Elec4621:
Advanced Digital Signal Processing
Chapter 8: Wiener and Adaptive Filtering

Dr. D. S. Taubman

May 2, 2011

1 Wiener Filtering

A Wiener filter is one which provides the Minimum Mean Squared Error (MMSE) prediction, $\bar{y}[n]$, of the outcomes, $y[n]$, from one random process, $Y[n]$, using knowledge of the outcomes, $x[n]$, from another random process, $X[n]$. Of course, the random processes $X[n]$ and $Y[n]$ cannot be completely independent (or uncorrelated) for the prediction to be of any use.

As in our previous study of linear prediction, we will assume that all random processes are WSS and we shall confine our attention to predictors which are linear functions of the input samples, $x[n]$ through $x[n-N]$. Under these conditions, our objective is to find the coefficients a_0 through a_N, which minimize the expected power of the error, $e[n] = y[n] - \bar{y}[n]$. That is, we wish to minimize

$$\begin{aligned} J &= \sigma_E^2 = E\left[\left(Y[n] - \bar{Y}[n]\right)^2\right] \\ &= E\left[\left(Y[n] - \sum_{k=0}^{N} a_k X[n-k]\right)^2\right] \end{aligned}$$

By direct differentiation, or by application of the orthogonality principle, we find that the minimizing solution must satisfy

$$\begin{aligned} 0 &= E\left[E[n]\, X[n-p]\right] \\ &= E\left[\left(Y[n] - \sum_{k=0}^{N} a_k x[n-k]\right) X[n-p]\right] \\ &= R_{YX}[p] - \sum_{k=0}^{N} a_k R_{XX}[p-k], \quad 0 \le p \le N \end{aligned}$$

which may be expressed in matrix form as

$$\underbrace{\begin{pmatrix} R_{XX}[0] & R_{XX}[1] & \cdots & R_{XX}[N] \\ R_{XX}[1] & R_{XX}[0] & \cdots & R_{XX}[N-1] \\ \vdots & \vdots & \ddots & \vdots \\ R_{XX}[N] & R_{XX}[N-1] & \cdots & R_{XX}[0] \end{pmatrix}}_{R_{\mathbf{X}}} \underbrace{\begin{pmatrix} a_0 \\ a_2 \\ \vdots \\ a_N \end{pmatrix}}_{\mathbf{a}} = \underbrace{\begin{pmatrix} R_{YX}[0] \\ R_{YX}[1] \\ \vdots \\ R_{YX}[N] \end{pmatrix}}_{\mathbf{r}} \tag{1}$$

The coefficients a_0 through a_N are the filter taps, $h[n] = a_n$, of an FIR filter which produces $\bar{y}[n]$ when applied to the source sequence, $x[n]$. This filter is known as the N^{th} order Wiener filter.

1.1 Relationship to Linear Prediction

It should be obvious from the form of the solution above that Wiener filtering and linear prediction are close relatives. In fact, ordinary linear predictor is a special case of the Wiener filtering problem, in which the target random process is $Y[n] = X[n+1]$. In this case,

$$R_{YX}[m] = E[Y[n]\,X[n-m]\,] = E[X[n+1]\,X[n-m]\,] = R_{XX}[m+1]$$

Substituting this into equation (1), we obtain the normal equations of an $N+1^{\text{st}}$ order linear predictor.

The fact that both Wiener filtering and linear prediction require us to invert the same symmetric, positive definite, Toeplitz matrix, $R_{\mathbf{X}}$, suggests that whatever good mechanisms we find for doing linear prediction should be useful also for Wiener filtering. This is indeed the case. In fact, we shall find that adaptive linear predictors lie at the heart of the best adaptive Wiener filters. We shall temporarily defer further consideration of this connection, however, until Section 5.

1.2 Unconstrained Wiener Filters

It is instructive to consider what happens if we abandon the requirement that the Wiener filter be finite in extent, or even causal. In this case, we seek any impulse response sequence, $h_\infty[n]$, which minimizes the expected squared error,

$$J = E\left[\left(Y[n] - \sum_{k=-\infty}^{\infty} h_\infty[k]\,X[n-k]\right)^2\right]$$

and again the solution is given by applying the orthogonality principle, from which we get

$$\begin{aligned} 0 &= E\left[\left(Y[n] - \sum_{k=-\infty}^{\infty} h_\infty[k]\, X[n-k]\right) X[n-p]\right] \\ &= R_{YX}[p] - \sum_{k=-\infty}^{\infty} h_\infty[k]\, R_{XX}[p-k] \\ &= R_{YX}[p] - (h_\infty \star R_{XX})[p], \quad \forall p \in \mathbb{Z} \end{aligned}$$

In the Fourier domain, the above relationship becomes

$$\hat{h}_\infty(\omega) = \frac{S_{YX}(\omega)}{S_{XX}(\omega)} \tag{2}$$

where $S_{YX}(\omega)$ is the cross-power spectral density between $Y[n]$ and $X[n]$, and $S_{XX}(\omega)$ is the power spectral density of $X[n]$.

Equation (2) provides us with a simple expression for the unconstrained MMSE solution to the general linear prediction problem. Taking the inverse DTFT of $\hat{h}_\infty(\omega)$ generally yields an infinitely long, non-causal impulse response, which cannot be practically implemented. However, we could consider designing a realizable filter (FIR or IIR) to match $\hat{h}_\infty(\omega)$ as closely as possible. As we shall see in Section 1.3, the constrained Wiener solution in equation (1) can be viewed as the result of such a filter design procedure.

Equation (2) is of particular interest in many problems since it provides us with an explanation of what the Wiener filter is trying to do. As you know, the best way to understand filters is in the frequency domain, since that is where there behaviour can be described simply as that of selectively attenuating (or amplifying) individual frequency components. Equation (2) tells us that the optimal Wiener filter should attenuate each frequency component by the ratio between the cross- and auto-power spectra, $S_{YX}(\omega)$ and $S_{XX}(\omega)$.

It is also worth deriving an expression for the prediction error power spectrum, based on the ideal Wiener filter, $\hat{h}_\infty(\omega)$. To do this, note that

$$\begin{aligned} R_{EE}[m] &= E[E[n]\, E[n-m]\,] \\ &= E\left[\left(Y[n] - \bar{Y}[n]\right)\left(Y[n-m] - \bar{Y}[n-m]\right)\right] \\ &= R_{YY}[m] + R_{\bar{Y}\bar{Y}}[m] - R_{\bar{Y}Y}[m] - R_{Y\bar{Y}}[m] \end{aligned}$$

Taking Fourier transforms, we get

$$\begin{aligned} S_{EE}(\omega) &= S_{YY}(\omega) + S_{\bar{Y}\bar{Y}}(\omega) - 2\Re\left(S_{\bar{Y}Y}(\omega)\right) \\ &= S_{YY}(\omega) + \left|\hat{h}_\infty(\omega)\right|^2 S_{XX}(\omega) - 2\Re\left(\hat{h}_\infty(\omega) S_{XY}(\omega)\right) \\ &= S_{YY}(\omega) + \frac{|S_{YX}(\omega)|^2}{S_{XX}(\omega)} - 2\Re\left(\frac{S_{YX}(\omega) S_{XY}(\omega)}{S_{XX}(\omega)}\right) \\ &= S_{YY}(\omega) - \frac{|S_{YX}(\omega)|^2}{S_{XX}(\omega)} \\ &= S_{YY}(\omega)\left(1 - \frac{|S_{YX}(\omega)|^2}{S_{YY}(\omega) S_{XX}(\omega)}\right) \end{aligned}$$

The ratio $|S_{YX}(\omega)|^2$ over $S_{YY}(\omega) S_{XX}(\omega)$ is a measure of how correlated X and Y are at any particular frequency. Its value is guaranteed to lie in the range 0 (completely uncorrelated) to 1 (perfectly correlated).

1.3 Relationship to Filter Design

Equation (1) bears a striking resemblance to the equations we obtained previously in connection with optimal FIR filter design, based upon a weighted mean squared error criterion. In particular, if $\hat{h}_d(\omega)$ is any desired frequency response and $\hat{\rho}(\omega)$ a frequency weighting function, the FIR least squares design objective is to find an order N filter, $h[n]$, which minimizes

$$\mathcal{E} = \frac{1}{2\pi}\int_{-\pi}^{\pi} |\hat{\rho}(\omega)|^2 \left|\hat{h}(\omega) - \hat{h}_d(\omega)\right|^2 d\omega$$

We found that this problem leads to the following linear equations

$$\underbrace{\begin{pmatrix} r[0] & r[1] & \cdots & r[N] \\ r[1] & r[0] & \cdots & r[N-1] \\ \vdots & \vdots & \ddots & \vdots \\ r[N] & r[N-1] & \cdots & r[0] \end{pmatrix}}_{R} \underbrace{\begin{pmatrix} a_0 \\ a_1 \\ \vdots \\ a_N \end{pmatrix}}_{\mathbf{a}} = \underbrace{\begin{pmatrix} d[0] \\ d[1] \\ \vdots \\ d[N] \end{pmatrix}}_{\mathbf{d}} \tag{3}$$

where $r[n]$ is the sequence whose DTFT is

$$\hat{r}(\omega) = |\hat{\rho}(\omega)|^2$$

and $d[n]$ is the sequence whose DTFT is

$$\hat{d}(\omega) = |\hat{\rho}(\omega)|^2 \hat{h}_d(\omega)$$

Comparing equations (1) and (3) we see that the optimal filter design strategy will design the optimal N^{th} order Wiener filter, if we set

$$r[n] = R_{XX}[n] \quad \Longrightarrow \quad |\hat{\rho}(\omega)|^2 = S_{XX}(\omega)$$

and

$$\begin{aligned} d[n] = R_{YX}[n] \quad &\Longrightarrow \quad |\hat{\rho}(\omega)|^2 \hat{h}_d(\omega) = S_{YX}(\omega) \\ &\Longrightarrow \quad \hat{h}_d(\omega) = \frac{S_{YX}(\omega)}{S_{XX}(\omega)} = \hat{h}_\infty(\omega) \end{aligned}$$

This is a most interesting result. It tells us firstly that the optimal constrained Wiener filter, $h[n]$, may be understood as the best approximation to the ideal unconstrained Wiener filter, $h_\infty[n]$, in the sense of minimizing a weighted squared error between $\hat{h}(\omega)$ and $\hat{h}_\infty(\omega)$.

Secondly, it tells us that the weighting function, $|\hat{\rho}(\omega)|^2$ which makes the filter design and the direct Wiener filter design problems identical is the power spectrum of the source process, $X[n]$. That is, at frequencies where $X[n]$ has very little power, the accuracy with which $\hat{h}(\omega)$ approximates $\hat{h}_\infty(\omega)$ is not very important. This makes perfect sense, since if very little power is going into the Wiener filter at some frequencies, it does not much matter what the filter does with those frequency components. Notice that the weighting function has no dependence at all upon the statistics of the target random process, $Y[n]$.

Thirdly, recall that the minimum mean squared error FIR filter design procedure reduces to windowing with the all-or-nothing window, if and only if the frequency weighting function is uniform. But this never happens in practical applications of Wiener filtering, since $S_{XX}(\omega)$ is uniform only if $X[n]$ is an uncorrelated (white) random process. Thus, the optimal N^{th} order Wiener filter is always better (usually much better) than the filter obtained by simply setting $h[n] = h_\infty[n]$ over the region of support, $0 \leq n \leq N$.

2 The LMS Algorithm

As we have seen, Wiener filters depend upon knowledge of the cross- and autocorrelation functions, $R_{YX}[m]$ and $R_{XX}[m]$. In some cases we may be able to estimate these functions ahead of time and fix the Wiener filter accordingly. More commonly though, the statistics may vary slowly with time, and we must adapt the Wiener filter coefficients on the basis of the outcomes which have been seen so far. One way to do this is to divide the time axis into blocks, of length L. In each block, we might collect L samples from the sequences $x[n]$ and $y[n]$, from which we estimate the correlation statistics and find the Wiener coefficients to be used in the next block. Strategies of this form are called block-adaptive.

One drawback of block-adaptive filters is that the filter coefficients are only able to respond to changes in the underlying statistics at block boundaries. We could consider estimating the statistics based on a moving window of length L, recalculating the statistics and the Wiener coefficients after each new sample arrives, but this would be extremely costly from a computational perspective. Block-adaptive schemes may also require substantial memory to hold the past outcomes used to estimate the correlation statistics.

In this section, we describe a simple algorithm, which can be used to incrementally adjust the Wiener coefficients as each new sample arrives. The

algorithm does not require us to keep track of past outcomes for the purpose of estimating statistics, although the algorithm is generally quite slow to "learn" or adapt to changes in the statistics.

The so-called LMS (Least Mean Squares) algorithm uses a simple gradient descent strategy to incrementally adjust the coefficient vector, $\mathbf{a} = (a_0, \ldots, a_N)^t$, towards the minimum of our quadratic objective function,

$$J(a_0, a_1, \ldots, a_N) = E\left[\left(Y[n] - \sum_{k=0}^{N} a_k X[n-k]\right)^2\right]$$

The gradient vector,

$$\nabla J = \begin{pmatrix} \frac{\partial J}{\partial a_0} \\ \frac{\partial J}{\partial a_1} \\ \vdots \\ \frac{\partial J}{\partial a_N} \end{pmatrix}$$

points in the direction of steepest ascent of the function $J(\mathbf{a})$, so that $-\nabla J$ points in the direction of steepest descent. The idea behind the LMS algorithm is to update the coefficients according to

$$\mathbf{a}^{(n+1)} = \mathbf{a}^{(n)} - \delta \cdot \nabla J^{(n)}$$

where $\nabla J^{(n)}$ is an estimate of ∇J formed at time n, $\mathbf{a}^{(n)}$ is the vector of Wiener coefficients at time n, and δ is a small positive constant which controls the rate at which the Wiener coefficients converge to the minimum of the quadratic function, $J(\mathbf{a})$. The following example illustrates these concepts for the simplest case in which there is only one coefficient to be adapted.

Example 1 *Consider the parabolic function, $J(a_0) = \alpha(a_0 - \beta)^2 + \gamma$, illustrated in Figure 1. It is important to realize that the parameters, α, β and γ are not actually known to the LMS algorithm. If they were, we could immediately recognize that $J(a_0)$ is minimized by setting $a_0 = \beta$.*

Now let $a_0^{(n)}$ denote the n^{th} estimate of a_0, starting from some arbitrary position, $a_0^{(0)}$. If we ignore any inaccuracies in our estimate of the gradient (there will be substantial inaccuracies in practice), the n^{th} gradient estimate becomes

$$\nabla J^{(n)} = 2\alpha\left(a_0^{(n)} - \beta\right)$$

and the gradient descent algorithm incrementally adjusts a_0 according to

$$\begin{aligned} a_0^{(n+1)} &= a_0^{(n)} - \delta \cdot \nabla J^{(n)} \\ &= a_0^{(n)} - 2\delta\alpha\left(a_0^{(n)} - \beta\right) \end{aligned}$$

Evidently, if we are so lucky as to select $\delta = \frac{1}{2\alpha}$, we immediately hit the jackpot, getting $a_0^{(n+1)} = \beta$, after which all subsequent gradient estimates will be 0. If

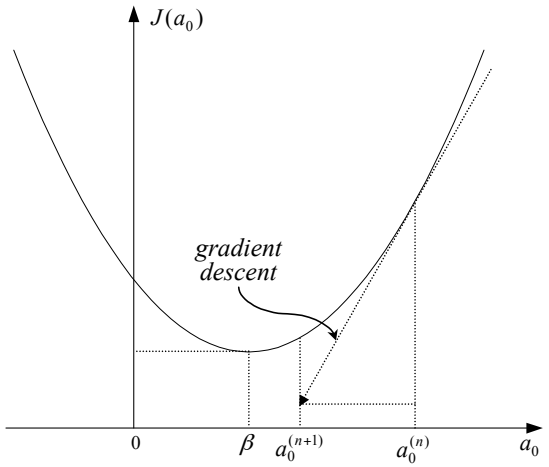

Figure 1: *Illustration of gradient descent of the one dimensional quadratic function, $J(a_0)$.*

we select $\delta < \frac{1}{2\alpha}$, we move toward the minimizing solution $a_0 = \beta$, while if we select $\delta > \frac{1}{2\alpha}$, we overshoot, and may even wind up with $a_0^{(n+1)}$ a worse solution than $a_0^{(n)}$. The challenge is to keep δ small enough that it is likely to be smaller than $\frac{1}{2\alpha}$.

Now returning to our adaptive Wiener filtering problem, we have

$$\nabla J = \begin{pmatrix} \frac{\partial J}{\partial a_0} \\ \frac{\partial J}{\partial a_1} \\ \vdots \\ \frac{\partial J}{\partial a_N} \end{pmatrix} = -2 \begin{pmatrix} E\left[E\left[n\right] X\left[n-0\right]\right] \\ E\left[E\left[n\right] X\left[n-1\right]\right] \\ \vdots \\ E\left[E\left[n\right] X\left[n-N\right]\right] \end{pmatrix} = -2 \begin{pmatrix} R_{EX}\left[0\right] \\ R_{EX}\left[1\right] \\ \vdots \\ R_{EX}\left[2\right] \end{pmatrix}$$

To accurately estimate ∇J, we would need to estimate the cross-correlation between the source process, $X\left[n\right]$, and the error process, $E\left[n\right]$, produced by the current set of filter parameters. But we keep changing the filter parameters. Instead, the LMS algorithm works with the following very crude approximation

$$\nabla J^{(n)} = -2 \begin{pmatrix} e\left[n\right] x\left[n-0\right] \\ e\left[n\right] x\left[n-1\right] \\ \vdots \\ e\left[n\right] x\left[n-N\right] \end{pmatrix}$$

That is, the LMS algorithm uses the instantaneous product between $e\left[n\right]$ and $x\left[n-m\right]$ in place of the statistical expectation or any kind of time average.

Of course, this approximation is likely to be wildly inaccurate. However, if we make only very small adjustments to the filter coefficients at each time

step (i.e., if δ is very small), the instantaneous approximation errors in $\nabla J^{(n)}$ can be expected to average out, so that we move slowly toward the optimal filter coefficients. When we get there, ∇J should be 0, but our instantaneous estimates, $\nabla J^{(n)}$, will not generally be zero, so the filter coefficients will fluctuate about their optimal values. The amount of fluctuation depends upon the descent parameter, δ. If δ is very small, the adaptive Wiener filter will converge only slowly to the optimal solution, but the coefficients will then remain very stable, unless or until the statistics change. If δ is larger, convergence is faster and the filter can track changes in the statistics more rapidly, but the coefficients will tend to bounce around their optimal values, leading to sub-optimal prediction.

If δ is too large, convergence might not be achieved at all, as suggested by Example 1, and the adaptive algorithm will be unstable. It can be shown, however, that the LMS algorithm will be stable so long as the descent parameter, δ, satisfies

$$0 < \delta < \frac{1}{\text{total input power}} = \frac{1}{(N+1)\, R_{XX}[0]}$$

where $R_{XX}[0]$ is the average power of each of the $N+1$ inputs to the Wiener filter.

3 Some Applications of Wiener Filtering

3.1 Deconvolution

A linear time-invariant system produces an output $r[n]$ (r for "response" or "received") from its input $s[n]$ (s for "source" or "sent"), which can be described by convolution,

$$r[n] = \sum_k h_s[k]\, s[n-k]$$

where $h_s[k]$ is the impulse response of the system. The object of deconvolution is to recover $s[n]$ from $r[n]$. This problem occurs in any application where the desired signal has been corrupted by a linear time-invariant system.

Example 2 *An audio recording is made using an inexpensive microphone, whose transfer function, $\hat{h}(\omega)$, has a highly non-uniform magnitude response. The input to the microphone may be modeled as the desired signal, $s[n]$, and the poor quality recording is $r[n]$. The task of enhancing the recording may be understood as that of recovering $s[n]$ from $r[n]$. You first measure the impulse response of the microphone, $h_s[k]$ – useful methods for this were suggested in Tutorial 5. You then need to deconvolve.*

Example 3 *An amplitude modulation system generates an output waveform, $s[n] = \sum_k A_k g[n - Pk]$ where A_k is an information bearing sequence of scaling factors, $g[n]$ is a "shaping" pulse, which determines the spectral properties of the modulated waveforms, and P is the symbol clock period, which is the number of time steps, n, between each pair of transmitted symbols, A_k. If this signal is*

corrupted by additive white noise, the most robust demodulation strategy is to estimate A_k as

$$\begin{aligned} A'_k &= \frac{1}{E_g} \sum_{n=0}^{L} g\,[n]\, s[Pk+n] \\ &= \frac{1}{E_g} \left(s \star \tilde{g}\right)[Pk] \end{aligned} \tag{4}$$

where the shaping pulse, $g\,[n]$, has support $0 \leq n \leq L$, and $E_g = \sum_n g^2\,[n]$ is the energy of $g\,[n]$. This is known as matched filtering.

In practice, however, the transmitted waveform is also subject to distortion in the communications channel. The channel distortion can often be modeled as an LTI system, with transfer function $\hat{h}_s\,(\omega)$, which converts the original modulated waveform, $s\,[n]$, to a received waveform, $r\,[n]$. In order to robustly recover the encoded sequence, A_k, the channel distortion must first be undone, which is a deconvolution problem. In many cases, the channel transfer function must be adaptively estimated, in which case the deconvolution problem is known as "adaptive channel equalization".

Since the system is LTI, its inverse must also be LTI. In fact, deconvolution would appear to be a simple matter: simply convolve $r\,[n]$ by the reciprocal filter, $h_r\,[n]$, whose Fourier transform satisfies

$$\hat{h}_r\,(\omega) = \frac{1}{\hat{h}_s\,(\omega)} \tag{5}$$

However, this apparent solution has the following major difficulties:

1. $\hat{h}_r\,(\omega)$ may not be a realizable filter. In most practical applications, $h_s\,[n]$ does not have a finite order Z-transform, since it arises in connection with a physical system, as opposed to a digital filter implementation. Consequently, $h_r\,[n]$ may not have finite numbers of poles and zeros. Even worse, if $\hat{h}_s\,(\omega)$ does not have minimum phase, no causally stable inverse filter exists, whether realizable or otherwise.

2. $\hat{h}_s\,(\omega)$ may well be close to zero at some frequencies, ω. At these frequencies, the reciprocal filter, $\hat{h}_r\,(\omega)$, must be very large, which will dramatically amplify the effects of noise or numerical round-off errors.

To overcome the first difficulty mentioned above, one might use one of the filter design techniques considered previously, taking $1/\hat{h}_s\,(\omega)$ to be our desired frequency response, $\hat{h}_d\,(\omega)$, and designing a realizable filter which approximates $\hat{h}_d\,(\omega)$ as accurately as possible in some suitable sense. Unfortunately, this approach will not directly deal with the common problem of noise amplification – item (2) above.

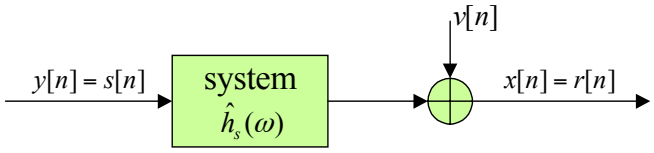

Figure 2: *LTI system with additive noise.*

To develop a Wiener deconvolution filter which is sensitive to both of the concerns raised above, we will associate $Y[n]$ with the source random process, $S[n]$, and $X[n]$ with the received signal, which we model as

$$X[n] = (h_s \star Y)[n] + V[n] \tag{6}$$

where $V[n]$ models the effect of additive random noise, as shown in Figure 2. The role of the Wiener filter is then to estimate $y[n] = s[n]$, as accurately as possible, from the noisy samples produced by the system in question. From equation 6, we may find the cross- and auto-correlation sequences as

$$\begin{aligned} R_{YX}[m] &= E\left[Y[n]\left(V[n-m] + \sum_k h_s[k]\, Y[n-m-k]\right)\right] \\ &= R_{YV}[m] + \sum_k h_s[k]\, R_{YY}[m+k] \\ &= R_{YV}[m] + \left(\tilde{h}_s \star R_{YY}\right)[m] \end{aligned}$$

and

$$\begin{aligned} R_{XX}[m] &= E\left[\left(V[n] + \sum_k h_s[k]\, Y[n-k]\right)\left(V[n-m] + \sum_k h_s[k]\, Y[n-m-k]\right)\right] \\ &= R_{VV}[m] + \sum_k h_s[k] \sum_j h_s[j]\, R_{YY}[m+k-j] + \sum_k h_s[k]\, R_{YV}[m-k] + \sum_j h_s[j]\, R_{VY}[m+k] \\ &= R_{VV}[m] + \left(h_s \star \tilde{h}_s \star R_{YY}\right)[m] + (h_s \star R_{YV})[m] + (h_s \star R_{YV})[-m] \end{aligned}$$

In most cases of practical interest, the noise is white and uncorrelated with the signal we are trying to estimate ($Y[n] = S[n]$ in this case), so $R_{YV}[m] = 0$ for all m and $R_{VV}[m] = \sigma_V^2 \delta[m]$. We then get

$$R_{YX}[m] = \left(\tilde{h}_s \star R_{YY}\right)[m] \tag{7}$$

$$R_{XX}[m] = \sigma_V^2 \delta[m] + \left(h_s \star \tilde{h}_s \star R_{YY}\right)[m] \tag{8}$$

3.1.1 Intuition from the Unconstrained Solution

It is instructive to begin by considering the unconstrained solution. In the Fourier domain, equations (7) and (8) become

$$\begin{aligned} S_{YX}(\omega) &= \hat{h}_s^*(\omega) S_{YY}(\omega) \\ S_{XX}(\omega) &= \sigma_V^2 + \left|\hat{h}_s(\omega)\right|^2 S_{YY}(\omega) \end{aligned}$$

and the unconstrained Wiener filter is given by

$$\hat{h}_\infty(\omega) = \frac{S_{YX}(\omega)}{S_{XX}(\omega)} = \frac{\hat{h}_s^*(\omega)}{\left|\hat{h}_s(\omega)\right|^2 + \sigma_V^2 / S_{YY}(\omega)} \tag{9}$$

To see how the possibility of noise is being factored into this solution, suppose we set $\sigma_V^2 = 0$. Then we get

$$\hat{h}_\infty(\omega) = \frac{\hat{h}_s^*(\omega)}{\left|\hat{h}_s(\omega)\right|^2} = \frac{1}{\hat{h}_s(\omega)}$$

which is the reciprocal filter of equation (5). Evidently, $\sigma_V^2 / S_{YY}(\omega)$ places a lower bound on the denominator in equation (9). At frequencies where $\left|\hat{h}_s(\omega)\right|^2 \gg \sigma_V^2 / S_{YY}(\omega)$, $\hat{h}_\infty(\omega)$ closely follows the reciprocal filter, while at frequencies where $\left|\hat{h}_s(\omega)\right|^2 \ll \sigma_V^2 / S_{YY}(\omega)$, the unconstrained Wiener filter has a response close to 0. The way to understand this is that wherever $\hat{h}_s(\omega)$ is very small, we do better to concentrate on filtering out the noise, rather than trying to recover the heavily attenuated signal.

It is important to realize that the Wiener deconvolution filter depends not just on the system impulse response, $\hat{h}_s(\omega)$, whose effects we are trying to undo, but also on the power spectrum of the uncorrupted signal, $Y[n] = S[n]$. Since we do not receive this signal, we may have some difficulty reliably estimating its statistics. Similarly, the noise statistics must also be estimated.

3.2 Decision Feedback Equalization

In a number of important applications, we get to observe the outcomes of the process, $Y[n] = S[n]$, which we are trying to estimate (after some delay, of course). One such application is that of adaptive channel equalization in a digital communications system. The context of this application has already been introduced in Example 3.

If the channel is approximately equalized and the demodulator is working well, we expect to recover the sequence of information bearing scaling factors, A_k, approximately. In a digital communication system, these analog quantities are thresholded to recover a discrete set of transmitted symbols. For example,

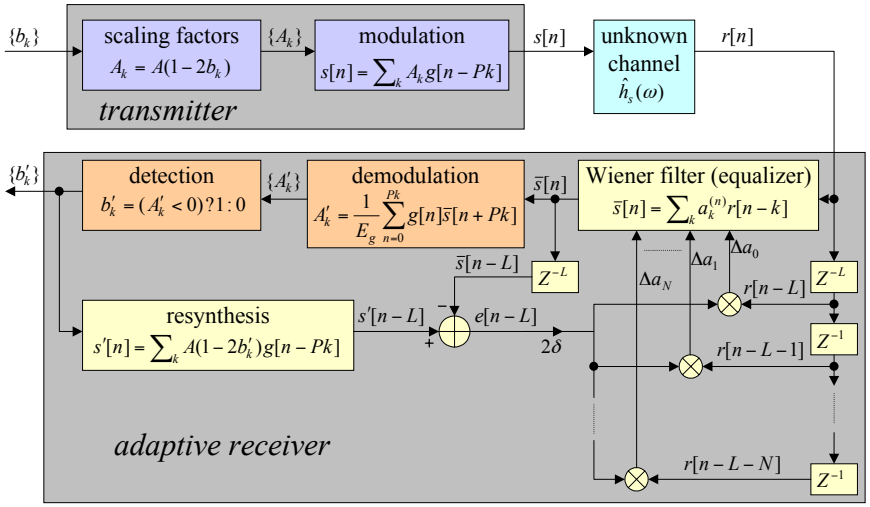

Figure 3: *Decision feedback equalizer using the LMS algorithm to adapt the Wiener filter coefficients.*

in binary ASK (Amplitude Shift Keying), the encoder uses only two amplitude values,

$$A_k = \begin{cases} A & \text{if } b_k = 0 \\ -A & \text{if } b_k = 1 \end{cases}$$

where b_k is a single binary digit, which is being communicated using the modulation scheme. In this case, the demodulator assumes that $b_k = 0$ was transmitted if it recovers $A'_k > 0$ using equation (4), and it decodes $b_k = 1$ if $A'_k \leq 0$. Even if our channel equalizer is not working all that well and even if there is a decent amount of noise, we expect to decode the binary digits correctly with high probability. Error probabilities of 1 in 10^3 are considered high in most communication environments.

Since we expect to recover the originally transmitted binary digits with high reliability, we can use this information to reconstruct the originally transmitted waveform, $s\,[n]$ (after some delay), from which we can recover the cross- and auto-correlation statistics and adapt our Wiener deconvolution filter (the channel equalizing filter). Such a system is known as a "decision feedback equalizer", since it relies upon the accuracy of the "thresholding decisions" which recover the transmitted bits, b_k, from the corrupted sequence of amplitude scaling factors, A_k.

Figure 3 shows a block diagram of a decision feedback equalizer which utilizes the LMS algorithm to adapt the Wiener filter coefficients. Notice that the demodulation operation in equation (4) requires us to look ahead L samples into the future, since the shaping pulse $g\,[n]$ has support $0 \leq n \leq L$. This means that there is a delay of at least L samples from the point at which the

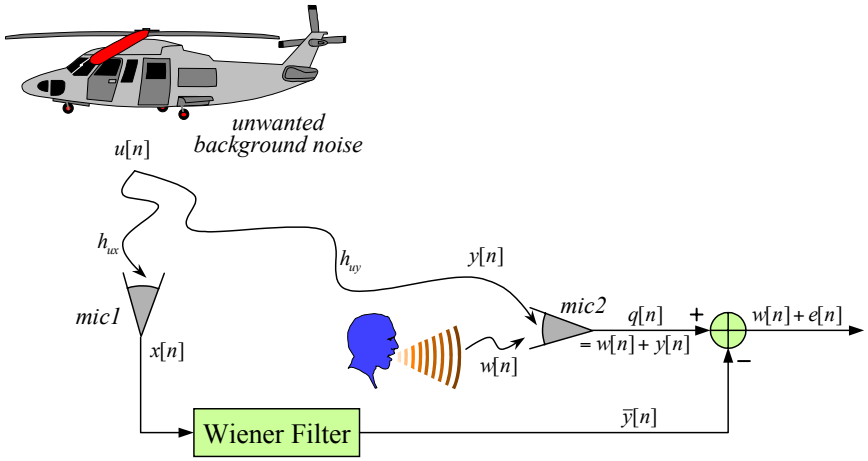

Figure 4: *Active background noise cancellation system.*

Wiener filter provides its estimate, $\bar{s}\,[n]$, of $s\,[n]$, to the point at which we are able to recover an estimate for $s\,[n]$ to use in driving the LMS algorithm. For this reason, the estimated gradient, $\nabla J^{(n)}$, which we use to update the Wiener filter coefficients from time step n to time step $n+1$, must be based on the outcome $s\,[n-L]$ and hence on the prediction error, $e\,[n-L]$. The gradient is thus estimated as

$$\nabla J^{(n)} = -2 \begin{pmatrix} e\,[n-L]\,x\,[n-L] \\ e\,[n-L]\,x\,[n-L-1] \\ \vdots \\ e\,[n-L]\,x\,[n-L-N] \end{pmatrix}$$

This explains the notation used in the figure.

3.3 Active "Noise" Cancellation

Active noise cancellation is a classic application for adaptive Wiener filtering. Figure 4 depicts a typical noise cancellation environment.The object is to remove unwanted background noise, $y\,[n]$, from the signal, $q\,[n] = y\,[n] + w\,[n]$ received at the output of a microphone ("mic2" in the figure). Figure 4 suggests a specific example in which the unwanted ambient noise comes from a helicopter's rotors, and the desired signal, $w\,[n]$, is the voice of the helicopter pilot speaking into the microphone.

In order to cancel out the ambient noise, we introduce a second microphone ("mic1" in the figure), usually positioned much closer to the source of the noise we wish to cancel than the speaker. Writing $u\,[n]$ for the noise signal at its

source, the noise component received at "mic1" is

$$x[n] = (h_{ux} \star u)[n]$$

while the noise component received at "mic2" is

$$y[n] = (h_{uy} \star u)[n]$$

where $\hat{h}_{ux}(\omega)$ and $\hat{h}_{uy}(\omega)$ are the LTI transfer functions describing the transmission path from the noise source to each of the two microphones. It follows that $y[n]$ and $x[n]$ should be connected through

$$\hat{y}(\omega) = \hat{x}(\omega) \frac{\hat{h}_{uy}(\omega)}{\hat{h}_{ux}(\omega)}$$

so we may interpret the role of the Wiener filter in Figure 4 as that of approximating the transfer function, $\hat{h}_{uy}(\omega)/\hat{h}_{ux}(\omega)$.

In practice, this transfer function is usually a slowly varying function, dependent upon the positions of the microphones and any interfering objects, such as the speaker's head. More significantly, there will invariably be additional sources of noise which ensure that $y[n]$ cannot be expressed as a deterministic function of $x[n]$.For these reasons, we must implement an adaptive Wiener filter, whose role is to provide the best estimate, $\bar{y}[n]$, of $y[n]$, using the cross- and auto-covariance statistics.

At first sight, it may appear that we have a problem. Since we do not have direct access to $y[n]$, it does not appear that we will be able to compute the error signal, $e[n] = y[n] - \bar{y}[n]$ needed to drive the LMS algorithm. More generally, it does not appear that we will be able to estimate the cross-correlation sequence, $R_{YX}[m]$, based on observations of the outputs from the two microphones. However, we do have access to $q[n]$ and

$$\begin{aligned} R_{QX}[m] &= R_{YX}[m] + R_{WX}[m] \\ &\approx R_{YX}[m] \end{aligned}$$

The last line follows from the fact that "mic1" is well separated from the speaker, so that $X[n]$ and $W[n]$ are largely uncorrelated. Proceeding on the assumption that $X[n]$ and $W[n]$ are uncorrelated, we may drive the LMS algorithm with $q[n] - \bar{y}[n]$ in place of $y[n] - \bar{y}[n]$. In particular, the gradient approximation is

$$\begin{aligned} \nabla J^{(n)} &= -2 \begin{pmatrix} (q[n] - \bar{y}[n]) x[n-0] \\ (q[n] - \bar{y}[n]) x[n-1] \\ \vdots \\ (q[n] - \bar{y}[n]) x[n-N] \end{pmatrix} \\ &\approx -2 \begin{pmatrix} R_{EX}[0] + R_{WX}[0] \\ R_{EX}[1] + R_{WX}[1] \\ \vdots \\ R_{EX}[N] + R_{WX}[N] \end{pmatrix} = -2 \begin{pmatrix} R_{EX}[0] \\ R_{EX}[1] \\ \vdots \\ R_{EX}[N] \end{pmatrix} \end{aligned}$$

4 Linear Prediction Revisited

In this section, we return to the linear prediction problem, in which $x[n]$ is to be predicted from the past N outcomes, $x[n-1]$ through $x[n-N]$. Our purpose is to show that linear prediction has a particularly elegant structure when realized as a lattice filter, and that the reflection coefficients of the lattice filter may be found directly from the auto-correlation function, $R_{XX}[m]$, without needing to solve the normal equations explicitly through matrix inversion. We shall see that each successive output from the upper branch of the lattice predictor is the prediction error from the optimal linear predictor of the corresponding order, and we shall also see that each successive output from the lower branch of the lattice predictor may be interpreted as a "backward" prediction error. In Section 5, we will show that the backward prediction errors are uncorrelated with each other and that they may provide an entirely different strategy for implementing and adapting the coefficients of Wiener filters.

The material presented here involves some less obvious mathematical tricks and you are not expected to be able to reproduce these in an exam. However, you should pay special attention to the results of the analysis, which are summarized in the preceding paragraph.

4.1 Preliminaries

Recall that the N^{th} order linear predictor for $x[n]$ is written

$$\bar{x}[n] = \sum_{k=1}^{N} a_k x[n-k]$$

and that the coefficients which minimize the power of the prediction error, $e[n] = x[n] - \bar{x}[n]$, are those which satisfy the normal equations,

$$\underbrace{\begin{pmatrix} R_{XX}[0] & R_{XX}[1] & \cdots & R_{XX}[N-1] \\ R_{XX}[1] & R_{XX}[0] & \cdots & R_{XX}[N-2] \\ \vdots & \vdots & \ddots & \vdots \\ R_{XX}[N-1] & R_{XX}[N-2] & \cdots & R_{XX}[0] \end{pmatrix}}_{R_{0:N-1}} \begin{pmatrix} a_1 \\ a_2 \\ \vdots \\ a_N \end{pmatrix} = \begin{pmatrix} R_{XX}[1] \\ R_{XX}[2] \\ \vdots \\ R_{XX}[N] \end{pmatrix}$$

Recall also that the LP analysis filter, whose output is the prediction error, $e[n]$, has coefficients,

$$h[n] = \begin{cases} 1 & n = 0 \\ -a_n & 0 < n \leq N \end{cases}$$

This is an N^{th} order FIR filter.

As suggested above, the lattice filter will turn out to have the property that each lattice stage outputs prediction error sequences corresponding to linear predictors of successively higher orders. For this reason, we will modify our notation to keep track of the order, N, of the analysis filter, $h_N[n]$, and of the

prediction error sequence, $e_N[n]$. We will also find it convenient to reorganize the normal equations as follows. First, move the vector on the right hand side into the matrix, to get

$$\begin{pmatrix} R_{XX}[1] & R_{XX}[0] & R_{XX}[1] & \cdots & R_{XX}[N-1] \\ R_{XX}[2] & R_{XX}[1] & R_{XX}[0] & \cdots & R_{XX}[N-2] \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ R_{XX}[N] & R_{XX}[N-1] & R_{XX}[N-2] & \cdots & R_{XX}[0] \end{pmatrix} \begin{pmatrix} h_N[0]=1 \\ h_N[1]=-a_1 \\ \vdots \\ h_N[N]=-a_N \end{pmatrix} = 0$$

and then augment the number of equations to get

$$\underbrace{\begin{pmatrix} R_{XX}[0] & R_{XX}[1] & R_{XX}[2] & \cdots & R_{XX}[N] \\ R_{XX}[1] & R_{XX}[0] & R_{XX}[1] & \cdots & R_{XX}[N-1] \\ R_{XX}[2] & R_{XX}[1] & R_{XX}[0] & \cdots & R_{XX}[N-2] \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ R_{XX}[N] & R_{XX}[N-1] & R_{XX}[N-2] & \cdots & R_{XX}[0] \end{pmatrix}}_{R_{0:N}} \begin{pmatrix} h_N[0] \\ h_N[1] \\ h_N[2] \\ \vdots \\ h_N[N] \end{pmatrix} = \begin{pmatrix} P_N \\ 0 \\ 0 \\ \vdots \\ 0 \end{pmatrix} \tag{10}$$

This new set of $N+1$ equations, involves the $(N+1)\times(N+1)$ Toeplitz autocorrelation matrix. In order to add the extra row at the top, we have had to introduce an extra quantity, P_N, on the right hand side. Although there are really only N unknowns, $h_N[1]$ through $h_N[N]$, any filter which satisfies these equations for some P_N, having $h_N[0]$, is the N^{th} order LP analysis filter.

Although the quantity, P_N, has been introduced principally for the structure of the augmented set of normal equations given above, it turns out that this quantity is exactly the power of the N^{th} order prediction error, which is why we have used the notation, P (for "power"). To see that P_N does indeed have this interpretation, observe that

$$E\left[E_N^2[n]\right] = E\left[E_N[n]\left(X[n]+\sum_{k=1}^{N} h_N[k]\,X[n-k]\right)\right] = E\left[E[n]\,X[n]\right]$$

where we have used the orthogonality principle, according to which $E[n]$ is orthogonal to (uncorrelated with) $X[n-1]$ through $X[n-N]$. Expanding this expression, we get

$$\begin{aligned} E\left[E_N^2[n]\right] &= E\left[X[n]\left(X[n]+\sum_{k=1}^{N} h_N[k]\,X[n-k]\right)\right] \\ &= R_{XX}[0]+\sum_{k=1}^{N} h_N[k]\,R_{XX}[k] = P_N \end{aligned}$$

4.2 Backward Prediction

The next step in our development of the lattice predictor is to consider backward prediction. The N^{th} order backward predictor forms an estimate, $\bar{X}_N[n-N]$,

for $X[n-N]$, based on the N future values, $X[n]$ through $X[n-N+1]$. Now the statistics of a stationary random process do not change if we reverse the time axis – one way to see this is that the auto-correlation function, $R_{XX}[m]$, is symmetric in m. It follows that the optimal backward predictor has the same coefficients as the forward predictor, and may be expressed as

$$\bar{x}_N[n-N] = \sum_{k=1}^{N} a_k x[n-N+k]$$

Write $b_N[n] = x[n-N] - \bar{x}_N[n-N]$ for the backward prediction error, which may then be expressed as

$$\begin{aligned} b_N[n] &= \sum_{k=0}^{N} h_N[k]\, x[n-N+k] \\ &= \sum_{j=N-k}^{N} h_N[N-j]\, x[n-j] \end{aligned}$$

Equivalently, $b_N[n]$ may be viewed as the output of an N^{th} order causal FIR filter, $g_N[n]$, having coefficients

$$g_N[n] = h_N[N-n]$$

We conclude that the backward LP analysis filter, $g_N[n]$ is the mirror image of $h_N[n]$. Note that the output of the backward LP analysis filter, $b_N[n]$, is the backward prediction error at time $n-N$.

We can write down a set of augmented normal equations for backward prediction, by substituting $g_N[n] = h_N[N-n]$ into equation (10) and reversing the order of all rows and columns in the various matrices and vectors[1]. The Toeplitz property of the auto-correlation matrix ensures that reversing the order of all rows and of all columns leaves it unchanged so that we get

$$\underbrace{\begin{pmatrix} R_{XX}[0] & R_{XX}[1] & R_{XX}[2] & \cdots & R_{XX}[N] \\ R_{XX}[1] & R_{XX}[0] & R_{XX}[1] & \cdots & R_{XX}[N-1] \\ R_{XX}[2] & R_{XX}[1] & R_{XX}[0] & \cdots & R_{XX}[N-2] \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ R_{XX}[N] & R_{XX}[N-1] & R_{XX}[N-2] & \cdots & R_{XX}[0] \end{pmatrix}}_{R_{0:N}} \begin{pmatrix} g_N[0] \\ g_N[1] \\ g_N[2] \\ \vdots \\ g_N[N] \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ \vdots \\ P_N \end{pmatrix} \tag{11}$$

4.3 The Levinson-Durbin Recursion

We now show that the forward and backward prediction error filters at order N may be used to derive the prediction error filter at order $N+1$ in a very simple

[1] This just reverses the order in which we are writing down the equations, but nothing else.

way. To do this, we first augment the forward and backward augmented normal equations one more time to increase the order to $N+1$. We do this by adding a row and a column (shown in bold) to the end of the auto-correlation matrix in equation (10), and introducing a new unknown, Δ_N, to accommodate the new equation represented by the last row.

$$\underbrace{\begin{pmatrix} R_{XX}[0] & R_{XX}[1] & \cdots & R_{XX}[N] & \mathbf{R}_{XX}[N+1] \\ R_{XX}[1] & R_{XX}[0] & \cdots & R_{XX}[N-1] & \mathbf{R}_{XX}[N] \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ R_{XX}[N] & R_{XX}[N-1] & \cdots & R_{XX}[0] & \mathbf{R}_{XX}[1] \\ \mathbf{R}_{XX}[N+1] & \mathbf{R}_{XX}[N] & \cdots & \mathbf{R}_{XX}[1] & \mathbf{R}_{XX}[0] \end{pmatrix}}_{R_{0:N+1}} \begin{pmatrix} h_N[0] \\ h_N[1] \\ \vdots \\ h_N[N] \\ 0 \end{pmatrix} = \begin{pmatrix} P_N \\ 0 \\ \vdots \\ 0 \\ \Delta_N \end{pmatrix} \tag{12}$$

For the backward normal equations, we add a new row and new column (shown in bold) to the beginning of the auto-correlation matrix in equation (11), which yields

$$\underbrace{\begin{pmatrix} \mathbf{R}_{XX}[0] & \mathbf{R}_{XX}[1] & \mathbf{R}_{XX}[2] & \cdots & \mathbf{R}_{XX}[N+1] \\ \mathbf{R}_{XX}[1] & R_{XX}[0] & R_{XX}[1] & \cdots & R_{XX}[N] \\ \mathbf{R}_{XX}[2] & R_{XX}[1] & R_{XX}[0] & \cdots & R_{XX}[N-1] \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \mathbf{R}_{XX}[N+1] & R_{XX}[N] & R_{XX}[N-1] & \cdots & R_{XX}[0] \end{pmatrix}}_{R_{0:N+1}} \begin{pmatrix} 0 \\ g_N[0] \\ g_N[1] \\ \vdots \\ g_N[N] \end{pmatrix} = \begin{pmatrix} \Delta'_N \\ 0 \\ 0 \\ \vdots \\ P_N \end{pmatrix} \tag{13}$$

It is easy to see that the unknowns, Δ_N and Δ'_N must be identical, since we have

$$\begin{aligned} \Delta'_N &= \sum_{n=0}^{N} g_N[n] R_{XX}[n+1] \\ &= {}_{k=N-n}\sum_{k=0}^{N} h_N[k] R_{XX}[N+1-k] = \Delta_N \end{aligned} \tag{14}$$

Finally, we form a new set of equations by adding equation (12) to equation (13) multiplied by k_1, which yields

$$R_{0:N+1} \times \begin{pmatrix} h_N[0] + k_N \cdot 0 \\ h_N[1] + k_N \cdot g_N[0] \\ \vdots \\ h_N[N] + k_N \cdot g_N[1] \\ 0 + k_N \cdot g_N[0] \end{pmatrix} = \begin{pmatrix} P_N + k_N \Delta_N \\ 0 \\ \vdots \\ 0 \\ \Delta_N + k_N P_N \end{pmatrix}$$

Let $h_{N+1}[n]$ be the $N+1^{\text{st}}$ order filter whose coefficients are given by

$$h_{N+1}[n] = h_N[n] + k_N g_N[n-1] \tag{15}$$

and set $k_N = -\frac{\Delta_N}{P_N}$, so that $\Delta_N + k_N P_N = 0$ and write

$$P_{N+1} = P_N + k_N \Delta_N = P_N \left(1 - k_N^2\right) \tag{16}$$

Then the $h_{N+1}[0] = 1$ and the new filter satisfies

$$R_{0:N+1} \times \begin{pmatrix} h_{N+1}[0] \\ h_{N+1}[1] \\ \vdots \\ h_{N+1}[N+1] \end{pmatrix} = \begin{pmatrix} P_{N+1} \\ 0 \\ \vdots \\ 0 \end{pmatrix}$$

which are the augmented normal equations for the $N+1^{\text{st}}$ order LP analysis filter.

Equation (15) gives us a recursive procedure for finding the linear prediction coefficients of any order. The recursive procedure may be described by the following algorithm, which is known as the "Levinson-Durbin" algorithm.

- Initialize $h_0[0] = 1$, $g_0[0] = 1$ and $P_0 = R_{XX}[0]$
- For $N = 0, 1, 2, \ldots$
  - Evaluate Δ_N using equation (14)
  - Evaluate $k_N = -\frac{\Delta_N}{P_N}$
  - Find the next order analysis filter coefficients from

$$\begin{aligned} h_{N+1}[n] &= h_N[n] + k_N g_N[n-1] \\ &= h_N[n] + k_N h_N[N+1-n] \end{aligned}$$

    using $h_N[n] = 0$ for $n > N$.
  - Find the power of the $N+1^{\text{st}}$ order prediction error from $P_{N+1} = \left(1 - k_N^2\right) P_N$

An important property of the Levinson-Durbin recursion is that the order $N+1$ coefficients may be derived from the order N coefficients using on the order of only $2N$ multiplications. This means that the complete cost of the recursive procedure for deriving the N^{th} order coefficients is roughly N^2 multiplications. By contrast, the cost of directly inverting the N^{th} order auto-correlation matrix is on the order of N^3 multiplications.

4.4 Lattice Structure for Linear Prediction

You should recall from our earlier discussion of FIR lattice filters that the two outputs from the N^{th} stage of an arbitrary lattice filter also correspond to mirror image filters, whose impulse responses we denoted $h_N[n]$ and $g_N[n]$. Moreover, we the lattice was characterized by exactly the update relationship as that found above as the Levinson-Durbin recursion. Specifically,

$$H_{N+1}(z) = H_N(z) + k_N z^{-1} G_N(z)$$

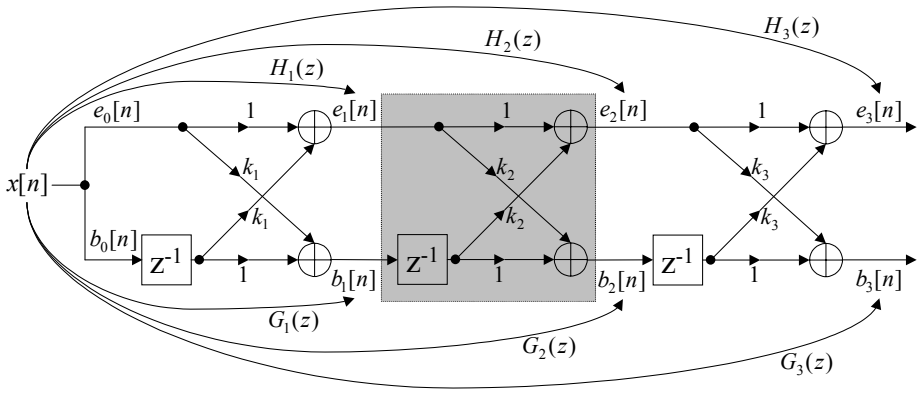

Figure 5: *Lattice linear predictor, showing the forward and backward prediction errors appearing at the outputs of successive stages.*

where we used the term "reflection coefficients" to refer to the k_N values. The lattice filter structure is repeated here in Figure 5 for your convenience. Note that the outputs from the N^{th} lattice stage are the forward and backward prediction errors, respectively.

According to equation (16), the prediction error power at the output of the N^{th} stage is related to that at the output of the previous stage according to

$$P_N = P_{N-1}\left(1 - k_N^2\right)$$

from which we may conclude that the reflection coefficients must all have magnitude strictly less than 1, unless the source process is perfectly predictable (never happens in practice, for obvious reasons). Moreover, by working the Levinson-Durbin algorithm in reverse, it is possible to select an auto-correlation sequence, $R_{XX}[m]$, whose lattice predictor will have any given set of reflection coefficients, with magnitude less than 1. It follows that the first N reflection coefficients, together with the variance of the source, $R_{XX}[0]$, provide an alternate description of $R_{XX}[m]$ in the range $0 \leq m \leq N$.

As noted in our discussion of lattice filter, the FIR lattice filter can be shown to have minimum phase if and only if the reflection coefficients all have magnitude $|k_N| < 1$, from which we may deduce that the LP analysis filters, $H_N(z)$, always have minimum phase and are hence are causally invertible. The corresponding LP synthesis filters may be implemented by the all-pole lattice structure studied previously.

5 Wiener Filtering by Linear Prediction

In this section, we show that an N^{th} order Wiener filter may be implemented by applying a simple set of weights to the backward prediction error sequences

produced by the source sequence's lattice predictor. In this way, the problem of predicting outcomes of one random process, $Y[n]$, based on inputs from another random process, $X[n]$, is largely solved once we are able to form a linear prediction of $X[n]$ from its own past samples.

The first key observation is that the backward prediction error samples produced along the lower branch of the lattice predictor are uncorrelated. To see this, recall that $b_N[n]$ is the backward prediction error at time $n-N$. From the orthogonality principle, then, the $B_N[n]$ must be orthogonal to (uncorrelated with) $X[n-N+1]$ through $X[n]$. But for each $k>0$,

$$B_{N-k}[n] = \sum_{p=0}^{N-k} g_{N-k}[p]\, X[n-p]$$

is a linear combination of the variables with which B_N is orthogonal. It follows that $B_{N-k}[n]$ must be orthogonal to (uncorrelated with) $B_N[n]$ for all k, so that the complete set of backward prediction errors, $B_m[n]$, are mutually orthogonal, for $0 \le m \le N$.

The second key observation is that any linear function of $x[n]$ through $x[n-N]$ may be expressed as a linear function of the backward prediction errors, $b_0[n]$ through $b_N[n]$. In particular, if

$$\bar{y}[n] = \alpha_0 x[n] + \alpha_1 x[n-1] + \cdots + \alpha_N x[n-N]$$

then $\bar{y}[n]$ can also be written as

$$\bar{y}[n] = \beta_0 b_0[n] + \beta_1 b_1[n] + \cdots + \beta_N b_N[n]$$

where the coefficients are related through

$$\begin{aligned}
\alpha_N &= \beta_N \\
\alpha_{N-1} &= \beta_{N-1} + g_N[1]\,\beta_N \\
&\vdots \\
\alpha_{N-k} &= \beta_{N-k} + \sum_{p=1}^{k} g_{N-k+p}[p]\,\beta_{N-k+p}
\end{aligned}$$

We conclude that an N^{th} order Wiener filter may be implemented in the manner depicted in Figure 6. The value of this structure is that the coefficients, β_k, are much simpler to find and to adaptively estimate than the original Wiener coefficients, α_k, precisely because the backward prediction errors are all mutually orthogonal. The Wiener minimization objective becomes

$$\begin{aligned}
J &= E\left[\left(Y[n] - \bar{Y}[n]\right)^2\right] \\
&= E\left[\left(Y[n] - \sum_{k=0}^{N} \beta_k B_k[n]\right)^2\right]
\end{aligned}$$

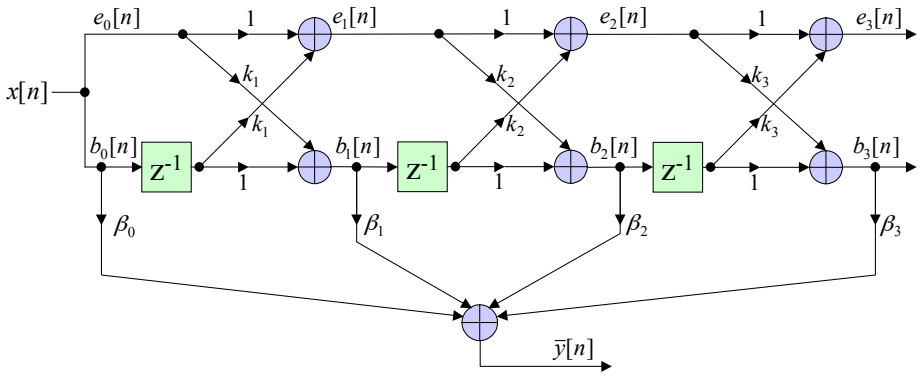

Figure 6: *Wiener predictor for $y[n]$, implemented as a set of independently adjustable weights, β_k, on the backward prediction errors from the lattice predictor for $x[n]$.*

whose solution satisfies (by differentiation or the orthogonality principle)

$$\begin{aligned} 0 &= E\left[B_p[n]\left(Y[n]-\sum_{k=0}^{N}\beta_k B_k[n]\right)\right] \\ &= E\left[B_p[n]\,Y[n]\right]-\sum_{k=0}^{N}\beta_k E\left[B_p[n]\,B_k[n]\right] \\ &= E\left[B_p[n]\,Y[n]\right]-\beta_p E\left[\left(B_p[n]\right)^2\right] \end{aligned}$$

from which we get

$$\beta_k=\frac{E\left[B_k[n]\,Y[n]\right]}{E\left[\left(B_p[n]\right)^2\right]}=\frac{E\left[B_k[n]\,Y[n]\right]}{P_k}$$

We already have the prediction error powers, P_k, from the Levinson-Durbin recursion, so the task of estimating β_k is equivalent to that of estimating the correlation between the target output, $Y[n]$, and the k^{th} backward prediction error, $B_k[n]$. In an adaptive setting, this can be done using a simple moving window correlator, without resorting to potentially unstable feedback estimators like the LMS algorithm.

We see, then, that Wiener prediction of $Y[n]$ from $X[n]$ through $X[n-N]$ (adaptive or otherwise) is essentially no harder and no easier than linear prediction of $X[n]$ from $X[n-1]$ through $X[n-N]$. For this reason, there has been considerable focus on adaptive algorithms for robustly and quickly estimating the reflection coefficients of a lattice predictor, based on observations of the sequence. We do not have time to review these algorithms here, but note that they are able to achieve faster convergence, with higher accuracy than the

LMS algorithm introduced previously. The interested reader is directed to the Gradient-Adaptive Lattice (GAL) algorithm as an excellent starting point.