TIME-VARYING COMPLEX-VALUED POLE-ZERO SYSTEMS IDENTIFICATION

S. T. Alexander

Center for Communications and Signal Processing
Box 7914, North Carolina State University
Raleigh, NC 27695-7911

CCSP-TR-87/15

October 1987
ABSTRACT

A new fast adaptive recursive least squares filter structure is presented for identification of time-varying complex-valued pole-zero (ARMA) systems. The orders of the pole (AR) estimation and zero (MA) estimation filters may be chosen independently to match any a priori knowledge of signal environment. This new structure is applicable to situations such as radar, sonar, channel equalization and echo cancellation in which passband data is downshifted to baseband, producing the familiar complex-valued signal. The derivation is explained in detail to illustrate the geometrical principles involved.
1. INTRODUCTION

The techniques of fast recursive least squares (RLS) filtering have recently received a great deal of investigation due to their rapid convergence rates and the variety of applications to which these techniques may be applied. The two generic structures for these fast RLS filters are lattice and transversal and each form is very useful depending upon the particular application. Since this paper addresses a transversal-type RLS filter the reader is referred to the works by Lee, et al. [1], Turner [2], and Friedlander [3] for more information on fast lattice RLS filters.

Fast RLS transversal filters may be derived from either a matrix algebraic approach, such as those by Falconer and Ljung [4] and Carayannis, et al. [5], or by using vector space concepts. The vector space approach has been used by several researchers [6,7] and leads to filters with a minimum complexity, while at the same time retaining an intuitive geometrical understanding of the RLS operations. Cioffi [8] and Cioffi and Kailath [6] used such a vector space approach to derive all-zero fast transversal filters which have come to be collectively known as FTF (Fast Transversal Filter) methods.

Many of the applications of FTF-type methods have thus far been for the case of all zero prediction or FIR (finite impulse response) prediction. In this method the values of a "data" signal \( s(n) \) are used to predict the values of a "desired" signal \( r(n) \). For the system identification problem this corresponds to figure 1(a), in which \( s(n) \) is the system input, \( r(n) \) is the system output, and the prediction \( \hat{r}(n) \) is given by the FIR filter output

\[
\hat{r}(n) = \sum_{k=0}^{q-1} b_k(n)s(n-k)
\]

(1.1)

For adaptive identification some property of the prediction error \( e(n) \) in figure 1 is used for time-varying adjustment of the \( b_k(n) \). This type of predictor can be quite accurate for systems producing outputs generated by a difference equation of the form
\[ r(n) = \sum_{k=0}^{q'-1} \beta_k s(n-k) \]  
(1.2)

where the \( \beta_k \) are the true system parameters and \( q' \) is the true system order. Accuracy in this case means that as the power of the \( e(n) \) signal is minimized the \( b_k(n) \) closely approach the true \( \beta_k \). If the order \( q \) of the predictor can be made greater than the true system order \( q' \) then the accuracy of the predictor (1.1) is usually sufficient. For further information on systems identification procedures the interested reader is referred to [9,10].

However, many systems identification problems involve trying to identify systems or processes which have a pole-zero structure; that is, the true system output is generated by the difference equation

\[ r(n) = \sum_{k=0}^{q'-1} \beta_k s(n-k) + \sum_{k=1}^{p'} \alpha_k r(n-k) \]  
(1.3)

where the \( \alpha_k \) are the true system "pole" coefficients and \( p' \) is the true system zero order. It is straightforward [17] to show that one structure for identifying such a system has two FIR filters approximating a "pole-zero" structure, giving

\[ \hat{r}(n) = \sum_{k=0}^{q-1} b_k(n) s(n-k) + \sum_{k=1}^{p} a_k(n) r(n-k) \]  
(1.4)

A block diagram of such a predictor is given in figure 1(b).

It should be noted that this problem is different from that of infinite impulse response or IIR adaptive filtering. In IIR adaptive filtering the output \( \hat{r}(n) \) of the adaptive filter is fed back to its input, where it is recursively filtered along with the system input \( s(n) \). This causes the resulting error surface of the IIR filter to be multi-modal (i.e., higher order than quadratic) which complicates the analysis and convergence issues [10,12]. The adaptive structure under investigation in this paper consists of two complex fast least squares FIR filters, one operating on the systems input \( s(n) \) and the other operating on the previous systems output \( r(n-1) \).
One approach to formulating this pole-zero predictor for systems identification could be the use of a multichannel FTF. Multichannel methods [6] have been derived for predicting multiple desired signals (which now become desired vector sequences) based upon collections of FTF "matrices" operating on multiple data vector sequences; i.e., the multiple data channels. This type of approach could be used for the pole-zero problem previously described since the system input \( s(n) \) and previous output \( r(n - 1) \) could be assigned to two data channels of input signals. However, one limitation of this approach is that derivations done thus far [6,8] require the FTF predictors in each channel to be the same order. A similar restriction was required on an ARMA-type predictor configured in a lattice form by Morf, et al [11].

In general there is no physical reason to necessarily expect the true zero order, \( q' \), and the true pole order, \( p' \), to be equal. Therefore, it would be very desirable to derive an LS pole-zero predictor having the flexibility of differing orders \( p \) and \( q \). To date no work has appeared applying the vector space approach to the case of identification for complex, time-varying systems of arbitrary pole-zero order. The work in this paper addresses this problem and derives an adaptive least squares structure having these desired properties. The techniques developed in this paper have such a flexibility of arbitrary orders \( p \) and \( q \), plus being able to track the time-varying nature of the system. Additionally, the techniques to be developed in this paper are applicable to complex-valued signals and systems. As such, the methods of this paper extends the work in [19] which addressed only the case of stationary real data. The algorithms developed in this paper can be applied immediately to such diverse areas as passband data echo cancellation, channel equalization, and target identification and tracking for active sonar and radar.
The body of this paper is organized as follows. Section 2 formulates complex, time-varying pole-zero system identification as a least squares prediction problem. Section 3 then investigates the projection matrix and transversal filter operators and discusses some of their properties. Sections 4 and 5 then give the main results of this paper which is the derivation of the algorithm using vector space concepts. Finally, Section 6 discusses implementation considerations and areas for future research.

2. LEAST SQUARES FORMULATION

The derivation of the time-varying, complex, pole-zero fast RLS adaptive filter will be done using a vector space approach. This approach has been used for joint process transversal estimation [7], multichannel transversal filtering [8], and fast lattice filters [1]. However, no work to date has used the vector space approach for arbitrary pole and zero orders in the complex, time-varying case. The techniques presented in this paper address this problem. Moreover, the vector space approach provides an intuitive geometrical understanding for the filter derivation. For a tutorial article on this approach the reader is directed to [13] and more detailed developments are available in [7,17].

The required signal vectors may be determined by referring to figure 1(b). For complex data the linear LS filter minimizes the error measure [4]

$$e(n) = \sum_{i=1}^{n} \lambda^{n-i} e^*(i|n)e(i|n)$$

where $e(i|n)$ is the error in predicting (or estimating) the $i$th signal sample using the filter computed at time $n$, and $\lambda < 1$ is the exponential weighting factor which more heavily weighs recent data. The symbol "*" denotes complex conjugation. The error is given by

$$e(i|n) = r(i) - \hat{r}(i|n)$$

where $\hat{r}(i|n)$ is the prediction of $r(i)$. The specific form of the predictor will be discussed directly. Assuming that computation begins at time 1 and all signal values are zero prior to
this time then (2.1) can be rewritten as

\[ \epsilon(n) = [\Omega(n)\epsilon(n)]^H \Omega(n)\epsilon(n) \]  

(2.3)

where \( \epsilon(n) \) is the \( n \)-component vector of errors,

\[ \epsilon(n) = [\epsilon(1|n), \ldots, \epsilon(n-1|n), \epsilon(n|n)]^T, \]  

(2.4)

\( H \) denotes the conjugate transpose (or Hermitian transpose) operation, and \( T \) denotes the simple transpose. The \( n \times n \) diagonal matrix \( \Omega(n) \) has elements which are the square roots of the \( \lambda^{-i} \):

\[ \Omega(n) = \text{diag} \left[ \sqrt{\lambda_1}, \ldots, \sqrt{\lambda_2}, \sqrt{\lambda}, \ 1 \right] \]  

(2.5)

Equation (2.3) is expressed in the very important form of an inner product in a linear vector space. This inner product has the interpretation of a "length" operation and can be defined for general vectors \( u, v \), and real diagonal weighting matrix \( \Omega \) by

\[ \langle u, v \rangle_\Omega = (\Omega u)^H \Omega v = u^H \Omega^2 v \]  

(2.6)

From (2.3) this notation immediately produces

\[ \epsilon(n) = \langle \epsilon(n), \epsilon(n) \rangle_\Omega \]  

(2.7)

A notation which will be very useful in the work to follow is to define

\[ u_\Omega(n) = \Omega(n)u(n) \]  

(2.8)

That is, a vector \( u(n) \) premultiplied by the weighting matrix \( \Omega(n) \) produces the weighted vector denoted by the subscript \( \Omega \). For example, \( e_\Omega(n) = \Omega(n)e(n) \). This provides the alternate inner product definition for \( \epsilon(n) \):

\[ \epsilon(n) = \langle e_\Omega(n), e_\Omega(n) \rangle = e_\Omega^H(n)e_\Omega(n) \]  

(2.9)

For the identification of a pole-zero system as shown in figure 1(b), the prediction \( \hat{r}(i|n) \) becomes

\[ \hat{r}(i|n) = \sum_{k=0}^{q-1} b_k(n)s(i-k) + \sum_{k=1}^{p} a_k(n)r(i-k) \]  

(2.10)

Note that the system output in (2.10) must be delayed by at least one sample to be used in
the prediction of \( r(n) \). Therefore, define the delayed signal \( y(n) \) as

\[
y(n) = r(n-1)
\]  

(2.11)

This substitution (2.11) will simplify the derivation of the LS algorithm and cause no loss of generality. Next, define \( n \)-component vectors of system inputs and outputs by

\[
s(n) = [s(1), \cdots, s(n-1), s(n)]^T
\]  

(2.12a)

\[
y(n) = [y(1), \cdots, y(n-1), y(n)]^T
\]  

(2.12b)

Additionally, define the shift operator \( z^{-k} \) such that the \( n \) component vector \( z^{-k}v(n) \) is

\[
z^{-k}v(n) = [0, \cdots, 0, v(1), \cdots, v(n-k)]^T
\]  

(2.13)

The first non-zero element of \( z^{-k}v(n) \) is \( v(1) \), which appears in the \( k \)th component. Using the definitions (2.11)-(2.13) it is straightforward to show that the entire error vector in (2.4) may be written as

\[
e(n) = r(n) - \sum_{k=0}^{p-1} z^{-k}y(n)a_k(n) - \sum_{k=0}^{q-1} z^{-k}s(n)b_k(n)
\]  

(2.14)

This expression is easily obtained by using (2.2) and (2.10) for each scalar error \( e(i|n) \) and then using (2.11), (2.12) to form the resulting vectors. To simplify this expression define \( S_{0,q-1}(n) \) as the \( n \times q \) matrix of input data

\[
S_{0,q-1}(n) = \begin{bmatrix} s(n), \cdots, z^{-1}s(n), \cdots, z^{-(q-1)}s(n) \end{bmatrix}
\]  

(2.15)

and \( Y_{0,p-1}(n) \) as the \( n \times p \) matrix of systems output data

\[
Y_{0,p-1}(n) = \begin{bmatrix} y(n), \cdots, z^{-1}y(n), \cdots, z^{-(p-1)}y(n) \end{bmatrix}
\]  

(2.16)

The indices on these matrices correspond to the amount of "shift" in each column vector. These definitions allow the error to be written as

\[
e(n) = r(n) - Y_{0,p-1}(n) \begin{bmatrix} a(n) \\ b(n) \end{bmatrix}
\]  

(2.17)

where \( a(n) \) and \( b(n) \) are the vectors of the \( a_k(n) \) and \( b_k(n) \), respectively. The weighted error vector is easily seen from (2.17) to be given by
\[
\Omega(n)\mathbf{e}(n) = \Omega(n)\mathbf{r}(n) - \Omega(n)\begin{bmatrix} Y_{0,p-1}(n) & S_{0,q-1}(n) \end{bmatrix} \begin{bmatrix} \mathbf{a}(n) \\ \mathbf{b}(n) \end{bmatrix} \tag{2.18}
\]

Using the general substitution (2.8) in (2.18) then gives

\[
\mathbf{e}_\Omega(n) = \mathbf{r}_\Omega(n) - \begin{bmatrix} \Omega(n)Y_{0,p-1}(n) & \Omega(n)S_{0,q-1}(n) \end{bmatrix} \begin{bmatrix} \mathbf{a}(n) \\ \mathbf{b}(n) \end{bmatrix} \tag{2.19}
\]

Finally, let \( L = p + q \) and define the partitioned input/output data matrix as \( \mathbf{X}_0(n) \):

\[
\mathbf{X}_0(n) = \begin{bmatrix} \Omega(n)Y_{0,p-1}(n) & \Omega(n)S_{0,q-1}(n) \end{bmatrix} \tag{2.20}
\]

and define the \( L \) component LS filter vector \( \mathbf{w}_L(n) \) as

\[
\mathbf{w}_L(n) = \begin{bmatrix} \mathbf{a}(n) \\ \mathbf{b}(n) \end{bmatrix} \tag{2.21}
\]

Using (2.20) and (2.21) in (2.19) now provides a compact form for the weighted error vector

\[
\mathbf{e}_\Omega(n) = \mathbf{r}_\Omega(n) - \mathbf{i}_\Omega(n) \tag{2.22a}
\]

where \( \mathbf{i}_\Omega(n) \) is the linear prediction of the system output

\[
\mathbf{i}_\Omega(n) = \mathbf{X}_0(n)\mathbf{w}_L(n) \tag{2.22b}
\]

Referring to (2.22) it is easy to see that \( \mathbf{i}_\Omega(n) \) is a linear combination of the columns of \( \mathbf{X}_0(n) \) and therefore lies in the column space of \( \mathbf{X}_0(n) \). This will be very important for formulating the vector space approach to the LS problem.

The preceding has formulated the identification of the complex, pole/zero, time varying system as the following LS problem: minimize the norm (or length) squared of the weighted error vector \( \mathbf{e}_\Omega(n) \),

\[
\mathbf{e}(n) = <\mathbf{e}_\Omega(n),\mathbf{e}_\Omega(n)> \tag{2.9}
\]

subject to the constraint of the linear prediction form \( \mathbf{i}_\Omega(n) \) from (2.22). The solution to this problem is based on the projection theorem \([7,14,15]\) which states that the error vector must be orthogonal to the vector space spanned by the columns of the \( \mathbf{X}_0(n) \) matrix. This requires that the inner product of \( \mathbf{e}_\Omega(n) \) with each column of \( \mathbf{X}_0(n) \) must be zero. Referring to (2.15) and (2.16) this gives the requirements
Using (2.22) in (2.23) and grouping the results using (2.20) then leads to the important normal equations:

\begin{align}
<X_0(n), X_0(n)> w_L(n) &= <X_0(n), r_\Omega(n)> \\
\end{align}

For the present assume the \( L \times L \) matrix \( <X_0(n), X_0(n)> \) has full rank. Thus, from (2.24)

\begin{align}
w_L(n) &= <X_0(n), X_0(n)>^{-1} <X_0(n), r_\Omega(n)>.
\end{align}

For example, the current time \( n \) prediction of the scalar systems output \( r(n) \) would be given by

\begin{equation}
\hat{f}(n|n) = w_L^T(n)x_L(n)
\end{equation}

where \( x_L(n) \) is the \( L \)-length vector of data,

\begin{equation}
x_L(n) = [y(n), \cdots, y(n-p+1), s(n), \cdots, s(n-q+1)]^T
\end{equation}

The least squares systems identification problem is thus to find the vector \( w_L(n) \) of filter coefficients, since this vector will converge to the set of unknown system coefficients. However, the block oriented matrix approach of (2.24) requires at least \( O(L^2) \) operations to solve the normal equations (2.24). Clearly, this is an improbable approach if the system is time-varying and tracking of the system is required. The recursive method to be derived in the next section overcomes these difficulties and provides a technique for computing \( w_L(n) \) from a knowledge of \( w_L(n-1) \) using only \( O(L) \) operations per update.

3. PROJECTION MATRIX AND TRANSVERSAL FILTER OPERATOR

Notice that (2.22b) and (2.25) allow the prediction \( \hat{r}_\Omega(n) \) to be written as

\begin{equation}
\hat{r}_\Omega(n) = P_0(n)r_\Omega(n)
\end{equation}

where \( P_0(n) \) is the \( n \times n \) matrix

\begin{equation}
P_0(n) = X_0(n)<X_0(n), X_0(n)>^{-1}X_0^H(n)
\end{equation}
Since (2.19) showed that $\mathbf{f}_{\Omega}(n)$ is in the column space of $X_0(n)$ then $P_0(n)$ has the following interpretation: $P_0(n)$ operating on any vector $\mathbf{v}$ produces the component of $\mathbf{v}$ in the column space of $X_0(n)$. Stated another way, $P_0(n)\mathbf{v}$ produces the projection of $\mathbf{v}$ onto $\{X_0(n)\}$, the data subspace spanned by the columns of $X_0(n)$. Brackets around a matrix (or vector) will be used to denote the associated vector space. In this paper a linear vector space $\{V\}$ is the set of all vectors $\mathbf{v}$ such that $\mathbf{v}$ is a linear combination of the basis vectors $\mathbf{v}_k$; that is,

$$\{V\} = \{\mathbf{v} \mid \mathbf{v} = \sum_{k=1}^{L} \alpha_k \mathbf{v}_k\}$$

(3.3)

where the $\alpha_k$ are scalar weights. For a rigorous examination of vector spaces the reader is directed to [15,16]. The advantage of formulating RLS identification as a vector space optimization is that the columns of the matrix $X_0(n)$ may be used directly as the basis vectors of the data subspace $\{X_0(n)\}$. In this manner the least squares solution is optimized to the exact set of acquired data (as represented in the columns of $X_0(n)$). This suggests that the $\mathbf{w}_L(n)$ filter coefficients are the "optimal" weightings on the basis vectors of $\{X_0(n)\}$ to produce the LS prediction $\mathbf{f}_{\Omega}(n)$.

Using these geometrical concepts it is easy to see that the minimum length error vector $\mathbf{e}_\Omega(n)$ is that vector which is orthogonal to the data subspace $\{X_0(n)\}$ as shown in figure 2. Mathematically,

$$\mathbf{e}_\Omega(n) = P_0^\perp(n)\mathbf{r}_\Omega(n) ,$$

(3.4)

where $P_0^\perp(n)$ is the orthogonal projection matrix

$$P_0^\perp(n) = I - P_0(n) .$$

(3.5)

For any vector $\mathbf{v}$, the operation $P_0(n)\mathbf{v}$ provides the component of $\mathbf{v}$ in $\{X_0(n)\}$ and $P_0^\perp(n)\mathbf{v}$ gives the component orthogonal to $\{X_0(n)\}$. This leads to the following orthogonal decomposition of $\mathbf{r}_\Omega(n)$ with respect to $\{X_0(n)\}$:
or using (3.1) and (3.4)

\[ r_\Omega(n) = \tilde{r}_\Omega(n) + e_\Omega(n). \]  

(3.7)

The preceding has discussed the estimate \( \tilde{r}_\Omega(n) \) in terms of the vector space \( \{X_\Omega(n)\} \). Since in systems identification the immediate interest is in \( w_L(n) \), it is necessary to next determine the interpretation and computation of \( w_L(n) \) in the vector space approach. This problem is simplified by defining the transversal filter operator \( K_0(n) \) as the \( L \times n \) matrix

\[ K_0(n) = (X_\Omega(n),X_\Omega(n))^H \]  

(3.8)

which from (2.22) and (3.2) allows the LS filter \( w_L(n) \) to be written as

\[ w_L(n) = K_0(n)r_\Omega(n) \]  

(3.9)

In (3.8) the subscript "0" and the time argument "n" denote that the operator is computed using \( X_\Omega(n) \). Thus, (3.9) in words has the interpretation "\( w_L(n) \) is the optimal (LS) predictor for the vector \( r_\Omega(n) \) using the subspace \( \{X_\Omega(n)\}\)." Since (3.9) is valid for time \( n \) it is also valid for \( n-1 \):

\[ w_L(n-1) = K_0(n-1)r_\Omega(n-1) \]  

(3.10)

Therefore, from (3.9) and (3.10) the recursive LS problem of computing \( w_L(n) \) from \( w_L(n-1) \) is very clear. Since

\[ r_\Omega(n) = \begin{bmatrix} \lambda^k r_\Omega(n-1) \\ r(n) \end{bmatrix}, \]  

(3.11)

then \( r_\Omega(n) \) is known at time \( n \) and the LS problem consists of updating the transversal filter operator from \( K_0(n-1) \) to \( K_0(n) \).

In summary of the procedure thus far, the systems identification problem was first formulated in the appropriate linear vector space. Then the concept of projecting a vector onto a subspace and the required transversal filter implementation led to the transversal filter operator. Last, recursive updating required that \( K_0(n) \) be computed from \( K_0(n-1) \). With
the preceding as background, the derivation of the complex pole/zero algorithm proceeds in
the next section.

4. WEIGHT VECTOR UPDATE

The derivation of the fast algorithm requires the concept of the $n$ component unit time
vector

$$\pi(n) = [0,0, \cdots, 0,1]^T$$  \hspace{1cm} (4.1)
sometimes also called the pinning vector [6] or the time annihilation vector [1]. Updating
the transversal filter operator is done by creating the appended vector space \{\(X_0(n), \pi(n)\)\} and
finding the transversal filter operator for the appended space. One outcome of this
approach is that the required relation between \(K_0(n)\) and \(K_0(n-1)\) results. To derive this
relation and other results to come, three vector space recursions will be very useful. These
recursions are well-established and the reader is directed to [6,7,18] for their derivations.
First, the projection matrix update for a general appended subspace \(\{U, V\}\) is

$$P_{UV} = P_U + P_U^1 V (P_U^1 V, P_U^1 V)^{-1} V^H P_U^1$$  \hspace{1cm} (4.2a)
The updated projection matrix is independent of whether the new information in \(V\) is
appended at the left or right (i.e., \(VU\) or \(UV\)). However, since the transversal filter opera-
tor orders the resulting LS filter coefficients the order of \(V\) and \(U\) does make a difference in
computing the transversal filter operator. In general, \(K_{UV} \neq K_{UV}\) and both updates are
needed in the ensuing derivation. These updates are [6,7]

$$K_{UV} = \begin{bmatrix} 0 \\ K_U \end{bmatrix} + \begin{bmatrix} I \\ -K_U V \end{bmatrix} <P_U^1 V, P_U^1 V>^{-1} V^H P_U$$  \hspace{1cm} (4.2b)

$$K_{UV} = \begin{bmatrix} K_U \\ 0 \end{bmatrix} + \begin{bmatrix} -K_U V \\ I \end{bmatrix} <P_U^1 V, P_U^1 V>^{-1} V^H P_U$$  \hspace{1cm} (4.2c)
The dimension of the lower partition elements in (4.2b) and (4.2c) must be compatible with
the dimension of \(V\). Additionally, an inner product update needed later is given by
Other useful properties which are easily verified are

\[ P_W P_W = P_W, \quad K_W P_W = K_W \]  

where \( W \) is any general matrix. Thus, letting \( U = X_0(n) \), \( V = \pi(n) \) in (4.2c) and using (4.3):

\[
\begin{align*}
K_{0,\pi}(n) &= \begin{bmatrix} K_{0}(n) & -K_{0}(n) \pi(n) \end{bmatrix} + \begin{bmatrix} -K_{0}(n) \pi(n) \end{bmatrix} \pi(n) \pi(n)^T P_0^T(n) P_0(n). 
\end{align*}
\]  

(4.4)

where from (3.8) \( K_{0,\pi}(n) \) is given by substituting the appended matrix \([X_0(n), \pi(n)]\) for \( X_0(n) \). However, \( K_{0,\pi}(n) \) may be decomposed as [6,7]

\[
\begin{bmatrix} K_{0}(n-1) & 0_n \end{bmatrix} z^T 1
\]  

(4.5)

where \( K_{0}(n-1) \) is the \( L \times (n-1) \) matrix defining the transversal filter operator at time \( n-1 \) and \( z^T \) is an \((n-1)\times 1\) vector not specifically needed in the ensuing computation. Postmultiplying (4.4) by \( r_{\Omega}(n) \) from (3.11), using (4.5) and keeping the top partition of (4.4) then gives

\[
\lambda^n K_{0}(n-1) r_{\Omega}(n-1) = K_{0}(n) r_{\Omega}(n) - \frac{g_L(n) \pi^T(n) P_0^T(n) r_{\Omega}(n)}{\gamma_L(n)}
\]  

(4.6)

where \( g_L(n) \) is the \( L \times 1 \) vector

\[
g_L(n) = K_{0}(n) \pi(n)
\]  

(4.7)

and \( \gamma_L(n) \) is the real scalar

\[
\gamma_L(n) = <\pi(n), P_0^T(n) \pi(n)> 
\]  

(4.8)

The vector \( g_L(n) \) is the important LS gain vector, and (4.7) gives it the geometrical interpretation of "the optimal (LS) prediction of \( \pi(n) \) using the data matrix \( X_0(n) \). The scalar \( \gamma_L(n) \) is related to the "angular change" between the data subspaces at time \( n-1 \) and time \( n \). It is straightforward using (4.7) and (4.8) to show that \( \gamma_L(n) \) has the alternate form

\[
\gamma_L(n) = 1 - g_L^T(n) x_L(n)
\]  

(4.9)

where \( x_L(n) \) is the data vector defined in (2.27). Next, using (3.4) and (3.9) in (4.6) provides
\[ w_L(n) = \lambda^n w_L(n-1) + \frac{e(n|n)}{\gamma_L(n)} g_L(n) \]  
(4.10)

where from (3.4) the scalar \( e(n|n) \) is the \( n \)th component of the error vector \( e_{\Omega(n)} \), given by

\[ e(n|n) = d(n) - x_L^T(n) w_L(n) \]  
(4.11)

Substituting (4.10) into (4.11) and using (4.9) then provides the relation

\[ e(n|n) = \gamma_L(n) e(n|n-1) \]  
(4.12)

where \( e(n|n-1) \) is the "a priori" error

\[ e(n|n-1) = d(n) - \lambda^n x_L^T(n) w_L(n-1) \]  
(4.13)

Using (4.12) and (4.13) in (4.10) then gives the desired update

\[ w_L(n) = \lambda^n w_L(n-1) + e(n|n-1) g_L(n) \]  
(4.14)

The recursion (4.14) states that the "new" \( L \)-component LS predictor is the "old" predictor \( w_L(n-1) \) plus a correction vector. The recursion (4.14) demonstrates that the gain vector must be computed in order to update the LS predictor. The remainder of the derivation for the time-varying, complex, pole-zero system identification method thus consists of determining the update for the unknown gain \( g_L(n) \) on the right hand side of (4.14).

5. GAIN VECTOR UPDATE

The gain vector update for \( g_L(n) \) is most easily derived by considering two different partitions of the \((L + 2) \times n\) data matrix \( X(n) \), where

\[ X(n) = \Omega(n) \begin{bmatrix} Y_{0,\rho}(n) & S_{0,q}(n) \end{bmatrix} \]  
(5.1)

The \( X(n) \) matrix has added the vectors \( z^{-\rho} y(n) \) and \( z^{-q} s(n) \) to the \( X_0(n) \) matrix. The benefit of \( X(n) \) can be seen by considering \( X_1(n) \), the matrix of delayed data,

\[ X_1(n) = \Omega(n) \begin{bmatrix} Y_{1,\rho}(n) & S_{1,q}(n) \end{bmatrix} \]  
(5.2)

It is straightforward to show that
Thus \( X(n) \) incorporates basis vectors from both the present data subspace \( \{X_0(n)\} \) and the immediately past subspace \( \{X_0(n-1)\} = \{X_1(n)\} \). Since \( \{X(n)\} \) has \( L+2 \) basis vectors then the projection of \( \pi(n) \) onto \( \{X(n)\} \) will result in \( g_{L+2}(n) \), the \( (L+2) \)-length LS predictor of \( \pi(n) \). One consideration is that (4.2b) for updating will require partitioning \( \{X(n)\} \) in terms of \( X_1(n) \) in order to compute \( g_{L+2}(n) \) in terms of \( g_L(n-1) \). However, recall from (4.10) that there is a one-to-one correspondence between \( g_L(n) \) and \( w_L(n) \); that is, the \( i \)th gain component, \( g_i(n) \), is used to update the \( i \)th weight, \( w_i(n) \), in (4.10). Similarly, \( g_{L+2}(n) \) will provide a one-to-one correspondence with the components of \( w_{L+2}(n) \). One present inconvenience is that \( X(n) \) does not immediately lend itself to such an obvious partitioning. This is seen by observing that

\[
X(n) = \begin{bmatrix}
y(n), Y_{1,p}(n), s(n), S_{1,q}(n)\end{bmatrix} = \begin{bmatrix} Y_{0,p-1}(n), z^{-p} y(n), S_{0,q-1}(n), z^{-q} s(n)\end{bmatrix}
\]

To use (4.2b) or (4.2c) requires that the "new" or "old" basis vectors be partitioned as a contiguous block within the \( X_0(n) \) or \( X_1(n) \) matrices. This can be done quite easily by defining the \( n \times n \) orthogonal permutation matrices \( Q_f \) and \( Q_b \), such that

\[
X(n)Q_f = \begin{bmatrix} x_0(n) & X_1(n) \end{bmatrix}
\]

(5.4)

\[
X(n)Q_b = \begin{bmatrix} x_0(n) & x_{p,q}(n) \end{bmatrix}
\]

(5.5)

where \( x_0(n) \) is the \( n \times 2 \) matrix of data vectors added at time \( n \),

\[
x_0(n) = \begin{bmatrix} y(n) & s(n) \end{bmatrix}
\]

(5.6)

and \( x_{p,q}(n) \) is the \( n \times 2 \) matrix of data vectors which will be deleted from the time \( n \) prediction:

\[
x_{p,q}(n) = \begin{bmatrix} z^{-p} y(n) & z^{-q} s(n) \end{bmatrix}
\]

(5.7)

These permutation matrices now partition the columns of \( X(n) \) such that (4.2b) can be used
for the update. Since \( Q_f \) and \( Q_b \) are orthogonal it follows that

\[
Q_f Q_f^T = I, \quad Q_b Q_b^T = I
\]

(5.8)

One consideration is that this permutation alters the ordering of the resulting gain vector components. However, the correct ordering may easily be restored, as shown below.

The objective of the current section is to use (4.2b) so that \( g_L(n-1) \) appears on the right hand side of the resulting expression. This is done by substituting \( U = X_1(n) \), \( V = x_0(n) \), and \( W = \pi(n) \) in (4.2b) and using (5.3):

\[ g_{L+2}(n) = \begin{bmatrix} 0 \\ g_L(n-1) \end{bmatrix} + \begin{bmatrix} I_{2\times2} \\ -K_1(n)x_0(n) \end{bmatrix} <x_0(n), P_1^\dagger(n)x_0(n)>^{-1} x_0^H(n) P_1^\dagger(n) \pi(n) \]

(5.9)

Using (5.4) and (5.8) it may be shown that the needed \((L+2)\)-length gain vector \( g_{L+2}(n) \) and the result of (5.9) are related by

\[ g_{L+2}(n) = Q_f g_{L+2}(n) \]

(5.10)

In (5.9) the previous gain vector \( g_L(n-1) \) now appears on the right hand side. To simplify the other terms note that the \( n \times 2 \) matrix \( P_1^\dagger(n)x_0(n) \) has components which are simply the errors in predicting \( y(n) \) and \( s(n) \) using the data matrix \( X_1(n) \). Since the basis vectors of \( \{X_1(n)\} \) are delayed with respect to \( y(n) \) and \( s(n) \) then this is a forward prediction. Let \( e_f(n) \) denote the resulting forward error matrix

\[ e_f(n) = P_1^\dagger(n)x_0(n) = x_0(n) - X_1(n)F(n) \]

(5.11)

where \( F(n) \) is the \( L \times 2 \) forward prediction filter matrix. Since \( F(n) \) is a least squares filter, then in a manner analogous to (3.9) it may be defined in terms of a transversal filter operator for \( X_1(n) \) and the desired signal \( x_0(n) \):

\[ F(n) = K_1(n)x_0(n) \]

(5.12)

Finally, define the \( 2 \times 2 \) forward error residual matrix \( R_f(n) \) by

\[ R_f(n) = <e_f(n), e_f(n)> \]

(5.13)

Therefore, using the substitutions (5.11)-(5.13) in (5.9) gives
\[
g_{L+2}(n) = \begin{bmatrix} 0 \\ g_{L}(n-1) \end{bmatrix} + \begin{bmatrix} I_{2\times2} \\ -F(n) \end{bmatrix} R_{f}^{-1}(n) e_{f}^{*}(n | n) 
\]

where \( e_{f}(n | n) \) is the 2×1 vector of the current time errors and "*" denotes its conjugate:

\[
e_{f}^{*}(n | n) = \langle e_{f}(n), \pi(n) \rangle
\]

Using a similar approach with (4.2c) and the substitutions \( U = X_{0}(n), V = x_{p,q}(n) \) and \( W = \pi(n) \), the following result (after some simplification) is obtained:

\[
g_{L+2}^{b}(n) = \begin{bmatrix} g_{L}(n) \\ 0 \end{bmatrix} + \begin{bmatrix} -B(n) \\ I_{2\times2} \end{bmatrix} R_{b}^{-1}(n) e_{b}^{*}(n | n) = \begin{bmatrix} f_{L}(n) \\ f(n) \end{bmatrix}
\]

Partitioning \( g_{L+2}^{b}(n) \) in (5.16) into the \( L \times 1 \) vector \( f_{L}(n) \) and the \( 2 \times 1 \) vector \( f(n) \) will be very helpful in deriving the computational form for the updates. In deriving (5.16) the following definitions have been used: the filter \( B(n) \) is the \( L \times 2 \) LS backward filter matrix for predicting \( x_{p,q}(n) \) using \( X_{0}(n) \); the \( n \times 2 \) matrix \( e_{b}(n | n) \) is the matrix of backward prediction errors,

\[
e_{b}(n) = P_{O}^{T} x_{p,q}(n) = x_{p,q}(n) - X_{0}(n) B(n)
\]

and \( e_{b}(n | n) \) is the \( 2 \times 1 \) vector of current error values

\[
e_{b}^{*}(n | n) = \langle e_{b}(n), \pi(n) \rangle
\]

Also in (5.16) \( R_{b}(n) \) is the \( 2 \times 2 \) matrix of backward error residuals

\[
R_{b}(n) = \langle e_{b}(n), e_{b}(n) \rangle
\]

The important point to note from (5.16) is that the necessary gain update \( g_{L}(n) \) now appears on the right hand side. Furthermore, it is easily shown that the results of (5.14) and (5.16) are related by

\[
q_{L+2}^{b}(n) = Q_{b}^{T} Q_{f} g_{L+2}^{f}(n)
\]

The relations in (5.20) are no more than a re-ordering of the \( g_{L+2}^{f}(n) \) coefficients. Thus, if \( g_{L+2}^{f}(n) \) can be calculated then \( f_{L}(n) \) and \( f(n) \) are known and \( g_{L+2}^{b}(n) \) can be calculated. Furthermore, (5.16) gives the useful relation

\[
f(n) = R_{b}^{-1}(n) e_{b}^{*}(n | n)
\]
Therefore, the immediate interest is in computing $g_{L+2}(n)$ in (5.14).

**FORWARD FILTER UPDATES**

Returning to (5.14) it is seen that three parameters - $F(n)$, $R_f(n)$, $e_f(n \mid n)$ - must be updated to compute $g_{L+2}(n)$. In order to update $F(n)$, from (5.12) $F(n)$ is seen to be an LS filter. Therefore, (4.2c) may be used to compute its update by making the substitutions $U = X_I(n)$, $V = \pi(n)$, and $W = x_0(n)$ in (4.2c). The top partition of the resulting expression gives

$$
\lambda L F(n-1) = F(n) - g_L(n-1) < \pi(n), P_I^{-1}(n) > ^{-1} e_f^T(n \mid n)
$$

where (4.10) and (5.12) have also been used. It is straightforward using (4.9) to show that

$$
\gamma_L(n-1) = < \pi(n), P_I^{-1}(n) \pi(n) >
$$

where $\gamma_L(n)$ is the angle parameter defined by (4.8). Thus, rearranging (5.22) provides the update for $F(n)$:

$$
F(n) = \lambda L F(n-1) + \frac{g_L(n-1)e_f^T(n \mid n)}{\gamma_L(n-1)}
$$

Next, the $e_f(n \mid n)$ update may be obtained by substituting (5.24) into (5.11) and simplifying. This provides

$$
e_f(n \mid n) = \frac{\gamma_L(n-1)}{\gamma_L(n-1) - g_L(n-1)x_L(n-1)} e_f(n \mid n-1)
$$

where

$$
e_f(n \mid n-1) = x_0(n) - \lambda L F^T(n-1)x_L(n-1)
$$

is the 2×1 error vector produced by predicting the new data samples at time $n$ using the forward filter available at time $n-1$. Then using (4.9), the denominator of (5.25) may be simplified, giving

$$
e_f(n \mid n) = \gamma_L(n-1)e_f(n \mid n-1)
$$

which completes the update for $e_f(n \mid n)$. Note that (5.27) can now be used to compute
Finally in this section the matrix $R_f(n)$ is updated by substituting $Z=W=x_0(n)$, $U=X_1(n)$, and $V=\pi(n)$ in (4.2d). Doing this and simplifying gives

$$R_f(n) = \lambda R_f(n-1) + \frac{e_f^r(n \mid n)e_f^T(n \mid n)}{\gamma_L(n-1)}$$

Equation (5.28) defines the residual matrix in terms of the current forward error matrix $e_f(n \mid n)$, which has been previously computed by (5.27).

**BACKWARD FILTER UPDATES**

Examining the top partition in (5.16), the desired gain $g_L(n)$ is seen to be given by

$$g_L(n) = f_L(n) + B(n)f(n)$$

where (5.21) has also been used. All quantities in (5.29) are known at this time except $B(n)$ which must be updated. It is straightforward to show that using (4.2c) with $U = X_0(n)$, $V = \pi(n)$, and postmultiplying by $x_{p,q}(n)$ leads to

$$B(n) = \lambda^n B(n-1) + \frac{g_L(n)}{\gamma_L(n)} e_b^T(n \mid n)$$

Comparing (5.29) and (5.30) it is seen that $g_L(n)$ and $B(n)$ are defined in terms of one another and therefore (5.29) can not be used in its present form. This is resolved by substituting (5.30) for $B(n)$ in (5.29) after the $e_b(n \mid n)$ and $\gamma_L(n)$ updates have been derived.

To derive the $e_b(n \mid n)$ update, take the transpose of (5.30), postmultiply by $x_L(n)$ from (2.27), and subtract the result from $x_{p,q}(n)$, where

$$x_{p,q}(n) = [y(n-p), s(n-q)]^T$$

is the 2x1 vector containing the last two elements of $x_{p,q}(n)$. After simplifying the result, this process leads to

$$e_b(n \mid n) = \gamma_L(n)e_b(n \mid n-1)$$

where, similar to $e_f(n \mid n-1)$,
\[ e_b(n \mid n-1) = x_p(n) - \lambda^n B^T(n-1)x_L(n) \] (5.32)

To update \( \gamma_L(n) \) first recognize that, analogous to (4.9), \( \gamma_{L+2}(n) \) is defined by

\[ \gamma_{L+2}(n) = 1 - x_L^T(n)g_{L+2}(n) \] (5.33)

Next, note that the \( L + 2 \) length vector \( x_{L+2}(n) \) can be partitioned in two ways:

\[ x_{L+2}(n) = Q_f \begin{bmatrix} x_0(n) \\ x_L(n-1) \end{bmatrix} \] (5.34a)

\[ x_{L+2}(n) = Q_b \begin{bmatrix} x_L(n) \\ x_p(n) \end{bmatrix} \] (5.34b)

Using (5.34a), (5.10), and (5.14) in (5.33) then leads to

\[ \gamma_{L+2}(n) = \gamma_L(n-1) - e_f^T(n \mid n)R_f^{-1}(n)e_f^*(n \mid n) \] (5.35)

from which using (5.27) gives

\[ \gamma_{L+2}(n) = [1 - e_f^T(n \mid n-1)R_f^{-1}(n)e_f^*(n \mid n)]\gamma_L(n-1) \] (5.36)

Then using (5.34b) in (5.33) provides

\[ \gamma_{L+2}(n) = \gamma_L(n) - e_b^T(n \mid n)R_b^{-1}(n)e_b^*(n \mid n) \] (5.37)

from which using (5.31) and (5.21) gives

\[ \gamma_L(n) = [1 - e_b^T(n \mid n-1)f(n)]^{-1}\gamma_{L+2}(n) \] (5.38)

Equations (5.36) and (5.38) thus provide a two step method for updating \( \gamma_L(n) \) from \( \gamma_L(n-1) \).

At this point it is now simple to derive the update for \( g_L(n) \). Substitute (5.30) and (5.31) into (5.29) and then use (5.38) in the simplification, obtaining

\[ g_L(n) = \frac{\gamma_L(n)}{\gamma_{L+2}(n)} [f_L(n) + \lambda^n B(n-1)f(n)] \] (5.39)

With (5.39), the derivation of all the relations needed for the complete complex, time-varying, pole-zero systems identification algorithm has been completed. For convenience, the complete set of recursions is listed in Table 1, together with a simple initialization scheme. This initialization is very simple computationally, but does perturb the computed
solution slightly from the true LS solution during start-up. However, this effect diminishes rapidly as a few samples are acquired. It should be mentioned that the initialization of recursive LS filters is still an active area of research [6,14].

6. SUMMARY AND FUTURE RESEARCH

This paper has presented a derivation using geometrical methods of a fast pole-zero identification algorithm for complex, time-varying systems. Currently, research is being conducted on analytically determining the ability of the method to track the changing systems parameters. One important topic being investigated concerns the relative sensitivity of the method to perturbations in the AR and MA parameters separately.

Additionally, there are several questions remaining concerning the numerical stability (i.e., the finite precision arithmetic effects) of the pole-zero algorithm, since many of the so-called "fast" algorithms have tended toward numerical instability. Work is currently proceeding to theoretically analyze these numerical effects for the fast, complex, pole-zero time-varying algorithm. In fact, the entire area of numerical effects for adaptive filters is currently an important and active research area for many adaptive methods, as pointed out in the excellent survey article by Cioffi [20].
REFERENCES


