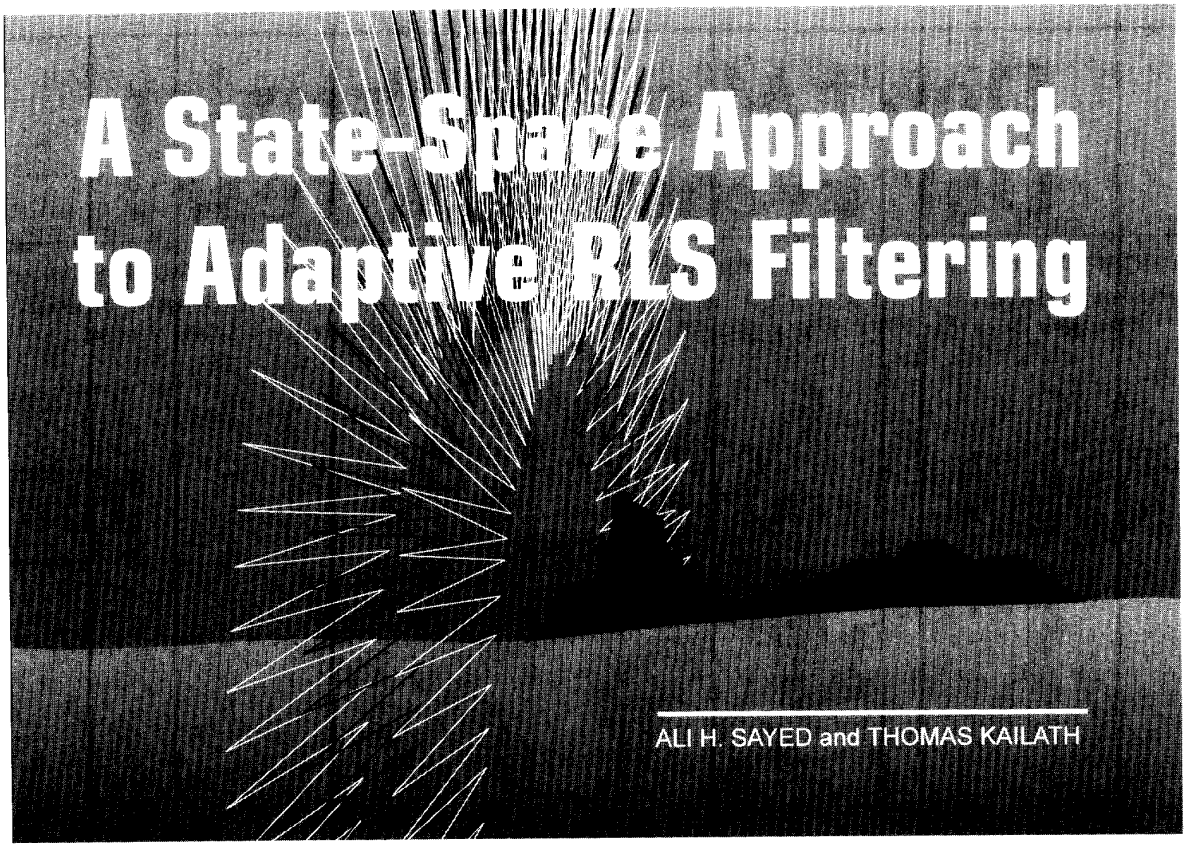

Adaptive filtering is gaining favor in numerous applications to help cope with time-variations of system parameters, and to compensate for the lack of *a priori* knowledge of the statistical properties of the input data. Over the last several years, a wide range of algorithms has been developed. These fall into four main groups: recursive least squares (RLS) algorithms and the corresponding fast versions; QR- and Inverse QR-least squares algorithms; least-squares lattice (LSL) and QR decomposition-based least squares lattice (QRD-LSL) algorithms; and gradient-based algorithms such as the least-mean square (LMS) algorithm.

It is practically impossible to list all the relevant references and all the major contributors to this field. However, the books [1]-[7], along with their extensive lists of references, should provide an excellent idea of the main results in this area. We shall, however, most often use the widely referenced textbook of Haykin [1] as a guide throughout our presentation.

The methods employed to analyze adaptive filter problems are quite varied, as becomes clear if one scans through the available literature [1-23]. Very often, different arguments and techniques are employed to derive different versions of the same algorithm. This has the obvious advantage of leading to interesting explorations of new approaches. But it has the disadvantage, in several instances, of leading to solutions that are sometimes complex to describe, and with seemingly little connection with already established results.

This makes software and hardware implementations rather complicated. More importantly, perhaps, is that it also has the disadvantage of obscuring potential connections that should obviously exist among variants of the same algorithm; only recently has there been some discussion of the close relations between some of the algorithms (e.g., [3] and [20]-[25]).

Our purpose in this article is to present yet another approach, not for the sake of adding to the already long list of available approaches, but for the sake of achieving two important goals. The first one is to show how several different variants of the recursive least-squares algorithm can be directly related to the widely studied Kalman filtering problem of estimation and control. Only very special instances of this relation have been considered thus far in the literature. Reference [26] was perhaps the first to rephrase the *growing memory* recursive least-squares problem in a stochastic state-space framework, with the unknown state corresponding to the unknown weight vector (see also [5], pp. 331-335). Various attempts to incorporate the case of exponentially decaying memory were also made, but in all of them some annoying discrepancies remained that were overcome essentially by fiat (e.g., [1], pp. 502-504). This lack of a direct correspondence appears to have inhibited application of the extensive body of Kalman filter results to the adaptive filtering problem.

However, using a simple device, we can obtain a perfectly

1053-5888/94/$4.00©1994IEEE

matched state-space model for the case of exponentially decaying memory, with a direct correspondence between the variables in the exponentially weighted RLS problem and the variables in the state-space estimation problem. The main benefit of this is that recursive state-space estimation problems have been extensively studied since the sixties, especially in the control engineering literature (e.g., [27-30]). Besides the celebrated Riccati-equation-based Kalman filtering algorithm, many algorithmic and implementational alternatives have been studied over the years. These include the so-called information filter forms and for certain kinds of time-variant state-space models (including those encountered in adaptive filtering), the Riccati recursions can be replaced by the order-of-magnitude faster Chandrasekhar recursions [31]; moreover, all these variants have certain computationally better square-root (or array) forms. The interesting fact is that when the exponentially-weighted RLS filtering problem is reformulated in state-space form, the now well-known alternative Kalman filtering solutions turn out to be equivalent to the various classes of adaptive filtering algorithms derived in the last decade (Tables 1 and 2). In fact, among others, all the algorithms in Haykin's book [1] can be obtained in this way.

Table 1: Most common adaptive schemes.

Adaptive Algorithm	Order-Recursive	Fixed-Order	Cost per iteration
RLS		x	$O(M^2)$
QR and Inverse QR		x	$O(M^2)$
FTF, FAEST		x	$O(M)$
Least-squares lattice	x		$O(M)$
QRD-based lattice	x		$O(M)$
LMS		x	$O(M)$

Table 2: State-space estimation algorithms vs. Adaptive algorithms.

State-Space Estimation Algorithm	Adaptive Algorithm(s)
Riccati-based Kalman filter	RLS
Chandrasekhar form	FTF and FAEST
Information form	QR
Square-root covariance form	Inverse QR
Recursive information form	QRD- and least-squares lattices
MinMax or H^∞ filter	gradient RLS/LMS

Our second important goal is to present all the different versions of the RLS algorithm in computationally convenient square-root forms: a prearray of numbers has to be triangularized by a rotation, or a sequence of elementary rotations, in order to yield a postarray of numbers. The quantities needed to form the next prearray can then be read off from the entries of the postarray, and the procedure can be repeated; the explicit forms of the rotation matrices are not needed in most cases. This is more truly an algorithm in the sense that it operates on a set of numbers and provides another set of numbers, with no explicit equations involved. The rotations themselves can be implemented in a variety of by now well-known ways: as a sequence of elementary circular or hyperbolic rotations, in square-root- and/or division-free forms, as Householder transformations, etc. These may differ in computational complexity, numerical behavior, and ease of hardware (VLSI) implementation. But, if preferred, explicit expressions for the rotation matrices can also be written down, thus leading to explicit sets of equations in contrast to the square-root form.

Notation

The following notational conventions will be useful to remember. We shall use small boldface letters to denote vectors and capital boldface letters to denote matrices. Also, given a positive definite matrix $\mathbf{A}$, $\mathbf{A} > 0$, a square-root factor will be defined as any matrix, say $\mathbf{A}^{1/2}$, such that $\mathbf{A} = (\mathbf{A}^{1/2})(\mathbf{A}^{1/2})^*$, where the * denotes Hermitian conjugation (complex conjugation for scalars). Such square-root factors are clearly not unique. They can be made unique, e.g., by insisting that the factors be Hermitian or that they be triangular (with positive diagonal elements). In most applications, the triangular form is preferred. For convenience, we shall also write:

$$(\mathbf{A}^{1/2})^* = \mathbf{A}^{*/2},\ (\mathbf{A}^{1/2})^{-1} = \mathbf{A}^{-1/2},\ (\mathbf{A}^{-1/2})^* = \mathbf{A}^{-*/2}$$

Thus, note the expressions $\mathbf{A} = \mathbf{A}^{1/2}\mathbf{A}^{*/2}$, $\mathbf{A}^{-1} = \mathbf{A}^{-*/2}\mathbf{A}^{-1/2}$. Also, we shall not confine ourselves to real data. Complex quantities will be allowed for the sake of generality. Finally, the symbol $\mathbf{I}_n$ will denote the identity matrix of size $n \times n$.

Square-Root or Array Algorithms

The square-root or array form is so important that it will be worthwhile to explain its generic form here, and give a couple of examples showing the compactness and simplicity it can bring.

An array algorithm is described via rotation operations on a prearray of numbers, chosen to obtain a certain zero pattern in a postarray where the desired quantities can be read out. Schematically, we have

$$\begin{bmatrix} x & x & x & x \\ x & x & x & x \\ x & x & x & x \\ x & x & x & x \end{bmatrix} \Theta = \begin{bmatrix} x & 0 & 0 & 0 \\ x & x & 0 & 0 \\ x & x & x & 0 \\ x & x & x & x \end{bmatrix}$$

where Θ is any rotation matrix that triangularizes the prearray. In general, Θ is required to be a **J**- unitary matrix, in the sense that $\Theta \mathbf{J} \Theta^* = \mathbf{J}$, where $\mathbf{J}$ is a signature matrix with ± 1's on the diagonal, and zeros elsewhere. The unitary case corresponds to $\mathbf{J} = \mathbf{I}$. A rotation Θ that transforms the prearray

to triangular form can be achieved in a variety of ways: by using a sequence of elementary Givens and hyperbolic rotations [33], Householder transformations [33]-[35], square-root-free versions of such rotations (e.g., [33] and [36-38]), etc.

Elementary Circular and Hyperbolic Rotations

An elementary 2×2 unitary rotation Θ (also known as Givens or circular rotation) takes a row vector $\mathbf{x} = [\, a \; b \,]$ and rotates it to lie along the basis vector $\mathbf{e}_0 = [\, 1 \; 0 \,]$. More precisely, it performs the transformation

$$[\, a \;\; b \,]\,\Theta = \left[\sqrt{|a|^2+|b|^2} \;\; 0\right] \tag{1a}$$

The quantity $\sqrt{|a|^2 + |b|^2}$ that appears on the right-hand side is consistent with the fact that the prearray and the postarray must have equal Euclidean norms. An expression for Θ is given by

$$\Theta = \frac{1}{\sqrt{1+|\rho|^2}} \begin{bmatrix} 1 & \rho \\ \rho^* & -1 \end{bmatrix} \qquad \text{where } \rho = \frac{b}{a},\ a \neq 0 \tag{1b}$$

In the trivial case, $a = 0$, we simply choose Θ as the permutation matrix,

$$\Theta = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$$

Also note that, in the special case of real data, a general unitary rotation as in Eq. 1b can be expressed in the alternative form:

$$\Theta = \begin{bmatrix} c & s \\ s & -c \end{bmatrix}$$

where the so-called cosine and sine parameters, c and s, respectively, are defined by

$$c = \frac{1}{\sqrt{1 + |\rho|^2}} \quad , \quad s = \frac{\rho}{\sqrt{1 + |\rho|^2}}$$

This justifies the name *circular rotation* for Θ, since the effect of Θ is to rotate the original vector $\mathbf{x}$ along the *circle* of equation $x^2 + y^2 = |a|^2 + |b|^2$, by an angle θ, determined by the inverse of the above cosine and/or sine parameters, $\theta = \tan^{-1}\rho$, in order to align it with the basis vector e_o. The trivial case, $a = 0$, corresponds to a rotation of 90 degrees in an appropriate clockwise (if $b \geq 0$) or anti-clockwise (if $b < 0$) direction.

On the other hand, an elementary 2×2 hyperbolic rotation Θ takes a row vector $\mathbf{x} = [\, a \;\; b \,]$ and rotates it to lie either along the basis vector $\mathbf{e}_0 = [1\ 0\,]$ (if $|\,a\,| > |\,b\,|$ or along the basis vector $\mathbf{e}_1 = $ (if $|\,a\,| < |\,b\,|$). More precisely, it performs either of the transformations

$$[\, a \;\; b \,]\,\Theta = [\, \sqrt{|a|^2-|b|^2} \;\; 0 \,] \qquad \text{if } |a| > |b| \tag{2a}$$

$$[\, a \;\; b \,]\,\Theta = [\, 0 \;\; \sqrt{|b|^2-|a|^2} \,] \qquad \text{if } |a| < |b| \tag{2b}$$

The quantity $\sqrt{\pm(|a|^2 - |b|^2)}$ that appears on the right-hand side of the above expressions is consistent with the fact that the prearray, [a b], and the postarrays must have equal **J**-norms. By the **J**-norm of a row vector $\mathbf{x}$, we mean the indefinite quantity $\mathbf{x}\,\mathbf{J}\,\mathbf{x}^*$, which can be positive, negative, or even zero. Here,

$$\mathbf{J} = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} = (1 \oplus -1)$$

An expression for a **J**-unitary hyperbolic rotation Θ that achieves Eq. 2a or 2b is given by:

$$\Theta = \frac{1}{\sqrt{1 - |\rho|^2}} \begin{bmatrix} 1 & -\rho \\ -\rho^* & 1 \end{bmatrix}$$

$$\text{where} \quad \rho = \frac{b}{a},\ a \neq 0,\ |a| > |b| \tag{2c}$$

$$\Theta = \frac{1}{\sqrt{1 - |\rho|^2}} \begin{bmatrix} 1 & -\rho \\ -\rho^* & 1 \end{bmatrix}$$

$$\text{where} \quad \rho^* = \frac{a}{b},\ b \neq 0,\ |a| < |b| \tag{2d}$$

Also note that, in the case of real data, a general hyperbolic rotation as in Eqs. 2c or 2d can be expressed in the alternative form:

$$\Theta = \begin{bmatrix} ch & -sh \\ -sh & ch \end{bmatrix}$$

where the so-called hyperbolic cosine and sine parameters, ch and sh, respectively, are defined by

$$ch = \frac{1}{\sqrt{1-|\rho|^2}} \quad , \quad sh = \frac{\rho}{\sqrt{1-|\rho|^2}}$$

This justifies the name *hyperbolic rotation* for Θ, since the effect of Θ is to rotate the original vector $\mathbf{x}$ along the *hyperbola* of equation $x^2 - y^2 = |a|^2 - |b|^2$, by an angle θ determined by the inverse of the above hyperbolic cosine and/or sine parameters, $\theta = \tanh^{-1}\rho$, in order to align it with the appropriate basis vector. Note also that the special case $|a|=|b|$ corresponds to a row vector $\mathbf{x} = [\, a \;\; b\,]$ with zero hyperbolic norm since $|a|^2 - |b|^2 = 0$. It is then easy to see that there does not exist a hyperbolic rotation that will rotate $\mathbf{x}$ to lie along either bases vectors.

The above expressions for the circular and hyperbolic rotations involve square-root operations. In many situations,

however, it may be desirable to avoid the computation of square-roots since it is usually very expensive. For this and other reasons, square-root and division-free versions of the above elementary rotations were also developed and constitute an attractive alternative (e.g., [36-38]). One could also use unitary or **J**-unitary Householder reflections to simultaneously annihilate several entries in a row, e.g., to transform [x x x x] directly to the form [x' 0 0 0] (e.g., [33-35]). Combinations of rotations and reflections can also be used.

Rotation Example: Modularity and Parallelizability

Consider, for example, a 4-column (real) prearray **A** along with a 4×4 signature matrix **J**,

$$\mathbf{A}=\begin{bmatrix} a & b & c & d \\ x & x & x & x \\ x & x & x & x \\ \vdots & \vdots & \vdots & \vdots \\ x & x & x & x \end{bmatrix}, \quad \mathbf{J}=\begin{bmatrix} 1 & & & \\ & 1 & & \\ & & -1 & \\ & & & -1 \end{bmatrix}$$

and assume that we are interested in applying a **J**-unitary transformation Θ to **A** in order to align its first row along the basis vector $\mathbf{e}_0$, viz., we want

$$\mathbf{A}\,\Theta=\begin{bmatrix} \alpha & 0 & 0 & 0 \\ x' & x' & x' & x' \\ x' & x' & x' & x' \\ \vdots & \vdots & \vdots & \vdots \\ x' & x' & x' & x' \end{bmatrix}$$

Then one way to achieve this, among many possible options, would be the following: we first annihilate the b entry by using a circular rotation that leaves unchanged the last two columns of the prearray,

$$\begin{bmatrix} a & b & c & d \\ x & x & x & x \\ x & x & x & x \\ \vdots & \vdots & \vdots & \vdots \\ x & x & x & x \end{bmatrix}\begin{bmatrix} c_0 & s_0 & & \\ s_0 & -c_0 & & \\ & & 1 & \\ & & & 1 \end{bmatrix}=\begin{bmatrix} a_1 & 0 & c & d \\ \# & \# & x & x \\ \# & \# & x & x \\ \vdots & \vdots & \vdots & \vdots \\ \# & \# & x & x \end{bmatrix}$$

We then annihilate the d entry by using a second circular rotation that leaves unchanged the first two columns of the prearray,

$$\begin{bmatrix} a_1 & 0 & c & d \\ \# & \# & x & x \\ \# & \# & x & x \\ \vdots & \vdots & \vdots & \vdots \\ \# & \# & x & x \end{bmatrix}\begin{bmatrix} 1 & & & \\ & 1 & & \\ & & c_1 & s_1 \\ & & s_1 & -c_1 \end{bmatrix}=\begin{bmatrix} a_1 & 0 & c_1 & 0 \\ \# & \# & * & * \\ \# & \# & * & * \\ \vdots & \vdots & \vdots & \vdots \\ \# & \# & x & * \end{bmatrix}$$

We finally annihilate the c_1 entry by using a hyperbolic rotation, which leaves unchanged the second and fourth columns of the prearray (assuming $|c_1| < |a_1|$),

$$\begin{bmatrix} a_1 & 0 & c_1 & 0 \\ \# & \# & * & * \\ \# & \# & * & * \\ \vdots & \vdots & \vdots & \vdots \\ \# & \# & * & * \end{bmatrix}\begin{bmatrix} ch_1 & & -sh_1 & \\ & 1 & & \\ -sh_1 & & ch_1 & \\ & & & 1 \end{bmatrix}=\begin{bmatrix} \alpha & 0 & 0 & 0 \\ y & \# & y & * \\ y & \# & y & * \\ \vdots & \vdots & \vdots & \vdots \\ y & \# & y & * \end{bmatrix}$$

The parameters that define the previous rotations are clearly given by

$$c_0=\frac{a}{\sqrt{|a|^2+|b|^2}}\ , \quad s_0=\frac{b}{\sqrt{|a|^2+|b|^2}}\ ,$$

$$c_1=\frac{c}{\sqrt{|c|^2+|d|^2}}\ , \quad s_1=\frac{d}{\sqrt{|c|^2+|d|^2}}\ ,$$

$$ch_1=\frac{a_1}{\sqrt{|a_1|^2-|c|^2}}\ , \quad sh_1=\frac{c_1}{\sqrt{|a_1|^2-|c_1|^2}}\ .$$

The different adaptive schemes can be rewritten in terms of square-root arrays, where the necessary operations are elementary rotations as described above. Such array descriptions lend themselves rather directly to parallelizable and modular implementations. Indeed, once a rotation matrix is chosen, then all the rows of the prearray undergo the same rotation transformation and can thus be processed in parallel. Returning to the above example, where we started with the prearray **A**, we see that the parameters of the first rotation, (c_0, s_0), are completely determined by the entries a and b. Once the first rotation is specified, all the rows of **A** are then transformed by it, and can thus be processed in parallel, and by the same functional (rotation) block, to obtain the desired postarray. The procedure continues by determining the second rotation and so on.

Two RLS Algorithms in Array Forms

To show the compactness and simplicity the array form can bring, we display in Table 3 the usual set of equations for the so-called FTF algorithm (the table and notation is taken from [1], p. 591, for illustrative purposes). Table 4 shows the corresponding square-root array form that will be derived later. Tables 5 and 6 show the corresponding forms for the so-called QRD-LSL algorithm (Table 5 is extracted from [1], p. 664). As it turns out, there are clearly significant (conceptual, if nothing else) advantages in the more compact forms. Some authors have deduced the compact array forms by a careful study and reorganization of the explicit sets of equations in Tables 3 and 5 ([3], p. 465, and [22]). In our approach, the difference is that we can immediately write down the compact forms of Tables 4 and 6, with the equations in Tables 3 and 5 being only one among several sets of (cumbersome) explicit equations that one could derive, if desired.

Stochastic and Deterministic Least-Squares Problems

The application of the least-squares criterion to estimation problems in both the deterministic and the stochastic settings

Table 3: The FTF algorithm in explicit form.

$$\eta_M(n) = \mathbf{a}_M^*(n-1)\mathbf{u}_{M+1}(n)$$
$$f_M(n) = \gamma_M(n-1)\eta_M(n)$$
$$F_M(n) = \lambda F_M(n-1) + \eta_M(n)f_M^*(n)$$
$$\gamma_{M+1}(n) = \lambda\frac{F_M(n-1)}{f_M(n)}\gamma_M(n-1)$$
$$\bar{\mathbf{k}}_{M+1}(n) = \begin{bmatrix} 0 \\ \bar{\mathbf{k}}_M(n-1) \end{bmatrix} + \lambda^{-1}\frac{\eta_M(n)}{F_M(n-1)}\mathbf{a}_M(n-1)$$
$$\mathbf{a}_M(n) = \mathbf{a}_M(n-1) - f_M^*(n)\begin{bmatrix} 0 \\ \bar{\mathbf{k}}_M(n-1) \end{bmatrix}$$
$$\Psi_M(n) = \lambda B_M(n-1)\bar{k}_{M+1,M+1}(n)$$
$$\gamma_M(n) = \left[1 - \Psi_M^*(n)\gamma_{M+1}(n)\bar{k}_{M+1,M+1}(n)\right]^{-1}\gamma_{M+1}(n)$$
$$\text{RescueVariable} = \left[1 - \Psi_M^*(n)\gamma_{M+1}(n)\gamma_{M+1}(n)\bar{k}_{M+1,M+1}(n)\right]$$
$$b_M(n) = \gamma_M(n)\Psi_M(n)$$
$$B_M(n) = \lambda B_M(n-1) + \Psi_M(n)b_M^*(n)$$
$$\begin{bmatrix} \bar{\mathbf{k}}_M(n) \\ 0 \end{bmatrix} = \bar{\mathbf{k}}_{M+1}(n) - \bar{k}_{M+1,M+1}(n)\mathbf{c}_M(n-1)$$
$$\mathbf{c}_M(n) = \mathbf{c}_M(n-1) - b_M^*(n)\begin{bmatrix} \bar{\mathbf{k}}_M(n) \\ 0 \end{bmatrix}$$
$$\alpha_M(n) = d(n) - \hat{\mathbf{w}}_M^*(n-1)\mathbf{u}_M(n)$$
$$e_M(n) = \gamma_M(n)\alpha_M(n)$$
$$\hat{\mathbf{w}}_M(n) = \hat{\mathbf{w}}_M(n-1) + \bar{\mathbf{k}}_M(n)e_M^*(n)$$

Table 4: A square-root version of the FTF algorithm.

$$\begin{bmatrix} r_e^{1/2}(i) & \bar{\mathbf{h}}_{i+1}\mathbf{L}_i \\ \begin{bmatrix} 0 \\ \bar{\mathbf{c}}_i \end{bmatrix} & \lambda^{-1/2}\mathbf{L}_i \end{bmatrix}\Theta_i = \begin{bmatrix} r_e^{1/2}(i+1) & 0 \\ \begin{bmatrix} \bar{\mathbf{c}}_{i+1} \\ 0 \end{bmatrix} & \mathbf{L}_{i+1} \end{bmatrix}$$

will play a central role in our development. Certainly, several other optimization criteria can be used for estimation problems, but for signal processing one of the most important, at least in the sense of having had the most applications, is the linear least-squares criterion. This striking criterion was perhaps first developed by Gauss in his work on celestial mechanics [39]. It was later successfully employed by Wiener [40] in the early 1940s in the context of stationary processes observed over infinite or semi-infinite intervals. But in the late 1950s, interest shifted to nonstationary processes with known finite-dimensional state-space models, where the availability of the computationally efficient recursive Kalman-Bucy filter [28, 41] has essentially monopolized the field in the last few decades. We thus include a brief overview of the topic (e.g., [27, 29, 30]).

The Stochastic Problem

Let $\mathbf{z}$ be an $n\times 1$ zero-mean random vector, and let $\mathbf{y}$ be a column of $(N+1)$ zero-mean random vectors of size $p\times 1$ each,

$$\mathbf{y} = \begin{bmatrix} \mathbf{y_0} \\ \mathbf{y}_1 \\ \vdots \\ \mathbf{y}_N \end{bmatrix}$$

Table 5: The QRD-LSL algorithm in explicit form.

For time $n = 1,2,\ldots$ and order $m = 1,2,\ldots$ M, repeat

$$B_{m-1}(n-1) = \lambda B_{m-1}(n-2) = |\varepsilon_{b,m-1}(n-1)|^2$$
$$c_{b,m-1}(n-1) = \frac{\lambda^{1/2}B_{m-1}^{1/2}(n-2)}{B_{m-1}^{1/2}(n-1)}$$
$$s_{b,m-1}(n-1) = \frac{\varepsilon_{b,m-1}^*(n-1)}{B_{m-1}^{1/2}(n-1)}$$
$$\varepsilon_{f,m}(n) = c_{b,m-1}(n-1)\varepsilon_{f,m-1}(n) - s_{b,m-1}^*(n-)\lambda^{1/2}\pi_{f,m-1}^*(n-1)$$
$$\pi_{f,m-1}^*(n) = c_{b,m-1}(n-1)\lambda^{1/2}\pi_{f,m-1}^*(n-1) = s_{b,m-1}(n-1)\varepsilon_{f,m-1}(n)$$
$$\gamma_m^{1/2}(n-1) = c_{b,m-1}(n-1)\gamma_{m-1}^{1/2}(n-1)$$
$$F_{m-1}(n) = \lambda F_{m-1}(n-1) + |\varepsilon_{f,m-1}(n)|^2$$
$$c_{f,m-1}(n) = \frac{\lambda^{1/2}F_{m-1}^{1/2}(n-1)}{F_{m-1}^{1/2}(n)}$$
$$s_{f,m-1}(n) = \frac{\varepsilon_{f,m-1}^*(n)}{F_{m-1}^{1/2}(n)}$$
$$\varepsilon_{b,m}(n) = c_{f,m-1}(n)\varepsilon_{b,m-1}(n-1) - s_{f,m-1}^*(n)\lambda^{1/2}\pi_{b,m-1}^*(n-1)$$
$$\pi_{b,m-1}^*(n) = c_{f,m-1}(n)\lambda^{1/2}\pi_{b,m-1}^*(n-1) + s_{f,m-1}(n)\varepsilon_{b,m-1}(n-1)$$

For time $n = 1,2,\ldots$, repeat

$$B_M(n) = \lambda B_M(n-1) + |\varepsilon_{b,M}(n)|^2$$
$$c_{b,M}(n) = \frac{\lambda^{1/2}B_M^{1/2}(n-1)}{B_M^{1/2}(n)}$$
$$s_{b,M}(n) = \frac{\varepsilon_{b,M}^*(n)}{B_M^{1/2}(n)}$$
$$\varepsilon_{M+1}(n) = c_{b,M}(n)\varepsilon_M(n) - s_{b,M}^*(n)\lambda^{1/2}p_M^*(n-1)$$
$$p_M^*(n) = c_{b,M}(n)\lambda^{1/2}p_M^*(N-1) + s_{b,M}(n)\varepsilon_M(n)$$
$$\gamma_{M+1}^{1/2}(n) = c_{b,M}(n)\gamma_M^{1/2}(n)$$
$$e_{M+1}(n) = \gamma_{M+1}^{1/2}(n)\varepsilon_{M+1}(n)$$

For $m = 1,2,\ldots, M$, set

$$\pi_{f,m-1}(0) = \pi_{b,m-1}(0) = 0,\ p_m(0) = 0$$

For $m = 0,1,\ldots, M$, set

$$B_m(-1) = F_m(0) = \delta > 0$$

For $n = 1,2,\ldots$, compute

$$\varepsilon_{f,0}(n) = \varepsilon_{b,0}(n) = u(n),\ \varepsilon_0(n) = d(n),\ \gamma_0(n) = 1.$$

Table 6: Square-root QRD-LSL algorithm.

Set $\bar{q}_M^f(0) = \bar{q}_M^b(-1) = 0$, $\bar{\Phi}_M^f(0) = \bar{\Phi}_M^b(-1) = \frac{1}{\sqrt{\mu}} > 0$

For each $N \geq 0$, set $\bar{f}_0(N) = \bar{b}_0(N) = u(N)$, $\gamma_0(N) = 1$, and repeat for $M \geq 0$

$$\begin{bmatrix} \sqrt{\lambda}\,\bar{\Phi}_M^{b/2}(N-1) & \bar{b}_M^*(N) \\ \sqrt{\lambda}\,\bar{q}_M^{*B}(N-1) & \bar{f}_M^*(N+1) \\ 0 & \gamma_M^{1/2}(N) \end{bmatrix}\bar{\Theta}_{M,N}^b = \begin{bmatrix} \bar{\Phi}_M^{b/2}(N) & 0 \\ \bar{q}_M^{*b}(N) & \bar{f}_{M+1}^*(N+1) \\ b_M(N)\bar{\Phi}_M^{*b/2}(N) & \gamma_{M+1}^{1/2}(N) \end{bmatrix},$$

$$\begin{bmatrix} \sqrt{\lambda}\,\bar{\Phi}_M^{f/2}(N) & \bar{f}_M^*(N+1) \\ \sqrt{\lambda}\,\bar{q}_M^{*f}(N) & \bar{b}_M^*(N) \\ 0 & \gamma_M^{1/2}(N) \end{bmatrix}\bar{\Theta}_{M,N+1}^f = \begin{bmatrix} \bar{\Phi}_M^{f/2}(N+1) & 0 \\ \bar{q}_M^{*f}(N+1) & \bar{b}_{M+1}^*(N+1) \\ f_M(N+1)\bar{\Phi}_M^{*f/2}(N+1) & \gamma_{M+1}^{1/2}(N+1) \end{bmatrix}$$

$\Theta_{M,N+1}^f$ and $\Theta_{M,N}^b$ are unitary rotations

Relations between the RLS and Kalman Variables

The different variants of the Kalman recursions, when applied to (Eq. 21c), will now provide different algorithms for the solution of the RLS minimization problem (Eq. 19b). But note that the variables used in (Eq. 21c) are scaled versions of the variables used in (Eq. 19b). For ex-

ample, $y(i)$ is a scaled version of $d(i)$. So is $\mathbf{x}_i$ relative to $\mathbf{w}$. This means that when writing down the state-space estimation algorithms that correspond to (Eq. 21c), we should then proceed to replace the variables of (Eq. 21c) by the corresponding RLS variables. This would allow us to describe the algorithms in terms of the original RLS variables. This section is meant to clarify the connections between the RLS variables $\{d(i), \mathbf{w}, \mathbf{u}_i, \Pi_0, \Phi_i, \mathbf{w}_i\}$ and the Kalman variables.

The problem of interest is to estimate the value of $\mathbf{z}$ by means of a linear operation on *observed* values of the $\{\mathbf{y}_i\}$, say,

$$\begin{aligned}\hat{\mathbf{z}} &= \mathbf{K}_0\mathbf{y}_0 + \mathbf{K}_1\mathbf{y}_1 + \ldots + \mathbf{K}_N\mathbf{y}_N \\ &= [\,\mathbf{K}_0\,\mathbf{K}_1\,\ldots\,\mathbf{K}_n\,]\,\mathbf{y} = \mathbf{K}\mathbf{y}\end{aligned} \tag{3a}$$

where the $\mathbf{K}_i$ are $n\times p$ matrices, and $\mathbf{K}$ is thus a constant matrix to be determined so as to minimize the trace of the mean-square error (m.s.e.) matrix defined by

$$\text{m.s.e.} = E\,[(\mathbf{z}-\mathbf{K}\mathbf{y})\,(\mathbf{z}-\mathbf{K}\mathbf{y})^*]$$

Here, E denotes expected value. The optimal solution is obtained by requiring that the error variable, $\mathbf{z} - \mathbf{K}\mathbf{y}$, be uncorrelated with the observation vector $\mathbf{y}$, viz.,

$$E\,[(\mathbf{z}-\mathbf{K}\mathbf{y})\,\mathbf{y}^*] = \mathbf{0}$$

which leads to $\mathbf{K}(E\mathbf{y}\mathbf{y}^*) = E\mathbf{z}\mathbf{y}^*$. That is, the optimal linear least-mean squares estimate (l.l.m.s.e., for short) of $\mathbf{z}$ reduces to solving the linear system of equations,

$$\mathbf{K}\mathbf{R}_y = \mathbf{R}_{zy} \tag{3b}$$

where we have denoted the autocorrelation matrix $E\mathbf{y}\mathbf{y}^*$ by $\mathbf{R}_y$ and the cross-correlation vector $E\mathbf{z}\mathbf{y}^*$ by $\mathbf{R}_{zy}$. Assuming the invertibility of $\mathbf{R}_y$, we get the following expression for $\hat{\mathbf{z}}$,

$$\hat{\mathbf{z}} = \mathbf{R}_{zy}\mathbf{R}_y^{-1}\mathbf{y}\ . \tag{3c}$$

An extremely important special case, which will be central to our later analysis, occurs when the observation vector $\mathbf{y}$ and the variable $\mathbf{z}$ are *linearly* related such as

$$\mathbf{y} = \mathbf{A}\mathbf{z} + \mathbf{v}\ . \tag{4a}$$

Here, $\mathbf{A}$ stands for a constant matrix of appropriate dimensions ($(N+1)p\times n$)), and $\mathbf{v}$ is a zero-mean random noise vector with autocorrelation matrix $\mathbf{R}_v$, $E\mathbf{v}\mathbf{v}^* = \mathbf{R}_v$, and which is assumed to be uncorrelated with $\mathbf{z}$, $E\mathbf{z}\mathbf{v}^* = \mathbf{0}$. The autocorrelation matrix of $\mathbf{z}$ is also assumed to be known, $E\mathbf{z}\mathbf{z}^* = \mathbf{R}_z$. Under these assumptions, the autocorrelation matrix $\mathbf{R}_y$ and the cross-correlation vector $\mathbf{R}_{zy}$ in Eq. 3b can be evaluated in terms of the known quantities $\{\mathbf{A}, \mathbf{R}_z, \mathbf{R}_v\}$. Indeed,

$$\mathbf{R}_y = \mathbf{A}\mathbf{R}_z\mathbf{A}^* + \mathbf{R}_v, \quad \mathbf{R}_{zy} = \mathbf{R}_z\mathbf{A}^*$$

The l.l.m.s.e. of $\mathbf{z}$ will then be given by

$$\hat{\mathbf{z}} = \mathbf{R}_z\mathbf{A}^*\,[\mathbf{A}\mathbf{R}_z\mathbf{A}^* + \mathbf{R}_v]^{-1}\,\mathbf{y} \tag{4b}$$

If it further happens that $\mathbf{R}_z$ and $\mathbf{R}_v$ are nonsingular matrices, then the above expression for $\hat{\mathbf{z}}$ can be rewritten in an equivalent form that will be convenient for later analysis,

$$\hat{\mathbf{z}} = [\mathbf{R}_z^{-1} + \mathbf{A}^*\mathbf{R}_v^{-1}\,\mathbf{A}]^{-1}\,\mathbf{A}^*\mathbf{R}_v^{-1}\mathbf{y} \tag{4c}$$

In deriving Eq. 4c from 4b, we employed a very useful matrix identity, often known as the matrix inversion lemma ([42], p. 656]). It states that if $\mathbf{E}$ and $\mathbf{C}$ are two invertible matrices then

$$(\mathbf{E}+\mathbf{B}\mathbf{C}\mathbf{D})^{-1} = \mathbf{E}^{-1} - \mathbf{E}^{-1}\mathbf{B}(\mathbf{D}\mathbf{E}^{-1}\mathbf{B} + \mathbf{C}^{-1})^{-1}\mathbf{D}\mathbf{E}^{-1} \tag{5}$$

This can be verified by multiplying the right-hand side by $(\mathbf{E}+\mathbf{B}\mathbf{C}\mathbf{D})$ and verifying that the identity matrix results. The results of this discussion are summarized in Table 7.

Table 7: Linear least mean-squares estimation: The stochastic zero-mean case.

Given Data	l.l.m.s.e of z
$\{\mathbf{z},\mathbf{y}\}$ $\{\mathbf{R}_y, \mathbf{R}_{zy}\}$ $E\mathbf{z} = E\mathbf{y} = 0$.	$\hat{\mathbf{z}} = \mathbf{R}_{zy}\mathbf{R}_y^{-1}\,\mathbf{y}$.
$\mathbf{y} = \mathbf{A}\mathbf{z} + \mathbf{v}$ $\{\mathbf{R}_z, \mathbf{R}_v\}$ $E\mathbf{z} = E\mathbf{y} = E\mathbf{v} = 0,\ E\mathbf{z}\mathbf{v}^* = 0$	$\hat{\mathbf{z}} = \mathbf{R}_z\mathbf{A}^*[\mathbf{A}\mathbf{R}_z\mathbf{A}^* + \mathbf{R}_v]^{-1}\,\mathbf{y}$ or $\hat{\mathbf{z}} = [\mathbf{R}_z^{-1} + \mathbf{A}^*\,\mathbf{R}_v^{-1}\mathbf{A}]^{-1}\,\mathbf{A}^*\mathbf{R}_v^{-1}\,\mathbf{y}$

Non-Zero Means

What if the random variables $\mathbf{z}$ and $\mathbf{y}$ in Eq. 3a are not zero-mean, say $E\mathbf{z} = \bar{\mathbf{z}}$, $E\mathbf{y} = \bar{\mathbf{y}}$? The problem of interest then corresponds to estimating $\mathbf{z} - \bar{\mathbf{z}}$ as a linear combination of the entries of $\mathbf{y} - \bar{\mathbf{y}}$. If we introduce the change of variables $\mathbf{z}' = (\mathbf{z} - \bar{\mathbf{z}})$ and $\mathbf{y}' = (\mathbf{y} - \bar{\mathbf{y}})$, then this is equivalent to estimating a zero-mean vector $\mathbf{z}'$ from a linear combination of the entries of a zero-mean vector $\mathbf{y}'$. This is precisely the setting studied in the previous section and, according to Eq. 3b, the solution is given by $\mathbf{z}' = \mathbf{K}'\mathbf{y}'$ where $\mathbf{K}'$ is obtained by solving the linear system of equations

$$\mathbf{K}'\mathbf{R}_{y'} = \mathbf{R}_{z'y'},\ \mathbf{R}_{y'} = E\mathbf{y}'\mathbf{y}'^*,\ \mathbf{R}_{z'y'} = E\mathbf{z}'\mathbf{y}'^*\ .$$

The matrix $\mathbf{R}_{y'}$ is equal to the covariance matrix of $\mathbf{y}$, where by the covariance matrix of the random variable $\mathbf{y}$, denoted by $\text{cov}(\mathbf{y})$, we mean $E(\mathbf{y} - \bar{\mathbf{y}})\,(\mathbf{y} - \bar{\mathbf{y}})^*$. Also, $\mathbf{R}_{z'y'}$ is equal to the cross-covariance vector $\text{cov}(\mathbf{z},\mathbf{y}) = E(\mathbf{z} - \bar{\mathbf{z}})(\mathbf{y} - \bar{\mathbf{y}})^*$. Therefore, the l.l.m.s.e. estimate of $\mathbf{z}$ satisfies

$$(\hat{\mathbf{z}} - \bar{\mathbf{z}}) = \text{cov}(\mathbf{z},\mathbf{y})\,\text{cov}^{-1}(\mathbf{y})\,[\mathbf{y} - \bar{\mathbf{y}}] \tag{6a}$$

Comparing with Eq. 3c, we see that the non-zero mean case simply corresponds to replacing $\hat{\mathbf{z}}$ and $\hat{\mathbf{y}}$, by $(\hat{\mathbf{z}}-\bar{\mathbf{z}})$ and $(\mathbf{y}-\bar{\mathbf{y}})$ respectively, as well as replacing the autocorrelation and cross-correlation matrices $\mathbf{R}_{\mathbf{y}}$ and $\mathbf{R}_{\mathbf{zy}}$ by the covariance and cross-covariance matrices cov(**y**) and cov(**z**,**y**), respectively. In the important special case of a linear relation between **y** and **z** as in Eq. 4a, where we now assume cov(**z**, **v**) = 0, Eqs. 4b and 4c should be replaced by (using $\bar{\mathbf{y}} = \mathbf{A}\bar{\mathbf{z}} + \bar{\mathbf{v}}$)

$$\hat{\mathbf{z}} = \bar{\mathbf{z}} + \text{cov}(\mathbf{z})\,\mathbf{A}^*\,[\mathbf{A}\text{cov}(\mathbf{z})\mathbf{A}^* + \text{cov}(\mathbf{v})]^{-1}\,[\mathbf{y} - \mathbf{A}\bar{\mathbf{z}} - \bar{\mathbf{v}}] \tag{7a}$$

$$\hat{\mathbf{z}} = \bar{\mathbf{z}} + [\text{cov}^{-1}(\mathbf{z}) + \mathbf{A}^*\text{cov}^{-1}(\mathbf{v})\mathbf{A}]^{-1}\,\mathbf{A}^*\text{cov}^{-1}(\mathbf{v})\,[\mathbf{y} - \mathbf{A}\bar{\mathbf{z}} - \bar{\mathbf{v}}] \tag{7b}$$

The results of this section are summarized in Table 8. We now move on to review the deterministic version of the least-squares problem and highlight connections with the stochastic point of view.

Table 8: Linear least mean-squares estimation: The stochastic non-zero-mean case.

Given Data	l.l.m.s.e of z
{**z**,**y**} cov (**y**), cov (**z**) $E\mathbf{z} = \bar{\mathbf{z}}$, $E\mathbf{y} = \bar{\mathbf{y}}$	$\hat{\mathbf{z}} = \bar{\mathbf{z}} + \text{cov}(\mathbf{z},\mathbf{y})\,\text{cov}^{-1}(\mathbf{y})\,[\mathbf{y} - \bar{\mathbf{y}}]$
$\mathbf{y} = \mathbf{A}\mathbf{z} + \mathbf{v}$ {cov(**z**), cov (**v**)} cov(**z**,**v**) = 0 $E\mathbf{z} = \bar{\mathbf{z}}$, $E\mathbf{y} = \bar{\mathbf{y}}$, $E\mathbf{v} = \bar{\mathbf{v}}$	$\hat{\mathbf{z}} = \bar{\mathbf{z}} + \text{cov}(\mathbf{z})\,\mathbf{A}^*[\,\mathbf{A}\text{cov}(\mathbf{z})\,\mathbf{A}^* + \text{cov}(\mathbf{v})]^{-1}\,[\mathbf{y} - \mathbf{A}\bar{\mathbf{z}} - \bar{\mathbf{v}}]$ or $\hat{\mathbf{z}} = \bar{\mathbf{z}} + [\text{cov}^{-1}(\mathbf{z}) + \mathbf{A}^*\text{cov}^{-1}(\mathbf{v})\,\mathbf{A}]^{-1}\mathbf{A}^*\text{cov}^{-1}(\mathbf{v})\,[\mathbf{y} - \mathbf{A}\bar{\mathbf{z}} - \bar{\mathbf{v}}]$

The Deterministic Problem

We now let **z** represent a column vector of *n unknown* parameters, rather than a vector of random variables. We are further given $(N+1)$ *noisy* measurements $\{\mathbf{y}_i\}$ that are assumed to be linearly related to **z** as follows

$$\mathbf{y}_i = \mathbf{A}_i\,\mathbf{z} + \mathbf{v}_i$$

That is, the noise component is assumed to be additive. We are then required to estimate **z** from the measurements $\{\mathbf{y}_i\}$ as we further elaborate.

The $(N+1)$ measurements are grouped together into a single matrix expression:

$$\begin{bmatrix}\mathbf{y}_0\\ \mathbf{y}_1\\ \vdots\\ \mathbf{y}_N\end{bmatrix} = \begin{bmatrix}\mathbf{A}_0\\ \mathbf{A}_1\\ \vdots\\ \mathbf{A}_N\end{bmatrix}\mathbf{z} + \begin{bmatrix}\mathbf{v}_0\\ \mathbf{v}_1\\ \vdots\\ \mathbf{v}_N\end{bmatrix}$$

or, more compactly,

$$\mathbf{y} = \mathbf{A}\mathbf{z} + \mathbf{v} \tag{8a}$$

The basic distinction between the earlier relation (Eq. 4a) and Eq. 8a is that the quantities in the latter expression are all assumed to be deterministic, while Eq. 4a involves random variables. This distinction will become more apparent as we proceed.

Because of the noise component **v** in Eq. 8a, the observed vector **y** does not lie in the range space of the matrix **A**. Our objective then is to determine an estimate for **z**, say $\hat{\mathbf{z}}$, in order to minimize the square of the distance between **y** and $\mathbf{A}\hat{\mathbf{z}}$, viz.,

$$\min_{\mathbf{z}} \|\mathbf{y} - \mathbf{A}\mathbf{z}\|_2^2 \tag{8b}$$

The resulting $\hat{\mathbf{z}}$ is often called the least-squares solution, while $\mathbf{A}\hat{\mathbf{z}}$ is called the linear least-squares estimate (l.l.s.e.) of **y**.

The solution to Eq. 8b follows from a simple projection argument: the vector **y** does not necessarily lie in the range space of the matrix **A**. The orthogonal projection of **y** onto this range space yields a vector $\hat{\mathbf{y}}$ that is closest to **y** in the least-squares sense, since the resulting error vector $(\mathbf{y} - \hat{\mathbf{y}})$ will be orthogonal to the range of **A**. The orthogonality condition is equivalent to $\mathbf{A}^*(\mathbf{y} - \hat{\mathbf{y}}) = 0$. That is (and replacing $\hat{\mathbf{y}}$ by $\mathbf{A}\hat{\mathbf{z}}$,

$$\mathbf{A}^*(\mathbf{y} - \mathbf{A}\hat{\mathbf{z}}) = 0$$

which implies that the least-squares solution can be obtained by solving the linear system of equations

$$\mathbf{A}^*\mathbf{A}\hat{\mathbf{z}} = \mathbf{A}^*\mathbf{y}$$

If **A** is further assumed to be full rank, then we can alternatively write

$$\hat{\mathbf{z}} = (\mathbf{A}^*\mathbf{A})^{-1}\,\mathbf{A}^*\mathbf{y} \tag{8c}$$

A more general optimization criterion that is often used instead of Eq. 8b is the following:

$$\min_{\mathbf{z}} [(\mathbf{z} - \mathbf{z}_0)^*\Pi_0^{-1}(\mathbf{z} - \mathbf{z}_0) + \|\mathbf{y} - \mathbf{A}\mathbf{z}\|_2^2] \tag{9a}$$

This is still a quadratic cost function in the unknown vector **z**, but it includes the additional term $(\mathbf{z} - \mathbf{z}_0)^*\Pi_0^{-1}(\mathbf{z} - \mathbf{z}_0)$, where Π_0 is a given positive-definite (weighting) matrix and $\mathbf{z}_0$ is also a given vector. Choosing $\Pi_0 = \infty\,\mathbf{I}$ leads us back to the original expression. (8b)

The point is that the freedom in choosing Π_0 allows us to incorporate additional apriori knowledge into the statement of the problem. Indeed, different choices for Π_0 will indicate how confident we are about the closeness of the optimal

solution $\hat{\mathbf{z}}$ to the given vector $\mathbf{z}_0$. Assume, for example, that we set $\Pi_0 = \varepsilon\, \mathrm{I}$, where ε is a very small positive number. Then, the first term in the new cost function (Eq. 9a) becomes dominant. It is thus not hard to see that, in this case, the cost will be minimized if we choose $\hat{\mathbf{z}}$ close enough to $\mathbf{z}_0$ in order to annihilate the effect of the first term. In simple words, a "small" Π_0 reflects a high confidence that $\mathbf{z}_0$ is a good and close enough guess for the optimal solution $\hat{\mathbf{z}}$. On the other hand, a "large" Π_0 indicates a high degree of uncertainty in the initial guess $\mathbf{z}_0$.

To facilitate the solution of Eq. 9a, we introduce the change of variables $\mathbf{z}' = \mathbf{z} - \mathbf{z}_0$ and $\mathbf{y}' = \mathbf{y} - \mathbf{A}\mathbf{z}_0$. Then, Eq. 9a becomes:

$$\min_{\mathbf{z}'} \left[\mathbf{z}'^{*} \Pi_0^{-1} \mathbf{z}' + \|\mathbf{y}' - \mathbf{A}\mathbf{z}'\|_2^2 \right]$$

Differentiating with respect to $\mathbf{z}'$ and setting the gradient equal to zero yields the solution

$$\hat{\mathbf{z}}' = \left[\Pi_0^{-1} + \mathbf{A}^{*}\mathbf{A} \right]^{-1} \mathbf{A}^{*}\mathbf{y}'$$

Substituting for $\hat{\mathbf{z}}' = \hat{\mathbf{z}} - \mathbf{z}_0$ and $\mathbf{y}' = \mathbf{y} - \mathbf{A}\mathbf{z}_0$ leads to the optimal solution

$$\hat{\mathbf{z}} = \mathbf{z}_0 + \left[\Pi_0^{-1} + \mathbf{A}^{*}\mathbf{A} \right]^{-1} \mathbf{A}^{*} \left[\mathbf{y} - \mathbf{A}\mathbf{z}_0 \right] \qquad (9b)$$

We also see here that instead of requiring the invertibility of $\mathbf{A}^{*}\mathbf{A}$, as in Eq. 8c, we now require the invertibility of the matrix $[\Pi_0^{-1} + \mathbf{A}^{*}\mathbf{A}]$ .. This is yet another reason in favor of the modified criterion (Eq. 9a), since it allows us to relax the full rank condition on $\mathbf{A}$. The results of this discussion are summarized in Table 9.

Table 9: Linear least-squares estimation: The deterministic case.

Optimization Problem	Solution
$\{\mathbf{z},\mathbf{y}\}$ $\min_{\mathbf{z}} \|\mathbf{y} - \mathbf{A}\mathbf{z}\|_2^2$ $\mathbf{A}$ full rank	$\hat{\mathbf{z}} = (\mathbf{A}^{*}\mathbf{A})^{-1}\mathbf{A}^{*}\mathbf{y}$
$\{\mathbf{z},\mathbf{y},\mathbf{z}_0,\Pi_0\}$ $\min_{\mathbf{z}} [(\mathbf{z} - \mathbf{z}_0)^{*}\Pi_0^{-1}(\mathbf{z} - \mathbf{z}_0) + \|\mathbf{y} - \mathbf{A}\mathbf{z}\|_2^2]$ Π_0 positive-definite	$\hat{\mathbf{z}} = \mathbf{z}_0 + [\Pi_0^{-1}]$ $+ \mathbf{A}^{*}\mathbf{A}]^{-1}\,\mathbf{A}^{*}\,[\mathbf{y} - \mathbf{A}\mathbf{z}_0]$

Equivalence of the Problems

A comparison of expression (Eq. 9b) with the earlier result (Eq. 7b) is now in order. Equation 7b provides the l.l.m.s.e. of $\mathbf{z}$ in a stochastic framework, while (Eq. 9b) provides the least-squares solution of (Eq. 9a) in a deterministic framework. But it is clear that if we replace the quantities in (Eq. 7b) by:

$$\mathrm{cov}(\mathbf{z}) = \Pi_0,\ \bar{\mathbf{z}} = \mathbf{z}_0,\ \mathrm{cov}(\mathbf{v}) = \mathbf{I},\ \bar{\mathbf{v}} = 0$$

then the stochastic expression (Eq. 7b) coincides with the deterministic solution (Eq. 9b). This equivalence plays a central role in our analysis. It allows us to move back and forth between the deterministic and the stochastic frameworks rather smoothly. Table 10 summarizes the relations between the variables in both frameworks.

We now proceed to assume that the observations $\{\mathbf{y}_i\}$ in the linear model (Eq. 4a) admit an underlying state-space structure. This would then allow us to introduce the Kalman filter as an efficient recursive procedure for the solution of (Eq. 4b).

The Kalman Filter and Adaptive Problems

We try to keep the presentation as simple and straightforward as possible in order to convey the main ideas. The reader is referred to [27, 30] and the many references therein, for further information and discussion on the subject. We shall often limit ourselves to the results that are essential to the discussion in future sections. We consider a $p\times 1$ stochastic process $\{\mathbf{y}_i\}$ that admits an n-dimensional state-space representation of the form

$$\begin{aligned} \mathbf{x}_{i+1} &= \mathbf{F}_i\,\mathbf{x}_i + \mathbf{G}_i\,\mathbf{r}_i \\ \mathbf{y}_i &= \mathbf{H}_i\,\mathbf{x}_i + \mathbf{v}_i, \quad \text{for } i \geq 0 \end{aligned} \qquad (10a)$$

where i is a discrete-time index, $\mathbf{F}_i$, $\mathbf{H}_i$, and $\mathbf{G}_i$ are known $n\times n$, $p\times n$, and $n\times m$ matrices, respectively, and the $\{\mathbf{r}_i\}$ and $\{\mathbf{v}_i\}$ are uncorrelated zero-mean stochastic variables with covariance matrices

$$E\mathbf{v}_i\,\mathbf{v}_j^{*} = \mathbf{R}_i\,\delta_{ij},\ E\mathbf{r}_i\,\mathbf{r}_j^{*} = \mathbf{Q}_i\,\delta_{ij},\ \mathbf{R}_i > 0$$

The symbol δ_{ij} is the Kronecker delta function, which is equal to unity when $i = j$, and zero elsewhere. We also assume that

Table 10: Equivalence of the stochastic and deterministic frameworks.

Stochastic Framework	Deterministic Framework
Stochastic variables $\{\mathbf{z},\mathbf{y}\}, \mathbf{y} = \mathbf{A}\mathbf{z} + \mathbf{v}$	Deterministic variables $\{\mathbf{z},\mathbf{y}\}, \mathbf{y} = \mathbf{A}\mathbf{z} + \mathbf{v}$
$\bar{\mathbf{z}} = E\mathbf{z}$	Initial guess of $\mathbf{z}$, $\mathbf{z}_0$
$\mathrm{cov}(\mathbf{z})$	Weighting matrix Π_0
$\bar{\mathbf{y}} = \mathbf{A}\,\bar{\mathbf{z}}$	Initial guess of $\mathbf{y}$, $\mathbf{A}\mathbf{z}_0$
$\bar{\mathbf{v}} = E\mathbf{v} = 0$, $\mathrm{cov}(\mathbf{v}) = \mathbf{I}$	- - -
$\hat{\mathbf{z}}$	$\hat{\mathbf{z}}$
$\min_{\mathbf{K}} \mathrm{trace}\ \{\mathrm{cov}\ (\mathbf{z} - \mathbf{K}\mathbf{y})\}$	$\min_{\mathbf{z}} [(\mathbf{z} - \mathbf{z}_0)^{*}\Pi_0^{-1}(\mathbf{z} - \mathbf{z}_0) + \|\mathbf{y} - \mathbf{A}\mathbf{z}\|_2^2]$
$\hat{\mathbf{z}} = \bar{\mathbf{z}} + [\mathrm{cov}^{-1}(\mathbf{z}) + \mathbf{A}^{*}\mathbf{A}]^{-1}\,\mathbf{A}^{*}\,[\mathbf{y} - \mathbf{A}\bar{\mathbf{z}}]$	$\hat{\mathbf{z}} = \mathbf{z}_0 + [\Pi_0^{-1} + \mathbf{A}^{*}\mathbf{A}]^{-1}\,\mathbf{A}^{*}\,[\mathbf{y} - \mathbf{A}\mathbf{z}_0]$

the initial state $\mathbf{x}_0$ is a random variable of mean $\bar{\mathbf{x}}_0$, and covariance matrix Π_0, and which is uncorrelated with $\mathbf{r}_i$ and $\mathbf{v}_i$, for all time instants i,

$$E\,\mathbf{x}_0 = \bar{\mathbf{x}}_0\,,\ \mathrm{cov}(\mathbf{x}_0) = \Pi_0\,,\ \mathrm{cov}(\mathbf{x}_0,\mathbf{r}_i) = \mathrm{cov}(\mathbf{x}_0,\mathbf{v}_i) = 0$$

Let $\hat{\mathbf{x}}_i$ and $\hat{\mathbf{y}}_i$ denote the linear least-mean squares estimates of x_i and y_i given the first i observations $\{\mathbf{y}_0,\mathbf{y}_1,\ldots\mathbf{y}_{i-1}\}$, respectively. We are interested in determining a recursive procedure for computing the so-called innovations variables,

$$\mathbf{e}_i = \mathbf{y}_i - \hat{\mathbf{y}}_i = \mathbf{y}_i - \mathbf{H}_i\hat{\mathbf{x}}_i$$

The reason for the term "innovations" is that $\mathbf{e}_i$ is the "new information" in the variable $\mathbf{y}_i$ that is not in the previous observations $\{\mathbf{y}_0, \ldots \mathbf{y}_{i-1}\}$. Also, it is not hard to see that the $\{\mathbf{e}_i\}$ are uncorrelated to each other and that the sets $\{\mathbf{y}_{0,\ldots},\mathbf{y}_i\}$ and $\{\mathbf{e}_{0,\ldots}\mathbf{e}_i\}$ can be linearly related to each other for all i, $i = 0,1,\ldots,N$. Since $\hat{\mathbf{y}}_i = \mathbf{H}_i\hat{\mathbf{x}}_i$, a recursive procedure for finding the $\{\mathbf{e}_i\}$ is effectively a procedure for "updating" $\hat{\mathbf{x}}_i$ to $\hat{\mathbf{x}}_{i+1}$. An efficient solution is given by the celebrated Kalman filtering algorithm [27, 28, 30]. The efficiency arises from the fact that to find $e_0, \ldots e_N$ takes $O(N^3)$ computations in general, while if the $\{\mathbf{y}_i\}$ have a state-space model with n states, and $n < N$ as is usual, then the Kalman filter requires only $O(N n^3)$ computations to find $\{e_0, \ldots e_N\}$. Moreover, if the state-space model has a special structure, the effort can be reduced to $O(N n^2)$ computations by using the so-called Chandrasekhar recursions. For the special forms arising in adaptive filtering, these flop counts reduce to $O(N^2)$ and $O(N n)$, respectively.

The interested reader may consult the Appendix where we have included, for the sake of completeness and illustration, a derivation of the (covariance) Kalman algorithm.

Algorithm: The Kalman Filter *The l.l.m.s. estimates* $\hat{\mathbf{y}}_i$ and $\hat{\mathbf{x}}_i$ *can be recursively computed as follows:start with* $\hat{\mathbf{x}}_0 = \bar{\mathbf{x}}_0$, *and repeat for* $i \geq 0$,

$$\begin{aligned}\hat{\mathbf{y}}_i &= \mathbf{H}_i\hat{\mathbf{x}}_i \\ \hat{\mathbf{x}}_{i+1} &= \mathbf{F}_i\hat{\mathbf{x}}_i + \mathbf{K}_i\mathbf{R}_{e,i}^{-1}(\mathbf{y}_i - \hat{\mathbf{y}}_i)\end{aligned} \tag{10b}$$

where the quantities $\mathbf{R}_{e,i}$ *and* $\mathbf{K}_i$ *are computed via the expressions*

$$\mathbf{R}_{e,i} = \mathbf{H}_i\,\mathbf{P}_i\,\mathbf{H}_i^* + \mathbf{R}_i\,,\ \mathbf{K}_i = \mathbf{F}_i\mathbf{P}_i\mathbf{H}_i^* \tag{10c}$$

and $\mathbf{P}_i$ *satisfies the Riccati difference recursion,*

$$\mathbf{P}_{i+1} = \mathbf{F}_i\,\mathbf{P}_i\,\mathbf{F}_i^* - \mathbf{K}_i\,\mathbf{R}_{e,i}^{-1}\,\mathbf{K}_i^* + \mathbf{G}_i\,\mathbf{Q}_i\,\mathbf{G}_i^*,\ \mathbf{P}_0 = \Pi_0 \tag{10d}$$

As explained in the Appendix, the quantities $\mathbf{R}_{e,i}$ and $\mathbf{P}_i$ can be interpreted as the covariance matrices of the innovations $\{\mathrm{e}_i\}$, and the state-estimation error, $\tilde{\mathbf{x}}_i = \mathbf{x}_i - \hat{\mathbf{x}}_i$, respectively, $\mathbf{R}_{e,i} = \mathrm{cov}(\mathbf{e}_i)$,$\mathbf{P}_i = \mathrm{cov}(\tilde{\mathbf{x}}_i)$. Also $\mathbf{K}_i = \mathrm{cov}(\mathbf{x}_{i+1},\mathbf{e}_i)$. The number of operations (i.e., multiplications and additions) that are needed in going from index i to index $(i+1)$ in the Riccati recursion (Eq. 10d) is $O(n^3)$, where n is the state dimension.

Special Case: Adaptive RLS (Fixed-Order) Problems

A special case of the state-space model (Eq. 10a) turns out to be of crucial importance in our future development. For the moment, we shall only elaborate on some of its properties and implications.

The model of interest here is one with the following special choices:

$$\mathbf{G}_i = \mathbf{Q}_i = 0\ \ \mathbf{R}_i = \mathbf{I},\ \ \mathbf{F}_i = \lambda^{-1/2}\,\mathbf{I}$$

where λ is a positive scalar less than or equal to one $(0 \ll \lambda \leq 1)$. That is,

$$\begin{aligned}\mathbf{x}_{i+1} &= \lambda^{-1/2}\mathbf{x}_i \\ \mathbf{y}_i &= \mathbf{H}_i\mathbf{x}_i + \mathbf{v}_i\end{aligned} \tag{11a}$$

with

$$E\mathbf{x}_0 = \bar{\mathbf{x}}_0,\ \mathrm{cov}(\mathbf{x}_0) = \Pi_0,\ E\mathbf{v}_i\mathbf{v}_j^* = \mathbf{I}\,\delta_{ij}$$

The associated Kalman filter equations collapse to the following:

$$\hat{\mathbf{x}}_{i+1} = \lambda^{-1/2}\left[\hat{\mathbf{x}}_i + \mathbf{P}_i\mathbf{H}_i^*\left[\mathbf{H}_i\mathbf{P}_i\mathbf{H}_i^* + \mathbf{I}\right]^{-1}(\mathbf{y}_i - \mathbf{H}_i\hat{\mathbf{x}}_i)\right],\quad \hat{\mathbf{x}}_0 = \bar{\mathbf{x}}_0$$

$$\mathbf{P}_{i+1} = \lambda^{-1}[\mathbf{P}_i - \mathbf{P}_i\mathbf{H}_i^*[\mathbf{H}_i\mathbf{P}_i\mathbf{H}_i^* + \mathbf{I}]^{-1}\mathbf{H}_i\mathbf{P}_i],\ \mathbf{P}_0 = \Pi_0 \tag{11b}$$

Assume now that we run the above Kalman recursions from i=0 to i=N. We shall then end up with the l.l.m.s.e. of $\mathbf{x}_{N+1}$ given the first N observations $\{\mathbf{y}_0,\mathbf{y}_1,\ldots,\mathbf{y}_N\}$,

$\hat{\mathbf{x}}_{N+1}$ = l.l.m.s.e. of $\mathbf{x}_{N+1}$ given $\{\mathbf{y}_0,\mathbf{y}_1,\ldots,\mathbf{y}_N\}$.

But we know from the special model (Eq. 11a) that $\mathbf{x}_{N+1}$ is simply a scaled version of the initial state $\mathbf{x}_0$ since

$$\mathbf{x}_{N+1} = [\lambda^{-1/2}]^{(N+1)}\,\mathbf{x}_0$$

If we denote, for convenience, $\mathbf{z}$=$\mathbf{x}_0$ then

$$\left[\lambda^{1/2}\right]^{(N+1)}\hat{\mathbf{x}}_{N+1}=\hat{\mathbf{z}},\ \text{the l.l.m.s.e. of } \mathbf{x}_0, \text{ given}\{\mathbf{y}_0,\ldots,\mathbf{y}_N\}\ .$$

Let us now see what is the interpretation of this observation in the deterministic setting.

For this purpose, we expand (11a) in order to emphasize the linear relation between $\mathbf{z}=\mathbf{x}_0$ and the observations $\{\mathbf{y}_i\}$,

$$\begin{bmatrix}\mathbf{y}_0\\ \mathbf{y}_1\\ \mathbf{y}_2\\ \vdots\\ \mathbf{y}_N\end{bmatrix}=\begin{bmatrix}\boldsymbol{I}_p & & & & \\ & \lambda^{-1/2}\boldsymbol{I}_p & & & \\ & & \lambda^{-1}\boldsymbol{I}_p & & \\ & & & \vdots & \\ & & & & [\lambda^{-1/2}]^N\boldsymbol{I}_p\end{bmatrix}\begin{bmatrix}\mathbf{H}_0\\ \mathbf{H}_1\\ \mathbf{H}_2\\ \vdots\\ \mathbf{H}_N\end{bmatrix}\mathbf{x}_0+\begin{bmatrix}\mathbf{v}_0\\ \mathbf{v}_1\\ \mathbf{v}_2\\ \vdots\\ \mathbf{v}_N\end{bmatrix} \quad (11c)$$

where we denote the matrix multiplying $\mathbf{x}_0$ by $\mathbf{A}$. The last expression is of the same form $\mathbf{y}=\mathbf{A}\mathbf{z}+\mathbf{v}$, with $\mathbf{z}=\mathbf{x}_0$. By referring to Table 10, we see that the corresponding deterministic least-squares problem is the following:

$$\min_{\mathbf{x}_0}\left[(\mathbf{x}_0-\bar{\mathbf{x}}_0)^*\,\Pi_0^{-1}\,(\mathbf{x}_0-\bar{\mathbf{x}}_0)+\|\mathbf{y}-\mathbf{A}\mathbf{x}_0\|_2^2\right]$$

which can be rewritten as

$$\min_{\mathbf{x}_0}\left[(\mathbf{x}_0-\bar{\mathbf{x}}_0)^*\,\Pi_0^{-1}\,(\mathbf{x}_0-\bar{\mathbf{x}}_0)+\sum_{i=0}^{N}\|\mathbf{y}_i-\mathbf{H}_i\mathbf{x}_i\|_2^2\right] \quad (11d)$$

subject to $\mathbf{x}_{i+1}=\lambda^{-1/2}\,\mathbf{x}_i$. The adaptive problem to be considered later will be shown to collapse to minimizing a cost function of this type. Hence, its solution will be immediately obtained by reformulating it as a state-space estimation problem with an underlying state-space model of the special form (Eq. 11a). We summarize this basic fact as a theorem for later reference.

Theorem: *Consider a set of* $(N+1)$ *deterministic data* $\{\mathbf{y}_i,\mathbf{x}_i\}_{i=0}^{N}$, *where the* $\mathbf{y}_i$ *are* $p\times 1$ *column vectors, the* $\mathbf{x}_i$ *are* $n\times 1$ *column vectors, and* $\mathbf{x}_{i+1}=\lambda^{-1/2}\,\mathbf{x}_i$, *for a positive real scalar,* λ. *Consider further* $\mathrm{p}\times\mathrm{n}$ *matrices* $\mathbf{H}_i$, *a positive-definite matrix* Π_0 *and an* $n\times 1$ *column vector* $\bar{\mathbf{x}}_0$. *The solution of the least-squares minimization problem*

$$\min_{\mathbf{x}_0}\left[(\mathbf{x}_0-\bar{\mathbf{x}}_0)^*\Pi_0^{-1}(\mathbf{x}_0-\bar{\mathbf{x}}_0)+\sum_{i=0}^{N}\|\mathbf{y}_i-\mathbf{H}_i\mathbf{x}_i\|_2^2\right]$$

is equal to $(\lambda^{(N+1)/2})\,\hat{\mathbf{x}}_{N+1}$, *where* $\hat{\mathbf{x}}_{N+1}$ *is recursively computed as follows: start with* $\mathbf{P}_0=\Pi_0$, $\hat{\mathbf{x}}_0=\bar{\mathbf{x}}_0$ *and repeat for* $i=1, 2, \ldots, N$:

$$\hat{\mathbf{x}}_{i+1}=\lambda^{-1}\left[\hat{\mathbf{x}}_i+\mathbf{P}_i\mathbf{H}_i^*[\mathbf{H}_i\mathbf{P}_i\mathbf{H}_i^*+\mathbf{I}]^{-1}(\mathbf{y}_i-\mathbf{H}_i\hat{\mathbf{x}}_i)\right]$$

$$\mathbf{P}_{i+1}=\lambda^{-1}\left[\mathbf{P}_i-\mathbf{P}_i\mathbf{H}_i^*[\mathbf{H}_i\mathbf{P}_i\mathbf{H}_i^*+\mathbf{I}]^{-1}\,\mathbf{H}_i\mathbf{P}_i\right]$$

The result of the theorem plays a fundamental role in our derivation. It establishes *the* link between the deterministic setting of the recursive least-squares (RLS) problem and the stochastic setting of a related Kalman filtering problem. This link will allow us to move back and forth, and very conveniently, from one interpretation to another. Also, we should highlight at this stage that, while in the statement of the theorem the $\mathbf{H}_i$ are matrix quantities and the $\mathbf{y}_i$ are vector quantities, this is really more general than what is needed to handle the RLS problem that we treat in future sections. In these sections, we shall focus, for ease of exposition, on the case where the $\mathbf{H}_i$ are replaced by row vectors, $\mathbf{h}_i$, and the $\mathbf{y}_i$ are replaced by scalars, $y(i)$. But it should be clear from the result of the theorem that our development is equally applicable to vector observations $\mathbf{y}_i$ and to matrices $\mathbf{H}_i$.

Special Case: Adaptive Lattice (Order-Recursive) Problems

Another special case of the state-space model (Eq. 10a) will play a significant role in the derivation of the so-called adaptive lattice filters. It is identical to the model in (Eq. 11a) but with *scalar* quantities. That is, the state vector $\mathbf{x}_i$ is replaced by a scalar (one-dimensional) state $x(i)$, the output vector $\mathbf{y}_i$ is replaced by a scalar observation $y(i)$, the noise vector $\mathbf{v}_i$ is replaced by a scalar signal $v(i)$, and the matrix $\mathbf{H}_i$ is replaced by a scalar $h(i)$ (note that we use parentheses, $(\cdot)$, to indicate time dependency for scalar quantities. In the vector case we have been using subscripts),

$$\begin{aligned}x(i+1)&=\lambda^{-1/2}x(i)\\ y(i)&=h(i)x(i)+v(i)\end{aligned} \quad (12a)$$

with

$$Ex(0)=\bar{x}(0),\ \mathrm{cov}(\mathrm{x}(0))=\pi(0),\ Ev(i)v^*(j)=\delta_{ij}$$

This is clearly a special case of the earlier model (Eq. 11a), and the result of the previous theorem is thus immediately applicable.

Theorem: *Consider a set of* $(N+1)$ *deterministic scalar data* $\{y(i),x(i)\}_{i=0}^{N}$, *where* $x(i+1)=\lambda^{-1/2}x(i)$, *for a positive real scalar* λ. *Consider further scalars* $h(i)$, *a positive number* $\pi(0)$, *and a scalar* $\bar{x}(0)$. *The solution of the least-squares minimization problem*

$$\min_{x(0)}\left[(x(0)-\bar{x}(0))^*\pi^{-1}(0)(x(0)-\bar{x}(0))+\sum_{i=0}^{N}|y(i)-h(i)x(i)|^2\right] \quad (12b)$$

is equal to $\left[\lambda^{(N+1)/2}\right]\hat{x}(N+1)$, *where* $\hat{x}(N+1)$ *is recursively computed as follows: start with* $p(0)=\pi(0)$, $\hat{x}(0)=\bar{x}(0)$ *and repeat for* $i = 1, 2, \ldots, N$:

$$\hat{x}(i+1) = \frac{\lambda^{-1/2}}{1 + p(i)\,|h(i)|^2}\,[\hat{x}(i) + p(i)\,h^*(i)\,y(i)]$$

$$p(i+1) = \frac{\lambda^{-1}p(i)}{1 + p(i)\,|h(i)|^2}$$

To get a geometric interpretation for the above result, we note that expression (Eq. 11c) collapses to the following:

$$\begin{bmatrix} y(0) \\ y(1) \\ y(2) \\ \vdots \\ y(N) \end{bmatrix} = \begin{bmatrix} h(0) \\ \lambda^{-1/2}\,h(1) \\ \lambda^{-1}\,h(2) \\ \vdots \\ (\lambda^{-1/2})^N\,h(N) \end{bmatrix} x(0) + \begin{bmatrix} v(0) \\ v(1) \\ v(2) \\ \vdots \\ v(N) \end{bmatrix} \qquad (12c)$$

and the problem then is to determine a scalar coefficient in order to "match" the column vector $\mathbf{y}$ on the left-hand side of (Eq. 12c) with the column vector $\boldsymbol{a}$, which multiplies $x(0)$, in the least-squares sense specified by Eq. 12b. In the special case $\pi(0)=\infty$, this is equivalent to projecting the vector $\mathbf{y}$ onto the vector $\boldsymbol{a}$ leading to (cf.(8c)):

$$\hat{z} = \frac{1}{\boldsymbol{a}^*\boldsymbol{a}}\,\boldsymbol{a}^*\mathbf{y} = \text{the l.l.m.s.e. of } x(0), \text{ given } [y(0), \ldots y(N)]$$

Algorithmic Variants of the Kalman Filter

There are several variants to the Kalman filter recursions (Eqs.10b)-(10d). These essentially differ in the ways they propagate the quantities $\mathbf{K}_i$, $\mathbf{R}_{e,i}$ and $\mathbf{P}_i$ that are needed in the Kalman recursions. We shall not discuss these algorithmic variants in details here, but shall rather focus on the particular alternatives that are relevant to our subsequent discussion. For more details, the reader may consult [27, 30] and [43]-[45].

We shall concentrate on the special model (Eq. 11a) rather than consider the general state-space description given in (Eq. 10a). But we hasten to add that the discussion that follows can be easily extended to models of the form (Eq. 10a). This constitutes a significant strength of the state-space formulation: it allows us, for instance, to consider more general matrices $\mathbf{F}_i$ in (Eq. 11a) e.g., F_i = diagonal $\{\lambda_1^{-1/2}, \lambda_2^{-1/2}, \ldots, \lambda_n^{-1/2}\}$. It also allows for more general matrices $\mathbf{G}_i$, $\mathbf{Q}_i$ and $\mathbf{R}_i$. The careful reader will soon realize that these more general cases can be handled by proper extensions of the arguments in this paper. But we shall concentrate here on the special model (Eq. 11a).

The results described in the rest of this section are summarized in Table 11. Readers familiar with the Kalman filter may wish to examine it briefly and go on to the next section on adaptive filtering. Other readers may also do this, returning for some details to the appropriate portions of this section as they are cited in the RLS problems later.

The (Covariance) Kalman Filter

The model in (Eq. 11a) assumes $\mathbf{G}_i=\mathbf{Q}_i=0$, $\mathbf{R}_i=\mathbf{I}$ and $\mathbf{F}_i = \lambda^{-1/2}\,\mathbf{I}$. The associated Kalman filter equations are summarized below for ease of reference.

Algorithm: *Given the state-space model (11a), the l.l.m.s. estimate* $\hat{\mathbf{x}}_i$ *can be recursively computed via the recursions: start with* $\hat{\mathbf{x}}_0 = \bar{\mathbf{x}}_0$ *and* $\mathbf{P}_0 = \Pi_0$, *and repeat for* $i \geq 0$,

$$\hat{\mathbf{x}}_{i+1} = \lambda^{-1/2}\hat{\mathbf{x}}_i + \mathbf{K}_i\,\mathbf{R}_{e,i}^{-1}\,(\mathbf{y}_i - \mathbf{H}_i\,\hat{\mathbf{x}}_i)$$

$$\mathbf{R}_{e,i} = \mathbf{H}_i\mathbf{P}_i\mathbf{H}_i^* + \mathbf{I}, \quad \mathbf{K}_i = \lambda^{-1/2}\mathbf{P}_i\mathbf{H}_i^*$$

$$\mathbf{P}_{i+1} = \lambda^{-1}\mathbf{P}_i - \mathbf{K}_i\mathbf{R}_{e,i}^{-1}\mathbf{K}_i^* \qquad (13)$$

Assuming that $p << n$, as often happens, the number of operations (*i.e.*, multiplications and additions) that are needed in going from index i to index $(i+1)$ is $O(pn^2)$, where n is the state dimension and p is the output dimension. This is smaller that the $O(n^3)$ figure that we mentioned earlier because we are now assuming a very special matrix $\mathbf{F}_i$, viz., a multiple of the identity.

The Information Filter

The recursions (Eq. 13) propagate the Riccati variable $\mathbf{P}_i$. In several applications, however, the uncertainty in the initial state $\mathbf{x}_0$ may be high. That is, $\Pi_0 = \sigma\,\mathbf{I}$ with $\sigma >> 1$, which implies that the starting point for the Riccati recursion involves large numbers. This is particularly the case in a standard least-squares minimization problem, such as Eq. 11d, where Π_0 is assumed to be infinite and where the problem of interest is to solve for $\mathbf{x}_0$ in

$$\min_{\mathbf{x}_0} \|\mathbf{y} - \mathbf{A}\mathbf{x}_0\|_2^2 .$$

For such problems, it is preferable to propagate the inverse of the Riccati variable, $\mathbf{P}_i^{-1}$ rather than $\mathbf{P}_i$ itself. The resulting algorithm is known as the information filter (see, e.g., [27, 43]). In contrast, the original Kalman filter recursions are often called the covariance form. It can be derived from the recursions of the previous algorithm rather immediately by invoking the matrix inversion lemma (5).

Algorithm: The Information Filter *Given the state-space model (Eq. 11a), the l.l.m.s. estimate* $\hat{\mathbf{x}}_i$ *can also be recursively computed via the recursions: start with* $\mathbf{P}_0^{-1}\hat{\mathbf{x}}_0 = \Pi_0^{-1}\bar{\mathbf{x}}_0$ *and* $\mathbf{P}_0^{-1} = \Pi_0^{-1}$, *and repeat for* $i \geq 0$

$$\mathbf{P}_{i+1}^{-1} = \lambda\,[\mathbf{P}_i^{-1} + \mathbf{H}_i^*\mathbf{H}_i]$$

Table 11: Algorithmic variants of the Kalman filter.

The assumed model is:

$\mathbf{G}_i = \mathbf{Q}_i = 0,\ \mathbf{R}_i = \mathbf{I},\ \mathbf{F}_i = \lambda^{-1/2}\mathbf{I},$

$\mathbf{x}_{i+1} = \lambda^{-1/2}\mathbf{x}_i,\ \mathbf{y}_i = \mathbf{H}_i\mathbf{x}_i + \mathbf{v}_i,$

$E\mathbf{x}_0 + \bar{\mathbf{x}}_0,\ \mathrm{cov}(\mathbf{x}_0) = \Pi_0,\ E\mathbf{v}_i\mathbf{v}_j^* = \mathbf{I}\delta_{ij},\ \hat{\mathbf{x}}_0 = \bar{\mathbf{x}}_0,\ \mathbf{P}_0 = \Pi_0.$

Filter Name	Expressions
The Covariance Kalman Filter	$\hat{\mathbf{x}}_{i+1} = \lambda^{-1/2}\hat{\mathbf{x}}_i + \mathbf{K}_i\,\mathbf{R}_{e,i}^{-1}(\mathbf{y}_i - \mathbf{H}_i\hat{\mathbf{x}}_i)$ $\mathbf{R}_{e,i} = \mathbf{H}_i\mathbf{P}_i\mathbf{H}_i^* + \mathbf{I},\ \mathbf{K}_i = \lambda^{-1/2}\mathbf{P}_i\mathbf{H}_i^*$ $\mathbf{P}_{i+1} = \lambda^{-1}\mathbf{P}_i - \mathbf{K}_i\mathbf{R}_{e,i}^{-1}\mathbf{K}_i^*$
The Information Filter	$\mathbf{P}_{i+1}^{-1} = \lambda[\mathbf{P}_i^{-1} + \mathbf{H}_i^*\mathbf{H}_i]$ $\mathbf{P}_{i+1}^{-1}\hat{\mathbf{x}}_{i+1} = \lambda^{1/2}\,[\mathbf{P}_i^{-1}\hat{\mathbf{x}}_i + \mathbf{H}_i^*\mathbf{y}_i]$ $\mathbf{R}_{e,i}^{-1} = \mathbf{I} - \lambda\mathbf{H}_i\mathbf{P}_{i+1}\mathbf{H}_i^*$ $\mathbf{R}_{e,i}^{-1}\mathbf{e}_i = \mathbf{y}_i - \lambda^{1/2}\mathbf{H}_i\hat{\mathbf{x}}_{i+1}$ $\hat{\mathbf{x}}_{i+1} = [\mathbf{P}_{i+1}^{-1}]^{-1}\,[\mathbf{P}_{i+1}^{-1}\hat{\mathbf{x}}_{i+1}]$
The Square-Root Covariance Filter	$\begin{bmatrix}\mathbf{I} & \mathbf{H}_i\mathbf{P}_i^{1/2}\\ 0 & \lambda^{-1/2}\mathbf{P}_i^{1/2}\end{bmatrix}\Theta_i = \begin{bmatrix}\mathbf{R}_{e,i}^{1/2} & 0\\ \bar{\mathbf{K}}_{p,i} & \mathbf{P}_{i+1}^{1/2}\end{bmatrix}$ $\hat{\mathbf{x}}_{i+1} = \lambda^{-1/2}\hat{\mathbf{x}}_i + \bar{\mathbf{K}}_{p,i}[\mathbf{R}_{e,i}^{1/2}]^{-1}\,(\mathbf{y}_i - \mathbf{H}_i\hat{\mathbf{x}}_i)$
The Extended Square-Root Information Filter	$\begin{bmatrix}\lambda^{1/2}\mathbf{P}_i^{-*/2} & \lambda^{1/2}\mathbf{H}_i^*\\ \hat{\mathbf{x}}_i^*\mathbf{P}_i^{-*/2} & \mathbf{y}_i^*\\ 0 & \mathbf{I}\\ \lambda^{-1/2}\mathbf{P}_i^{1/2} & 0\end{bmatrix}\Theta_i = \begin{bmatrix}\mathbf{P}_{i+1}^{-*/2} & 0\\ \hat{\mathbf{x}}_{i+1}^*\mathbf{P}_{i+1}^{-*/2} & \mathbf{e}_i^*\mathbf{R}_{e,i}^{-*/2}\\ \lambda^{1/2}\mathbf{H}_i\mathbf{P}_{i+1}^{1/2} & \mathbf{R}_{e,i}^{-*/2}\\ \mathbf{P}_{i+1}^{1/2} & -\bar{\mathbf{K}}_{p,i}\end{bmatrix}$ $\hat{\mathbf{x}}_{i+1} = \lambda^{-1/2}\hat{\mathbf{x}}_i + \bar{\mathbf{K}}_{p,i}\,[\mathbf{R}_{e,i}^{-1/2}\mathbf{e}_i] = [\mathbf{P}_{i+1}^{1/2}]\,[\mathbf{P}_{i+1}^{-1/2}\hat{\mathbf{x}}_{i+1}]\,.$
The Square-Root Chandrasekhar Filter	$\mathbf{H}_i = \mathbf{H}_{i+1}\Psi$ $\lambda^{-1}\Pi_0 - \mathbf{K}_0\mathbf{R}_{e,0}^{-1}\mathbf{K}_0^* - \Psi\Pi_0\Psi^* = \mathbf{L}_0\mathbf{S}_0\mathbf{L}_0^*$ $\begin{bmatrix}\mathbf{R}_{e,i}^{1/2} & \mathbf{H}_{i+1}\mathbf{L}_i\\ \Psi\bar{\mathbf{K}}_{p,i} & \lambda^{-1/2}\mathbf{L}_i\end{bmatrix}\Theta_i = \begin{bmatrix}\mathbf{R}_{e,i+1}^{1/2} & 0\\ \bar{\mathbf{K}}_{p,i+1} & \mathbf{L}_{i+1}\end{bmatrix}$ $\hat{\mathbf{x}}_{i+1} = \lambda^{-1/2}\hat{\mathbf{x}}_i + \bar{\mathbf{K}}_{p,i}[\mathbf{R}_{e,i}^{1/2}]^{-1}\,(\mathbf{y}_i - \mathbf{H}_i\hat{\mathbf{x}}_i)$
The Explicit Chandrasekhar Filter	$\mathbf{H}_i = \mathbf{H}_{i+1}\Psi$ $\lambda^{-1}\Pi_0 - \mathbf{K}_0\mathbf{R}_{e,i}^{-1}\mathbf{K}_0^* - \Psi\,\Pi_0\Psi^* = -\mathbf{L}_0^{(u)}\mathbf{R}_{r,0}^{-1}\mathbf{L}_0^{*(u)}$ $\begin{bmatrix}\mathbf{R}_{e,i} & \mathbf{H}_{i+1}\mathbf{L}_i^{(u)}\\ \Psi\mathbf{K}_i & \lambda^{-1/2}\mathbf{L}_i^{(u)}\\ \mathbf{L}_i^{*(u)}\mathbf{H}_{i+1}^* & \mathbf{R}_{r,i}\end{bmatrix}\Sigma_i = \begin{bmatrix}\mathbf{R}_{e,i+1} & 0\\ \mathbf{K}_{i+1} & \mathbf{L}_{i+1}^{(u)}\\ 0 & \mathbf{R}_{r,i+1}\end{bmatrix}$ $\Sigma_i = \begin{bmatrix}\mathbf{I} & -\mathbf{R}_{e,i}^{-1}\mathbf{H}_{i+1}\mathbf{L}_i^{(u)}\\ -\mathbf{R}_{r,i}^{-1}\mathbf{L}_i^{*(u)}\mathbf{H}_{i+1}^* & \mathbf{I}\end{bmatrix}$ $\hat{\mathbf{x}}_{i+1} = \lambda^{-1/2}\hat{\mathbf{x}}_i + \mathbf{K}_i[\mathbf{R}_{e,i}]^{-1}\,(\mathbf{y}_i - \mathbf{H}_i\hat{\mathbf{x}}_i)$

$$\mathbf{P}_{i+1}^{-1}\hat{\mathbf{x}}_{i+1} = \lambda^{1/2}\,[\mathbf{P}_i^{-1}\hat{\mathbf{x}}_i + \mathbf{H}_i^*\mathbf{y}_i]$$

Moreover, the inverse of innovations covariance matrix, $\mathbf{R}_{e,i}$, *and the normalized innovation* $\mathbf{R}_{e,i}^{-1}\mathbf{e}_i$ *are given by*

$$\mathbf{R}_{e,i}^{-1} = \mathbf{I} - \lambda\mathbf{H}_i\mathbf{P}_{i+1}\mathbf{H}_i^*,\ \mathbf{R}_{e,i}^{-1}\mathbf{e}_i = \mathbf{y}_i - \lambda^{1/2}\mathbf{H}_i\hat{\mathbf{x}}_{i+1}$$

The state-estimate at any particular time-instant can be recovered from

$$\left[\mathbf{P}_{i+1}^{-1}\right]^{-1}\left[\mathbf{P}_{i+1}^{-1}\hat{\mathbf{x}}_{i+1}\right] = \hat{\mathbf{x}}_{i+1}\,. \qquad (14)$$

Note that the information filter propagates the quantity $\mathbf{P}_i^{-1}\hat{\mathbf{x}}_i$ rather than $\hat{\mathbf{x}}_i$ itself. Also, the number of operations (*i.e.*, multiplications and additions) that are needed in going from index i to index $(i+1)$ is still $O(pn^2)$, where n is the state dimension and p is the output dimension.

The Square-Root Kalman Filter

The Kalman recursions (Eq. 13) propagate the covariance matrix $\mathbf{P}_i$ via a Riccati difference equation. Due to roundoff errors, however, the Riccati recursion may not guarantee a positive-definite matrix $\mathbf{P}_i$ at all times i. This problem can be avoided by using an alternative so-called square-root array form of the Kalman filter equations. The array form propagates a square-root factor of $\mathbf{P}_i$, viz., $\mathbf{P}_i^{1/2}$. By squaring $\mathbf{P}_i^{1/2}$ we shall always be guaranteed to obtain a positive-definite covariance matrix, $\mathbf{P}_i = \mathbf{P}_i^{1/2}\mathbf{P}_i^{*/2}$.

Before proceeding to the derivation of the square-root filter, we first state and prove a simple result from matrix theory that plays an important role in the argument.

Lemma: *Given two* $n \times m$ $(n \le m)$ *matrices* $\mathbf{A}$ *and* $\mathbf{B}$. *Then* $\mathbf{AA}^* = \mathbf{BB}^*$ *if, and only if, there exists an* $m \times m$ *unitary matrix* Θ $(\Theta\Theta^* = \mathbf{I}_m)$ *such that* $\mathbf{A} = \mathbf{B}\Theta$.

Proof: One implication is immediate. If there exists a unitary matrix Θ such that $\mathbf{A} = \mathbf{B}\Theta$ then $\mathbf{AA}^* = (\mathbf{B}\Theta)(\mathbf{B}\Theta)^* = \mathbf{B}(\Theta\Theta^*)\mathbf{B}^* = \mathbf{BB}^*$. One proof for the converse implication follows by invoking the singular value decompositions of $\mathbf{A}$ and $\mathbf{B}$,

$$\mathbf{A} = \mathbf{U}_\mathbf{A}\,[\,\Sigma_\mathbf{A}\ 0\,]\,\mathbf{V}_\mathbf{A}^*,$$
$$\mathbf{B} = \mathbf{U}_\mathbf{B}\,[\,\Sigma_\mathbf{B}\ 0\,]\,\mathbf{V}_\mathbf{B}^*$$

where $\mathbf{U}_\mathbf{A}$ and $\mathbf{U}_\mathbf{B}$ are $n \times n$ unitary matrices, $\mathbf{V}_\mathbf{A}$ and $\mathbf{V}_\mathbf{B}$ are $m \times m$ unitary matrices, and $\Sigma_\mathbf{A}$ and $\Sigma_\mathbf{B}$ are $n \times n$ diagonal matrices with nonnegative (ordered) entries. The squares of the diagonal entries of $\Sigma_\mathbf{A}$ ($\Sigma_\mathbf{B}$) are the eigenvalues of $\mathbf{AA}^*$

(BB*). Moreover, $\mathbf{U}_A$,$\mathbf{U}_B$ can be constructed from an orthonormal basis for the right eigenvectors of **AA* (BB*)**. Hence, it follows from the identity $\mathbf{AA}^* = \mathbf{BB}^*$ that we have $\Sigma_A = \Sigma_B$ and $\mathbf{U_A} = \mathbf{U_B}$. Let $\Theta = \mathbf{V}_B \mathbf{V}_A{}^*$. We then get $\Theta\Theta^* = \mathbf{I}_m$ and $\mathbf{B}\,\Theta = \mathbf{A}$.

We now use the above result to motivate the square-root form of the Kalman filter. For this purpose, we note that the Kalman recursions (Eq. 13) can be expressed in factored form as

$$\begin{bmatrix} \mathbf{I} & \mathbf{H}_i\mathbf{P}_i^{1/2} \\ 0 & \lambda^{-1/2}\,\mathbf{P}_i^{1/2} \end{bmatrix} \begin{bmatrix} \mathbf{I} & 0 \\ \mathbf{P}_i^{*/2}\mathbf{H}_i^* & \lambda^{-1/2}\mathbf{P}_i^{*/2} \end{bmatrix}$$

$$= \begin{bmatrix} \mathbf{R}_{\mathbf{e},i}^{1/2} & 0 \\ \mathbf{K}_i\mathbf{R}_{\mathbf{e},i}^{-*/2} & \mathbf{P}_{i+1}^{1/2} \end{bmatrix} \begin{bmatrix} \mathbf{R}_{\mathbf{e},i}^{*/2}\mathbf{R}_{\mathbf{e},i}^{-1/2}\mathbf{K}_i^* \\ 0 \quad \mathbf{P}_{i+1}^{*/2} \end{bmatrix}$$

The above equality fits into the statement of the lemma . We thus conclude that there exists a unitary matrix Θ_i that relates the arrays (an interesting geometrical motivation for the existence of such rotations can be found in [43])

$$\begin{bmatrix} \mathbf{I} & \mathbf{H}_i\mathbf{P}_i^{1/2} \\ 0 & \lambda^{-1/2}\,\mathbf{P}_i^{1/2} \end{bmatrix} \Theta_i = \begin{bmatrix} \mathbf{R}_{e,i}^{1/2} & 0 \\ \mathbf{K}_i\mathbf{R}_{e,i}^{-*/2} & \mathbf{P}_{i+1}^{1/2} \end{bmatrix} \tag{15a}$$

In fact any unitary matrix Θ_i that takes the prearray of numbers

$$\begin{bmatrix} \mathbf{I} & \mathbf{H}_i\mathbf{P}_i^{1/2} \\ 0 & \lambda^{-1/2}\,\mathbf{P}_i^{1/2} \end{bmatrix}$$

and triangularizes it; thus leading to a postarray of numbers of the form

$$\begin{bmatrix} \mathbf{I} & \mathbf{H}_i\mathbf{P}_i^{1/2} \\ 0 & \lambda^{-1/2}\,\mathbf{P}_i^{1/2} \end{bmatrix} \Theta_i = \begin{bmatrix} \mathbf{X} & 0 \\ \mathbf{Y} & \mathbf{Z} \end{bmatrix} \tag{15b}$$

for some $\{\mathbf{X},\mathbf{Y},\mathbf{Z},\}$ achieves the transformation (Eq. 15a). To verify that the quantities $\{\mathbf{X},\mathbf{Y},\mathbf{Z}\}$ can indeed be chosen as $\{\mathbf{R}_{\mathbf{e},i}^{1/2}, \mathbf{K}_i\mathbf{R}_{\mathbf{e},i}^{-*/2}, \mathbf{P}_{i+1}^{1/2}\}$, we proceed by squaring both sides of (Eq. 15b),

$$\begin{bmatrix} \mathbf{I} & \mathbf{H}_i\mathbf{P}_i^{1/2} \\ 0 & \lambda^{-1/2}\,\mathbf{P}_i^{1/2} \end{bmatrix} \Theta_i\,\Theta_i^* \begin{bmatrix} \mathbf{I} & \mathbf{H}_i\mathbf{P}_i^{1/2} \\ 0 & \lambda^{-1/2}\,\mathbf{P}_i^{1/2} \end{bmatrix}^* = \begin{bmatrix} \mathbf{X} & 0 \\ \mathbf{Y} & \mathbf{Z} \end{bmatrix} \begin{bmatrix} \mathbf{X} & 0 \\ \mathbf{Y} & \mathbf{Z} \end{bmatrix}^*$$

and comparing terms on both sides of the equality to get the identities:

$$\mathbf{XX}^* = \mathbf{I} + \mathbf{H}_i\mathbf{P}_i\mathbf{H}_i^* = \mathbf{R}_{\mathbf{e},i}$$

$$\mathbf{YX}^* = \lambda^{-1/2}\mathbf{P}_i\mathbf{H}_i^* = \mathbf{K}_i$$

$$\mathbf{ZZ}^* = \lambda^{-1}\mathbf{P}_i - \mathbf{K}_i\mathbf{R}_{\mathbf{e},i}^{-1}\mathbf{K}_i = \mathbf{P}_{i+1}$$

Hence, we can choose $\mathbf{X} = \mathbf{R}_{\mathbf{e},i}^{1/2}$, $\mathbf{Y} = \mathbf{K}_i\mathbf{R}_{\mathbf{e},i}^{-*/2} \equiv \bar{\mathbf{K}}_{p,i}$, and $\mathbf{Z} = \mathbf{P}_{i+1}^{1/2}$ .We are thus led to the square-root version of the Kalman recursion [27, 43].

Algorithm: The Square-Root Kalman Filter *Given the state-space model (Eq. 11a), the l.l.m.s. estimate* $\hat{\mathbf{x}}_i$ *can also be recursively computed via the recursions: start with* $\hat{\mathbf{x}}_0 = \bar{\mathbf{x}}_0$ *and* $\mathbf{P}_0^{1/2} = \Pi_0^{1/2}$ *, and repeat for* $i \geq 0$*,*

$$\begin{bmatrix} I & \mathbf{H}_i\mathbf{P}_i^{1/2} \\ 0 & \lambda^{-1/2}\,\mathbf{P}_i^{1/2} \end{bmatrix} \Theta_i = \begin{bmatrix} \mathbf{R}_{\mathbf{e},i}^{1/2} & 0 \\ \bar{\mathbf{K}}_{p,i} & \mathbf{P}_{i+1}^{1/2} \end{bmatrix} \tag{15c}$$

where Θ_i *is any unitary transformation that produces the block zero entry in the postarray, and*

$$\hat{\mathbf{x}}_{i+1} = \lambda^{-1/2}\hat{\mathbf{x}}_i + \bar{\mathbf{K}}_{p,i}\left[\mathbf{R}_{\mathbf{e},i}^{1/2}\right]^{-1}(\mathbf{y}_i - \mathbf{H}_i\hat{\mathbf{x}}_i)\,.$$

The number of operations needed in going from step i to step (i+1) is still $O(n^2)$, the same order as the Riccati-based algorithm.

An Extended Square-Root Information Filter

The information filter equations of the previous algorithm can also be expressed in square-root form, where a square-root factor of $\mathbf{P}_i^{-1}$ is propagated rather than $\mathbf{P}_i$-1 itself. Since the argument is essentially similar to what we have done in the previous section, we only highlight the major steps here. We form the prearray of numbers

$$\begin{bmatrix} \lambda^{1/2}\mathbf{P}_i^{-*/2} & \lambda^{1/2}\mathbf{H}_i^* \\ \hat{\mathbf{x}}_i^*\mathbf{P}_i^{-*/2} & \mathbf{y}_i^* \\ \mathbf{0} & \mathbf{I} \\ \lambda^{-1/2}\mathbf{P}_i^{-*/2} & \mathbf{0} \end{bmatrix}$$

and choose any unitary matrix Θ_i that introduces a block zero in the second block entry of the top row, say

$$\begin{bmatrix} \lambda^{1/2}\mathbf{P}_i^{-*/2} & \lambda^{1/2}\mathbf{H}_i^* \\ \hat{\mathbf{x}}_i^*\mathbf{P}_i^{-*/2} & \mathbf{y}_i^* \\ \mathbf{0} & \mathbf{I} \\ \lambda^{-1/2}\mathbf{P}_i^{-*/2} & \mathbf{0} \end{bmatrix} \Theta_i = \begin{bmatrix} \mathbf{X} & \mathbf{0} \\ \mathbf{Y} & \mathbf{Z} \\ \mathbf{W} & \mathbf{T} \\ \mathbf{S} & \mathbf{V} \end{bmatrix}.$$

By squaring and comparing terms on both sides of the equality

$$\begin{bmatrix} \lambda^{1/2}\mathbf{P}_i^{-*/2} & \lambda^{1/2}\mathbf{H}_i^* \\ \hat{\mathbf{x}}_i^*\mathbf{P}_i^{-*/2} & \mathbf{y}_i^* \\ \mathbf{0} & \mathbf{I} \\ \lambda^{-1/2}\mathbf{P}_i^{-*/2} & \mathbf{0} \end{bmatrix} \Theta_i \Theta_i^* \begin{bmatrix} \lambda^{1/2}\mathbf{P}_i^{-*/2} & \lambda^{1/2}\mathbf{H}_i^* \\ \hat{\mathbf{x}}_i^*\mathbf{P}_i^{-*/2} & \mathbf{y}_i^* \\ \mathbf{0} & \mathbf{I} \\ \lambda^{-1/2}\mathbf{P}_i^{-*/2} & \mathbf{0} \end{bmatrix}^*$$

$$= \begin{bmatrix} \mathbf{X} & \mathbf{0} \\ \mathbf{Y} & \mathbf{Z} \\ \mathbf{W} & \mathbf{T} \\ \mathbf{S} & \mathbf{V} \end{bmatrix} \begin{bmatrix} \mathbf{X} & \mathbf{0} \\ \mathbf{Y} & \mathbf{Z} \\ \mathbf{W} & \mathbf{T} \\ \mathbf{S} & \mathbf{V} \end{bmatrix}^*$$

we readily conclude that we can make the following identifications:

$$\mathbf{X} = \mathbf{P}_{i+1}^{-*/2},\ \mathbf{Y} = \hat{\mathbf{x}}_{i+1}^* \mathbf{P}_{i+1}^{-*/2},\ \mathbf{W} = \lambda^{1/2}\mathbf{H}_i\mathbf{P}_{i+1}^{1/2},$$
$$\mathbf{T} = \mathbf{R}_{\mathbf{e},i}^{-*/2},\ \mathbf{Z} = \mathbf{e}_i^*\,\mathbf{R}_{\mathbf{e},i}^{-*/2},\ \mathbf{S} = \mathbf{P}_{i+1}^{1/2},\ \mathbf{V} = -\bar{\mathbf{K}}_{p,i}.$$

Algorithm: The Extended Square-Root Information Filter *Given the state space model (Eq. 11a), the l.l.m.s. estimate* $\hat{\mathbf{x}}_i$ *can also be recursively computed via the recursions: start with* $\hat{\mathbf{x}}_0 = \bar{\mathbf{x}}_0$, $\mathbf{P}_0^{1/2} = \Pi_0^{1/2}$, $\mathbf{P}_0^{-1/2} = \Pi_0^{-1/2}$, *and repeat for* $i \geq 0$,

$$\begin{bmatrix} \lambda^{1/2}\mathbf{P}_i^{-*/2} & \lambda^{1/2}\mathbf{H}_i^* \\ \hat{\mathbf{x}}_i^*\mathbf{P}_i^{-*/2} & \mathbf{y}_i^* \\ \mathbf{0} & \mathbf{I} \\ \lambda^{-1/2}\mathbf{P}_i^{-*/2} & \mathbf{0} \end{bmatrix} \Theta = \begin{bmatrix} \mathbf{P}_{i+1}^{-*/2} & \mathbf{0} \\ \hat{\mathbf{x}}_{i+1}\mathbf{P}_{i+1}^{-*/2} & \mathbf{e}_i^*\mathbf{R}_{\mathbf{e},i}^{-*/2} \\ \lambda^{1/2}\mathbf{H}_i\mathbf{P}_{i+1}^{1/2} & \mathbf{R}_{\mathbf{e},i}^{-*/2} \\ \mathbf{P}_{i+1}^{1/2} & -\bar{\mathbf{K}}_{p,i} \end{bmatrix} \tag{16a}$$

where Θ_i *is any unitary matrix that produces the block zero entry in the top block row of the postarray. The state estimate is given by*

$$\hat{\mathbf{x}}_{i+1} = \lambda^{-1/2}\hat{\mathbf{x}}_i + \bar{\mathbf{K}}_{p,i}\left[\mathbf{R}_{\mathbf{e},i}^{-1/2}\mathbf{e}_i\right] = \left[\mathbf{P}_{i+1}^{1/2}\right]\left[\mathbf{P}_{i+1}^{-1/2}\hat{\mathbf{x}}_{i+1}\right]$$

where the quantities $\{\mathbf{R}_{\mathbf{e},i}^{-1/2}\mathbf{e}_i, \mathbf{P}_{i+1}^{-1/2}\hat{\mathbf{x}}_{i+1}\}$ *and* $\{\bar{\mathbf{K}}_{p,i}, \mathbf{P}_{i+1}^{1/2}\}$ *are read from the entries of the second and last lines of the postarray, respectively.*

The number of operations needed in going from step i to step $(i+1)$ is still $O(n^2)$. The term "extended" is used here to indicate that the above array is a simple extension of the usual form of the square-root information filter as given in [43]: the top three block lines are standard in the square-root description of the information filter, while the last block line is borrowed from the square-root Kalman filter (Eq. 15c). An alternative interpretation of (Eq. 16a) as one that avoids the backsubstitution required to obtain $\hat{\mathbf{x}}_i$ from $\mathbf{P}_i^{-1/2}\hat{\mathbf{x}}_i$ (viz., to use the so-called Faddeeva's method) is described in [50].

An extension may also be noted here. If the covariance matrix of the noise signal $\mathbf{v}_i$ in the model (Eq. 11a) is not the identity matrix but rather $E\mathbf{v}_i\mathbf{v}_j^* = \mathbf{R}_i$, then the previous square-root arguments are still applicable and they immediately show that the array equation (Eq. 16a) becomes

$$\begin{bmatrix} \lambda^{1/2}\mathbf{P}_i^{-*/2} & \lambda^{1/2}\mathbf{H}_i^*\,\mathbf{R}_i^{-*/2} \\ \hat{\mathbf{x}}_i^*\mathbf{P}_i^{-*/2} & \mathbf{y}_i^*\,\mathbf{R}_i^{-*/2} \\ \mathbf{0} & \mathbf{I}^*\mathbf{R}_i^{-*/2} \\ \lambda^{-1/2}\mathbf{P}_i^{-*/2} & \mathbf{0} \end{bmatrix} \Theta = \begin{bmatrix} \mathbf{P}_{i+1}^{-*/2} & \mathbf{0} \\ \hat{\mathbf{x}}_{i+1}\mathbf{P}_{i+1}^{-*/2} & \mathbf{e}_i^*\mathbf{R}_{\mathbf{e},i}^{-*/2} \\ \lambda^{1/2}\mathbf{R}_i^{-*/2}\mathbf{H}_i\mathbf{P}_{i+1}^{1/2} & \mathbf{R}_{\mathbf{e},i}^{-*/2} \\ \mathbf{P}_{i+1}^{1/2} & -\bar{\mathbf{K}}_{p,i} \end{bmatrix} \tag{16b}$$

Hence, the equality $\mathbf{R}_{\mathbf{e},i}^{-1}\,\mathbf{e}_i = \mathbf{y}_i - \lambda^{1/2}\,\mathbf{H}_i\,\hat{\mathbf{x}}_{i+1}$ in the third prior algorithm converts to

$$\mathbf{R}_{\mathbf{e},i}^{-1}\,\mathbf{e}_i = \mathbf{R}_i^{-1}\left[\mathbf{y}_i - \lambda^{1/2}\mathbf{H}_i\,\hat{\mathbf{x}}_{i+1}\right]$$

This modification is useful when a weighted least-squares criterion is used., as happens in a later section.

The Square-Root Chandrasekhar Filter

All the algorithmic variants presented so far have an $O(n^2)$ computational complexity, and this is true whether or not the state-space model (Eq. 11a) or, more generally, the model in (Eq. 10a) have constant parameters $\{\mathbf{F}_i, \mathbf{G}_i, \mathbf{H}_i, \mathbf{Q}_i, \mathbf{R}_i\}$. However, one expects a computationally more efficient procedure in the case of constant-parameter systems $\{\mathbf{F},\mathbf{G},\mathbf{H},\mathbf{Q},\mathbf{R}\}$ (see, e.g., [44]–[45]) or in the case of time-variant models that exhibit structure in their time-variation (see, e.g., [31]).

The Kalman recursions require the Riccati variable $\mathbf{P}_i$ in order to compute the covariance matrix $\mathbf{R}_{\mathbf{e},i}$ and the gain matrix $\mathbf{K}_i$, which are in turn used to update the state estimate according to (10b). But an alternative procedure exists in the case of constant-parameter, as well as structured, state-space models [31, 43], where the $\mathbf{R}_{\mathbf{e},i}$ and $\mathbf{K}_i$ are computed without the need to explicitly evaluate $\mathbf{P}_i$.

To clarify this, we follow [31] and say that a general state-space model (Eq. 10a) is *structured* if there exist $n \times n$ matrices Ψ_i such that $\mathbf{F}_i$, $\mathbf{H}_i$, and $\mathbf{G}_i$ vary according to the following rules:

$$\mathbf{H}_i = \mathbf{H}_{i+1}\Psi_i,\ \mathbf{F}_{i+1}\,\Psi_i = \Psi_{i+1}\mathbf{F}_i,\ \mathbf{G}_{i+1} = \Psi_{i+1}\mathbf{G}_i \tag{17a}$$

Constant-parameter systems satisfy (Eq. 17a) with $\Psi_i = \mathrm{I}$. Also, the covariance matrices $\mathbf{R}_i$ and $\mathbf{Q}_i$ are assumed constant for all i; ($\mathbf{R}_i = \mathbf{R}$, $\mathbf{Q}_i = \mathbf{Q}$). More general cases are treated in [31].

The special model (Eq. 11a) exhibits $\mathbf{G}_i = \mathbf{Q}_i = 0$, $\mathbf{R}_i = 0$ and $\mathbf{F}_i = \lambda^{1/2}\mathbf{I}$. The $\mathbf{G}_i$ and $\mathbf{F}_i$ obviously satisfy (Eq. 17a) for any constant matrix Ψ. Hence, the model (Eq. 11a) will be structured if a Ψ exists such that the $\mathbf{H}_i$ matrix satisfies $\mathbf{H}_i = \mathbf{H}_{i+1}\,\Psi$. We shall see later that, in the context of RLS filtering, a shift structure in the input data results in a structured model (Eq. 11a) with a particular Ψ and, hence, the result of this section will be immediately applicable. So

assume there exists a Ψ such that $\mathbf{H}_i = \mathbf{H}_{i+1}\Psi$. We now verify that this leads to an order of magnitude speed up of the Kalman filter, for sufficiently sparse Ψ.

The savings in computation are achieved by considering the difference matrix $\mathbf{P}_{i+1} - \Psi\,\mathbf{P}_i\,\Psi^*$, as we further elaborate. If this difference has rank α then we can (nonuniquely) factor it as

$$\mathbf{P}_{i+1} - \Psi\mathbf{P}_i\Psi^* = \mathbf{L}_i\mathbf{S}_i\mathbf{L}_i^*$$

where $\mathbf{L}_i$ is an $n \times \alpha$ matrix, and $\mathbf{S}_i$ is an $\alpha \times \alpha$ signature matrix with as many ± 1's as $\mathbf{P}_{i+1} - \Psi\,\mathbf{P}_i\,\Psi^*$ has positive and negative eigenvalues. We now form the prearray of numbers

$$\begin{bmatrix} \mathbf{R}_{e,i}^{1/2} & \mathbf{H}_{i+1}\mathbf{L}_i \\ \Psi\,\overline{\mathbf{K}}_{p,i} & \lambda^{-1/2}\mathbf{L}_i \end{bmatrix}$$

and choose any $\mathbf{J} = (\mathbf{I} \oplus \mathbf{S}_i)$ -unitary matrix Θ_i, that block-triangularizes the prearray,

$$\begin{bmatrix} \mathbf{R}_{e,i}^{1/2} & \mathbf{H}_{i+1}\mathbf{L}_i \\ \Psi\overline{\mathbf{K}}_{p,i} & \lambda^{-1/2}\mathbf{L}_i \end{bmatrix} \Theta_i = \begin{bmatrix} \mathbf{X} & \mathbf{0} \\ \mathbf{Y} & \mathbf{Z} \end{bmatrix}$$

By comparing the J-norms on both sides of the equality,

$$\begin{bmatrix} \mathbf{R}_{e,i}^{1/2} & \mathbf{H}_{i+1}\mathbf{L}_i \\ \Psi\overline{\mathbf{K}}_{p,i} & \lambda^{-1/2}\mathbf{L}_i \end{bmatrix} \Theta_i \begin{bmatrix} \mathbf{I} & \mathbf{0} \\ \mathbf{0} & \mathbf{S}_i \end{bmatrix} \Theta_i^* \begin{bmatrix} \mathbf{R}_{e,i}^{1/2} & \mathbf{H}_{i+1}\mathbf{L}_i \\ \Psi\overline{\mathbf{K}}_{p,i} & \lambda^{-1/2}\mathbf{L}_i \end{bmatrix}^*$$

$$= \begin{bmatrix} \mathbf{X} & \mathbf{0} \\ \mathbf{Y} & \mathbf{Z} \end{bmatrix} \begin{bmatrix} \mathbf{I} & \mathbf{0} \\ \mathbf{0} & \mathbf{S}_i \end{bmatrix} \begin{bmatrix} \mathbf{X} & \mathbf{0} \\ \mathbf{Y} & \mathbf{Z} \end{bmatrix}^*$$

we conclude that we can make the identifications $\boldsymbol{X} = \mathbf{R}_{e,i+1}^{1/2}$, $\boldsymbol{Y} = \overline{\mathbf{K}}_{p,i+1}$ and

$$\mathbf{Z}\mathbf{S}_i\mathbf{Z}^* = \mathbf{P}_{i+2} - \Psi\mathbf{P}_{i+1}\Psi^*\,.$$

We can thus choose $\mathbf{Z} = \mathbf{L}_{i+1}$ and set $\mathbf{S}_{i+1} = \mathbf{S}_i$ since, by definition, $\mathbf{P}_{i+2} - \Psi\,\mathbf{P}_{i+1}\,\Psi^* = \mathbf{L}_{i+1}\,\mathbf{S}_{i+1}\,\mathbf{L}_{i+1}{}^*$. Note that our argument shows that the inertia matrix $\mathbf{S}_i$ does not vary with i, and we can choose it equal to $\mathbf{S}_0$, where $\mathbf{S}_0$ is defined via the factorization

$$\mathbf{P}_1 - \Psi\mathbf{P}_0\Psi^* = (\lambda^{-1}\Pi_0 - \mathbf{K}_0\mathbf{R}_{\mathbf{e},0}^{-1}\mathbf{K}_0^*) - \Psi\Pi_0\Psi^* = \mathbf{L}_0\mathbf{S}_0\mathbf{L}_0^*$$

The point is that we often get $\alpha << n$. The resulting arrays then propagate an $n \times \alpha$ factor, $\mathbf{L}_i$, rather than the $n \times n$ matrix $\mathbf{P}_i$.

Algorithm: The Square-Root Chandrasekhar Filter *Given the state space model (Eq. 11a), and assuming there exists an $n \times n$ matrix Ψ such that $\mathbf{H}_i = \mathbf{H}_{i+1}\Psi$, then the l.l.m.s. estimate $\hat{\mathbf{x}}_i$ can also be recursively computed as follows: let*

$$\mathbf{R}_{e,0} = \mathbf{I} + \mathbf{H}_0\Pi_0\mathbf{H}_0^*\,,\quad \mathbf{K}_0 = \lambda^{-1/2}\Pi_0\mathbf{H}_0^*$$

and factor the difference $\left[\lambda^{-1}\Pi_0 - \mathbf{K}_0\mathbf{R}_{\mathbf{e},0}^{-1}\mathbf{K}_0^* - \Psi\Pi_0\Psi^*\right]$ *as*

$$\lambda^{-1}\Pi_0 - \mathbf{K}_0\mathbf{R}_{\mathbf{e},0}^{-1}\mathbf{K}_0^* - \Psi\Pi_0\Psi^* = \mathbf{L}_0\mathbf{S}_0\mathbf{L}_0^*$$

where $\boldsymbol{L}_0$ is $n \times \alpha$, $\boldsymbol{S}_0$ is an $\alpha \times \alpha$ signature matrix, and α is the rank of the difference on the left-hand side.

Now start with $\hat{\mathbf{x}}_0 = \bar{\mathbf{x}}_0, \mathbf{R}_{e,0}^{1/2} = [\mathbf{I} + \mathbf{H}_0\Pi_0\mathbf{H}_0^*]^{1/2}$ $\overline{\mathbf{K}}_{p,0}, \mathbf{L}_0$ *and $\boldsymbol{S}_0$ as above, and Ψ, and repeat for $i \geq 0$,*

$$\begin{bmatrix} \mathbf{R}_{e,i}^{1/2} & \mathbf{H}_{i+1}\mathbf{L}_i \\ \Psi\overline{\mathbf{K}}_{p,i} & \lambda^{-1/2}\mathbf{L}_i \end{bmatrix} \Theta_i = \begin{bmatrix} \mathbf{R}_{e,i+1}^{1/2} & 0 \\ \overline{\mathbf{K}}_{p,i+1} & \mathbf{L}_{i+1} \end{bmatrix} \tag{17b}$$

where Θ_i is any $\mathbf{J} = (\mathbf{I} \oplus \boldsymbol{S}_0)$ – unitary matrix that produces the block zero entry in the postarray. Moreover,

$$\hat{\mathbf{x}}_{i+1} = \lambda^{-1/2}\hat{\mathbf{x}}_i + \overline{\mathbf{K}}_{p,i}\left[\mathbf{R}_{e,i}^{1/2}\right]^{-1}(\mathbf{y}_i - \mathbf{H}_i\hat{\mathbf{x}}_i)$$

where the quantities $\mathbf{R}_{e,i}^{1/2}$ and $\overline{\mathbf{K}}_{p,i}$ are propagated in the postarray.

In many situations we get $\alpha << n$. Examples will be given later in the adaptive filter area. If the matrix Ψ is sparse enough so that the product $\Psi\overline{\mathbf{K}}_{p,i}$ would require $O(n)$ operations, then the number of operations needed per iteration is $O(n)$ for $p << n$ and $\alpha << n$.

The Chandrasekhar Filter in Explicit Form

We also remark that the Chandrasekhar recursions can be alternatively expanded and rewritten in an unnormalized form [43, 44]. This is achieved by considering the alternative factorization

$$\mathbf{P}_{i+1} - \Psi\mathbf{P}_i\Psi^* = -\mathbf{L}_i^{(u)}\mathbf{R}_{r,i}^{-1}\mathbf{L}_i^{*(u)}$$

where $\mathbf{R}_{r,i}$ is an $\alpha \times \alpha$ matrix that is not necessarily restricted to a signature matrix. Using the factors $\mathbf{L}_i(\mathrm{u})$ and $\mathbf{R}_{r,i}$ one can check that the follwing equations hold (see, e.g., [31, 45, 47]).

Algorithm : The Explicit Chandrasekhar Filter *Given the state space model (11a), and assuming there exists an $n \times n$ matrix Ψ such that $\boldsymbol{H}_i = \boldsymbol{H}_{i+1}\Psi$, then the l.l.m.s. estimate $\hat{\mathbf{x}}_i$ can also be recursively computed as follows: let*

$$\mathbf{R}_{\mathbf{e},0} = \mathbf{I} + \mathbf{H}_0\Pi_0\mathbf{H}_0^*,\ \mathbf{K}_0 = \lambda^{-1/2}\Pi_0\mathbf{H}_0^*$$

and factor the difference $\left[\lambda^{-1}\Pi_0 - \mathbf{K}_0\mathbf{R}_{e,0}^{-1}\mathbf{K}_0^* - \Psi\Pi_0\Psi^*\right]$ *as*

$$\lambda^{-1}\Pi_0 - \mathbf{K}_0\mathbf{R}_{e,0}^{-1}\mathbf{K}_0^* - \Psi\Pi_0\Psi^* = -\mathbf{L}_0^{(u)}\mathbf{R}_{r,0}^{-1}\mathbf{L}_0^{*(u)}$$

where $\mathbf{L}_0(u)$ *is* $n \times \alpha$, $\mathbf{R}_{r,0}$ *is* $\alpha \times \alpha$, *and* α *is the rank of the difference on the left-hand side. Now start with* $\hat{\mathbf{x}}_0 = \bar{\mathbf{x}}_0$, $\mathbf{R}_{e,0}$, $\mathbf{K}_{p,0}$, $\mathbf{R}_{r,0}$, $\mathbf{L}_0^{(u)}$ *and* Ψ, *and repeat for* $i \geq 0$,

$$\begin{bmatrix} \mathbf{R}_{e,i} & \mathbf{H}_{i+1}\mathbf{L}_i^{(u)} \\ \Psi\mathbf{K}_i & \lambda^{-1/2}\mathbf{L}_i^{(u)} \\ \mathbf{L}_i^{*(u)}\mathbf{H}_{i+1}^* & \mathbf{R}_{r,i} \end{bmatrix} \Sigma_i = \begin{bmatrix} \mathbf{R}_{e,i+1} & 0 \\ \mathbf{K}_{i+1} & \mathbf{L}_{i+1}^{(u)} \\ 0 & \mathbf{R}_{r,i+1} \end{bmatrix} \tag{18}$$

where Σ_i *is given by*

$$\Sigma_i = \begin{bmatrix} \mathbf{I}_p & -\mathbf{R}_{e,i}^{-1}\mathbf{H}_{i+1} \\ -\mathbf{R}_{r,i}^{-1}\mathbf{L}_i^{*(u)}\mathbf{H}_{i+1}^* & \mathbf{I}_\alpha \end{bmatrix},$$

Moreover,

$$\hat{\mathbf{x}}_{i+1} = \lambda^{-1/2}\hat{\mathbf{x}}_i + \mathbf{K}_i\left[\mathbf{R}_{e,i}\right]^{-1}(\mathbf{y}_i - \mathbf{H}_i\hat{\mathbf{x}}_i).$$

It can be further verified that the above Σ_i satisfies the generalized unitarity relation

$$\Sigma_i^* \begin{bmatrix} \mathbf{R}_{e,i} & 0 \\ 0 & -\mathbf{R}_{r,i} \end{bmatrix} \Sigma_i = \begin{bmatrix} \mathbf{R}_{e,i+1} & \mathbf{0} \\ \mathbf{0} & -\mathbf{R}_{r,i+1} \end{bmatrix}$$

We may finally note that the above recursions are also related to the famous Levinson and Schur algorithms in prediction theory, as discussed in [24, 25, 47]. Table 11 summarizes the different forms of the Kalman recursions that we considered so far for the state-space model (11a).

The Recursive Least-Squares (RLS) Problem

We now move to formulate and solve the recursive least-squares problem by invoking the equivalence result (Eq. 11d) and by employing the state-space tools presented so far.

The core problem in adaptive filtering is the following. Consider a sequence of (N+1) scalar data points, $\{d(i)\}_{i=0}^N$, also known as reference or desired signals, and a sequence of (N+1) row vectors $\{\mathbf{u}_i\}_{i=0}^N$, also known as input signals. Each input vector $\mathbf{u}_i$ is a $1 \times M$ row vector whose individual entries we denote by $\{u_j(i)\}_{j=1}^M$ viz.,

$$\mathbf{u}_i = \left[u_1(i)\ u_2(i) \ldots u_M(i)\right] \tag{19a}$$

The entries of $\mathbf{u}_i$ can be regarded as the values of M input channels at time i: channels 1 through M. Consider also a known column vector $\bar{\mathbf{w}}$ and a positive-definite weighting matrix Π_0. The objective is to determine an $M \times 1$ column vector $\mathbf{w}$, also known as the weight vector, so as to minimize the weighted error sum:

$$\mathrm{E} = (\mathbf{w} - \bar{\mathbf{w}})^*\left[\lambda^{-(N+1)}\Pi_0\right]^{-1}(\mathbf{w} - \bar{\mathbf{w}}) + \sum_{i=0}^{N} \lambda^{N-i}|d(i) - \mathbf{u}_i\mathbf{w}|^2 \tag{19b}$$

where λ is a positive scalar that is less than or equal to one ($0 << \lambda \leq 1$). It is often called the forgetting factor since past data is exponentially weighted less than the more recent data. The special case $\lambda = 1$ is known as the growing memory case, since, as the length N of the data grows, the effect of past data is not attenuated. In contrast, the exponentially decaying memory case ($\lambda < 1$) is more suitable to time-variant environments. Also, the factor $\lambda^{-(N+1)}$ that multiplies Π_0 in the error-sum expression (Eq. 19b) can be incorporated into the weighting matrix Π_0. But it is left explicit for convenience as will become apparent later (see expressions (Eq. 20a) and (Eq. 20c) below). We shall denote the individual entries of $\mathbf{w}$ by $\{w(i)\}_{i=1}^M$,

$$\mathbf{w} = [w(1)\ w(2) \ldots w(M)]^{\mathbf{T}}$$

Before proceeding any further, let us provide a pictorial depiction of the problem at hand, see Figure 1. At each time instant i, the inputs of the M channels are linearly combined via the coefficients of the weight vector and the resulting signal is compared with the desired signal d(i). This results in a residual error $\varepsilon(i) = d(i) - \mathbf{u}_i\mathbf{w}$, for every i, and the objective is to find a weight vector $\mathbf{w}$ in order to minimize the (exponentially weighted) squared-sum of the residual errors over an interval of interest, say from $i = 0$ up to $i = N$.

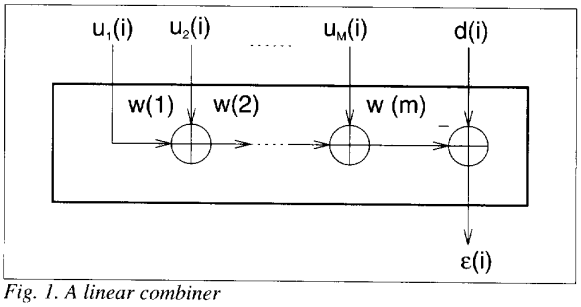

Fig. 1. A linear combiner

We also redraw the linear combiner of Figure 1 in a more compact form as in Figure 2. The input data $\{\mathbf{u}_i\}$ are fed into the linear combiner one at a time: $\mathbf{u}_0$, followed by $\mathbf{u}_1$, and so on. The reference signals $\{d(i)\}$ are also fed into the combiner sequentially starting with $d(0)$ and then $d(1)$ and so on. The objective is to determine a column vector $\mathbf{w}$ such that the *scalar* sequence generated by the inner products $\{\mathbf{u}_i\mathbf{w}\}$ follows the reference sequence $\{d(i)\}$ in the sense that the (exponentially weighted) squared-sum, or energy, of the resulting

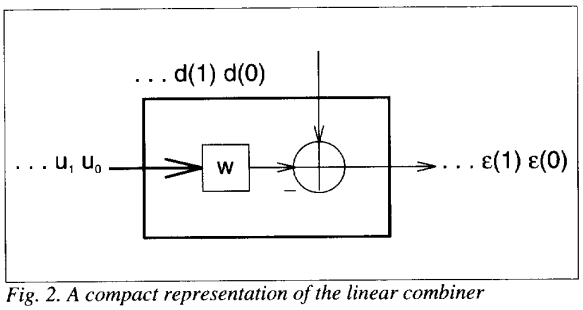

Fig. 2. A compact representation of the linear combiner

residuals $\{\varepsilon(0), \varepsilon(1), ...\}$ is minimized.

The linear combiner in either Figures 1 or 2 is said to be of order M since it is determined by M coefficients $\{w(j)\}_{j=1}^{M}$. However, a special case of the combiner in either figure will be relevant while deriving the lattice adaptive filters . It corresponds to a *first-order* combiner, viz., one that compares a *scalar* sequence of reference signals $d(i)$ with another also *scalar* sequence of input signals $u(i)$ via a single weighting coefficient, say w, as depicted in Figure 3. [As a side note, we remark that for the lattice filters that we discuss later , the scalar sequences $\{d(i), u(i)\}$ will be the sequences of forward and backward prediction errors.]

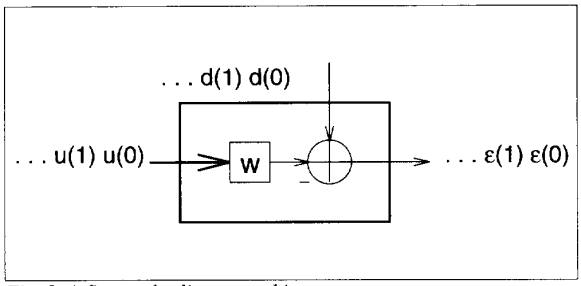

Fig. 3. A first-order linear combiner.

The expression for the weighted error-sum (Eq. 19b) can be rewritten in matrix form as well. For this purpose, we introduce the residual vector $\mathbf{e}_N$, the reference vector $\boldsymbol{d}_N$, the data matrix $\mathbf{D}_N$, and a diagonal weighting matrix Λ_N,

$$\mathbf{e}_N = \begin{bmatrix} d(0)-\mathbf{u}_0\mathbf{w} \\ d(1)-\mathbf{u}_1\mathbf{w} \\ d(2)-\mathbf{u}_2\mathbf{w} \\ \vdots \\ d(N)-\mathbf{u}_N\mathbf{w} \end{bmatrix} = \boldsymbol{d}_N - \boldsymbol{D}_N\mathbf{w}$$

$$= \begin{bmatrix} d(0) \\ d(1) \\ d(2) \\ \vdots \\ d(N) \end{bmatrix} - \begin{bmatrix} u_1(0) & u_2(0) & \dots & u_M(0) \\ u_1(1) & u_2(2) & \dots & u_M(1) \\ u_1(2) & u_2(2) & \dots & u_M(2) \\ \vdots & & & \vdots \\ u_1(N) & u_2(N) & \dots & u_M(N) \end{bmatrix} \mathbf{w} \qquad (19c)$$

$$\Lambda_N^{1/2} = \begin{bmatrix} (\lambda^{1/2})^N & & & \\ & (\lambda^{1/2})^{N-1} & & \\ & & \vdots & \\ & & & (\lambda^{1/2})^2 \\ & & & & 1 \end{bmatrix} \qquad (19d)$$

It then follows that

$$\mathrm{E} = (\mathbf{w}-\overline{\mathbf{w}})^*\left[\lambda^{-(N+1)}\Pi_0\right]^{-1}(\mathbf{w}-\overline{\mathbf{w}}) + \|\Lambda_N^{1/2}\mathbf{e}_N\|_2^2 . \qquad (19e)$$

The quantity $\Lambda_N^{1/2}\mathbf{e}_N$ is an exponentially weighted error vector,

$$\Lambda_N^{1/2}\mathbf{e}_N = \Lambda_N^{1/2}\mathbf{d}_N - \Lambda_N^{1/2}\mathbf{D}_N\mathbf{w}.$$

We conclude from (Eq. 9a) and (Eq. 9b) that the optimal solution $\hat{\mathbf{w}}$ is given by

$$(\hat{\mathbf{w}}-\overline{\mathbf{w}}) = \left[\lambda^{(N+1)}\Pi_0^{-1} + \mathbf{D}_N^*\Lambda_N\mathbf{D}_N\right]^{-1}\mathbf{D}_N^*\Lambda_N[\mathbf{d}_N - \mathbf{D}_N\overline{\mathbf{w}}]$$

This can be rewritten more compactly by introducing the "covariance" matrix Φ_N and the "cross-covariance" vector $\mathbf{s}_N$,

$$\Phi_N = \left[\lambda^{(N+1)}\Pi_0^{-1} + \mathbf{D}_N^*\Lambda_N\mathbf{D}_N\right]$$

$$\mathbf{s}_N = \mathbf{D}_N^*\Lambda_N\left[\mathbf{d}_N - \mathbf{D}_N\overline{\mathbf{w}}\right] \qquad (20a)$$

Then, the optimal solution is obtained by solving the following (also known as normal) linear system of equations,

$$\Phi_N(\hat{\mathbf{w}}-\overline{\mathbf{w}}) = \mathbf{s}_N \qquad (20b)$$

It is also rather straightforward to verify that Φ_N and s_N satisfy simple time-update relations, viz.,

$$\Phi_{N+1} - \lambda\Phi_N = \mathbf{u}_{N+1}^*\,\mathbf{u}_{N+1},$$
$$\mathbf{s}_{N+1} - \lambda\mathbf{s}_N = \mathbf{u}_{N+1}^*\left[d(N+1) - \mathbf{u}_{N+1}\overline{\mathbf{w}}\right], \qquad (20c)$$

with $\Phi_{-1} = \Pi_0^{-1}$ and $\mathbf{s}_{-1} = 0$. Hence, Φ_{N+1} and Φ_N differ only by a rank-one matrix.

The solution $\hat{\mathbf{w}}$ obtained by solving (Eq. 20b) is the optimal weight estimate based on the available data from time i = 0 up to time $i = N$. We shall denote it from now on by $\mathbf{w}_N$,

$$\Phi_N(\mathbf{w}_N - \overline{\mathbf{w}}) = \mathbf{s}_N$$

The subscript $_N$ in $\mathbf{w}_N$ indicates that the data up to, and including, time N were used. This is to differentiate it from the estimate obtained by using a different number of data points. Indeed, the main objective of the recursive least-

squares (RLS) problem is to show how to update the estimate $\mathbf{w}_N$, which is based on the data from time 0 to time N, to the estimate $\mathbf{w}_{N+1}$, which is based on the data from time 0 to time $(N+1)$, without the need to resolve a new set of linear equations of the form

$$\Phi_{N+1}\,(\hat{\mathbf{w}}_{N+1} - \overline{\mathbf{w}}) = \mathbf{s}_{N+1}.$$

Such a recursive update of the weight estimate should be possible since the coefficient matrices $\Phi_{\mathrm{N\ and}}\ \Phi_{N+1}$ of the associated linear systems differ only by a rank-one matrix. In fact, a wide variety of algorithms has been devised for this end and our purpose here is to derive these different schemes in a unified framework that is based on insights gained from state-space estimation techniques.

The State-Space Formulation

Let us now rework the error-sum expression (Eq. 19b) in order to reduce it to the same form (Eq. 11d) that we discussed previously. If the λ were equal to 1 then expression (Eq. 19b) would be identical in form to expression (Eq. 11d) and we could immediately apply the theorem stated after Eq. 11d.For a general non-unity λ, though, the expression (Eq. 19b) can be reworked to the familiar form (Eq. 11d) as follows:

$$\mathrm{E} = (\mathbf{w} - \overline{\mathbf{w}})^* \left[\lambda^{-(N+1)}\Pi_0\right]^{-1} (\mathbf{w} - \overline{\mathbf{w}}) + \sum_{i=0}^{N} \lambda^{N-i} |d(i) - \mathbf{u}_i \mathbf{w}|^2$$

$$= \lambda^N \left[(\mathbf{w} - \overline{\mathbf{w}})^* [\lambda^{-1}\Pi_0]^{-1} (\mathbf{w} - \overline{\mathbf{w}}) + \sum_{i=0}^{N} \left| \frac{d(i)}{(\sqrt{\lambda})^i} - \mathbf{u}_i \frac{\mathbf{w}}{(\sqrt{\lambda})^i} \right|^2 \right]$$

$$= \lambda^N \left[(\mathbf{x}_0 - \overline{\mathbf{x}}_0^*) [\lambda^{-1}\Pi_0]^{-1} (\mathbf{x}_0 - \overline{\mathbf{x}}_0) + \sum_{i=0}^{N} |y(i) - \mathbf{u}_i \mathbf{x}_i|^2 \right] \quad (21a)$$

where we have defined the quantities,

$$y(i) = \frac{d(i)}{(\sqrt{\lambda})^i}, \quad \mathbf{x}_i = \frac{\mathbf{w}}{(\sqrt{\lambda})^i}, \quad \mathbf{x}_0 = \mathbf{w}, \quad \overline{\mathbf{x}}_0 = \overline{\mathbf{w}}. \quad (21b)$$

Note that it follows from the definition of $\mathbf{x}_i$ that $\mathbf{x}_{i+1} = \lambda^{-1/2}\mathbf{x}_i$. This manipulation shows that we can always rescale the original problem (Eq. 19b), with an exponential factor λ, to an equivalent minimization problem with $\lambda = 1$, viz., (the constant factor λ_N does not affect the optimization problem)

$$\min_{\mathbf{x}_0} \left[(\mathbf{x}_0 - \overline{\mathbf{x}}_0)^* \left[\lambda^{-1}\Pi_0\right]^{-1} (\mathbf{x}_0 - \overline{\mathbf{x}}_0) + \sum_{i=0}^{N} |y(i) - \mathbf{u}_i \mathbf{x}_i|^2 \right]$$

where $\{\overline{\mathbf{x}}_0, y(i), \mathbf{u}_i, \Pi_0, N, \lambda\}$ are known and $\mathbf{x}_{i+1} = \lambda^{-1/2}\mathbf{x}_i$. This problem is clearly a special case of the equivalence result of the theorem stated after Eq. 11d. We thus conclude that the weight estimate can be recursively updated by writing down the state-space estimation algorithm (and its many variants) that corresponds to the following special M-dimensional state-space model,

$$\mathbf{x}_{i+1} = \lambda^{-1/2}\mathbf{x}_i$$
$$y(i) = \mathbf{u}_i \mathbf{x}_i + v(i), \quad (21c)$$

with

$$\mathbf{x}_0 = \mathbf{w}, \ \overline{\mathbf{x}}_0 = \overline{\mathbf{w}}, \ \mathrm{cov}(\mathbf{x}_0) = \lambda^{-1}\Pi_0, \ Ev(i)v^*(j) = \delta_{ij}.$$

This is a special case of the model (Eq. 11a) that we considered previously: the $\mathrm{y_i}$ is now a scalar variable $\mathbf{y}(i)$ and the matrix $\mathbf{H}_i$ is replaced by the row vector $\mathbf{u}_i$.

Also, an attractive feature of the scaling provided by (Eq. 21b) is that it leads to a state-space model (Eq. 21c) with a constant (in fact, equal to unity) noise variance, $Ev(i)v^*(j) = \delta_{ij}$. This will be helpful later when we further impose shift structure on the input channels and show that the resulting model (Eq. 21c) can be regarded as structured in the sense defined previously and for which the Chandrasekhar recursions will be immediately applicable. Furthermore, it also turns out, as shown in the next section, that the scaling (Eq. 21b) leads to a simple relation between the Riccati variable, $\mathbf{P_{i+1}}$, associated with the model (Eq. 21c) and the inverse of the "covariance" matrix, Φ_i^{-1}, associated with the least-squares problem (Eq. 20b): $\mathbf{P}_{i+1}$ will be a constant (viz., λ^{-1}) times Φ_i^{-1}. This means that if we write down the Riccati difference equation for $\mathbf{P}_{i+1}$ then it translates almost immediately to the widely-known RLS recursion for Φ_{i-1}. Other types of scaling may be used to define alternative auxiliary models as in (Eq. 21c). This would lead to alternative relations between the RLS and the Kalman variables, which can then be used to relate a Kalman-type algorithm to an RLS-type algorithm in precisely the same way as done in the remaining part of this paper. The scaling in (Eq. 21b) is motivated by the simple relation that it provides between $\mathbf{P}_{i+1}$ and Φ_i^{-1} and by the constant noise variance.

Relations between the RLS and Kalman Variables

The different variants of the Kalman recursions, when applied to (Eq. 21c), will now provide different algorithms for the solution of the RLS minimization problem (Eq. 19b). But note that the variables used in (Eq. 21c) are scaled versions of the variables used in (Eq. 19b). For example, y(i) is a scaled version of d(i). So is $\mathbf{x}_i$ relative to $\mathbf{w}$. This means that when writing down the state-space estimation algorithms that correspond to (Eq. 21c), we should then proceed to replace the variables of (Eq. 21c) by the corresponding RLS variables. This would allow us to describe the algorithms in terms of the original RLS variables. This section is meant to clarify the connections between the RLS variables $\{d(i), \mathbf{w}, \mathbf{u}_i, \Pi_0, \Phi_i, \mathbf{w}_i\}$ and the Kalman variables.

The Kalman recursions that correspond to the model (Eq. 21c) are given by: $\hat{\mathbf{x}}_0 = \bar{\mathbf{x}}_0 = \bar{\mathbf{w}}$, $\mathbf{P}_0 = \lambda^{-1}\Pi_0$,

$$\hat{\mathbf{x}}_{i+1} = \lambda^{-1/2}\hat{\mathbf{x}}_i + \mathbf{k}_i r_e^{-1}(i)e(i) \tag{22a}$$

where $e(i) = y(i) - \mathbf{u}_i\hat{\mathbf{x}}_i$, $\mathbf{k}_i = \lambda^{-1/2}\mathbf{P}_i\mathbf{u}_i^*$, $r_e(i) = 1 + \mathbf{u}_i\mathbf{P}_i\mathbf{u}_i^*$, and $\mathbf{P}_i$ satisfies the Riccati difference equation

$$\mathbf{P}_{i+1} = \lambda^{-1}\left[\mathbf{P}_i - \mathbf{P}_i\mathbf{u}_i^* r_e^{-1}(i)\mathbf{u}_i\mathbf{P}_i\right] \tag{22b}$$

Recall that $\mathbf{w}_i$ denotes the estimate of $\mathbf{w}$ that is based on the data $\{d(j), \mathbf{u}_j\}$ from time $j = 0$ to time $j = i$. Likewise, the state-estimate $\hat{\mathbf{x}}_{i+1}$ denotes the estimate of the state $\mathbf{x}_{i+1}$ that is based on observations $\{y(j)\}$ from time $j = 0$ to time $j = i$, and they are related as

$$\hat{\mathbf{x}}_{i+1} = \mathbf{w}_i/(\sqrt{\lambda})^{i+1}$$

With the RLS problem we also associate two residuals at each time instant i: the *apriori* error $e_a(i)$, defined by

$$e_a(i) = d(i) - \mathbf{u}_i\mathbf{w}_{i-1}$$

and the *aposteriori* error $e_p(i)$, defined by

$$e_p(i) = d(i) - \mathbf{u}_i\mathbf{w}_i$$

These residuals can be easily related to the Kalman filter innovation $e(i)$, which is defined by $e(i) = y(i) - \mathbf{u}_i\hat{\mathbf{x}}_i$ Its variance is denoted by $r_e(i)$. Now note that

$$e(i) = y(i) - \mathbf{u}_i\hat{\mathbf{x}}_i = \frac{1}{(\sqrt{\lambda})^i}\left[d(i) - \mathbf{u}_i\mathbf{w}_{i-1}\right] = \frac{1}{(\sqrt{\lambda})^i}\,e_a(i)$$

On the other hand, the expression for the aposteriori error $e_p(i)$ leads to

$$\begin{aligned} e_p(i) &= d(i) - \mathbf{u}_i\mathbf{w}_i = d(i) - (\sqrt{\lambda})^{i+1}\mathbf{u}_i\hat{\mathbf{x}}_{i+1} \\ &= d(i) - (\sqrt{\lambda})^{i+1}\mathbf{u}_i\left[\lambda^{-1/2}\hat{\mathbf{x}}_i + \mathbf{k}_i r_e^{-1}(i)e(i)\right] \\ &= \left[d(i) - \mathbf{u}_i\mathbf{w}_{i-1}\right] - \lambda^{1/2}\mathbf{u}_i\mathbf{k}_i r_e^{-1}(i)e_a(i) \\ &= \left[1 - \sqrt{\lambda}\mathbf{u}_i\mathbf{k}_i r_e^{-1}(i)\right]e_a(i) \\ &= \left[1 - \frac{\mathbf{u}_i\mathbf{P}_i\mathbf{u}_i^*}{1 + \mathbf{u}_i\mathbf{P}_i\mathbf{u}_i^*}\right]e_a(i) = r_e^{-1}(i)e_a(i) \end{aligned}$$

This means that the so-called *conversion factor* that converts the apriori error $e_a(i)$ to the aposteriori error $e_p(i)$, and which we denote by $\gamma(i)$ (following 1, page 578), is equal to $r_e^{-1}(i)$.

Finally, if we apply the matrix inversion lemma (Eq. 5) to (Eq. 22b) we obtain

$$\mathbf{P}_{i+1}^{-1} - \lambda\mathbf{P}_i^{-1} = \lambda\mathbf{u}_i^*\mathbf{u}_i\,,\quad \mathbf{P}_0^{-1} = \lambda\Pi_0^{-1}$$

and it readily follows (recall (Eq. 20c)) that $\mathbf{P}_{i+1}^{-1}$ is a scaled version of the "covariance" matrix Φ_i that we defined in (Eq. 20a). More precisely,

$$\mathbf{P}_{i+1}^{-1} = \lambda\Phi_i$$

If we denote the inverse of the "covariance" matrix Φ_i by $\bar{\mathbf{P}}_i$, *viz.*, $\bar{\mathbf{P}}_i = \Phi_i^{-1}$, then we also have that

$$\mathbf{P}_{i+1} = \lambda^{-1}\bar{\mathbf{P}}_i$$

[This notation is chosen to reflect the one conventional in RLS estimation – see (Eq. 23). Otherwise a different symbol than $\bar{\mathbf{P}}$ would be better.]

We further denote $\sqrt{\lambda}\mathbf{k}_i r_e^{-1}(i)$ by $\mathbf{g}_i$, which implies that $\sqrt{\lambda}\bar{\mathbf{k}}_{p,i} = \mathbf{g}_i\gamma^{-*/2}(i) \equiv \bar{\mathbf{g}}_i$. The correspondence between the Kalman and the RLS variables, and which follows from the scaling (Eq. 21b), are summarized in Table 12. The entries of the table allow us to translate a Kalman-type algorithm to an RLS-type algorithm, and vice-versa.

Different Classes of Fixed-Order Adaptive Algorithms

Now that we have established a clear correspondence between the Kalman and the RLS variables, we can proceed to write down the different variants of the Kalman recursions in terms

Table 12: Correspondence between the Kalman and RLS variables.

Kalman Variables	RLS Variables	Description
$y(i)$	$d(i)/(\sqrt{\lambda})^i$	Reference signal.
$\mathbf{x}_i$	$\mathbf{w}/(\sqrt{\lambda})^i$	Weight vector.
$\hat{\mathbf{x}}_{i+1}$	$\mathbf{w}_i/(\sqrt{\lambda})^{i+1}$	Weight estimate.
$\lambda\mathbf{P}_{i+1}$	$\Phi_i^{-1} = \bar{\mathbf{P}}_i$	Inverse of covariance matrix.
$\sqrt{\lambda}\,\mathbf{k}_{p,i}\,r_e^{-1}(i)$	$\mathbf{g}_i$	Gain vector.
$\bar{\mathbf{k}}_{p,i}$	$\frac{1}{\sqrt{\lambda}}\mathbf{g}_i\,\gamma^{-*/2}(i)$	Normalized gain vector.
$\bar{\mathbf{k}}_{p,i}$	$\frac{1}{\sqrt{\lambda}}\bar{\mathbf{g}}_i$	Normalized gain vector.
$e(i)$	$\frac{e_a(i)}{(\sqrt{\lambda})^i}$	Apriori error.
$e(i)$	$\frac{e_p(i)}{(\sqrt{\lambda})^i}\,r_e(i)$	Aposteriori error.
$r_e^{-1}(i)$	$\gamma(i)$	Conversion factor.
$\bar{\mathbf{x}}_0$	$\bar{\mathbf{w}}$	Initial guess.
$\hat{\mathbf{x}}_0$	$\mathbf{w}_{-1} = \bar{\mathbf{w}}$	Initial guess.
$\mathbf{P}_0$	$\lambda^{-1}\,\Pi_0$	Weighting matrix.

of the original RLS variables.

The RLS Algorithm

The first step is to use the correspondences of Table 12 in order to rewrite the Kalman recursions (Eq. 22a) and (Eq. 22b) in terms of the RLS variables. This leads to the widely known RLS algorithm (see, e.g., [1, p. 483]).

Algorithm: *Consider a set of* $(N+1)$ *data* $\{\mathbf{u}_i, d(i)\}_{i=0}^{N}$, *where the* $\mathbf{u}_i$ *are* $1 \times M$ *row vectors and the* $d(i)$ *are scalars. Consider also an* $M \times 1$ *column vector* $\overline{\mathbf{w}}$, *an* $M \times M$ *positive-definite matrix* Π_0 *and a scalar* λ $(0 << \lambda \leq 1)$. *The solution of the minimization problem*

$$\min_{\mathbf{w}} \left[(\mathbf{w}-\overline{\mathbf{w}})^* \left[\lambda^{-(N+1)} \Pi_0 \right]^{-1} (\mathbf{w}-\overline{\mathbf{w}}) + \sum_{i=0}^{N} \lambda^{N-i} |d(i) - \mathbf{u}_i \mathbf{w}|^2 \right]$$

can be recursively computed as follows: start with $\mathbf{w}_{-1} = \overline{\mathbf{w}}$, $\overline{\mathbf{P}}_{-1} = \Pi_0$ *and repeat for* $i \geq 0$:

$$\begin{aligned} \mathbf{w}_i &= \mathbf{w}_{i-1} + \mathbf{g}_i \left[d(i) - \mathbf{u}_i \mathbf{w}_{i-1} \right] \\ \mathbf{g}_i &= \frac{\lambda^{-1} \overline{\mathbf{P}}_{i-1} \mathbf{u}_i^*}{1 + \lambda^{-1} \mathbf{u}_i \overline{\mathbf{P}}_{i-1} \mathbf{u}_i^*} \\ \overline{\mathbf{P}}_i &= \lambda^{-1} \left[\overline{\mathbf{P}}_{i-1} - \mathbf{g}_i \mathbf{u}_i \overline{\mathbf{P}}_{i-1} \right] \end{aligned} \tag{23}$$

Then the optimal solution is $\mathbf{w} = \mathbf{w}_N$. *The computational complexity of the algorithm is* $O(M^2)$ *per iteration.*

The equivalence between the Kalman recursions and the RLS recursions is rather well-known in the special case $\lambda = 1$. However, surprising as it may seem, it has not been clearly shown for $\lambda \neq 1$. The reason is that the equivalence is usually established by working out the solution of the RLS problem and comparing it with the Kalman filter equations for an appropriate state-space model of the form (Eq. 21c) (with $\lambda = 1$, see, e.g., [5, pp. 331–335]). However, as noted in [1, pp. 502–504], when $\lambda \neq 1$, a complete equivalence of the RLS solution and the Kalman filter recursions is not immediate. The simple, in retrospect, device is a proper scaling of certain variables, as shown in (Eq. 21b), along with a proper identification of the correspondences between Kalman-type variables and RLS-type variables as detailed in Table 12.

The Square-Root RLS (Inverse QR) Algorithm

We now write down the square-root Kalman filter (Eq. 15c) that corresponds to (Eq. 21c), viz., $\hat{\mathbf{x}}_0 = \overline{\mathbf{x}}_0 = \overline{\mathbf{w}}$, and $\mathbf{P}_0^{1/2} = \lambda^{-1/2} \Pi_0^{1/2}$, and for $i \geq 0$,

$$\begin{bmatrix} 1 & \mathbf{u}_i \mathbf{P}_i^{1/2} \\ 0 & \lambda^{-1/2} \mathbf{P}_i^{1/2} \end{bmatrix} \Theta_i = \begin{bmatrix} r_e^{1/2}(i) & 0 \\ \overline{\mathbf{k}}_{p,i} & \mathbf{P}_{i+1}^{1/2} \end{bmatrix}$$

where Θ_i is any unitary transformation that produces the block zero entry in the postarray, and

$$\hat{\mathbf{x}}_{i+1} = \lambda^{-1/2} \hat{\mathbf{x}}_i + \overline{\mathbf{k}}_{p,i} \left[r_e^{1/2}(i) \right]^{-1} (y(i) - \mathbf{u}_i \hat{\mathbf{x}}_i)$$

In terms of the RLS variables, this leads to the so-called square-root RLS algorithm.

Algorithm: *Consider a set of* $(N+1)$ *data* $\{\mathbf{u}_i, d(i)\}_{i=0}^{N}$ *where the* $\mathbf{u}_i$ *are* $1 \times M$ *row vectors and the* $d(i)$ *are scalars. Consider also an* $M \times 1$ *column vector* $\overline{\mathbf{w}}$, *an* $M \times M$ *positive-definite matrix* Π_0 *and a scalar* λ $(0 << \lambda \leq 1)$. *The solution of the minimization problem*

$$\min_{\mathbf{w}} \left[(\mathbf{w}-\overline{\mathbf{w}})^* \left[\lambda^{-(N+1)} \Pi_0 \right]^{-1} (\mathbf{w}-\overline{\mathbf{w}}) + \sum_{i=0}^{N} \lambda^{N-i} |d(i) - \mathbf{u}_i \mathbf{w}|^2 \right]$$

can be recursively computed as follows: start with $\mathbf{w}_{-1} = \overline{\mathbf{w}}$, $\overline{\mathbf{P}}_{-1}^{1/2} = \Pi_0^{1/2}$ *and repeat for* $i \geq 0$:

$$\begin{bmatrix} 1 & \frac{1}{\sqrt{\lambda}} \mathbf{u}_i \overline{\mathbf{P}}_{i-1}^{1/2} \\ 0 & \frac{1}{\sqrt{\lambda}} \overline{\mathbf{P}}_{i-1}^{1/2} \end{bmatrix} \Theta_i = \begin{bmatrix} \gamma^{-*/2}(i) & 0 \\ \mathbf{g}_i \gamma^{-*/2}(i) & \overline{\mathbf{P}}_i^{1/2} \end{bmatrix}$$

where Θ_i *is any unitary rotation that produces the block zero entry in the postarray, and*

$$\mathbf{w}_i = \mathbf{w}_{i-1} + \left[\mathbf{g}_i \gamma^{-*/2}(i) \right] \left[\gamma^{-*/2}(i) \right]^{-1} \left[d(i) - \mathbf{u}_i \mathbf{w}_{i-1} \right]$$

Then the optimal solution is $\mathbf{w} = \mathbf{w}_N$. *The computational complexity of the algorithm is* $O(M^2)$ *per iteration.*

The above recursions are often known as the square-root RLS algorithm (see, e.g., [5]) but, more recently, they have also been refered to as the inverse QR algorithm (see, e.g., [48, 49]). An extension of this result to the mutlichannel case is described in [51]. The reason for the terminology is that the recursions propagate a square-root factor of the inverse of the "covariance" matrix Φ_i, while the conventional QR algorithm that we derive in the next two sections propagates a square-root factor of Φ_i itself.

The Conventional QR Algorithm

We now write down the first three lines of the square-root information array (Eq. 6a) that corresponds to (Eq. 21c), viz., $\hat{\mathbf{x}}_0 = \overline{\mathbf{x}}_0 = \overline{\mathbf{w}}$, $\mathbf{P}_0^{1/2} = \lambda^{-1/2} \Pi_0^{1/2}$, and for $i \geq 0$,

$$\begin{bmatrix} \lambda^{1/2}\mathbf{P}_i^{-*/2} & \lambda^{1/2}\mathbf{u}_i^* \\ \hat{\mathbf{x}}_i^*\mathbf{P}_i^{-*/2} & y^*(i) \\ 0 & 1 \end{bmatrix} \Theta_i = \begin{bmatrix} \mathbf{P}_{i+1}^{-*/2} & 0 \\ \hat{\mathbf{x}}_{i+1}^*\mathbf{P}_{i+1}^{-*/2} & e^*(i)r_e^{-*/2}(i) \\ \lambda^{1/2}\mathbf{u}_i\mathbf{P}_{i+1}^{1/2} & r_e^{-*/2}(i) \end{bmatrix}$$

where Θ_i is any unitary matrix that produces the block zero entry in the postarray. In terms of the RLS variables, this leads to the conventional QR algorithm (see, e.g., 1 [p. 518, pp. 534 – 538]. First a clarification on the notation used below in the statement of the algorithm. The quantity $\hat{\mathbf{x}}_i^*\mathbf{P}_i^{-*/2}$ that appears in the second line of the above prearray translates into $\frac{1}{(\sqrt{\lambda})^{i-1}}\mathbf{w}_{i-1}^*\Phi_{i-1}^{1/2}$. We then denote the term $\mathbf{w}_{i-1}^*\Phi_{i-1}^{1/2}$ by $\mathbf{q}_{i-1}^*$, i.e.,

$$\mathbf{q}_{i-1} = \Phi_{i-1}^{*/2}\mathbf{w}_{i-1}$$

It further follows from the normal equations (Eq. 20b) that

$$\mathbf{q}_{i-1} = \Phi_{i-1}^{-1/2}\mathbf{s}_{i-1} + \Phi_{i-1}^{*/2}\bar{\mathbf{w}}$$

Algorithm: *Consider a set of* $(N+1)$ *data* $\{\mathbf{u}_i, d(i)\}_{i=0}^{N}$, *where the* $\mathbf{u}_i$ *are* $1 \times M$ *row vectors and the* $d(i)$ *are scalars. Consider also an* $M \times 1$ *column vector* $\bar{\mathbf{w}}$, an $M \times M$ *positive-definite matrix* Π_0 *and a scalar* λ $(0 << \lambda \leq 1)$. *The solution of the minimization problem*

$$\min_{\mathbf{w}} \left[(\mathbf{w}-\bar{\mathbf{w}})^* \left[\lambda^{-(N+1)}\Pi_0\right]^{-1} (\mathbf{w}-\bar{\mathbf{w}}) + \sum_{i=0}^{N} \lambda^{N-i} |d(i) - \mathbf{u}_i\mathbf{w}|^2 \right]$$

can be recursively computed as follows: start with $\mathbf{w}_{-1} = \bar{\mathbf{w}}, \Phi_{-1}^{1/2} = \Pi_0^{-*/2}, \mathbf{q}_{-1} = \Pi_0^{-1/2}\bar{\mathbf{w}}$, *and repeat for* $i \geq 0$:

$$\begin{bmatrix} \sqrt{\lambda}\Phi_{i-1}^{1/2} & \mathbf{u}_i^* \\ \sqrt{\lambda}\mathbf{q}_{i-1}^* & d^*(i) \\ 0 & 1 \end{bmatrix} \Theta_i = \begin{bmatrix} \Phi_i^{1/2} & 0 \\ \mathbf{q}_i & e_a^*(i)\gamma^{1/2}(i) \\ \mathbf{u}_i\Phi_i^{-*/2} & \gamma^{1/2}(i) \end{bmatrix} \tag{24}$$

Then the optimal solution is $\mathbf{w} = \mathbf{w}_N$, *where* $\mathbf{w}_N$ *is obtained by solving the upper triangular system of equations:*

$$\Phi_N^{*/2}\mathbf{w}_N = \mathbf{q}_N$$

The computational complexity of the algorithm is $O(M^2)$ *per iteration.*

The Extended QR Algorithm

The conventional QR solution determines the optimal weight $\mathbf{w}_N$ by solving a triangular linear system of equations, e.g., via back-substitution,

$$\Phi_N^{*/2}\mathbf{w}_N = \mathbf{q}_N$$

A major drawback of a back-substitution step is that it involves serial operations and, thus, does not lend itself to a parallelizable implementation. However, this step can be completely avoided by further invoking the last line of the square-root information form (Eq. 16a) that we have ignored so far. The resulting algorithm is an extended version of the QR recursion and completely avoids the backsubstitution step (see also [22, 24, 50]). The extended square-root information filter that corresponds to (Eq. 21c) is the following: $\hat{\mathbf{x}}_0 = \bar{\mathbf{x}}_0 = \bar{\mathbf{w}}$, $\mathbf{P}_0^{1/2} = \lambda^{-1/2}\Pi_0^{1/2}, \mathbf{P}_0^{-1/2} = \lambda^{1/2}\Pi_0^{-1/2}$, and for $i \geq 0$,

$$\begin{bmatrix} \lambda^{1/2}\mathbf{P}_i^{-*/2} & \lambda^{1/2}\mathbf{u}_i^* \\ \hat{\mathbf{x}}_i^*\mathbf{P}_i^{-*/2} & y^*(i) \\ 0 & 1 \\ \lambda^{-1/2}\mathbf{P}_i^{1/2} & 0 \end{bmatrix} \Theta_i = \begin{bmatrix} \mathbf{P}_{i+1}^{-*/2} & 0 \\ \hat{\mathbf{x}}_{i+1}^{-*/2} & e^*(i)r_e^{-*/2}(i) \\ \lambda^{1/2}\mathbf{u}_i\mathbf{P}_{i+1}^{-*/2} & r_e^{-*/2}(i) \\ \mathbf{P}_{i+1}^{1/2} & -\bar{\mathbf{k}}_{p,i} \end{bmatrix}$$

where Θ_i is any unitary matrix that produces the block zero entry in the top block row of the postarray. The state estimate is then given by

$$\hat{\mathbf{x}}_{i+1} = \lambda^{-1/2}\hat{\mathbf{x}}_i + \bar{\mathbf{k}}_{p,i}\left[r_e^{-1/2}(i)e(i)\right]$$

where the quantities $\bar{\mathbf{k}}_{p,i}$ and $r_e^{-1/2}(i)e(i)$ are immediately read from the entries of the postarray (second and last lines). This expression allows us to update the weight estimate without invoking a back-substitution step. In terms of the RLS variables, we obtain the following algorithm.

Algorithm: *Consider a set of* $(N+1)$ *data* $\{\mathbf{u}_i, d(i)\}_{i=0}^{N}$, *where the* $\mathbf{u}_i$ *are* $1 \times M$ *row vectors and the* $d(i)$ *are scalars. Consider also an* $M \times 1$ *column vector* $\bar{\mathbf{w}}$, *an* $M \times M$ *positive-definite matrix* Π_0 *and a scalar* λ $(0 << 1 \leq 1)$. *The solution of the minimization problem*

$$\min_{\mathbf{w}} \left[(\mathbf{w}-\bar{\mathbf{w}})^* \left[\lambda^{-(N+1)}\Pi_0\right]^{-1} (\mathbf{w}-\bar{\mathbf{w}}) + \sum_{i=0}^{N} \lambda^{N-i} |d(i) - \mathbf{u}_i\mathbf{w}|^2 \right]$$

can be recursively computed as follows: start with $\mathbf{w}_{-1} = \bar{\mathbf{w}}$, $\Phi_{-1}^{1/2} = \Pi_0^{-*/2}$, $\Phi_{-1}^{-1/2} = \Pi_0^{*/2}$, $\mathbf{q}_{-1} = \Pi_0^{-1/2}\bar{\mathbf{w}}$, *and repeat for* $i \geq 0$:

$$\begin{bmatrix} \sqrt{\lambda}\Phi_{i-1}^{1/2} & \mathbf{u}_i \\ \sqrt{\lambda}\mathbf{q}_{i-1}^* & d^*(i) \\ 0 & 1 \\ \frac{1}{\sqrt{\lambda}}\Phi_{i-1}^{-*/2} & 0 \end{bmatrix} \Theta_i = \begin{bmatrix} \Phi_i^{1/2} & 0 \\ \mathbf{q}_i^* & e_a^*(i)\gamma^{1/2}(i) \\ \mathbf{u}_i\Phi_i^{-*/2} & \gamma^{1/2}(i) \\ \Phi_i^{-*/2} & -\mathbf{g}_i\gamma^{-*/2}(i) \end{bmatrix} \tag{25a}$$

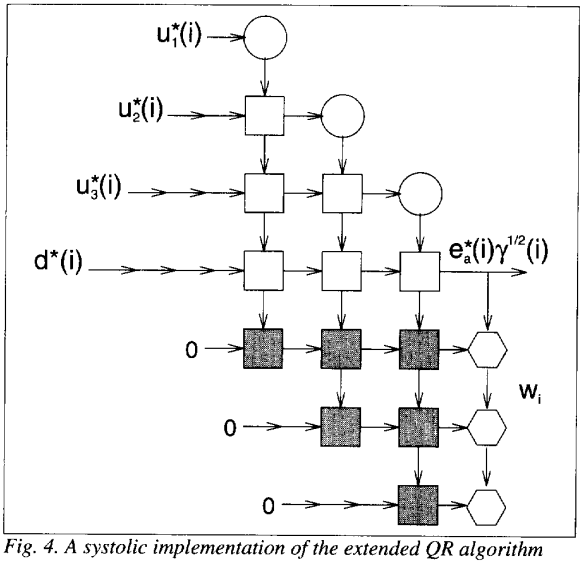

Fig. 4. A systolic implementation of the extended QR algorithm

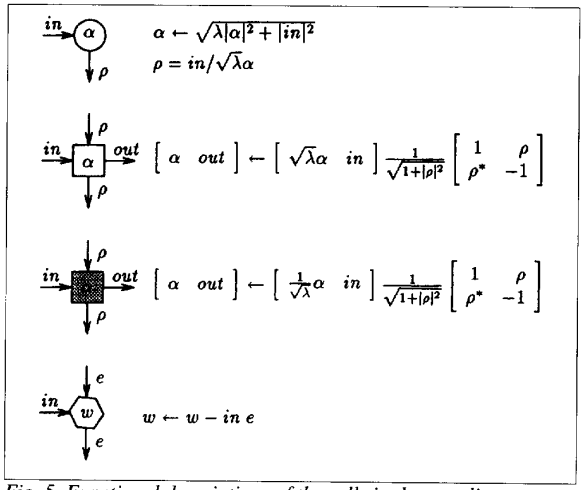

Fig. 5. Functional descriptions of the cells in the systolic array.

$$\mathbf{w}_i = \mathbf{w}_{i-1} + \left[\mathbf{g}_i\gamma^{-*/2}(i)\right]\left[e_a^*(i)\gamma^{1/2}(i)\right]^* \tag{25b}$$

where the quantities $\mathbf{g}_i\gamma^{-*/2}(i)$ *and* $e_a^*(i)\gamma^{1/2}(i)$ *are read directly from the second and last lines of the postarray. The optimal solution is* $\mathbf{w} = \mathbf{w}_N$*, which can also be computed via* $\mathbf{w}_N = \Phi_N^{-*/2}\mathbf{q}_N$*. Moreover, the computational complexity per step is* $O(M^2)$*.*

The time-update (Eq. 25b) of the weight-estimate uses only quantities that are immediately available from the postarray. This procedure admits a systolic implementation as depicted in Figures 4 and 5 for a thrid-order linear combiner. The triangular array on the top rotates $\mathbf{u}_i^*$ and $\sqrt{\lambda}\Phi_{i-1}^{1/2}$ into $\Phi_i^{1/2}$ and zero. The triangular array on the bottom rotates $\frac{1}{\sqrt{\lambda}}\Phi_{i-1}^{-*/2}$ and zero to yield $\Phi_i^{-*/2}$ and $\mathbf{g}_i\gamma^{-*/2}(i)$. The linear array rotates $d^*(i)$ and $\sqrt{\lambda}\mathbf{q}_{i-1}^*$ into $\mathbf{q}_i^*$ and $e_a^*(i)\gamma_i^{1/2}$. A motivation for this solution using insights from the theory of structured matrices is provided in [50].

Therefore, the extended QR algorithm is precisely the extended information form of the standard Kalman filter; however this connection could not be established till the RLS problem was properly recast into a state-space estimation problem.

Fast Transversal Filter Algorithms

We showed in the previous sections that the RLS adaptive algorithm, and several of its variants, can be obtained by setting up a suitable state-space model, viz., model (Eq. 21c), and by using variations of the state-space estimation algorithm.

Now, the state-space model we set up has a very special structure: $\mathbf{F}_i = \lambda^{-1/2}\,\mathbf{I}$, $\mathbf{G}_i = 0$, and $\mathbf{R}_i = 1$ are constant, while

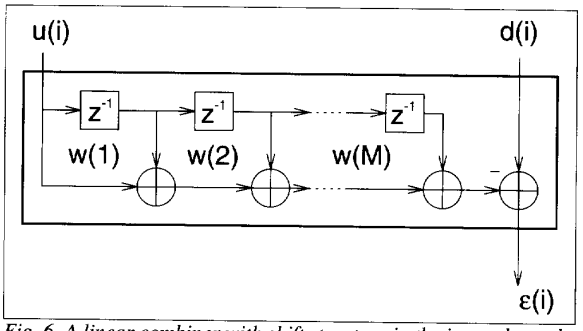

Fig. 6. A linear combiner with shift structure in the input channels.

$\mathbf{H}_i = \mathbf{u}_i$ is not. The entries of the input vector $\mathbf{u}_i$ are the values of M channels at time i (cf. (Eq. 19a)). But it often happens in practice that the channel inputs are not totally independent. In fact, they are usually delayed versions of a single input signal as follows. If we denote the value of the first channel at time i by $u(i)$ (instead of $u_1(i)$ as we did before in (Eq. 19a), then the input vector at time i will exhibit a shift structure of the form

$$\mathbf{u}_i = [u(i)\ u(i-1)\ \ldots\ u(i-M+1)] \tag{26a}$$

This has a simple pictorial representation as shown in Figure 6. The term z^{-1} represents a unit-time delay. The structure that takes $u(i)$ as an input and provides the inner product $\sum_{j=1}^{M} u(i+1-j)w(j)$ as an output is widely known as a transversal or as an FIR (finite-impulse response) filter.

The shift structure in $\mathbf{u}_i$ suggests that we might be able to get fast RLS algorithms by using the extended Chandrasekhar

recursions in place of the Riccati-based recursions. In fact this is true, and many results in the literature can be obtained in a more transparent (square-root array) form, and many variations and extensions derived in this way. Consider, for instance, two successive input vectors $\mathbf{u}_i$ and $\mathbf{u}_{i+1}$,

$$\mathbf{u}_i = [u(i)\ u(i-1)\ \ldots\ u(i-M+2)\ u(i-M+1)]$$
$$\mathbf{u}_{i+1} = [u(i+1)\ u(i)\ \ldots\ u(i-M+3)\ u(i-M+2)]$$

It is clear that the first $(M-1)$ entries of $\mathbf{u}_i$ and the last $(M-1)$ entries of $\mathbf{u}_{i+1}$ coincide. Alternatively, $\mathbf{u}_i$ and $\mathbf{u}_{i+1}$ can be related as follows:

$$\mathbf{u}_i = \mathbf{u}_{i+1}\mathbf{Z} + u(i-M+1)[0\ \ldots\ 0\ \ 1\ ] \tag{26b}$$

where $\mathbf{Z}$ denotes the lower triangular shift matrix with ones on the first subdiagonal and zeros elsewhere. Multiplying $\mathbf{u}_{i+1}$ by $\mathbf{Z}$ from the right corresponds to shifting its entries by one position to the left. The relation (Eq. 26b) can be simplified if we consider an alternative, but equivalent, state-space model instead of (Eq. 21c).

More specifically, we consider the following $(N+1)$ –dimensional state-space model (the model in (Eq. 21c) was M –dimensional):

$$\mathbf{x}_{i+1} = \lambda^{-1/2}\mathbf{x}_i$$
$$y(i) = \mathbf{h}_i\mathbf{x}_i + v(i) \tag{27a}$$

with

$$\mathbf{x}_0 = \begin{bmatrix}\mathbf{w}\\ \mathbf{0}\end{bmatrix},\ \bar{\mathbf{x}}_0 = \begin{bmatrix}\overline{\mathbf{w}}\\ \mathbf{0}\end{bmatrix},\ Ev(i)v^*(j) = \delta_{ij},$$

$$\mathrm{cov}(\mathbf{x}_0) = \begin{bmatrix}\lambda^{-1}\Pi_0 & \mathbf{0}\\ \mathbf{0} & \mathbf{0}\end{bmatrix} = \lambda^{-1}\Pi_0 \oplus \mathbf{0}$$

and where the $(N+1)$-dimensional row vector $\mathbf{h}_i$ is defined as

$$\mathbf{h}_i = [u(i)\ u(i-1)\ \ldots\ u(0)\ \mathbf{0}\ ]$$

That is, $\mathbf{h}_i$ has all the input data from time 0 up to and including time i. The remaining entries are filled with zeros. This is again a special case of (Eq. 11a). It is also evident that by extending the state with zeros, the product $\mathbf{h}_i\mathbf{x}_i$ in (Eq. 27a) is still equal to the product $\mathbf{u}_i\mathbf{x}_i$ in (Eq. 21c). Moreover, the top M entries of the state-estimate will again provide us with the weight-estimate. But note now that the $\mathbf{h}_i$ satisfies

$$\mathbf{h}_i = \mathbf{h}_{i+1}\mathbf{Z}$$

which is a simpler relation than (Eq. 26b). It also immediately implies that the extended state-space model (Eq. 27a) is structured in the sense that we defined earlier via (17a). We thus expect to obtain a fast algorithm for state-estimation via the Chandrasekhar recursions (Eq. 17b). We shall see that this is indeed the case.

Returning to (Eq. 27a), if we write down the corresponding Kalman recursions (Eq. 13) we obtain

$$\hat{\mathbf{x}}_{i+1} = \lambda^{-1/2}\hat{\mathbf{x}}_i + \mathbf{k}_i r_e^{-1}(i)[y(i) - \mathbf{h}_i\hat{\mathbf{x}}_i]$$
$$r_e(i) = 1 + \mathbf{h}_i\mathbf{P}_i\mathbf{h}_i^*,\ \ \mathbf{k}_i = \lambda^{-1/2}\mathbf{P}_i\mathbf{h}_i^*$$
$$\mathbf{P}_{i+1} = \lambda^{-1}\left[\mathbf{P}_i - \mathbf{P}_i\mathbf{h}_i^* r_e^{-1}(i)\mathbf{h}_i\mathbf{P}_i\right]$$

with $\mathbf{P}_0 = \lambda^{-1}\Pi_0 \oplus \mathbf{0}$. Due to the trailing zeros in $\mathbf{P}_0$ and $\mathbf{h}_i$, it can be easily verified that the gain vectors $\mathbf{k}_i$ and $\bar{\mathbf{k}}_{p,i} = \mathbf{k}_i r_e^{-*/2}(i)$ also have trailing zeros and we express them as

$$\mathbf{k}_i = \begin{bmatrix}\mathbf{c}_i\\ \mathbf{0}\end{bmatrix},\ \ \bar{\mathbf{k}}_{p,i} = \begin{bmatrix}\bar{\mathbf{c}}_i\\ \mathbf{0}\end{bmatrix}$$

where $\mathbf{c}_i$ and $\bar{\mathbf{c}}_i$ are $M \times 1$. The computational complexity of the above algorithm is $O(M^2)$ operations (multiplications and additions) per time step. However, though time-variant, the special structure of $\mathbf{h}_i$, viz., $\mathbf{h}_i = \mathbf{h}_{i+1}\mathbf{Z}$, can be exploited to reduce the operation count to $O(M)$. This is done by invoking the Chandrasekhar recursions. To apply the Chandrasekhar recursions (Eq. 17b), we first need a factorization of the difference $\mathbf{P}_1 - \mathbf{Z}\mathbf{P}_0\mathbf{Z}^*$. So assume $\mathbf{P}_1 - \mathbf{Z}\mathbf{P}_0\mathbf{Z}^*$ turns out to be of low rank, say α (more on this further ahead), and let us factor it as

$$\mathbf{L}_0\mathbf{S}_0\mathbf{L}_0^* = \mathbf{P}_1 - \mathbf{Z}\mathbf{P}_0\mathbf{Z}^*$$

where $\mathbf{L}_0$ and $\mathbf{S}_0$ are $(N+1) \times \alpha$ and $\alpha \times \alpha$ matrices, respectively. The factor $\mathbf{L}_0$ also has trailing zeros and we partition it in the form

$$\mathbf{L}_0 = \begin{bmatrix}\tilde{\mathbf{L}}_0\\ \mathbf{0}\end{bmatrix}$$

where $\tilde{\mathbf{L}}_0$ is $(M+1) \times \alpha$. Let also $\tilde{\mathbf{h}}_i$ denote a row with the first $M+1$ coefficients of $\mathbf{h}_i$. Writing down the extended Chandrasekhar recursions (Eq. 17b) for the state-space model (Eq. 27a) we obtain (as anticipated in Table 4) the following square-root algorithm: start with $\hat{\mathbf{x}}_0 = \bar{\mathbf{x}}_0 = \overline{\mathbf{w}}$, $\mathbf{P}_0 = \lambda^{-1}\Pi_0$, $r_e^{1/2}(0) = [1 + \mathbf{h}_0\mathbf{P}_0\mathbf{h}_0^*]^{1/2}$, $\mathbf{L}_0$ and $\mathbf{S}_0$ as above, and repeat for $i \geq 0$,

$$\begin{bmatrix} r_e^{1/2}(i) & \tilde{\mathbf{h}}_{i+1}\tilde{\mathbf{L}}_i \\ \begin{bmatrix}0\\ \bar{\mathbf{c}}_i\end{bmatrix} & \lambda^{-1/2}\tilde{\mathbf{L}}_i \end{bmatrix} \Theta_i = \begin{bmatrix} r_e^{1/2}(i+1) & 0 \\ \begin{bmatrix}\bar{\mathbf{c}}_{i+1}\\ 0\end{bmatrix} & \tilde{\mathbf{L}}_{i+1} \end{bmatrix} \tag{27b}$$

where Θ_i is any $\mathbf{J} = (1 \oplus \mathbf{S}_0)$–unitary matrix that produces the zero entry on the right hand-side of the above expression. Moreover,

$$\hat{\mathbf{x}}_{i+1} = \lambda^{-1/2}\hat{\mathbf{x}}_i + \begin{bmatrix} \bar{\mathbf{c}}_i \\ 0 \end{bmatrix} \left[r_e^{1/2}(i) \right]^{-1} (y(i) - \mathbf{h}_i \hat{\mathbf{x}}_i) \qquad (27c)$$

These arrays were also derived in [14, 52] by employing an alternative state-space description with constant coefficients. However, our point of view enables certain further deductions. Note that the state-space model (Eq. 27a) is structured with a very special matrix Ψ relating $\mathbf{h}_i$ and $\mathbf{h}_{i+1}$, viz., $\Psi = \mathbf{Z}$. But our previous derivation allows for more general matrices Ψ, which need not be restricted to the shift matrix. Hence, our derivation is equally applicable to more general structures in the input data, other than the conventional shift structure. This readily allows us to derive faster recursions even for more general matrices Ψ, which is a clear manifestation of the generality and strength of the state-space point of view. Such extensions will be discussed elsewhere.

Normalized Fast Transversal Filters

The computational complexity of each step in (Eq. 27b) is $O(\alpha M)$, where the value of α depends on the choice of Π_0, as we further elaborate. Let us first rewrite (Eq. 27b) in terms of standard RLS variables. We already know that $r_e^{1/2}(i) = \gamma^{-*/2}(i)$ and $\bar{\mathbf{c}}_i = \frac{1}{\sqrt{\lambda}} \mathbf{g}_i \gamma^{-*/2}(i)$. It remains to identify $\tilde{\mathbf{h}}_{i+1}$ and $\tilde{\mathbf{L}}_i$. Recall that $\tilde{\mathbf{h}}_{i+1}$ denotes a row with the first M + 1 coefficients of $\mathbf{h}_{i+1}$. It follows that

$$\tilde{\mathbf{h}}_{i+1} = [u(i+1)\ \ u(i)\ \ u(i-1)\ \ \ldots\ \ u(i+2-M)\ \ u(i+1-M)]$$

The first M entries of $\tilde{\mathbf{h}}_{i+1}$ are precisely those of $\mathbf{u}_{i+1}$, while its last M entries are those of $\mathbf{u}_i$. We can thus partition $\tilde{\mathbf{h}}_{i+1}$ in either of the following two forms:

$$\tilde{\mathbf{h}}_{i+1} = \begin{bmatrix} \mathbf{u}_{i+1} & u(i+1-M) \end{bmatrix} = [u(i+1)\ \ \mathbf{u}_i] \qquad (28)$$

As for the matrix $\mathbf{L}_{i+1}$, we need to determine $\mathbf{L}_0$. For this purpose, we focus now, for simplicity, on the so-called *prewindowed* case where it is assumed that no input data is available prior to and including time $i = 0$ (other cases can be found in [52]). That is, $u(i) = 0$ for $i \le 0$. In this case, we get

$$\mathbf{k}_0 = \lambda^{-1/2}\mathbf{P}_0\mathbf{h}_0^* = \mathbf{0}$$

Therefore, $\bar{\mathbf{k}}_{p,0} = 0$ and $\mathbf{P}_1 = \lambda^{-1}\mathbf{P}_0$, which leads to

$$\mathbf{P}_1 - \mathbf{Z}\mathbf{P}_0\mathbf{Z}^* = \lambda^{-1}\left\{ \lambda^{-1} \begin{bmatrix} \Pi_0 & \mathbf{0} \\ \mathbf{0} & 0 \end{bmatrix} - \mathbf{Z} \begin{bmatrix} \Pi_0 & \mathbf{0} \\ \mathbf{0} & 0 \end{bmatrix} - \mathbf{Z}^* \right\}$$

It is thus clear that different choices for Π_0 lead to different values for α, the rank of $\mathbf{P}_1 - \mathbf{Z}\mathbf{P}_0\mathbf{Z}^*$. A simple choice here is to let

$$\Pi_0 = diagonal\ \{\lambda^2, \lambda^3, \ldots, \lambda^{M+1}\}$$

This leads to a rank-two difference matrix,

$$\mathbf{P}_1 - \mathbf{Z}\mathbf{P}_0\mathbf{Z}^* = \begin{bmatrix} 1 & & \\ & \mathbf{0} & \\ & & -\lambda^M \\ & & & \mathbf{0} \end{bmatrix}$$

which then implies that we can choose (compare with [1, page 600])

$$\tilde{\mathbf{L}}_0 = \begin{bmatrix} 1 & 0 \\ \mathbf{0} & \mathbf{0} \\ 0 & \lambda^{M/2} \end{bmatrix}, \quad \mathbf{S}_0 = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$$

Algorithm: *Consider a set of* $(N+1)$ *data* $\{\mathbf{u}_i, d(i)\}_{i=0}^{N}$, *where the* $\mathbf{u}_i$ *are* $1 \times M$ *row vectors and the* $d(i)$ *are scalars assumed equal to zero for* $i \le 0$. *Consider also an* $M \times 1$ *column vector* $\bar{\mathbf{w}}$, *an* $M \times M$ *positive-definite matrix,*

$$\Pi_0 = diagonal\ \{\lambda^2, \lambda^3, \ldots, \lambda^{M+1}\}$$

where $0 << \lambda \le 1$. *The solution of the minimization problem*

$$\min_{\mathbf{w}} \left[(\mathbf{w}-\bar{\mathbf{w}})^* \left[\lambda^{-(N+1)}\Pi_0 \right]^{-1} (\mathbf{w}-\bar{\mathbf{w}}) + \sum_{i=0}^{N} \lambda^{N-i} |d(i) - \mathbf{u}_i\mathbf{w}|^2 \right]$$

can be recursively computed as follows: start with $\mathbf{w}_{-1} = \bar{\mathbf{w}}$, $\gamma^{-*/2}(0) = 1$, $\tilde{\mathbf{h}}_1 = [u(1)\ \ \mathbf{0}]$, $\mathbf{g}_0 = \mathbf{0}$,

$$\tilde{\mathbf{L}}_0 = \begin{bmatrix} 1 & 0 \\ \mathbf{0} & \mathbf{0} \\ 0 & \lambda^{M/2} \end{bmatrix}, \quad \mathbf{S}_0 = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$$

and repeat for $i \ge 0$

$$\begin{bmatrix} \gamma^{-*/2}(i) & [\,u(i+1)\ \ \mathbf{u}_i\,]\tilde{\mathbf{L}}_i \\ \begin{bmatrix} 0 \\ \mathbf{g}_i\gamma^{-*/2}(i) \end{bmatrix} & \tilde{\mathbf{L}}_i \end{bmatrix} \Theta_i$$

$$= \begin{bmatrix} \gamma^{-*/2}(i+1) & 0 \\ \begin{bmatrix} \mathbf{g}_{i+1}\gamma^{-*/2}(i+1) \\ 0 \end{bmatrix} & \sqrt{\lambda}\,\tilde{\mathbf{L}}_{i+1} \end{bmatrix}$$

where Θ_i *is any* $J = (1 \oplus \mathbf{S}_0)$ *– unitary matrix that produces the zero entry on the right hand-side of the above expression, and* $\tilde{\mathbf{L}}_i$ *is a two-column matrix. Moreover,*

$$\mathbf{w}_i = \mathbf{w}_{i-1} + \left[\mathbf{g}_i\gamma^{-*/2}(i)\right]\left[\gamma^{-*/2}(i)\right]^{-1}\,[d(i) - \mathbf{u}_i\mathbf{w}_{i-1}]$$

The optimal solution is $\mathbf{w} = \mathbf{w}_N$, *and the computational complexity is* $O(M)$ *per iteration.*

The above recursions represent a square-root version of fast RLS algorithms known as FTF [12] (Fast Transversal Filter) and FAEST [17] (Fast Aposteriori Error Sequential Technique). We may proceed to write down an explicit expression for the rotation Θ_i ; thus leading to an explicit set of recursions that relate the different quantities in the arrays. But a distinctive feature of a square-root formulation as above is that the rotation matrix Θ_i need not be explicitly formed and moreover it can be implemented in a variety of ways, as explained before. Different choices for Θ_i correspond to different procedures for achieving the desired zero pattern in the postarray. Each choice would then lead to a different algorithm. The discussion in [52] and [21] [pp. 254–262] is a vivid example of how explicit rewriting of the recursions can lead to several variants. We can not pretend here to be able to detail all the possible implementations of these rotations. But we instead stress the general theme and the general structure of our descriptions: we take a prearray of numbers and rotate it, in one of several possible ways, to obtain a desired pattern of zeros in the postarray. The same theme will arise again when we derive the adaptive lattice filters: a general square-root array is derived, and then several explicit relations are deduced from the array in later sections, thus leading to other forms of lattice filters.

Fast Transversal Filters in Explicit Form

To illustrate the procedure, we shall show how to write down one set of explicit Chandrasekhar recursions by invoking (Eq. 18).

Using the same diagonal matrix Π_0 as in the previous section, we introduce the factorization

$$\mathbf{P}_1 - \mathbf{Z}\mathbf{P}_0\mathbf{Z}^* = -\mathbf{L}_0^{(u)}\mathbf{R}_{r,0}^{-1}\mathbf{L}_0^{*(u)}$$

$$= -\begin{bmatrix}1 & 0\\ \mathbf{0} & \mathbf{0}\\ 0 & 1\\ \mathbf{0} & \mathbf{0}\end{bmatrix}\begin{bmatrix}-1 & 0\\ 0 & \lambda^{-M}\end{bmatrix}^{-1}\begin{bmatrix}1 & 0\\ \mathbf{0} & \mathbf{0}\\ 0 & 1\\ \mathbf{0} & \mathbf{0}\end{bmatrix}^{\mathbf{T}}$$

with the identifications (compare with [1 page 600])

$$\tilde{\mathbf{L}}_0^{(u)} = \begin{bmatrix}1 & 0\\ \mathbf{0} & \mathbf{0}\\ 0 & 1\end{bmatrix},\ \mathbf{R}_{r,0} = \begin{bmatrix}-1 & 0\\ 0 & \lambda^{-M}\end{bmatrix}$$

In terms of the original RLS variables we can then show the following.

Algorithm: *Consider a set of* (N+1) *data* $\{\mathbf{u}_i, d(i)\}_{i=0}^{N}$, *where the* $\mathbf{u}_i$ *are* $1 \times M$ *row vectors and the* $d(i)$ *are scalars assumed equal to zero for* $i \le 0$. *Consider also an* $M \times 1$ *column vector* $\overline{\mathbf{w}}$, *an* $M \times M$ *positive-definite matrix,*

$$\Pi_0 = \textit{diagonal}\ \{\lambda^2, \lambda^3, \ldots, \lambda^{M+1}\}$$

where $0 << \lambda \le 1$. *The solution of the minimization problem*

$$\min_{\mathbf{w}}\left[(\mathbf{w}-\overline{\mathbf{w}})^*\left[\lambda^{-(N+1)}\Pi_0\right]^{-1}(\mathbf{w}-\overline{\mathbf{w}}) + \sum_{i=0}^{N}\lambda^{N-i}|d(i)-\mathbf{u}_i\mathbf{w}|^2\right]$$

can be recursively computed as follows: start with $\mathbf{w}_{-1} = \overline{\mathbf{w}}, \gamma^{-1}(0) = 1, \tilde{\mathbf{h}}_1 = [\,u(1)\ \ 0\,]$, $\mathbf{g}_0 = 0$,

$$\tilde{\mathbf{L}}_0^{(u)} = \begin{bmatrix}1 & 0\\ \mathbf{0} & \mathbf{0}\\ 0 & 1\end{bmatrix},\ \mathbf{R}_{r,0} = \begin{bmatrix}-1 & 0\\ 0 & \lambda^{-M}\end{bmatrix}$$

and repeat for $i \ge 0$

$$\begin{bmatrix}\gamma^{-1}(i) & [u(i+1)\ \mathbf{u}_i]\tilde{\mathbf{L}}_i^{(u)}\\ \begin{bmatrix}0\\ \mathbf{g}_i\gamma^{-1}(i)\end{bmatrix} & \tilde{\mathbf{L}}_i^{(u)}\\ \tilde{\mathbf{L}}_i^{*(u)}\begin{bmatrix}u^*(i+1)\\ \mathbf{u}_i^*\end{bmatrix} & \mathbf{R}_{r,i}\end{bmatrix}\Sigma_i$$

$$= \begin{bmatrix}\gamma^{-1}(i+1) & \mathbf{0}\\ \begin{bmatrix}\mathbf{g}_{i+1}\gamma^{-1}(i+1)\\ 0\end{bmatrix} & \sqrt{\lambda}\,\tilde{\mathbf{L}}_{i+1}^{(u)}\\ \mathbf{0} & \mathbf{R}_{r,i+1}\end{bmatrix}$$

where

$$\Sigma_i = \begin{bmatrix}1 & -\gamma(i)\,[\,u(i+1)\ \ \mathbf{u}_i\,]\,\tilde{\mathbf{L}}_i^{(u)}\\ -\mathbf{R}_{r,i}^{-1}\tilde{\mathbf{L}}_i^{*(u)}\begin{bmatrix}u^*(i+1)\\ \mathbf{u}^*{}_i\end{bmatrix} & \mathbf{I}_2\end{bmatrix}$$

and $\tilde{\mathbf{L}}_i^{(u)}$ *is a two-column matrix. Moreover,*

$$\mathbf{w}_i = \mathbf{w}_{i-1} + \left[\mathbf{g}_i\gamma^{-1}(i)\right]\left[\gamma^{-1}(i)\right]^{-1}\,[d(i) - \mathbf{u}_i\mathbf{w}_{i-1}]$$

The optimal solution is $\mathbf{w} = \mathbf{w}_N$, *and the computational complexity is* $O(M)$ *per iteration.*

Forward and Backward Prediction Filters; the Conversion Factor

The Chandrasekhar recursions bring up important and interesting connections with certain forward and backward prediction problems, which we briefly explore here to show how

some useful facts in recursive least-squares theory (see, e.g., [1, p. 573–581].) fall out rather immediately from simple state-space arguments. [This section may be skipped on a first reading.].

An interesting point to notice is that the Chandrasekhar recursions (Eq. 27b) (and more generally (Eq. 17b) are only functions of the matrices that characterize the state-space description, viz., $\{\mathbf{F}_i,\mathbf{G}_i,\mathbf{H}_i,\mathbf{L}_0,\Pi_i\,.\}$. They do not depend on the observed data, which appear only in the update equation for the state-estimate, e.g., (Eq. 27c). This means that if we have two state-space models with the same matrices $\{\mathbf{F}_i,\mathbf{G}_i,\mathbf{H}_i,\mathbf{L}_0,\Pi_0\}$, but with different state-vectors and observed data, then the same Chandrasekhar recursions can be used to propagate the quantities needed to update the state-estimates for both models.

This motivates us to introduce two prediction problems (the terminology of forward and backward prediction problems is further explored in a later section.). Assume we are interested in solving the following minimization problem (compare with (Eq. 19b))

$$\min_{\mathbf{w}_M^f}\begin{bmatrix}(\mathbf{w}_M^f-\overline{\mathbf{w}}_M^f)^*\left[\lambda^{-(N+1)}\Pi_0\right]^{-1}(\mathbf{w}_M^f-\overline{\mathbf{w}}_M^f)\\ +\sum_{i=0}^{N}\lambda^{N-i}|u(i+1)-\mathbf{u}_i\mathbf{w}_M^f|^2\end{bmatrix} \tag{29a}$$

where, comparing with (Eq. 19b), we have replaced $d(i)$ by $u(i+1)$. The resulting solution, $\mathbf{w}_{M,N}^f$, will provide us with an Mth order forward predictor for $u(N+1)$ in the sense that $\mathbf{u}_N\mathbf{w}_{M,N}^f$ will serve as an estimate for $u(N+1)$ in terms of the last M inputs $\{u(N), u(N-1),..., u(N-M+1)\}$ that are present in $\mathbf{u}_N$. The exact same arguments as in earlier sections will then show that this problem can be solved by considering the following auxiliary state-space model

$$\begin{aligned}\mathbf{x}_{i+1}^f &= \lambda^{-1/2}\mathbf{x}_i^f\\ \frac{u(i+1)}{(\sqrt{\lambda})^i} &= \mathbf{h}_i\mathbf{x}_i^f+v^f(i)\end{aligned} \tag{29b}$$

with

$$\mathbf{x}_0^f=\begin{bmatrix}\mathbf{w}_M^f\\ \mathbf{0}\end{bmatrix},\hat{\mathbf{x}}_0=\begin{bmatrix}\overline{\mathbf{w}}_M^f\\ \mathbf{0}\end{bmatrix},\ Ev^f(i)v^{*f}(j)=\delta_{ij}\,,$$

$$\mathrm{cov}(\mathbf{x}_0^f)=\begin{bmatrix}\lambda^{-1}\Pi_0 & \mathbf{0}\\ \mathbf{0} & \mathbf{0}\end{bmatrix}$$

which has the same structure as the model we set up earlier in (Eq. 27a). The update of the state-estimate then leads to a time-update for the forward predictor vector: $\mathbf{w}_{M,N-1}^f\rightarrow\mathbf{w}_{M,N}^f$, and to the apriori forward prediction error at time $(N+1)$, $\left[u(N+1)-\mathbf{u}_N\mathbf{w}_{M,N-1}^f\right]$ (i.e., based on the data up to time $(N-1)$). This can also be converted to the aposteriori prediction error $\left[u(N+1)-\mathbf{u}_N\mathbf{w}_{M,N}^f\right]$ via the conversion factor $r_e^{-f}(N)$, which we shall denote by $\gamma_M^f(N)$ (this is to be consistent with later notation – the superscript f will in fact be dropped shortly).

Similarly, we can set up a backward prediction problem of order M by solving the minimization problem

$$\min_{\mathbf{w}_M^b}\begin{bmatrix}(\mathbf{w}_M^b-\overline{\mathbf{w}}_M^b)^*\left[\lambda^{-(N+1)}\Pi_0\right]^{-1}(\mathbf{w}_M^b-\overline{\mathbf{w}}_M^b)\\ +\sum_{i=0}^{N}\lambda^{N-i}|u(i-M)-\mathbf{u}_i\mathbf{w}_M^b|^2\end{bmatrix} \tag{30a}$$

where, comparing with (Eq. 19b), we have replaced $d(i)$ by $u(i-M)$. The resulting solution, $\mathbf{w}_{M,N}^b$, will provide us with an M^{th} order backward predictor for $u(i-M)$, in the sense that $\mathbf{u}_i\mathbf{w}_{M,N}^b$ will serve as an estimate for $u(N-M)$ in terms of the future M inputs $\{u(N), u(N-1),..., u(N-M+1)\}$ that are present in $\mathbf{u}_N$. This can also be solved by considering the following auxiliary state-space model

$$\begin{aligned}\mathbf{x}_{i+1}^b &= \lambda^{-1/2}\mathbf{x}_i^b\\ \frac{u(i-M)}{(\sqrt{\lambda})^i} &= \mathbf{h}_i\mathbf{x}_i^b+v^b(i)\,,\end{aligned} \tag{30b}$$

with

$$\mathbf{x}_0^b=\begin{bmatrix}\mathbf{w}_M^b\\ \mathbf{0}\end{bmatrix},\bar{\mathbf{x}}_0=\begin{bmatrix}\overline{\mathbf{w}}_M^b\\ \mathbf{0}\end{bmatrix},\ Ev^b(i)v^{*b}(j)=\delta_{ij}\,,$$

$$\mathrm{cov}(\mathbf{x}_0^b)=\begin{bmatrix}\lambda^{-1}\Pi_0 & \mathbf{0}\\ \mathbf{0} & \mathbf{0}\end{bmatrix},$$

which again has the same structure as the model we set up earlier in (Eq. 27a) as well as in (Eq. 29b). The update of the state-estimate then leads to a time-update for the backward predictor vector: $\mathbf{w}_{M,N-1}^b\rightarrow\mathbf{w}_{M,N}^b$, and to the apriori backward prediction error at time N, $\left[u(N-M)-\mathbf{u}_N\mathbf{w}_{M,N-1}^b\right]$ (i.e., based on the data up to time $(N-1)$). This can also be converted to the aposteriori prediction error $\left[u(N-M)-\mathbf{u}_N\mathbf{w}_{M,N}^b\right]$ via the conversion factor $r_e^{-b}(N)$, which we shall denote by $\gamma_M^b(N)$ (the superscript ^{b} will also be dropped shortly).

But note that the values of the Riccati variables and the innovation covariances for both state-space models (Eq. 29b) and (Eq. 30b) are identical, since the $\mathbf{h}_i$ and the initial condition $\lambda^{-1}\,\Pi_0$ are the same. Therefore, the same Chandrasekhar recursions can be used to propagate the gain matrices that are necessary for the time-update of the prediction vectors. Consequently, the gain vectors in the following updates are the same, say $\mathbf{g}_{M,N}$

$$\mathbf{w}^f_{M,N} = \mathbf{w}^f_{M,N-1} + \mathbf{g}_{M,N}\left[u(N+1) - \mathbf{u}_N\mathbf{w}^f_{M,N-1}\right]$$
$$\mathbf{w}^b_{M,N} = \mathbf{w}^b_{M,N-1} + \mathbf{g}_{M,N}\left[u(N-M) - \mathbf{u}_N\mathbf{w}^b_{M,N-1}\right]$$

These two equations are often written at different time instants as (see, e.g., [1, p. 573, p.576])

$$\mathbf{w}^f_{M,N-1} = \mathbf{w}^f_{M,N-2} + \mathbf{g}_{M,N-1}\left[u(N) - \mathbf{u}_{N-1}\mathbf{w}^f_{M,N-2}\right]$$
$$\mathbf{w}^b_{M,N} = \mathbf{w}^b_{M,N-1} + \mathbf{g}_{M,N}\left[u(N-M) - \mathbf{u}_N\mathbf{w}^b_{M,N-1}\right]$$

Moreover, the conversion factors for both the forward and the backward prediction problems coincide since $r^f_e(N) = r^b_e(N)$. Hence,

$$\frac{u(N+1) - \mathbf{u}_N\mathbf{w}^f_{M,N}}{u(N+1) - \mathbf{w}^f_{M,N-1}} = \frac{u(N-M) - \mathbf{u}_N\mathbf{w}^b_{M,N}}{u(N-M) - \mathbf{u}_N\mathbf{w}^b_{M,N-1}}$$

or, equivalently,

$$\gamma^f_M(N) = \gamma^b_M(N) \tag{31}$$

We can, therefore, drop the superscripts f and b from the conversion factors and just write $\gamma_M(N)$. These are important relations in recursive least-squares theory (see, e.g., [1, p. 580 – 581]).

Different Classes of Order-Recursive Adaptive Algorithms

The adaptive algorithms derived so far are fixed-order solutions of (Eq. 19b) in the sense that they recursively evaluate weight estimates $\mathbf{w}_i$ that correspond to a fixed-order combiner, say of order M. In other words, the size of w, and of its successive estimates $\mathbf{w}_i$, was fixed and equal to M all through the recursions. From now on we shall be dealing with weight-vectors of varying dimensions. We shall therefore write $\mathbf{w}_M$ instead of $\mathbf{w}$ to indicate a weight vector of size $M \times 1$, and we shall write $\mathbf{w}_{M,i}$ instead of $\mathbf{w}_i$ to indicate an estimate at time i of the weight vector $\mathbf{w}_M$ of size M. Apart from these notational inconveniences, the central theme will still prevail. We shall also assume here, for simplicity and ease of exposition, that the weighting matrix Π_0 in (Eq. 19b) is very large, i.e., $\Pi_0 \rightarrow \infty\mathbf{I}$. This simplifies (Eq. 19b) to

$$\min_{\mathbf{w}} \sum_{i=0}^{N} \lambda^{N-i} |d(i) - \mathbf{u}_i\mathbf{w}|^2$$

But we shall later include finite Π_0 weighting factors into the minimization of the resulting forward and backward prediction problems.

So assume that we wish to increase the filter order, say from M to $M+1$, while still processing only the original $(N+1)$ data $\{\mathbf{u}_i, d(i)\}_{i=0}^{N}$. In other words, suppose that instead of solving

$$\min_{\mathbf{w}_M}\left[\sum_{i=0}^{N} \lambda^{N-i} |d(i) - \mathbf{u}_{M,i}\mathbf{w}_M|^2\right] \tag{32a}$$

with

$$\mathbf{u}_{M,i} = [u(i)\ \ u(i-1)\ \ldots\ u(i-M+1)]$$

we are now interested in solving

$$\min_{\mathbf{w}_{M+1}}\left[\sum_{i=0}^{N} \lambda^{N-i} |d(i) - \mathbf{u}_{M+1,i}\mathbf{w}_{M+1}|^2\right] \tag{32b}$$

with

$$\mathbf{u}_{M+1,i} = [u(i)\ \ u(i-1)\ \ldots\ u(i-M+1)\ u(i-M)\ ]$$

Let $\mathbf{w}_{M,N}$ and $\mathbf{w}_{M+1,N}$ denote the corresponding solutions. The adaptive algorithms of the previous sections give an explicit recursive (or time-update) relation between $\mathbf{w}_{M,N}$ and $\mathbf{w}_{M,N-1}$. We are now interested in a recursive (or order-update) relation between $\mathbf{w}_{M,N}$ and $\mathbf{w}_{M+1,N}$.

The first M components of $\mathbf{w}_{M+1,N}$ seem to have no simple relation to $\mathbf{w}_{M,N}$. But there is an alternative to the FIR implementation of Figure 6 that allows us to easily carryover the information from previous computations for the order M filter. This is the so-called lattice filter, which is based on the following idea. Suppose that our interest in solving (Eq. 32a) is not to explicitly determine the weight estimate $\mathbf{w}_{M,N}$, but rather to determine estimates for the reference signals $\{d(\cdot)\}$, say

$$\hat{d}_M(N) = \mathbf{u}_{M,N}\mathbf{w}_{M,N} = \text{estimate of } d(N) \text{ of order } M$$

Likewise, for the higher-order problem,

$$\hat{d}_{M+1}(N) = \mathbf{u}_{M+1,N}\mathbf{w}_{M+1,N} = \text{estimate of } d(N) \text{ of order } M+1$$

The lattice solution allows us to update $\hat{d}_M(N)$ to $\hat{d}_{M+1}(N)$ without explicitly computing the weight estimates $\mathbf{w}_{M,N}$ and $\mathbf{w}_{M+1,N}$. This is achieved by orthogonalizing the input data as we explain in the next section.

Joint Process Estimation and Prediction Problems

The major step here is to note that, for all i, the input vectors $\mathbf{u}_{M,i}$ and $\mathbf{u}_{M+1,i}$ have the same first M entries. We shall now argue that this observation allows us to reduce the minimization of (Eq. 32a) to the equivalent problem of solving two *first-order* least-squares minimization problems of the type depicted earlier in Figure 3, and for which the special case of (Eq. 12b) is immediately applicable.

There is no loss of generality in our argument here if we

focus on particular values for M and λ, say $M = 3$ and $\lambda = 1$, since our purpose in this section is to motivate the need for forward and backward residuals in simple terms. We shall then proceed to show in the next two sections how to embed the problem of updating these residuals into the state-space framework; thus deriving several variants of the corresponding lattice algorithms.

So assume we want to solve the following problem: minimize over $\mathbf{w}_3$ the cost function (the data $\{u(\cdot)\}$ is assumed to be zero prior to the initial time instant $i = 0$, viz., $u(i) = 0$ for $i < 0$)

$$\left\| \begin{bmatrix} d(0) \\ d(1) \\ \vdots \\ d(N) \end{bmatrix} - \begin{bmatrix} u(0) & 0 & 0 \\ u(1) & u(0) & 0 \\ \vdots & \vdots & \vdots \\ u(N) & u(N-1) & u(N-2) \end{bmatrix} \begin{bmatrix} w_3(1) \\ w_3(2) \\ w_3(3) \end{bmatrix} \right\|_2^2 \quad (33a)$$

As explained above, we shall denote the optimal solution by $\mathbf{w}_{3,N}$. The subscript $_N$ indicates that it is an estimate based on the data $u(\cdot)$ up to time N. Determining $\mathbf{w}_{3,N}$ corresponds to determining the entries of a 3-dimensional weight vector so as to approximate the column vector $\mathbf{d}_N$ by the linear combination $\mathbf{U}_3\mathbf{w}_{3,N}$ in the least-squares sense (Eq. 33a) where $\mathbf{U}_3$ is the matrix multiplying $\mathbf{w}_3$ in (33a). We thus say that expression (Eq. 33a) defines a third-order estimator for the reference sequence $\{d(\cdot)\}$. The resulting aposteriori estimation errors will be denoted by

$$\begin{bmatrix} \varepsilon_3(0) \\ \varepsilon_3(1) \\ \vdots \\ \varepsilon_3(N) \end{bmatrix} = \begin{bmatrix} d(0) \\ d(1) \\ \vdots \\ d(N) \end{bmatrix} - \begin{bmatrix} u(0) & 0 & 0 \\ u(1) & u(0) & 0 \\ \vdots & \vdots & \vdots \\ u(N) & u(N-1) & u(N-2) \end{bmatrix} \begin{bmatrix} w_{3,N}(1) \\ w_{3,N}(2) \\ w_{3,N}(3) \end{bmatrix}$$

where $\varepsilon_3(i)$ denotes the aposteriori estimation error in estimating $d(i)$ from a linear combination of the 3 most recent inputs,

$$\varepsilon_3(i) = d(i) - \mathbf{u}_{3,i}\mathbf{w}_{3,N}$$

Let $\tilde{\mathbf{d}}_3 = [\varepsilon_3(0) \ldots \varepsilon_3(N)]^{\mathbf{T}}$ and recall from the discussion, prior to expression (8c) that the aposteriori residual vector $\tilde{\mathbf{d}}_3$ has to be orthogonal to the data matrix $\mathbf{U}_3$, viz., $\mathbf{U}_3^*\tilde{\mathbf{d}}_3 = 0$. We also know that the optimal solution $\mathbf{w}_{3,N}$ provides an estimate $\mathbf{U}_3\mathbf{w}_{3,N}$ that is the closest element in the range space of $\mathbf{U}_3$ to the column vector $\mathbf{d}_N$.

Now assume that we wish to solve the next-higher order problem, viz., of order $M = 4$: minimize over $\mathbf{w}_4$ the cost function

$$\left\| \begin{bmatrix} d(0) \\ d(1) \\ \vdots \\ d(N-1) \\ d(N) \end{bmatrix} - \begin{bmatrix} u(0) & 0 & 0 & 0 \\ u(1) & u(0) & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots \\ u(N-1) & u(N-2) & u(N-3) & u(N-4) \\ u(N) & u(N-1) & u(N-2) & u(N-3) \end{bmatrix} \begin{bmatrix} w_4(1) \\ w_4(2) \\ w_4(3) \\ w_4(4) \end{bmatrix} \right\|_2^2 \quad (33b)$$

This statement is very close to (Eq. 33a) except for an extra column in the data matrix $\mathbf{U}_4$ that multiplies $\mathbf{w}_4$: the first three columns of $\mathbf{U}_4$ coincide with those of $\mathbf{U}_3$, while the last column of $\mathbf{U}_4$ contains the extra new data that are needed for a fourth-order estimator,

$$\mathbf{U}_4 = \begin{bmatrix} & 0 \\ & 0 \\ \mathbf{U}_3 & \cdots \\ & u(N-4) \\ & u(N-3) \end{bmatrix}$$

The problem in (Eq. 33b) requires us to linearly combine the four columns of $\mathbf{U}_4$ in order to compute the fourth-order estimates of $\{d(0), d(1), \ldots, d(N)\}$. In other words, it requires us to determine the closest element in the range space of $\mathbf{U}_4$ to the same column vector $\mathbf{d}_N$. We already know what is the closest element to $\mathbf{d}_N$ in the range of $\mathbf{U}_3$, which is a submatrix of $\mathbf{U}_4$. This suggests that we should try to decompose the range space of $\mathbf{U}_4$ into two orthogonal subspaces, viz.,

$$\text{Range}(\mathbf{U}_4) = \text{Range}(\mathbf{U}_3) \oplus \text{Range}(\mathbf{m})$$

where $\mathbf{m}$ is a column vector that is orthogonal to $\mathbf{U}_3$, $\mathbf{U}_3^*\mathbf{m} = 0$. The notation Range $(\mathbf{U}_3) \oplus$ Range$(\mathbf{m})$ also means that every element in the range space of $\mathbf{U}_4$ can be expressed as a linear combination of the columns of $\mathbf{U}_3$ and $\mathbf{m}$.

But this decomposition can be easily accomplished by projecting the last column of $\mathbf{U}_4$ onto the range space of its first three columns and keeping the residual vector as $\mathbf{m}$. This is precisely a Gram-Schmidt orthogonalization step and it is equivalent to the following minimization problem: minimize over $\mathbf{w}_3^b$

$$\left\| \begin{bmatrix} 0 \\ 0 \\ \vdots \\ u(N-4) \\ u(N-3) \end{bmatrix} - \begin{bmatrix} u(0) & 0 & 0 \\ u(1) & u(0) & 0 \\ \vdots & \vdots & \vdots \\ u(N-1) & u(N-2) & u(N-3) \\ u(N) & u(N-1) & u(N-2) \end{bmatrix} \begin{bmatrix} w_3^b(1) \\ w_3^b(2) \\ w_3^b(3) \end{bmatrix} \right\|_2^2 \quad (34a)$$

This is also a special case of (Eq. 32a) where we have replaced the sequence $\{d(0), \ldots, d(N)\}$ by $\{0,0,0,u(0), \ldots, u(N-4), u(N-3)\}$. We shall denote the optimal solution by $\mathbf{w}_{3,N}^b$. The subscript N indicates that it is an estimate based on the data $u(\cdot)$ up to time N. Determining $\mathbf{w}_{3,N}^b$ corresponds to determining the entries of a 3-dimensional weight vector so as to approximate the last column of $\mathbf{U}_4$ by a linear combination of the columns of $\mathbf{U}_3$, viz., $\mathbf{U}_3\mathbf{w}_{3,N}^b$, in the least-squares sense. Note that the entries in every row of the data matrix $\mathbf{U}_3$ are the three "future" values corresponding to the entry in the last column of $\mathbf{U}_4$. Hence, the last element of the above

linear combination serves as a *backward* prediction of $u(N-3)$ in terms of $\{u(N),u(N-1),u(N-2)\}$. A similar remark holds for the other entries. The superscript b stands for *backward*. We thus say that expression (Eq. 34a) defines a third-order backward prediction problem. The resulting backward *aposteriori* prediction errors will be denoted by

$$\begin{bmatrix} b_3(0) \\ b_3(1) \\ \vdots \\ b_3(N-1) \\ b_3(N) \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ \vdots \\ u(N-4) \\ u(N-3) \end{bmatrix} - \begin{bmatrix} u(0) & 0 & 0 \\ u(1) & u(0) & 0 \\ \vdots & \vdots & \vdots \\ u(N-1) & u(N-2) & u(N-3) \\ u(N) & u(N-1) & u(N-2) \end{bmatrix} \begin{bmatrix} w^b_{3,N}(1) \\ w^b_{3,N}(2) \\ w^b_{3,N}(3) \end{bmatrix}$$

where $b_3(i)$ denotes the aposteriori backward prediction error in estimating $u(i-3)$ from a linear combination of the future 3 inputs,

$$b_3(i) = u(i-3) - \mathbf{u}_{3,i}\mathbf{w}^b_{3,N}$$

Let $\mathbf{b}_3 = [b_3(0) \ldots b_3(N)]^{\mathbf{T}}$ and recall from the discussion in Section 3.2, prior to expression (Eq. 8c), that the aposteriori backward residual vector $\mathbf{b}_3$ has to be orthogonal to the data matrix $\mathbf{U}_3$, $\mathbf{U}_3^*\mathbf{b}_3 = 0$, which thus implies that it can be chosen as the **m** column that we mentioned earlier, viz.,

$$\text{Range}(\mathbf{U}_4) = \text{range}(\mathbf{U}_3) \oplus \text{Range}(\mathbf{b}_3) \qquad (34b)$$

Our original motivation for introducing the aposteriori backward residual vector $\mathbf{b}_3$ was the desire to solve the fourth-order problem (Eq. 33b), not afresh, but in a way so as to exploit the solution of lower order; thus leading to an order-recursive algorithm.

Assume we have available $\tilde{\mathbf{d}}_3$ and $\mathbf{b}_3$, which are both orthogonal to $\mathbf{U}_3$. Knowing that $\mathbf{b}_3$ leads to an orthogonal decomposition of the range of $\mathbf{U}_4$ as in (Eq. 34b), then updating $\tilde{\mathbf{d}}_3$ into a fourth-order aposteriori residual vector $\tilde{\mathbf{d}}_4$, which has to be orthogonal to $\mathbf{U}_4$, simply corresponds to projecting the vector $\tilde{\mathbf{d}}_3$ onto the vector $\mathbf{b}_3$.

To clarify this, assume we pose the problem of projecting $\tilde{\mathbf{d}}_3$ onto $\mathbf{b}_3$, which is equivalent to asking for a *scalar* coefficient φ_3 so as to minimize

$$\|\tilde{\mathbf{d}}_3 - \varphi_3\mathbf{b}_3\|_2^2$$

This is clearly a standard least-squares problem (recall Eq. 12c). The optimal solution for the scalar φ_3 is given by

$$\hat{\varphi}_3 = \frac{1}{\mathbf{b}_3^*\mathbf{b}_3}\mathbf{b}_3^*\tilde{\mathbf{d}}_3.$$

Let **x** denote the resulting residual vector: $\mathbf{x} = \tilde{\mathbf{d}}_3 - \hat{\varphi}_3\mathbf{b}_3$. It is clearly orthogonal to $\mathbf{b}_3$. The claim is that **x** can be taken as $\tilde{\mathbf{d}}_4$. To verify this we need to show that **x** is orthogonal to $\mathbf{U}_4$, viz., $\mathbf{U}_4^*\mathbf{x} = 0$, or equivalently,

$$\begin{bmatrix} \mathbf{U}_3^* \\ 0\,0 \ldots u^*(N-4)\ u^*(N-3) \end{bmatrix} \mathbf{x} = 0$$

Now $\mathbf{U}_3^*\mathbf{x}$ is clearly zero since **x** is a linear combination of $\tilde{\mathbf{d}}_3$ and $\mathbf{b}_3$ and both vectors are orthogonal to $\mathbf{U}_3$. As for the column $[0\ 0\ \ldots\ u(N-4)\ u(N-3)]^{\mathbf{T}}$, it is also orthogonal to **x** because it is in the range space of $\mathbf{U}_4$ and, consequently, it can be expressed as a linear combination of $\mathbf{b}_3$ and the columns of $\mathbf{U}_3$. The vector **x** is orthogonal to both $\mathbf{b}_3$ and $\mathbf{U}_3$.

We thus know how to update $\tilde{\mathbf{d}}_3$ into $\tilde{\mathbf{d}}_4$ by projecting $\tilde{\mathbf{d}}_3$ onto $\mathbf{b}_3$. To proceed with this order update procedure, we need to know how to order-update the backward residual vector as well. That is, we need to know how to go from $\mathbf{b}_3$ to $\mathbf{b}_4$. This can be determined by following similar arguments, which motivates us to introduce the forward prediction problem: minimize over $\mathbf{w}_3^f$ the cost function

$$\left\| \begin{bmatrix} u(1) \\ u(2) \\ \vdots \\ u(N+1) \end{bmatrix} - \begin{bmatrix} u(0) & 0 & 0 \\ u(1) & u(0) & 0 \\ \vdots & \vdots & \vdots \\ u(N) & u(N-1) & u(N-2) \end{bmatrix} \begin{bmatrix} w_3^f(1) \\ w_3^f(2) \\ w_3^f(3) \end{bmatrix} \right\|_2^2 \qquad (35a)$$

As explained before, we shall denote the optimal solution by $\mathbf{w}^f_{3,N}$. The subscript $_N$ indicates that it is an estimate based on the data $u(\cdot)$ up to time N. Determining $\mathbf{w}^f_{3,N}$ corresponds to determining the entries of a 3-dimensional weight vector so as to approximate the column vector

$$\begin{bmatrix} u(1) \\ u(2) \\ \vdots \\ u(N+1) \end{bmatrix}$$

by a linear combination of the columns of $\mathbf{U}_3$, viz., $\mathbf{U}_3\mathbf{w}^f_{3,N}$. Note that the entries of the successive rows of the data matrix $\mathbf{U}_3$ are the past three inputs relative to the corresponding entries of the column vector. Hence, the last element of the linear combination $\mathbf{U}_3\mathbf{w}^f_{3,N}$ serves as a *forward* prediction of $u(N+1)$ in terms of $\{u(N), u(N-1), u(N-2)\}$. A similar remark holds for the other entries. The superscript f stands for *forward*. We thus say that expression (Eq. 35a) defines a third-order forward prediction problem. The resulting *aposteriori* forward prediction errors will be denoted by

$$\begin{bmatrix} f_3(1) \\ f_3(2) \\ \vdots \\ f_3(N+1) \end{bmatrix} = \begin{bmatrix} u(1) \\ u(2) \\ \vdots \\ u(N+1) \end{bmatrix} - \begin{bmatrix} u(0) & 0 & 0 \\ u(1) & u(0) & 0 \\ \vdots & \vdots & \vdots \\ u(N) & u(N-1) & u(N-2) \end{bmatrix} \begin{bmatrix} w_3^f(1) \\ w_3^f(2) \\ w_3^f(3) \end{bmatrix}$$

where $f_3(i)$ denotes the aposteriori forward prediction error in estimating $u(i)$ from a linear combination of the past 3 inputs,

$$f_3(i) = u(i) - \mathbf{u}_{3,i}\mathbf{w}_{3,N}^f$$

Now assume that we wish to solve the next-higher order problem, viz., of order $M = 4$: minimize over $\mathbf{w}_4^f$ the cost function

$$\left\| \begin{bmatrix} u(1) \\ u(2) \\ \vdots \\ u(N+1) \end{bmatrix} - \begin{bmatrix} u(0) & 0 & 0 & 0 \\ u(1) & u(0) & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots \\ u(N) & u(N-1) & u(N-2) & u(N-3) \end{bmatrix} \begin{bmatrix} w_4^f(1) \\ w_4^f(2) \\ w_4^f(3) \\ w_4^f(4) \end{bmatrix} \right\|_2^2 \tag{35b}$$

We again observe that this statement is very close to (Eq. 35a) except for an extra column in the data matrix $\mathbf{U}_4$, in precisely the same way as happened with $\tilde{\mathbf{d}}_4$ and $\mathbf{b}_3$. We can thus obtain $\mathbf{f}_4$ by projecting $\mathbf{f}_3$ onto $\mathbf{b}_3$ and taking the residual vector as $\mathbf{f}_4$,

$$\min_{\eta_3} \|\mathbf{f}_3 - \eta_3 \mathbf{b}_3\|_2^2$$

This is clearly a standard least-squares problem. The optimal solution for the scalar η_3 is given by

$$\hat{\eta}_3 = \frac{1}{\mathbf{b}_3^* \mathbf{b}_3} \mathbf{b}_3^* \mathbf{f}_3$$

and $\mathbf{f}_4 = \mathbf{f}_3 - \hat{\eta}_3 \mathbf{b}_3$. Also, using the orthogonality condition we can easily relate the norm of the resulting error vector $\mathbf{f}_4$ to that of the previous error vector $\mathbf{f}_3$. Indeed,

$$\begin{aligned} \|\mathbf{f}_4\|_2^2 &= \mathbf{f}_4^* \mathbf{f}_4 = \mathbf{f}_4^* [\mathbf{f}_3 - \hat{\eta}_3 \mathbf{b}_3] \\ &= \mathbf{f}_4^* \mathbf{f}_3 \quad \textit{since} \mathbf{f}_4 \textit{ is orthogonal to } \mathbf{b}_3 \\ &= \mathbf{f}_3^* \mathbf{f}_3 - \hat{\eta}_3 \mathbf{f}_3^* \mathbf{b}_3 \\ &= \|\mathbf{f}_3\|_2^2 - \frac{|\mathbf{f}_3^* \mathbf{b}_3|^2}{\|\mathbf{b}^3\|_2^2} \end{aligned} \tag{36a}$$

where in the last equality we used the expression for $\hat{\eta}_3$. It is thus clear that the forward residual error decreases in norm with increasing prediction orders. Similarly, the backward residual vector $\mathbf{b}_3$ can be updated to $\mathbf{b}_4$ by projecting $\mathbf{b}_3$ onto $\mathbf{f}_3$,

$$\min_{\xi_3} \|\mathbf{b}_3 - \xi_3 \mathbf{f}_3\|_2^2$$

and we get

$$\|\mathbf{b}_4\|_2^2 = \|\mathbf{b}_3\|_2^2 - \frac{|\mathbf{b}_3^* \mathbf{f}_3|^2}{\|\mathbf{f}_3\|_2^2} \tag{36b}$$

Two First-Order Linear Combiners

We argued in the previous section that if we are given the forward and backward residual vectors $\mathbf{f}_3$ and $\mathbf{b}_3$, then the higher-order forward residual $\mathbf{f}_4$ can be obtained by simply setting up a first-order least-squares problem, viz., by projecting $\mathbf{f}_3$ onto $\mathbf{b}_3$. This is equivalent to saying: use the entries of the forward vector $\mathbf{f}_3$ as *reference* signals and the entries of the backward vector $\mathbf{b}_3$ as *input* signals in a first-order combiner, as anticipated earlier in Figure 3. The diagram is repeated in Figure 7 with the appropriate signals.

Likewise, the backward residual vector $\mathbf{b}_3$ can be updated to $\mathbf{b}_4$ by simply projecting $\mathbf{b}_3$ onto $\mathbf{f}_3$. This is again equivalent to the following. Use the entries of the backward vector $\mathbf{b}_3$ as *reference* signals and the entries of the forward vector $\mathbf{f}_3$ as *input* signals in a first-order combiner, as we also anticipated in Figure 3. The diagram is again repeated in Figure 8 with the appropriate signals, where we have denoted the scalar weights in Figures 7 and 8 by η_3 and ξ_3, respectively. It is also clear from the definition of the residual vectors $\mathbf{f}_3$ and $\mathbf{b}_3$ that, in the general case, the initial conditions should be:

$$\mathbf{f}_0 = \begin{bmatrix} u(1) \\ u(2) \\ \vdots \\ u(N+1) \end{bmatrix}, \quad \mathbf{b}_0 = \begin{bmatrix} u(0) \\ u(1) \\ \vdots \\ u(N) \end{bmatrix}$$

That is, $f_0(j) = \mathbf{b}_0(j) = u(j)$ for all $j \geq 0$.

The following statement summarizes the discussion so far, but for general values of M, N and λ. [We shall concentrate, for the time being, on the fundamental problem of updating the forward and backward residuals. The update of the estimation residuals $\{\varepsilon_M(\cdot)\}$ follows immediately and will be considered later.] .

Theorem: *Consider a set of data points* $\{u(i)\}_{i=0}^N$ *and let* $\mathbf{u}_{M,i}$ *denote an* $M \times 1$ *row vector of the form*

$$\mathbf{u}_{M,i} = [u(i) \;\; u(i-1) \ldots u(i-M+1)]$$

Define the aposteriori forward residual vector of order M

$$\mathbf{f}_M = \begin{bmatrix} f_M(1) \\ f_M(2) \\ \vdots \\ f_M(N+1) \end{bmatrix} = \begin{bmatrix} u(1) \\ u(2) \\ \vdots \\ u(N+1) \end{bmatrix} - \begin{bmatrix} \mathbf{u}_{M,0} \\ \mathbf{u}_{M,1} \\ \vdots \\ \mathbf{u}_{M,N} \end{bmatrix} \mathbf{w}_{M,N}^f$$

where $\mathbf{w}_{M,N}^f$ *is the optimal solution of the minimization problem (recall the definition of the weighting matrix* $\Lambda_N^{1/2}$ *in (Eq. 19d))*

$$\min_{\mathbf{w}_M^f} \|\Lambda_N^{1/2} \left[\mathbf{u}^f - \mathbf{U}_M \mathbf{w}_M^f\right]\|_2^2$$

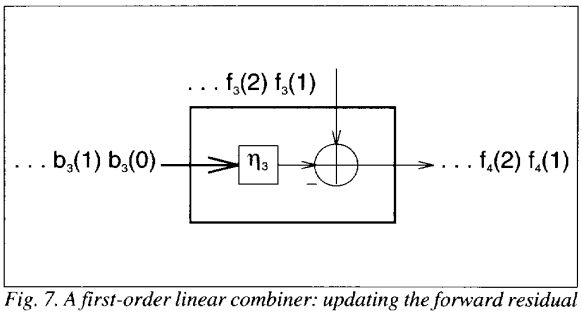

Fig. 7. A first-order linear combiner: updating the forward residual errors.

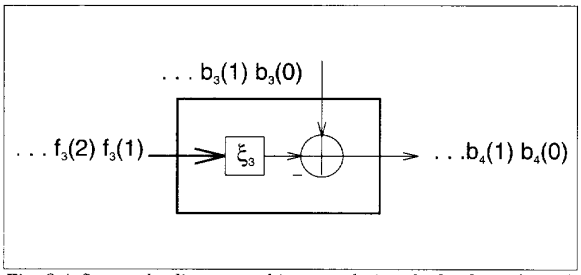

Fig. 8.A first-order linear combiner: updating the backward residual errors.

where $\mathbf{u}^b = [u(1)\ u(2)\ldots u(n+1)]^{\mathbf{T}}$

Define also the aposteriori backward residual vector of order M as

$$\mathbf{b}_M = \begin{bmatrix} b_M(0) \\ \vdots \\ b_M(N-1) \\ b_M(N) \end{bmatrix} = \begin{bmatrix} 0 \\ \vdots \\ u(N-M-1) \\ u(N-M) \end{bmatrix} - \begin{bmatrix} \mathbf{u}_{M,0} \\ \vdots \\ \mathbf{u}_{M,N-1} \\ \mathbf{u}_{M,N} \end{bmatrix} \mathbf{w}^b_{M,N}$$

where $\mathbf{w}^b_{M,N}$ *is the optimal solution of the minimization problem*

$$\min_{\mathbf{w}^b_M} \|\Lambda_N^{1/2}\left[\mathbf{u}^b - \mathbf{U}_M\mathbf{w}^b_M\right]\|_2^2$$

where $\mathbf{u}^b = [0\ \ldots\ u\ (N\text{-}M\text{-}1)\ u(N\text{-}M)]^{\mathbf{T}}$

Then the residual vectors can be order-updated by solving the following two first-order least-squares problems:

$$\mathbf{f}_{M+1} = \mathbf{f}_M - \hat{\eta}_M\mathbf{b}_M\,,\quad \mathbf{b}_{M+1} = \mathbf{b}_M - \hat{\xi}_M\mathbf{f}_M \tag{37a}$$

where $\hat{\eta}_M$ *and* $\hat{\xi}_M$ *are optimal solutions of the following:*

$$\min_{\eta_M} \sum_{j=0}^{N} \lambda^{N-j}\,|f_M(j+1) - \eta_M b_M(j)|^2$$

$$\min_{\xi_M} \sum_{j=0}^{N} \lambda^{N-j}\,|b_M(j-1) - \xi_M f_M(j+1)|^2 \tag{37b}$$

The initial conditions are $f_0(j) = b_0(j) = u(j)$ *for all j.*

More generally, we may replace (Eq. 37b) by the minimization criteria (with $\mu > 0$)

$$\min_{\eta_M}\left[\frac{\lambda^{(N+1)}}{\mu}\,|\eta_M - \bar{\eta}_M|^2 + \sum_{j=0}^{N}\lambda^{N-j}\,|f_M(j+1) - \eta_M b_M(j)|^2\right] \tag{38a}$$

$$\min_{\xi_M}\left[\frac{\lambda^{(N+1)}}{\mu}\,|\xi_M - \bar{\xi}_M|^2 + \sum_{j=0}^{N}\lambda^{N-j}\,|b_M(j) - \xi_M f_M(j+1)|^2\right] \tag{38b}$$

where we have added a positive weighting factor μ and initial guesses $\bar{\eta}_M$ and $\bar{\xi}_M$. The resulting $\hat{\eta}_M$ and $\hat{\xi}_M$ are then used to update the residuals in (Eq. 37a). It is often the case, though, that $\bar{\eta}_M = \bar{\xi}_M = 0$, which will be assumed hereafter, without loss of generality. Also, μ is often chosen as a very large positive number, which reduces (Eq. 38a) and (Eq. 38b) to (Eq. 37c).

Two First-Order State-Space Models

The result stated in the previous theorem reduces the problem of order-updating the prediction residuals to the solution of two first-order least-squares problems: in one case the forward residuals are used as the reference signals and the backward residuals are used as input signals, while in the other case the roles of the residuals are reversed.

This fits nicely into the setting anticipated earlier in Eq. 12b. Indeed, we can set up two first-order state-space models that correspond to the solution of the above order-update problems (Eq. 38a) and (Eq. 38b), in exactly the same way that we argued for Eq. 12b.

Define the state-variables

$$x_1(i) = \frac{\eta_M}{(\sqrt{\lambda})^i}\,,\quad x_2(i) = \frac{\xi_M}{(\sqrt{\lambda})^i}$$

as well as the normalized signals

$$y_1(i) = \frac{f_M(i+1)}{(\sqrt{\lambda})^i}\,,\quad y_2(i) = \frac{b_M(i)}{(\sqrt{\lambda})^i}$$

Then the appropriate models are

$$\begin{aligned} &x_1(i+1) = \lambda^{-1/2}x_1(i) \qquad x_2(i+1) = \lambda^{-1/2}x_2(i) \\ &y_1(i) = b_M(i)x_1(i) + v_1(i),\quad y_2(i) = f_M(i+1)x_2(i) + v_2(i) \end{aligned} \tag{39}$$

with $\bar{x}_1(0) = \bar{x}_2(0) = 0$, $\mathrm{cov}(x_1(0)) = \lambda^{-1}\mu = \mathrm{cov}(x_2(0))$, and $Ev_1(i)v_1^*(j) = Ev_2(i)v_2^*(j) = \delta_{ij}$.

If we now proceed to write down the different variants of

the Kalman recursions that correspond to the above two state-space estimation problems, and then translate the variables to the corresponding quantities in the least-squares setting, we get different variants of adaptive lattice filters. The argument follows precisely what we did earlier. For this reason, we shall try to be brief.

QR-Decomposition-Based Least-Squares Lattice (QRD-LSL) Filters

We start by writing down the extended square-root information form that corresponds to each of the first-order state-space models in (Eq. 39), thus leading to extended QR arrays in exactly the same manner as we did earlier in Eq. 25a. The resulting square-root arrays turn out to be central to the derivation of different variants of adaptive lattice algorithms. Indeed, we shall see in the next three sections that other lattice versions discussed in the literature can be regarded as alternative rewritings of these square-root arrays. [This should come as no surprise since the arrays correspond to the extended square-root information filter form, and this form is clearly equivalent to the other variants of the Kalman filter – more on this further ahead.]

To begin with, the two-column array for the order-update of the forward residuals is (apart from the notational differences, this is a specialization, for the time instant $i = N$ and to the scalar case, of the array (Eq. 25a) that we wrote earlier for a general least-squares problem)

$$\begin{bmatrix} \sqrt{\lambda}\,\Phi_M^{b/2}(N-1) & b_M^*(N) \\ \sqrt{\lambda}\,q_M^{*b}(N-1) & f_M^*(N+1) \\ 0 & 1 \\ \dfrac{1}{\sqrt{\lambda}}\,\Phi_M^{-*b/2}(N-1) & 0 \end{bmatrix} \Theta_{M,N}^b$$

$$= \begin{bmatrix} \Phi_M^{b/2}(N) & 0 \\ q_M^{*b}(N) & \dfrac{f_{M+1}^*(N+1)}{\gamma_{M+1}^{*/2}(N)} \\ b_M(N)\Phi_M^{-*b/2}(N) & \gamma_{M+1}^{1/2}(N) \\ \Phi_M^{-*b/2}(N) & -\bar{g}_M^b(N) \end{bmatrix} \tag{40a}$$

where we have defined the *scalar* "covariance" and "cross-covariance" quantities (cf. (Eq. 20a))

$$\Phi_M^b(N) = \lambda^{(N+1)}\mu^{-1} + \sum_{i=0}^{N} \lambda^{N-j} b_M^*(j) b_M(j),\ \Phi_M^b(-1) = \mu^{-1}$$

$$s_M^b(N) = \sum_{i=0}^{N} \lambda^{N-j} b_M^*(j)\, f_M(j+1),\ s_M^b(-1) = 0$$

$$q_M^b(N) = \frac{s_M^b(N)}{\Phi_M^{b/2}(N)},\ q_M^b(-1) = 0$$

and $\bar{g}_M^b(N)$ denotes the normalized gain $g_M^b(N)\gamma_{M+1}^{-*/2}(N)$.

The second entry of the second row of the postarray is denoted by $f_{M+1}^*(N+1)\gamma_{M+1}^{-*/2}(N)$. We recall from (Eq. 25a) that it should be the product of an apriori error and the square-root of the conversion factor or, equivalently, the product of an aposteriori error and the inverse of the square-root of the conversion factor. We have used this second interpretation to write $f_{M+1}^*(N+1)\gamma_{M+1}^{-*/2}(N)$ since $f_{M+1}^*(N+1)$ denotes the aposteriori error. Also, the term $\gamma_{M+1}(N)$ is used to denote the conversion factor of order $(M+1)$, whose inverse converts the *aposteriori* forward residual $f_{M+1}(N+1)$, of order $(M+1)$ and at time $(N+1)$, to the corresponding *apriori* forward residual, say

$$\alpha_{m+1}(N+1)\gamma_{M+1}(N) = f_{M+1}(N+1) \tag{40b}$$

We could have clearly written $\alpha_{M+1}^*(N+1)\gamma_{M+1}^{1/2}(N)$ instead of $f_{M+1}^*(N+1)\gamma_{M+1}^{-*/2}(N)$. This would have allowed us to express the recursion in terms of *apriori* residuals rather than *aposteriori* residuals, and vice-versa. This explains the origin of existing variants of lattice algorithms that work either with *apriori* or *aposteriori* errors (or combination of both). It is a matter of choice. Two explicit examples to this effect are considered in the next two sections.

Correspondingly, the two-column array for the order-update of the backward residuals is

$$\begin{bmatrix} \sqrt{\lambda}\,\Phi_M^{f/2}(N) & f_M^*(N+1) \\ \sqrt{\lambda}\,q_M^{*f}(N) & b_M^*(N) \\ 0 & 1 \\ \dfrac{1}{\sqrt{\lambda}}\,\Phi_M^{-*f/2}(N) & 0 \end{bmatrix} \Theta_{M,N+1}^f$$

$$= \begin{bmatrix} \Phi_M^{f/2}(N+1) & 0 \\ q_M^{*f}(N+1) & \dfrac{b_{M+1}^*(N+1)}{\gamma_{M+1}^{*/2}(N+1)} \\ f_M(N+1)\Phi_M^{-*f/2}(N+1) & \gamma_{M+1}^{1/2}(N+1) \\ \Phi_M^{-*f/2}(N) & -\bar{g}_M^f(N+1) \end{bmatrix}$$

where we have defined the *scalar* "covariance" and "cross-covariance" quantities (cf. (Eq. 20a))

$$\Phi_M^f(N+1) = \lambda^{(N+1)}\mu^{-1} + \sum_{j=1}^{N+1} \lambda^{N+1-j} f_M^*(j) f_M(j), \Phi_M^f(0) = \mu^{-1}$$

$$s_M^b(N+1) = \sum_{j=1}^{N+1} \lambda^{N+1-j} f_M^*(j) b_M(j-1), s_M^f(0) = 0$$

$$q_M^f(N+1) = \frac{s_M^f(N+1)}{\Phi_M^{f/2}(N+1)}, \quad q_M^f(0) = 0$$

and $\bar{g}_M^f(N+1)$ denotes the normalized gain $g_M^f(N+1)\gamma_{M+1}^{-*/2}(N+1)$.

Also, the term $\gamma_{M+1}(N+1)$ is used to denote the conversion factor, whose inverse converts the aposteriori backward residual $b_{M+1}(N+1)$, of order $M+1$ and at time $(N+1)$, to the corresponding *apriori* backward residual, say (recall (31))

$$\beta_{m+1}(N+1)\gamma_{M+1}(N+1) = b_{M+1}(N+1).$$

The normalized residuals $f_{M+1}^*(N+1)\gamma_{M+1}^{-*/2}(N)$ and $b_{M+1}^*(N+1)\gamma_{M+1}^{-*/2}(N+1)$ are often called *angle-normalized* residuals and will be denoted by $\bar{f}_{M+1}^*(N+1)$ and $\bar{b}_{M+1}^*(N+1)$, respectively. A justification follows by observing that, in the case of real data, the 2×2 rotation matrices can be expressed in terms of sine and cosine parameters, say

$$\Theta_{M,N}^b = \begin{bmatrix} c & s \\ -s & c \end{bmatrix}$$

Then the third line of the array (Eq. 40a) shows that the square-root of $\gamma_{M+1}(N)$ is equal to the cosine parameter.

It is also straightforward to see that we can rewrite the above arrays so as to propagate the angle-normalized residuals instead of the aposteriori residuals themselves. This is a matter of convenience as we readily check. Define the diagonal matrix of conversion factors,

$$\Gamma_M = \text{diagonal}\,\{\gamma_M(0), \gamma_M(1), \ldots, \gamma_M(N)\}$$

and the vectors of forward and backward *apriori* prediction errors, respectively,

$$\alpha_M = \begin{bmatrix} \alpha_M(1) \\ \vdots \\ \alpha_M(N+1) \end{bmatrix}, \quad \beta_M = \begin{bmatrix} \beta_M(0) \\ \vdots \\ \beta_M(N) \end{bmatrix}$$

then $\mathbf{f}_M = \Gamma_M \alpha_M$ and $\mathbf{b}_M = \Gamma_M \beta_M$. Thus projecting $\mathbf{f}_M$ onto $\mathbf{b}_M$ is equivalent to projecting $\Gamma_M \alpha_M$ onto $\Gamma_M \beta_M$, viz.,

$$\min_{\eta_M} \|\Lambda_N^{1/2}(\mathbf{f}_M - \eta_M \mathbf{b}_M)\|_2^2 = \min_{\eta_M} \|\Lambda_N^{1/2}\Gamma_M(\alpha_M - \eta_M \beta_M)\|_2^2 \tag{41a}$$

We can therefore introduce the following equivalent state-space models, which are essentially identical to (Eq. 39) but use the apriori residuals instead,

$$\begin{aligned} x_1(i+1) &= \lambda^{-1/2} x_1(i), & x_2(i+1) &= \lambda^{-1/2} x_2(i) \\ y_1(i) &= \beta_M(i) x_1(i) + v_1(i), & y_2(i) &= \alpha_M(i+1) x_2(i) + v_2(i) \end{aligned} \tag{42}$$

with

$$y_1(i) = \frac{\alpha_M(i+1)}{(\sqrt{\lambda})^i}, \quad y_2(i) = \frac{\beta_M(i)}{(\sqrt{\lambda})^i}$$

cov $(x_1(0)) = \lambda^{-1}\mu = \text{cov}(x_2(0))$ no and (recall our earlier remark prior to expression (Eq. 16b)

$$Ev_1(i)v_1^*(j) = Ev_2(i)v_2^*(j) = \bar{\gamma}_M^{-1}(i)$$

Writing down the extended square-root information filters for each of the above models will then lead us to a square-root version of the so-called QRD-LSL algorithm (compare with [1, p. 664], [22, p. 1158] and [19] – we are, for the time being and for simplicity, ignoring the joint process estimation part. This is discussed further ahead where we show that it simply corresponds to adding one more line to the arrays). It should also be clear that not all the lines of the two arrays are really necessary to order-update the prediction residuals. They are nevertheless kept for completeness. For example, the last line in each of the arrays given below are not explicitly used in the description of the QRD-LSL in [1]. Also, although we are exhibiting the arrays in terms of angle-normalized quantities, they can as well be written in terms of unnormalized quantities as in (Eq. 40a). In other words, apriori or aposteriori prediction errors can also be used by appropriately modifying the arrays.

Algorithm: *The solutions of the minimization criteria (Eq. 38a) and (Eq. 38b), along with the order update relation (Eq. 37a) for the forward and backward residual errors, can be recursively updated as follows: start with* $\bar{\Phi}_M^{b/2}(-1) = \bar{\Phi}_M^{f/2}(0) = \frac{1}{\sqrt{\mu}}, \bar{q}_M^b(-1) = 0 = \bar{q}_M^f(0)$, *(*$\mu$ *is usually a large positive number). For each time instant* $N \geq 0$ do:

Set $\bar{f}_0(N) = \bar{b}_0(N) = u(N)$ and $\gamma_0(N) = 1$,

repeat for each order $M = 0,1,2,\ldots,$ Max -1,

$$\begin{bmatrix} \sqrt{\lambda}\,\bar{\Phi}^{b/2}_{M}(N-1) & \bar{b}^{*}_{M}(N) \\ \sqrt{\lambda}\,\bar{q}^{*b}_{M}(N-1) & \bar{f}^{*}_{M}(N+1) \\ 0 & \gamma^{1/2}_{M}(N) \\ \frac{1}{\sqrt{\lambda}}\bar{\Phi}^{*b/2}_{M}(N-1) & 0 \end{bmatrix} \bar{\Theta}^{b}_{M,N}$$

$$= \begin{bmatrix} \bar{\Phi}^{b/2}_{M}(N) & 0 \\ \bar{q}^{*b}_{M}(N) & \bar{f}^{*}_{M+1}(N+1) \\ b_{M}(N)\bar{\Phi}^{*b/2}_{M}(N) & \gamma^{1/2}_{M+1}(N) \\ \bar{\Phi}^{*b/2}_{M}(N) & -\tilde{g}^{b}_{M}(N) \end{bmatrix}$$

$$\begin{bmatrix} \sqrt{\lambda}\bar{\Phi}^{f}{}_{M}/2(N) & \bar{f}^{*}_{M}(N+1) \\ \sqrt{\lambda}\bar{q}^{*f}_{M}(N) & \bar{b}^{*}_{M}(N) \\ 0 & \gamma^{1/2}_{M}(N) \\ \frac{1}{\sqrt{\lambda}}\bar{\Phi}^{*f/2}_{M}(N) & 0 \end{bmatrix} \bar{\Theta}^{f}_{M,N+1}$$

$$= \begin{bmatrix} \bar{\Phi}^{f}{}_{M}/2(N+1) & 0 \\ \bar{q}^{*f}_{M}(N+1) & \bar{b}^{*}_{M+1}(N+1) \\ f_{M}(N+1)\bar{\Phi}^{*f/2}_{M}(N+1) & \gamma^{1/2}_{M+1}(N+1) \\ \bar{\Phi}^{*f/2}_{M}(N+1) & -\tilde{g}^{f}_{M}(N+1) \end{bmatrix}$$

where the quantities $\{\bar{\Phi}^{b}_{M}(N), \bar{\Phi}^{f}_{M}(N), \bar{s}^{b}_{M}(N), \bar{s}^{f}_{M}(N), \bar{q}^{b}_{M}(N), \bar{q}^{f}_{M}(N)\}$ *are defined as before except with angle-normalized residuals. The unitary matrices* $\bar{\Theta}^{b}_{M,N}$ *and* $\bar{\Theta}^{f}_{M,N+1}$ *can be determined by the requirement that the (1,2) entries in the corresponding postarrays must be zero.*

Least-Squares Lattice (LSL) Filters Using Aposteriori Residuals

Let us now elaborate on our earlier claim that the arrays of the above algorithm are indeed central to the derivation of other variants of adaptive lattice algorithms. That is, we shall verify that by expanding the above arrays we get different lattice versions. But we shall also later show, and in order to be consistent with the spirit of our formulation, how to get these other lattice versions by invoking the other forms of the Kalman recursions, other than the extended square-root information form that we used to derive the prior algorithm .

To obtain an explicit set of equations for the least-squares lattice algorithm that uses aposteriori residuals, we simply compare (some of the terms) on both sides of the the arrays of the above algorithm, by squaring, as we did, for e.g., in the derivation of the square-root Kalman filter. This first leads to the following expressions in terms of angle-normalized residuals:

$$\bar{\Phi}^{f}_{M}(N+1) = \lambda\bar{\Phi}^{f}_{M}(N) + |\bar{f}_{M}(N+1)|^{2}$$
$$\bar{s}^{f}_{M}(N+1) = \lambda\bar{s}^{f}_{M}(N) + \bar{f}^{*}_{M}(N+1)\bar{b}_{M}(N)$$
$$\bar{b}^{f}_{M}(N)\gamma^{1/2}_{M}(N) = \bar{b}_{M+1}(N+1)\gamma^{1/2}_{M+1}(N+1) + f_{M}(N+1)\bar{\Phi}^{f}_{M}(N+1)\bar{s}^{f}_{M}(N+1)$$
$$\gamma_{M+1}(N+1) = \gamma_{M}(N) - \frac{|f_{M}(N+1)|^{2}}{\bar{\Phi}^{f}_{M}(N+1)} \qquad (43a)$$

and

$$\bar{\Phi}^{b}_{M}(N) = \lambda\bar{\Phi}^{b}_{M}(N-1) + |\bar{b}_{M}(N)|^{2}$$
$$\bar{s}^{b}_{M}(N) = \lambda\bar{s}^{b}_{M}(N-1) + \bar{b}^{*}_{M}(N)\bar{f}_{M}(N+1)$$
$$\bar{f}_{M}(N+1)\gamma^{1/2}_{M}(N) = \bar{f}_{M+1}(N+1)\gamma^{1/2}_{M+1}(N) + b_{M}(N)\bar{\Phi}^{b}_{M}(N)\bar{s}^{b}_{M}(N)$$

$$\gamma_{m+1}(N) = \gamma_{M}(N) - \frac{|b_{m}(N)|^{2}}{\bar{\Phi}^{b}_{M}(N)} \qquad (43b)$$

If we now replace $\bar{f}_{M}(N+1)\gamma^{1/2}_{M}(N)$ by $f_{M}(N+1)$ and $\bar{b}_{M+1}(N+1)\gamma^{1/2}_{M+1}(N+1)$ by $b_{M+1}(N+1)$ we get the following so-called LSL algorithm using aposteriori errors, (e.g., [22] and [1], p. 619]).

Algorithm: *The solutions of the minimization criteria (Eq. 38a) and (Eq. 38b), along with the order update relation (Eq. 37a) for the forward and backward residual errors, can be recursively updated as follows: start with* $\bar{\Phi}^{b/2}_{M}(-1) = \bar{\Phi}^{f/2}_{M}(0) = \frac{1}{\sqrt{\mu}}$ *(μ is usually a large positive number), and* $\bar{s}^{f}_{M}(0) = 0$. *For each time instant* $N \geq 0$ *do:*

Set $f_0(N) = b_0(N) = u(N)$ *and* $\gamma_0(N) = 1$,

repeat for each order $M = 0,1,2,\ldots,Max-1$,

$$\bar{\Phi}^{f}_{M}(N+1) = \lambda\bar{\Phi}^{f}_{M}(N) + \frac{|f_{M}(N+1)|^{2}}{\gamma_{M}(N)}$$
$$\bar{\Phi}^{b}_{M}(N) = \lambda\bar{\Phi}^{b}_{M}(N-1) + \frac{|b_{M}(N)|^{2}}{\gamma_{M}(N)}$$
$$\bar{s}^{f}_{M}(N+1) = \lambda\bar{s}^{f}_{M}(N) + \frac{\bar{f}^{*}_{M}(N+1)b_{M}(N)}{\gamma_{M}(N)} = \bar{s}^{*b}_{M}(N) \equiv \bar{s}_{M}(N+1)$$

$$f_{M+1}(N+1) = f_{M}(N+1) + b_{M}(N)k^{f}_{M+1}(N+1)$$
$$b_{M+1}(N+1) = b_{M}(N) + f_{M}(N+1)k^{*}_{M+1}(N+1)$$
$$k^{f}_{M+1}(N+1) = -\frac{\bar{s}^{*}_{M}(N+1)}{\bar{\Phi}^{b}_{M}(N)}$$
$$k^{b}_{M+1}(N+1) = -\frac{\bar{s}_{M}(N+1)}{\bar{\Phi}^{f}_{M}(N+1)}$$

$$\gamma_{M+1}(N) = \gamma_M(N) - \frac{|b_M(N)|^2}{\overline{\Phi}^b_M(N)}$$

It is worth pointing out that, for $\mu \to \infty$, the expressions for $\overline{\Phi}^b_M(N)$ and $\overline{\Phi}^f_M(N+1)$ can be interpreted as weighted energies of the backward and forward residuals, respectively. It thus immediately follows from (Eq. 36a) and (Eq. 36b) that they satisfy the following simple order-update relations:

$$\begin{aligned}\overline{\Phi}^f_{M+1}(N+1) &= \overline{\Phi}^f_M(N+1) - \frac{|\overline{s}^b_M(N)|^2}{\overline{\Phi}^b_M(N)} \\ \overline{\Phi}^b_{m+1} &= \overline{\Phi}^b_M(N) - \frac{|\overline{s}^f_M(N+1)|^2}{\overline{\Phi}^f_M(N+1)}\end{aligned} \tag{44}$$

These can then be used to order-update (rather than time-update) the residual energies in the above algorithm, which coincides with the form given in [1, p. 619], for instance.

The above lattice algorithm could have also been immediately obtained from a state-space point of view as follows: instead of applying the square-root information filter to the auxiliary models (Eq. 42), we simply write down the explicit information filter (Kalman filter). For example, the update expression for the inverse of the Riccati variable leads to the time-update recursions for $\overline{\Phi}^f_M(N+1)$ and $\overline{\Phi}^b_M(N)$ as shown in the statement of the prior algorithm above, since we already know that the inverse of the Riccati variable corresponds, apart from scaling, to the "covariance" matrix in the least-squares setting (recall the correspondences in Table 12).

As for the order-updates for the forward and backward residuals, these are simply the expressions for the computation of the the aposteriori errors. Consider, for example, $\alpha_{M+1}(N+1)$ and the associated state-space model (Eq. 42). The Information filter provides us with the normalized apriori error,

$$r_e^{-1}(N)e(N) = \gamma_M(N)\,[y_1(N) - \sqrt{\lambda}\beta_M(N)\hat{x}_1(N+1)]$$

In terms of the least-squares variables, the normalized error $r_e^{-1}(N)e(N)$ is equal to the aposteriori forward residual $\frac{1}{(\sqrt{\lambda})^N} f_{M+1}(N+1)$ and we get, by invoking the correspondences of Table 12, that $f_{M+1}(N+1) = f_M(N+1) - b_M(N)\eta_{M,N}$, i.e., we replaced $y_1(N)$ by $\alpha_M(N+1)/(\sqrt{\lambda})^N$ and $\hat{x}_1(N+1)$ by $\eta_{M,N}/(\sqrt{\lambda})^{N+1}$. The estimate of the scalar weight η, $\eta_{M,N}$, is often denoted by the (negative of the) so-called reflection coefficient $k^f_{M+1}(N+1)$ in the statement of the LSL algorithm. Hence, $k^f_{M+1}(N+1)$ is nothing but (apart from the sign) the state-estimate at time $(N+1)$. Indeed, $k^f_{M+1}(N+1)$ is defined as the ratio $k^f_{M+1}(N+1) = -\overline{s}^b_M(N)/\overline{\Phi}^b_M(N)$, which is precisely the solution of the normal equations associated with the first-order least-squares problem that updates the forward residual:

$$\overline{\Phi}^b_M(N)\left[-k^f_{M+1}(N+1)\right] = \overline{s}^b_M(N) \tag{45}$$

That is, $-k^f_{M+1}(N+1)$ is equal to both $(\sqrt{\lambda})^{N+1}\hat{x}_1(N+1)$ and $\eta_{M,N}$ (recall again the correspondences in Table 12). Moreover, the update for $\mathbf{P}^{-1}_{i+1}\hat{\mathbf{x}}_{i+1}$ prior to Eq. 14 converts to the update for the $s^f_M(\cdot)$ and $s^b_M(\cdot)$.

Lattice Filters Using Apriori Residuals with Error Feedback

Therefore, the LSL recursions given above correspond to the explicit recursions of the Information form. In particular, the reflection coefficients are computed as the ratio of the "cross-covariance" and the "covariance" quantities, as suggested by (Eq. 14). But we argued above that the reflection coefficients are nothing but (apart from the sign) the state-estimates. Hence, we can alternatively use the Kalman recursions to recursively update each one of them. For example, instead of computing $k^f_{M+1}(N+1)$ as the ratio $-\overline{s}^b_M(N)/\overline{\Phi}^b_M(N)$, we use the state-estimate update that follows by applying the Kalman filter to the auxiliary model (Eq. 42). This gives the followin equation,

$$\begin{aligned}&-\frac{k^f_{M+1}(N+1)}{(\sqrt{\lambda})_{N+1}} \\ &= -\lambda^{-1/2}\frac{k^f_{M+1}(N)}{(\sqrt{\lambda})^N} + \lambda^{-1/2}\frac{1}{\lambda\overline{\Phi}^b_M(N-1)}\beta^*_M(N)\frac{f_{M+1}(N+1)}{(\sqrt{\lambda})^N}\end{aligned}$$

or, more compactly,

$$k^f_{M+1}(N+1) = k^f_{M+1}(N) - \frac{\beta^*_M(N)\alpha_{M+1}(N+1)\gamma_{M+1}(N)}{\lambda\overline{\Phi}^b_M(N-1)} \tag{46}$$

If we now replace $\gamma_{M+1}(N)$ by

$$\gamma_{M+1}(N) = \gamma_M(N) - \frac{|b_M(N)|^2}{\overline{\Phi}^b_M(N)}$$

then the above time-update for the reflection coefficient collapses to the usual expression encountered in the literature and given below. A similar update follows for $k^b_{M+1}(N+1)$. Using these updates for the reflection coefficients and rewriting the previous algorithm in terms of the apriori residuals leads to the following so-called LSL algorithm with error-feedback (see, e.g., [1, p. 633]). Again, we can as well rewrite the expressions in terms of aposteriori errors, thus leading to a version of the algorithm that uses the aposteriori residuals. In the previous section we exhibited an example of a lattice filter that uses the aposteriori errors. We therefore exhibit

here, for the sake of illustration, an example of a lattice filter that uses the apriori residuals.

Algorithm: *The solutions of the minimization criteria (Eq. 38a) and (Eq. 38b), along with the order update relation (Eq. 37a) for the forward and backward residual errors, can be recursively updated as follows: start with* $\overline{\Phi}^{b}_{M}(-1) = \overline{\Phi}^{f}_{M}(0) = \frac{1}{\mu}$, $k^{f}_{M}(0) = k^{b}_{M}(0) = 0$, ($\mu$ *is usually a large positive number). For each time instant* $N \geq 0$ *do:*

Set $\alpha_0(N) = \beta_0(N) = u(N)$ *and* $\gamma_0(N) = 1$,
and repeat for each order $M = 0,1,2,\ldots,$ Max $- 1$,

$$\overline{\Phi}^{f}_{M}(N+1) = \lambda\overline{\Phi}^{f}_{M}(N) + \gamma_M(N)|\alpha_M(N+1)|^2$$

$$\overline{\Phi}^{b}_{M}(N) = \lambda\overline{\Phi}^{b}_{M}(N-1) + \gamma_M(N)|\beta_M(N)|^2$$

$$\alpha_{M+1}(N+1) = \alpha_M(N+1) + \beta_M(N)k^{f}_{M+1}(N)$$

$$\beta_{M+1}(N+1) = \beta_M(N) + \alpha_M(N+1)k^{b}_{M+1}(N)$$

$$k^{f}_{M+1}(N+1) = k^{f}_{M+1}(N) - \frac{\gamma_M(N)}{\overline{\Phi}^{b}_{M}(N)}\beta^{*}_{M}(N)\alpha_{M+1}(N+1)$$

$$k^{b}_{M+1}(N+1) = k^{b}_{M+1}(N) - \frac{\gamma_M(N)}{\overline{\Phi}^{f}_{M}(N+1)}\alpha^{*}_{M}(N+1)\beta_{M+1}(N+1)$$

$$\gamma_{M+1}(N) = \gamma_M(N) - \frac{|\gamma_M(N)\beta_M(N)|^2}{\overline{\Phi}^{b}_{M}(N)}$$

Normalized Least-Squares Lattice (LSL) Filters

The LSL filters considered in the previous two sections involve the propagation of two reflection coefficients, viz.,

$$k^{f}_{M+1}(N+1) = -\frac{\overline{s}^{*}_{M}(N+1)}{\overline{\Phi}^{b}_{M}(N)}, \quad k^{b}_{M+1}(N+1) = -\frac{\overline{s}_{M}(N+1)}{\overline{\Phi}^{f}_{M}(N+1)}$$

which can also be time-updated via relations of the form (Eq. 46). However, a normalized version of the recursions of the prior algorithm involving the propagation of a *single* reflection coefficient can also be derived [8].

For this purpose, we employ a proper normalization of the forward and backward reflection coefficients and define the normalized reflection coefficient $\rho_{M+1}(N+1)$ (we shall justify this definition, and the resulting expressions, in terms of state-space arguments further ahead),

$$\rho_{M+1}(N+1) = \frac{\overline{\Phi}^{b/2}_{M}(N)}{\overline{\Phi}^{f/2}_{M}(N+1)}k^{f}_{M+1}(N+1)$$

$$= \left[\frac{\overline{\Phi}^{f/2}_{M}(N+1)}{\overline{\Phi}^{b/2}_{M}(N)}k^{b}_{M+1}(N+1)\right]^{*}$$

We see that $\rho_{M+1}(N+1)$ is a scaled version of both $k^{f}_{M+1}(N+1)$ and $k^{*b}_{M+1}(N+1)$. We also define the "covariance-" or "energy-"normalized residuals,

$$\overline{f}_{M}(N+1) = \frac{\overline{f}_{M}(N+1)}{\overline{\Phi}^{f/2}_{M}(N+1)}, \quad \overline{b}_{M}(N) = \frac{\overline{b}_{M}(N)}{\overline{\Phi}^{b/2}_{M}(N)}$$

These normalizations allow us to reduce the number of recursions in the previous algorithm. It will turn out, for example, that the recursions for the covariances $\overline{\Phi}^{b}_{M}(N)$ and $\overline{\Phi}^{f}_{M}(N+1)$ can be dropped. Note for instance that the order-update relations (Eq. 44), with the help of (Eq. 45), can be reexpressed in terms of the normalized reflection coefficient,

$$\overline{\Phi}^{f}_{M+1}(N+1) = \overline{\Phi}^{f}_{M}(N+1)\left[1 - |\rho_{M+1}(N+1)|^2\right],$$

$$\overline{\Phi}^{b}_{M+1}(N+1) = \overline{\Phi}^{b}_{M}(N)\left[1 - |\rho_{M+1}(N+1)|^2\right]$$

Also, using the time-updates for either of the reflection coefficients in the prior algorithm we can obtain a time-update for the normalized coefficient $\rho_{M+1}(N+1)$. For example, starting with

$$k^{f}_{M+1}(N+1) = k^{f}_{M+1}(N) - \frac{\gamma_M(N)}{\overline{\Phi}^{b}_{M}(N)}\beta^{*}_{M}(N)\alpha_{M+1}(N+1)$$

and multiplying both sides by the ratio $\overline{\Phi}^{b/2}_{M}(N)/\overline{\Phi}^{f/2}_{M}(N+1)$, leads to the following sequence of easily verifiable identities,

$$\rho_{M+1}(N+1)$$

$$= \frac{\overline{\Phi}^{b/2}_{M}(N)}{\overline{\Phi}^{f/2}_{M}(N+1)}\left[k^{f}_{M+1}(N) - \frac{\overline{b}^{*}_{M}(N)\gamma^{1/2}_{M}(N)\alpha_{M+1}(N+1)}{\overline{\Phi}^{b}_{M}(N)}\right]$$

$$= (1 - |\overline{b}_{M}(N)|^2)\frac{\overline{\Phi}^{b/2}_{M}(N)}{\overline{\Phi}^{f/2}_{M}(N+1)}\frac{\overline{\Phi}^{f/2}_{M}(N)}{\overline{\Phi}^{b/2}_{M}(N-1)}\rho_{M+1}(N)$$

$$-\overline{b}_{M}(N)\overline{f}_{M}(N+1)$$

$$= (1 - |\overline{b}_{M}(N)|^2)\frac{\left[1 - |\overline{f}_{M}(N)|^2\right]^{1/2}}{\left[1 - |\overline{b}_{M}(N)|^2\right]^{1/2}}\rho_{M+1}(N) - \overline{b}^{*}_{M}(N)\overline{f}_{M}(N+1)$$

We therefore get

$$\rho_{M+1}(N+1) = \left[1 - |\overline{b}_{M}(N)|^2\right]^{*/2}\rho_{M+1}(N)\left[1 - |\overline{f}_{M}(N)|^2\right]^{1/2}$$

$$-\overline{b}^{*}_{M}(N)\overline{f}_{M}(N+1)$$

which incidentally is the famous Yule's PARCOR update formula discussed in [11].

Similary the recursions for the forward and backward residuals of the algorithm prior to Eq. 44 reduce to recursions

for the corresponding "covariance"-normalized residuals, as stated below.

Algorithm: *The solutions of the minimization criteria (Eq. 38a) and (Eq. 38b), along with the order update relation (Eq. 37a) for the forward and backward residual errors, can be recursively updated as follows: start with* $\rho_M(0) = 0$ *, and for each time instant* $N \geq 0$ *do:*

$$Set\ \bar{\bar{f}}_0(N) = \frac{u(N)}{\bar{\Phi}_0^{f/2}(N)},\ \bar{\bar{b}}_0(N) = \frac{u(N)}{\bar{\Phi}_0^{b/2}(N)},$$

repeat for each order $M = 0,1,2,\ldots,$ Max - 1,

$$\rho_{M+1}(N+1)$$

$$= \left[1 - |\bar{\bar{b}}_M(N)|^2\right]^{*/2} \rho_{M+1}(N) \left[1 - |\bar{\bar{f}}_M(N)|^2\right]^{1/2} - \bar{\bar{b}}_M^*(N)\bar{\bar{f}}_M(N+1)$$

$$\bar{\bar{f}}_{M+1}(N+1)$$

$$= \left[1 - |\bar{\bar{b}}_M(N)|^2\right]^{-1/2} \left[\bar{\bar{f}}_M(N+1) + \bar{\bar{b}}_M(N)\rho_{M+1}(N+1)\right] \rho_{M+1}^{e}(N+1)$$

$$\bar{\bar{b}}_{M+1}(N+1)$$

$$= \left[1 - |\bar{\bar{f}}_M(N+1)|^2\right]^{-1/2} \left[\bar{\bar{b}}_M(N) + \bar{\bar{f}}_M(N+1)\rho_{M+1}^*(N+1)\right] \rho_{M+1}^{c}(N+1)$$

where we have defined

$$\rho_{M+1}^{c}(N+1) = \left[1 - |\rho_{M+1}(N+1)|^2\right]^{1/2}$$

Note that the initialization requires apriori knowledge of the quantities $\bar{\Phi}_0^{f/2}(N)$ and $\bar{\Phi}_0^{b/2}(N)$. But recall that $\bar{\Phi}_0^{b}(N)$ is, by definition, related to the energy of the input signal up to time N, viz.,

$$\bar{\Phi}_0^{b}(N) = \lambda^{(N+1)}\mu^{-1} + \sum_{i=0}^{N} \lambda^{N-i}|u(i)|^2.$$

A similar remark holds for the quantity $\bar{\Phi}_0^{f/2}(N)$. The above algorithm, therefore, needs to be supplied with estimates for these quantities.

We further remark that the above recursions can also be interpreted as resulting from state-space estimation algorithms. Indeed, and following similar steps to what we did in (Eq. 41a), while reducing the original optimization problem on $\{\mathbf{f}_M, \mathbf{b}_M\}$ to an equivalent optimization problem on $\{\alpha_M, \beta_M\}$, we can also reduce the optimization problem on $\{\mathbf{f}_M, \mathbf{b}_M\}$ to another equivalent optimization problem on $\{\bar{\mathbf{f}}_M, \bar{\mathbf{b}}_M\}$. The estimation equations for the corresponding state-space models will then lead to the above recursions: the update for the state-estimate will lead to the recursion for $\rho_{M+1}(N+1)$, as was the case with $k_{M+1}^{f}(N+1)$, while the expressions for the innovations lead to the updates for $\bar{\bar{f}}_{M+1}(N+1)$ and $\bar{\bar{b}}_{M+1}(N+1)$. An interesting aspect of the state-space models that correspond to the "energy"-normalized variables $\{\bar{\mathbf{f}}_M, \bar{\mathbf{b}}_M\}$ is that (λ times) the associated Riccati variables at time $N+1$ is equal to unity. This is because the inverses of the Riccati variables are the "covariances" or the "energies" of the (normalized) sequences $\{\bar{\bar{f}}_M(\cdot)\}$ and $\{\bar{\bar{b}}_M(\cdot)\}$.

Other Forms of Lattice Filters

It is clear from the derivation so far that the variety of algorithms that can be written down is essentially a matter of taste as well as patience. The central theme though is always the same: once the recursive least-squares problem has been collapsed to two smaller first-order problems, then the solutions can be readily obtained by constructing two first-order auxiliary state-space models and then writing down the different estimation algorithms.

Indeed, the different variants of the Kalman state-space estimation algorithms lead to different relations among the variables of the forward and backward prediction problems. Any properly chosen collection of these relations will then lead to a valid lattice algorithm. For example, the QRD-LSL algorithm uses some of the lines in the extended information array. The LSL filter with aposteriori errors uses the explicit information filter equations, while the LSL filter with error feedback uses a combination of the information and the Kalman filters.

Yet a fourth variant in [20 p. 887] can also be rederived by choosing an appropriate collection of the rows of the arrays of the Algorithm stated after (Eq. 42), along with one of the order-updates in (Eq. 44), as we readily verify. It follows from the third lines of each of the arrays that the last expressions in (Eq. 43a) and (Eq. 43b) hold, viz.,

$$\gamma_{M+1}(N+1) = \gamma_M(N) - \frac{|f_M(N+1)|^2}{\bar{\Phi}_M^{f}(N+1)}$$

$$\gamma_{M+1}(N) = \gamma_M(N) - \frac{|b_M(N)|^2}{\bar{\Phi}_M^{b}(N)},\ \gamma_0(N) = 1$$

We thus conclude that the forward and backward residuals satisfy the following relation

$$\frac{|f_{M+1}(N+1)|^2}{\Phi_{M+1}^{f}(N+1)} - \frac{|b_{M+1}(N+1)|^2}{\Phi_{M+1}^{b}(N+1)}$$

$$= \frac{|f_M(N+1)|^2}{\Phi_M^{f}(N+1)} - \frac{|b_M(N)|^2}{\Phi_M^{b}(N)} \qquad (47a)$$

and that

$$\gamma_{M+1}^{1/2}(N) = \left[1 - \sum_{k=0}^{M} \frac{|b_k(N)|^2}{\bar{\Phi}_k^b(N)} \right]^{1/2}$$

It further follows from the second and third lines of the first array of the algorithm that

$$f_M(N+1) = f_{M+1}(N+1) + b_M(N)\bar{\Phi}_M^{*b/2}(N)\bar{q}_M^b(N) \quad \text{(47b)}$$

It is then easy to see that expressions (Eq. 47a)-(Eq. 47b) and (Eq. 44) can be alternatively expressed in the square-root form,

$$\begin{bmatrix} \bar{\Phi}_{M+1}^{f/2}(N+1) & \bar{q}_M^{*b}(N) \\ \dfrac{f_{M+1}(N+1)}{\Phi_{M+1}^{*f/2}(N+1)} & \dfrac{b_M(N)}{\Phi_M^{*b/2}(N)} \end{bmatrix} \bar{\Sigma}_{M,N}^f = \begin{bmatrix} \bar{\Phi}_M^{f/2}(N+1) & 0 \\ \dfrac{f_M(N+1)}{\Phi_M^{*f/2}(N+1)} & \dfrac{b_{M+1}(N+1)}{\Phi_{M+1}^{*b/2}(N+1)} \end{bmatrix}$$

since, by definition, $\bar{s}_M^b(N) = \bar{\Phi}_M^{b/2}(N)\bar{q}_M^b(N)$. Here $\bar{\Sigma}_{M,N}^f$ is a unitary rotation that produces the zero entry in the (1,2) position of the postarray.

We thus obtain the following so-called Hybrid QR/Lattice least-squares algorithm. The equations are spelled out explicitly in order to facilitate the comparison with [20 p. 887].

Algorithm: *The solutions of the minimization criteria (Eq. 38a) and (Eq. 38b), along with the order update relation (Eq. 37a) for the forward and backward residual errors, can be recursively updated as follows: start with* $\bar{\Phi}_M^{b/2}(-1) = \bar{\Phi}_M^{f/2}(0) = \frac{1}{\sqrt{\mu}}$ *(μ is usually a large positive number),* $\bar{q}_M^b(-1) = 0 = \bar{q}_M^f(0)$, $\bar{f}_0(N) = \bar{b}_0(N) = u(N)$ *and* $\gamma_0(N) = 1$.

(1) *For* $M = 0,1,2,...,\text{Max} - 1$ *do:*

$$\left[\sqrt{\lambda}\bar{q}_M^{*b}(N-1) \;\; \bar{f}_M^*(N+1) \right] \bar{\Theta}_{M,N}^b = \left[\bar{q}_M^{*b}(N) \;\; \bar{f}_{M+1}^*(N+1) \right]$$

(2) *Compute* $\bar{\Phi}_{Max}^{f/2}(N+1)$,

$$\left[\sqrt{\lambda}\bar{\Phi}_{Max}^{f/2}(N) \;\; \bar{f}_{Max}^*(N+1) \right] \bar{\Theta}_{Max,N+1}^f = \left[\bar{\Phi}_{Max}^{f/2}(N+1) \;\; 0 \right]$$

(3) *For* $M = \text{Max} - 1, \text{Max} - 2,...,0$ *do:*

$$\left[\bar{\Phi}_{M+1}^{f/2}(N+1) \;\; \bar{q}_M^{*b}(N) \right] \bar{\Sigma}_{M,N}^f = \left[\bar{\Phi}_M^{f/2}(N+1) \;\; 0 \right]$$

(4a) *Compute* $\dfrac{f_{Max}(N+1)}{\Phi_{Max}^{*f/2}(N+1)}$ *as (x denotes an irrelevant entry),*

$$\left[0 \;\; \gamma_{Max}^{1/2}(N) \right] \bar{\Theta}_{Max,N+1}^f = \left[\frac{f_{Max}(N+1)}{\bar{\Phi}_{Max}^{*f/2}(N+1)} \;\; x \right]$$

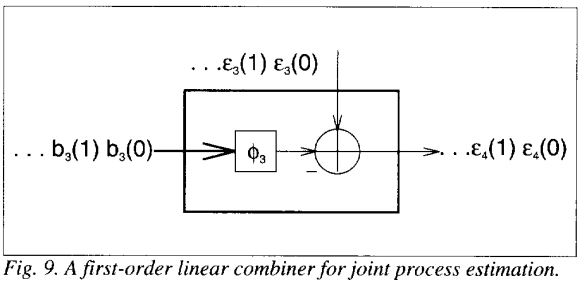

Fig. 9. A first-order linear combiner for joint process estimation.

(4b) *For* $M = \text{Max} - 1, \text{Max} - 2,...,0$ *do:*

$$\begin{bmatrix} \dfrac{f_{M+1}(N+1)}{\Phi_{M+1}^{*f/2}(N+1)} & \dfrac{b_M(N)}{\Phi_M^{*b/2}(N)} \end{bmatrix} \bar{\Sigma}_{M,N}^f = \begin{bmatrix} \dfrac{f_M(N+1)}{\Phi_M^{*f/2}(N+1)} & \dfrac{b_{M+1}(N+1)}{\Phi_{M+1}^{*b/2}(N+1)} \end{bmatrix}.$$

(5) *Compute* $\gamma_{Max}^{1/2}(N)$,

$$\gamma_{Max}^{1/2}(N) = \left[1 - \sum_{k=0}^{Max-1} \frac{|b_K(N)|^2}{\bar{\Phi}_k^b(N)} \right]^{1/2}$$

(6) *For* $M = \text{Max} - 1, \text{Max} - 2,...,0$ do:

$$\left[0 \;\; \gamma_M^{1/2}(N) \right] = \left[\frac{b_M(N)}{\bar{\Phi}_M^{*b/2}(N)} \;\; \gamma_{M+1}^{1/2}(N) \right] \left[\bar{\Theta}_{M,N}^b \right]^{-1}$$

The unitary matrices $\bar{\Theta}_{M,N}^b$ *and* $\bar{\Theta}_{M,N+1}^f$ *are determined by the requirement that the (1,2) entries in the corresponding postarrays must be zero.*

Other combinations of equations clearly exist, and the search for the most appropriate combination is not closed. We do not attempt to exhaust all the possibilities, but we do stress a central theme: the existing adaptive schemes have been derived at different places in the literature and at different times. The varied aspects of the recursions highlight the varied approaches to the problem. Each approach ends up with a set of recursions that very often does not look similar to other available sets. But the point of view taken in this article shows that the different adaptive solutions correspond in fact to using different combinations of a group of estimation equations, either in explicit or square-root forms. The entire group of equations is not needed per se since it contains more than enough relations. But different selections of the equations will lead to different algorithms. The question of which of these different possibilities is the best needs further exploration; moreover, it can be that what is best depends on the constraints arising in different situations.

The Filtering or Joint Process Array

We now return to the estimation of the sequence $\{d(\cdot)\}$. We argued in the section on *Joint Process Estimation and Prediction Problems* that if we are given the backward residual vector $\mathbf{b}_3$ and the third-order estimation residual vector $\tilde{\mathbf{d}}_3$, then the higher-order estimation residual vector $\tilde{\mathbf{d}}_4$ can be obtained by simply setting up a first-order least-squares problem: use the entries of the residual vector $\tilde{\mathbf{d}}_3$ as *reference* signals and the entries of the backward vector $\mathbf{b}_3$ as *input* signals in a first-order combiner, as we anticipated earlier in Figure 3. The diagram is repeated in Figure 9 with the appropriate signals.

This again fits nicely into the setting anticipated in Eq. 12c. Indeed, we can set up a first-order state-space model that corresponds to the solution of the above order-update problem, in exactly the same way as we argued in the lattice case. This leads to the following square-root array for joint process estimation:

Algorithm: *Consider the minimization problem*

$$\min_{\phi_M}\left[\frac{\lambda^{(N+1)}}{\mu}|\phi_M|^2+\sum_{j=0}^{N}\lambda^{N-j}|\varepsilon_M(j)-\phi_M b_M(j)|^2\right]$$

where $\{\varepsilon_M(j)\}_{j=0}^{N}$ *denote the entries of the* M^{th} *order aposteriori residual vector* $\tilde{\mathbf{d}}_M$. *The angle-normalized aposteriori residuals* $\{\bar{\varepsilon}_M(j)=\varepsilon_M(j)\gamma_M^{-1/2}(j)\}$ *can be order-updated as follows: start with* $\bar{\Phi}_M^{b/2}(-1)=\frac{1}{\sqrt{\lambda}}$, $\bar{q}_M^d(-1)=0$, (μ *is usually a large positive number). For each time instant* $N\geq 0$ *do:*

Set $\bar{b}_0(N)=u(n)$, $\bar{\varepsilon}_0(N)=d(N)$,*and* $\gamma_0(N)=1$.
Repeat for each order $M=0,1,2,\ldots,Max-1$,

$$\begin{bmatrix}\sqrt{\lambda}\,\bar{\Phi}_M^{b/2}(M-1) & \bar{b}_M^*(N)\\ \sqrt{\lambda}\,\bar{q}_M^{*d}(N-1) & \bar{\varepsilon}_M^*(N)\\ 0 & \gamma_M^{1/2}(N)\\ \frac{1}{\sqrt{\lambda}}\bar{\Phi}_M^{*b/2}(N-1) & 0\end{bmatrix}\bar{\Theta}_{M,N}^{b}$$

$$=\begin{bmatrix}\bar{\Phi}_M^{b/2}(N) & 0\\ \bar{q}_M^{*d}(N) & \bar{\varepsilon}_{M+1}^*(N)\\ b_M(N)\bar{\Phi}_M^{*b/2}(N) & \gamma_{M+1}^{1/2}(N)\\ \bar{\Phi}_M^{*b/2}(N) & -\tilde{g}_M^b(N)\end{bmatrix}$$

where we have defined

$$\bar{q}_M^d(N)=\frac{\bar{s}_M^d(N)}{\bar{\Phi}_M^{b/2}(N)}$$

$$\bar{s}_M^d(N)=\sum_{j=0}^{N}\lambda^{N-j}\bar{b}_M^*(j)\bar{\varepsilon}_M(j),\quad \bar{s}_M^d(-1)=0.$$

and the unitary matrix $\bar{\Theta}_{M,N}^{b}$ *is defined by the requirement that the (1,2) entry in the postarray must be zero.*

Note that the above array uses precisely the same rotation as the first array in the QRD-LSL algorithm. Hence, the second line in the above array can be included as one more line in the first array of the algorithm; thus completing the algorithm to also include the joint-process estimation part.

We can as well expand the above array in order to obtain an explicit update relation of the form (see e.g., [1, p. 619]),

$$\varepsilon_{M+1}(N)=\varepsilon_M(N)-\frac{\bar{s}_M^d(N)}{\bar{\Phi}_M^b(N)}b_M(N)$$

$$\bar{s}_M^d(N)=\lambda\bar{s}_M^d(N-1)+\frac{b_M(N)\varepsilon_M(N)}{\gamma_M(N)}$$

The Least-Mean Square (LMS) Algorithm

The derivation so far in the paper has been concerned with exact recursive solutions that minimize the cost function (Eq. 19b). Consider, for simplicity, a specialization of (Eq. 19b) with $\bar{\mathbf{w}}=0, \Pi_0\rightarrow\infty\,\mathbf{I}$, and $\lambda=1$,

$$\min_{\mathbf{w}}\sum_{i=0}^{N}|d(i)-\mathbf{u}_i\mathbf{w}|^2 \tag{48}$$

which is clearly a quadratic cost function in the unknown $\mathbf{w}$. The RLS solution propagates successive estimates of $\mathbf{w}$,

$$\mathbf{w}_i=\mathbf{w}_{i-1}+\mathbf{g}_i\,[d(i)-\mathbf{u}_i\mathbf{w}_{i-1}],\quad \mathbf{w}_{-1}=\mathbf{0},$$

using a gain vector $\mathbf{g}_i$ that is computed in terms of a variable $\mathbf{P}_i$ that satisfies the Riccati difference equation (Eq. 23). This requires $O(M^2)$ operations per iteration.

There is yet another celebrated class of adaptive algorithms that is based on steepest descent ideas (see, e.g., [6]). These algorithms are particularly appealing due to their simplicity and $O(M)$ computational efficiency, albeit at the price of slower convergence. In them, the weight vector is updated along the direction of the instantaneous gradient of the quadratic cost function thus leading to a weight update of the form

$$\mathbf{w}_i=\mathbf{w}_{i-1}+\mu\mathbf{u}_i^*[d(i)-\mathbf{u}_i\mathbf{w}_{i-1}],\quad \mathbf{w}_{-1}=0 \tag{49}$$

where μ is a so-called step-size parameter. This was dubbed by Widrow and Hoff [53] as the LMS (least-mean-square) algorithm. Though the name is a bit misleading, this is per-

haps the most widely known and most widely used adaptive filtering algorithm, and in diverse areas that range from channel equalization, to spectral estimation, to control theory, to antenna arrays, and no doubt other areas as well. This is because of its simplicity and proven robustness to disturbances and model errors. No rigorous proof of these good properties seems to have been found.

In the last decade, in the field of robust control (see, e.g., [55, 56]) there has been extensive studies of a different performance criterion, known as the H^∞ (or minimax) criterion. Very few exact H^∞ solutions are known, so it may be of additional interest to researchers in adaptive filtering that it has been recently shown that the LMS algorithm is optimal under the H^∞ criterion; this was again done by using an appropriate state-space model [54]. We can not enter into this topic here, but we may attempt to give the flavor of the results following the line of reasoning suggested in [57], where several (local and global) optimality criteria are further established for a wide class of gradient-type algorithms.

The argument that follows is intended to show that the weight estimates provided by the LMS algorithm guarantee the following bound

$$\frac{\Sigma_{i=0}^{N} |\mathbf{u}_i\mathbf{w} - \mathbf{u}_i\mathbf{w}_{i-1}|^2}{\mu^{-1}\|\mathbf{w} - \mathbf{w}_{-1}\|_2^2 + \Sigma_{i=0}^{N}|v(i)|^2} \leq 1. \tag{50a}$$

The numerator is the sum of the energies of the residuals $(\mathbf{u}_i\mathbf{w} - \mathbf{u}_i\mathbf{w}_{i-1})$ over $0 \leq i \leq N$. For each instant i, the difference $(\mathbf{u}_i\mathbf{w} - \mathbf{u}_i\mathbf{w}_{i-1})$ represents the error in estimating $\mathbf{u}_i\mathbf{w}$ by using $\mathbf{u}_i\mathbf{w}_{i-1}$. Likewise, the sum in the denominator consists of two terms: the energy of the noise signal over the same interval of time and the (weighted) energy of the weight error due to the initial guess. If we denote by T the operator that maps the disturbances $\{v(\cdot)\}_{i=0}^{N}$ and the initial uncertainty $\mu^{-1/2}(\mathbf{w} - \mathbf{w}_{-1})$ to the residuals $\{\mathbf{u}_i\mathbf{w} - \mathbf{u}_i\mathbf{w}_{i-1}\}_{i=0}^{N}$, then the inequality in (Eq. 50a) states that the 2-induced norm of T is always bounded by one. This explains the robust behaviour of the LMS algorithm: it shows that the energy of the residuals is always guaranteed to be upper bounded by the energy of the disturbances and the initial uncertainty.

A simple proof of (Eq. 50a) is the following [57]. Assume we are given noisy measurements $\{d(i)\}_{i=0}^{N}$,

$$d(i) = \mathbf{u}_i\mathbf{w} + v(i)$$

and that we are interested in finding a recursive update for the weight vector $\mathbf{w}$ so as to meet a certain optimality criterion. But let us for the moment ignore any optimality criterion and just note the following: if we pick any positive real number μ so as to satisfy the inequality

$$0 < \mu \leq \frac{1}{\mathbf{u}_i\mathbf{u}_i^*} \tag{50b}$$

and then choose *at will* any vector $\mathbf{q}$ as an estimate for the unknown weight $\mathbf{w}$, then it is always true that the following inequality holds

$$\frac{|\mathbf{u}_i\mathbf{w} - \mathbf{u}_i\mathbf{q}|^2}{\mu^{-1}\|\mathbf{w} - \mathbf{q}\|_2^2} \leq 1. \tag{50c}$$

This follows from the condition on μ and by noting that

$$|\mathbf{u}_i\mathbf{w} - \mathbf{u}_i\mathbf{q}|^2 = |\mathbf{u}_i(\mathbf{w} - \mathbf{q})|^2 \leq \|\mathbf{u}_i\|_2^2\|\mathbf{w} - \mathbf{q}\|_2^2.$$

The quantity in the numerator of (Eq. 50c) is the square of the error in estimating $\mathbf{u}_i\mathbf{w}$ by using $\mathbf{u}_i\mathbf{q}$. Likewise, the quantity in the denominator of (Eq. 50c) is the square of the distance between the true $\mathbf{w}$ and the estimate $\mathbf{q}$.

It is also certainly true that if we increase the denominator by any positive value, say $|v(i)|^2$, then the ratio is still bounded by 1,

$$\frac{|\mathbf{u}_i\mathbf{w} - \mathbf{u}_i\mathbf{q}|^2}{\mu^{-1}\|\mathbf{w} - \mathbf{q}\|^2 + |v(i)|^2} \leq 1. \tag{50d}$$

The inequalities (Eq. 50c) and (Eq. 50d) are valid for *any* data $\mathbf{u}_i$ as long as μ satisfies (Eq. 50b). That is, they are valid for any choice of $\mathbf{q}$. They are thus certainly valid for a $\mathbf{q}$ that is generated by the LMS algorithm (Eq. 49). So if we replace $\mathbf{q}$ by $\mathbf{w}_{i-1}$ we also get

$$\frac{|\mathbf{u}_i\mathbf{w} - \mathbf{u}_i\mathbf{w}_{i-1}|^2}{\mu^{-1}\|\mathbf{w} - \mathbf{w}_{i-1}\|^2 + |v(i)|^2} \leq 1. \tag{51a}$$

One might then wonder in what sense does the LMS recursion alter (Eq. 51a)? It turns out that it allows us to further tighten the inequality [57]. More precisely, it allows us to conclude that the following also holds,

$$\frac{|\mathbf{u}_i\mathbf{w} - \mathbf{u}_i\mathbf{w}_{i-1}|^2}{\mu^{-1}\|\mathbf{w} - \mathbf{w}_{i-1}\|_2^2 - \mu^{-1}\|\mathbf{w} - \mathbf{w}_i\|_2^2 + |v(i)|^2} \leq 1 \tag{51b}$$

Comparing (Eq. 51b) with (Eq. 51a) we see that the denominator of (Eq. 51b) is smaller since the positive term $\mu^{-1}\|\mathbf{w} - \mathbf{w}_i\|_2^2$ is being subtracted from the denominator of (Eq. 51a). But although the denominator got smaller, the ratio is still guaranteed to be bounded by one. A simple proof of (Eq. 51b) follows by noting that the inequality holds if

$$\mu^{-1}\|\mathbf{w} - \mathbf{w}_{i-1}\|_2^2 - \mu^{-1}\|\mathbf{w} - \mathbf{w}_i\|_2^2 + |v(i)|^2 - |\mathbf{u}_i\mathbf{w} - \mathbf{u}_i\mathbf{w}_{i-1}|^2 \geq 0$$

But the quantity on the left-hand side can be easily seen, after replacing $\mathbf{w}_i$ by the LMS update (Eq. 49), to be equal to

$(1 - \mu \mathbf{u}_i \mathbf{u}_i^*)|d(i) - \mathbf{u}_i \mathbf{w}_{i-1}|^2$, which is clearly always nonnegative.

So assume now that we run the LMS recursion up to time N and that

$$0 < \mu \leq \min_{0 \leq i \leq N} \frac{1}{\mathbf{u}_i \mathbf{u}_i^*}$$

Then the inequality in (Eq. 51b) holds for each time instant i, i.e.,

$$|\mathbf{u}_i \mathbf{w} - \mathbf{u}_i \mathbf{w}_{i-1}|_2^2 \leq \mu^{-1} \|\mathbf{w} - \mathbf{w}_{i-1}\|_2^2 - \mu^{-1} \|\mathbf{w} - \mathbf{w}_i\|_2^2 + |v(i)|^2$$

for every $0 \leq i \leq N$. Summing over i we conclude that we must have

$$\sum_{i=0}^{N} |\mathbf{u}_i \mathbf{w} - \mathbf{u}_i \mathbf{w}_{i-1}|^2 \leq \mu^{-1} \|\mathbf{w} - \mathbf{w}_{-1}\|_2^2 + \sum_{i=0}^{N} |v(i)|^2$$

which proves our earlier claim in (Eq. 50a). Extensions of this result to other classes of gradient-type algorithms, along with convergnece proofs, are provided in [57].

Concluding Remarks

The main point to stress here is that by a proper recasting of the original adaptive problem into a state-space form, we can derive many known adaptive filtering algorithms very directly and in computationally effective square-root versions. Even more interesting is the fact that insights gained from this formulation allow for immediate extensions. This is due to the fact that state-space models have a rich and long history and lend themselves rather easily to different types of manipulations and derivations. This constitutes a significant strength of the state-space formulation. It allows us, for instance, to consider more general matrices $\mathbf{F}_i$ in (Eq. 11a) rather than the particular choice $\mathbf{F}_i = \lambda^{-1/2} \mathbf{I}$, for e.g., $\mathbf{F}_i = \text{diagonal}\{\lambda_1^{-1/2}, \lambda_2^{-1/2}, ..., \lambda_n^{-1/2}\}$. It also allows for more general matrices $\mathbf{G}_i$, $\mathbf{Q}_i$ and $\mathbf{R}_i$, and for more general shift structures in the input signals via different choices of the matrix Ψ in the extended Chandrasekhar recursions. Indeed, different choices for Ψ, as well as $\mathbf{F}_i$, would allow us to consider alternative windowing schemes for the data. These extensions will be discussed elsewhere.

We have focused here on the single channel RLS problem, where the reference sequence $\{d(\cdot)\}$ and the input sequence $\{u(\cdot)\}$ were assumed to be scalar quantities. But we have noted in several places that the approach extends rather immediately to cases where the reference and input sequences consist of vector quantities (see, e.g., [25, 32]). We may remark that a strength of the multichannel square-root array formulation is that though the arrays naturally have block entries, the elementary rotations or reflections can be applied to the scalar entries, thus simplifying implementation issues (e.g., no matrix inversions will be necessary). Finally we may remark that the discussion in the last section can also be formulated in a state-space framework, but with the random variables allowed to exist in a certain indefinite metric space (see, e.g., [58]).

Acknowledgment

This work was supported in part by the Army Research Office under Grant DAAH04-93-G-0029, and the Advanced Research Projects Agency of the Department of Defense and was monitored by the Air Force Office of Scientific Research under Grant F49620-93-1-0085.

Dr. Sayed's work was also supported by a fellowship from the Fundacao de Amparo a Pesquisa do Estado de Sao Paulo, Brazil, while on leave from the Department of Electrical Engineering, Escola Politecnica da Universidade de Sao Paulo, Brazil.

Ali H. Sayed is an Assistant Professor in the Department of Electrical and Computer Engineering, University of California, Santa Barbara, CA, 93106. *Thomas Kailath* is a Professor in the Department of Electrical Engineering, Stanford University, Stanford, CA, 94305 .

APPENDIX

A Derivation of the Riccati-Based Kalman Filter

The innovations are

$$\mathbf{e}_i = \mathbf{y}_i - \hat{\mathbf{y}}_i = \mathbf{y}_i - (\mathbf{H}_i \hat{\mathbf{x}}_i + \hat{\mathbf{v}}_i)$$
$$= \mathbf{y}_i - \mathbf{H}_i \hat{\mathbf{x}}_i = \mathbf{H}_i \tilde{\mathbf{x}}_i + \hat{\mathbf{v}}_i$$

where we have defined $\tilde{\mathbf{x}}_i = \mathbf{x}_i - \hat{\mathbf{x}}_i$. If we compute the covariance of the innovations using the above equation we readily see that

$$\mathbf{R}_{\mathbf{e},i} = \text{cov}(\mathbf{e}_i) = \mathbf{R}_i + \mathbf{H}_i \mathbf{P}_i \mathbf{H}_i^* ,$$

where we have defined the error covariance matrix, $\mathbf{P}_i = \text{cov}(\tilde{\mathbf{x}}_i)$. For well-posedness, we need the positive-definiteness of the covariance matrix of the observations $\mathbf{y}_i$, which is easily seen to imply that the $\mathbf{R}_{\mathbf{e},i}$ have to be positive-definite. The Kalman filter can now be readily derived by using the orthogonality of the innovations and the state-space structure. Thus we first write

$$\hat{\mathbf{x}}_{i+1} = \sum_{j=0}^{i} \text{cov}(\mathbf{x}_{i+1}, \mathbf{e}_j) \text{cov}^{-1}(\mathbf{e}_j) \mathbf{e}_j .$$

To seek a recursion we decompose the above as

$$\hat{\mathbf{x}}_{i+1} = \sum_{j=0}^{i} \mathrm{cov}(\mathbf{x}_{i+1},\mathbf{e}_j)\mathbf{R}_{\mathbf{e},j}^{-1}\mathbf{e}_j + \mathbf{K}_i\mathbf{R}_{\mathbf{e},i}^{-1}\mathbf{e}_i,$$

where we have defined $\mathbf{K}_i = \mathrm{cov}(\mathbf{x}_{\mathbf{i+1}},\mathbf{e}_i)$. Now

$$\begin{aligned}\mathrm{cov}(\mathbf{x}_{i+1},\mathbf{e}_i) &= \mathbf{F}_i\mathrm{cov}(\mathbf{x}_i,\mathbf{e}_i) + \mathbf{G}_i\mathrm{cov}(\mathbf{r}_i,\mathbf{e}_i)\\ &= \mathbf{F}_i\,\mathrm{cov}(\mathbf{x}_i, \mathbf{H}_i\tilde{\mathbf{x}}_i + \mathbf{v}_i) + 0\\ &= \mathbf{F}_i\,\mathrm{cov}(\tilde{\mathbf{x}}_i, \mathbf{H}_i\tilde{\mathbf{x}}_i) + 0 = \mathbf{F}_i\mathbf{P}_i\mathbf{H}_i^*\end{aligned}$$

Note also that the first summation can be rewritten as

$$\mathbf{F}_i\sum_{j=0}^{i-1}\mathrm{cov}(\mathbf{x}_i,\mathbf{e}_j)\mathbf{R}_{\mathbf{e},j}^{-1}\,\mathbf{e}_j + \mathbf{G}_i\sum_{j=0}^{i-1}\mathrm{cov}(\mathbf{r}_i,\mathbf{e}_j)\mathbf{R}_{\mathbf{e},j}^{-1}\mathbf{e}_j = \mathbf{F}_i\hat{\mathbf{x}}_i + 0$$

Combining these facts we find

$$\hat{\mathbf{x}}_{i+1} = \mathbf{F}_i\hat{\mathbf{x}}_i + \mathbf{K}_i\mathbf{R}_{\mathbf{e},i}^{-1}\mathbf{e}_i.$$

It now remains to find a recursion for $\mathbf{P}_i$. To this end, note that if we define the covariance matrices $\Pi_i = \mathrm{cov}(\mathbf{x}_i)$ and $\Sigma_i = \mathrm{cov}(\hat{\mathbf{x}}_i)$, then the orthogonality of the projection and error yields, $\mathbf{P}_i = \Pi_i - \Sigma_i$. Using the state-space equation (Eq. 10a) it is easy to find the following recursion for Π_i,

$$\Pi_{i+1} = \mathbf{F}_i\,\Pi_i\,\mathbf{F}_i^* + \mathbf{G}_i\mathbf{Q}_i^*\mathbf{G}_i^*\,.$$

Likewise, from the recursion $\hat{\mathbf{x}}_{i+1} = \mathbf{F}_i\hat{\mathbf{x}}_i + \mathbf{K}_i\mathbf{R}_{\mathbf{e},i}^{-1}\mathbf{e}_i$ one obtains,

$$\Sigma_{i+1} = \mathbf{F}_i\Sigma_i\mathbf{F}_i^* + \mathbf{K}_i\mathbf{R}_{\mathbf{e},i}^{-1}\mathbf{K}_i^*\,.$$

Subtracting the above two equations yields the desired Riccati recursion for $\mathbf{P}_i$,

$$\mathbf{P}_{i+1} = \mathbf{F}_i\mathbf{P}_i\mathbf{F}_i^* + \mathbf{G}_i\mathbf{Q}_i\mathbf{G}_i - \mathbf{K}_i\mathbf{R}_{\mathbf{e},i}^{-1}\mathbf{K}_i^*\,.$$

Of course, the $\mathbf{K}_i$ and $\mathbf{R}_{\mathbf{e},i}$ can be computed in other ways as shown in *Algorithmic Variants of the Kalman Filter* via square-root arrays or, when the model has special structure, via the Chandrasekhar recursions.

References

1. S. Haykin, *Adaptive Filter Theory*, NJ: Prentice Hall, second edition, 1991.

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

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

4. S. J. Orfanidis, *Optimum Signal Processing*, McGraw-Hill, Inc., second edition, 1988.

5. P. Strobach, *Linear Prediction Theory*, Berlin Heidelberg: Springer-Verlag, 1990.

6. B. Widrow and S. D. Stearns, *Adaptive Signal Processing*, NY: Prentice-Hall, Inc., 1985.

7. N. Kalouptsidis and S. Theodoridis, *Adaptive System Identification and Signal Processing Algorithms*, NJ: Prentice Hall, 1993.

8. D. T. L. Lee, M. Morf, and B. Friedlander, Recursive least-squares ladder estimation algorithms. *IEEE Transactions on Circuits and Systems*, vol. CAS-28, no. 6, pp. 467-481, June 1981.

9. B. Porat, B. Friedlander, and M. Morf, Square root covariance ladder algorithms, *IEEE Transactions on Automatic Control*, vol. AC-27, no. 4, pp. 813-829, August 1982.

10. B. Friedlander, Lattice filters for adaptive processing, *Proceedings of the IEEE*, vol. 70, no. 8, pp. 829–867, August 1982.

11. H. Lev-Ari, T. Kailath, and J. Cioffi, Least squares adaptive lattice and transversal filters: A unified geometrical theory, *IEEE Transactions on Information Theory*, vol. IT-30, no. 2, pp. 222–236, March 1984.

12. J. Cioffi and T. Kailath, Fast recursive-least-squares transversal filters for adaptive filtering, *IEEE Transactions on Acoustics, Speech and Signal Processing*, vol. ASSP-32, pp. 304–337, April 1984.

13. S. T. Alexander, Fast adaptive filters: A geometrical approach, *IEEE Signal Processing Magazine*, vol. 3, no. 4, pp. 18–28, October 1986.

14. D. T. M. Slock and T. Kailath, Numerically stable fast transversal filters for recursive least squares adaptive filtering, *IEEE Transactions on Signal Processing*, vol. SP-39, no. 1, pp. 92–114, January 1991.

15. M. Morf, *Fast Algorithms for Multivariable Systems*, PhD thesis, Stanford University, Stanford, CA, 1974.

16. M. Morf, T. Kailath, and L. Ljung, Fast algorithms for recursive identification, *Proc. IEEE Conference on Decision and Control*, pp. 916–921, Clearwater Beach, FL, December 1976.

17. G. Carayannis, D. Manolakis, and N. Kalouptsidis, A fast sequential algorithm for least squares filtering and prediction, *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. ASSP-31, no. 6, pp. 1394–1402, December 1983.

18. I. K. Proudler, J. G. McWhirter, and T. J. Shepherd, Computationally efficient QR decomposition approach to least squares adaptive filtering, *IEE Proceedings*, vol. 138, no. 4, pp. 341–353, August 1991.

19. J. Cioffi, The fast adaptive rotor's RLS} algorithm, *IEEE Transactions on Acoustics, Speech and Signal Processing*, vol. ASSP-38, pp. 631–653, 1990.

20. P. A. Regalia and M. G. Bellanger, On the duality between fast QR methods and lattice methods in least squares adaptive filtering, *IEEE Transactions on Signal Processing*, vol. 39, no. 4, pp. 879–891, April 1991.

21. D. T. M. Slock, *Fast Algorithms for Fixed-Order Recursive Least-Squares Parameter Estimation*, PhD thesis, Stanford University, Stanford, CA, 1989.

22. B. Yang and J. F. Bohme, 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.

23. K. Zhao, *Order-Recursive Least Squares Adaptive Algorithms*, PhD thesis, Northeastern University, August 1992.

24. A. H. Sayed, *Displacement Structure in Signal Processing and Mathematics*, PhD thesis, Stanford University, Stanford, CA, August 1992.

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

26. D. Godard, Channel equalization using a Kalman filter for fast data transmission, *IBM J. Res. Develop.*, vol. 18, pp. 267–273, May 1974.

27. T. Kailath, *Lectures on Wiener and Kalman Filtering}. NY: Springer-Verlag, second edition, 1981.*

28. R. E. Kalman, A new approach to linear filtering and prediction problems, *Trans. ASME J. Basic Eng.*, vol. 82, pp. 34–45, March 1960.

29. T. Kailath, A view of three decades of linear filtering theory, *IEEE Transactions on Information Theory*, vol. IT-20, no. 2, pp. 146–181, March 1974.

30. B. D. O. Anderson and J. B. Moore, *Optimal Filtering*, NJ: Prentice-Hall

Inc., 1979.

31. A. H. Sayed and T. Kailath, Extended Chandrasekhar recursions, *IEEE Transactions on Automatic Control*, vol. AC-39, no. 3, pp. 619–623, March 1994.

32. B. H. Khalaj, A. H. Sayed, and T. Kailath, A unified approach to multichannel least-squares algorithms, *Proc. ICASSP*, vol. 5, pp. 523–526, Minneapolis, MN, April 1993.

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

34. 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.

35. 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.

36. M. Gentleman, Least squares computations by Givens transformations. *J. Inst. Math. Appl.*, vol. 12, pp. 329–336, 1973.

37. S. J. Hammarling, A note on modifications to the Givens plane rotation, *J. Inst. Math. Appl.*, vol. 13, pp. 215–218, 1974.

38. S. F. Hsieh, K. J. R. Liu, and K. Yao, A unified square-root-free approach for QRD-based recursive least-squares estimation, *IEEE Transactions on Signal Processing*, vol. SP-41, no. 3, pp. 1405–1409, March 1993.

39. C. F. Gauss, *Theory of the Motion of Heavenly Bodies*, NY: Dover, New York, 1963 (English translation of *Theoria Motus Corporum Coelestium, 1809).*

40. N. Wiener, *Extrapolation, Interpolation, and Smoothing of Stationary Time Series*, MA: MIT Press, 1942.

41. R. E. Kalman and R. S. Bucy, New results in linear filtering and prediction theory, *Trans. ASME J. Basic Eng.*, vol. 83, pp. 95–107, December 1961.

42. T. Kailath, *Linear Systems*, NJ: Prentice-Hall, 1980.

43. 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.

44. M. Morf, G. S. Sidhu, and T. Kailath, Some new algorithms for recursive estimation in constant, linear, discrete-time systems, *IEEE Transactions on Automatic Control*, vol. AC-19, no. 4, pp. 315–323, August 1974.

45. T. Kailath, A. C. Vieira, and M. Morf, Orthogonal transformation (square–root) implementations of the generalized Chandrasekhar and generalized Levinson algorithms, in *Lecture Notes in Control and Information Sciences*, A. Bensoussan and J. Lions, eds. New York: Springer-Verlag, vol. 32, pp. 81–91, 1978.

46. A.H. Sayed and T. Kailath, Structured matrices and fast RLS adaptive filtering, *Proc. 2nd IFAC Workshop on Algorithms and Architectures for Real-Time Control*, P. J. Fleming and W. H. Kwon, eds., Seoul, Korea: Pergamon Press, pp. 211–216, September 1992.

47. B. Friedlander, T. Kailath, M. Morf, and L. Ljung, Extended Levinson and Chandrasekhar equations for general discrete-time linear estimation problems, *IEEE Transactions on Automatic Control*, vol. AC-23, no. 4, pp. 653–659, August 1978.

48. 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.

49. C. T. Pan and R. J. Plemmons, Least-squares modifications with inverse factorizations: Parallel implications, *J. Comput. Appl. Math.*, vol. 27, pp. 109–127, 1989.

50. A. H. Sayed, H. Lev-Ari, and T. Kailath, Time-variant displacement structure and triangular arrays, to appear in *IEEE Transactions on Signal Processing*, vol. SP-42, no. 5, pp. 1052-1062, May 1994.

51. P. Park and T. Kailath, An extended inverse QR Algorithm, to appear in *Signal Processing.*

52. A. Houacine, Regularized fast recursive least squares algorithms for adaptive filtering, *IEEE Transactions on Signal Processing*, vol. SP-39, no. 4, pp. 860–870, April 1991.

53. B. Widrow and M. E. Hoff, Jr., Adaptive switching circuits, *IRE WESCON Conv. Rec.*, Pt. 4, pp. 96–104, 1960.

54. B. Hassibi, A. H. Sayed, and T. Kailath, LMS is H^{∞} optimal, *Proc. Conference on Decision and Control*, vol. 1, pp. 74–79, San Antonio, Texas, December 1993.

55. J. C. Doyle, K. Glover, P. Khargonekar, and B. Francis, State-space solutions to standard H_2 and H^{∞} control problems, *IEEE Transactions on Automatic Control*, vol. AC-34, no. 8, pp. 831–847, August 1989.

56. P.P. Khargonekar and K. M. Nagpal, Filtering and smoothing in an H^{∞}– setting, *IEEE Trans. on Automatic Control*, vol. AC-36, pp. 151–166, 1991.

57. A. H. Sayed and M. Rupp, Local and global optimality criteria for gradient-type algorithms, submitted for publication.

58. B. Hassibi, A. H. Sayed, and T. Kailath, Recursive linear estimation in Krein spaces - Part I: Theory, *Proc. Conference on Decision and Control*, vol. 4, pp. 3489–3494, San Antonio, Texas, December 1993.