A Derivation of a General Order, Multichannel, Fast Transversal, Recursive Least Squares Filter Algorithm

by

L.J. Faber
S.H. Ardalan
and S.T. Alexander

Center for Communications and Signal Processing
Department of Electrical Computer Engineering
North Carolina State University

February 1986

CCSP-TR-86/5

ABSTRACT

This report examines the derivation of a general order, multiple input version of the fast transversal recursive least squares filter algorithm. The new algorithm, in which the order (or number of taps) associated with each input channel is independent, is distinguished from the previously published multichannel form of fast transversal RLS algorithm, wherein each input channel is constrained to have the same order. This additional flexibility allows assignment of filter resources to particular channels, or independent assignment to the pole/zero estimators in an ARMA system identification application. A summary of the new algorithm is given, in proper order of execution. An operations count is also provided for each equation.

1. Introduction

Fast transversal recursive least squares (RLS) algorithms [1],[2] have filter update computational costs of order N, the filter length. Although these algorithms suffer from long-term instability problems, their convergence superiority in many applications, compared to the common LMS (gradient) algorithms, have made them an active area of research. The purpose of this paper is to present the derivation of a multichannel fast transversal recursive least squares algorithm with general order inputs. The approach in this derivation is to use the method of geometric projections onto a vector space defined by the input data. The application of this technique from linear algebra to the transversal filter case was first given by Cioffi and Kailath [1]. The algorithm in this paper is distinquished from the multichannel algorithm shown there in which the order of the filter for each channel was constrained to be equal; here the order of each channel is completely arbitrary. This property, which allows greater flexibility in some applications, causes certain difficulties in the derivation. The key to resolving these difficulties is the use of permutation matrices, similar to those used by Falconer and Ljung [3] in their application of the fast Kalman algorithm to adaptive equalization. The derivation in this paper is a generalization of the work by Ardalan [4], which described an application to a particular two channel system in which the (unit delayed) output signal is used as one input channel. This is called the ARMA or pole/zero estimation case, using the equation-error form for system identification [5]. The notation used herein is similar to that of [7] and [8].

1.1. Problem Description.

The goal of the analysis is to recursively determine the length N transversal filter $\mathbf{w}_N(n)$, in the system shown in Figure 1, such that the optimal estimate of the (joint) process $\{d(n)\}$ is formed by a transversal filtering of the two input processes $\{x(n)\}$ and $\{y(n)\}$. These processes are real data sequences where data is assumed zero for $n \leq 0$; this is called the prewindowed case. (Note: only two input channels are considered for notational simplicity, but the extension to the arbitrary p-channel case will usually be obvious.) Here optimality is defined in a cumulative or *exact least squares* sense; this can be explained as follows. The estimation error incurred at time i, based upon the filter at time n is defined

$$e(i|n) = d(i) - \hat{d}(i|n) = d(i) - \mathbf{z}_N^T(i)\mathbf{w}_N(n), \quad 1 \leq i \leq n, \tag{1.1}$$

where $\hat{d}(i|n)$ represents an estimate of $d(i)$ using the transversal filter of time n, and $\mathbf{z}_N(i)$ is a length N vector of past input samples from both channels:

$$(N\times 1) \quad \mathbf{z}_N(i) = [\mathbf{x}_L^T(i), \mathbf{y}_M^T(i)]^T, \quad \text{with} \quad N = L + M, \tag{1.2}$$

$(L\times 1)\quad \mathbf{x}_L(i) = [x(i),\, x(i-1),\, \cdots,\, x(i-L+1)]^T$, and

$(M\times 1)\quad \mathbf{y}_M(i) = [y(i),\, y(i-1),\, \cdots,\, y(i-M+1)]^T$.

The length N transversal filter $\mathbf{w}_N(n)$ is partitioned into subfilters of lengths L and M, corresponding to the respective input channels. Thus it is desired to find the

$$\underset{w.r.t.\ \mathbf{w}_N(n)}{\text{minimum}} \sum_{i=1}^{n} e^2(i|n). \tag{1.3}$$

This is the well known least squares (LS) problem, which can be succinctly expressed in vector notation.

1.2. Data Vectors and Matrices.

The basic concept in the method of geometric projections is to locate vector estimates within a subspace which is spanned by a set of vectors formed from the input data. Vectors are formed of all samples of the input channels from time 1 to time n:

$$(n\times 1)\quad \mathbf{x}(n) = [x(1),\, x(2),\, \cdots,\, x(n)]^T, \qquad \mathbf{y}(n) = [y(1),\, y(2),\, \cdots,\, y(n)]^T. \tag{1.4}$$

A similar vector is defined with samples of the joint process $\{d(n)\}$. It is also convenient to define a 'pinning vector' [1]

$$(n\times 1)\quad \pi(n) = [0,\, \cdots,\, 0,\, 1]^T. \tag{1.5}$$

This is the unit vector in the direction of the newest data sample, and will prove useful later.

A vector $\mathbf{z}(n)$ is now defined with columns consisting of the input vectors. Additionally, $\mathbf{z}_b(n)$ is defined with columns of input vectors that are each time delayed by the order of the filter assigned to that channel:

$$(n\times 2)\quad \mathbf{z}(n) = [\mathbf{x}(n),\, \mathbf{y}(n)], \qquad \mathbf{z}_b(n) = [z^{-L}\mathbf{x}(n),\, z^{-M}\mathbf{y}(n)]. \tag{1.6}$$

Here z^{-L} for example, is the L-unit time delay operator. Note that the $\mathbf{z}_b(n)$ is still of length n but with leading elements of zero, since the data is prewindowed. It is also convenient to define the last rows of these vectors as

$$(1\times 2)\quad z(n) = [x(n),\, y(n)] = \pi^T(n)\mathbf{z}(n), \qquad z_b(n) = [x(n-L),\, y(n-M)]. \tag{1.7}$$

Note that these rows can be generated by inner products with $\pi(n)$. If a matrix is now defined by the following relationships

$$
\begin{aligned}
(n\times N)\quad \mathbf{Z}_0(n) &= [\mathbf{X}_0(n),\ \mathbf{Y}_0(n)] \\
&= [\mathbf{x}(n),\ z^{-1}\mathbf{x}(n),\ \cdots,\ z^{-L+1}\mathbf{x}(n),\ \mathbf{y}(n),\ \cdots,\ z^{-M+1}\mathbf{y}(n)] \\
&= \left[\begin{array}{cccc|cccc}
x(1) & 0 & \cdots & 0 & y(1) & 0 & \cdots & 0 \\
x(2) & x(1) & & \vdots & y(2) & y(1) & & \vdots \\
\vdots & \vdots & & x(1) & \vdots & \vdots & & y(1) \\
\vdots & \vdots & & \vdots & \vdots & \vdots & & \vdots \\
x(n) & x(n-1) & \cdots & x(n-L+1) & y(n) & y(n-1) & \cdots & y(n-M+1)
\end{array}\right]
\end{aligned}
\tag{1.8}
$$

then a vector representing the estimation error resulting from the transversal filter $\mathbf{w}_N(n)$ can be defined

$$
(n\times 1)\quad \mathbf{e}(n\,|n) = \mathbf{d}(n) - \hat{\mathbf{d}}(n\,|n) = \mathbf{d}(n) - \mathbf{Z}_0(n)\mathbf{w}_N(n), \tag{1.9}
$$

where each element in $\mathbf{e}(n\,|n)$ corresponds to a scalar error from (1.1). A row of $\mathbf{Z}_0(n)$ thus represents the data in the transversal filter $\mathbf{w}_N(n)$ at a particular time; the final row of $\mathbf{Z}_0(n)$, which represents the latest data set in the filter, is $\mathbf{z}_N^T(n)$ defined in (1.2).

Other useful matrices can be defined. If each column of $\mathbf{Z}_0(n)$ is unit delayed:

$$
\begin{aligned}
(n\times N)\quad \mathbf{Z}_1(n) &= [\mathbf{X}_1(n),\ \mathbf{Y}_1(n)] = [z^{-1}\mathbf{X}_0(n),\ z^{-1}\mathbf{Y}_0(n)] \\
&= [z^{-1}\mathbf{x}(n),\ \cdots,\ z^{-L}\mathbf{x}(n),\ z^{-1}\mathbf{y}(n),\ \cdots,\ z^{-M}\mathbf{y}(n)]\,.
\end{aligned}
\tag{1.10}
$$

Consider the $\mathbf{Z}_0(n)$ matrix augmented with one additional column for each of the input channels, representing the addition of a tap for each channel subfilter:

$$
\begin{aligned}
(n\times N+2)\quad \mathbf{Z}_+(n) &= [\mathbf{X}_0(n),\ z^{-L}\mathbf{x}(n),\ \mathbf{Y}_0(n),\ z^{-M}\mathbf{y}(n)] \\
&= [\mathbf{x}(n),\ \mathbf{X}_1(n),\ \mathbf{y}(n),\ \mathbf{Y}_1(n)]
\end{aligned}
\tag{1.11}
$$

Note that this augmented matrix can be described in two ways, using the previously defined matrices. The final row of $\mathbf{Z}_+(n)$ defines a vector, which can also be decomposed in two ways:

$$
\begin{aligned}
(N+2\times 1)\quad \mathbf{z}_{N+2}(n) &= [\mathbf{x}_L^T(n),\ x(n-L),\ \mathbf{y}_M^T(n),\ y(n-M)]^T \\
&= [x(n),\ \mathbf{x}_L^T(n-1),\ y(n),\ \mathbf{y}_M^T(n-1)]^T\,.
\end{aligned}
\tag{1.12}
$$

1.3. Projection Operators.

From (1.9), each element (tap) of $\mathbf{w}_N(n)$ defines a weight applied to a particular column of $\mathbf{Z}_0(n)$; *i.e.*, the estimate vector $\hat{\mathbf{d}}(n\,|n)$ is a *linear combination* of the columns. Equivalently, the estimate vector must lie in the column space of $\mathbf{Z}_0(n)$. The columns span a subspace of R^n, the space of all real-valued, length n vectors. These key

observations lead to the concept of *projection* onto a subspace as a means of locating the least squares error estimate of a vector within that subspace [9]. It is a well known result of linear algebra that the minimum length (norm) error vector corresponds to that estimate of a vector found by taking its geometric (perpendicular) projection onto the subspace of input vectors. This is shown conceptually in Figure 2 for a two-dimensional subspace. Note that the error vector $\mathbf{e}(n\,|n)$ and the estimate vector $\hat{\mathbf{d}}(n\,|n)$ are perpendicular; this is a general orthogonality principle for LS solutions. A operator matrix $\mathbf{P}$ can be formed having the desired property of projection onto a particular subspace. The operator which projects onto $\mathbf{Z}_0(n)$ is found by solving the matrix equation:

$$\min_{w.r.t.\ \mathbf{w}_N(n)} \mathbf{e}^T(n\,|n)\mathbf{e}(n\,|n) = \min_{w.r.t.\ \mathbf{w}_N(n)} \epsilon(n)\,. \tag{1.13}$$

Differentiating $\epsilon(n)$ with respect to $\mathbf{w}_N(n)$ and equating to zero yields the LS prediction

$$\hat{\mathbf{d}}(n) = \mathbf{Z}_0(n)\mathbf{w}_N(n) = \mathbf{Z}_0(n)[\mathbf{Z}_0^T(n)\mathbf{Z}_0(n)]^{-1}\mathbf{Z}_0^T(n)\,\mathbf{d}(n) = \mathbf{P}_0(n)\,\mathbf{d}(n)\,, \tag{1.14}$$

where it is assumed for simplicity that the inverse exists. The projection operator can thus be defined

$$(n\times n)\quad \mathbf{P}_0(n) = \mathbf{Z}_0(n)[\mathbf{Z}_0^T(n)\,\mathbf{Z}_0(n)]^{-1}\,\mathbf{Z}_0^T(n)\,. \tag{1.15}$$

Note that the least squares estimate of an arbitrary vector is thus obtained by premultiplying by a matrix that is a function of only the input data vectors! The operator $\mathbf{P}_1(n)$ is similarly defined for the $\mathbf{Z}_1(n)$ subspace.

An operator can also be found to generate the LS error vector. Substituting (1.13) into (1.9) gives

$$\mathbf{e}(n\,|n) = \mathbf{d}(n) - \mathbf{P}_0(n)\mathbf{d}(n) = [\mathbf{I} - \mathbf{P}_0(n)]\,\mathbf{d}(n) = \mathbf{P}_0^{\perp}(n)\,\mathbf{d}(n)\,. \tag{1.16}$$

This defines the orthogonal projection operator

$$\begin{aligned}&(n\times n)\quad \mathbf{P}_0^{\perp}(n) = \mathbf{I} - \mathbf{P}_0(n);\quad \text{similarly,}\\ &(n\times n)\quad \mathbf{P}_1^{\perp}(n) = \mathbf{I} - \mathbf{P}_1(n)\,.\end{aligned} \tag{1.17}$$

These operators generate the projection of a vector perpendicular to their defining subspaces.

1.4. Transversal Filter Operators.

From (1.6) and (1.9), the optimal transversal filter can be written

$$\mathbf{w}_N(n) = [\mathbf{Z}_0^T(n)\mathbf{Z}_0(n)]^{-1}\mathbf{Z}_0^T(n)\,\mathbf{d}(n)\,. \tag{1.18}$$

A transversal filter operator (matrix) can then be defined

$$(N\times n)\quad \mathbf{K}_0(n) = [\mathbf{Z}_0^T(n)\,\mathbf{Z}_0(n)]^{-1}\,\mathbf{Z}_0^T(n)\,. \tag{1.19}$$

By premultiplication, this operator creates the transversal filter (for the input data) which would generate the unique LS estimate of a vector on the subspace of $\mathbf{Z}_0(n)$. Similar operators can be defined on the other subspaces shown earlier:

$$\begin{aligned}(N\times n)\quad & \mathbf{K}_1(n) = [\mathbf{Z}_1^T(n)\,\mathbf{Z}_1(n)]^{-1}\,\mathbf{Z}_1^T(n), \quad \text{and} \\ (N+2\times n)\quad & \mathbf{K}_+(n) = [\mathbf{Z}_+^T(n)\,\mathbf{Z}_+(n)]^{-1}\,\mathbf{Z}_+^T(n)\,.\end{aligned} \tag{1.20}$$

Several transversal filters can be generated using these operators on previously defined vectors, including a forward predictor for the input channels, a backward predictor, and a gain filter. These filters, and their associated prediction errors and residuals (squared error vector norms) are summarized with definitions in Table I. The need for these quantities will be seen during the algorithm derivation. Most are not directly evaluated from their definitions; instead, relationships between the quantities are exploited to allow tremendous savings in computations. Of particular interest, the gain filter $\mathbf{g}_N(n)$ is the LS filter on $\mathbf{Z}_0(n)$ for predicting the pinning vector $\pi(n)$. The associated prediction error $\gamma(n)$ (see I.18) was shown in [6] to be the squared cosine of the angle between $\mathbf{z}(n)$ and its estimate in the subspace of $\mathbf{Z}_1(n)$. Thus, $\gamma(n)$ is a measure of the amount of new information (innovation) in the latest data sample, and is a key parameter in the derivation.

1.5. Permutation Operators.

The permutation operators S defined here are row-permuted identity matrices, such that pre-multiplication of a vector by S will cause rearrangement of the vector's rows. Such matrices have the important property of orthogonality: $\mathbf{S}\mathbf{S}^T = \mathbf{I}$ or $\mathbf{S}^T = \mathbf{S}^{-1}$. The $(N+2)\times(N+2)$ 'forward' and 'backward' permutation operator matrices, $\mathbf{S}_f$ and $\mathbf{S}_b$ respectively, are defined by their effect on the augmented vector $\mathbf{z}_{N+2}(n)$:

$$\begin{aligned}\mathbf{S}_f\,\mathbf{z}_{N+2}(n) &= [x(n),\, y(n),\, \mathbf{x}_L^T(n-1),\, \mathbf{y}_M^T(n-1)]^T = [z(n),\, \mathbf{z}_N^T(n-1)]^T, \quad \text{and} \\ \mathbf{S}_b\,\mathbf{z}_{N+2}(n) &= [\,\mathbf{x}_L^T(n),\, \mathbf{y}_M^T(n),\, x(n-L),\, y(n-M)]^T = [\mathbf{z}_N^T(n),\, z_b(n)]^T\,.\end{aligned} \tag{1.21}$$

From these definitions it is easy to also describe matrix column rearrangement:

$$\begin{aligned}\mathbf{Z}_+(n)\mathbf{S}_f^T &= [\mathbf{x}(n),\, \mathbf{y}(n),\, \mathbf{X}_1(n),\, \mathbf{Y}_1(n)] = [\mathbf{z}(n),\, \mathbf{Z}_1(n)], \quad \text{and} \\ \mathbf{Z}_+(n)\mathbf{S}_b^T &= [\mathbf{X}_0(n),\, \mathbf{Y}_0(n),\, z^{-L}\mathbf{x}(n),\, z^{-M}\mathbf{y}(n)] = [\mathbf{Z}_0(n),\, \mathbf{z}_b(n)]\,.\end{aligned} \tag{1.22}$$

The need for permutation operators arises in the multichannel derivation because of the form of the updates used for the projection and transversal filter operators. These updates, shown in the next section, require that augmenting vectors be appended to either

the left or right of the data matrix; however, from (1.11) it can be seen that augmenting columns naturally appear 'inside' the data matrix. Thus, permutation operators are used to rearrange these columns so that the update forms may be properly applied.

Using the property of orthogonality of the permutation operator, the transversal filter operator for the permuted, augmented subspace can be deduced:

$$\begin{aligned}
&(N+2\times n)\quad \mathbf{K}_{+}^{f}(n) = [\mathbf{S}_f \mathbf{Z}_{+}^{T}(n)\mathbf{Z}_{+}(n)\mathbf{S}_f^{T}]^{-1}\mathbf{S}_f \mathbf{Z}_{+}^{T}(n) = \mathbf{S}_f \mathbf{K}_{+}(n);\ \text{similarly,} \\
&(N+2\times n)\quad \mathbf{K}_{+}^{b}(n) = \mathbf{S}_b \mathbf{K}_{+}(n).
\end{aligned} \tag{1.23}$$

Pre-multiplying a vector with a transversal filter operator $\mathbf{K}$ yields the transversal filter which would generate the LS error estimate of that vector, in the defining subspace of the operator. Each row of $\mathbf{K}$ transforms the vector into a particular filter tap. Thus, permutation of the rows of $\mathbf{K}$ is equivalent to permutation of the elements of the transversal filter. This concept leads to the following equations for permuted gain filters (refer to Table I):

$$\begin{aligned}
&(N+2\times 1)\quad \mathbf{g}_{N+2}^{f}(n) = \mathbf{S}_f \mathbf{g}_{N+2}(n) = \mathbf{S}_f \mathbf{K}_{+}(n)\pi(n) = \mathbf{K}_{+}^{f}(n)\pi(n);\ \text{similarly,} \\
&(N+2\times 1)\quad \mathbf{g}_{N+2}^{b}(n) = \mathbf{S}_b \mathbf{g}_{N+2}(n) = \mathbf{K}_{+}^{b}(n)\pi(n).
\end{aligned} \tag{1.24}$$

Using the orthogonality property of $\mathbf{S}$, these can be combined to show:

$$\mathbf{g}_{N+2}^{b}(n) = \mathbf{S}_b \mathbf{S}_f^{T} \mathbf{g}_{N+2}^{f}(n) \tag{1.25}$$

2. Generalized Updates of Operators.

2.1. Projection Operator Order Update.

In order to derive the recursive algorithm, it is necessary to consider the problem of updating the projection operator when new columns are added to the defining subspace. In general, the projection operator onto a subspace U is defined

$$P_U = U<U,U>^{-1}U^T = U(U^TU)^{-1}U^T \tag{2.1}$$

where $<\cdot,\cdot>$ represents the inner product on the subspace U. The properties of symmetry ($P_U^T = P_U$) and idempotence ($P_U^2 = P_U$) are readily verified for this operator. The general perpendicular projection operator can be defined:

$$P_U^{\perp} = I - P_U, \tag{2.2}$$

also with the properties of symmetry and idempotence. Augmenting the subspace U with columns V yields the following order updates for the projection operators:

$$\begin{aligned} P_{U,V} &= P_U + P_{\{P_U^{\perp}V\}} = P_U + P_U^{\perp}V<V, P_U^{\perp}V>^{-1}V^TP_U^{\perp}, \\ P_{U,V}^{\perp} &= I - P_{U,V} = P_U^{\perp} - P_U^{\perp}<V, P_U^{\perp}V>^{-1}V^TP_U^{\perp}. \end{aligned} \tag{2.3}$$

The projection operator is updated by incorporating only the new information from the augmented columns, namely the subspace spanned by the new columns which is orthogonal to the old subspace. Pre- and post-multiplying (2.3) by arbitrary vectors r^T and w gives the useful identity

$$<r, P_{U,V}^{\perp}w> = <r, P_U^{\perp}w> - <r, P_U^{\perp}V><V, P_U^{\perp}V>^{-1}<V, P_U^{\perp}w> \tag{2.4}$$

2.2. Transversal Filter Operator Order Update.

The general transversal filter operator defined on the subspace U is

$$K_U = <U,U>^{-1}U^T. \tag{2.5}$$

The following properties are easily demonstrated:

$$K_UU = I, \quad UK_U = P_U, \quad K_UP_U = K_U. \tag{2.6}$$

From (2.6) and assuming $U_{n\times N}$, $V_{n\times p}$

$$K_{U,V}[U_{n\times N}, V_{n\times p}] = I_{N+p} = \begin{bmatrix} I_N & 0_{N\times p} \\ 0_{p\times N} & I_p \end{bmatrix}.$$

Thus, using (2.3), the update for post-appended columns is

$$K_{U,V} = K_{U,V}P_{U,V} = \begin{bmatrix} K_U \\ 0 \end{bmatrix} + \begin{bmatrix} -K_U V \\ I \end{bmatrix} \langle V, P_U^{\perp} V \rangle^{-1} V^T P_U^{\perp} . \tag{2.7a}$$

Similarly, it can be shown for pre-appended columns

$$K_{V,U} = \begin{bmatrix} 0 \\ K_U \end{bmatrix} + \begin{bmatrix} I \\ -K_U V \end{bmatrix} \langle V, P_U^{\perp} V \rangle^{-1} V^T P_U^{\perp} . \tag{2.7b}$$

2.3. Time Updates of Operators.

Using (2.3) and (2.7) with $U = Z_0(n)$ or $U = Z_1(n)$ and $V = \pi(n)$, it can be shown that

$$\begin{aligned} P_{0,\pi}(n) &= \begin{bmatrix} P_0(n-1) & 0 \\ 0^T & 1 \end{bmatrix} \quad \text{and} \quad P_{0,\pi}^{\perp}(n) = \begin{bmatrix} P_0^{\perp}(n-1) & 0 \\ 0^T & 0 \end{bmatrix} \\ P_{1,\pi}(n) &= \begin{bmatrix} P_1(n-1) & 0 \\ 0^T & 1 \end{bmatrix} \quad \text{and} \quad P_{1,\pi}^{\perp}(n) = \begin{bmatrix} P_1^{\perp}(n-1) & 0 \\ 0^T & 0 \end{bmatrix} . \end{aligned} \tag{2.8}$$

Also, the transversal filter operator for the subspace augmented with $\pi(n)$ can be expressed

$$K_{0,\pi}(n) = \begin{bmatrix} K_0(n-1) & 0 \\ r_0^T & 1 \end{bmatrix}, \quad K_{1,\pi}(n) = \begin{bmatrix} K_1(n-1) & 0 \\ r_1^T & 1 \end{bmatrix} . \tag{2.9}$$

Here the r^T are $(n-1)$ element vectors not needed in the following derivation. These two sets of equations, along with (2.3) and (2.7) can be systematically applied to complete the vector space derivation of the fast transversal algorithm.

3. Derivation of the Fast Multichannel Algorithm

Now that the preliminary updates and definitions have been determined, it is possible to derive the solution to the LS problem posed initially. The derivation, necessarily abbreviated here, basically only requires substitutions into the update forms, and reduction using the definitions. The derivation begins with a recursive update of the joint prediction filter, and then systematically finds recursions for other variables as they appear. The algorithm is summarized in Table II. in proper order of execution. An operation count is also provided for each equation.

3.1. Joint Prediction Filter

From (2.7a) with $\mathbf{U} = \mathbf{Z}_0(n)$, $\mathbf{V} = \pi(n)$,

$$\mathbf{K}_{0,\pi}(n) = \begin{bmatrix} \mathbf{K}_0(n) \\ \mathbf{0} \end{bmatrix} + \begin{bmatrix} -\mathbf{K}_0(n)\pi(n) \\ \mathbf{I} \end{bmatrix} <\pi(n), \mathbf{P}_0^{\perp}(n)\pi(n)>^{-1} \pi^T(n)\mathbf{P}_0^{\perp}(n). \tag{3.1}$$

Post-multiply (3.1) by $\mathbf{d}(n)$, and use expansion (2.9) with definitions (I.4), (I.18):

$$\begin{bmatrix} \mathbf{K}_0(n-1) & \mathbf{0} \\ \mathbf{r}^T & 1 \end{bmatrix} \begin{bmatrix} \mathbf{d}(n-1) \\ d(n) \end{bmatrix} = \begin{bmatrix} \mathbf{K}_0(n) \\ \mathbf{0} \end{bmatrix} \mathbf{d}(n) + \begin{bmatrix} -\mathbf{g}_N(n) \\ 1 \end{bmatrix} \gamma^{-1}(n) <\pi(n), \mathbf{P}_0^{\perp}(n)\mathbf{d}(n)>$$

Take just the upper partitions and use (I.1), (I.13), (I.5)

$$\mathbf{w}_N(n) = \mathbf{w}_N(n-1) + \mathbf{c}_N(n)e(n|n). \tag{3.2}$$

3.2. Joint Process Error.

Pre-multiply (3.2) by $\mathbf{z}_N^T(n)$ and subtract from $d(n)$

$$d(n) - \mathbf{z}_N^T(n)\mathbf{w}_N(n) = d(n) - \mathbf{z}_N^T(n)\mathbf{w}_N(n-1) - \mathbf{z}_N^T(n)\mathbf{c}_N(n)e(n|n). \tag{3.3}$$

With definitions (I.12), (I.13)

$$e(n|n) = \gamma(n)e(n|n-1). \tag{3.4}$$

3.3. Gain Filter.

From (1.22), with $\mathbf{V} = \mathbf{z}(n)$, $\mathbf{U} = \mathbf{Z}_1(n)$, the subspace $[\mathbf{V}, \mathbf{U}]$ is $\mathbf{Z}_+(n)\mathbf{S}_f^T$ and the associated TF operator is $\mathbf{K}_+^f(n)$. Use (2.7b) and post-multiply by $\pi(n)$,

$$\mathbf{K}_+^f(n)\pi(n) = \begin{bmatrix} \mathbf{0} \\ \mathbf{K}_1(n) \end{bmatrix} \pi(n) + \begin{bmatrix} \mathbf{I} \\ -\mathbf{K}_1(n)\mathbf{z}(n) \end{bmatrix} <\mathbf{z}(n), \mathbf{P}_1^{\perp}(n)\mathbf{z}(n)>^{-1} <\mathbf{z}(n), \mathbf{P}_1^{\perp}(n)\pi(n)>$$

It is readily shown that $\mathbf{K}_1(n)\pi(n) = \mathbf{g}_N(n-1)$; thus, use definitions (I.2), (I.8), (I.21), and

(I.15)

$$\mathbf{g}^{f}_{N+2}(n) = \begin{bmatrix} 0 \\ \mathbf{g}_N(n-1) \end{bmatrix} + \begin{bmatrix} \mathbf{I}_2 \\ -\mathbf{f}_N(n) \end{bmatrix} \epsilon_f^{-1}(n) e_f^T(n\,|\,n)\,. \tag{3.5}$$

Similarly, from (2.7a) with $\mathbf{U} = \mathbf{Z}_0(n)$, $\mathbf{V} = \mathbf{z}_b(n)$, post-multiply by $\pi(n)$

$$\mathbf{K}^{b}_{+}(n)\pi(n) = \begin{bmatrix} \mathbf{K}_0(n) \\ \mathbf{0} \end{bmatrix} \pi(n) + \begin{bmatrix} -\mathbf{K}_0(n)\mathbf{z}_b(n) \\ \mathbf{I} \end{bmatrix} <\mathbf{z}_b(n), \mathbf{P}_0^{\perp}(n)\mathbf{z}_b(n)>^{-1} <\mathbf{z}_b(n), \mathbf{P}_0^{\perp}(n)\pi(n)>$$

From (I.10), (I.4), (I.3), (I.17), and (I.22)

$$\mathbf{g}^{b}_{N+2}(n) = \begin{bmatrix} \mathbf{g}_N(n) \\ \mathbf{0} \end{bmatrix} + \begin{bmatrix} -\mathbf{b}_N(n) \\ \mathbf{I}_2 \end{bmatrix} \epsilon_b^{-1}(n) e_b^T(n\,|\,n) \tag{3.6}$$

or with (1.25)

$$\begin{bmatrix} \mathbf{g}_N(n) \\ \mathbf{0} \end{bmatrix} = \mathbf{S}_b \mathbf{S}_f^T \mathbf{g}^{f}_{N+2}(n) + \begin{bmatrix} \mathbf{b}_N(n) \\ -\mathbf{I}_2 \end{bmatrix} \epsilon_b^{-1}(n) e_b^T(n\,|\,n)\,. \tag{3.7}$$

From (3.6) we define the last (2×1) element of $\mathbf{g}^{b}_{N+2}(n)$ as

$$g^{b}_{+}(n) = \epsilon_b^{-1}(n) e_b^T(n\,|\,n)\,. \tag{3.8a}$$

Also, with (I.10), the last element of $\mathbf{c}^{b}_{N+2}(n)$ is defined

$$c^{b}_{+}(n) = \gamma_{+}^{-1}(n)\epsilon_b^{-1}(n) e_b^T(n\,|\,n)\,. \tag{3.8b}$$

3.4. Forward Prediction Filter.

Use (2.7a) with $\mathbf{U} = \mathbf{Z}_1(n)$, $\mathbf{V} = \pi(n)$ and expansion (2.9); post-multiply by $\mathbf{z}(n)$ and take only top partition.

$$\mathbf{f}_N(n) = \mathbf{f}_N(n-1) + \mathbf{c}_N(n-1) e_f(n\,|\,n)\,. \tag{3.10}$$

3.5. Forward Prediction Error.

Use (3.10), pre-multiply by $\mathbf{z}_N^T(n-1)$, and subtract from $z(n)$.

$$z(n) - \mathbf{z}_N^T(n-1)\mathbf{f}_N(n) = z(n) - \mathbf{z}_N^T(n-1)\mathbf{f}_N(n-1) - \mathbf{z}_N^T(n-1)\mathbf{c}_N(n-1) e_f(n\,|\,n)\,. \tag{3.11}$$

With definitions

$$e_f(n\,|\,n) = \gamma(n-1) e_f(n\,|\,n-1)\,. \tag{3.12}$$

3.6. Forward Prediction Residual.

Use (2.4) with $\mathbf{U} = \mathbf{Z}_1(n)$, $\mathbf{V} = \pi(n)$, and $\mathbf{r}^T = \mathbf{w} = \mathbf{z}(n)$,

$$\begin{aligned}<\mathbf{z}(n), \mathbf{P}^{\perp}_{1,\pi}(n)\mathbf{z}(n)> &= <\mathbf{z}(n), \mathbf{P}^{\perp}_{1}(n)\mathbf{z}(n)> - <\mathbf{z}(n), \mathbf{P}^{\perp}_{1}(n)\pi(n)> \\ &\quad \cdot <\pi(n), \mathbf{P}^{\perp}_{1}(n)\pi(n)>^{-1} <\pi(n), \mathbf{P}^{\perp}_{1}(n)\mathbf{z}(n)>\end{aligned} \tag{3.13}$$

Use (2.8) and partition $\mathbf{z}(n) = [\mathbf{z}^T(n-1), z^T(n)]^T$,

$$\epsilon_f(n) = \epsilon_f(n-1) + e_f^T(n|n-1)e_f(n|n). \tag{3.14}$$

3.7. Backward Prediction Filter.

Use (2.7b) with $\mathbf{V} = \pi(n)$, $\mathbf{U} = \mathbf{Z}_0(n)$, postmultiply by $\mathbf{z}_b(n)$.

$$\mathbf{b}_N(n) = \mathbf{b}_N(n-1) + \mathbf{c}_N(n)e_b(n|n). \tag{3.16}$$

3.8. Backward Prediction Error.

Pre-multiply (3.16) by $\mathbf{z}_N^T(n)$ and subtract from $z_b(n)$

$$z_b(n) - \mathbf{z}_N^T(n)\mathbf{b}_N(n) = z_b(n) - \mathbf{z}_N^T(n)\mathbf{b}_N(n-1) - \mathbf{z}_N^T(n)\mathbf{c}_N(n)e_b(n|n). \tag{3.17}$$

With definitions

$$e_b(n|n) = \gamma(n)e_b(n|n-1). \tag{3.18}$$

3.9. Backward Prediction Residual.

Use (2.4), with $\mathbf{U} = \mathbf{Z}_0(n)$, $\mathbf{V} = \pi(n)$, and $\mathbf{r}^T = \mathbf{w} = \mathbf{z}_b(n)$; use (2.8) and partition $\mathbf{z}_b(n) = [\mathbf{z}_b^T(n-1), z_b^T(n)]^T$ to give

$$\epsilon_b(n) = \epsilon_b(n-1) + e_b^T(n|n-1)e_b(n|n). \tag{3.20}$$

3.10. Gain Error.

Use (I.19) and substitute (1.24a)

$$\gamma_+(n) = 1 - \mathbf{z}_{N+2}^T(n)\mathbf{S}_f^T\mathbf{g}_{N+2}^f(n).$$

Use (3.5), and note that $\mathbf{z}_{N+2}^T(n)\mathbf{S}_f^T = [z(n), \mathbf{z}_N^T(n-1)]$. Definitions (I.15), (I.18) provide

$$\gamma_+(n) = \gamma(n-1) - e_f(n|n)\epsilon_f^{-1}(n)\,e_f^T(n|n). \tag{3.21}$$

Similarly, use (I.19) and substitute (1.24b),

$$\gamma_+(n) = 1 - z_{N+2}^T(n) S_b^T g_{N+2}^b(n)$$

Now, use (3.6) and note $z_{N+2}^T(n) S_b^T = [z_N^T(n),\ z_b(n)]$,

$$\gamma(n) = \gamma_+(n) + e_b(n|n)\epsilon_b^{-1}(n)\, e_b^T(n|n)\ . \tag{3.22}$$

Thus, use (3.8) and (3.18),

$$\gamma(n) = \gamma_+(n)[1 - \gamma_+(n) e_b(n|n-1) c_+^b(n)]^{-1}\ . \tag{3.23}$$

3.11. Normalized Gain Filter.

Substitute (3.10) into (3.5); use (3.22), and divide through by $\gamma_+(n)$

$$c_{N+2}^f(n) = \begin{bmatrix} 0 \\ c_N(n-1) \end{bmatrix} + \begin{bmatrix} I_2 \\ -f_N(n-1) \end{bmatrix} \epsilon_f^{-1}(n) e_f^T(n|n) \gamma_+^{-1}(n)\ . \tag{3.24}$$

Similarly, use (3.6), substitute (3.16), and divide by $\gamma_+(n)$

$$\begin{bmatrix} c_N(n) \\ 0 \end{bmatrix} = c_{N+2}^b(n) + \begin{bmatrix} b_N(n-1) \\ -I_2 \end{bmatrix} c_+^b(n)\ . \tag{3.25}$$

3.12. Simpler BPE Update.

Instead of using the definition (I.16) to evaluate $e_b(n|n-1)$, requiring $2N$ multiplies, a simpler form can be found. Post-multiply (3.20) by (3.8b) to yield

$$\epsilon_b(n) c_+^b(n) = \gamma_+^{-1}(n) e_b^T(n|n) = \epsilon_b(n-1) c_+^b(n) + e_b^T(n|n) e_b(n|n-1) c_+^b(n)\ . \tag{3.26}$$

Thus, with simple algebra,

$$\epsilon_b(n-1) c_+^b(n) = \gamma_+^{-1}(n) e_b^T(n|n)[1 - \gamma_+(n) e_b(n|n-1) c_+^b(n)]\ , \tag{3.27}$$

but from (3.23), the term in brackets is equal to $\gamma^{-1}(n)\gamma_+(n)$. With (3.18),

$$e_b^T(n|n-1) = \epsilon_b(n-1)\, c_+^b(n)\ . \tag{3.28}$$

This form requires only p^2 multiplies, where p is the number of channels.

4. References

[1] J.M. Cioffi and T. Kailath, "Fast, Recursive-Least-Squares Transversal Filters for Adaptive Filtering," *IEEE Trans. on Acoustics, Speech and Sig. Proc.,* Vol. ASSP-32, No. 2, April 1984, pp. 304-321.

[2] G. Carayannis, D.G. Manolakis, and N. Kalouptsidis, "A Fast Sequential Algorithm for Least-Squares Filtering and Prediction," *IEEE Trans. on Acoustics, Speech, and Sig. Proc.,* Vol. ASSP-31, No. 6, pp. 1394, 1402, Dec. 1983.

[3] D.D. Falconer and L. Ljung, "Application of Fast Kalman Estimation to Adaptive Equalization," *IEEE Trans. on Comm.,* Vol. COM-26, No. 10, pp. 1439-1446, Oct. 1978.

[4] S.H. Ardalan, "Derivation of the Fast Pole-Zero (ARMA) Recursive Least Squares Algorithm Using Geometric Projections," Proceedings, Intl. Conf. on Acoust., Speech and Sig. Proc., ICASSP 1986, Tokyo, Japan.

[5] K.J. Astrom and P. Eykhoff, "System Identification--A Survey", *Automatica,* Vol. 7, pp. 123-162, 1971.

[6] D.T. Lee, M. Morf, B. Friedlander, "Recursive Least Squares Ladder Estimation Algorithms," *IEEE Trans. on Acoustics, Speech, and Sig. Proc., Vol. ASSP-29, No. 3, pp. 627-641, June 1981.*

[7] M.L. Honig and D.G. Messerschmidt, *Adaptive Filters,* Kluwer, Boston, MA, 1985.

[8] S.T. Alexander, *Adaptive Signal Processing,* Springer-Verlag, New York, NY, (to be published 1986).

[9] B. Noble and J.W. Daniel, *Applied Linear Algebra,* Prentice-Hall, Inc., Englewood Cliffs, N.J., 1977.

Faber, Ardalan, Alexander
Center Communic. and Sig. Process.
North Carolina State University
Raleigh, NC 27695 (919) 737-2338

"Derivation of a General Order, Multichannel, Fast Transversal, Recursive Least Squares Filter Algorithm"
Area: Digital Signal Processing

Table I.
Summary of Defining Relationships for
Two-Channel, General Order Derivation

EQ	DIM	DEFINITION	
		Transversal Filters	
I.1	$N\times 1$	$\mathbf{w}_N(n) = \mathbf{K}_0(n)\mathbf{d}(n)$	*joint predictor*
2	$N\times 2$	$\mathbf{f}_N(n) = \mathbf{K}_1(n)\mathbf{z}(n)$	*forward predictor*
3	$N\times 2$	$\mathbf{b}_N(n) = \mathbf{K}_0(n)\mathbf{z}_b(n)$	*backward predictor*
4	$N\times 1$	$\mathbf{g}_N(n) = \mathbf{K}_0(n)\pi(n)$	*gain (unit predictor)*
5	$N\times 1$	$\mathbf{c}_N(n) = \gamma^{-1}(n)\mathbf{g}_N(n)$	*normalized gain*
6	$N+2\times 1$	$\mathbf{g}_{N+2}(n) = \mathbf{K}_+(n)\pi(n)$	*augmented gain*
7	$N+2\times 1$	$\mathbf{c}_{N+2}(n) = \gamma_+^{-1}(n)\mathbf{g}_{N+2}(n)$	*norm. augm. gain*
8	$N+2\times 1$	$\mathbf{g}^f_{N+2}(n) = \mathbf{K}^f_+(n)\pi(n)$	*forward shifted augm. gain*
9	$N+2\times 1$	$\mathbf{c}^f_{N+2}(n) = \gamma_+^{-1}(n)\mathbf{g}^f_{N+2}(n)$	*norm. f. shift. augm. gain*
10	$N+2\times 1$	$\mathbf{g}^b_{N+2}(n) = \mathbf{K}^b_+(n)\pi(n)$	*backward shift. augm. gain*
11	$N+2\times 1$	$\mathbf{c}^b_{N+2}(n) = \gamma_+^{-1}(n)\mathbf{g}^b_{N+2}(n)$	*norm. b. shift. augm. gain*
		Prediction Errors	
12	1×1	$e(n\,\vert\,n-1) = d(n) - \mathbf{z}_N^T(n)\mathbf{w}_N(n-1)$	
13	1×1	$e(n\,\vert\,n) = d(n) - \mathbf{z}_N^T(n)\mathbf{w}_N(n) = <\pi(n), \mathrm{P}_0^{\downarrow}(n)\mathrm{d}(n)>$	
14	1×2	$\mathbf{e}_f(n\,\vert\,n-1) = \mathbf{z}(n) - \mathbf{z}_N^T(n-1)\mathbf{f}_N(n-1)$	
15	1×2	$\mathbf{e}_f(n\,\vert\,n) = \mathbf{z}(n) - \mathbf{z}_N^T(n-1)\mathbf{f}_N(n) = <\pi(n), \mathrm{P}_1^{\downarrow}(n)\mathbf{z}(n)>$	
16	1×2	$\mathbf{e}_b(n\,\vert\,n-1) = \mathbf{z}_b(n) - \mathbf{z}_N^T(n)\mathbf{b}_N(n-1)$	
17	1×2	$\mathbf{e}_b(n\,\vert\,n) = \mathbf{z}_b(n) - \mathbf{z}_N^T(n)\mathbf{b}_N(n) = <\pi(n), \mathrm{P}_0^{\downarrow}(n)\mathbf{z}_b(n)>$	
18	1×1	$\gamma(n) = 1 - \mathbf{z}_N^T(n)\mathbf{g}_N(n) = <\pi(n), \mathrm{P}_0^{\downarrow}(n)\pi(n)>$	
19	1×1	$\gamma_+(n) = 1 - \mathbf{z}_{N+2}^T(n)\mathbf{g}_{N+2}(n)$	
		Residuals	
20	1×1	$\epsilon(n) = <\mathbf{e}(n\,\vert\,n), \mathbf{e}(n\,\vert\,n)> = <\mathbf{d}(n), \mathrm{P}_0^{\downarrow}(n)\mathbf{d}(n)>$	
21	2×2	$\boldsymbol{\epsilon}_f(n) = <\mathbf{e}_f(n\,\vert\,n), \mathbf{e}_f(n\,\vert\,n)> = <\mathbf{z}(n), \mathrm{P}_1^{\downarrow}(n)\mathbf{z}(n)>$	
22	2×2	$\boldsymbol{\epsilon}_b(n) = <\mathbf{e}_b(n\,\vert\,n), \mathbf{e}_b(n\,\vert\,n)> = <\mathbf{z}_b(n), \mathrm{P}_0^{\downarrow}(n)\mathbf{z}_b(n)>$	

Table II.

Summary: General Order, Two Channel Transversal RLS Algorithm

EQ	REF	DIM	OPS	Computation
II.1	1.14	1×2	$2N$	$e_f(n\|n-1) = x(n) - \mathbf{z}_N^T(n-1)\mathbf{f}_N(n-1)$
2	3.12	1×2	2	$e_f(n\|n) = \gamma(n-1)e_f(n\|n-1)$
3	3.14	2×2	3*	$\epsilon_f(n) = \epsilon_f(n-1) + e_f^T(n\|n-1)e_f(n\|n)$
4		2×2	5*	$\epsilon_f^{-1}(n)$
5	3.21	1×1	6	$\gamma_+(n) = \gamma(n-1) - e_f(n\|n)\epsilon_f^{-1}(n)e_f^T(n\|n)$
6	3.24	$N+2\times 1$	$2N-2$	$\mathbf{c}_{N+2}^f(n) = \begin{bmatrix}\mathbf{0}\\ \mathbf{c}_N(n-1)\end{bmatrix} - \begin{bmatrix}\mathbf{I}_2\\ -\mathbf{f}_N(n-1)\end{bmatrix}\epsilon_f^{-1}(n)e_f^T(n\|n)\gamma_+^{-1}(n)$
7	1.25	$N+2\times 1$	0**	$\mathbf{c}_{N+2}^b(n) = \mathbf{S}_b\mathbf{S}_f^T\mathbf{c}_{N+2}^f(n)$
8	3.8	2×1	0**	$c_+^b(n) =$ *last 2 elements in* $\mathbf{c}_{N+2}^b(n)$ *vector*
9	3.10	$N\times 2$	$2N$	$\mathbf{f}_N(n) = \mathbf{f}_N(n-1) - \mathbf{c}_N(n-1)e_f(n\|n)$
10	3.28	2×1	4	$e_b^T(n\|n-1) = \epsilon_b(n-1)c_+^b(n)$
11	3.23	1×1	4	$\gamma(n) = \gamma_+(n)[1 - \gamma_+(n)e_b(n\|n-1)c_+^b(n)]^{-1}$
12	3.18	1×2	2	$e_b(n\|n) = \gamma(n)e_b(n\|n-1)$
13	3.20	2×2	3*	$\epsilon_b(n) = \epsilon_b(n-1) + e_b^T(n\|n-1)e_b(n\|n)$
14	3.25	$N\times 1$	$2N$	$\begin{bmatrix}\mathbf{c}_N(n)\\ \mathbf{0}\end{bmatrix} = \mathbf{c}_{N+2}^b(n) + \begin{bmatrix}\mathbf{b}_N(n-1)\\ -\mathbf{I}_2\end{bmatrix}c_+^b(n)$
15	3.16	$N\times 2$	$2N$	$\mathbf{b}_N(n) = \mathbf{b}_N(n-1) + \mathbf{c}_N(n)e_b(n\|n)$
16	1.12	1×1	N	$e(n\|n-1) = d(n) - \mathbf{z}_N^T(n)\mathbf{w}_N(n-1)$
17	3.4	1×1	1	$e(n\|n) = \gamma(n)e(n\|n-1)$
18	3.2	$N\times 1$	N	$\mathbf{w}_N(n) = \mathbf{w}_N(n-1) + \mathbf{c}_N(n)e(n\|n)$
Total OPS ($\times, \div$)			$12N+32$	

Notes: * indicates that matrix symmetry allows reduction in OPS.
** since rearrangement is predetermined.

Table I.
Summary of Defining Relationships for
Two-Channel, General Order Derivation

EQ	DIM	DEFINITION	
		Transversal Filters	
1.1	$N \times 1$	$\mathbf{w}_N(n) = \mathbf{K}_0(n)\mathbf{d}(n)$	*joint predictor*
2	$N \times 2$	$\mathbf{f}_N(n) = \mathbf{K}_1(n)\mathbf{z}(n)$	*forward predictor*
3	$N \times 2$	$\mathbf{b}_N(n) = \mathbf{K}_0(n)\mathbf{z}_b(n)$	*backward predictor*
4	$N \times 1$	$\mathbf{g}_N(n) = \mathbf{K}_0(n)\boldsymbol{\pi}(n)$	*gain (unit predictor)*
5	$N \times 1$	$\mathbf{c}_N(n) = \gamma^{-1}(n)\mathbf{g}_N(n)$	*normalized gain*
6	$N-2 \times 1$	$\mathbf{g}_{N-2}(n) = \mathbf{K}_+(n)\boldsymbol{\pi}(n)$	*augmented gain*
7	$N-2 \times 1$	$\mathbf{c}_{N-2}(n) = \gamma_-^{-1}(n)\mathbf{g}_{N-2}(n)$	*norm. augm. gain*
8	$N-2 \times 1$	$\mathbf{g}^f_{N-2}(n) = \mathbf{K}^f_+(n)\boldsymbol{\pi}(n)$	*forward shifted augm. gain*
9	$N-2 \times 1$	$\mathbf{c}^f_{N-2}(n) = \gamma_-^{-1}(n)\mathbf{g}^f_{N-2}(n)$	*norm. f. shift. augm. gain*
10	$N-2 \times 1$	$\mathbf{g}^b_{N-2}(n) = \mathbf{K}^b_+(n)\boldsymbol{\pi}(n)$	*backward shift. augm. gain*
11	$N-2 \times 1$	$\mathbf{c}^b_{N-2}(n) = \gamma_-^{-1}(n)\mathbf{g}^b_{N-2}(n)$	*norm. b. shift. augm. gain*
		Prediction Errors	
12	1×1	$e(n\|n-1) = d(n) - \mathbf{z}_N^T(n)\mathbf{w}_N(n-1)$	
13	1×1	$e(n\|n) = d(n) - \mathbf{z}_N^T(n)\mathbf{w}_N(n) = <\boldsymbol{\pi}(n), \mathbf{P}_0^{\perp}(n)\mathbf{d}(n)>$	
14	1×2	$e_f(n\|n-1) = z(n) - \mathbf{z}_N^T(n-1)\mathbf{f}_N(n-1)$	
15	1×2	$e_f(n\|n) = z(n) - \mathbf{z}_N^T(n-1)\mathbf{f}_N(n) = <\boldsymbol{\pi}(n), \mathbf{P}_1^{\perp}(n)\mathbf{z}(n)>$	
16	1×2	$e_b(n\|n-1) = z_b(n) - \mathbf{z}_N^T(n)\mathbf{b}_N(n-1)$	
17	1×2	$e_b(n\|n) = z_b(n) - \mathbf{z}_N^T(n)\mathbf{b}_N(n) = <\boldsymbol{\pi}(n), \mathbf{P}_0^{\perp}(n)\mathbf{z}_b(n)>$	
18	1×1	$\gamma(n) = 1 - \mathbf{z}_N^T(n)\mathbf{g}_N(n) = <\boldsymbol{\pi}(n), \mathbf{P}_0^{\perp}(n)\boldsymbol{\pi}(n)>$	
19	1×1	$\gamma_-(n) = 1 - \mathbf{z}_{N-2}^T(n)\mathbf{g}_{N-2}(n)$	
		Residuals	
20	1×1	$\epsilon(n) = <\mathbf{e}(n\|n), \mathbf{e}(n\|n)> = <\mathbf{d}(n), \mathbf{P}_0^{\perp}(n)\mathbf{d}(n)>$	
21	2×2	$\boldsymbol{\epsilon}_f(n) = <\mathbf{e}_f(n\|n), \mathbf{e}_f(n\|n)> = <\mathbf{z}(n), \mathbf{P}_1^{\perp}(n)\mathbf{z}(n)>$	
22	2×2	$\boldsymbol{\epsilon}_b(n) = <\mathbf{e}_b(n\|n), \mathbf{e}_b(n\|n)> = <\mathbf{z}_b(n), \mathbf{P}_0^{\perp}(n)\mathbf{z}_b(n)>$	