An Algorithm for Obtaining the Correlation Functions Between Forward and Backward Prediction Error Filters

Suppose we have a sequence of N real reflection coefficients, C_1, C_2, C_3, ..., C_N. The C's are thus real numbers with magnitudes less than unity. This sequence can be used to generate a sequence of prediction error filters, where the first prediction error filter is $1 + C_1 z$ in z-transform notation. The higher order P.E. filters are then obtained by the following recursion algorithm. If $1 + \Gamma_1^m z + \Gamma_2^m z^2 + \ldots + \Gamma_m^m z^m$ is the mth P.E.F., then the $m+1$ th filter is obtained by

$$\begin{Bmatrix} 1 \\ \Gamma_1^{m+1} \\ \Gamma_2^{m+1} \\ \vdots \\ \Gamma_m^{m+1} \\ \Gamma_{m+1}^{m+1} \end{Bmatrix} = \begin{Bmatrix} 1 \\ \Gamma_1^m \\ \Gamma_2^m \\ \vdots \\ \Gamma_m^m \\ 0 \end{Bmatrix} + C_{m+1} \begin{Bmatrix} 0 \\ \Gamma_m^m \\ \Gamma_{m-1}^m \\ \vdots \\ \Gamma_1^m \\ 1 \end{Bmatrix} .$$

We note that $\Gamma_{m+1}^{m+1} = C_{m+1}$.

Suppose we have a stationary time series, x_n, whose autocorrelation function is $\Phi(\tau)$, $\tau=0, 1, 2, \ldots$, and which has no particular relation with the above sequence of P.E.F. Suppose that the physically realizable mth P.E.F. is convolved with this time series as if we were trying to generate the forward prediction error time series. This forward prediction error time series, u_n, does not have minimum power or a white spectrum unless the

P.E.F. happened to be the optimum least mean square error P.E.F. corresponding to $\Phi(\tau)$. We are explicitly not assuming this to be the case. Thus the autocorrelation function of u_n is basically arbitrary but can be calculated from knowledge of $\Phi(\tau)$ and the P.E.F.

We also can flip the P.E.F. over so that it is completely non-physically realizable and convolve this backward P.E.F. with x_n . The autocorrelation function of this backward prediction error time series, w_n , is the same as that of the u_n time series. However, the cross-correlation function, $\Psi(\tau)$, between u_n and w_n is a new statistic and can also be determined from knowledge of $\Phi(\tau)$ and the P.E.F. Thus, given $\Phi(\tau)$, for each P.E.F. there is a corresponding pair of functions, one an autocorrelation function and the other a cross-correlation function. The sequence of P.E. filters thus generates a sequence of correlation function pairs. The purpose of this paper is to show how this sequence of function pairs can be calculated recursively, given $\Phi(\tau)$, $\tau=0, 1, \ldots, N$ and $C_1, C_2, \ldots, C_N$.

Consider the time series, x_n, and the two filter outputs u_n and w_n as shown in Figure 1 for the 1st order P.E.F.

...	w_{-2}	w_{-1}	w_0	w_1	w_2	w_3	w_4	...
			1	C_1				
...	x_{-2}	x_{-1}	x_0	x_1	x_2	x_3	x_4	...
					C_1	1		
...	u_{-2}	u_{-1}	u_0	u_1	u_2	u_3	u_4	...

Figure 1

The autocorrelation function for this first order case is defined to be

$$\Phi_1(\tau) = \overline{u_n\, u_{n+\tau}} = \overline{w_n\, w_{n+\tau}}$$

and the cross-correlation function is defined to be $\Psi_1(\tau) = \overline{w_n u_{n+\tau}}$, where the bars indicate average values. For example, in figure 1 the relative location of the backward and forward filters are such that the average product of their outputs is $\Psi_1(3)$. (We assume the output of a P.E.F. is at the location of the unity weight.)

To develop the algorithm and to provide a self evident proof, consider Figure 2.

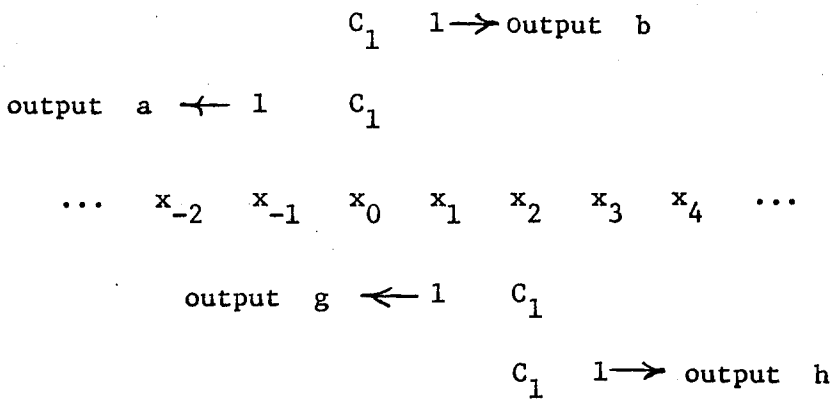

Figure 2

In this figure, four filtered outputs, a, b, g, and h, are shown, two from forward first order filters and two from backward first order filters with the relative displacements given by the relative locations of the unity weights in the filters. Figure 2 is set up to illustrate how we go from the first order correlation functions $\Phi_1(\tau)$ and $\Psi_1(\tau)$ to the second order correlation functions $\Phi_2(\tau)$ and $\Psi_2(\tau)$. In particular, to obtain $\Phi_2(2)$, we can use C_2 to combine the filters which produce outputs a and b to get the second order backward P.E.F. which has output $a + C_2 b$, (see Figure 3).

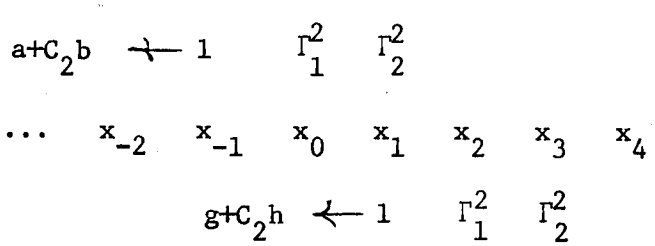

Figure 3

Likewise, the g and h output filters can be combined to get the second order backward filter which is delayed by two steps relative to the top filter, as shown in Figure 3. The correlation between these two outputs is $\Phi_2(2)$ and can be calculated by noting that

$$(a + C_2 b) * (g + C_2 h) =$$

$$a * g + C_2 b * g + C_2\, a * h + C_2^2\, b * h$$

$$= \Phi_1(2) + C_2\, \Psi_1(0) + C_2\, \Psi_1(4) + C_2^2\, \Phi_1(2) ,$$

where the $*$ stands for the correlation operation.

Figure 4 shows how the $\Psi_2(4)$ value is obtained as

$$(a + C_2 b) * (h + C_2 g) =$$

$$a * h + C_2 b * h + C_2\, a * g + C_2^2\, b * g$$

$$= \Psi_1(4) + C_2\, \Phi_1(2) + C_2\, \Phi_1(2) + C_2^2\, \Psi_1(0) .$$

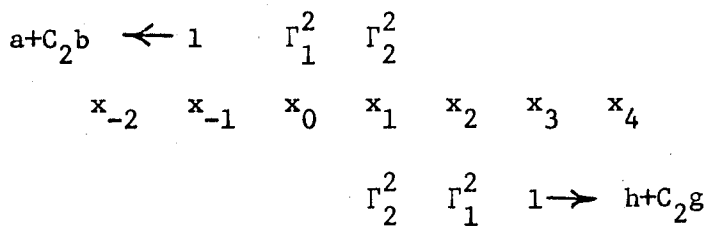

Figure 4

Likewise Figure 5 shows how the $\Psi_2(0)$ value is obtained as

$$(g + C_2 h) * (b + C_2 a) =$$

$$g * b + C_2 h * b + C_2 g * a + C_2^2 \, h * a$$

$$= \Psi_1(0) + C_2 \, \Phi_1(2) + C_2 \, \Phi_1(2) + C_2^2 \, \Psi_1(4) .$$

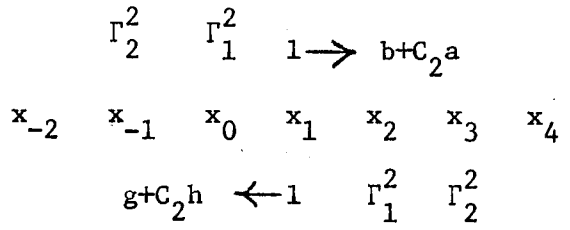

Figure 5

The fourth possible combination leads again to the calculation of $\Phi_2(2)$.

These equations for going from the first order to the second order functions can be generalized for any lag value to give

$$\Phi_2(\tau) = \Phi_1(\tau) + C_2 \, \Psi_1(2-\tau) + C_2 \, \Psi_1(2+\tau) + C_2^2 \, \Phi_1(\tau)$$

$$\Psi_1(2+\tau) = \Psi_1(2+\tau) + C_2 \, \Phi_1(\tau) + C_2 \, \Phi_1(\tau) + C_2^2 \, \Psi_1(2-\tau)$$

$$\Psi_1(2-\tau) = \Psi_1(2-\tau) + C_2 \, \Phi_1(\tau) + C_2 \, \Phi_1(\tau) + C_2^2 \, \Psi_1(2-\tau) .$$

To see how to generalize these equations to the case of going from the mth order to the m+1 th order, we can study Figure 6 which shows the case

of going from the second order to the third order.

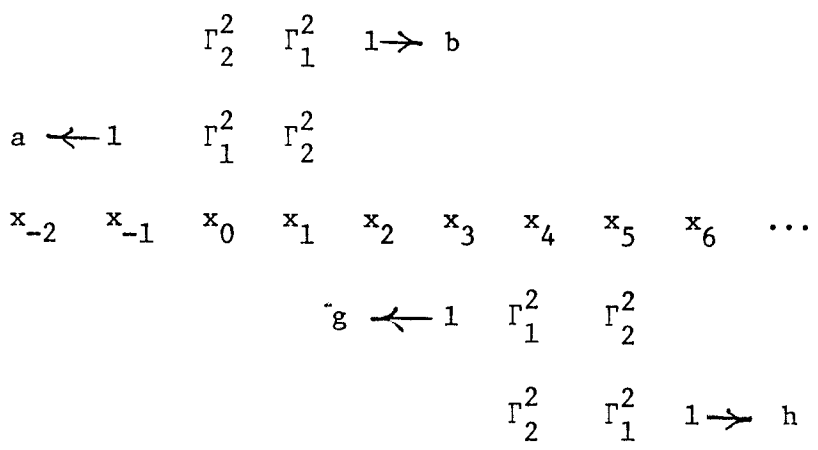

Figure 6

Here, by careful inspection and noting that Figure 6 corresponds to the particular case of $\tau=4$,

$$\Phi_3(\tau) = \Phi_2(\tau) + C_3\,\Psi_2(3-\tau) + C_3\,\Psi_2(3+\tau) + C_3^2\,\Phi_2(\tau)$$

$$\Psi_3(3+\tau) = \Psi_2(3+\tau) + C_3\,\Phi_2(\tau) + C_3\,\Phi_2(\tau) + C_3^2\,\Psi(3-\tau)$$

$$\Psi_3(3-\tau) = \Psi_2(3-\tau) + C_3\,\Phi_2(\tau) + C_3\,\Phi_2(\tau) + C_3^2\,\Psi(3+\tau)\ .$$

We can now see that the general recursion equations in going from the m th to the m+1 th case for any lag τ must be

$$\Phi_{m+1}(\tau) = (1+C_{m+1}^2)\,\Phi_m(\tau) + C_{m+1}(\Psi_m(m+1+\tau) + \Psi_m(m+1-\tau))$$

$$\Psi_{m+1}(m+1\pm\tau) = \Psi_m(m+1\pm\tau) + 2\,C_{m+1}\,\Phi_m(\tau) + C_{m+1}^2\,\Psi_m(m+1\mp\tau)\ .$$

These three equations are the basic recursion equations of this paper. We next wish to set up a computer algorithm for calculating the correlation functions recursively. We shall use a three column array for in place storage of results.

Figure 7 shows this array with the values it contains after the mth step in the algorithm. The original autocorrelation function is assumed to be known out to a maximum lag of N . Note that initially for m=0, the table contains only values from the original autocorrelation function, since $\Psi_0(\tau) = \Phi_0(\tau)$. (Figure 10 needs to be studied to obtain the complete array values after any step in the recursion.)

$\Psi_m(N)$	$\Phi_m(N-1-m)$	$\Psi_m(2-N+2m)$
$\Psi_m(N-1)$	$\Phi_m(N-2-m)$	$\Psi_m(3-N+2m)$
⋮	⋮	⋮
$\Psi_m(m+1+\tau)$	$\Phi_m(\tau)$	$\Psi_m(m+1-\tau)$

Figure 7

To show the basic transition step in the algorithm, Figure 8 blocks out the replacement process involved in using the three basic recursion equations with $\tau=1$ and $m=1$.

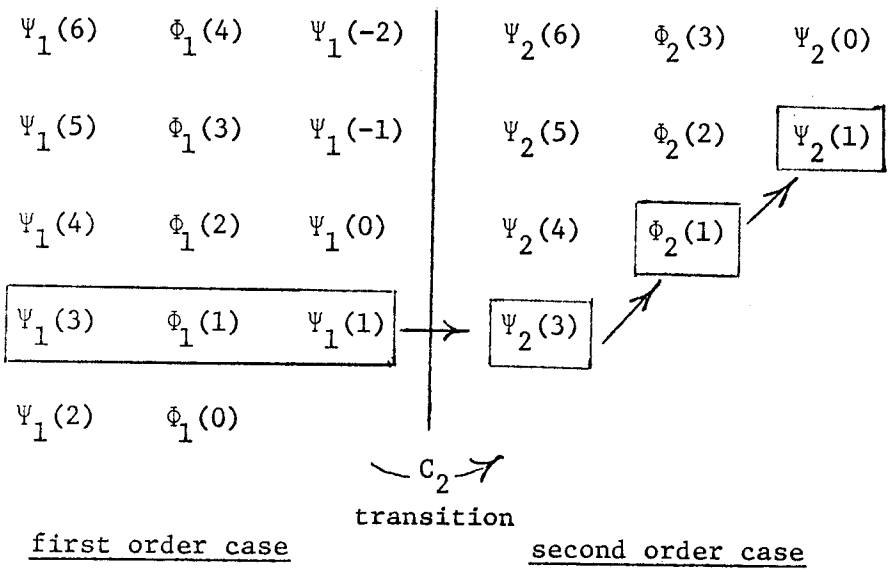

Figure 8

By using this upward shifting in the replacement procedure, the three terms on any row in the new array are then ready to be combined in the next transition from m to m+1 . For example, $\Psi_2(5)$, $\Phi_2(2)$ and $\Psi_2(1)$ are the three terms needed in the basic recursive equations when m=2 and τ=2 . To avoid wasteful temporary shortage, we should note that the table replacement proceeds downward from the top of the table. Figure 9 given the order in which the new terms in the table should be calculated.

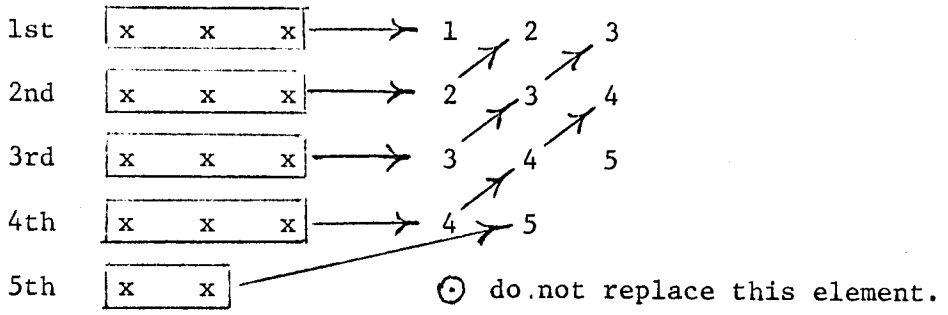

Figure 9

We note that steps 1, 2 and the last one can be specialized for efficiency. In particular, the last step should not replace the element in the left hand column. Also, the table is one shorter after each transition from m to m+1 . This shortening is expected if one considers that if the original autocorrelation function is known out to lag N , then the autocorrelation function for the first order P.E.F. can be calculated only out to lag N-1 and so forth. As a final observation, if the algorithm is run to completion, the final values in the array are as shown in Figure 10.

$$
\begin{array}{lll}
\Psi_{N-1}(N) & \Phi_{N-1}(0) & \Psi_{N-2}(N-2) \\
\Psi_{N-2}(N-1) & \Phi_{N-2}(0) & \Psi_{N-3}(N-3) \\
\Psi_{N-3}(N-2) & \Phi_{N-3}(0) & \Psi_{N-4}(N-4) \\
\vdots & \vdots & \vdots \\
\Psi_{1}(2) & \Phi_{1}(0) & \Psi_{0}(0) \\
\Psi_{0}(1) & \Phi_{0}(0) &
\end{array}
$$

Figure 10

The first and second columns of figure 10 are the most interesting since the $\Psi_{m-3}(m-2)$ is the correlation between the forward and backward m-3th order P.E.F. when they are in the correct relative position to be combined into the next higher order forward and backward P.E.F. In particular, if our reflection coefficients are not given initially, but are calculated as needed from the table by the formula, $C_n = -\Psi_{n-1}(n)/\Phi_{n-1}(0)$,

we will actually be calculating the reflection coefficients corresponding to $\Phi_0(\tau)$ without calculating the prediction error filters.

To derive the formula for the number of multiplications required to do the algorithm, we first look at the basic recursion equations from a programming point of view. The basic equations can be written as

$$\Phi_{m+1}(\tau) = \Phi_m(\tau) + C_{m+1} * [\ \Psi_m(m+1+\tau) + \Psi_m(m+1-\tau) + Q\ ]$$

$$\Psi_{m+1}(m+1+\tau) = \Psi_m(m+1+\tau) + Q + Q + C^2 * \Psi_m(m+1-\tau)$$

$$\Psi_{m+1}(m+1-\tau) = \Psi_m(m+1-\tau) + Q + Q + C^2 * \Psi_m(m+1+\tau)$$

where $Q = C_{m+1} * \Phi_m(\tau)$. Thus in the main loop, only four multiplications $(*)$ are required to exercise the basic recursion triplet. To run the algorithm to completion, this triplet of equations must be used $(N-1) + \ldots + 2 + 1$ times or $N(N-1)/2$ times. Thus, the number of multiplications in the main loop is $2N(N-1)$. This can be compared with $N(N-1)$ multiplications involved in calculating the P.E.F. corresponding to $\Phi_0(\tau)$ using the Levinson algorithm. Of course, in this latter case the special relationship of the C_n to $\Phi_0(\tau)$ allow the factor of two saving in the number of multiplications.