Time-Variant Displacement Structure and Triangular Arrays

Ali H. Sayed, *Member, IEEE,* Hanoch Lev-Ari, *Member, IEEE,* and Thomas Kailath, *Fellow, IEEE*

***Abstract*— We extend the concept of displacement structure to time-variant matrices and use it to efficiently and recursively propagate the Cholesky factor of such matrices. A natural implementation of the algorithm is via a modular triangular array of processing elements. When the algorithm is applied to solve the normal equations that arise in adaptive least-squares filtering, we get the so-called QR algorithm, with the extra bonus of a parallelizable procedure for determining the weight vector. It is shown that the general algorithm can also be implemented in time-variant lattice form; a specialization of this result yields a time-variant Schur algorithm.**

I. INTRODUCTION

THE notion of displacement structure provides a natural framework for the solution of many problems in signal processing and mathematics, and represents a powerful and unifying tool for exploiting the existing structure in several applications (see [1], [2] for surveys on the origin and history of the subject). In this paper we extend the notion of structured matrices to the time-variant setting and show that we can, as well, study matrices that exhibit structured time-variations. We shall say that an $n \times n$ matrix $R(t)$ has a time-variant Toeplitz-like displacement structure if the matrix difference $\nabla R(t)$ defined by

$$\nabla R(t) = R(t) - F(t)R(t-\Delta)F^*(t)$$

has low rank, say $r(t)$ (usually $r(t) \ll n$), for some lower triangular $n \times n$ matrix $F(t)$ whose diagonal elements we shall denote by $\{f_i(t)\}_{i=0}^{n-1}$. The symbol $*$ stands for Hermitian conjugation (complex conjugation for scalars), and the indices t and $(t-\Delta)$ denote two discrete-time instants. It follows from the low rank property that we can factor $\nabla R(t)$ and write

$$R(t) - F(t)R(t-\Delta)F^*(t) = G(t)J(t)G^*(t) \qquad (1)$$

Manuscript received October 24, 1992; revised May 19, 1993. The associate editor coordinating the review of this paper and approving it for publication was Dr. James Zeidler. This work was supported in part by the Air Force Office of Scientific Research, Air Force Systems Command under Contract AFOSR91-0060, and by the Army Research Office under contract DAAL03-89-K-0109. The work of A. H. Sayed was supported by a fellowship from Fundação de Amparo à Pesquisa do Estado de São Paulo while on leave from the Department of Electrical Engineering, Escola Politécnica da Universidade de São Paulo, Brazil.

A. H. Sayed is with the Department of Electrical and Computer Engineering, University of California, Santa Barbara, CA, USA 93106.

H. Lev-Ari is with the Department of Electrical and Computer Engineering, Northeastern University, Boston MA, USA 02115.

T. Kailath is with the Information Systems Laboratory, Stanford University, Stanford, CA, USA 94305.

IEEE Log Number 9216667.

where $G(t)$ is an $n \times r(t)$ so-called generator matrix and $J(t)$ is an $r(t) \times r(t)$ signature matrix with as many ± 1's as $\nabla R(t)$ has strictly positive or negative eigenvalues. The notation $I_{p(t)}$ refers to the $p(t) \times p(t)$ identity matrix and

$$J(t) = \begin{bmatrix} I_{p(t)} & \mathbf{0} \\ \mathbf{0} & -I_{q(t)} \end{bmatrix}, r(t) = p(t) + q(t).$$

Special cases of the time-variant structure (1) often arise in adaptive filtering [3]–[5], where one is usually faced with the task of computing a new estimate at time t upon the arrival of a new datum, given the old estimate at time $t-1$. In the standard recursive least-squares setting (see, e.g., [3], [4]), this problem reduces to solving normal equations where the coefficient matrix, say $\Phi(t)$, varies with time as follows: $\Phi(t) - \lambda\Phi(t-1) = \mathbf{u}^*(t)\mathbf{u}(t)$, for some scalar $0 < \lambda \leq 1$, and row vector $\mathbf{u}(t)$. This is clearly a special case of (1) with $R(t) = \Phi(t)$, $F(t) = \sqrt{\lambda}I$, $r(t) = 1$, $J(t) = 1$, $G(t) = \mathbf{u}^*(t)$ and $\Delta = 1$. In this case, $\Phi(t)$ and $\lambda\Phi(t-1)$ differ by a rank one "update" matrix. In other problems, such as in the block RLS formulation (see, e.g., [4], [6]), the matrices $\Phi(t)$ and $\lambda\Phi(t-1)$ differ by a higher order rank update, where the single column $\mathbf{u}^*(t)$ is replaced by a matrix with multiple columns. More general forms of the time-variant displacement structure (1) arise in the study of time-variant interpolation problems and matrix completion problems, as discussed in [7]–[11].

In this paper we consider a general time-variant structured matrix $R(t)$ as in (1) and show how to exploit the displacement structure to efficiently and recursively propagate its Cholesky factor. We also discuss an associated triangular array interpretation, and show that the derived algorithm can be implemented by an array of elementary cells composed of elementary rotations and time-variant tapped-delay filters. We then consider the special case of the recursive least-squares problem and show that the derived algorithm collapses to the now widely studied QR algorithm (see [3], [4], [12]–[14] and the references therein), with the extra ingredient of allowing for a parallel extraction of the weight vector.

To conclude this introduction, let us state a readily established matrix result that since [15] has played an important role in the derivation of all so-called square-root algorithms.

Lemma 1.1 Consider two $n \times m$ ($n \leq m$) matrices A and B. If $AJA^* = BJB^*$ is of full rank, for some $m \times m$ signature matrix $J = (I_p \oplus -I_q), p+q = m$, then there exists an $m \times m$ J-unitary matrix Θ ($\Theta J \Theta^* = J$) such that $A = B\Theta$.

Proof: One proof follows by invoking the hyperbolic singular value decompositions of A and B (see, e.g., [16] or

([17], pp. 43–45), viz.

$$A = U_A \begin{bmatrix} \Sigma_{A,+} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \Sigma_{A,-} & \mathbf{0} \end{bmatrix} V_A^*,$$

$$B = U_B \begin{bmatrix} \Sigma_{B,+} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \Sigma_{B,-} & \mathbf{0} \end{bmatrix} V_B^*$$

where U_A and U_B are $n \times n$ unitary matrices, V_A and V_B are $m \times m$ J-unitary matrices, and $\Sigma_{A,+}, \Sigma_{A,-}$, $\Sigma_{B,+}$, and $\Sigma_{B,-}$, are $p' \times p'$, $q' \times q'$, $p' \times p'$ and $q' \times q'$ diagonal matrices, respectively, with $p' + q' = n$. It further follows from the full rank condition and the equality $AJA^* = BJB^*$, that $\Sigma_{A,+} = \Sigma_{B,+}$, $\Sigma_{A,-} = \Sigma_{B,-}$, and that we can choose $U_A = U_B$. Let $\Theta = JV_BJV_A^*$ then $\Theta J \Theta^* = J$ and $B\Theta = A$.

II. Recursive Cholesky Factorization

The positive-definiteness of $R(t)$ guarantees the existence of a unique (lower triangular) Cholesky factor $\overline{L}(t)$ such that $R(t) = \overline{L}(t)\overline{L}^*(t)$. We shall denote the nonzero parts of the columns of $\overline{L}(t)$ by $\{\overline{l}_i(t)\}_{i=0}^{n-1}$. We shall also from now on, and without loss of generality, write $(t-1)$ instead of $(t-\Delta)$.

We first verify that because of the special time-variant structure (1) (and with an additional "sparsity" condition on $F(t)$), the Cholesky factor of $R(t)$ can be updated from time $(t-1)$ to time t in $O(r(t)n^2)$ operations (multiplications and additions), viz., $\overline{L}(t-1) \rightarrow \overline{L}(t)$ in $O(r(t)n^2)$ operations. It follows from (1) that we can write

$$[\overline{L}(t) \;\; \mathbf{0}] \begin{bmatrix} \overline{L}^*(t) \\ \mathbf{0} \end{bmatrix} = [F(t)\overline{L}(t-1) \;\;\; G(t)] \times \begin{bmatrix} I_n & \mathbf{0} \\ \mathbf{0} & J(t) \end{bmatrix} \begin{bmatrix} \overline{L}^*(t-1)F^*(t) \\ G^*(t) \end{bmatrix}.$$

This last expression fits into the statement of Lemma 1.1. Hence, there exists an $(I_n \oplus J(t))$-unitary matrix $\Gamma(t)$ such that

$$[\overline{L}(t) \quad \mathbf{0}] = [F(t)\overline{L}(t-1) \quad G(t)]\Gamma(t). \tag{2}$$

In other words, $\Gamma(t)$ is an $(I_n \oplus J(t))$–unitary rotation that produces the block zero entry in the postarray on the left-hand side of the above expression. Schematically, $\Gamma(t)$ takes a prearray of *numbers* of the form

$$\begin{bmatrix} x & & & x & x & x \\ x & x & & x & x & x \\ x & x & x & x & x & x \end{bmatrix}$$

and transforms it to a postarray of *numbers* of the form

$$\begin{bmatrix} x & & & 0 & 0 & 0 \\ x & x & & 0 & 0 & 0 \\ x & x & x & 0 & 0 & 0 \end{bmatrix}.$$

This transformation can clearly be implemented as a sequence of elementary transformations, say $\Gamma_0(t), \Gamma_1(t), \cdots$, that produces the block zero in the postarray by introducing one zero (row) at a time, such as

$$\begin{bmatrix} x & & & x & x & x \\ x & x & & x & x & x \\ x & x & x & x & x & x \end{bmatrix} \xrightarrow{\Gamma_0(t)} \begin{bmatrix} x & & & 0 & 0 & 0 \\ x & x & & x & x & x \\ x & x & x & x & x & x \end{bmatrix}$$
$$\xrightarrow{\Gamma_1(t)} \cdots \xrightarrow{\Gamma_2(t)} \begin{bmatrix} x & & & 0 & 0 & 0 \\ x & x & & 0 & 0 & 0 \\ x & x & x & 0 & 0 & 0 \end{bmatrix}.$$

We now verify that this procedure leads to a simple array algorithm that efficiently computes $\overline{L}(t)$ from the knowledge of $\overline{L}(t-1), F(t)$ and $G(t)$. The annihilation of each row can be achieved in *several* ways and we do not pretend to exhaust all possibilities. We shall instead describe the general picture and some possible options.

Let $d_0(t)$ and $g_0(t)$ denote the $(0,0)$ entry of $R(t)$ and the first row of $G(t)$, respectively. It follows from the displacement equation (1) and from the positive-definiteness of $R(t)$ that

$$d_0(t) = f_0(t)d_0(t-1)f_0^*(t) + g_0(t)J(t)g_0(t)^* > 0.$$

Consequently, the first row vector of the prearray in (2), viz., $[f_0(t)d_0^{1/2}(t-1) \quad \mathbf{0} \quad g_0(t)]$, has positive $(I_n \oplus J(t))$-norm (by the J-norm of a row vector $\mathbf{x}$, where J is a signature matrix, we mean the indefinite quantity $\mathbf{x}J\mathbf{x}^*$, which can be positive, negative, or even zero). Hence, by Lemma 1.1, we can always find an $(I_n \oplus J(t))$-unitary transformation $\Gamma_0(t)$ such that the above row is reduced to the form $[d_0^{1/2}(t) \quad \mathbf{0} \quad \mathbf{0}]$. That is

$$[f_0(t)d_0^{1/2}(t-1) \quad \mathbf{0} \quad g_0(t)]\Gamma_0(t) = [d_0^{1/2}(t) \quad \mathbf{0} \quad \mathbf{0}].$$

It is clear that we can express this last transformation more compactly by dropping the zero entries that are common to both the pre- and postarrays, viz.

$$[f_0(t)d_0^{1/2}(t-1) \quad g_0(t)]\overline{\Gamma}_0(t) = [d_0^{1/2}(t) \quad \mathbf{0}]$$

where $\overline{\Gamma}_0(t)$ is a $(1 \oplus J(t))$-unitary rotation. The relation between $\overline{\Gamma}_0(t)$ and $\Gamma_0(t)$ is evident: if we write

$$\overline{\Gamma}_0(t) = \begin{bmatrix} a_0(t) & \mathbf{b}_0(t) \\ \mathbf{c}_0(t) & \mathbf{s}_0(t) \end{bmatrix}$$

where $a_0(t)$ is a scalar, $\mathbf{b}_0(t)$ is a $1 \times r(t)$ row vector, $\mathbf{c}_0(t)$ is an $r(t) \times 1$ column vector, and $\mathbf{s}_0(t)$ is an $r(t) \times r(t)$ matrix, then $\Gamma_0(t)$ is given by

$$\Gamma_0(t) = \begin{bmatrix} a_0(t) & \mathbf{0} & \mathbf{b}_0(t) \\ \mathbf{0} & I_{n-1} & \mathbf{0} \\ \mathbf{c}_0(t) & \mathbf{0} & \mathbf{s}_0(t) \end{bmatrix}.$$

We see that the effect of $\Gamma_0(t)$, when applied to the prearray of expression (2), is to annihilate the first row of $G(t)$ by pivoting with the first column of $F(t)\overline{L}(t-1)$, while keeping unaltered the remaining columns of $F(t)\overline{L}(t-1)$. This can be expressed as follows

$$[F(t)\overline{L}(t-1) \quad G(t)]\Gamma_0(t) = \left[\overline{l}_0(t) \; \begin{matrix} \mathbf{0} \\ F_1(t)\overline{L}_1(t-1) \end{matrix} \; \begin{matrix} \mathbf{0} \\ G_1(t) \end{matrix}\right] \tag{3}$$

where $F_1(t)$ and $\overline{L}_1(t-1)$ are the submatrices obtained after deleting the first row and column of $F(t)$ and $\overline{L}_0(t-1) = \overline{L}(t-1)$, respectively. The first column of the postarray, i.e., the array on the right-hand side of (3) has to be $\bar{l}_0(t)$ by virtue of the identity (2).

Let us now check the significance of the matrix $G_1(t)$ that is produced on the right-hand side of (3). Comparing the $(I_n \oplus J(t))$-norm on both sides we obtain

$$\begin{aligned} F(t)\overline{L}(t-1)\overline{L}^*(t-1)F^*(t) + G(t)J(t)G^*(t) \\ = \bar{l}_0(t)\bar{l}_0^*(t) + \begin{bmatrix} \mathbf{0} \\ F_1(t)\overline{L}_1(t-1) \end{bmatrix} [\mathbf{0}\ \overline{L}_1^*(t-1)F_1^*(t)] \\ + \begin{bmatrix} \mathbf{0} \\ G_1(t) \end{bmatrix} J[\mathbf{0}\quad G_1^*(t)]. \end{aligned}$$

But the Cholesky factor of the Schur complement of $r_{00}(t-1)$ in $R(t-1)$ is $\overline{L}_1(t-1)$ itself (see, e.g., [18], [19]). That is, $R_1(t-1) = \overline{L}_1(t-1)\overline{L}_1^*(t-1)$, where $R_1(t-1)$ denotes the Schur complement. Hence, using (1) we get

$$R(t) - \bar{l}_0(t)\bar{l}_0^*(t) = \begin{bmatrix} 0 & \mathbf{0} \\ \mathbf{0} & F_1(t)R_1(t-1)F_1^*(t) \end{bmatrix} + \begin{bmatrix} 0 & \mathbf{0} \\ \mathbf{0} & G_1(t)J(t)G_1^*(t) \end{bmatrix}.$$

Consequently

$$R_1(t) - F_1(t)R_1(t-1)F_1^*(t) = G_1(t)J(t)G_1^*(t)$$

since

$$R(t) - \bar{l}_0(t)\bar{l}_0^*(t) = \begin{bmatrix} 0 & \mathbf{0} \\ \mathbf{0} & R_1(t) \end{bmatrix}.$$

This shows that $G_1(t)$ is a generator matrix of the Schur complement $R_1(t)$, with respect to the displacement operation $R_1(t) - F_1(t)R_1(t-1)F_1^*(t)$.

We can now proceed by annihilating the first row of $G_1(t)$ via an $(I_{n-1} \oplus J(t))$-unitary transformation $\Gamma_1(t)$

$$\begin{aligned} [F_1(t)\overline{L}_1(t-1) \quad G_1(t)]\Gamma_1(t) \\ = \begin{bmatrix} \bar{l}_1(t) & \mathbf{0} & \mathbf{0} \\ & F_2(t)\overline{L}_2(t-1) & G_2(t) \end{bmatrix}, \end{aligned}$$

where $F_2(t)$ and $\overline{L}_2(t-1)$ are the submatrices obtained after deleting the first row and column of $F_1(t)$ and $\overline{L}_1(t-1)$, respectively, and so on. In summary, each transformation $\Gamma_{i-1}(t)$ produces a generator matrix of the Schur complement $R_i(t) \equiv \left[r_{mj}^{(i)}(t)\right]_{m,j=0}^{n-i-1}$, of the leading $i \times i$ submatrix in $R(t)$, where the successive Schur complements are related as follows

$$\begin{bmatrix} 0 & \mathbf{0} \\ \mathbf{0} & R_{i+1}(t) \end{bmatrix} = R_i(t) - \bar{l}_i(t)\bar{l}_i^*(t).$$

Here $\bar{l}_i(t)$ is equal to $l_i(t)d_i^{-1/2}(t)$ where $l_i(t)$ and $d_i(t)$ denote the first column and the $(0,0)$ entry of $R_i(t)$, respectively. We are thus led to the following algorithm.

Algorithm 2.1 The Cholesky factor of a positive-definite Hermitian matrix $R(t)$ with time-variant Toeplitz-like structure, viz., $R(t) - F(t)R(t-1)F^*(t) = G(t)J(t)G^*(t)$, with $F(t)$ lower triangular, can be time-updated by using the following recursive procedure: start with $F_0(t) = F(t)$, $G_0(t) = G(t)$, $\overline{L}_0(t-1) = \overline{L}(t-1)$ and repeat for $i = 0, 1, \cdots, n-1$:

a) At step i we have $F_i(t)$ and $G_i(t)$. Let $g_i(t)$ denote the first row of $G_i(t)$.
b) Choose a convenient $(I_{n-i} \oplus J(t))$−unitary transformation $\Gamma_i(t)$ that performs the rotation

$$\begin{aligned} \left[f_i(t)d_i^{1/2}(t-1) \quad \mathbf{0} \quad g_i(t)\right]\Gamma_i(t) \\ = \left[d_i^{1/2}(t) \quad \mathbf{0} \quad \mathbf{0}\right]. \end{aligned}$$

c) Applying $\Gamma_i(t)$ to the prearray leads to

$$\begin{aligned} [F_i(t)\overline{L}_i(t-1) \quad G_i(t)]\ \Gamma_i(t) \\ = \begin{bmatrix} \bar{l}_i(t) & \mathbf{0} & \mathbf{0} \\ & F_{i+1}(t)\overline{L}_{i+1}(t-1) & G_{i+1}(t) \end{bmatrix}, \end{aligned} \tag{4}$$

where $F_{i+1}(t)$ and $\overline{L}_{i+1}(t-1)$ are the submatrices obtained by deleting the first row and column of each of $F_i(t)$ and $\overline{L}_i(t-1)$, respectively. Moreover, the matrix $G_{i+1}(t)$ that appears in the postarray is a generator matrix of the $(i+1)$th Schur complement $R_{i+1}(t)$. That is

$$\begin{aligned} R_{i+1}(t) - F_{i+1}(t)R_{i+1}(t-1)F_{i+1}(t) \\ = G_{i+1}(t)J(t)G_{i+1}^*(t). \end{aligned}$$

The column vectors $\{\bar{l}_i(t)\}_{i=0}^{n-1}$ constitute the columns of the Cholesky factor $\overline{L}(t)$. □

In other words, each transformation $\Gamma_i(t)$ is chosen so as to annihilate a row in the postarray (as in (4)). Each such transformation then produces *one* column of the desired Cholesky factor $\overline{L}(t)$. A sequence of k ($k < n$) transformations $\{\Gamma_0(t), \Gamma_1(t), \cdots, \Gamma_{k-1}(t)\}$, would clearly produce the first k columns of $\overline{L}(t)$. More explicitly (we shall use this fact later while deriving a systolic solution for the normal equations)

$$\begin{aligned} [F(t)\overline{L}(t-1) \quad G(t)] \xrightarrow{\Gamma_0(t),\cdots,\Gamma_{k-1}(t)} \\ \begin{bmatrix} & & 0 & \mathbf{0} & \mathbf{0} \\ & & \vdots & \vdots & \vdots \\ \bar{l}_0(t) & \cdots & 0 & \mathbf{0} & \mathbf{0} \\ & & \bar{l}_{k-1}(t) & \mathbf{0} & \mathbf{0} \\ & & & F_k\bar{l}_k(t-1) & G_k(t) \end{bmatrix}. \end{aligned} \tag{5}$$

The overall effect of the sequence of transformations $\Gamma_0(t)$, $\Gamma_1(t), \cdots, \Gamma_{n-1}(t)$, is to produce the block zero entry in the postarray of expression (2), and thus to update the Cholesky factor from time $(t-1)$ to time t.

Finally, observe that $G_i(t)$ has $(n-i)r(t)$ elements as compared to $(n-i)^2$ in $R_i(t)$ and usually $r(t) \ll n$. If the matrix $F(t)$ is sparse enough so that the complexity of computing the matrix-vector product $F(t)\mathbf{x}$ is $O(n)$ operations (multiplications and additions), for any column vector $\mathbf{x}$,

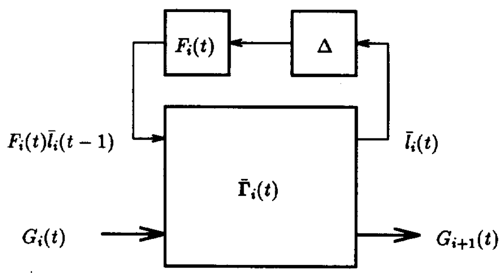


Fig. 1. One step of the generator recursion.

then n steps of the above recursive procedure would require $O(r(t)n^2)$ operations per time step.

III. Triangular Array Implementation

Fig. 1 depicts one step of Algorithm 2.1. Each such step is characterized by a $(1 \oplus J(t))$-unitary transformation $\overline{\Gamma}_i(t)$, and a storage element Δ that stores the present value of $\bar{l}_i(t)$ to the next time instant.

The generator matrix $G_i(t)$ and the column vector $F_i(t)\bar{l}_i(t-1)$ undergo the transformation $\overline{\Gamma}_i(t)$ and yield the next generator $G_{i+1}(t)$, as well as the ith column of the Cholesky factor, $\bar{l}_i(t)$. That is (this is a compact rewriting of (4), where we have dropped the entry $F_{i+1}(t)\overline{L}_{i+1}(t-1)$ that is common to both the pre- and post-arrays)

$$[F_i(t)\bar{l}_i(t-1) \quad G_i(t)]\,\overline{\Gamma}_i(t) = \begin{bmatrix} \bar{l}_i(t) & \begin{matrix} \mathbf{0} \\ G_{i+1}(t) \end{matrix} \end{bmatrix}.$$

The matrix-vector product $F_i(t)\bar{l}_i(t-1)$ indicated in the feedback line in Fig. 1 can be implemented by a time-variant tapped-delay filter. To clarify this, we first remark that the rows of the prearray, $[F_i(t)\bar{l}_i(t-1) \quad G_i(t)]$, are fed one row at a time through $\overline{\Gamma}_i(t)$. Moreover, recall that $F_i(t)$ is a lower triangular matrix whose diagonal entries are $\{f_i(t), f_{i+1}(t), \cdots\}$. We shall denote its off-diagonal elements by $\{\xi_{i+p,j}(t)\}_{p\geq 1, j\geq 0}$

$$F_i(t) = \begin{bmatrix} f_i(t) & & & & \\ \xi_{i+1,0}(t) & f_{i+1}(t) & & \mathbf{O} & \\ \xi_{i+2,0}(t) & \xi_{i+2,1}(t) & f_{i+2}(t) & & \\ \xi_{i+3,0}(t) & \xi_{i+3,1}(t) & \xi_{i+3,2}(t) & f_{i+3}(t) & \\ \vdots & & & & \ddots \end{bmatrix}.$$

If we denote the entries of $\bar{l}_i(t-1)$ by

$$\bar{l}_i(t-1) = [\bar{l}_{i,0}(t-1) \quad \bar{l}_{i,1}(t-1) \quad \cdots]^T$$

then the computation of the elements of the column vector $F_i(t)\bar{l}_i(t-1)$ reduces to inner product evaluations. The jth entry of the resulting column is the inner product of the jth row of $F_i(t)$ with $\bar{l}_i(t-1)$. Hence, the entries of $F_i(t)\bar{l}_i(t-1)$ can be obtained as outputs of time-variant tapped-delay filters, whose coefficients are given by the rows of $F_i(t)$. This is shown in Fig. 2.

The Δ block stores the elements of $\bar{l}_i(t)$ for the next time instant, and multiplication by $F_i(t)$ corresponds to processing by time-variant finite-impulse-response filters whose coefficients

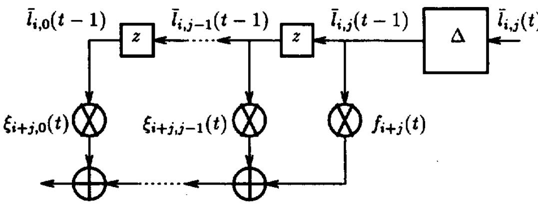


Fig. 2. Time-variant tapped delay line.

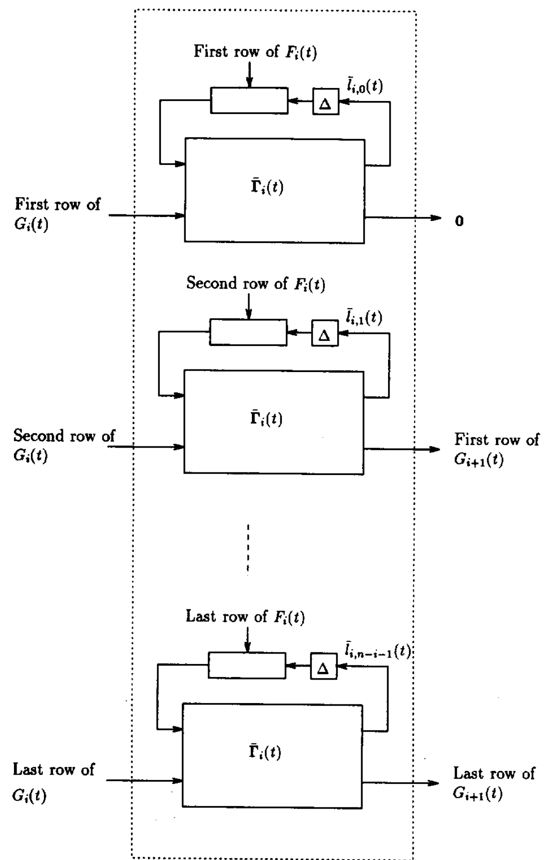


Fig. 3. A layer of elementary sections for one generator step.

vary (for a fixed t) as follows: when the first row of $G_i(t)$ is fed through $\overline{\Gamma}_i(t)$, the filter coefficients are the elements of the first row of $F_i(t)$. When the second row of $G_i(t)$ is fed in, the filter coefficients are the elements of the second row of $F_i(t)$, and so on. More precisely, recall that $G_i(t)$ has $(n-i)$ rows. Hence, we can decompose Fig. 1 into $(n-i)$ elementary sections as shown in Fig. 3.

Each section consists of the same $(1 \oplus J(t))$-unitary transformation $\overline{\Gamma}_i(t)$ and a tapped-delay filter whose coefficients are the corresponding row in the $F_i(t)$ matrix. One row of $G_i(t)$ is applied to each section. The outputs of the layer are then the rows of $G_{i+1}(t)$. Notice that the output of the top section is zero, since each generator step produces one zero row.

If we represent each section in Fig. 3 by a square box, then n steps of the generator recursion of Algorithm 2.1 correspond to a triangular array as depicted in Fig. 4.

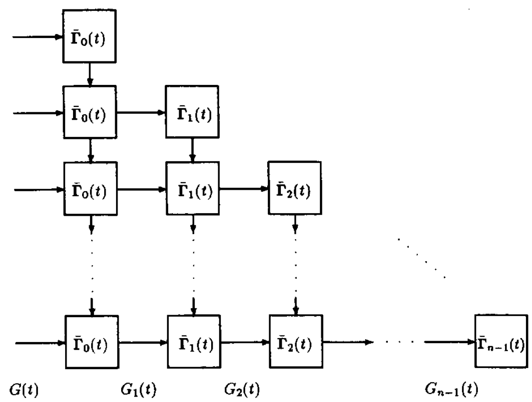


Fig. 4. Triangular/array implementation of the generator recursion.

The first layer of the array operates on the n rows of $G(t)$ and produces the $n-1$ rows of $G_1(t)$. The second layer operates on the rows of $G_1(t)$ and produces the $n-2$ rows of $G_2(t)$, and so on. It is clear that once the rows of $G(t)$ propagate through the first layer, the array can already receive the rows of $G(t+1)$, etc.

Before proceeding further in discussing simplifications of the recursion of Algorithm 2.1, we pause for a while and illustrate the application of this algorithm to an important problem that arises in adaptive filtering. We show that the simple recursive procedure described in Algorithm 2.1 collapses to the well-known QR algorithm for solving the so-called normal equations (see, e.g., ([3], ch. 14), ([4], ch. 5) and promptly yields a *parallel* method for the extraction of the weight vector.

IV. An Application: The QR Adaptive Algorithm

In adaptive filtering one is often faced with the task of solving a linear system of equations of the form

$$\Phi(t)w(t) = \theta(t) \tag{6}$$

where $\Phi(t)$ is an $n \times n$ positive-definite Hermitian matrix usually referred to as the autocorrelation matrix, $\theta(t)$ is an $n \times 1$ column vector known as the cross-correlation vector and $w(t)$ is an $n \times 1$ so-called weight vector. The equations (6) are often termed the *normal equations*. The reader is referred to [3], [4], [20] for more on the derivation and motivation of the normal equations in the context of the recursive least-squares problem.

In adaptive filtering, and in other applications, it often happens that the quantities $\Phi(t)$ and $\theta(t)$ in (6) satisfy the time recursions

$$\Phi(t) - \lambda\Phi(t-1) = \mathbf{u}^*(t)\mathbf{u}(t) \tag{7}$$

$$\theta(t) - \lambda\theta(t-1) = d(t)\mathbf{u}^*(t) \tag{8}$$

where λ is a positive scalar $(0 < \lambda \leq 1)$, $\mathbf{u}(t)$ is a row vector

$$\mathbf{u}(t) = [u_1(t) \quad u_2(t) \quad \cdots \quad u_n(t)]$$

and $d(t)$ is a scalar. In the adaptive applications, one usually encounters the problem of solving (6) for successive time instants $t, t+1, t+2, \cdots$, and one expects a computationally efficient procedure for going through the calculations, $w(t) \to w(t+1) \to w(t+2) \to \cdots$, since the corresponding quantities, $(\Phi(t), \theta(t))$, $(\Phi(t+1), \theta(t+1))$, $(\Phi(t+2), \theta(t+2)), \cdots$ are closely related because of (7) and (8). For example, $\Phi(t+1)$ and $\lambda\Phi(t)$ differ only by a rank-one matrix $\mathbf{u}^*(t+1)\mathbf{u}(t+1)$. We shall now show how to exploit the existing time-variant (or low-rank) structure in a straightforward manner, by using the results developed in the previous sections.

To begin with, observe that $\Phi(t)$ is a time-variant structured matrix and (7) is a special case of (1) with $F(t) = \sqrt{\lambda}I_n$, $G(t) = \mathbf{u}^*(t)$, $\Delta = 1$, and $J(t) = 1$. Hence, we can propagate its Cholesky factor (denoted by $\overline{L}_\Phi(t)$) via the square-root array (recall (2))

$$[\sqrt{\lambda}\overline{L}_\Phi(t-1) \quad \mathbf{u}^*(t)]\Gamma(t) = [\overline{L}_\Phi(t) \quad \mathbf{0}] \tag{9}$$

where $\Gamma(t)$ is any unitary transformation $(\Gamma(t)\Gamma^*(t) = I_{n+1})$ that produces the block zero in the postarray. Expression (9) is the so-called QR recursion that updates the Cholesky factor of the autocorrelation matrix through a sequence of unitary rotations [3], [4], [12]. Once $\overline{L}_\Phi(t)$ is determined, then one way to obtain the weight vector $w(t)$ is to solve the following triangular system (via back-substitution)

$$\overline{L}_\Phi^*(t)w(t) = \overline{L}_\Phi^{-1}(t)\theta(t).$$

This, however, does not yield a fully parallelizable algorithm.

We now extend an embedding technique used in [21] to the time-variant setting and derive a parallel procedure for obtaining $w(t)$ by exploiting the notion of time-variant displacement structure, as introduced in the previous sections. The main point is to start by expressing $w(t)$ as a Schur complement in a suitable block matrix, and then to properly exploit the structure of this matrix.

Consider the following $(2n \times (n+1))$ extended matrix

$$R(t) = \begin{bmatrix} \Phi(t) & \theta(t) \\ I_n & \mathbf{0} \end{bmatrix}$$

and note that the Schur complement of $\Phi(t)$ in $R(t)$ is $-\Phi^{-1}(t)\theta(t)$, which is equal to (minus) the desired weight vector $w(t)$. This already suggests the following route: if we can show how to efficiently go from the Schur complement at time $t-1$ to the Schur complement at time t, then we obtain an efficient procedure for going from $w(t-1)$ to $w(t)$. But this is precisely what is provided by Algorithm 2.1, as we now further elaborate.

We shall, for convenience, redefine $R(t)$ as a $(2n \times 2n)$ *square* matrix

$$R(t) = \begin{bmatrix} \Phi(t) & \theta(t) & \mathbf{0} \\ I_n & \mathbf{0} & \mathbf{0} \end{bmatrix} \tag{10}$$

where the Schur complement of $\Phi(t)$ is now $[-w(t) \quad \mathbf{0}]$, which still completely identifies $w(t)$.

The relations (7) and (8) indicate that the matrix $R(t)$ in (10) is clearly a time-variant Toeplitz-like matrix. In fact, it readily follows that

$$R(t) - \begin{bmatrix} \lambda I_n & \mathbf{0} \\ \mathbf{0} & I_n \end{bmatrix} R(t-1) = \begin{bmatrix} \mathbf{u}^*(t) \\ \mathbf{0} \end{bmatrix} [\mathbf{u}(t) \quad d(t) \quad \mathbf{0}]$$

which is a special case of a *non-Hermitian* time-variant displacement equation of the form

$$R(t) - F(t)R(t-1)A^*(t) = G(t)J(t)B^*(t)$$

with $A(t) = I_{2n}, J(t) = 1, \Delta = 1, F(t) = (\lambda I_n \oplus I_n)$

$$G(t) = \begin{bmatrix} \mathbf{u}^*(t) \\ \mathbf{0} \end{bmatrix} \quad \text{and} \quad B(t)\begin{bmatrix} \mathbf{u}^*(t) \\ d(t) \\ \mathbf{0} \end{bmatrix}.$$

Although $R(t)$ has a non-Hermitian structure, we are only interested in its first n Schur complementation steps (that is, in its first n triangular factors) and the leading $n \times n$ submatrix of $R(t)$ is Hermitian (equal to $\Phi(t)$).

Let us first check the form of the first n triangular factors of $R(t)$. To begin with, observe that the Cholesky factor of $\Phi(t)$ ($\overline{L}_\Phi(t)$) is clearly a part of the first n triangular factors of $R(t)$, since $\Phi(t)$ is a leading submatrix of $R(t)$ in (10). Moreover, we can express $R(t)$ as the product of lower and upper triangular factors, $L(t)$ and $U(t)$ say, $R(t) = L(t)U(t)$, where $L(t)$ and $U(t)$ have the forms

$$L(t) = \begin{bmatrix} \overline{L}_\Phi(t) & \mathbf{0} \\ X & ? \end{bmatrix}, \qquad U(t) = \begin{bmatrix} \overline{L}_\Phi^*(t) & Y \\ \mathbf{0} & ? \end{bmatrix}$$

for some matrices X and Y (the symbol "?" stands for irrelevant entries). We can be more specific about the values of X and Y. By comparing the entries on both sides of the equality

$$\begin{bmatrix} \Phi(t) & \theta(t) & \mathbf{0} \\ I_n & \mathbf{0} & \mathbf{0} \end{bmatrix} = \begin{bmatrix} \overline{L}_\Phi(t) & \mathbf{0} \\ X & ? \end{bmatrix}\begin{bmatrix} \overline{L}_\Phi^*(t) & Y \\ \mathbf{0} & ? \end{bmatrix}$$

we readily conclude that

$$X = \overline{L}_\Phi^{-*}(t), \qquad Y = \begin{bmatrix} \overline{L}_\Phi^{-1}(t)\theta(t) & \mathbf{0} \end{bmatrix}.$$

That is, the first n columns of the factor $L(t)$ and the first n rows of the factor $U(t)$ are completely determined by $\overline{L}_\Phi(t)$ and $\theta(t)$, viz.

$$\begin{bmatrix} \overline{L}_\Phi(t) \\ \overline{L}_\Phi^{-*}(t) \end{bmatrix}, \begin{bmatrix} \overline{L}_\Phi^*(t) & \overline{L}_\Phi^{-1}(t)\theta(t) & \mathbf{0} \end{bmatrix}.$$

We are interested in computing the Schur complement of $\Phi(t)$ in $R(t)$. Hence, we only need to apply the *first* n recursive steps (known as Schur reduction steps) to $R(t)$ (which is $2n \times 2n$) and get its *first* n triangular factors. But since the leading $n \times n$ submatrix of $R(t)$ is $\Phi(t)$ itself, then these *first* n Schur reduction steps can be clearly achieved by using the *same* transformation $\Gamma(t)$ as in (9), which rotates $\overline{L}_\Phi(t-1)$ into $\overline{L}_\Phi(t)$. That is (recall (5) and the discussion preceding it)

$$\begin{bmatrix} \sqrt{\lambda}\overline{L}_\Phi(t-1) & \mathbf{u}^*(t) \\ \frac{1}{\sqrt{\lambda}}\overline{L}_\Phi^{-*}(t-1) & \mathbf{0} \end{bmatrix}\Gamma(t) = \begin{bmatrix} \overline{L}_\Phi(t) & \mathbf{0} \\ \overline{L}_\Phi^{-*}(t) & \overline{\Delta w}(t) \end{bmatrix} \tag{11}$$

$$\begin{bmatrix} \sqrt{\lambda}\overline{L}_\Phi(t-1) & \mathbf{u}^*(t) \\ \sqrt{\lambda}\theta^*(t-1)\overline{L}_\Phi^{-*}(t-1) & d^*(t) \\ \mathbf{0} & \mathbf{0} \end{bmatrix}\Gamma(t) = \begin{bmatrix} \overline{L}_\Phi(t) & \mathbf{0} \\ \theta^*(t)\overline{L}_\Phi^{-*}(t) & \overline{e}(t) \\ \mathbf{0} & \mathbf{0} \end{bmatrix} \tag{12}$$

where we designated the resulting entries in the postarrays by $\overline{\Delta w}(t)$ and $\overline{e}(t)$. Moreover, it is easy to check (by multiplying both arrays and comparing entries, as we did while checking the significance of $G_1(t)$ after (3) in Section II) that

$$-[w(t) \quad 0] + [w(t-1) \quad 0] = \overline{\Delta w}(t)[\overline{e}^*(t) \quad \mathbf{0}].$$

This is the (non-Hermitian) analog of the statement given in Algorithm 2.1 (and (5)) that

$$\overline{\Delta w}(t), \quad \begin{bmatrix} \overline{e}(t) \\ \mathbf{0} \end{bmatrix}$$

are generator matrices of the Schur complement of $\Phi(t)$ in $R(t)$, viz., $[-w(t) \quad 0]$. Therefore, $w(t) - w(t-1) = -\overline{\Delta w}(t)\overline{e}^*(t)$. Moreover, the arrays (11) and (12) can be compactly grouped together as

$$\begin{bmatrix} \sqrt{\lambda}\overline{L}_\Phi(t-1) & \mathbf{u}^*(t) \\ \sqrt{\lambda}\theta^*(t-1)\overline{L}_\Phi^{-*}(t-1) & d^*(t) \\ \frac{1}{\sqrt{\lambda}}\overline{L}_\Phi^{-*}(t-1) & \mathbf{0} \end{bmatrix}\Gamma(t) = \begin{bmatrix} \overline{L}_\Phi(t) & \mathbf{0} \\ \theta^*(t)\overline{L}_\Phi^{-*}(t) & \overline{e}(t) \\ \overline{L}_\Phi^{-*}(t) & \overline{\Delta w}(t) \end{bmatrix} \tag{13}$$

which constitute the desired algorithm for propagating the Cholesky factor $\overline{L}_\Phi(t)$ as well as the weight update vector $\overline{\Delta w}(t)$. It can be verified that the quantity $\overline{e}(t)$ is a normalized version of the so-called *a posteriori* error [3], [4], [8]. In summary, we are led to the following array algorithm.

Algorithm 4.1 The solution of the normal equations (6) that arise in the recursive-least squares problem can be recursively updated by using the array equation (13), where $\Gamma(t)$ is any unitary matrix that produces the zero block in the postarray and $w(t) = w(t-1) - \overline{\Delta w}(t)\overline{e}^*(t)$. □

The first two block lines of array (13) constitute the so-called QR algorithm of the adaptive filtering literature (see, e.g ([3], ch. 14), ([4], ch. 5). This has been for some time the main tool for solving the normal equations, where the weight vector $w(t)$ is then obtained by solving the linear system $\overline{L}_\Phi^*(t)w(t) = \overline{L}_\Phi^{-1}(t)\theta(t)$ via back-substitution. McWhirter [13], [14] introduced an alternative approach for determining $w(t)$ by exploiting the fact that the normalized error $\overline{e}(t)$ also appears in the postarray. However, his solution requires "freezing" the triangular array at each instant. Alternative recent systolic derivations have been proposed in [22], [23]. We further remark that the complete QR array, as stated in Algorithm 4.1, was also independently derived by Yang and Böhme [24], who suggested adding the third block line

$$\begin{bmatrix} \frac{1}{\sqrt{\lambda}}\overline{L}_\Phi^{-*}(t-1) & \mathbf{0} \end{bmatrix}\Gamma(t) = \begin{bmatrix} \overline{L}_\Phi^{-*}(t) & \overline{\Delta w}(t) \end{bmatrix}$$

to the QR array, but without *a priori* motivation (in their approach, the arrays are not introduced directly as in our method, but are inferred from an explicit set of equations describing the QR algorithm). Our derivation gives directly all three lines of the array and makes clear the significance of each line, as depicted in Fig. 5.

The entries of the column vector $\mathbf{u}^*(t)$ are rotated along with $\sqrt{\lambda}\overline{L}_\Phi(t-1)$ into zero. This updates the Cholesky factor into $\overline{L}_\Phi(t)$, which is stored in the left triangular array for the next time instant. The right triangular array rotates the zero

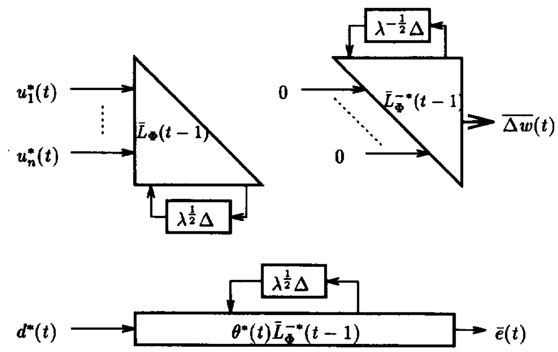


Fig. 5. Block diagram of the complete QR array algorithm.

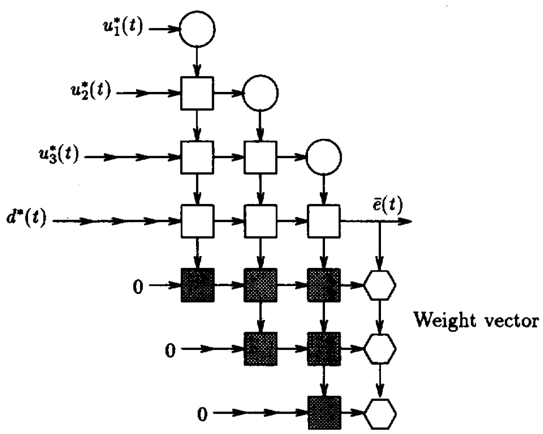


Fig. 6. A systolic implementation of the QR algorithm.

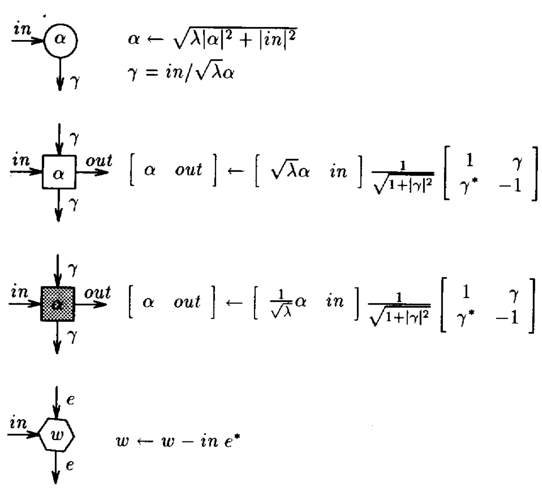


Fig. 7. Functional descriptions of the cells in the systolic array.

vector along with $\frac{1}{\sqrt{\lambda}}\overline{L}_\Phi^{-*}(t-1)$ into the weight increment $\overline{\Delta w}(t)$ and updates the inverse of the Cholesky factor. The last (line) array rotates $d^*(t)$ along with $\sqrt{\lambda}\theta^*(t-1)\overline{L}_\Phi^{-*}(t-1)$ into the normalized error $\overline{e}(t)$, and updates into $\theta^*(t)\overline{L}_\Phi^{-*}(t)$. A more detailed description is shown in Figs. 6 and 7.

The approach presented here of the extended QR algorithm treats the problem as a special case of Algorithm 2.1 and we feel provides more insight into the significance of the arrays. A natural extension of the theory discussed here is to consider normal equations as in (6), where $\Phi(t)$ and $\theta(t)$ are allowed to exhibit a more general time-variant structure of the form (compare with (7) and (8))

$$\Phi(t) - F(t)\Phi(t-1)F^*(t) = G(t)J(t)G^*(t)$$
$$\theta(t) - F(t)\theta(t-1) = G(t)J(t)D^*(t)$$

for some lower triangular matrix $F(t)$, a generator matrix $G(t)$, a signature matrix $J(t)$ (e.g., $J(t) = I$), and a data row vector $D(t)$. It is straightforward to check that the corresponding array algorithm is of the form

$$\begin{bmatrix} F(t)\overline{L}_\Phi(t-1) & G(t) \\ \theta^*(t-1)\overline{L}_\Phi^{-*}(t-1) & D(t) \\ F^{-*}(t)\overline{L}_\Phi^{-*}(t-1) & \mathbf{0} \end{bmatrix}\Gamma(t) = \begin{bmatrix} \overline{L}_\Phi(t) & \mathbf{0} \\ \theta^*(t)\overline{L}_\Phi^{-*}(t) & \overline{E}(t) \\ \overline{L}_\Phi^{-*}(t) & \overline{\Delta W}(t) \end{bmatrix}$$

where $\Gamma(t)$ is an arbitrary $(I_n \oplus J(t))$-unitary transformation that produces the block zero entry in the post-array, $\overline{E}(t)$ is a row vector, $\overline{\Delta W}(t)$ is a matrix, and

$$w(t) - F^{-*}(t)w(t-1) = -\overline{\Delta W}(t)J(t)\overline{E}^*(t).$$

V. Time-Variant Lattice Implementations

We now return to the recursions of Algorithm 2.1 and examine more closely some computational issues. In particular, we show that the general algorithm can be implemented in a time-variant lattice form, by invoking the notion of proper generators.

Referring to Fig. 1 we recall that each generator step is characterized by a $(1 \oplus J(t))$-unitary transformation $\overline{\Gamma}_i(t)$, chosen so as to perform the following rotation

$$\left[f_i(t)d_i^{1/2}(t-1) \quad g_i(t)\right]\overline{\Gamma}_i(t) = \left[d_i^{1/2}(t) \quad \mathbf{0}\right]. \qquad (14)$$

Here $d_i^{1/2}(t-1)$ is the top entry of the ith column $\overline{l}_i(t-1)$. That is, $\overline{\Gamma}_i(t)$ annihilates the row vector $g_i(t)$. Now there are many possible choices for achieving such a transformation, and we describe here one such possibility (among *many* others). First note that the row vector $g_i(t)$ can have positive, negative, or zero $J(t)$-norm, which follows from the equality

$$d_i(t) - |f_i(t)|^2 d_i(t-1) = g_i(t)J(t)g_i^*(t).$$

In other words, the sign of the quantity $g_i(t)J(t)g_i^*(t)$ depends on the sign of the difference $d_i(t) - |f_i(t)|^2 d_i(t-1)$. Hence, $g_i(t)J(t)g_i^*(t)$ is sign indefinite and we have to consider three possible cases while looking for a suitable rotation $\overline{\Gamma}_i(t)$.

A. Positive $J(t)$-Norm

Assume $g_i(t)J(t)g_i^*(t) > 0$, then it follows from Lemma 1.1 that we can always choose a $J(t)$-unitary matrix $\Theta_i(t)$ that reduces $g_i(t)$ to the form

$$g_i(t)\Theta_i(t) = [\delta_i(t) \quad 0 \quad \cdots \quad 0] \qquad (15)$$

where $\delta_i(t)$ is a positive scalar. By comparing the $J(t)-$ norm on both sides of (15) we conclude that the value of $\delta_i(t)$ is $\delta_i(t) = \sqrt{g_i(t)J(t)g_i^*(t)}$. Hence, the effect of $\Theta_i(t)$ is to reduce the generator $G_i(t)$ to the form

$$G_i(t)\Theta_i(t) = \begin{bmatrix} \delta_i(t) & 0 & \cdots & 0 \\ x & x & x & x \\ \vdots & \vdots & \vdots & \vdots \\ x & x & x & x \end{bmatrix} \equiv \overline{G}_i(t)$$

where the first row of $\overline{G}_i(t)$ lies along the basis vector $[1 \quad 0 \quad \cdots \quad 0]$. Clearly, $\overline{G}_i(t)$ is also a generator of $R_i(t)$ since $G_i(t)J(t)G_i^*(t) = \overline{G}_i(t)J(t)\overline{G}_i^*(t)$. We say that $\overline{G}_i(t)$ is a *proper* generator of $R_i(t)$ and $\Theta_i(t)$ is a $J(t)$-unitary rotation that transforms $G_i(t)$ to proper form. This can be achieved in a variety of ways: by using a sequence of elementary Givens and hyperbolic rotations [18], Householder transformations [26]–[28], etc. Referring to (14), we see that the effect of $\Theta_i(t)$ is the following

$$\begin{aligned} \left[f_i(t)d_i^{1/2}(t-1) \quad g_i(t)\right] \begin{bmatrix} 1 & \mathbf{0} \\ \mathbf{0} & \Theta_i(t) \end{bmatrix} \\ = \left[f_i(t)d_i^{1/2}(t-1) \quad \delta_i(t) \quad \mathbf{0}\right] \quad (16) \end{aligned}$$

where we still need to annihilate the nonzero entry $\delta_i(t)$. This can be achieved by using a 2×2 elementary unitary rotation $U_i(t)$

$$\begin{aligned} \left[f_i(t)d_i^{1/2}(t-1) \quad \delta_i(t) \quad \mathbf{0}\right] \begin{bmatrix} U_i(t) & \mathbf{0} \\ \mathbf{0} & I_{(r(t)-1)} \end{bmatrix} \\ = \left[d_i^{1/2}(t) \quad \mathbf{0}\right]. \end{aligned}$$

Hence, we can implement $\overline{\Gamma}_i(t)$ as the sequence of two transformations $\Theta_i(t)$ and $U_i(t)$

$$\overline{\Gamma}_i(t) = \begin{bmatrix} 1 & \mathbf{0} \\ \mathbf{0} & \Theta_i(t) \end{bmatrix} \begin{bmatrix} U_i(t) & \mathbf{0} \\ \mathbf{0} & I_{(r(t)-1)} \end{bmatrix}$$

and we thus have

$$\begin{aligned} \left[F_i(t)\bar{l}_i(t-1) \quad G_i(t)\right] \begin{bmatrix} 1 & \mathbf{0} \\ \mathbf{0} & \Theta_i(t) \end{bmatrix} \\ = \left[F_i(t)\bar{l}_i(t-1) \quad \overline{G}_i(t)\right] \end{aligned}$$

and

$$\begin{aligned} \left[F_i(t)\bar{l}_i(t-1) \quad \overline{G}_i(t)\right] \begin{bmatrix} U_i(t) & \mathbf{0} \\ \mathbf{0} & I_{(r(t)-1)} \end{bmatrix} \\ = \begin{bmatrix} \bar{l}_i(t) & \mathbf{0} \\ & G_{i+1}(t) \end{bmatrix}. \end{aligned}$$

This is graphically depicted in Fig. 8. The generator $G_i(t)$ is transformed to proper form by $\Theta_i(t)$. The last columns of $\overline{G}_i(t)$ are kept unchanged and constitute the last columns of $G_{i+1}(t)$, while the first column of $\overline{G}_i(t)$ is rotated with $F_i(t)\bar{l}_i(t-1)$ (by $U_i(t)$) in order to yield a zero row.

We mentioned earlier that $\Theta_i(t)$ can be implemented in a variety of ways. We can also give a global expression for $\Theta_i(t)$. Since $g_i(t)$ has positive $J(t)$-norm, then at least one of the first $p(t)$ entries of $g_i(t)$ is nonzero. Hence, we can always assume that the leftmost entry is nonzero, by choosing a convenient $J(t)$-unitary rotation $P_i(t)$, for instance, and

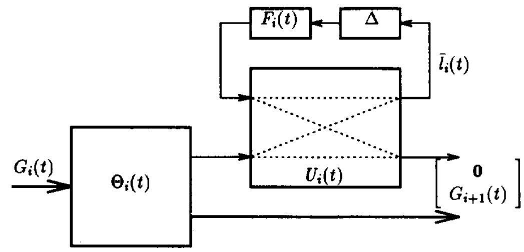


Fig. 8. A positive proper-step of the generator recursion.

by using $G_i(t)P_i(t)$ instead of $G_i(t)$. A global expression for $\Theta_i(t)$ is given as follows: partition the signature matrix $J(t)$ and the first row $g_i(t)$ into $J(t) = (1 \oplus -E(t))$, $E(t) = (-I_{(p(t)-1)} \oplus I_{q(t)})$ and $g_i(t) = [g_{i0}(t) \quad \overline{g}_i(t)]$, $g_{i0}(t)$ is a scalar and $\overline{g}_i(t)$ is a $1 \times (r(t)-1)$ row vector. If we define the row vector (also known as Schur parameter or reflection coefficient), $\gamma_i(t) = g_{i0}^{-1}(t)\overline{g}_i(t)E(t)$, then $\Theta_i(t)$ can be expressed in the form [25]

$$\Theta_i(t) = \begin{bmatrix} 1 & -\gamma_i(t) \\ -\gamma_i^*(t) & E(t) \end{bmatrix} \begin{bmatrix} \sigma_i(t) & \mathbf{0} \\ \mathbf{0} & \Lambda_i(t) \end{bmatrix}$$

where $\sigma_i(t) = (1 - \gamma_i(t)E(t)\gamma_i^*(t))^{-1/2}$, and $\Lambda_i(t)$ is an $(r(t)-1) \times (r(t)-1)$ matrix that satisfies $\Lambda_i(t)E(t)\Lambda_i^*(t) = (E(t) - \gamma_i^*(t)\gamma_i(t))^{-1}$. It follows from the positive definiteness of $R(t)$ that $(1 - \gamma_i(t)E(t)\gamma_i^*(t)) > 0$ and hence, $\sigma_i(t)$ is well-defined.

B. Negative $J(t)$-Norm

Assume $g_i(t)J(t)g_i^*(t) < 0$, then it follows from Lemma 1.1 that we can always choose a $J(t)$-unitary matrix $\Theta_i(t)$ that reduces $g_i(t)$ to the form

$$g_i(t)\Theta_i(t) = [0 \quad \cdots \quad 0 \quad \delta_i(t)] \quad (17)$$

where $\delta_i(t)$ is a positive scalar. By comparing the $J(t)$-norm on both sides of (17) we conclude that the value of $\delta_i(t)$ is $\delta_i(t) = \sqrt{-g_i(t)J(t)g_i^*(t)}$. Hence, the action of $\Theta_i(t)$ is to reduce the generator $G_i(t)$ to the form

$$G_i(t)\Theta_i(t) = \begin{bmatrix} 0 & \cdots & 0 & \delta_i(t) \\ x & x & x & x \\ \vdots & \vdots & \vdots & \vdots \\ x & x & x & x \end{bmatrix} \equiv \overline{G}_i(t)$$

where the first row of $\overline{G}_i(t)$ lies along the basis vector $[0 \quad \cdots \quad 0 \quad 1]$. Clearly, $\overline{G}_i(t)$ is also a generator of $R_i(t)$ since $G_i(t)J(t)G_i^*(t) = \overline{G}_i(t)J(t)\overline{G}_i^*(t)$. We also say that $\overline{G}_i(t)$ is a *proper* generator of $R_i(t)$, and $\Theta_i(t)$ is a $J(t)$-unitary rotation that transforms $G_i(t)$ to proper form. We see that the effect of $\Theta_i(t)$ is the following

$$\begin{aligned} \left[f_i(t)d_i^{1/2}(t-1) \quad g_i(t)\right] \begin{bmatrix} 1 & \mathbf{0} \\ \mathbf{0} & \Theta_i(t) \end{bmatrix} \\ = \left[f_i(t)d_i^{1/2}(t-1) \quad \mathbf{0} \quad \delta_i(t)\right] \end{aligned}$$

where we still need to annihilate the nonzero entry $\delta_i(t)$. This can be achieved by using an elementary hyperbolic rotation

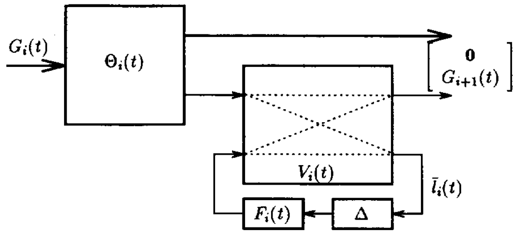


Fig. 9. A negative proper-step of the generator recursion.

$V_i(t)$

$$[f_i(t)d_i^{1/2}(t-1) \quad \delta_i(t)]\begin{bmatrix} \mathrm{ch}_i(t) & -\mathrm{sh}_i(t) \\ -\mathrm{sh}_i^*(t) & \mathrm{ch}_i(t)\end{bmatrix} = [d_i^{1/2}(t) \quad \mathbf{0}]$$

where we denoted the rotation parameters of $V_i(t)$ by

$$V_i(t) = \begin{bmatrix} \mathrm{ch}_i(t) & -\mathrm{sh}_i(t) \\ -\mathrm{sh}_i^*(t) & \mathrm{ch}_i(t)\end{bmatrix}.$$

This discussion shows that we can also implement $\overline{\Gamma}_i(t)$ as the sequence of two transformations $\Theta_i(t)$ and $V_i(t)$, as shown in Fig. 9. The generator $G_i(t)$ is transformed to proper form by $\Theta_i(t)$. The first columns of $\overline{G}_i(t)$ are kept unchanged and constitute the first columns of $G_{i+1}(t)$, while the last column of $\overline{G}_i(t)$ is rotated with $F_i(t)\bar{l}_i(t-1)$ (by $V_i(t)$) in order to yield a zero row.

We can also write down, as in the previous section, a global expression for $\Theta_i(t)$: define $E(t)$ by $J(t) = E(t) \oplus -1$ and partition $g_i(t) = [\bar{g}_i(t) \quad g_{i,r(t)-1}(t)]$, where $\bar{g}_i(t)$ is a $1 \times (r(t)-1)$ row vector. The row vector $\gamma_i(t)$ defined by $\gamma_i(t) = g_{i,r(t)-1}^{-1}(t)\bar{g}_i(t)E(t)$ is called a time-variant Schur parameter (or reflection coefficient). The transformation $\Theta_i(t)$ is then given by

$$\Theta_i(t) = \begin{bmatrix} E(t) & -\gamma_i^*(t) \\ -\gamma_i(t) & 1\end{bmatrix}\begin{bmatrix} \Lambda_i(t) & \mathbf{0} \\ \mathbf{0} & \sigma_i(t)\end{bmatrix}$$

where $\sigma_i(t) = (1 - \gamma_i(t)E(t)\gamma_i^*(t))^{-1/2}$ and $\Lambda_i(t)$ is an $(r(t)-1) \times (r(t)-1)$ matrix that satisfies $\Lambda_i(t)E(t)\Lambda_i(t)^* = (E(t) - \gamma_i^*(t)\gamma_i(t))^{-1}$.

C. Zero $J(t)$-Norm

If $g_i(t)$ has zero $J(t)$-norm then $f_i(t)d_i^{1/2}(t-1)$ is necessarily nonzero, since $d_i(t) > 0$. Therefore, we can use $f_i(t)d_i^{1/2}(t-1)$ as a pivot element in order to annihilate all the entries of $g_i(t)$ in (14).

We again remark that the implementation of $\overline{\Gamma}_i(t)$ is highly nonunique, and that the discussion in the last subsections reveals one such possible implementation that leads to simple time-variant lattice sections as depicted in Figs. 8 and 9.

VI. Two Special Cases

We now present two examples that correspond to special cases of the theory developed so far. The first example assumes a strictly lower triangular matrix $F(t)$, which leads to further simplifications in the lattice picture. The second example is a time-variant extension of the classical Schur algorithm.

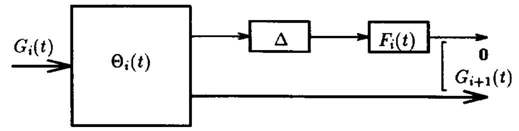


Fig. 10. A proper-step with $F(t)$ strictly lower triangular.

A. Strictly Lower Triangular $F(t)$

When $F(t)$ is strictly lower triangular, the diagonal entries $\{f_i(t)\}$ are zero and consequently

$$d_i(t) = g_i(t)J(t)g_i^*(t).$$

Hence, $g_i(t)$ necessarily has positive $J(t)$-norm since $d_i(t) > 0$ due to the positive-definiteness of $R(t)$. We are thus reduced to the special case studied in Section V, which corresponds to positive $J(t)$-norm: we first choose a $J(t)$-unitary matrix $\Theta_i(t)$ that reduces $g_i(t)$ to proper form, viz.

$$g_i(t)\Theta_i(t) = [\delta_i(t) \quad 0 \quad \cdots \quad 0].$$

That is, the effect of $\Theta_i(t)$ is the following, where we use $f_i(t) = 0$ (compare with (16))

$$[0 \quad g_i(t)]\begin{bmatrix} 1 & \mathbf{0} \\ \mathbf{0} & \Theta_i(t)\end{bmatrix} = [0 \quad \delta_i(t) \quad \mathbf{0}].$$

We still need to annihilate the nonzero entry $\delta_i(t)$. This can now be achieved by simply permuting the first two columns of the postarray

$$[0 \quad \delta_i(t) \quad \mathbf{0}]\begin{bmatrix} 0 & 1 & \mathbf{0} \\ 1 & 0 & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & I_{(r(t)-1)}\end{bmatrix} = [\delta_i(t) \quad 0 \quad \mathbf{0}]$$

which readily leads to the simple lattice picture shown in Fig. 10.

The generator $G_i(t)$ is transformed to proper form by $\Theta_i(t)$. The last columns of $\overline{G}_i(t)$ are kept unchanged and constitute the last columns of $G_{i+1}(t)$, while the first column of $\overline{G}_i(t-1)$, which has been stored in Δ, is multiplied by $F_i(t)$. This can be compactly expressed as follows

$$\begin{bmatrix} \mathbf{0} \\ G_{i+1}(t)\end{bmatrix} = F_i(t)G_i(t-1)\Theta_i(t-1)\begin{bmatrix} 1 & \mathbf{0} \\ \mathbf{0} & \mathbf{0}\end{bmatrix} + G_i(t)\Theta_i(t)\begin{bmatrix} 0 & \mathbf{0} \\ \mathbf{0} & I_{(r(t)-1)}\end{bmatrix}$$

which has the the following interpretation: multiply $G_i(t)$ by $\Theta_i(t)$ and keep the last columns; multiply the first column of $G_i(t-1)\Theta_i(t-1)$ by $F_i(t)$; these two steps result in $G_{i+1}(t)$.

B. A Time-Variant Schur Algorithm

We now consider a special time-variant Toeplitz-like structure that corresponds to $r(t) = 2$, $J(t) = J = (1 \oplus -1)$, and $F(t)$ is the lower triangular shift matrix Z with ones on the first subdiagonal and zeros elsewhere. We denote the columns of $G_i(t)$ by $\mathbf{u}_i(t)$ and $\mathbf{v}_i(t)$, viz.

$$G_i(t) = [\mathbf{u}_i(t) \quad \mathbf{v}_i(t)] \equiv \begin{bmatrix} u_{ii}(t) & v_{ii}(t) \\ u_{i+1,i}(t) & v_{i+1,i}(t) \\ u_{i+2,i}(t) & v_{i+2,i}(t) \\ \vdots & \vdots \end{bmatrix}.$$

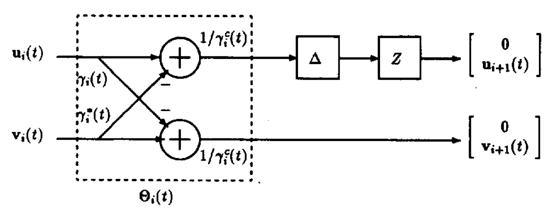


Fig. 11. One step of the time-variant Schur algorithm.

Under these conditions, we are reduced to the following time-variant structure

$$R(t) - ZR(t-1)Z^* = G(t)JG^*(t)$$

which should be compared with the *time-invariant* counterpart that is usually considered during the study of the classical Schur algorithm (see, e.g., [2], [19]), viz.

$$R - ZRZ^* = GJG^*.$$

We can verify easily that the array picture discussed in Section V.A reduces to a simple algorithm, which is a straightforward generalization of the time-invariant counterpart [2]: choose $\Theta_i(t)$ such that $g_i(t)$ is reduced to the form (observe that we now always have $g_i(t)Jg_i^*(t) = d_i(t) > 0$) $g_i(t)\Theta_i(t) = [\delta_i(t) \quad 0]$, where $\delta_i(t)$ is a scalar. A possible choice for $\Theta_i(t)$ is the following hyperbolic rotation: let $\gamma_i(t) = v_{ii}(t)/u_{ii}(t)$ then

$$\Theta_i(t) = \frac{1}{\sqrt{1-|\gamma_i(t)|^2}}\begin{bmatrix} 1 & -\gamma_i(t) \\ -\gamma_i^*(t) & 1 \end{bmatrix}.$$

The array picture of the generator recursion is then given by

$$\begin{bmatrix} \mathbf{0} \\ G_{i+1}(t) \end{bmatrix} = ZG_i(t-1)\Theta_i(t-1)\begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix} + G_i(t)\Theta_i(t)\begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix}$$

which has the the following simple array interpretation: multiply $G_i(t)$ by $\Theta_i(t)$ and keep the second column; shift down the first column of $G_i(t-1)\Theta_i(t-1)$; these two steps result in $G_{i+1}(t)$.

This is depicted in Fig. 11 where we defined $\gamma_i^c(t) = \sqrt{1-|\gamma_i(t)|^2}$. The block with a Δ represents a storage element where the first column of $G_i(t)\Theta_i(t)$ is stored for the next time instant. The entries of the first column of $G_i(t)$ propagate through the top line, while the entries of the second column propagate through the bottom line.

VII. Concluding Remarks

We extended the notion of displacement structure to the time-variant setting and used it to obtain a fast recursive algorithm for finding the triangular (Cholesky) factors of matrices with a general time-variant displacement structure. We also presented a triangular array implementation of the recursive algorithm. An application was made to an important special case that arises in adaptive filtering. We showed that in this case the algorithm collapsed to the widely known QR algorithm with the additional ingredient of providing a parallel procedure for extracting the weight vector. Further applications of the general displacement structure considered here to the study of lossless time-variant systems, to time-variant interpolation problems, and to matrix completion problems will be presented elsewhere, though see [7]–[10].

References

[1] T. Kailath, "From Kalman filtering to innovations, martingales, scattering and other nice things," *Mathematical System Theory: The Influence of R. E. Kalman*, A. C. Antoulas, ed., pp. 55–88, Berlin: Springer-Verlag, 1991.

[2] T. Kailath, "Signal processing applications of some moment problems," *Moments in Mathematics*, H. Landau, ed., American Mathematical Society, vol. 37, pp. 71–109, 1987.

[3] S. Haykin, *Adaptive Filter Theory*, NJ: Prentice Hall, second edition, 1991.

[4] J. G. Proakis, C. M. Rader, F. Ling and C. L. Nikias, *Advanced Digital Signal Processing*, NY: Macmillan Publishing Co., 1992.

[5] A. H. Sayed and T. Kailath, "A state-space approach to adaptive filtering," *Proc. ICASSP*, vol. 3, pp. 559–562, Minneapolis, MN, April, 1993.

[6] H. Sakai, "A vectorized systolic array for block RLS using inverse factorizations," in *Proc. ICASSP*, vol. 4, pp. 233–236, San Francisco, CA, March, 1992.

[7] A. H. Sayed, T. Constantinescu and T. Kailath, "Lattice structures for time-variant interpolation problems," *Proc. IEEE Conference on Decision and Control*, vol. 1, pp. 116–121, Tucson, Arizona, December, 1992.

[8] A. H. Sayed, "Displacement Structure in Signal Processing and Mathematics." *Ph.D. Thesis*, Stanford University, Stanford, CA, August 1992.

[9] A. H. Sayed, T. Constantinescu and T. Kailath, "Square-root algorithms for structured matrices, interpolation, and completion problems," to appear in the series *IMA volumes in Mathematics and Its Applications*, Springer-Verlag.

[10] A. H. Sayed, T. Constantinescu and T. Kailath, "Time-variant displacement structure and interpolation problems," to appear in *IEEE Transactions on Automatic Control*.

[11] T. Constantinescu, A. H. Sayed and T. Kailath, "Displacement structure and completion problems," to appear in *SIAM J. Matrix. Anal. Appl.*

[12] G. H. Golub, "Numerical methods for solving linear least squares problems," *Numer. Math.*, vol. 7, pp. 206–216, 1965.

[13] J. G. McWhirter, "Recursive least squares minimization using a systolic array," in *Proc. SPIE Int. Soc. Opt. Eng.*, vol. 431, pp. 105–112, 1983.

[14] J. G. McWhirter and T. J. Shepherd, "Systolic array processor for MVDR beamforming," *IEE Proceedings*, Pt. F., vol. 136, no. 2, pp. 75–80, April, 1989.

[15] M. Morf and T. Kailath, "Square root algorithms for least squares estimation," *IEEE Transactions on Automatic Control*, vol. AC-20, no. 4, pp. 487–497, August, 1975.

[16] R. Onn, A. O. Steinhardt and A. W. Bojanczyk, "The Hyperbolic singular value decomposition and applications," *IEEE Transactions on Signal Processing*, vol. SP-39, no. 7, pp. 1575–1588, July, 1991.

[17] R. Ackner, "Fast Algorithms for Indefinite Matrices and Meromorphic Functions." *Ph.D. Thesis*, Stanford University, Stanford, CA, December, 1991.

[18] G. H. Golub and C. F. Van Loan, *Matrix Computations*, Baltimore: The Johns Hopkins University Press, second edition, 1989.

[19] H. Lev-Ari and T. Kailath, "Triangular factorization of structured Hermitian matrices," *Operator Theory: Advances and Applications*, vol. 18, pp. 301–324, 1986.

[20] M. L. Honig and D. G. Messerschmitt, *Adaptive Filters-Structures, Algorithms and Applications*, Kluwer Academic Publishers, 1984.

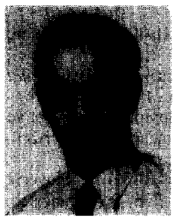
[21] T. Kailath and J. Chun, "Generalized displacement structure for block-Toeplitz, Toeplitz-block and Toeplitz-derived matrices," *SIAM J. Matrix Anal. Appl.*, vol. 15, no. 1, pp. 114-128, Jan. 1994.

[22] S. T. Alexander and A. L. Ghirnikar, "A method for recursive least squares filtering based upon an inverse QR decomposition," *IEEE Transactions on Signal Processing*, vol. SP-41, no. 1, pp. 20–30, January, 1993.

[23] M. Moonen and J. Vandewalle, "A systolic array for recursive least squares computations," *IEEE Transactions on Signal Processing*, vol. SP-41, no. 2, pp. 906–912, February, 1993.

[24] B. Yang and J. F. Böhme, "Rotation-based RLS algorithms: Unified derivations, numerical properties, and parallel implementations," *IEEE Transactions on Signal Processing*, vol. SP-40, no. 5, pp. 1151–1167, May, 1992.

[25] H. Lev-Ari and T. Kailath, "Lattice filter parametrization and modeling of nonstationary processes," *IEEE Transactions on Information Theory*, vol. IT-30, no. 1, pp. 2–16, January, 1984.
[26] C. M. Rader and A. O. Steinhardt, Hyperbolic Householder transformations. *IEEE Transactions on Acoustics, Speech and Signal Processing*, vol. ASSP-34, no. 6, pp. 1589–1602, December, 1986.
[27] A. W. Bojanczyk and A. O. Steinhardt. "Stabilized Hyperbolic Householder transformations," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. ASSP-37, no. 8, pp. 1286–1288, August, 1989.
[28] G. Cybenko and M. Berry, "Hyperbolic Householder algorithms for factoring structured matrices," *SIAM J. Matrix Anal. Appl.*, vol. 11, no. 4, pp. 499–520, October 1990.



Ali H. Sayed (S'90–M'92) was born in São Paulo, Brazil, in 1963. In 1981 he graduated in first place in the National Lebanese Baccalaureat, obtaining the highest scores in the history of the examination. In 1989 he received, with distinction, the M.S. degree in electrical engineering from the University of São Paulo, São Paulo, Brazil, and in 1992 he received the Ph.D. in electrical engineering from Stanford University, Stanford, CA.

From September 1992 to August 1993, he was a Research Associate with the Information Systems Laboratory at Stanford University. He is currently an Assistant Professor in the Department of Electrical and Computer Engineering, University of California, Santa Barbara. His current research interests are in the areas of adaptive and stochastic signal processing, robust filtering and control, and structured computations in systems and mathematics.

Hanoch Lev-Ari (S'78–M'84) received the B.Sc. (summa cum laude) and the M.Sc. in 1978 from the Technion, Israel Institute of Technology, Haifa, Israel, in 1971 and 1978, respectively, both in electrical engineering.

He served in the Israel Defense Forces from 1971 to 1976 as an Electronic Systems Engineer. In 1985 he held a joint appointment as an Adjunct Research Professor of Electrical Engineering at the Naval Postgraduate School, Monterey, CA, and as a Research Associate with the Information Systems Laboratory, Stanford University, where he remained as a Senior Research Associate until 1990. He is currently an Associate Professor with the Department of Electrical and Computer Engineering, Northeastern University, Boston, MA. His present areas of interest include model-based spectrum analysis and estimation for nonstationary signals; wavelet representation of random signals; and adaptive linear and nonlinear prediction techniques with applications to channel equalization, over-the-horizon (OTH) radar, and array processing of high-frequency signals. His past research has involved a number of mathematical techniques and a variety of applications in signal processing and linear systems, including: lossless cascade models for multiple-input/multiple-output systems; extension of maximum-entropy techniques to multidimensional signal processing; modeling, analysis, and design of parallel algorithms and architectures; and characterization of structured matrices.

Dr. Lev-Ari served as an Associate Editor of *Circuits, Systems, and Signal Processing*, and of *Integration, The VLSI Journal*, and is a member of SIAM.

Thomas Kailath (S'57–M'62–F'70) was educated in Poona, India and received the S.M. and Sc.D. degrees from the Massachusetts Institute of Technology, Cambridge, in 1959 and 1961, respectively.

He was with the Jet Propulsion Laboratories, Pasadena, CA, from October 1961 to December 1962 and also taught part-time at the California Institute of Technology, Pasadena. He served as Director of the Information Systems Laboratory, Stanford, CA, from 1971 to 1980, as Associate Department Chairman from 1981 to 1987, and currently holds the Hitachi America Professorship in Engineering. He has held short-term appointments at several institutions around the world. He has worked in a number of areas including information theory, communications, computation, control, signal processing, VLSI design, statistics, linear algebra, and operator theory. His recent interests include applications of signal processing, computation and control to problems in semiconductor manufacturing, and wireless communications. He is the author of *Linear Systems* (Prentice-Hall, 1980) and *Lectures on Wiener and Kalman Filtering* (Springer-Verlag, 1981).

Dr. Kailath has held Guggenheim, Churchill, and Royal Society followships, among others, and has received awards from the IEEE Information Theory Society and the American Control Council, in addition to the Technical Achievement and Society Awards of the IEEE Signal Processing Society. He served as President of the IEEE Information Theory Society in 1975 and has been awarded honorary doctorates by Linköping University, Sweden, and Strathclyde University, Scotland. He is a Fellow of the Institute of Mathematical Statistics and a member of the National Academy of Engineering.