THE BELL SYSTEM TECHNICAL JOURNAL
Vol. 62, No. 10, December 1983
Printed in U.S.A.

Recursive Fixed-Order Covariance Least-Squares Algorithms

By M. L. HONIG*

(Manuscript received May 6, 1983)

This paper derives fixed-order recursive Least-Squares (LS) algorithms that can be used in system identification and adaptive filtering applications such as spectral estimation, and speech analysis and synthesis. These algorithms solve the sliding-window and growing-memory covariance LS estimation problems, and require less computation than both unnormalized and normalized versions of the computationally efficient order-recursive (lattice) covariance algorithms previously presented. The geometric or Hilbert space approach, originally introduced by Lee and Morf to solve the prewindowed LS problem, is used to systematically generate least-squares recursions. We show that combining subsets of these recursions results in prewindowed LS lattice and fixed-order (transversal) algorithms, and in sliding-window and growing-memory covariance lattice and transversal algorithms. The paper discusses both least-squares prediction and joint-process estimation.

I. INTRODUCTION

Computationally efficient recursive Least-Squares (LS) algorithms have recently attracted attention in applications such as adaptive equalization,[1–4] echo cancellation,[5] and speech analysis and synthesis[6,7] because of their fast convergence properties when compared to older least-mean-square or gradient adaptation techniques.[8–10] Since the work on computationally efficient LS algorithms by Morf and others first appeared in Refs. 11 and 12, numerous papers have followed that produce computationally efficient algorithms that solve different types

* AT&T Information Systems.

of autoregressive LS estimation problems.[13–17] In general, these algorithms fall into four categories: (1) prewindowed recursive LS, (2) sliding-window recursive LS, (3) growing-memory covariance recursive LS, and (4) nonrecursive LS algorithms. Each of the first three categories has two subcategories: fixed-order, or transversal, algorithms; and order-recursive, or lattice, algorithms.

References 1 and 12 present a prewindowed LS algorithm that satisfies a transversal filter structure (fast Kalman algorithm). Subsequent Refs. 7, 18, and 19 describe prewindowed and growing-memory covariance LS algorithms that satisfy a lattice structure. Normalized prewindowed LS lattice algorithms that involve fewer recursions than the original unnormalized versions, and which have the important advantage that all internal variables are less than or equal to unity in magnitude are presented in the more recent Ref. 13. Reference 14 extends the normalized lattice algorithms to solve the sliding-window and growing-memory covariance LS problems. The recursive algorithms mentioned so far require order N arithmetic operations per iteration to update the filter parameters, where N is the order of the filter. A computationally efficient order-recursive algorithm that solves the set of linear equations for the covariance LS prediction problem has been presented in Ref. 11, and extended to the joint-process-estimation case in Ref. 17. These algorithms require order N^2 operations to compute the LS prediction coefficients and are nonrecursive in the sense that the solution generated at time interval i is not used to generate the solution at time interval $i + 1$.

This paper attempts to unify and extend the previous work by (1) systematically generating all of the recursions needed to derive all of the previously mentioned algorithms, and (2) using these recursions to derive new recursive fixed-order sliding-window and growing-memory covariance LS algorithms. These new algorithms solve directly for the prediction- or autogressive-model coefficients, and involve significantly less computation than both the unnormalized and normalized versions of the order-recursive or covariance lattice algorithms presented in Ref. 14. In addition, in some applications it may be advantageous to work directly with the prediction- or autogressive-model coefficients, rather than the set of reflection coefficients produced by lattice algorithms. The algorithms mentioned in the previous paragraph, along with the new ones derived here, are obtained by appropriately arranging subsets of least-squares recursions. The geometric or Hilbert space approach originally used by Lee and Morf[20] to derive the prewindowed LS lattice algorithm is used to derive all of the basic least-squares recursions in a cohesive manner. In this paper, however, only scalar-valued data are considered.

The next section defines the sliding-window and growing-memory covariance LS problems to be solved. Then Section III reviews the geometric approach to LS estimation. Fundamental order and time updates for the least-squares projection operator are given in Section IV, with derivations in Appendix A. In Section V these projection updates systematically derive least-squares recursions. Section VI gives fixed-order covariance algorithms and Section VII extends the preceding discussion to the joint-process-estimation case. Appendix B lists subsets of recursions in Sections V and VII that constitute other LS algorithms.

II. PROBLEM STATEMENT

We start by defining a sequence of data values $y_0, y_1, \cdots, y_i$, where i is the current time index. A linear least-squares *forward predictor* of order n chooses the coefficients $f_{j|n}$ to minimize

$$\epsilon_f(i|n) = \sum_{m=i'}^{i} \left(y_m - \sum_{j=1}^{n} f_{j|n} y_{m-j} \right)^2, \tag{1}$$

where i' to i is the time window of interest. The coefficients $f_{j|n}$, $1 \le j \le n$, are called the forward-prediction coefficients. A linear least-squares *backward predictor* of order n chooses the backward-prediction coefficients $b_{j|n}$, $1 \le j \le n$, to minimize

$$\epsilon_b(i|n) = \sum_{m=i'}^{i} \left(y_{m-n} - \sum_{j=1}^{n} b_{j|n} y_{m-j+1} \right)^2. \tag{2}$$

Minimization of (1) rather than (2) is generally desired for a given application. The forward and backward prediction problems stated above are closely related, however, and the LS algorithms to be presented use the backward prediction coefficients to solve for the forward prediction coefficients in a computationally efficient manner.

If, instead of estimating future values of the *same* process, we wish to estimate another related process $\{x_i\}$, the least-squares cost function becomes

$$\epsilon_x(i|n) = \sum_{m=i'}^{i} \left(x_m - \sum_{j=1}^{n} c_{j|n} y_{m-j+1} \right)^2, \tag{3}$$

where tap coefficients $c_{j|n}$ replace the prediction coefficients $f_{j|n}$ and $b_{j|n}$. The cost function (3) is relevant to joint-process-estimation problems such as channel equalization and echo cancellation. In the case of channel equalization, y_i is the ith sample of the channel output, and x_i is the ith channel symbol.

Setting the derivatives of the cost functions (1), (2), and (3) with respect to the prediction (tap) coefficients equal to zero results in the following linear equations:

$$\Phi_{i'-1,i-1|n}\mathbf{f}(i|n) = \sum_{j=i'}^{i} y_j\mathbf{y}_{j-1|n}, \tag{4a}$$

$$\Phi_{i',i|n}\mathbf{b}(i|n) = \sum_{j=i'}^{i} y_{j-n}\mathbf{y}_{j|n}, \tag{4b}$$

and

$$\Phi_{i',i|n}\mathbf{c}(i|n) = \sum_{j=i'}^{i} x_j\mathbf{y}_{j|n}, \tag{4c}$$

where

$$\mathbf{f}^T(i|n) \equiv [f_{1|n}f_{2|n} \cdots f_{n|n}], \tag{5a}$$

$$\mathbf{b}^T(i|n) \equiv [b_{1|n}b_{2|n} \cdots b_{n|n}], \tag{5b}$$

$$\mathbf{c}^T(i|n) \equiv [c_{1|n}c_{2|n} \cdots c_{n|n}], \tag{5c}$$

$$\mathbf{y}^T_{j|n} \equiv [y_jy_{j-1} \cdots y_{j-n+1}], \tag{6}$$

and the covariance matrix

$$\Phi_{i',i|n} \equiv \sum_{j=i'}^{i} \mathbf{y}_{j|n}\mathbf{y}^T_{j|n}. \tag{7}$$

Suppose now that $i' = 0$, and that y_0 is the first available data sample. The least-squares solutions for **f**, **b**, and **c**, obtained by solving (4), are undefined since they depend on the unspecified data values $y - 1$, $y - 2$, $\cdots$, y_{-n}. The simplest, and perhaps most popular, technique for circumventing this problem is to assume all data values y_j, $j < 0$, are zero. The least-squares solutions resulting from this so-called prewindowed estimation are then well defined as long as the covariance matrix is nonsingular. In applications such as speech modelling, however, where estimates of the prediction coefficient vector $\mathbf{f}(i|n)$ are desired given relatively few data samples, prewindowed estimation may result in undesirable edge effects from assuming data is zero outside a given window. For these types of applications, it is desirable to estimate the prediction coefficients without any assumptions concerning the data outside the time window of interest.

Covariance least-squares estimation replaces the lower time limit i' in (1), (2), and (3) by n, so that only known data values are used to compute the LS prediction (tap) coefficients. The improved estimates so obtained are not without cost, however. The resulting covariance LS algorithms derived in this paper and elsewhere[7] involve more computation than prewindowed LS algorithms. Notice that at each iteration i, the prediction coefficients are computed from $i + 1$ data values. Because the number of data samples entering the least-squares computation grows with time, this type of estimation has been called growing-memory covariance estimation.[16]

Finally, another windowing technique that has attracted attention recently is the sliding-window technique, in which the lower time limit i' in (1) and (2) is replaced by $i - M + n + 1$, and in (3) by $i - N + n$, where M is a predetermined constant. At each iteration the least-squares prediction coefficients are therefore computed from a fixed number (M) of data samples. Notice that data samples outside the time window $i - M + 1$ to i have no effect on the least-squares solution for **f**, **b**, and **c** at time i, i.e., they are totally forgotten. This is in contrast to more conventional exponential forgetting techniques that reduce the effects of past data samples in a more continuous fashion.[16] The sliding window is therefore useful in applications where the autoregressive model changes abruptly with time, or where undesirable transients periodically affect the data samples. In the former case, when the model parameters change, the sliding window eventually discards data values corresponding to previous model parameters. In the latter case, the sliding window eventually discards corrupted data values.

Computationally efficient recursive algorithms that solve the growing-memory covariance and sliding-window LS estimation problems will be derived in Sections V through VII. The next section develops the necessary mathematical background by reviewing the geometric interpretation of linear least-squares estimation.

III. MATHEMATICAL BACKGROUND

Given two vectors **X** and **Y** having the same dimension i, the inner product of **X** and **Y** is defined to be

$$\langle \mathbf{X}, \mathbf{Y} \rangle \equiv \mathbf{X}^T \mathbf{W} \mathbf{Y}, \tag{8}$$

where **W** is some prespecified $i \times i$ weighting matrix. As an example, a typical weighting matrix is the exponential weighting matrix

$$\mathbf{W}_i = [1 \; w \; w^2 \; \cdots \; w^{i-1}]\mathbf{I}, \tag{9}$$

where **I** is the $i \times i$ identity matrix. For convenience, we will assume that **W** is the identity matrix. Modification of the results in this paper to the case where **W** is arbitrary is straightforward. The distance between two vectors **X** and **Y** with the same dimension is therefore the regular Euclidean distance,

$$\mathrm{d}(\mathbf{X}, \mathbf{Y}) = \| \mathbf{Y} - \mathbf{X} \| \equiv \langle \mathbf{Y} - \mathbf{X}, \mathbf{Y} - \mathbf{X} \rangle^{1/2}. \tag{10}$$

The (nth order) projection of a vector **Y** onto a subspace (or manifold) M, which is spanned by the n vectors $\{\mathbf{X}_1, \mathbf{X}_2, \cdots, \mathbf{X}_n\}$, is denoted as $P_M\mathbf{Y}$. The orthogonal projection of $\mathbf{Y}$ onto M is defined as

$$P_M^\perp \mathbf{Y} \equiv \mathbf{Y} - P_M\mathbf{Y}, \tag{11}$$

and is orthogonal to the subspace M. This implies that

$$\langle \mathbf{X}_j, \mathbf{Y} - P_M\mathbf{Y} \rangle = 0, \qquad \text{for} \quad j = 1, \cdots, n. \tag{12}$$

Since $P_M\mathbf{Y}$ lies in M, there exist constants, or regression coefficients $f_1, f_2, \cdots, f_n$ such that

$$P_M\mathbf{Y} = \sum_{j=1}^{n} f_j \mathbf{X}_j = \mathbf{Sf}, \tag{13}$$

where $\mathbf{S} = [\mathbf{X}_1 \cdots \mathbf{X}_n]$ and $\mathbf{f}^T = [f_1 \cdots f_n]$. Using (12) and (13), it is easy to show that

$$\mathbf{f} = (\mathbf{S}^T\mathbf{S})^{-1}\mathbf{S}^T\mathbf{X} \tag{14}$$

and

$$P_M\mathbf{Y} = \mathbf{S}(\mathbf{S}^T\mathbf{S})^{-1}\mathbf{S}^T\mathbf{Y}, \tag{15}$$

assuming $\mathbf{S}^T\mathbf{S}$ is nonsingular.

The linear least-squares estimate of $\mathbf{Y}$, based upon the vectors $\mathbf{X}_1, \cdots, \mathbf{X}_n$, is formed by choosing $f_1, \cdots, f_n$ such that

$$\| \underline{\epsilon} \|^2 \equiv \| \mathbf{Y} - \sum_{j=1}^{n} f_j \mathbf{X}_j \|^2 \tag{16}$$

is minimized. Differentiating this quantity with respect to f_j and setting the result equal to zero gives

$$\hat{\mathbf{Y}} \equiv \sum_{j=1}^{n} f_j \mathbf{X}_j = P_M\mathbf{Y}, \tag{17}$$

and the vector of estimation errors,

$$\underline{\epsilon} = \mathbf{Y} - \sum_{j=1}^{n} f_j \mathbf{X}_j = P_M^\perp \mathbf{Y}. \tag{18}$$

We have identified the operator P as a least-squares projection.

IV. PROJECTION-OPERATOR UPDATE FORMULAS

In this section some fundamental relationships satisfied by the least-squares projection operator are presented. These projection updates fall into two main categories: order updates and time updates. Under time updates are two subcategories, forward and backward time updates. We point out in advance that a total of three projection-operator updates will be used throughout this paper: one order update, one forward time update, and one backward time update. In addition, one forward and one backward time update for inner products will be needed.

4.1 Order updates

Given two vectors, $\mathbf{Y}$ and $\mathbf{X}$, and a linear space M spanned by the vectors $\mathbf{X}_1, \mathbf{X}_2, \cdots, \mathbf{X}_n$, all in $\mathbf{R}^i$, suppose we wish to calculate the least-squares estimate of $\mathbf{Y}$ based upon the vectors $\mathbf{X}_1, \cdots, \mathbf{X}_n$ and $\mathbf{X}$. In particular, we wish to find coefficients a_j, $j = 1, \cdots, n$, and b such that $\| \mathbf{Y} - (\sum_{j=1}^n a_j\mathbf{X}_j + b\mathbf{X}) \|^2$ is minimized. From the discussion in the last section, we know that the least-squares estimate of $\mathbf{Y}$ is

$$\sum_{j=1}^{n} a_j\mathbf{X}_j + b\mathbf{X} = P_{\{M+\mathbf{X}\}}\mathbf{Y}, \tag{19}$$

where $\{M + \mathbf{X}\}$ denotes the space spanned by M and $\mathbf{X}$. We can write the following orthogonal decomposition of the space $\{M + \mathbf{X}\}$,[21]

$$\{M + \mathbf{X}\} = M \oplus \{P_M^{\perp}\mathbf{X}\}. \tag{20}$$

By the Hilbert space projection theorem,[21] we have that for any vector $\mathbf{Y} \in \mathbf{R}^i$,

$$P_{\{M+\mathbf{X}\}}\mathbf{Y} = P_M\mathbf{Y} + P_{\{P_M^{\perp}\mathbf{X}\}}\mathbf{Y}. \tag{21}$$

Figure 1 illustrates this equation for the special case $n = 1$. The projection of $\mathbf{Y}$ onto the space spanned by two vectors $\mathbf{X}_1$ and $\mathbf{X}_2$ is shown as the sum of the two projections $P_{\mathbf{X}_1}\mathbf{Y}$ and $P_{\{P_{\mathbf{X}_1}^{\perp}\mathbf{X}_2\}}\mathbf{Y}$.

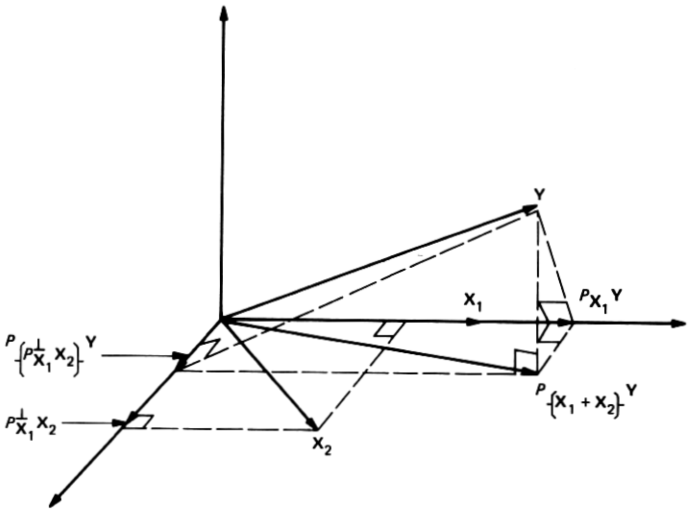

Fig. 1—Decomposition of $P_{\{\mathbf{X}_1+\mathbf{X}_2\}}\mathbf{Y}$.

Equation (21) constitutes a fundamental order update for the least-squares projection operator. The $(n + 1)$st-order projection $P_{\{M+\mathbf{x}\}}$ is expressed as the sum of the nth order projection P_M and the first order projection $P_{\{P_M^\perp \mathbf{x}\}}$. By subtracting both sides of (21) from $\mathbf{Y}$, we obtain the following order update for the *orthogonal* projection operator $P^\perp$,

$$P^\perp_{\{M+\mathbf{x}\}}\mathbf{Y} = P^\perp_M\mathbf{Y} - P_{\{P_M^\perp \mathbf{x}\}}\mathbf{Y}. \tag{22}$$

4.2 Forward time updates

The forward time updates derived in this section compute a least-squares projection at time i given the same least-squares projection at time $i - 1$. These recursions, when combined with the order recursions in the last subsection, can be used to derive prewindowed LS algorithms. We first consider the following vectors $\mathbf{X}_{i_0,i}$ and $\mathbf{Y}_{i_0,i}$, which are composed of data samples from time i_0 to i, i.e.,

$$\mathbf{X}^T_{i_0,i} = [x_i \; x_{i-1} \; \cdots \; x_{i_0}], \tag{23a}$$

and

$$\mathbf{Y}^T_{i_0,i} = [y_i \; y_{i-1} \; \cdots \; y_{i_0}]. \tag{23b}$$

For notational convenience, in this section only we will omit the lower time subscript on the data vectors and assume it to be i_0. Our objective is to compute the linear least-squares estimate of $\mathbf{Y}_i$, given $\mathbf{X}_i$ in terms of a least-squares estimate that does not use the most recent value y_i. With this in mind we define the unit vector

$$\mathbf{u}_i^T = [1 \;\; 0 \; \cdots \; 0 \;\; 0], \tag{24}$$

which has the same dimension as $\mathbf{Y}_i$, i.e., $\mathbf{u}_i \in \mathbf{R}^{i-i_0+1}$. Associated with $\mathbf{u}_i$ is the space spanned by $\mathbf{u}_i$, or the space of most recent data values, denoted as U_i. Note that $P_{U_t}\mathbf{Y}_i = y_i\mathbf{u}_i$. For notational convenience we define a tilde operator as follows,

$$\tilde{\mathbf{Y}}_i \equiv P^\perp_{U_i}\mathbf{Y}_i = [0 \;\; y_{i-1} \; y_{i-2} \; \cdots \; y_{i_0+1} \; y_{i_0}], \tag{25}$$

i.e., $\tilde{\mathbf{Y}}_i$ is the projection of $\mathbf{Y}_i$ onto the subspace of past data values.

The basic prediction problem is illustrated in Fig. 2, where $\mathbf{Y}_i$ is a vector having its endpoint in back of the plane of the paper and $\mathbf{X}_i$ has its endpoint in front of the plane of the paper. We are given the vector $\mathbf{X}_i$, from which the least-squares estimate of $\mathbf{Y}_i$, $P_{\mathbf{X}_i}\mathbf{Y}_i$, is to be recursively obtained. At time i we therefore assume a regression coefficient a computed at time $i - 1$ (i.e., $P_{\mathbf{X}_{i-1}}\mathbf{Y}_{i-1} = a\mathbf{X}_{i-1}$, or equivalently, $P_{\tilde{\mathbf{X}}_i}\tilde{\mathbf{Y}}_i = a\tilde{\mathbf{X}}_i$), which we wish to modify using the most recent data values y_i and x_i. Figure 2 therefore shows $P_{\mathbf{X}_i}\mathbf{Y}_i$ decomposed into the two vectors $a\mathbf{X}_i$ and $P_{\mathbf{X}_i}(\mathbf{Y}_i - a\mathbf{X}_i)$. Figure 3 illustrates the plane spanned by $\mathbf{X}_i$, $\tilde{\mathbf{X}}_i$, and U_i. Since ABC and ADE are similar

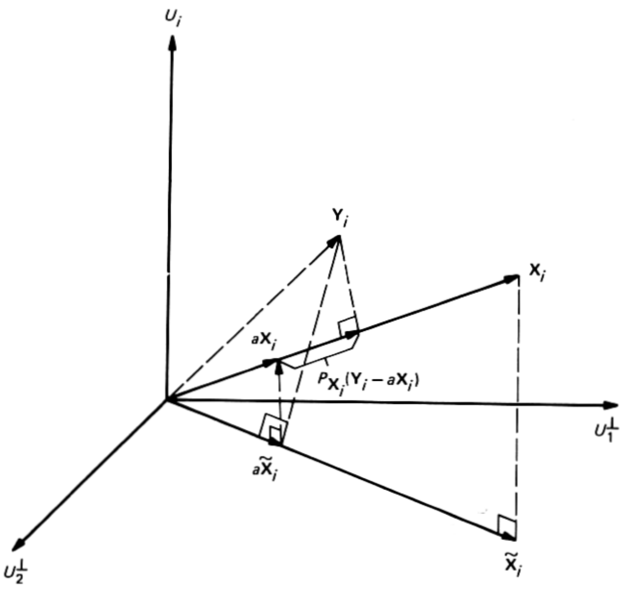

Fig. 2—Decomposition of $P_{\mathbf{X}_i}\mathbf{Y}_i$.

triangles,

$$\frac{\text{AB}}{\text{AD}} = \frac{\text{AC}}{\text{AE}} = a, \tag{26}$$

so that $\overline{\text{AC}} = a\mathbf{X}_i$. Figure 4 attempts to include vectors not shown in Fig. 2 and again illustrates the decomposition of $P_{\mathbf{X}_i}\mathbf{Y}_i$. (Only the endpoint of $\mathbf{Y}_i$ is in Fig. 4.)

Assume now that the vector $\mathbf{X}_i$ is replaced by the *subspace* M_i spanned by the vectors $\mathbf{X}_{1,i}, \mathbf{X}_{2,i}, \cdots, \mathbf{X}_{n,i}$. Let

$$\mathbf{S}_i = [\mathbf{X}_{1,i}\ \mathbf{X}_{2,i}\ \cdots\ \mathbf{X}_{n,i}] \tag{27}$$

and

$$\tilde{\mathbf{S}}_i = [\tilde{\mathbf{X}}_{1,i}\ \tilde{\mathbf{X}}_{2,i}\ \cdots\ \tilde{\mathbf{X}}_{n,i}]. \tag{28}$$

We define the projection

$$P_{M_{i|i-1}}\mathbf{Y}_i \equiv \mathbf{S}_i[\tilde{\mathbf{S}}_i^T\tilde{\mathbf{S}}_i]^{-1}\tilde{\mathbf{S}}_i^T\tilde{\mathbf{Y}}_i = \mathbf{S}_i\mathbf{f}, \tag{29}$$

i.e., $P_{M_{i|i-1}}\mathbf{Y}_i$ lies in M_i, but uses regression coefficients computed at time $i-1$. Referring to Fig. 3, $P_{\mathbf{X}_{i|i-1}}\mathbf{Y}_i = a\mathbf{X}_i$. Appendix A shows that

$$P_{M_i}\mathbf{Y}_i = P_{M_{i|i-1}}\mathbf{Y}_i + P_{M_i}\mathbf{u}_i\langle\mathbf{u}_i, P_{M_i}^{\perp}\mathbf{Y}_i\rangle\sec^2\theta_i, \tag{30}$$

where

$$\sin^2\theta_i = \langle\mathbf{u}_i, P_{M_i}\mathbf{u}_i\rangle = \|P_{M_i}\mathbf{u}_i\|^2$$

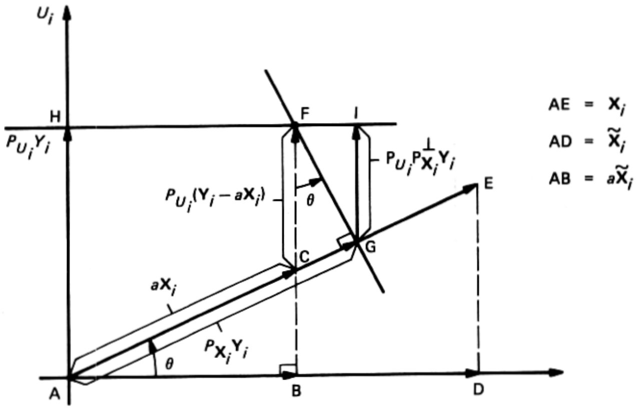

Fig. 3—Plane spanned by $\mathbf{X}_i$ and $\tilde{\mathbf{X}}_i$.

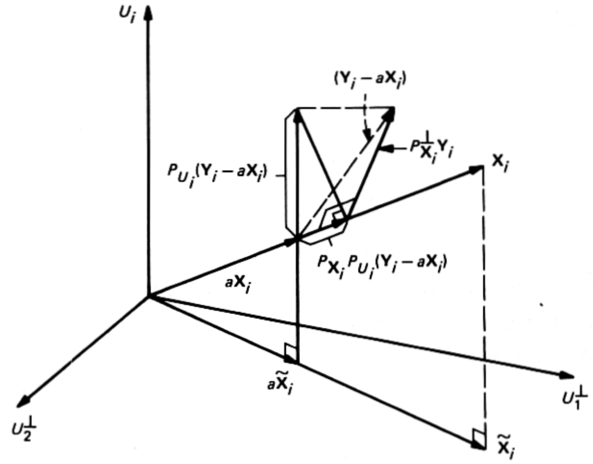

Fig. 4—Rotated view of Fig. 2.

$$= (\mathbf{u}_i^T\mathbf{S}_i)(\mathbf{S}_i^T\mathbf{S}_i)^{-1}(\mathbf{S}_i^T\mathbf{u}_i), \tag{31}$$

and

$$\sec^2\theta_i = \frac{1}{1 - \sin^2\theta_i}. \tag{32}$$

The variable θ_i can be interpreted as the angle between the spaces spanned by the matrices of basis vectors $\mathbf{S}_i$ and $\tilde{\mathbf{S}}_i$. Referring to Fig. 3, the angle θ is given by

$$\sin^2\theta = 1 - \frac{\|\tilde{\mathbf{X}}_i\|^2}{\|\mathbf{X}_i\|^2} = \frac{x_i^2}{\|\mathbf{X}_i\|^2}, \tag{33}$$

and measures the unexpectedness of the data received at time i. Notice that (33) can be rewritten as (31), where M_i and $\mathbf{S}_i$ are replaced by $\mathbf{X}_i$.

We obtain the following time update for the orthogonal projection operator by subtracting both sides of (30) from $\mathbf{Y}_i$,

$$P_{M_i}^{\perp}\mathbf{Y}_i = P_{M_{i|i-1}}^{\perp}\mathbf{Y}_i - P_{M_i}\mathbf{u}_i\langle \mathbf{u}_i, P_{M_i}^{\perp}\mathbf{Y}_i\rangle \sec^2\theta_i. \tag{34}$$

One more relation that will be useful in the following section is a recursive equation for the inner product $\langle \mathbf{v}_i, P_{M_i}^{\perp}\mathbf{Y}_i\rangle$, where $\mathbf{v}_i$ is an arbitrary vector in $\mathbf{R}^{i-i_0+1}$. This recursion, which is derived in Appendix A, is

$$\langle \mathbf{v}_i, P_{M_i}^{\perp}\mathbf{Y}_i\rangle = \langle \tilde{\mathbf{v}}_i, P_{\tilde{M}_i}^{\perp}\tilde{\mathbf{Y}}_i\rangle + \langle \mathbf{u}_i, P_{M_i}^{\perp}\mathbf{v}_i\rangle\langle \mathbf{u}_i, P_{M_i}^{\perp}\mathbf{Y}_i\rangle \sec^2\theta_i, \tag{35}$$

where $\tilde{M}_i$ is the space spanned by $\tilde{\mathbf{S}}_i$.

4.3 Backward time updates

Consider again the data vectors $\mathbf{X}_i$ and $\mathbf{Y}_i$ defined by (23). Suppose we wish to compute the linear least-squares estimate of $\mathbf{Y}_i$ given $\mathbf{X}_i$ in terms of a least-squares estimate that does not use the most *distant* or *past* values y_{i_0} and x_{i_0}. Clearly, this problem can be solved in exactly the same fashion as the time-update problem stated at the beginning of the last section. By turning the vectors $\mathbf{Y}_i$ and $\mathbf{X}_i$ upside down, and assuming that y_{i_0} and x_{i_0} are the most recent samples, one can solve this problem by using time updates already derived. The same argument holds when $\mathbf{X}_i$ is replaced by the subspace M_i spanned by vectors $\mathbf{X}_{1,i}$, $\mathbf{X}_{2,i}$, $\cdots$, $\mathbf{X}_{n,i}$. In this case we wish to calculate the projection $P_{M_i}\mathbf{Y}_i$ in terms of a projection onto the space spanned by the matrix of basis vectors $\mathbf{S}_i$ in which the *bottom row* has been replaced by zeros. This is in contrast to the previous time updates, which expressed $P_{M_i}\mathbf{Y}_i$ in terms of a projection onto the space spanned by $\mathbf{S}_i$ in which the *top* row has been replaced by zeroes (i.e., $\tilde{M}_i$).

In analogy with the notation defined in the last section, we define the unit vector

$$\mathbf{u}_{i_0}^T = [0 \ 0 \ \cdots \ 0 \ 1] \in \mathbf{R}^{i-i_0+1}, \tag{36}$$

and the space spanned by $\mathbf{u}_{i_0}$ as U_{i_0}. We also define the following asterisk operator in analogy with the previous tilde operator,

$$\mathbf{Y}_i^* = P_{U_{i_0}}^{\perp}\mathbf{Y}_i = [y_i \ y_{i-1} \ \cdots \ y_{i_0+1} \ 0]^T. \tag{37}$$

Similarly,

$$\mathbf{S}_i^* = [\mathbf{X}_{1,i}^* \ \mathbf{X}_{2,i}^* \ \cdots \ \mathbf{X}_{n,i}^*]. \tag{38}$$

The projection of $\mathbf{Y}_i$ onto M_i using regression coefficients computed from $\mathbf{S}_i^*$ is defined as

$$P_{M_{i_0|i_0+1}}\mathbf{Y}_i \equiv \mathbf{S}_i[\mathbf{S}_i^{*T}\mathbf{S}_i^*]^{-1}[\mathbf{S}_i^{*T}\mathbf{Y}_i]. \tag{39}$$

The regression coefficients that multiply the basis vectors of M_i are in this case elements of the vector $[\mathbf{S}_i^{*T}\mathbf{S}_i^*]^{-1}[\mathbf{S}_i^{*T}\mathbf{Y}_i]$.

The derivation of (30) can be repeated with $\mathbf{u}_i$ replaced by $\mathbf{u}_{i_0}$, tildes replaced by asterisks, and $P_{M_{i|i-1}}$ replaced by $P_{M_{i_0|i_0+1}}$ to give the following projection decomposition,

$$P_{M_i}\mathbf{Y}_i = P_{M_{i_0|i_0+1}}\mathbf{Y}_i + P_{M_i}\mathbf{u}_{i_0}\langle \mathbf{u}_{i_0}, P_{M_i}^{\perp}\mathbf{Y}_i\rangle \sec^2\theta_i^*, \tag{40}$$

where

$$\begin{aligned}\sin^2\theta_i^* &= \| P_{M_i}\mathbf{u}_{i_0}\|^2 \\ &= (\mathbf{u}_{i_0}^T\mathbf{S}_i)(\mathbf{S}_i^T\mathbf{S}_i)^{-1}(\mathbf{S}_i^T\mathbf{u}_{i_0}) \\ &= \langle \mathbf{u}_{i_0}, P_{M_i}\mathbf{u}_{i_0}\rangle,\end{aligned} \tag{41}$$

and

$$\sec^2\theta_i^* = \frac{1}{1-\sin^2\theta_i^*}. \tag{42}$$

Subtracting both sides of (40) from $\mathbf{Y}_i$ gives

$$P_{M_i}^{\perp}\mathbf{Y}_i = P_{M_{i_0|i_0+1}}^{\perp}\mathbf{Y}_i - P_{M_i}\mathbf{u}_{i_0}\langle \mathbf{u}_{i_0}, P_{M_i}^{\perp}\mathbf{Y}_i\rangle \sec^2\theta_i^*. \tag{43}$$

Finally, the following update for inner products is analogous to (35),

$$\langle \mathbf{v}_i, P_{M_i}^{\perp}\mathbf{Y}_i\rangle = \langle \mathbf{v}_i^*, P_{M_i^*}^{\perp}\mathbf{Y}_i^*\rangle + \langle \mathbf{u}_{i_0}, P_{M_i}^{\perp}\mathbf{v}_i\rangle\langle \mathbf{u}_{i_0}, P_{M_i}^{\perp}\mathbf{Y}_i\rangle \sec^2\theta_i^*. \tag{44}$$

This completes the presentation of projection-operator recursions needed to derive the least-squares recursions in Sections V and VII. All order updates for variables entering the least-squares algorithms to be presented can be derived from (22). Similarly, all forward and backward time updates for vectors entering these algorithms can be derived from (34) and (43), respectively, and all forward and backward time updates for inner products can be derived from (35) and (44), respectively.

V. LEAST-SQUARES RECURSIONS

5.1 Notation

Referring to the definition (23), a shift operator z^{-j} is defined by

$$z^{-j}\mathbf{Y}_{i_0,i}^T = [\, y_{i-j}\; y_{i-j-1}\; \cdots\; y_{i_0-j}]. \tag{45}$$

Equations (1) and (2) can now be rewritten as

$$\epsilon_f(i\,|\,n) = \| \mathbf{Y}_{i_0+n,i} - \sum_{j=1}^{n} f_{j\,|\,n}(z^{-j}\mathbf{Y}_{i_0+n,i})\|^2 \tag{46a}$$

and

$$\epsilon_b(i\,|\,n) = \| z^{-n}\mathbf{Y}_{i_0+n,i} - \sum_{j=1}^{n} b_{j\,|\,n}(z^{-j+1}\mathbf{Y}_{i_0+n,i}) \|^2, \tag{46b}$$

where i' has been replaced by $i_0 + n$. A matrix of shifted data vectors is denoted as

$$\mathbf{S}_{i_0+n,i}(l, n) = [z^{-l}\mathbf{Y}_{i_0+n,i}\; z^{-l-1}\mathbf{Y}_{i_0+n,i}\; \cdots\; z^{-n}\mathbf{Y}_{i_0+n,i}], \tag{47}$$

where $l < n$. The space spanned by the columns of $\mathbf{S}_{i_0+n,i}(l, n)$, which is a subspace generated by past data values, is denoted as $M_{i_0+n,i}(l, n)$. For notational convenience we will omit the lower time index of $\mathbf{S}$ and M and assume that it is always $i_0 + n$. Notice that we can write the covariance matrix defined by (7) as

$$\Phi_{i_0+n,i\,|\,n} = \mathbf{S}_i^T(0, n-1)\mathbf{S}_i(0, n-1). \tag{48}$$

Two types of updates exist for least-squares parameters: order updates and time updates. The time updates in this section generally fall into two categories. Given some LS parameter ξ (i.e., the forward prediction vector $\mathbf{f}$ or the prediction residual), we wish to find (1) a recursion for ξ computed from the data samples $\{y_{i_0}, y_{i_0+1}, \cdots, y_i\}$ in terms of ξ computed from the data samples $\{y_{i_0}, y_{i_0+1}, \cdots, y_{i-1}\}$ (forward time update), and (2) a recursion for ξ computed from the data samples $\{y_{i_0}, y_{i_0+1}, \cdots, y_i\}$ in terms of ξ computed from the data samples $\{y_{i_0+1}, y_{i_0+2}, \cdots, y_i\}$ (backward time update). Associated with the variable ξ is therefore an order index n and the time indices of the data used in the least-squares computation. If the data values $\{y_{i_0}, y_{i_0+1}, \cdots, y_i\}$ are used to compute ξ, then the indices i_0 and i must be specified. This is in contrast to the prewindowed case where only i need be specified since i_0 is always zero.

Throughout the rest of this paper, the starting-time index of the generic parameter ξ will appear as a subscript, and the current-time index will appear as a function argument. As an example $\xi_{i_0}(i\,|\,n)$ implies that the data values $\{y_{i_0}, y_{i_0+1}, \cdots, y_i\}$ are used to compute the nth order variable ξ. The following variables are needed to derive the LS algorithms in the next section:

1. Forward and backward prediction vectors [from (4)],

$$\mathbf{f}_{i_0}(i\,|\,n) = \Phi^{-1}_{i_0+n-1,i-1\,|\,n}[\mathbf{S}_i^T(1, n)\mathbf{Y}_{i_0+n,i}] \tag{49a}$$

and

$$\mathbf{b}_{i_0}(i\,|\,n) = \Phi^{-1}_{i_0+n,i\,|\,n}[\mathbf{S}_i^T(0, n-1)(z^{-n}\mathbf{Y}_{i_0+n,i})]. \tag{49b}$$

2. Forward and backward prediction residual vectors,

$$\mathbf{E}_{f,i_0}(i\,|\,n) \equiv \mathbf{Y}_{i_0+n,i} - \mathbf{S}_i(1, n)\mathbf{f}_{i_0}(i\,|\,n) \tag{50a}$$

and

$$\mathbf{E}_{b,i_0}(i\,|\,n) \equiv z^{-n}\mathbf{Y}_{i_0+n,i} - \mathbf{S}_i(0,\,n-1)\mathbf{b}_{i_0}(i\,|\,n). \tag{50b}$$

3. Forward and backward prediction residuals (scalars),

$$e_{f,i_0}(i\,|\,n) \equiv \langle \mathbf{u}_i, \mathbf{E}_{f,i_0}(i\,|\,n)\rangle = y_i - \mathbf{f}_{i_0}^T(i\,|\,n)\mathbf{y}_{i-1|n} \tag{51a}$$

and

$$e_{b,i_0}(i\,|\,n) \equiv \langle \mathbf{u}_i,\, \mathbf{E}_{b,i_0}(i\,|\,n)\rangle = y_{i-n} - \mathbf{b}_{i_0}^T(i\,|\,n)\mathbf{y}_{i|n}. \tag{51b}$$

4. Forward and backward cost functions,

$$\epsilon_{f,i_0}(i\,|\,n) \equiv \|\mathbf{E}_{f,i_0}(i\,|\,n)\|^2, \qquad \epsilon_{b,i_0}(i\,|\,n) \equiv \|\mathbf{E}_{b,i_0}(i\,|\,n)\|^2. \tag{52}$$

5. PARtial CORrelation (PARCOR) coefficient,

$$k_{n,i_0}(i) \equiv \langle \mathbf{E}_{f,i_0+1}(i\,|\,n-1),\, \mathbf{E}_{b,i_0}(i-1\,|\,n-1)\rangle. \tag{53}$$

6. Auxiliary variables, or gains,

$$\mathbf{g}_{i_0+1}(i\,|\,n) \equiv \Phi_{i_0+n,i|n}^{-1}\mathbf{y}_{i|n}, \tag{54a}$$

$$\mathbf{h}_{i_0+1}(i\,|\,n) \equiv \Phi_{i_0+n,i|n}^{-1}\mathbf{y}_{i_0+n|n}, \tag{54b}$$

$$\gamma_{i_0+1}(i\,|\,n) \equiv \langle \mathbf{u}_i,\, P_{M_i(0,n-1)}\mathbf{u}_i\rangle = \mathbf{y}_{i|n}^T\Phi_{i_0+n,i|n}^{-1}\mathbf{y}_{i|n}, \tag{55a}$$

$$\gamma_{i_0+1}^*(i\,|\,n) \equiv \langle \mathbf{u}_{i_0},\, P_{M_i(0,n-1)}\mathbf{u}_{i_0}\rangle = \mathbf{y}_{i_0+n|n}^T\Phi_{i_0+n,i|n}^{-1}\mathbf{y}_{i_0+n|n}, \tag{55b}$$

and

$$\alpha_{i_0+1}(i\,|\,n) \equiv \langle \mathbf{u}_{i_0},\, P_{M_i(0,n-1)}\mathbf{u}_i\rangle = \mathbf{y}_{i|n}^T\Phi_{i_0+n,i|n}^{-1}\mathbf{y}_{i_0+n|n}. \tag{55c}$$

Notice that

$$\mathbf{S}_i(0,\,n-1)\mathbf{g}_{i_0+1}(i\,|\,n) = P_{M_i(0,n-1)}\mathbf{u}_i; \tag{56a}$$

and

$$\mathbf{S}_i(0,\,n-1)\mathbf{h}_{i_0+1}(i\,|\,n) = P_{M_i(0,n-1)}\mathbf{u}_{i_0}; \tag{56b}$$

and that

$$\gamma_{i_0+1}(i\,|\,n) = \mathbf{g}_{i_0+1}^T(i\,|\,n)\mathbf{y}_{i|n}, \tag{57a}$$

$$\gamma_{i_0+1}^*(i\,|\,n) = \mathbf{h}_{i_0+1}^T(i\,|\,n)\mathbf{y}_{i_0+n|n}, \tag{57b}$$

and

$$\alpha_{i_0+1}(i\,|\,n) = \mathbf{g}_{i_0+1}^T(i\,|\,n)\mathbf{y}_{i_0+n|n} = \mathbf{h}_{i_0+1}^T(i\,|\,n)\mathbf{y}_{i|n}. \tag{57c}$$

Using the notation in the last section, the gains γ and γ^* are, respectively, $\sin^2\theta_i$ and $\sin^2\theta_i^*$, where θ_i and θ_i^* are, respectively, the angles between $M_i(0,\,n-1)$ and $\tilde{M}_i(0,\,n-1)$, and between $M_i(0,\,n-1)$ and $M_i^*(0,\,n-1)$.

At each time instant our objective is to minimize the cost functions

$\epsilon_{f,i_0}(i \mid n)$ and $\epsilon_{b,i_0}(i \mid n)$. From the discussion in Section III it follows that

$$\mathbf{E}_{f,i_0}(i \mid n) = P^{\perp}_{M_i(1,n)} \mathbf{Y}_{i_0+n,i} \tag{58a}$$

and

$$\mathbf{E}_{b,i_0}(i \mid n) = P^{\perp}_{M_i(0,n-1)} (z^{-n} \mathbf{Y}_{i_0+n,i}). \tag{58b}$$

The following variables, which are closely related to the prediction residuals, are also needed:

$$\begin{aligned} \mathbf{E}'_{f,i_0}(i \mid n) &\equiv P^{\perp}_{M_{i|i-1}(1,n)} \mathbf{Y}_{i_0+n,i} \\ &= \mathbf{Y}_{i_0+n,i} - \mathbf{S}_i(1, n) \mathbf{f}_{i_0}(i-1 \mid n), \end{aligned} \tag{59a}$$

and

$$\begin{aligned} \mathbf{E}'_{b,i_0}(i \mid n) &\equiv P^{\perp}_{M_{i|i-1}(0,n-1)} (z^{-n} \mathbf{Y}_{i_0+n,i}) \\ &= z^{-n} \mathbf{Y}_{i_0+n,i} - \mathbf{S}_i(0, n-1) \mathbf{b}_{i_0}(i-1 \mid n), \end{aligned} \tag{59b}$$

i.e., $\mathbf{E}'_f$ and $\mathbf{E}'_b$ are the forward and backward residual vectors obtained by using prediction vectors computed at the *previous* time interval. The top components of $\mathbf{E}'_{f,i_0}(i \mid n)$ and $\mathbf{E}'_{b,i_0}(i \mid n)$ are, respectively,

$$\begin{aligned} e'_{f,i_0}(i \mid n) &\equiv \langle \mathbf{u}_i, \mathbf{E}'_{f,i_0}(i \mid n) \rangle \\ &= y_i - \mathbf{f}^T_{i_0}(i-1 \mid n) \mathbf{y}_{i-1|n} \end{aligned} \tag{60a}$$

and

$$\begin{aligned} e'_{b,i_0}(i \mid n) &\equiv \langle \mathbf{u}_i, \mathbf{E}'_{b,i_0}(i \mid n) \rangle \\ &= y_{i-n} - \mathbf{b}^T_{i_0}(i-1 \mid n) \mathbf{y}_{i|n}. \end{aligned} \tag{60b}$$

The nth order forward prediction residual computed at time $i_0 + n$ using the tap vector $\mathbf{f}_{i_0}(i \mid n)$ is

$$\begin{aligned} e^*_{f,i_0}(i \mid n) &\equiv \langle \mathbf{u}_{i_0}, \mathbf{E}_{f,i_0}(i \mid n) \rangle \\ &= y_{i_0+n} - \mathbf{f}^T_{i_0}(i \mid n) \mathbf{y}_{i_0+n-1|n}. \end{aligned} \tag{61}$$

The forward residual vector at time i using the forward prediction vector calculated from the data samples $\{y_{i_0+1}, \cdots, y_i\}$ is

$$\begin{aligned} \mathbf{E}_{f,i_0|i_0+1}(i \mid n) &\equiv P^{\perp}_{M_{i_0|i_0+1}(1,n)} \mathbf{Y}_{i_0+n,i} \\ &= \mathbf{Y}_{i_0+n,i} - \mathbf{S}_i(1, n) \mathbf{f}_{i_0+1}(i \mid n). \end{aligned} \tag{62}$$

The variables $e^*_{b,i_0}(i \mid n)$ and $\mathbf{E}_{b,i_0|i_0+1}(i \mid n)$ are similarly defined.

Notice that the time indices associated with a residual vector change in accordance with the projection space, i.e.,

$$P^{\perp}_{M_i(1,n-1)} \mathbf{Y}_{i_0+n,i} = \mathbf{E}_{f,i_0+1}(i \mid n-1), \tag{63a}$$

and

$$P^{\perp}_{M_i(1,n-1)}(z^{-n}\mathbf{Y}_{i_0+n,i}) = \mathbf{E}_{b,i_0}(i-1\,|\,n-1). \tag{63b}$$

The recursions needed to derive the algorithms in the next section are now generated systematically. By appropriately defining the vectors and subspaces entering the projection order update (22), order updates are derived for all of the basic variables defined by (49) through (55). We then use the forward and backward time updates (34) and (43) to obtain forward and backward time updates for the basic vectors defined by (49), (50), and (54). Finally, the forward and backward time updates for inner products (35) and (44) are applied to $k_{n,i_0}(i)$, $\epsilon_{f,i_0}(i\,|\,n)$, and $\epsilon_{b,i_0}(i\,|\,n)$. It would take up too much space to explicitly define the vectors and subspaces that must be substituted in the projection update used to derive each recursion. Consequently, only the results are stated, with a few representative examples worked out in more detail.

5.2 Order updates

The following order updates are obtained by using the projection order update (22) [or equivalently (21)]. The lth through the mth component of $\mathbf{f}_{i_0}(i\,|\,n)$ is denoted by $[\mathbf{f}_{i_0}(i\,|\,n)]_{l,m}$, and $[\mathbf{f}_{i_0}(i\,|\,n)]_j$ is the jth component of $\mathbf{f}_{i_0}(i\,|\,n)$. The same notation is used for the backward prediction vector $\mathbf{b}_{i_0}(i\,|\,n)$ and the gain vectors $\mathbf{g}_{i_0}(i\,|\,n)$ and $\mathbf{h}_{i_0}(i\,|\,n)$.

$$\mathbf{E}_{f,i_0}(i\,|\,n) = \mathbf{E}_{f,i_0+1}(i\,|\,n-1) - \frac{k_{n,i_0}(i)}{\epsilon_{b,i_0}(i-1\,|\,n-1)} \cdot \mathbf{E}_{b,i_0}(i-1\,|\,n-1), \tag{64a}$$

$$\mathbf{E}_{b,i_0}(i\,|\,n) = \mathbf{E}_{b,i_0}(i-1\,|\,n-1) - \frac{k_{n,i_0}(i)}{\epsilon_{f,i_0+1}(i\,|\,n-1)} \cdot \mathbf{E}_{f,i_0+1}(i\,|\,n-1), \tag{64b}$$

$$\epsilon_{f,i_0}(i\,|\,n) = \epsilon_{f,i_0+1}(i\,|\,n-1) - \frac{k^2_{n,i_0}(i)}{\epsilon_{b,i_0}(i-1\,|\,n-1)}, \tag{65a}$$

$$\epsilon_{b,i_0}(i\,|\,n) = \epsilon_{b,i_0}(i-1\,|\,n-1) - \frac{k^2_{n,i_0}(i)}{R_{f,i_0+1}(i\,|\,n-1)}, \tag{65b}$$

$$[\mathbf{f}_{i_0}(i\,|\,n)]_n = \frac{k_{n,i_0}(i)}{\epsilon_{b,i_0}(i-1\,|\,n-1)}, \tag{66a}$$

$$[\mathbf{f}_{i_0}(i\,|\,n)]_{1,n-1} = \mathbf{f}_{i_0+1}(i\,|\,n-1) - [\mathbf{f}_{i_0}(i\,|\,n)]_n\mathbf{b}_{i_0}(i-1\,|\,n-1), \tag{66b}$$

$$[\mathbf{b}_{i_0}(i\,|\,n)]_1 = \frac{k_{n,i_0}(i)}{\epsilon_{f,i_0+1}(i\,|\,n-1)}, \tag{67a}$$

$$[\mathbf{b}_{i_0}(i\,|\,n)]_{2,n} = \mathbf{b}_{i_0}(i-1\,|\,n-1) - [\mathbf{b}_{i_0}(i\,|\,n)]_1 \mathbf{f}_{i_0+1}(i\,|\,n-1), \tag{67b}$$

$$[\mathbf{g}_{i_0}(i\,|\,n+1)]_{n+1} = \frac{e_{b,i_0}(i\,|\,n)}{\epsilon_{b,i_0}(i\,|\,n)}, \tag{68a}$$

$$[\mathbf{g}_{i_0}(i\,|\,n+1)]_{1,n} = \mathbf{g}_{i_0+1}(i\,|\,n) - [\mathbf{g}_{i_0}(i\,|\,n+1)]_{n+1}\mathbf{b}_{i_0}(i\,|\,n), \tag{68b}$$

$$[\mathbf{g}_{i_0}(i\,|\,n+1)]_1 = \frac{e_{f,i_0}(i\,|\,n)}{\epsilon_{f,i_0}(i\,|\,n)}, \tag{69a}$$

$$[\mathbf{g}_{i_0}(i\,|\,n+1)]_{2,n+1} = \mathbf{g}_{i_0}(i-1\,|\,n) - [\mathbf{g}_{i_0}(i\,|\,n+1)]_1 \mathbf{f}_{i_0}(i\,|\,n), \tag{69b}$$

$$[\mathbf{h}_{i_0}(i\,|\,n+1)]_{n+1} = \frac{e^*_{b,i_0}(i\,|\,n)}{\epsilon_{b,i_0}(i\,|\,n)}, \tag{70a}$$

$$[\mathbf{h}_{i_0}(i\,|\,n+1)]_{1,n} = \mathbf{h}_{i_0+1}(i\,|\,n) - [\mathbf{h}_{i_0}(i\,|\,n+1)]_{n+1}\mathbf{b}_{i_0}(i\,|\,n), \tag{70b}$$

$$[\mathbf{h}_{i_0}(i\,|\,n+1)]_1 = \frac{e^*_{f,i_0}(i\,|\,n)}{\epsilon_{f,i_0}(i\,|\,n)}, \tag{71a}$$

$$[\mathbf{h}_{i_0}(i\,|\,n+1)]_{2,n+1} = \mathbf{h}_{i_0}(i-1\,|\,n) - [\mathbf{h}_{i_0}(i\,|\,n+1)]_1 \mathbf{f}_{i_0}(i\,|\,n), \tag{71b}$$

$$\gamma_{i_0}(i\,|\,n+1) = \gamma_{i_0+1}(i\,|\,n) + \frac{e^2_{b,i_0}(i\,|\,n)}{\epsilon_{b,i_0}(i\,|\,n)}, \tag{72a}$$

$$\gamma_{i_0}(i\,|\,n+1) = \gamma_{i_0}(i-1\,|\,n) + \frac{e^2_{f,i_0}(i\,|\,n)}{\epsilon_{f,i_0}(i\,|\,n)}, \tag{72b}$$

$$\gamma^*_{i_0}(i\,|\,n+1) = \gamma^*_{i_0+1}(i\,|\,n) + \frac{e^{*2}_{b,i_0}(i\,|\,n)}{\epsilon_{b,i_0}(i\,|\,n)}, \tag{73a}$$

$$\gamma^*_{i_0}(i\,|\,n+1) = \gamma^*_{i_0}(i-1\,|\,n) + \frac{e^{*2}_{f,i_0}(i\,|\,n)}{\epsilon_{f,i_0}(i\,|\,n)}, \tag{73b}$$

$$\alpha_{i_0}(i\,|\,n+1) = \alpha_{i_0+1}(i\,|\,n) + \frac{e_{b,i_0}(i\,|\,n)e^*_{b,i_0}(i\,|\,n)}{\epsilon_{b,i_0}(i\,|\,n)}, \tag{74a}$$

and

$$\alpha_{i_0}(i\,|\,n+1) = \alpha_{i_0}(i-1\,|\,n) + \frac{e_{f,i_0}(i\,|\,n)e^*_{f,i_0}(i\,|\,n)}{\epsilon_{f,i_0}(i\,|\,n)}. \tag{74b}$$

As an example, (64a) is derived from (22), where M is replaced by $M_i(1, n-1)$, $\mathbf{X}$ is replaced by $z^{-n}\mathbf{Y}_{i_0+n,i}$, and $\mathbf{Y}$ is replaced by $\mathbf{Y}_{i_0+n,i}$.

By observing that $\mathbf{E}_{b,i_0}(i-1\,|\,n-1)$ is orthogonal to $M_i(1, n-1)$, it is clear that

$$\begin{aligned}\langle \mathbf{Y}_{i_0+n,i}, \mathbf{E}_{b,i_0}(i-1\,|\,n-1)\rangle &= \langle \mathbf{E}_{f,i_0+1}(i\,|\,n-1), \\ &\qquad \cdot \mathbf{E}_{b,i_0}(i-1\,|\,n-1)\rangle \\ &= k_{n,i_0}(i). \end{aligned} \tag{75}$$

Recursions (65) are obtained by taking norms of (64) respectively. The recursions (68) through (71) are obtained from (21), where $\mathbf{Y}$ is replaced by $\mathbf{u}_i$ and $\mathbf{u}_{i_0}$, respectively. Making the same substitutions in (21) and then taking inner products with $\mathbf{u}_i$ or $\mathbf{u}_{i_0}$ gives recursions (72) through (74).

5.3 Forward time updates

The following forward time updates are obtained from the (orthogonal) projection operator forward time update (34):

$$\mathbf{E}_{f,i_0}(i\,|\,n) = \mathbf{E}'_{f,i_0}(i\,|\,n) - [P_{M_i(1,n)}\mathbf{u}_i]\,\frac{e_{f,i_0}(i\,|\,n)}{1-\gamma_{i_0}(i-1\,|\,n)}, \tag{76a}$$

$$\mathbf{E}_{b,i_0}(i\,|\,n) = \mathbf{E}'_{b,i_0}(i\,|\,n) - [P_{M_i(0,n-1)}\mathbf{u}_i]\,\frac{e_b(i\,|\,n)}{1-\gamma_{i_0+1}(i\,|\,n)}, \tag{76b}$$

$$\mathbf{f}_{i_0}(i\,|\,n) = \mathbf{f}_{i_0}(i-1\,|\,n) + \mathbf{g}_{i_0}(i-1\,|\,n)\,\frac{e_{f,i_0}(i\,|\,n)}{1-\gamma_{i_0}(i-1\,|\,n)}, \tag{77a}$$

$$\mathbf{b}_{i_0}(i\,|\,n) = \mathbf{b}_{i_0}(i-1\,|\,n) + \mathbf{g}_{i_0+1}(i\,|\,n)\,\frac{e_{b,i_0}(i\,|\,n)}{1-\gamma_{i_0+1}(i\,|\,n)}, \tag{77b}$$

$$\mathbf{h}_{i_0}(i\,|\,n) = \mathbf{h}_{i_0}(i-1\,|\,n) - \mathbf{g}_{i_0}(i\,|\,n)\,\frac{\alpha_{i_0}(i\,|\,n)}{1-\gamma_{i_0}(i\,|\,n)}, \tag{78}$$

$$\gamma^*_{i_0}(i\,|\,n) = \gamma^*_{i_0}(i-1\,|\,n) - \frac{\alpha^2_{i_0}(i\,|\,n)}{1-\gamma_{i_0}(i\,|\,n)}, \tag{79}$$

and

$$\alpha_{i_0}(i\,|\,n) = \mathbf{y}^T_{i|n}\mathbf{h}_{i_0}(i-1\,|\,n)[1-\gamma_{i_0}(i\,|\,n)]. \tag{80}$$

Equation (78) is obtained from (34), where M_i is replaced $M_i(0, n)$ and $\mathbf{Y}_i$ is replaced by $\mathbf{u}_{i_0}$. Equations (79) and (80) are obtained by making the same substitutions in (34) and then taking inner products with $\mathbf{u}_{i_0}$ and $\mathbf{u}_i$, or by premultiplying (78) by $\mathbf{y}^T_{i_0+n-1|n}$ and $\mathbf{y}^T_{i|n}$, respectively. Taking the inner product of (76) with $\mathbf{u}_i$ and $\mathbf{u}_{i_0}$, respectively gives the following recursions:

$$e'_{f,i_0}(i\,|\,n) = \frac{e_{f,i_0}(i\,|\,n)}{1 - \gamma_{i_0}(i-1\,|\,n)}, \tag{81a}$$

$$e'_{b,i_0}(i\,|\,n) = \frac{e_{b,i_0}(i\,|\,n)}{1 - \gamma_{i_0+1}(i\,|\,n)}, \tag{81b}$$

$$e^*_{f,i_0}(i\,|\,n) = e^*_{f,i_0}(i-1\,|\,n) - e_{f,i_0}(i\,|\,n)\,\frac{\alpha_{i_0}(i-1\,|\,n)}{1 - \gamma_{i_0}(i-1\,|\,n)}, \tag{82a}$$

and

$$e^*_{b,i_0}(i\,|\,n) = e^*_{b,i_0}(i-1\,|\,n) - e_{b,i_0}(i\,|\,n)\,\frac{\alpha_{i_0+1}(i\,|\,n)}{1 - \gamma_{i_0+}(i\,|\,n)}. \tag{82b}$$

5.4 *Backward time updates*

The following backward time updates are obtained from the projection operator backward time update (43):

$$\mathbf{E}_{f,i_0}(i\,|\,n) = \mathbf{E}_{f,i_0\,|\,i_0+1}(i\,|\,n) - [P_{M_i(1,n)}\mathbf{u}_{i_0}]\,\frac{e^*_{f,i_0}(i\,|\,n)}{1 - \gamma^*_{i_0}(i-1\,|\,n)}, \tag{83a}$$

$$\mathbf{E}_{b,i_0}(i\,|\,n) = \mathbf{E}_{b,i_0\,|\,i_0+1}(i\,|\,n) - [P_{M_i(0,n-1)}\mathbf{u}_{i_0}]\,\frac{e^*_{b,i_0}(i\,|\,n)}{1 - \gamma^*_{i_0+1}(i\,|\,n)}, \tag{83b}$$

$$\mathbf{f}_{i_0+1}(i\,|\,n) = \mathbf{f}_{i_0}(i\,|\,n) - \mathbf{h}_{i_0}(i-1\,|\,n)\,\frac{e^*_{f,i_0}(i\,|\,n)}{1 - \gamma^*_{i_0}(i-1\,|\,n)}, \tag{84a}$$

$$\mathbf{b}_{i_0+1}(i\,|\,n) = \mathbf{b}_{i_0}(i\,|\,n) - \mathbf{h}_{i_0+1}(i\,|\,n)\,\frac{e^*_{b,i_0}(i\,|\,n)}{1 - \gamma^*_{i_0+1}(i\,|\,n)}, \tag{84b}$$

$$\mathbf{g}_{i_0}(i\,|\,n) = \mathbf{g}_{i_0+1}(i\,|\,n) - \mathbf{h}_{i_0}(i\,|\,n)\,\frac{\alpha_{i_0}(i\,|\,n)}{1 - \gamma^*_{i_0}(i\,|\,n)}, \tag{85}$$

$$\gamma_{i_0}(i\,|\,n) = \gamma_{i_0+1}(i\,|\,n) - \frac{\alpha^2_{i_0}(i\,|\,n)}{1 - \gamma^*_{i_0}(i\,|\,n)}, \tag{86}$$

and

$$\alpha_{i_0}(i\,|\,n) = \mathbf{y}^T_{i_0+n-1\,|\,n}\mathbf{g}_{i_0+1}(i\,|\,n)[1 - \gamma^*_{i_0}(i\,|\,n)]. \tag{87}$$

Equation (85) is obtained by replacing $\mathbf{Y}_i$ by $\mathbf{u}_i$ in (43). Equations (86) and (87) are obtained by premultiplying (85) by $\mathbf{y}^T_{i\,|\,n}$ and $\mathbf{y}^T_{i_0+n-1\,|\,n}$, respectively. The following recursions are obtained by taking the inner product of (83) with $\mathbf{u}_i$, respectively:

$$e_{f,i_0+1}(i\,|\,n) = e_{f,i_0}(i\,|\,n) + e^*_{f,i_0}(i\,|\,n)\,\frac{\alpha_{i_0}(i-1\,|\,n)}{1 - \gamma^*_{i_0}(i-1\,|\,n)}, \tag{88a}$$

and

$$e_{b,i_0+1}(i\,|\,n) = e_{b,i_0}(i\,|\,n) + e^*_{b,i_0}(i\,|\,n)\,\frac{\alpha_{i_0+1}(i\,|\,n)}{1-\gamma^*_{i_0+1}(i\,|\,n)}. \tag{88b}$$

The recursions that result from taking inner products with $\mathbf{u}_{i_0}$ will not be used and are therefore omitted.

5.5 Inner product updates

The following recursions are obtained from the forward time update for inner products (35):

$$k_{n,i_0}(i) = k_{n,i_0}(i-1) + e_{f,i_0+1}(i\,|\,n-1)e_{b,i_0}(i-1\,|\,n-1) \cdot \frac{1}{1-\gamma_{i_0+1}(i-1\,|\,n-1)}, \tag{89}$$

$$\epsilon_{f,i_0}(i\,|\,n) = \epsilon_{f,i_0}(i-1\,|\,n) + e^2_{f,i_0}(i\,|\,n)\,\frac{1}{1-\gamma_{i_0}(i-1\,|\,n)}, \tag{90a}$$

and

$$\epsilon_{b,i_0}(i\,|\,n) = \epsilon_{b,i_0}(i-1\,|\,n) + e^2_{b,i_0}(i\,|\,n)\,\frac{1}{1-\gamma_{i_0+1}(i\,|\,n)}. \tag{90b}$$

The following recursions are obtained from the backward time update for inner products (44):

$$k_{n,i_0}(i) = k_{n,i_0+1}(i) + e^*_{f,i_0+1}(i\,|\,n-1)e^*_{b,i_0}(i-1\,|\,n-1) \cdot \frac{1}{1-\gamma^*_{i_0+1}(i-1\,|\,n-1)}, \tag{91}$$

$$\epsilon_{f,i_0}(i\,|\,n) = \epsilon_{f,i_0+1}(i\,|\,n) + e^{*2}_{f,i_0}(i\,|\,n)\,\frac{1}{1-\gamma^*_{i_0}(i-1\,|\,n)}, \tag{92a}$$

and

$$\epsilon_{b,i_0}(i\,|\,n) = \epsilon_{b,i_0+1}(i\,|\,n) + e^{*2}_{b,i_0}(i\,|\,n)\,\frac{1}{1-\gamma^*_{i_0+1}(i\,|\,n)}. \tag{92b}$$

Equations (89) and (91) are obtained by using (35) and (44), where $\mathbf{v}_i$ is replaced by $\mathbf{Y}_{i_0+n,i}$, $\mathbf{Y}_i$ is replaced by $z^{-n}\mathbf{Y}_{i_0+n,i}$, and M_i is replaced by $M_i(1, n-1)$, respectively. The previous set of recursions (64) through (92) are complete in the sense that any existing least-squares alogrithm can be derived by manipulating suitable subsets of these recursions.

VI. RECURSIVE FIXED-ORDER COVARIANCE ALGORITHMS

6.1 Sliding-window algorithm

Combining (60), (61), (68) through (73), (77), (81a), (84), (90a), and (92a), gives the following sliding-window LS algorithm for the prediction coefficients. Where unspecified, the order of the variable is assumed to be N, the order of the least-squares filter. Also, the starting time index is denoted as i_0. If the sliding window contains M data values, then $i_0 = i - M + 1$,

$$e'_{f,i_0}(i) = y_i - \mathbf{f}^T_{i_0}(i-1)\mathbf{y}_{i-1}, \tag{93a}$$

$$\mathbf{f}_{i_0}(i) = \mathbf{f}_{i_0}(i-1) + \mathbf{g}_{i_0}(i-1)e'_{f,i_0}(i), \tag{93b}$$

$$e_{f,i_0}(i) = e'_{f,i_0}(i)[1 - \gamma_{i_0}(i-1)], \tag{93c}$$

$$e^*_{f,i_0}(i) = y_{i_0+N} - \mathbf{f}^T_{i_0}(i)\mathbf{y}_{i_0+N-1}, \tag{93d}$$

$$\epsilon_{f,i_0}(i) = \epsilon_{f,i_0}(i-1) + e'_{f,i_0}(i)e_{f,i_0}(i), \tag{93e}$$

$$\mathbf{g}_{i_0}(i\,|\,N+1)]_1 = \frac{e_{f,i_0}(i)}{\epsilon_{f,i_0}(i)}, \tag{93f}$$

$$[\mathbf{g}_{i_0}(i\,|\,N-1)]_{2,N+1} = \mathbf{g}_{i_0}(i-1) - [\mathbf{g}_{i_0}(i\,|\,N+1)]_1\mathbf{f}_{i_0}(i), \tag{93g}$$

$$[\mathbf{h}_{i_0}(i\,|\,N+1)]_1 = \frac{e^*_{f,i_0}(i)}{\epsilon_{f,i_0}(i)}, \tag{93h}$$

$$[\mathbf{h}_{i_0}(i\,|\,N+1)]_{2,N+1} = \mathbf{h}_{i_0}(i-1) - [\mathbf{h}_{i_0}(i\,|\,N+1)]_1\mathbf{f}_{i_0}(i), \tag{93i}$$

$$\mathbf{f}_{i_0+1}(i) = \mathbf{f}_{i_0}(i) - \mathbf{h}_{i_0}(i-1)\,\frac{e^*_{f,i_0}(i)}{1 - \gamma^*_{i_0}(i-1)}, \tag{93j}$$

$$\epsilon_{f,i_0+1}(i) = \epsilon_{f,i_0}(i) - \frac{e^{*2}_{f,i_0}(i)}{1 - \gamma^*_{i_0}(i-1)}, \tag{93k}$$

$$e'_{b,i_0}(i) = y_{i-N} - \mathbf{b}^T_{i_0}(i-1)\mathbf{y}_i, \tag{93l}$$

$$\mathbf{b}_{i_0}(i) = \frac{\mathbf{b}_{i_0}(i-1) + e'_{b,i_0}(i)[\mathbf{g}_{i_0}(i\,|\,N+1)]_{1,N}}{1 - e'_{b,i_0}(i)[\mathbf{g}_{i_0}(i\,|\,N+1)]_{N+1}}, \tag{93m}$$

$$e^*_{b,i_0}(i) = y_{i_0} - \mathbf{b}^T_{i_0}(i)\mathbf{y}_{i_0+N}, \tag{93n}$$

$$\mathbf{g}_{i_0+1}(i) = [\mathbf{g}_{i_0}(i\,|\,N+1)]_{1,N} + [\mathbf{g}_{i_0}(i\,|\,N+1)]_{N+1}\mathbf{b}_{i_0}(i), \tag{93o}$$

$$\mathbf{h}_{i_0+1}(i) = [\mathbf{h}_{i_0}(i\,|\,N+1)]_{1,N} + [\mathbf{b}_{i_0}(i\,|\,N+1)_{N+1}\mathbf{b}_{i_0}(i), \tag{93p}$$

$$\gamma_{i_0+1}(i) = \frac{\gamma_{i_0}(i-1) + e_{f,i_0}(i)[\mathbf{g}_{i_0}(i\,|\,N+1)]_1 - e'_{b,i_0}(i)[\mathbf{g}_{i_0}(i\,|\,N+1)]_{N+1}}{1 - e'_{b,i_0}(i)[\mathbf{g}_{i_0}(i\,|\,N+1)]_{N+1}}, \tag{93q}$$

$$\gamma^*_{i_0+1}(i) = \gamma^*_{i_0}(i-1) + e^*_{f,i_0}(i)[\mathbf{h}_{i_0}(i\,|\,N+1)]_1 - e^*_{b,i_0}[\mathbf{h}_{i_0}(i\,|\,N+1)]_N, \tag{93r}$$

and

$$\mathbf{b}_{i_0+1}(i) = \mathbf{b}_{i_0}(i) - \mathbf{h}_{i_0+1}(i)\,\frac{e^*_{b,i_0}(i)}{1-\gamma^*_{i_0+1}(i)}. \tag{93s}$$

The recursions (93m) and (93q) were not listed in the previous section, but are easily obtained by solving (77b) and (68b) simultaneously for $\mathbf{b}_{i_0}(i)$, and by substituting (68a), (69a), and (81b) into (72), and solving for $\gamma_{i_0+1}(i\,|\,n)$. Notice that all data samples in the sliding window $(y_{i-M+1}, \ldots, y_i)$ must be stored. This is also true of the order-recursive sliding-window algorithm presented in Ref. 14. If division is counted as multiplication, then the algorithm (93) requires $12N + 16$ multiplies and $12N + 12$ additions at each iteration. In contrast, the unnormalized sliding-window lattice predictor (see Appendix B) requires $16N$ multiplies and $10N$ additions per iteration, and the normalized lattice predictor[16] requires $30N$ multiplies, $18N$ additions, and $6N$ square roots per iteration.

Because sliding-window algorithms have finite memory, initialization for these algorithms is basically the same as for the prewindowed case, i.e., the data y_i can be assumed to be zero for $i < 0$. After M iterations, where M is the window length, these data points are discarded. The algorithm (93) is therefore initialized by setting the gains γ and γ^*, and the elements of the vectors **f**, **b**, **g**, and **h** equal to zero, and letting

$$\epsilon_{f,i_0}(0) = \delta, \tag{94}$$

where δ is chosen to ensure that the algorithm remains stable. It is easily verified that for time $i < M - N - 1$, where M is the length of the sliding window, the algorithm (93) becomes a modified version of the prewindowed LS transversal (fast Kalman) algorithm.[1,22]

6.2 Growing-memory covariance algorithm

The following fixed-order growing-memory covariance algorithm is obtained by combining (60), (68b), (69), (77), (78), (80), (85), (87), and (90a). The lower index of the window i_0 is assumed to be zero. For notational convenience we define the following variables,

$$\beta_{i_0}(i\,|\,n) \equiv \frac{\alpha_{i_0}(i\,|\,n)}{1-\gamma_{i_0}(i\,|\,n)} \tag{95a}$$

and

$$\beta^*_{i_0}(i\,|\,n) \equiv \frac{\alpha_{i_0}(i\,|\,n)}{1 - \gamma^*_{i_0}(i\,|\,n)}. \tag{95b}$$

Where unspecified, the lower time index and the order of the variables are equal to zero and N, respectively,*

$$e_f'(i) = y_i = \mathbf{f}^T(i-1)\mathbf{y}_{i-1}, \tag{96a}$$

$$\mathbf{f}(i) = \mathbf{f}(i-1) + \mathbf{g}(i-1)e_f'(i), \tag{96b}$$

$$e_f(i) = y_i - \mathbf{f}^T(i)\mathbf{y}_{i-1}, \tag{96c}$$

$$\epsilon_f(i) = \epsilon_f(i-1) + e_f(i)e_f'(i), \tag{96d}$$

$$[\mathbf{g}(i\,|\,N+1)]_1 = \frac{e_f(i)}{\epsilon_f(i)}, \tag{96e}$$

$$[\mathbf{g}(i\,|\,N+1)]_{2,N+1} = \mathbf{g}(i-1) - [\mathbf{g}(i\,|\,N+1)]_1\mathbf{f}(i), \tag{96f}$$

$$e_b'(i) = y_{i-N} - \mathbf{b}^T(i-1)\mathbf{y}_i, \tag{96g}$$

$$\mathbf{b}(i) = \frac{\mathbf{b}(i-1) + e_b'(i)[\mathbf{g}(i\,|\,N+1)]_{1,N}}{1 - e_b'(i)[\mathbf{g}(i\,|\,N+1)]_{N+1}}, \tag{96h}$$

$$\mathbf{g}_1(i) = [\mathbf{g}(i\,|\,N+1)]_{1,N} + [\mathbf{g}(i\,|\,N+1)]_{N+1}\mathbf{b}(i), \tag{96i}$$

$$\beta(i) = \mathbf{y}_i^T\mathbf{h}(i-1), \tag{96j}$$

$$\beta^*(i) = \mathbf{y}_{N-1}^T\mathbf{g}_1(i), \tag{96k}$$

$$\mathbf{g}(i) = \frac{\mathbf{g}_1(i) - \beta^*(i)\mathbf{h}(i-1)}{1 - \beta(i)\beta^*(i)}, \tag{96l}$$

and

$$\mathbf{h}(i) = \mathbf{h}(i-1) - \beta(i)\mathbf{g}(i). \tag{96m}$$

Notice that this algorithm can be applied only if $i > N$. Otherwise, the least-squares variables of order N are undefined and cannot be used to compute the same least-squares variables at the successive time interval. Initialization of this algorithm can be performed, however, by using an order-recursive algorithm for $i < N$ to increase the order of the filter by one at each successive time iteration. An order-recursive algorithm for the prediction coefficients is obtained by combining (89), (66a), (67a), top components of (64a) and (64b), (66b), (67b), (65a) and (65b), (82a), (88a), (84a), (92a), (71), (73b), (72), and (74b). This algorithm is basically the same as the covariance lattice

* The author recently discovered that this algorithm has been independently derived in Ref. 23 using an algebraic approach.

algorithm presented in Refs. 7 and 19, except that additional order recursions for the prediction vectors **f** and **b** have been added. It is not explicitly stated in an effort to conserve space. Order-recursive computation of **f** and **b** requires order N^2 arithmetic operations per iteration, rather than order N operations per iteration, as required by the fixed-order algorithm. Not all N components of the vectors **f** and **b** need to be updated at each iteration for $i < N$, however. If data is first received at time $i = 0$, the recursions listed above can be used for $n = 0$ up to $n = i$. At time $i = N$ all of the variables that enter the fixed-order algorithm (96) have been computed by the order-recursive algorithm except for $\mathbf{g}(i)$, $\beta(i)$ and $\beta^*(i)$. The gain $\mathbf{g}(i)$ is the only variable needed at the next iteration of the fixed-order algorithm and can be computed by first using (96j) to calculate $\beta(i)$ and then using (96m) to solve for $\mathbf{g}(i)$.

Derivation of initial conditions for the order-recursive initialization routine is significantly more complicated than for the sliding-window algorithm. This is because for $i = n$, the matrix $\Phi_{n,i|n}$ is guaranteed to be singular, and hence all variables are technically undefined. Reference 14 gives a convenient solution to this startup problem. By using a generalized inverse of a singular or nonsingular matrix, the least-squares projection operator P, given by (15), can be defined even when the matrix $\mathbf{S}^T\mathbf{S}$ is singular. If this generalized inverse is defined appropriately, it can be shown that the projection updates in Section IV hold even when the covariance matrix is singular. This implies that all of the recursions listed in the last paragraph that constitute the order-recursive initialization routine can be used starting from $i = 0$ with the following initial conditions:

$$\mathbf{f}(0\,|\,0) = \mathbf{b}(0\,|\,0) = \mathbf{f}_1(-1\,|\,0) = \mathbf{h}(-1\,|\,0) = \mathbf{0}, \tag{97a}$$

$$k_n(-1) = 0, \qquad 1 \leqslant n \leqslant N, \tag{97b}$$

$$\gamma(-1\,|\,0) = \gamma^*(-1\,|\,0) = \alpha(-1\,|\,0) = 0, \tag{97c}$$

and

$$e_f^*(0\,|\,n) = \begin{cases} y_0 & \text{for} \quad n = 0 \\ 0 & \text{for} \quad n > 0. \end{cases} \tag{97d}$$

At each iteration $i < N$,

$$e_f(i\,|\,0) = e_b(i\,|\,0) = y_i \tag{97e}$$

and

$$\epsilon_f(i\,|\,0) = \epsilon_b(i\,|\,0) = \epsilon_f(i-1\,|\,0) + y_i^2. \tag{97f}$$

Counting division as multiplication, the algorithm (96) requires

$11N + 7$ multiplies and $11N + 1$ additions per iteration. To compare, the unnormalized growing-memory lattice predictor (see Appendix B) requires $22N$ multiplies and $12N$ additions per iteration. The normalized lattice algorithm requires $30N$ multiplies, $18N$ additions, and $6N$ square roots per iteration. We point out that the fixed-order covariance algorithm specified by (96) is not unique. In particular, equation (96c) can be replaced by (81a). The extra recursion (93q) must be added to compute γ, however. This type of modification has been applied to the fast Kalman algorithm, and has resulted in improved numerical properties.[22]

VII. EXTENSIONS TO JOINT-PROCESS ESTIMATION

The algorithms presented so far solve the LS prediction problem wherein the sums (1) and (2) are minimized. In applications such as channel equalization, echo and noise cancellation, and adaptive line enhancement, two processes, $\{x_j\}$ and $\{y_j\}$, are given, and our objective is to estimate the $\{x_j\}$ process in terms of the $\{y_j\}$ process. The vector of estimation errors is denoted as

$$\begin{aligned}\mathbf{E}_{x,i_0+1}(i\,|\,n) &\equiv \mathbf{X}_{i_0+n,i} - \sum_{j=0}^{n-1} c_{j+1|n}(z^{-j}\mathbf{Y}_{i_0+n,i})\\ &= \mathbf{X}_{i_0+n,i} - \mathbf{S}_i(0,\, n-1)\mathbf{c}_{i_0+1}(i\,|\,n),\end{aligned} \tag{98}$$

where $\mathbf{X}_{i_0,i}$ is defined by (23a), $\mathbf{c}_{i_0+1}(i\,|\,n)$ is the n-dimensional vector of regression coefficients at time i used to estimate $\mathbf{X}_{i_0+n,i}$ [given by (4c)] where $i' = i_0 + n$), and the lower time subscript of $\mathbf{E}_x$ and $\mathbf{c}$ denotes the time index of the starting value from the y sequence (i.e., y_{i_0+1}), which is used in the least-squares computation. Our objective is to choose $\mathbf{c}_{i_0+1}(i\,|\,n)$ such that

$$\epsilon_{x,i_0+1}(i\,|\,n) \equiv \|\,\mathbf{E}_{x,i_0+1}(i\,|\,n)\,\|^2 \tag{99}$$

is minimized. The discussion in Section III implies that

$$\mathbf{E}_{x,i_0+1}(i\,|\,n) = P^{\perp}_{M_i(0,n-1)}\mathbf{X}_{i_0+n,i}. \tag{100}$$

We now use the projection recursions in Section IV to derive order and time updates for $\mathbf{E}_{x,i_0}(i\,|\,n)$ and $\mathbf{c}_{i_0}(i\,|\,n)$. Details are again omitted since they are basically the same as before. Combining recursions in this section with the prediction algorithms of the last section results in recursive algorithms that solve the LS joint-process-estimation problem.

The following notation, which is analogous to the notation in Section 5.1, is first defined:

1. Cross-correlation coefficient,

$$k^{(x)}_{n+1,i_0}(i) \equiv \langle \mathbf{X}_{i_0+n,i},\ \mathbf{E}_{b,i_0}(i\,|\,n)\rangle$$

$$= \langle \mathbf{E}_{x,i_0+1}(i\,|\,n),\ \mathbf{E}_{b,i_0}(i\,|\,n)\rangle. \qquad (101)$$

2. Current residual (scalar),

$$e_{x,i_0}(i\,|\,n) = \langle \mathbf{u}_i,\ \mathbf{E}_{x,i_0}(i\,|\,n)\rangle$$

$$= x_i - \mathbf{c}^T_{i_0}(i\,|\,n)\mathbf{y}_{i|n}. \qquad (102)$$

3. Past residual (scalar),

$$e^*_{x,i_0+1}(i\,|\,n) = \langle \mathbf{u}_{i_0},\ \mathbf{E}_{x,i_0+1}(i\,|\,n)\rangle$$

$$= x_{i_0+n} - \mathbf{c}^T_{i_0+1}(i\,|\,n)\mathbf{y}_{i_0+n|n}. \qquad (103)$$

4. Oblique residual

$$e'_{x,i_0}(i\,|\,n) \equiv x_i - \mathbf{c}^T_{i_0}(i-1\,|\,n)\mathbf{y}_{i|n}. \qquad (104)$$

The following order recursions are obtained from (22):

$$\mathbf{E}_{x,i_0}(i\,|\,n+1) = \mathbf{E}_{x,i_0+1}(i\,|\,n) - \frac{k^{(x)}_{n+1,i_0}(i)}{\epsilon_{b,i_0}(i\,|\,n)}\,\mathbf{E}_{b,i_0}(i\,|\,n), \qquad (105)$$

$$\epsilon_{x,i_0}(i\,|\,n+1) = \epsilon_{x,i_0+1}(i\,|\,n) - \frac{k^{(x)2}_{n+1,i_0}(i)}{\epsilon_{b,i_0}(i\,|\,n)}, \qquad (106)$$

$$[\mathbf{c}_{i_0}(i\,|\,n+1)]_{n+1} = \frac{k^{(x)}_{n+1,i_0}(i)}{\epsilon_{b,i_0}(i\,|\,n)}, \qquad (107a)$$

and

$$[\mathbf{c}_{i_0}(i\,|\,n+1)]_{1,n} = \mathbf{c}_{i_0+1}(i\,|\,n) - [\mathbf{c}_{i_0}(i\,|\,n+1)]_{n+1}\mathbf{b}_{i_0}(i\,|\,n). \qquad (107b)$$

Derivation of the following forward time updates involves a straightforward application of (34) and (35), where $\mathbf{Y}_i$ is replaced by $\mathbf{X}_{i_0+n,i}$ and M_i is replaced by $M_i(0, n-1)$:

$$\mathbf{c}_{i_0}(i\,|\,n) = \mathbf{c}_{i_0}(i-1\,|\,n) + e'_{x,i_0}(i\,|\,n)\mathbf{g}_{i_0}(i\,|\,n), \qquad (108)$$

$$k^{(x)}_{n+1,i_0}(i) = k^{(x)}_{n+1,i_0}(i-1) + e_{x,i_0+1}(i\,|\,n)e_{b,i_0}(i\,|\,n)\,\frac{1}{1-\gamma_{i_0+1}(i\,|\,n)}, \qquad (109)$$

$$\epsilon_{x,i_0}(i\,|\,n) = \epsilon_{x,i_0}(i-1\,|\,n) + e^2_{x,i_0}(i\,|\,n)\,\frac{1}{1-\gamma_{i_0}(i\,|\,n)}, \qquad (110)$$

$$e^*_{x,i_0}(i\,|\,n) = e^*_{x,i_0}(i-1\,|\,n) - e_{x,i_0}(i\,|\,n)\,\frac{\alpha_{i_0}(i\,|\,n)}{1-\gamma_{i_0}(i\,|\,n)}, \qquad (111)$$

and

$$e'_{x,i_0}(i\,|\,n) = \frac{e_{x,i_0}(i\,|\,n)}{1 - \gamma_{i_0}(i\,|\,n)}. \tag{112}$$

Similarly, the following backward time updates are obtained from (43) and (44):

$$\mathbf{c}_{i_0}(i\,|\,n) = \mathbf{c}_{i_0+1}(i\,|\,n) + \mathbf{h}_{i_0}(i\,|\,n)\,\frac{e^*_{x,i_0}(i\,|\,n)}{1 - \gamma^*_{i_0}(i\,|\,n)}, \tag{113}$$

$$e_{x,i_0+1}(i\,|\,n) = e_{x,i_0}(i\,|\,n) + e^*_{x,i_0}(i\,|\,n)\,\frac{\alpha_{i_0}(i\,|\,n)}{1 - \gamma^*_{i_0}(i\,|\,n)}, \tag{114}$$

$$k^{(x)}_{n+1,i_0+1}(i) = k^{(x)}_{n+1,i_0}(i) - e^*_{x,i_0+1}(i\,|\,n)e^*_{b,i_0}(i\,|\,n)\,\frac{1}{1 - \gamma^*_{i_0+1}(i\,|\,n)}, \tag{115}$$

and

$$\epsilon_{x,i_0+1}(i\,|\,n) = \epsilon_{x,i_0}(i\,|\,n) - e^{*2}_{x,i_0}(i\,|\,n)\,\frac{1}{1 - \gamma^*_{i_0}(i\,|\,n)}. \tag{116}$$

Combining (104), (108), (103), and (113) (in that order) with the fixed-order sliding-window algorithm (93) gives the corresponding sliding-window joint-process-estimation algorithm. Adding these additional recursions results in a total computational complexity of $16N + 17$ multiplies and $16N + 13$ additions per iteration. This should be compared with $23N$ multiplies and $14N$ additions per iteration required by the unnormalized sliding-window lattice joint-process estimator. Initialization of these additional recursions is accomplished in a fashion analogous to the prediction recursions. In particular, the data y_i and x_i is assumed to be zero for $i < 0$, and $\mathbf{c}_{i_0}(-1\,|\,n) = \mathbf{0}$.

The fixed-order growing-memory algorithm (96) is extended to the joint-process-estimation case by adding the recursions (104) and (108). The order-recursive prediction algorithm listed in Section 6.2 is extended to the joint-process-estimation case by adding the recursions (105) (top component only), (109), (107), (111), and (113). In each case the variable $i_0 = 0$. Adding (104) and (108) to (96) results in a total computational complexity of $13N + 7$ multiplies and $13N + 1$ additions per iteration. This should be compared with $28N$ multiplies and $16N$ additions per iterations required by the growing-memory covariance lattice joint-process estimator. The following accomplishes the initialization of the additional recursions for the order-recursive algorithm:

$$k^{(x)}_n(-1) = 0, \qquad 1 \leqslant n \leqslant N, \tag{117a}$$

$$\mathbf{c}(-1\,|\,n) = \mathbf{0}, \qquad 0 \leqslant n \leqslant N, \tag{117b}$$

and

$$e_x^*(0 \mid n) = \begin{cases} x_0 & \text{for} \quad n = 0 \\ 0 & \text{for} \quad n > 0. \end{cases} \tag{117c}$$

The fixed-order algorithm is initialized by using the order-recursive algorithm for $i < N$.

VIII. CONCLUSIONS

We have presented new fixed-order algorithms that recursively solve the sliding-window and growing-memory covariance least-squares estimation problems. The fixed-order growing-memory algorithm requires approximately one half the number of multiplies and divides required by the analogous unnormalized order-recursive or lattice algorithm. The fixed-order sliding-window algorithm requires approximately 70 percent of the number of multiplies and divides required by the analogous lattice algorithm. These fixed-order algorithms also help complete the list of computationally efficient LS algorithms currently available. In particular, each type of windowing technique that has been proposed for the LS computation (i.e., prewindowed, growing-memory covariance, and the sliding window) has resulted in both computationally efficient fixed-order and order-recursive algorithms. The order-recursive algorithms offer the advantage of being able to dynamically choose the order of the autoregressive model, while the fixed-order algorithms require less computation.

Associated with the algorithms mentioned in this paper are performance issues such as the relative convergence speed of each algorithm given different types of stationary and nonstationary random inputs, and the evaluation of finite word-length effects. As an example, the relative performance improvement offered by LS covariance algorithms over LS prewindowed algorithms has yet to be ascertained in applications where the prediction coefficients must be estimated from relatively few data samples. These issues will play a crucial role in determining the practical value of the LS algorithms presented in this paper.

REFERENCES

1. D. D. Falconer and L. Ljung, "Application of Fast Kalman Estimation to Adaptive Equalization," IEEE Trans. Commun., *COM-26* (October 1978), pp. 1439–46.
2. E. Satorius and J. Pack, "Application of Least Squares Lattice Algorithms to Adaptive Equalization," IEEE Trans. Commun., *COM-29* (February 1981), pp. 136–42.
3. T. L. Lim and M. S. Mueller, "Rapid Equalizer Start-Up Using Least Squares Algorithms" 1980 Proc. IEEE ICC, Seattle, WA.
4. M. Mueller, "On the Rapid Initial Convergence of Least Squares Equalizer Adjustment Algorithms," B.S.T.J., *60*, No. 10 (December 1981), pp. 2345–58.

5. F. K. Soong and A. M. Pederson, "Fast Least-Squares (LS) in the Voice Echo Cancellation Application," Proc. 1982 IEEE ICASSP, Paris, France, May 1982.
6. L. R. Rabiner and R. W. Schafer, *Digital Processing of Speech Signals*, Englewood Cliffs, NJ: Prentice-Hall, 1978.
7. M. Morf and D. T. L. Lee, "Fast Algorithms for Speech Modeling," Technical Report No. M308-1, Information Systems Laboratory, Stanford University, Stanford, CA, December 1978.
8. B. Widrow, "Adaptive Filters," in *Aspects of Network and System Theory*, R. Kalman and N. De Claris, Eds. New York, NY: Holt, Rinehart, and Winston, 1971, pp. 563–87.
9. R. D. Gitlin, J. E. Mazo, and M. G. Taylor, "On the Design of Gradient Algorithms for Digitally Implemented Adjustment Filters," IEEE Trans. Circuit Theory, *CT-20* (March 1973), pp. 125–36.
10. L. J. Griffiths, "A Continuously-Adaptive Filter Implemented as a Lattice Structure," Proc. 1977 IEEE ICASSP, Hartford, CT (May 1977), pp. 683–6.
11. M. Morf, B. Dickinson, T. Kailath, and A. Vieira, "Efficient Solution of Covariance Equations for Linear Prediction," IEEE Trans. ASSP, *ASSP-25* (October 1977), pp. 429–33.
12. L. Ljung, M. Morf, and D. Falconer, "Fast Calculation of Gain Matrices for Recursive Estimation Schemes," Int. J. Contr. *27*, No. 1 (1978), pp. 1–19.
13. D. T. Lee, B. Friedlander, and M. Morf, "Recursive Least Squares Ladder Estimation Algorithms," IEEE Trans. ASSP, *ASSP-29* (June 1981), pp. 627–41.
14. B. Porat, B. Friedlander, and M. Morf, "Square-Root Covariance Ladder Algorithms," IEEE Trans. Aut. Control, *AC-27*, No. 4 (August 1982), pp. 813–29.
15. C. Samson, "A Unified Treatment of Fast Kalman Algorithms for Identification," Int. J. Control, *35*, No. 5 (May 1982), pp. 909–34.
16. B. Friedlander, "Lattice Filters for Adaptive Processing," Proc. IEEE, *70*, No. 8 (August 1982), pp. 829–67.
17. S. L. Marple, Jr., "Efficient Least Squares FIR System Identification," IEEE Trans. ASSP, *ASSP-29* (Feb. 1981), pp. 62–73.
18. M. Morf, D. T. Lee, J. R. Nickolls, and A. Vieira, "A Classification of Algorithms for ARMA Models and Ladder Realizations," Proc. 1977 IEEE Conf. ASSP, Hartford, CT (April 1977), pp. 13–9.
19. M. Morf, D. T. Lee, "Recursive Least Squares Ladder Forms for Fast Parameter Tracking," Proc. 1978 IEEE Conf. D&C, San Diego, CA (Jan. 12, 1979), pp. 1326–67.
20. D. T. L. Lee and M. Morf, "Recursive Square-Root Ladder Estimation Algorithms," Proc. 1980 IEEE ICASSP, Denver, CO, April 1980.
21. A. W. Naylor and G. R. Sell, *Linear Operator Theory in Engineering and Science*, New York, NY: Holt, Rinehart and Winston, Inc., 1971.
22. J. Cioffi and T. Kailath, "Fast, Fixed-Order, Least-Squares Algorithms for Adaptive Filtering," Proc. 1983 IEEE ICASSP, Boston MA, April 1983.
23. C. C. Halkias et al., "A New Generalized Recursion for the Fast Computation of the Kalman Gain to Solve the Covariance Equations," Proc. 1982 IEEE ICASSP, Paris, France (April 1982), pp. 1760–3.
24. M. L. Honig, "Performance of FIR Adaptive Filters Using Recursive Algorithms," Ph.D. Dissertation, Univ. of California, Berkeley, April 1981.

APPENDIX A

Derivation of (30)

We wish to prove (30). By definition,

$$P_{M_{i|i-1}}\mathbf{Y}_i = P_{\tilde{M}_i}\mathbf{Y}_i + P_{U_i}P_{M_{i|i-1}}\mathbf{Y}_i, \tag{118}$$

where $\tilde{M}$ is the subspace spanned by the column vectors of $\tilde{\mathbf{S}}_i$. Projecting both sides of (118) onto M_i gives

$$P_{M_i}P_{M_{i|i-1}}\mathbf{Y}_i = P_{M_i}P_{\tilde{M}_i}\mathbf{Y}_i + P_{M_i}P_{U_i}P_{M_{i|i-1}}\mathbf{Y}_i. \tag{119}$$

Now $P_{M_{i|i-1}}\mathbf{Y}_i$ lies in M_i, and hence

$$P_{M_i}P_{M_{i|i-1}}\mathbf{Y}_i = P_{M_{i|i-1}}\mathbf{Y}_i. \tag{120}$$

Also,

$$P_{M_i}P_{\tilde{M}_i}\mathbf{Y}_i = \mathbf{S}_i(\mathbf{S}_i^T\mathbf{S}_i)^{-1}(\mathbf{S}_i^T\tilde{\mathbf{S}}_i)(\tilde{\mathbf{S}}_i^T\tilde{\mathbf{S}}_i)^{-1}\tilde{\mathbf{S}}_i^T\mathbf{Y}_i$$
$$= \mathbf{S}_i(\mathbf{S}_i^T\mathbf{S}_i)^{-1}\tilde{\mathbf{S}}_i^T\mathbf{Y}_i$$
$$= P_{M_i}(\mathbf{Y}_i - P_{U_i}\mathbf{Y}_i). \tag{121}$$

Combining (118) through (121) gives

$$P_{M_i}\mathbf{Y}_i = P_{M_{i|i-1}}\mathbf{Y}_i + P_{M_i}P_{U_i}P^{\perp}_{M_{i|i-1}}\mathbf{Y}_i$$
$$= P_{M_{i|i-1}}\mathbf{Y}_i + (P_{M_i}\mathbf{u}_i)\langle\mathbf{u}_i, P^{\perp}_{M_{i|i-1}}\mathbf{Y}_i\rangle. \tag{122}$$

Subtracting both sides of (122) from $\mathbf{Y}_i$, and then taking inner products of both sides with $\mathbf{u}_i$ gives

$$\langle\mathbf{u}_i, P^{\perp}_{M_i}\mathbf{Y}_i\rangle = \langle\mathbf{u}_i, P_{M_{i|i-1}}\mathbf{Y}_i\rangle[1 - \langle\mathbf{u}_i, P_{M_i}\mathbf{u}_i\rangle]. \tag{123}$$

Combining (122) and (123), and using the definition (31) gives (30). [Ref. 24 gives a purely geometric proof of (30) for the case where M_i is spanned by one vector (as illustrated in Fig. 2).]

To derive the inner product update (35), we first rewrite (34) as

$$P^{\perp}_{M_i}\mathbf{Y}_i = P^{\perp}_{U_i}P^{\perp}_{M_{i|i-1}}\mathbf{Y}_i + P_{U_i}P^{\perp}_{M_{i|i-1}}\mathbf{Y}_i - P_{M_i}\mathbf{u}_i\langle\mathbf{u}_i, P^{\perp}_{M_i}\mathbf{Y}_i\rangle\sec^2\theta_i$$
$$= P^{\perp}_{U_i}P^{\perp}_{\tilde{M}_i}\mathbf{Y}_i + \mathbf{u}_i\langle\mathbf{u}_i, P^{\perp}_{M_{i|i-1}}\mathbf{Y}_i\rangle - P_{M_i}\mathbf{u}_i\langle\mathbf{u}_i, P^{\perp}_{M_i}\mathbf{Y}_i\rangle\sec^2\theta_i$$
$$= P^{\perp}_{U_i}P^{\perp}_{\tilde{M}_i}\mathbf{Y}_i + P^{\perp}_{M_i}\mathbf{u}_i\langle\mathbf{u}_i, P^{\perp}_{M_i}\mathbf{Y}_i\rangle\sec^2\theta_i. \tag{124}$$

Taking the inner product of both sides with $\mathbf{v}_i$ and using the fact that

$$\langle\mathbf{v}_i, P^{\perp}_{M_i}\mathbf{u}_i\rangle = \langle\mathbf{u}_i, P^{\perp}_{M_i}\mathbf{v}_i\rangle \tag{125}$$

gives (35).

APPENDIX B

Other Recursive Least-Squares Algorithms

The recursions in Section V and VII are complete in the sense that any of the existing computationally efficient LS prediction or joint-process-estimation algorithms can be derived from suitable subsets of these recursions. The purpose of this appendix is to illustrate this point by listing the recursions that enter the prewindowed LS transversal (fast Kalman) and lattice algorithms, the unnormalized sliding-window and growing-memory covariance lattice algorithms,[14] and the nonrecursive LS algorithm presented in Refs. 11 and 17. The list of recursions presented below does not completely describe each algorithm. For example, initialization is not discussed. Consistent steady-state algorithms can be formulated, however, by choosing the time indices and order of the variables in each recursion appropriately. The following algorithms apply to the more general joint-process-estimation case (eliminating the recursions from Section VII gives the analogous prediction algorithm):

1. Prewindowed transversal (fast Kalman) algorithm:[1] (60a), (77a), (51a), (90a), (69), (60b), (93m), (68b), (104), and (108).

2. Prewindowed lattice algorithm* (See Refs. 18 and 16): (89), (64a)† and (64b), (65a) and (65b), (72a), (109) and (105).

3. Sliding-window lattice algorithm:[14,16] (89), (64a) and (64b), (65a) and (65b), (72a), (73a), (91), (109), (105), and (115).

4. Growing-memory covariance lattice algorithm:[14,16] (89), (64a) and (64b), (65a) and (65b), (82a), (88a), (92a), (72b) and (72a), (73b), (74b), (109), (105), (111), and (114).

5. Nonrecursive LS algorithm:[11,17]

The following set of recursions, which represents a modified version of the algorithm presented in Ref. 17, can be used to compute $\mathbf{f}(i\,|\,N)$, $\mathbf{b}(i\,|\,N)$, and $\mathbf{c}(i\,|\,N)$, given by (4), in an order-recursive fashion starting with first-order least squares variables at time i. Initialization consists of computing these first-order variables via the definitions given in Section V.

(78) (for computing $\mathbf{h}(i-1\,|\,n)$), (85) (for computing $\mathbf{g}_1(i\,|\,n)$), (79), (86), (84a), (77b), (92a), (90b), (53), (66), (67), (65a) and (65b), (51b), (61), (68), (71), (57c), (72a), (73b), (103), (113), (101), (107).

Assuming that the covariance matrix $\Phi_{N,i|N+1}$ has been computed, a more convenient form for (53) is

$$\begin{aligned} k_n(i) &= \langle \mathbf{Y}_{n,i}, \mathbf{E}_b(i-1\,|\,n-1)\rangle \\ &= \mathbf{Y}_{n,i}^T[z^{-n}\mathbf{Y}_{n,i} - \mathbf{S}_i(1, n-1)\mathbf{b}(i-1\,|\,n-1)] \\ &= R_n - \sum_{j=1}^{n-1} R_{n-j}[\mathbf{b}(i-1\,|\,n-1)]_j, \end{aligned} \tag{126}$$

where $R_j = \mathbf{Y}_{n,i}^T(z^{-j}\mathbf{Y}_{n,i})$, and is the $(1, j+1)$st element of $\Phi_{n,i|n+1}$. Equation (101) can be similarly modified.

AUTHOR

Michael L. Honig, B.S. (Electrical Engineering), 1977, Stanford University; M.S. and Ph.D. (Electrical Engineering), University of California, Berkeley, 1978 and 1981, respectively; AT&T Bell Laboratories, 1981–1982, AT&T

* The normalized lattice algorithms presented in Refs. 13 and 14 can be obtained by making substitutions for the variables entering the unnormalized recursions presented here.[16,20,24]

† In the prewindowed and growing-memory covariance cases, the inner products of (64) and (105) with $\mathbf{u}_i$ are required. In the sliding-window case, the inner products of (64) and (105) with both $\mathbf{u}_i$ and $\mathbf{u}_{i_0}$ are required.

Information Systems, 1983—. At AT&T Bell Laboratories and AT&T Information Systems Mr. Honig has worked on modulation, coding, and echo cancellation of voiceband data signals; and performance analysis of local area networks. He is currently working on office information networks. Member, IEEE, Tau Beta Pi, Phi Beta Kappa.