (in addition to a time index). For example, we will write $w_{M,N}$ instead of w_N to indicate that it is a vector of order M that is computed by using data up to time N. We will also write $H_{M,N}$ instead of H_N to indicate that it is a matrix with row vectors of size M and with data up to time N so that problem (1) becomes

$$\min_{w}\left[\lambda^{N+1}w_M^*\Pi_M^{-1}w_M+(y_N-H_{M,N}w_M)^*\times W_N(y_N-H_{M,N}w_M)\right]$$

and its solution is $w_{M,N}$. In a similar vein, we write

$$\{\hat{y}_{M,N},e_{M,N},\epsilon_{M,N},e_M(N),\epsilon_M(N),\gamma_M(N),\xi_M(N)\}.$$

A. Forward Estimation Problem

Consider the input data matrix

$$H_{M,N}=\begin{bmatrix}u(0,0)&u(0,1)&\cdots&u(0,M-1)\\u(1,0)&u(1,1)&\cdots&u(1,M-1)\\u(2,0)&u(2,1)&\cdots&u(2,M-1)\\\vdots&\vdots&\ddots&\vdots\\u(N,0)&u(N,1)&\cdots&u(N,M-1)\end{bmatrix}$$

whose coefficient matrix is defined by

$$P_{M,N}^{-1}=\left(\lambda^{N+1}\Pi_M^{-1}+H_{M,N}^*W_NH_{M,N}\right).$$

Now, suppose that one more column is appended to $H_{M,N}$ from the left, i.e.,

$$\check{H}_{M+1,N}=[x_{0,N}\quad H_{M,N}] \tag{38}$$

and let

$$\check{P}_{M+1,N}^{-1}=\left(\lambda^{N+1}\check{\Pi}_{M+1}^{-1}+\check{H}_{M+1,N}^*W_N\check{H}_{M+1,N}\right)$$

where Π_M and $\check{\Pi}_{M+1}$ are assumed to be related via

$$\check{\Pi}_{M+1}^{-1}=\left(\mu\oplus\Pi_M^{-1}\right). \tag{39}$$

Then, it is straightforward to verify that

$$\check{P}_{M+1,N}^{-1}=\begin{bmatrix}\mu\lambda^{N+1}+x_{0,N}^*W_Nx_{0,N}&x_{0,N}^*W_NH_{M,N}\\H_{M,N}^*W_Nx_{0,N}&P_{M,N}^{-1}\end{bmatrix}.$$

Inverting, we obtain

$$\check{P}_{M+1,N}=\begin{bmatrix}0&0\\0&P_{M,N}\end{bmatrix}+\frac{1}{\zeta_M^f(N)}\times\begin{bmatrix}1\\-w_{M,N}^f\end{bmatrix}\begin{bmatrix}1&-w_{M,N}^{f*}\end{bmatrix} \tag{40}$$

where $w_{M,N}^f=P_{M,N}H_{M,N}^*W_Nx_{0,N}$ is the solution to the regularized forward prediction least-squares problem

$$\min_{w_M^f}\left[\lambda^{N+1}w_M^{f*}\Pi_M^{-1}w_M^f+\left(x_{0,N}-H_{M,N}w_M^f\right)^*\times W_N\left(x_{0,N}-H_{M,N}w_M^f\right)\right]. \tag{41}$$

This problem projects $x_{0,N}$ onto $\mathcal{R}(H_{M,N})$ in a regularized manner. Let

$$f_{M,N}=x_{0,N}-H_{M,N}w_{M,N}^f$$

denote the resulting (forward) estimation error vector. The quantity $\zeta_M^f(N)$ is defined as

$$\zeta_M^f(N)\triangleq\mu\lambda^{N+1}+\xi_M^f(N) \tag{42}$$

where $\xi_M^f(N)$ is the resulting minimum cost (cf. (6))

$$\xi_M^f(N)=x_{0,N}^*W_Nf_{M,N}.$$

Of course, the optimal solution of the forward prediction problem (41) can be updated recursively via an RLS algorithm of the form

$$w_{M,N}^f=w_{M,N-1}^f+k_{M,N}f_M(N) \tag{43}$$

$$g_{M,N}=\lambda^{-1}P_{M,N-1}u_{M,N}^*\gamma_M(N) \tag{44}$$

$$\gamma_M^{-1}(N)=1+\lambda^{-1}u_{M,N}P_{M,N-1}u_{M,N}^* \tag{45}$$

$$P_{M,N}=\lambda^{-1}P_{M,N-1}-g_{M,N}\gamma_M^{-1}(N)g_{M,N}^* \tag{46}$$

where $k_{M,N}=g_{M,N}\gamma_M^{-1}(N)$ and $f_M(N)$ is the last entry of $f_{M,N}$.

Substituting (40) into (44), it is immediate to see that we obtain the order update

$$\check{k}_{M+1,N}=\begin{bmatrix}0\\k_{M,N}\end{bmatrix}+\frac{\alpha_M^*(N)}{\lambda\zeta_M^f(N-1)}\begin{bmatrix}1\\-w_{M,N-1}^f\end{bmatrix} \tag{47}$$

where $\alpha_M(N)$ is the *a priori* forward prediction error defined via

$$f_M(N)=\alpha_M(N)\gamma_M(N)$$

where $f_M(N)$ is the last entry of $f_{M,N}$.

A similar order-update can be obtained for $\gamma_M(N)$ by substituting (40) into (45):

$$\check{\gamma}_{M+1}(N)=\gamma_M(N)-\frac{|f_M(N)|^2}{\zeta_M^f(N)}. \tag{48}$$

Note that (47) and (48) require the computation of the $\zeta_M^f(N)$, which admits the well-known recursion

$$\boxed{\zeta_M^f(N)=\lambda\zeta_M^f(N-1)+\alpha_M^*(N)f_M(N)} \tag{49}$$

Combining this recursion with (48), we obtain an alternative order update for $\gamma_M(N)$

$$\check{\gamma}_{M+1}(N)=\gamma_M(N)\frac{\lambda\zeta_M^f(N-1)}{\zeta_M^f(N)}. \tag{50}$$

B. Backward Estimation Problem

Similarly to the forward estimation problem, assume now that one more column is appended to $H_{M,N}$ from the right, i.e.,

$$\bar{H}_{M+1,N}=[H_{M,N}\quad x_{M,N}]. \tag{51}$$

Define the corresponding coefficient matrix

$$\bar{P}_{M+1,N}^{-1}=\left(\lambda^{N+1}\bar{\Pi}_{M+1}^{-1}+\bar{H}_{M+1,N}^*W_N\bar{H}_{M+1,N}\right)$$

where Π_M and $\bar{\Pi}_M$ are assumed to be related by

$$\bar{\Pi}_{M+1}^{-1} = \begin{bmatrix} \Pi_M^{-1} & c \\ c^* & \delta \end{bmatrix} \tag{52}$$

for some row vector c and scalar δ to be specified. Then, using (51), we have the equation shown at the bottom of the page. Again, inverting both sides, we get

$$\bar{P}_{M+1,N} = \begin{bmatrix} P_{M,N} & 0 \\ 0 & 0 \end{bmatrix} + \frac{1}{\zeta_M^b(N)} \begin{bmatrix} -q_N \\ 1 \end{bmatrix} [-q_N^* \quad 1]. \tag{53}$$

This equation has two main differences with respect to (40). First, the vector q_N is the sum of two quantities

$$q_N = w_{M,N}^b + t_N \tag{54}$$

where t_N is given by

$$t_N = \lambda^{N+1} P_{M,N} c \tag{55}$$

and $w_{M,N}^b$ is the solution to the (backward) least-squares problem

$$\min_{w_M^b} \left[\lambda^{N+1} w_M^{b*} \Pi_M^{-1} w_M^b + \left(x_{M,N} - H_{M,N} w_M^b\right)^* \times W_N \left(x_{M,N} - H_{M,N} w_M^b\right) \right]. \tag{56}$$

The resulting minimum cost is

$$\xi_M^b(N) = x_{M,N}^* W_N b_{M,N}.$$

Problem (56) projects $x_{M,N}$ onto $\mathcal{R}(H_{M,N})$, and the resulting (backward) estimation error vector is given by

$$b_{M,N} = x_{M,N} - H_{M,N} w_{M,N}^b.$$

Note that $w_{M,N}^b$ can be updated via a standard RLS algorithm via

$$w_{M,N}^b = w_{M,N-1}^b + k_{M,N} b_M(N) \tag{57}$$

$$g_{M,N} = \lambda^{-1} P_{M,N-1} u_{M,N}^* \gamma_M(N) \tag{58}$$

$$\gamma_M^{-1}(N) = 1 + \lambda^{-1} u_{M,N} P_{M,N-1} u_{M,N}^* \tag{59}$$

$$P_{M,N} = \lambda^{-1} P_{M,N-1} - g_{M,N} \gamma_M^{-1}(N) g_{M,N}^* \tag{60}$$

with an associated *a priori* backward prediction error

$$\beta_M(N) = x_M(N) - u_{M,N} w_{M,N-1}^b.$$

Substituting (60) into (55), we obtain a recursive relation for t_N (which is analogous to the time-update for $w_{M,N}^b$)

$$t_N = t_{N-1} - g_{M,N}(u_N t_{N-1}).$$

This equation, combined with the time update (57) for $w_{M,N}^b$, implies the following time-update for q_N:

$$q_N = q_{N-1} + \varrho_M(N) k_{M,N}$$

where $\varrho_M(N) = \varepsilon_M(N)\gamma_M(N)$, and

$$\varepsilon_M(N) = \beta_M(N) - u_{M,N} t_{N-1}.$$

In addition, the quantity $\zeta_M^b(N)$ in (53) is defined by

$$\zeta_M^b(N) \triangleq \xi_M^b(N) + \lambda^{N+1}\left(\delta - c^* q_N - w_{M,N}^{b*} c\right).$$

It is easy to obtain a time-update for this term. To see this, let us first define the quantities

$$\varsigma(N) \triangleq \xi_M^b(N) + \lambda^{N+1}\delta \tag{61}$$

$$p(N) \triangleq \lambda^{N+1} c^* q_N \tag{62}$$

$$r(N) \triangleq \lambda^{N+1} w_{M,N}^{b*} c \tag{63}$$

so that

$$\zeta_M^b(N) \triangleq \varsigma(N) - p(N) - r(N).$$

The first term on the right-hand side of the above equation can be updated similarly to $\xi_M^b(N)$

$$\varsigma(N) = \lambda\varsigma(N-1) + \beta_M(N) b_M(N). \tag{64}$$

Now, multiplying the time updates for $w_{M,N}^b$ and q_N by c^* from the left, the following equations can be obtained:

$$p(N) = \lambda p(N-1) + (u_{M,N} t_{N-1})^* \varrho_M(N) \tag{65}$$

$$r(N) = \lambda r(N-1) + b_M^*(N)(u_{M,N} t_{N-1}). \tag{66}$$

Adding (64)–(66), we get, after combining terms

$$\boxed{\zeta_M^b(N) = \lambda \zeta_M^b(N-1) + \varepsilon_M^*(N) \varrho_M(N)} \tag{67}$$

Observe that the variables $\{\zeta_M^b(N), \zeta_M^f(N)\}$ *do not* correspond to the exact values of the minimum costs for the backward and forward prediction problems (41) and (56). Only when $\lambda < 1$ and $N \to \infty$, they tend to coincide with the actual values $\{\xi_M^b(N), \xi_M^f(N)\}$.

Now, multiplying (53) from the right by $\bar{u}_{M+1,N+1}^*$, we obtain, similar to the forward estimation problem

$$\begin{bmatrix} k_{M,N} \\ 0 \end{bmatrix} = \bar{k}_{M+1,N} - \nu_M(N) \begin{bmatrix} -w_{M,N-1}^b \\ 1 \end{bmatrix} \tag{68}$$

where $\nu_M(N) = \varepsilon_M(N)/\lambda\zeta_M^b(N-1)$. The quantity $\nu_M(N)$ is usually referred to as the *rescue variable* and can be directly obtained as the last entry of $\bar{k}_{M+1,N}$ (to be computed further ahead).

Proceeding similarly to the derivation of (48), we also obtain

$$\bar{\gamma}_{M+1}(N) = \gamma_M(N) - \frac{|\varrho_M(N)|^2}{\zeta_M^b(N)}. \tag{69}$$

$$P_{M+1,N}^{-1} = \begin{bmatrix} P_{M,N}^{-1} & H_{M,N}^* W_N x_{M,N} + \lambda^{N+1} c \\ x_{M,N}^* W_N H_{M,N} + \lambda^{N+1} c^* & \delta\lambda^{N+1} + x_{M,N}^* W_N x_{M,N} \end{bmatrix}$$

Combining (67) and (69), we arrive at the following order update for $\gamma_M(N)$

$$\gamma_M(N) = \frac{\bar{\gamma}_{M+1}(N)}{1 - \bar{\gamma}_{M+1}(N)\varepsilon_M(N)\nu_M(N)}. \tag{70}$$

Note that the variables $\varepsilon_M(N)$ and $\varrho_M(N)$ play roles similar to the *a priori* and *a posteriori* backward prediction problems. However, although all the quantities related to the backward prediction problems satisfy identical recursive equations, here, they have different interpretations.

C. Exploiting Data Structure

We remarked earlier in Section II that a fast algorithm would update the gain vector k_N to k_{N+1} without relying on the explicit update of P_N to P_{N+1}. In (47), we showed how to order update $k_{M,N}$ to $\check{k}_{M+1,N}$, and using (68), we know how to order downdate $\bar{k}_{M+1,N+1}$ to $k_{M,N+1}$. Thus, in order to obtain a direct update from $k_{M,N}$ to $k_{M,N+1}$, we still need to know how to relate $\check{k}_{M+1,N}$ and $\bar{k}_{M+1,N+1}$. It turns out that in order to relate these two variables, we need to call on the structure of the regressors. Thus, consider again the extended quantities defined in (26)

$$\begin{aligned} u_{M+1,N} &= [u(N+1,0) \quad u_{M,N}] \\ \bar{u}_{M+1,N+1} &= [u_{M+1,N+1} \quad u(N,M-1)] \end{aligned}$$

and let Ψ denote any invertible matrix that relates them

$$\bar{u}_{M+1,N+1}\Psi = u_{M+1,N}.$$

That is, from this relation, it follows that

$$\bar{H}_{M+1,N+1} = \begin{bmatrix} 0 \\ \check{H}_{M+1,N} \end{bmatrix} \Psi^{-1}$$

where

$$\check{H}_{M+1,N} = [x_{0,N} \quad H_{M,N}]$$

and

$$\bar{H}_{M+1,N+1} = [H_{M,N+1} \quad x_{M,N}].$$

We then get

$$\begin{aligned} &\bar{P}_{M+1,N+1} \\ &\quad = \left(\lambda^{N+2}\bar{\Pi}_{M+1}^{-1} + \bar{H}_{M+1,N+1}^{*}W_{N+1}\bar{H}_{M+1,N+1}\right)^{-1} \\ &\quad = \left(\lambda^{N+2}\bar{\Pi}_{M+1}^{-1} + \Psi^{-*}\check{H}_{M+1,N}^{*}W_N\check{H}_{M+1,N}\Psi^{-1}\right)^{-1}. \end{aligned}$$

Note that if we could choose $\bar{\Pi}_{M+1}$ as

$$\bar{\Pi}_{M+1}^{-1} = \lambda^{-1}\Psi^{-*}\check{\Pi}_{M+1}^{-1}\Psi^{-1} \tag{71}$$

we then obtain a simple relation between $\{\bar{P}_{M+1,N+1}, P_{M+1,N}\}$

$$\boxed{\bar{P}_{M+1,N+1} = \Psi\check{P}_{M+1,N}\Psi^{*}} \tag{72}$$

From (72), we can obtain similar relations between the quantities $\{g_{M+1,N}, \bar{g}_{M+1,N+1}\}$ and $\{\gamma_{M+1}(N), \bar{\gamma}_{M+1}(N+1)\}$. Thus, multiplying both sides of (72) by $\bar{u}_{M,N+1}^{*}$ from the right, we get

$$\bar{g}_{M+1,N+1} = \Psi\check{g}_{M+1,N}. \tag{73}$$

If we further multiply (73) by $\bar{u}_{M,N+1}$ from the left and subtract 1 from both sides, we find

$$\bar{\gamma}_{M+1}(N+1) = \check{\gamma}_{M+1}(N) \tag{74}$$

which implies that

$$\boxed{\bar{k}_{M+1,N+1} = \Psi\check{k}_{M+1,N}} \tag{75}$$

as desired.

In order for (72) to hold, we still need to show how to choose Π_M, c, and δ in (52) in order to satisfy (71). Substituting (52) and (39) into (71), we get

$$\begin{bmatrix} \Pi_M^{-1} & c \\ c^* & \delta \end{bmatrix} = \lambda^{-1}\Psi^{-*}\begin{bmatrix} \mu & 0 \\ 0 & \Pi_M^{-1} \end{bmatrix}\Psi^{-1}. \tag{76}$$

Now, consider the matrix Ψ in (26) and (27) and partition Ψ^{-*} as

$$\Psi^{-*} = \begin{bmatrix} \bar{v} & \bar{T} \\ 0 & m \end{bmatrix}$$

(see the structure of Ψ^{-*} in Section VI), where we have the equations shown at the bottom of the next page. Expanding (76), we find that Π_M^{-1} should satisfy

$$\lambda\Pi_M^{-1} - \bar{T}\Pi_M^{-1}\bar{T}^* = \mu\bar{v}\bar{v}^*. \tag{77}$$

Therefore, if $|a_k| < \sqrt{\lambda}$, this Lyapunov equation admits a unique positive definite solution Π_M. This is because all the eigenvalues of $\bar{T}$ are either a_k^* or 0, and the pair $(\lambda^{-1/2}\bar{T}, \bar{v})$ is controllable. From (76), we then obtain

$$\begin{aligned} c &= \lambda^{-1}\bar{T}\Pi_M^{-1}m^* \\ \delta &= \lambda^{-1}m\Pi_M^{-1}m^* = \lambda^{-1}[\Pi^{-1}]_{M-1,M-1}. \end{aligned} \tag{78}$$

Substituting the above expressions into the definitions of the initial quantities

$$\left\{w_0^f, q_{-1}, k_{M,0}, \xi_M^b(0), \xi_M^f(-1)\right\}$$

we get the following initial conditions:

$$\zeta_M^f(-1) = \mu/\lambda \tag{79}$$

$$\zeta_M^b(0) = \lambda^{-1}[\Pi^{-1}]_{M-1,M-1} - c^*\Pi c \tag{80}$$

$$w_{M,0} = w_{M,-1}^f = k_{M,-1} = 0 \tag{81}$$

$$q_0 = \Pi c \tag{82}$$

$$\gamma_M(0) = 1. \tag{83}$$

In summary, the time update of the gain vector $k_{M,N}$, which is necessary to update the optimal solution

$$w_{M,N+1} = w_{M,N} + k_{M,N+1}e_M(N+1)$$

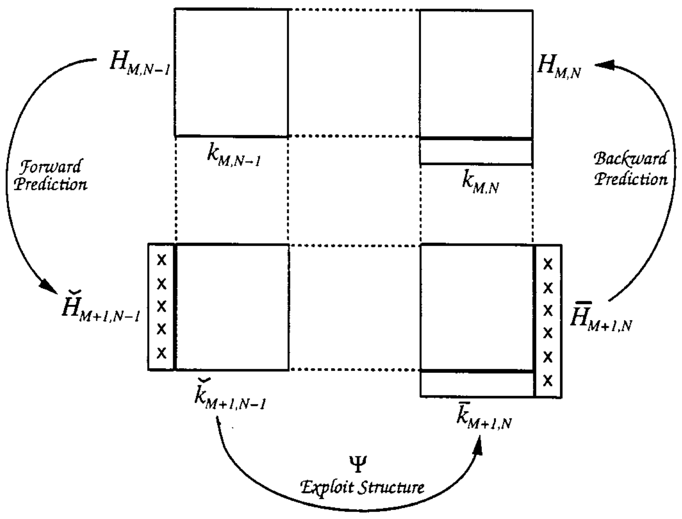


Fig. 4. Schematic procedure for computing the normalized gain vector.

can be efficiently performed in three steps:

1) order update $k_{M,N} \to \check{k}_{M+1,N}$ [see (47)];
2) time update $\check{k}_{M+1,N} \to \bar{k}_{M+1,N+1}$ [see (75)].
3) order downdate $\bar{k}_{M+1,N+1} \to k_{M,N+1}$ [see (68)].

Fig. 4 illustrates the above procedure for updating the normalized gain vector.

Note that when $a_k = 0$, we have $\Psi = I$, and therefore, $\bar{k}_{M+1,N+1} = k_{M+1,N}$, in which case, the recursions collapse to the FTF algorithm [2]. Moreover, (75) is the only recursion that uses the fact that the input data has structure. Table I lists the resulting algorithm.

Remark: The above results also provide us with an initial rank-two factorization for the array algorithm of Section IV. To see this, substitute the expressions (53) and (31) for $\{\check{P}_{M+1,N}, \bar{P}_{M+1,N+1}\}$ into (31) to get

$$\begin{bmatrix} P_N & 0 \\ 0 & 0 \end{bmatrix} - \Psi \begin{bmatrix} 0 & 0 \\ 0 & P_{N-1} \end{bmatrix} \Psi^* = L_N J L_N^* \tag{84}$$

where

$$L_N = \begin{bmatrix} \Psi \begin{bmatrix} 1 \\ -w^f_{M,N-1} \end{bmatrix} & \begin{matrix} -q_N \\ 1 \end{matrix} \end{bmatrix} \begin{bmatrix} \frac{1}{\zeta_M^{f/2}(N-1)} & \\ & \frac{1}{\zeta_M^{b/2}(N)} \end{bmatrix}. \tag{85}$$

This expression provides an interpretation for the columns of L_N in terms of the filters $\{w^f_{M,N-1}, q_N\}$.

TABLE I
EXTENDED FAST ADAPTIVE FILTER FOR ORTHONORMAL STRUCTURES

Initialization

μ is a small positive number; Π is the solution to (77);
c is given by (78).

$$\zeta^f_M(-1) = \mu/\lambda$$
$$\zeta^b_M(0) = \lambda^{-1}[\Pi^{-1}]_{M-1,M-1} - c^*\Pi c$$
$$w_{M,0} = w^f_{M,-1} = k_{M,-1} = 0$$
$$q_0 = \Pi c$$
$$\gamma_M(0) = 1$$

<u>*For $N \geq 0$, repeat*:</u>

$$u(N) = a_0 u(N-1) + \sqrt{1-|a_0|^2}\, s(N)$$

$$\alpha_M(N-1) = u(N) - u_{M,N-1} w^f_{M,N-2}$$
$$f_M(N-1) = \gamma_M(N-1)\alpha_M(N-1)$$
$$\check{k}_{M+1,N-1} = \begin{bmatrix} 0 \\ k_{M,N-1} \end{bmatrix} + \frac{\alpha^*_M(N-1)}{\lambda\zeta^f_M(N-2)} \begin{bmatrix} 1 \\ -w^f_{M,N-2} \end{bmatrix}$$
$$\zeta^f_M(N-1) = \lambda\zeta^f_M(N-2) + \alpha^*_M(N-1) f_M(N-1)$$
$$w^f_{M,N-1} = w^f_{M,N-2} + k_{M,N-1} f_M(N-1)$$
$$\bar{\gamma}_{M+1}(N) = \gamma_M(N-1)\frac{\lambda\zeta^f_M(N-2)}{\zeta^f_M(N-1)}$$

$$\bar{k}_{M+1,N} = \Psi\check{k}_{M+1,N-1}$$
$$\nu_M(N) = (\text{last entry of } \bar{k}_{M+1,N})$$
$$k_{M,N} = \bar{k}_{1:M,N} + \nu_M(N) q_{N-1}$$
$$\varepsilon_M(N) = \lambda\zeta^b_M(N-1)\nu^*_M(N)$$
$$\gamma_M(N) = \frac{\bar{\gamma}_{M+1}(N)}{1-\bar{\gamma}_{M+1}(N)\varepsilon_M(N)\nu_M(N)}$$
$$\varrho_M(N) = \gamma_M(N)\varepsilon_M(N)$$
$$\zeta^b_M(N) = \lambda\zeta^b_M(N-1) + \varepsilon^*_M(N)\varrho_M(N)$$
$$q_N = q_{N-1} + k_{M,N}\varrho_M(N)$$

$$\epsilon_M(N) = d(N) - u_{M,N} w_{M,N-1}$$
$$e_M(N) = \gamma_M(N)\epsilon_M(N)$$
$$w_{M,N} = w_{M,N-1} + k_{M,N} e_M(N)$$

VI. EVALUATION OF THE PRODUCT $\Psi\check{k}_{M+1,N-1}$

The fast algorithms in array and explicit forms require the computation of the product of a vector by the matrix Ψ. For example, consider the product $\Psi\check{k}_{M+1,N-1}$ in the fast algorithm of Table I. Our goal in this section is to show that this product

$$m = [0 \quad 0 \quad \cdots \quad 0 \quad 1]$$

and

$$\bar{v} = \begin{bmatrix} 1 & \frac{-a_0^* A_1}{A_0} & \frac{(a_0 a_1)^* A_2}{A_0} & \frac{-(a_0 a_1 a_2)^* A_3}{A_0} & \frac{(a_0 a_1 a_2 a_3)^* A_4}{A_0} & 0 \end{bmatrix}^T$$

can be evaluated efficiently by a network similar to the one of Fig. 3.

To begin with, recall from the discussion in Section IV that every network structure as in Fig. 2 gives rise to a relation of the form

$$\begin{bmatrix} u(N+1,0)^* \\ u(N+1,1)^* \\ u(N+1,2)^* \\ u(N,2)^* \end{bmatrix} = \underbrace{\begin{bmatrix} 1 & 0 & 0 & 0 \\ \frac{-a_0^* A_1}{A_0} & \frac{A_1}{A_0} & 1 & 0 \\ \frac{(a_0 a_1)^* A_2}{A_0} & \frac{-a_1^* A_2}{A_0} & -A_2 A_1 & a_2 \\ 0 & 0 & 0 & 1 \end{bmatrix}}_{\Psi^{-*}} \begin{bmatrix} u(N+1,0)^* \\ u(N,0)^* \\ u(N,1)^* \\ u(N,2)^* \end{bmatrix}. \tag{86}$$

In other words, the entries of two successive regressors are related as above or, more compactly, recursively, as given by

$$u(N+1,i) = a_i u(N,i) + \frac{A_i}{A_{i-1}} \times [u(N,i-1) - a_{i-1}^* u(N+1,i-1)].$$

This suggests that matrix-vector products of the form $y = \Psi^{-*}x$ can be efficiently computed as follows:

$$y(i) = a_i x(i) + \frac{A_i}{A_{i-1}}[x(i-1) - a_{i-1}^* y(i-1)] \tag{87}$$

where $\{x(i), y(i)\}$ denote the entries of $\{x, y\}$:

$$x = \begin{bmatrix} x(0) \\ x(1) \\ x(2) \\ \vdots \end{bmatrix} \quad y = \begin{bmatrix} y(0) \\ y(1) \\ y(2) \\ \vdots \end{bmatrix}.$$

A similar recursion can be used to evaluate the entries of $\Psi\check{k}_{M+1,N-1}$. To see this, we will elaborate further on the relation between Ψ and Ψ^{-*}.

Thus, recall that for $M = 5$, we have

$$\Psi = \begin{bmatrix} 1 & a_0^* & 0 & 0 & 0 & 0 \\ 0 & A_1 A_0 & a_1^* & 0 & 0 & 0 \\ 0 & -a_1 A_2 A_0 & A_2 A_1 & a_2^* & 0 & 0 \\ 0 & a_1 a_2 A_3 A_0 & -a_2 A_3 A_1 & A_3 A_2 & a_3^* & 0 \\ 0 & \frac{-a_1 a_2 a_3 A_0}{A_4} & \frac{a_2 a_3 A_1}{A_4} & \frac{-a_3 A_2}{A_4} & \frac{A_3}{A_4} & 0 \\ 0 & \frac{a_1 a_2 a_3 a_4 A_0}{A_4} & \frac{-a_2 a_3 a_4 A_1}{A_4} & \frac{-a_3 a_4 A_2}{A_4} & \frac{-a_4 A_3}{A_4} & 1 \end{bmatrix}$$

and Ψ^{-*} is given at the bottom of the page.

Now, note that Ψ and Ψ^{-*} have similar structures. In particular, the submatrices $\Psi_{1:M-2,:}$ and $\Psi^{-*}_{1:M-2,:}$ (which are defined by the rows lying between the horizontal lines) are related as follows. Introduce a matrix $\bar{\Psi}$ that is formed by replacing the $\{a_k^*\}$ in Ψ by their complex conjugates $\{a_k\}$

$$\bar{\Psi} = \begin{bmatrix} 1 & a_0 & 0 & 0 & 0 & 0 \\ 0 & A_1 A_0 & a_1 & 0 & 0 & 0 \\ 0 & -a_1^* A_2 A_0 & A_2 A_1 & a_2 & 0 & 0 \\ 0 & a_1^* a_2^* A_3 A_0 & -a_2^* A_3 A_1 & A_3 A_2 & a_3 & 0 \\ 0 & \frac{-a_1^* a_2^* a_3^* A_0}{A_4} & \frac{a_2^* a_3^* A_1}{A_4} & \frac{-a_3^* A_2}{A_4} & \frac{A_3}{A_4} & 0 \\ 0 & \frac{a_1^* a_2^* a_3^* a_4^* A_0}{A_4} & \frac{-a_2^* a_3^* a_4^* A_1}{A_4} & \frac{-a_3^* a_4^* A_2}{A_4} & \frac{-a_4^* A_3}{A_4} & 1 \end{bmatrix}.$$

Then, the *second* columns of Ψ and $\bar{\Psi}^{-*}$ will differ by a scaling factor, i.e.,

$$\Psi_{1:M-2,1} = A_0^2 \bar{\Psi}^{-*}_{1:M-2,1}$$

and all other columns will coincide.

Let $\Psi_{0,:}$ and $\Psi_{M-1,M}$ denote the top and last two rows of Ψ, respectively. Let us also partition $\check{k}_{M+1,N-1}$ as

$$\check{k}_{M+1,N-1} = \begin{bmatrix} \times \\ \check{k}_{M+1,N-1}(1) \\ \check{k}_{M+1,N-1}(2:M) \end{bmatrix}.$$

In this way, we can express the product $\Psi\check{k}_{M+1,N-1}$ as

$$\Psi\check{k}_{M+1,N-1} = \begin{bmatrix} \Psi_{0,:}\check{k}_{M+1,N-1} \\ \bar{\Psi}^{-*}_{1:M-2,:} \begin{bmatrix} 0 \\ A_0^2 \check{k}_{M+1,N-1}(1) \\ \check{k}_{M+1,N-1}(2:M) \end{bmatrix} \\ \Psi_{M-1:M,:}\check{k}_{M+1,N-1} \end{bmatrix} \tag{88}$$

$$= \begin{bmatrix} \check{k}_{M+1,N-1}(0) + a_0^* \check{k}_{M+1,N-1}(1) \\ \bar{\Psi}^{-*}_{1:M-2,:} k'_{M+1,N-1} \\ \Psi_{M-1:M,:}\check{k}_{M+1,N-1} \end{bmatrix} \tag{89}$$

where we have defined

$$k'_{M+1,N-1} = \begin{bmatrix} 0 \\ A_0^2 \check{k}_{M+1,N-1}(1) \\ \check{k}_{2:M,N-1} \end{bmatrix}$$

and where the zero entry that appears in the center submatrix will cancel the center part of the first column of $\bar{\Psi}^{-*}$ (for $M = 5$)

$$\begin{bmatrix} -a_0^* A_1/A_0 \\ (a_0 a_1)^* A_2/A_0 \\ -(a_0 a_1 a_2)^* A_3/A_0 \end{bmatrix}$$

and $\check{k}_{M+1,N-1}(1)$ denotes the second entry of $\check{k}_{M+1,N-1}$.

$$\Psi^{-*} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ \frac{-a_0^* A_1}{A_0} & \frac{A_1}{A_0} & a_1 & 0 & 0 & 0 \\ \frac{(a_0 a_1)^* A_2}{A_0} & \frac{-a_1^* A_2}{A_0} & A_2 A_1 & a_2 & 0 & 0 \\ \frac{-(a_0 a_1 a_2)^* A_3}{A_0} & \frac{(a_1 a_2)^* A_3}{A_0} & -a_2^* A_3^* A_1^* & A_3 A_2 & a_3 & 0 \\ \frac{(a_0 a_1 a_2 a_3)^* A_4}{A_0} & \frac{-(a_1 a_2 a_3)^* A_4}{A_0} & (a_2 a_3)^* A_4 A_1^* & -a_3^* A_4 A_2^* & A_4 A_3^* & a_4 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

Now, comparing Ψ^{-*} and $\bar{\Psi}^{-*}$, we find that they are identical, except that the $\{a_k\}$ in Ψ^{-*} are replaced by $\{a_k^*\}$. Therefore, matrix-vector products of the form $\bar{\Psi}^{-*}x$ can be evaluated in the same manner as $\Psi^{-*}x$ [i.e., as in (87) with $\{a_k\}$ replaced by $\{a_k^*\}$]. We thus find that the matrix product $\bar{\Psi}^{-*}_{1:M-2,:}k'_{M,N-1}$ can be computed efficiently via

$$\bar{k}_{M+1,N}(i) = a_i^* k'_{M+1,N-1}(i) + \frac{A_i}{A_{i-1}} \times [k'_{M+1,N-1}(i-1) - a_{i-1}\bar{k}_{M+1,N}(i-1)]. \quad (90)$$

In summary, the matrix-vector product $\Psi\check{k}_{M+1,N-1}$ can be computed efficiently as follows.

1. Define the vector

$$k'_{M+1,N-1} = \begin{bmatrix} 0 \\ A_0^2\check{k}_{M+1,N-1}(1) \\ \check{k}_{2:M,N-1} \end{bmatrix}.$$

2. Start with $\bar{k}_{M+1,N-1}(0) = 0$ and repeat for $i = 1$ to $M-2$

$$\bar{k}_{M+1,N}(i) = a_i^* k'_{M+1,N-1}(i) + \frac{A_i}{A_{i-1}}[k'_{M+1,N-1}(i-1) - a_{i-1}\bar{k}_{M+1,N}(i-1)].$$

This computes the center submatrix in the product (88).

3. Compute the first entry of $\bar{k}_{M+1,N}$ as

$$\bar{k}_{M+1,N}(0) = \check{k}_{M+1,N-1}(0) + a_0^*\check{k}_{M+1,N-1}(1).$$

4. Compute the inner product $\bar{k}_{M+1,N}(M-1) = \Psi_{M-1,:}\check{k}_{M+1,N-1}$.

5. Compute the last entry of $\bar{k}_{M+1,N}$ via

$$\bar{k}_{M+1,N}(M) = k'_{M+1,N-1}(M) - a_{M-1}\bar{k}_{M+1,N}(M-1).$$

Step 5 is simple since, except for the last entries, the last two rows of Ψ differ only by a scaling factor $-a_{M-1}$.

In the special case of a Laguerre model, Ψ has a lower triangular Toeplitz-like structure

$$\Psi = \begin{bmatrix} 1 & a^* & 0 & 0 & 0 & 0 \\ 0 & (1-|a|^2) & a^* & 0 & 0 & 0 \\ 0 & -a(1-|a|^2) & (1-|a|^2) & a^* & 0 & 0 \\ 0 & a^2(1-|a|^2) & -a(1-|a|^2) & (1-|a|^2) & a^* & 0 \\ 0 & -a^3 & a^2 & -a & 1 & 0 \\ 0 & a^4 & -a^3 & a^2 & -a & 1 \end{bmatrix} \quad (91)$$

which simplifies (90) to

$$\bar{k}_{M+1,N}(i) = a^* k'_{M+1,N-1}(i) + [k'_{M+1,N-1}(i-1) - a\bar{k}_{M+1,N}(i-1)].$$

Another possibility for the computation of the above matrix product is to rely on fast transform techniques by embedding a Toeplitz matrix into a larger $n \times n$ circulant matrix, which can then be diagonalized by a DFT, DHT, or any trigonometric transform matrix (note that the submatrix $\Psi_{0:M-2,1:M}$ is Toeplitz). A second technique is to express T as the sum of circulant and *skew*-circulant matrices, which again can be diagonalized by these transforms (see [25]).

The cost of the usual FIR FTF algorithm is known to be $\mathcal{O}(7M)$ operations. Here, due to steps 2 and 3 in the general model case, this computation simply amounts to $\mathcal{O}(9M)$ operations.

VII. Stability Issues

It is well known that the original fast fixed-order RLS recursions for shift structure data (FTF [2]), both in array and explicit forms, are unstable when implemented in finite precision arithmetic. Early mentions of such instability problems were reported in [2] and [3], even though the general idea behind the fast fixed-order algorithms had already been put forward in [4].

Of course, the numerical effects depend on the accuracy of the digital processor employed. However, increasing the wordlength *does not* completely solve the divergence problem. This can be verified by running simulations of the FTF algorithm in Matlab precision. It may take a while to diverge, but divergence will almost inevitably occur. The unstable behavior of the fast fixed-order RLS recursions can be better understood through the concepts of *backward consistency* and *minimality*, which is explained in [28]–[30] and which we briefly discuss here. Our goal is to extend the results of the FTF algorithm in [28]–[30] to the general case of orthonormal models.

A. Backward Consistency

The error propagation in fast RLS algorithms is originated in the prediction part of the recursions. The main idea is to represent the propagated quantities of the prediction section as the states $\mathbf{x}(N)$ of a nonlinear system, say

$$\mathbf{x}(N+1) = T[\mathbf{x}(N), s(N+1)] \quad (92)$$

where $s(N)$ is the input signal, and T is a memoryless nonlinearity that depends on the algorithm used. In the case of the FTF algorithm, the states are

$$\mathbf{x}(N) = \left\{ \begin{bmatrix} 1 \\ -w^f_{N-1} \end{bmatrix}, \zeta^f_M(N-1), \frac{k_{M,N}}{\gamma^{1/2}(N)}, \begin{bmatrix} -q_N \\ 1 \end{bmatrix}, \zeta^b_M(N), \gamma(N) \right\}. \quad (93)$$

Now, consider the perturbed system

$$\mathbf{x}'(N+1) = T[\mathbf{x}'(N), s(N+1)] + \delta(N) \quad (94)$$

where $\delta(N)$ is due to quantization. Then, state error $\mathbf{x}(N) - \mathbf{x}'(N)$ will remain bounded if (92) is exponentially stable for all states $\mathbf{x}(N)$ contained in a certain stability region $S_i(N)$ (the solution manifold) and if the perturbation δ does not push $\mathbf{x}'(N)$ outside $S_i(N)$.

Now, let $S_f(N)$ be the stability domain of the perturbed system (94). An algorithm is said to be *backward consistent* if the computed solution of a problem is the exact solution to a perturbed problem. The procedure for stability analysis is to check if $S_f(N) \subset S_i(N)$ for all N, in which case, its time recursions will be exponentially stable (see [29]).

B. Minimality

The answer to whether the fast fixed-order RLS filters are stable or not relies on the fact that these represent systems with *nonminimal* dimension, in which case, $S_f(N) \not\subset S_i(N)$, as shown in [28] and [29] for the FIR case. Our goal is to define a similar stability domain $S_i(N)$ for the FTF algorithm for the general model (22) by specifying the minimal components of the state vector $\mathbf{x}(N)$.

Thus, consider (37), which is written as

$$\begin{bmatrix} P_{N+1} & 0 \\ 0 & 0 \end{bmatrix} - \Psi \begin{bmatrix} 0 & 0 \\ 0 & P_N \end{bmatrix} \Psi^* = \frac{1}{\gamma(N)} \Psi \begin{bmatrix} 0 \\ k_{M,N} \end{bmatrix} \begin{bmatrix} 0 \\ k_{M,N} \end{bmatrix}^* \Psi^* + \lambda^{-1} L_N J L_N^* - \frac{1}{\gamma(N+1)} \begin{bmatrix} k_{M,N+1} \\ 0 \end{bmatrix} \begin{bmatrix} k_{M,N+1} \\ 0 \end{bmatrix}^* \tag{95}$$

and note that (12) can be written as

$$\begin{bmatrix} P_{N+1} & 0 \\ 0 & 0 \end{bmatrix} = \lambda^{-1} \begin{bmatrix} P_N & 0 \\ 0 & 0 \end{bmatrix} - \frac{1}{\gamma(N+1)} \begin{bmatrix} k_{M,N+1} \\ 0 \end{bmatrix} \begin{bmatrix} k_{M,N+1} \\ 0 \end{bmatrix}^*. \tag{96}$$

Substituting (96) into (95), we obtain the following rank-3 relation:

$$\begin{bmatrix} P_N & 0 \\ 0 & 0 \end{bmatrix} - \lambda\Psi \begin{bmatrix} 0 & 0 \\ 0 & P_N \end{bmatrix} \Psi^* = L_N J L_N^* + \frac{\lambda}{\gamma(N)} \Psi \begin{bmatrix} 0 \\ k_{M,N} \end{bmatrix} \begin{bmatrix} 0 \\ k_{M,N} \end{bmatrix}^* \Psi^* \tag{97}$$

where in the case of the FTF, L_N is defined in (85). This factorization is analogous to (8) in [29] and suggests that $3M+3$ variables are needed to propagate the solution. However, this is actually more than the necessary to represent these variables. To see this, consider a row vector that contains the basis functions $\mathcal{B}_k(z)$ of (22), which is given by

$$\bar{\mathcal{B}}_{M+1}(z) = [\mathcal{B}_M(z) \quad z^{-1}\mathcal{B}_M(z)\lambda^{-M/2}]$$

where

$$\mathcal{B}_M(z) \triangleq \left[\mathcal{B}_0(z) \quad \mathcal{B}_1(z)\lambda^{-1/2} \quad \cdots \quad \mathcal{B}_{M-1}(z)\lambda^{-(M-1)/2}\right]$$

and note that the following relation holds:

$$\bar{\mathcal{B}}_{M+1}(z)\lambda^{1/2}\Psi = [\mathcal{B}_0(z)\lambda^{1/2} \quad z^{-1}\mathcal{B}_M(z)]. \tag{98}$$

Now, multiply (97) by $\bar{\mathcal{B}}(z)$ from the left and $\bar{\mathcal{B}}^*(w)$ from the right, and define the following generating functions:

$$P(z,w^*) = \mathcal{B}_M(z) P_N \mathcal{B}_M^*(w) \tag{99}$$

$$W^f(z) \triangleq [\mathcal{B}_0(z)\lambda^{1/2} \quad z^{-1}\mathcal{B}_M(z)] \begin{bmatrix} 1 \\ -w^f_{M,N-1} \end{bmatrix} \times \frac{1}{\zeta_M^{f/2}(N-1)}$$

$$W^b(z) \triangleq [\mathcal{B}_M(z) \quad z^{-1}\mathcal{B}_M(z)\lambda^{-M/2}] \begin{bmatrix} -q_N \\ 1 \end{bmatrix} \frac{1}{\zeta_M^{b/2}(N)} \tag{100}$$

$$K(z) \triangleq z^{-1}\sqrt{\lambda}\mathcal{B}_M(z) k_{M,N} \gamma_M^{-1/2}(N)$$

and similarly for $\{W^{f*}(w), W^{b*}(w), K^*(w)\}$. We then obtain the following relation:

$$(1 - 1/zw^*)P(z,w^*) = W^f(z)W^{f*}(w) + K(z)K^*(w) - W^b(z)W^{b*}(w). \tag{101}$$

This equation extends (10) of [29] (see also [31, pp. 697]) to the case of (weighted) orthonormal bases [see (98)]. Now, choose $w = z$, and note that because P_N is positive definite, the right-hand side of (101) satisfies

$$W^f(z)W^{f*}(z) + K(z)K^*(z) - W^b(z)W^{b*}(z) \begin{cases} > 0, & |z| > 1 \\ = 0, & |z| = 1 \\ < 0, & |z| < 1 \end{cases}$$

which implies that $W^b(z)$ must not have zeros inside the unit circle. Note also that by choosing $w = 1/z^*$, we have that

$$W^b(z)W^{b*}(1/z^*) = W^f(z)W^{f*}(1/z^*) + K(z)K^*(1/z^*). \tag{102}$$

Therefore, $W^b(z)$ is seen uniquely defined from $\{W^f(e^{j\omega}), K(e^{j\omega})\}$ as the spectral factor of the right-hand side of (102) that has all its zeros outside the unit circle and all its poles inside the unit circle. The quantity $\zeta_M^b(N)$ is inferred by normalizing the last coefficient of $W^b(z)$ to unity.

Now, in order to completely characterize the minimal components of the FTF algorithm, we further need to establish one last relation. Thus, consider the analogous of (50) for the backward prediction problem:

$$\bar{\gamma}_{M+1}(N+1) = \gamma_M(N+1)\frac{\lambda\zeta_M^b(N)}{\zeta_M^b(N+1)}. \tag{103}$$

Using the fact that $\bar{\gamma}_{M+1}(N+1) = \check{\gamma}_{M+1}(N)$ [see (74)], we have the following time update relation for $\gamma_M(N)$:

$$\gamma_M(N+1) = \gamma_M(N)\frac{\zeta_M^f(N-1)}{\zeta_M^f(N)}\frac{\zeta_M^b(N+1)}{\zeta_M^b(N)}.$$

Solving for $\gamma_M(N)$, with the initial conditions $\gamma_M(0) = 1$, and $\{\zeta_M^f(-1), \zeta_M^b(0)\}$ as defined in (79) and (80), we obtain

$$\boxed{\gamma_M(N) = \frac{\zeta_M^f(-1)}{\zeta_M^b(0)}\frac{\zeta_M^b(N)}{\zeta_M^f(N-1)}} \tag{104}$$

Note that for shift data structure, $\zeta_M^f(-1) = \mu\lambda^{-1}$ and $\zeta_M^b(0) = \mu\lambda^{-(M+1)}$, so that (104) collapses to the well-known relation

$$\gamma(N) = \frac{\lambda^M \zeta_M^b(N)}{\zeta_M^f(N-1)}. \tag{105}$$

Equations (102) and (104) imply that, in fact, only $2M+1$ degrees of freedom are needed to represent the solution [i.e., M coefficients of $w_M^f(N-1)$, M coefficients of $k_{M,N}\gamma^{-1/2}(N)$, and $\zeta_M^f(N-1)$]. This is because $W^b(z)$ can be obtained from the spectral factorization (102), and $\zeta_M^b(N)$ is inferred by normalizing the last coefficient of $W^b(z)$ (in the orthonormal basis representation) to unity.

Therefore, the set S_i is represented by the variables such that

i) spectral factorization (102) is verified with respect to the orthonormal basis defined in (100);
ii) with $W^b(z)$ having all its zeros outside the unit circle and all poles inside the unit circle;
iii) likelihood variable $\gamma(N)$ obtained from (104) satisfies $0 \leq \gamma(N) < 1$.

In this case, the minimal components of the state vector $\mathbf{x}(N)$ are

$$\left\{ \begin{bmatrix} 1 \\ -w^f_{N-1} \end{bmatrix}, \zeta^f_M(N-1), k_{M,N}\gamma^{-1/2}(N) \right\}$$

and the error between the actual and computed quantities

$$\left\{ \begin{bmatrix} -q_N \\ 1 \end{bmatrix}, \zeta^b_M(N), \gamma(N) \right\}$$

is interpreted as a perturbation that leads the state outside the stability domain S_i.

The *rescuing mechanisms* proposed in [3] represent a rough way of projecting the state $\mathbf{x}'(N)$ back onto the manifold S_i once the likelihood variable becomes negative. The approach used in [3] amounts to monitoring the quantity (we describe the procedure for our algorithm)

$$1 - \bar{\gamma}_{M+1}(N)\varepsilon_M(N)\nu_M(N)$$

which appears in the denominator of (70). If it is positive, the algorithm continues its flow. Otherwise, we restart the algorithm as follows:

$$\begin{aligned} u_{M,N} &= 0 \\ w_{M,N} &= w_{M,N-1} \\ \zeta^f_M(N-1) &= \zeta^f_M(-1) \\ \zeta^b_M(N) &= \zeta^b_M(0) \\ w^f_{M,N-1} &= k_{M,N} = 0 \\ q_N &= \Pi c \\ \gamma_M(N) &= 1 \end{aligned}$$

that is, the algorithm is reinitialized with the current optimal solution. In our simulations, however, we noticed that the use of (104) in the rescuing section provides a more reliable solution:

$$\begin{aligned} u_{M,N} &= 0 \\ w_{M,N} &= w_{M,N-1} \\ \zeta^b_M(N) &= \frac{\zeta^b_M(0)}{\zeta^f_M(-1)}\zeta^f_M(N-1) \\ w^f_{M,N-1} &= k_{M,N} = 0 \\ q_N &= \Pi c \\ \gamma_M(N) &= 1. \end{aligned}$$

Fig. 5 illustrates the use of this rescue mechanism applied to the extended fast RLS algorithm. We used an $M = 5$-tap filter with $\lambda = 0.95$ in Matlab precision. It can be seen that after 450 iterations, the filter without a rescue mechanism becomes unstable. We observed that the rescuing mechanism that makes use of (104) appeared to be more robust to finite precision than the original rescuing mechanism of [3].

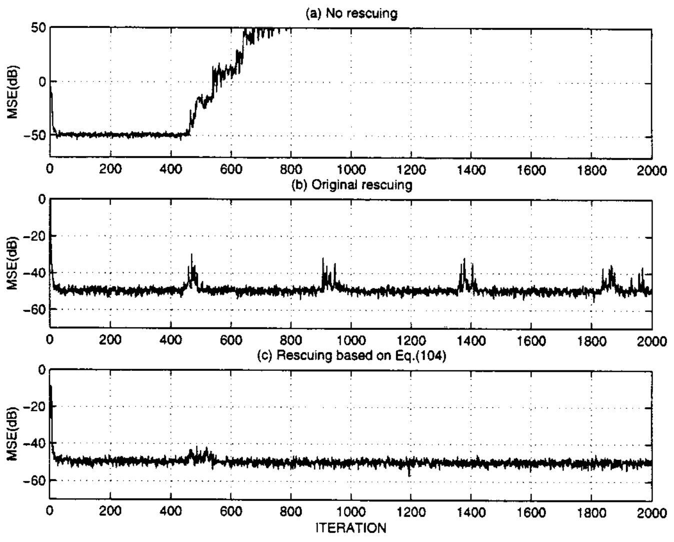


Fig. 5. (a) Extended FTF with no rescuing. (b) Original rescuing mechanism. (c) Rescuing based on (104).

Another approach for addressing the stability problem of the FTF algorithm was proposed in [27]. Although the resulting algorithm is claimed to be numerically stable, the method of analysis employed and the corresponding "stabilized" solution are valid only under some restrictive conditions, and instability can still occur in practice.

The idea behind the analysis in [27] was based on introducing redundancy into the computation of certain quantities, i.e., on computing some variables of the FTF algorithm in two different ways. In so doing, it is possible to obtain measurements of the numerical errors accumulated in these quantities. These measurements are then used in a feedback mechanism in order to "stabilize" the recursions.

Unfortunately, this stabilization procedure assumes a restricted class of stationary signals. Moreover, the forgetting factor λ has to be chosen very close to unity in order to avoid divergence (this is also the case for the nonstabilized algorithm). This condition hinders its use in important practical circumstances, as for example, in high-order adaptive filtering schemes and in cases of nonstationary enviroments.

VIII. Simulation Results

In this section, we run some system identification examples using a Laguerre adaptive structure. In all experiments, the Laguerre pole was optimized offline by adjusting the pole location and running the experiment until maximum cancellation was observed. (We remark that an efficient method to find the optimal pole position for a certain system is currently an open problem, and it is beyond the scope of this work.)

Example 1—IIR System Identification: Fig. 6 compares the MSE performance (obtained by averaging over 100 runs of the experiment) of the fast Laguerre filter with the fast FIR filter

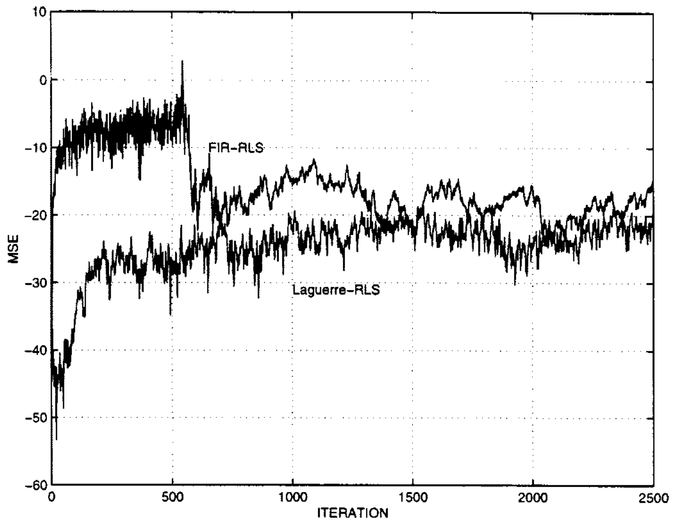


Fig. 6. Comparison of a six-tap fast Laguerre filter with a 500-tap fast FIR filter.

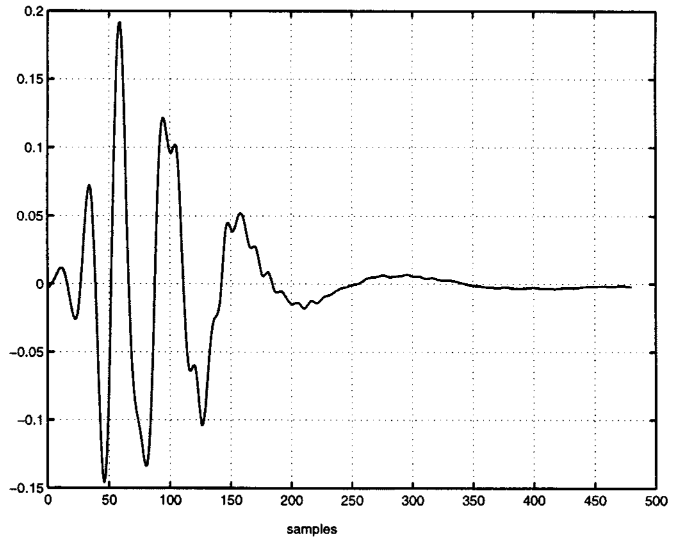


Fig. 7. Echo path impulse response.

(both implemented in explicit forms). The model used for the system identification example in Fig. 6 is the same from [24])

$$G(z) = \frac{0.0017z^{-1}(1+0.673z^{-1})}{(1-0.368z^{-1})(1-0.819z^{-1})(1-0.995z^{-1})}.$$

The input to the unknown model was taken as colored noise, and the signal-to-noise ratio (SNR) at the output was 50 dB. The Laguerre filter was implemented with six taps, with the pole located at $a = 0.95$, and the FIR filter with 500 taps. Both filters used $\lambda = 0.999$. We see from the figure that the Laguerre-RLS algorithm presents faster convergence and achieves a lower MSE level compared with the fast FIR-RLS algorithm.

Example 2—Typical Echo Path Identification: Fig. 7 shows a typical echo path for line echo cancelers. Here, the pole location of a Laguerre structure with 50 taps was fixed at $a = 0.7$. The corresponding fast FIR filter was simulated with 200 taps. Fig. 8 illustrates the learning curve for both schemes. Note that

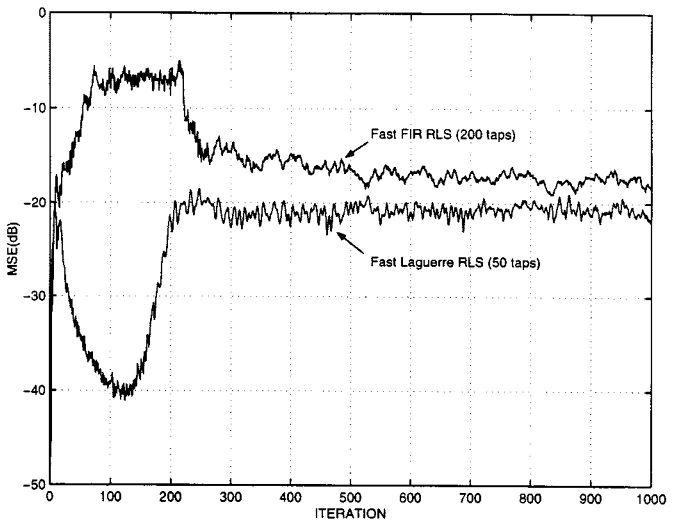


Fig. 8. Comparison of a 50-tap fast Laguerre filter with a 200-tap fast FIR filter for a typical line echo impulse response.

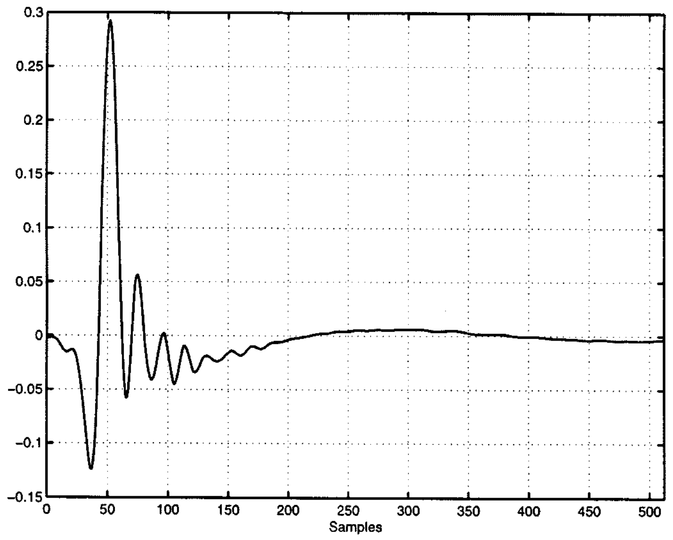


Fig. 9. Another echo path impulse response.

the Laguerre filter provides better cancellation during the initial iterations of the algorithm.

Example 3—Performance of Different Algorithms: Fig. 9 shows another typical echo path for line echo cancelers.

Here, the simulation was performed in order to attain the test scenario specified in the G.168 standard [33] for echo cancelers. The input used for the adaptive echo canceler was a composite source signal. The performance criterion is the *echo return line enhancement* (ERLE), which is defined as

$$\text{ERLE} = 10\log_{10}\left(\frac{\sum_{k=n}^{n-T}|e(k)|^2}{\sum_{k=n}^{n-T}|d(k)|^2}\right)$$

where T is a window of length 6000 (it has to be greater than 5600) according to the G.168 standard.

The simulation was run with normalized LMS (NLMS), fast FIR-RLS, and fast Laguerre-RLS algorithms, with $\lambda = 1$. The Laguerre pole location was set to $a = 0.7$. No measurement noise was added at the output, and the number of coefficients

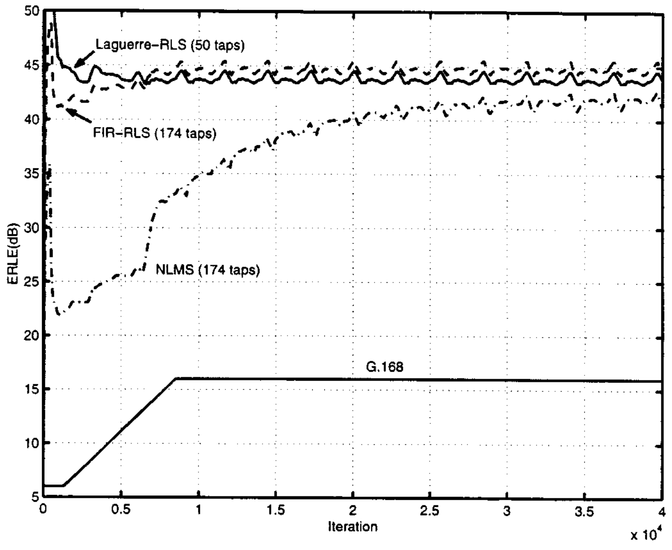


Fig. 10. ERLE for the NLMS, fast FIR-RLS, and fast Laguerre-RLS algorithms.

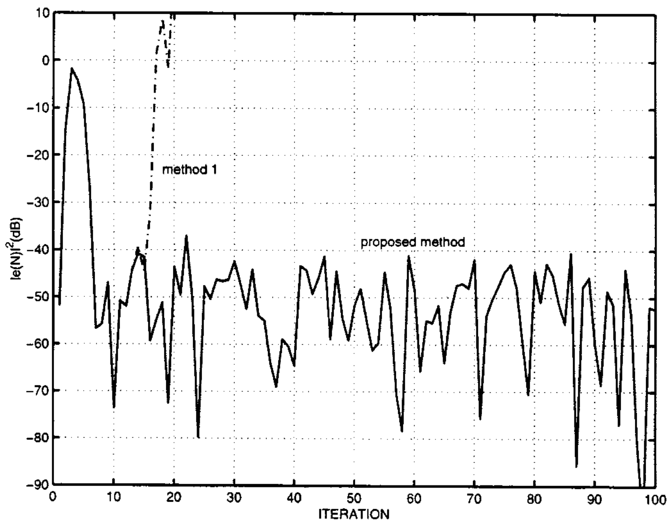


Fig. 11. Simulations when two methods of implementing the hyperbolic rotations are used.

in each adaptive filter was set such that approximately 45 dB cancellation was obtained. Fig. 10 shows the resulting ERLE. The NLMS and fast FIR-RLS adaptive algorithms achieve 45 dB cancellation using 174 taps for the adaptive filters. The fast Laguerre-RLS achieves the same cancellation with 50 adaptive coefficients.

Fig. 11 illustrates the effect of using two methods of implementing the hyperbolic rotations in the array-based algorithm as described in Section IV-B. We have observed that the proposed method is more reliable than the standard method of computing the rotation matrix Θ, although it can still encounter some numerical difficulties. In the array form, for $\lambda = 1$, and using Matlab precision, we did not notice such problems even over long simulations. [Actually, even the fast array algorithm for regression vectors with shift structure is also unstable for $\lambda = 1$ if the rotations are not implemented with care.]

IX. Conclusion

We have shown that the fast fixed-order RLS algorithms are not limited to tapped-delay-line data structures, as original derivations in the literature suggest. The approach here, following [8] and [9], shows that for more general data structures, we can derive fast filters in both array and explicit forms.

References

[1] 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.

[2] 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.

[3] D. W. Lin, "On digital implementation of the fast Kalman algorithm," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-32, pp. 998–1005, Oct. 1984.

[4] 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.

[5] D. T. L. Lee, M. Morf, and B. Friedlander, "Recursive least-squares ladder estimation algorithms," *IEEE Trans. Circuits. Syst.*, no. 6, pp. 467–481, June 1981.

[6] B. Friedlander, "Lattice filters for adaptive processing," *Proc. IEEE*, vol. 70, pp. 829–867, Aug. 1982.

[7] H. Lev-Ari, T. Kailath, and J. Cioffi, "Least-squares adaptive lattice and transversal filters: A unified geometrical theory," *IEEE Trans. Inform. Theory*, vol. IT-30, pp. 222–236, Mar. 1984.

[8] A. H. Sayed and T. Kailath, "Extended Chandrasekhar recursions," *IEEE Trans. Automat. Contr.*, vol. 39, no. 3, pp. 619–623, Mar. 1994.

[9] ——, "A state-space approach to adaptive RLS filtering," *IEEE Signal Processing Mag.*, vol. 11, pp. 18–60, July 1994.

[10] R. Merched and A. H. Sayed, "Order-recursive RLS Laguerre adaptive filtering," *IEEE Trans. Signal Processing*, vol. 48, pp. 3000–3010, Nov. 2000.

[11] ——, "RLS-Laguerre lattice adaptive filtering: Error-feedback, normalized and array-based algorithms," *IEEE Trans. Signal Processing*, vol. 49, pp. 2565–2576, Nov. 2001.

[12] T. Kailath, S. Kung, and M. Morf, "Displacement ranks of matrices and linear equations," *J. Math. Anal. Applicat.*, vol. 68, pp. 395–407, 1979.

[13] B. Wahlberg, "System identification using Laguerre models," *IEEE Trans., Automat. Contr.*, vol. 36, pp. 551–562, May 1991.

[14] P. M. Makila, "Approximation of stable systems by Laguerre filters," *Automatica*, vol. 26, pp. 333–345, Feb. 1990.

[15] G. A. Dumont and C. C. Zervos, "Adaptive control based on orthonormal series representation," in *Proc. 2nd IFAC Workshop Adaptive Syst. Contr. Signal Process.*, Lund, Sweden, 1988, pp. 371–376.

[16] P. Heuberger, B. Ninness, T. O. e Silva, P. Van den Hof, and B. Wahlberg, "Modeling and identification with orthogonal basis functions," presented at the 36th IEEE CDC Pre-Conf. Workshop, San Diego, CA, Dec. 1997.

[17] J. J. Shynk, "Adaptive IIR filtering," *IEEE Acoust., Speech, Signal Processing Mag.*, vol. 6, pp. 4–21, Apr. 1989.

[18] P. A. Regalia, *Adaptive IIR Filtering in Signal Processing and Control*. New York: Marcel Dekker, 1995.

[19] K. Steiglitz and L. E. McBride, "A technique for the identification of linear systems," *IEEE Trans. Automat. Contr.*, vol. AC–10, pp. 461–464, Oct. 1965.

[20] A. C. den Brinker, "Laguerre-domain adaptive filters," *IEEE Trans. Signal Processing*, vol. 42, pp. 953–956, Apr. 1994.

[21] B. Ninness and F. Gustafsson, "A unifying construction of orthonormal bases for system identification," *IEEE Trans. Automat. Contr.*, vol. 42, pp. 515–521, Apr. 1997.

[22] J. W. Davidson and D. D. Falconer, "Reduced complexity echo cancelation using orthonormal functions," *IEEE Trans. Circuits Syst.*, vol. 38, no. 1, pp. 20–28, Jan. 1991.

[23] L. Salama and J. E. Cousseau, "Efficient echo cancelation based on an orthogonal adaptive IIR realization," *Proc. SBT/IEEE Int. Telecomm. Symp.*, Aug. 1998.

[24] Z. Fejzo and H. Lev-Ari, "Adaptive Laguerre-lattice filters," *IEEE Trans. Signal Processing*, vol. 45, pp. 3006–3016, Dec. 1997.

[25] G. Heinig and K. Rost, “Representations of Toeplitz-plus-Hankel matrices using trigonometric transformations with application to fast matrix-vector multiplication,” *Linear Algebra Its Applicat.*, vol. 257–276, pp. 225–248, 1998.

[26] A. Houacine and G. Demoment, “Chandrasekhar adaptive regularizer for adaptive filtering,” in *Proc. ICASSP*, vol. 4, Apr. 1986, pp. 2967–2970.

[27] 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. 92–113, Jan. 1991.

[28] D. T. M. Slock, “The backward consistency concept and round-off error propagation dynamics in recursive least-squares algorithms,” *Opt. Eng.*, vol. 31, pp. 1153–1169, June 1992.

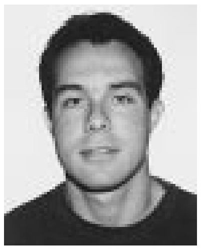
[29] P. A. Regalia, “Numerical stability issues in fast least-squares adaptation algorithms,” *Opt. Eng.*, vol. 31, pp. 1144–1152, June 1992.

[30] ——, “Numerical stability properties of a QR-based fast least squares algorithm,” *IEEE Trans. Signal Processing*, vol. 41, pp. 2006–2109, June 1993.

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

[32] P. A. Regalia and F. Desbouvries, “Displacement structures of covariance matrices, lossless systems, and numerical algorithm design,” *SIAM J. Matrix Anal. Appl.*, vol. 16, no. 2, pp. 536–564, Apr. 1995.

[33] ITU-T Recommendation G.168, “Digital network echo cancelers”, 1997.



Ricardo Merched (S’97) was born in Rio de Janeiro, Brazil. He received the B.S. and M.S. degrees in electrical engineering from Universidade Federal do Rio de Janeiro, in 1995 and 1997, respectively. He received the Ph.D. degree in electrical engineering from the University of California, Los Angeles, in September 2001.

His areas of interest include different aspects of adaptive filter design, especially multirate and fast adaptive filtering algorithms.

Ali H. Sayed (F’00) received the Ph.D. degree in electrical engineering in 1992 from Stanford University, Stanford, CA.

He is Professor of Electrical Engineering at the University of California, Los Angeles. He has over 160 journal and conference publications, is coauthor of the research monograh *Indefinite Quadratic Estimation and Control* (Philadelphia, PA: SIAM, 1999) and of the graduate-level textbook *Linear Estimation* (Englewood Cliffs, NJ: Prentice-Hall, 2000). He is also coeditor of the volume *Fast Reliable Algorithms for Matrices with Structure* (Philadelphia, PA: SIAM, 1999). He is a member of the editorial boards of the *SIAM Journal on Matrix Analysis and Its Applications* and of the *International Journal of Adaptive Control and Signal Processing*, and he has served as coeditor of special issues of the journal *Linear Algebra and Its Applications*. He has contributed several articles to engineering and mathematical encyclopedias and handbooks and has served on the program committees of several international meetings. He has also consulted with industry in the areas of adaptive filtering, adaptive equalization, and echo cancellation. His research interests span several areas including adaptive and statistical signal processing, filtering and estimation theories, equalization techniques for communications, interplays between signal processing and control methodologies, and fast algorithms for large-scale problems. To learn more about his work, visit the website of the UCLA Adaptive Systems Laboratory at http://www.ee.ucla.edu/asl.

Dr. Sayed is a recipient of the 1996 IEEE Donald G. Fink Award. He is Associate Editor of the IEEE TRANSACTIONS ON SIGNAL PROCESSING.