10. Block Adaptive Filters

Three other classes of adaptive filters; transform-domain adaptive filters, block adaptive filters and subband adaptive filters. These exist due to concerns of the computational complexity (per input sample point) and convergence of LMS-type filters.

**Transform-domain filters** exploit the de-correlation properties of some well-known signal transforms, such as the discrete Fourier transform (DFT) and the discrete cosine transform (DCT), in order to pre-whiten the input data and speed up filter convergence. The resulting improvement in performance is usually a function of the data correlation and, therefore, the degree of success in achieving the desired objective varies from one signal correlation to another. The computational cost continues to be $O(M)$ operations per sample for a filter of length $M$.

**Block adaptive filters**, on the other hand, reduce the computational cost by a factor $\alpha > 1$, while at the same time improving the convergence speed. This is achieved by processing the data on a block-by-block basis, as opposed to a sample-by-sample basis, and by exploiting the fact that many signal transforms admit efficient implementations. However, the reduction in cost and the improvement in convergence speed come at a cost. Block implementations tend to suffer from a delay problem in the signal path, and this delay results from the need to collect blocks of data before processing.

The class of **subband adaptive filters** is related to the class of block adaptive filters, except that it attempts to achieve better pre-whitening (or band partitioning) of the data via selection of what are called prototype filters for their *analysis and synthesis filter banks* (*Multirate Signal Processing*). While subband filters also succeed in reducing the computational cost by a factor $\alpha > 1$, their convergence and mean-square performance can be less than that of block filters. This is because the design of the analysis and synthesis filter banks is usually decoupled from the adaptive filter design and, in this process, performance degradation can occur.

Most important sections (from preamble)

10.1.

See Chapters 26 to 28 of the on-line text book.
10.1 Transform-Domain Adaptive Filters

From chapter 9 the performance of LMS is sensitive to the correlation of the input sequence \{u(i)\} and, more specifically, to the eigenvalues and eigenvalue spread of the covariance matrix.

\[
R_{uu} = E[u_i^H \cdot u_j]
\]

where \(u_i\) is the regression vector,

\[
u_i = [u(i) \quad u(i-1) \ldots u(i-M+1)]
\]

In processing, the smaller eigenvalues result in slower convergence and the larger eigenvalues limit the step sizes and, thereby, limiting the learning curve of the filter.

The best convergence and learning curve would result when the eigenvalues are nearly equal. If the data could be “whitened” to be more like Gaussian or white noise, then the eigenvalue spread could be significantly reduced. Therefore we are interested in techniques that pre-whiten the data prior to adaptive processing.

One method to approximately achieve whitening is to filter the data prior to processing. The filter must be special; however, based on the auto-correlation of the input data. As you may expect, this requires a-priori knowledge of the system, so may be hard to achieve in practice unless it is sufficient to use a collected auto-correlation matrix to define filter coefficients.

A second method to approximately achieve whitening is to transform the regressor by a pre-selected unitary transformation. Candidates transformations are the discrete Fourier transform (DFT) or discrete sine or cosine transforms (DST or DCT).
10.1.1 Pre-Whitening Filters

Pre-whitening data based on known (estimated) second-order statistics.

Assume that \( \{u(i)\} \) is zero-mean and wide-sense stationary (WSS) with known auto-correlation

\[
r(k) = E[u(i) \cdot u(i-k)^H]
\]

for \( k = 0, \pm 1, \pm 2, \ldots \)

Spectral Factorization

The power-spectral density of a signal is loosely defined as the Fourier transform of the signal autocorrelation.

\[
S_{uu}(e^{jw}) = \sum_{k=-\infty}^{\infty} r(k) \cdot e^{-jwk}
\]

In the z-domain

\[
S_{uu}(z) = \sum_{k=-\infty}^{\infty} r(k) \cdot z^{-k}
\]

which exists for \( r(k) \) exponentially bounded (also referred to as finite energy)

\[
|r(k)| \leq \beta \cdot |e|^{|k|}
\]

for some \( \beta > 0 \) and \( 0 < \alpha < 1 \).

For z-transform aficionados, the series is absolutely convergent for an annulus in the z-domain where \( \alpha < |z| \leq \alpha^{-1} \), the region of convergence. Therefore, it satisfies

\[
\sum_{k=-\infty}^{\infty} |r(k)| \cdot |z^{-k}| < \infty \quad \text{for all } \alpha < |z| \leq \alpha^{-1}
\]

The power spectral density has two important properties:

1. Hermitian symmetry, i.e., \( S_{uu}(e^{jw}) = S_{uu}(e^{jw})^H \) and is therefore real.
2. Nonnegativity on the unit circle, i.e., \( S_{uu}(e^{jw}) \geq 0 \) for \( 0 \leq w \leq 2\pi \).

As an additional consideration for the z-domain is para-Hermitian symmetry

\[
S_{uu}(z) = S_{uu}\left(\frac{1}{z^*}\right)
\]

the zeros and poles are (i) conjugate symmetry and (ii) reciprocal symmetry with the unit circle.

Based on para-Hermitian symmetry, the power spectral density can be written as:

\[
S_{uu}(z) = \sigma_u^2 \cdot \frac{\prod_{l=1}^{m} (z - z_l) \cdot (z^{-1} - z_l^*)}{\prod_{l=1}^{n} (z - p_l) \cdot (z^{-1} - p_l^*)}
\]

where \( z_l \) are the zeros and \( p_l \) are the poles of the function.
From this structure we can construct two functions, one with all the zeros and poles inside the unit circle and a second function with all the zeros and poles outside the unit circle (think about the definition of stability).

\[ S_{un}(z) = \sigma_u^2 \cdot A(z) \cdot A\left( \frac{1}{z^*} \right) \]

where \( \{\sigma_u^2, A(z)\} \) satisfy the conditions

1. \( \sigma_u^2 \); is a positive scalar.
2. \( A(z) \) is normalized to unity at infinity, i.e., \( A(\infty) = 1 \).
3. \( A(z) \) is a rational minimum-phase function (i.e., its poles and zeros are inside the unit circle).

Under these conditions, \( A(z) \) is unique, otherwise, there would be an infinite number of choices for the function. The function \( A(z) \) that meets the normalization condition is

\[
A(z) = z^{n-m} \cdot \frac{\prod_{i=1}^{m} (z - z_i)}{\prod_{l=1}^{n} (z - p_l)} = \frac{\prod_{i=1}^{m} (1 - z_i \cdot z^{-1})}{\prod_{l=1}^{n} (1 - p_l \cdot z^{-1})}
\]

for the zeros and poles inside the unit circle.

Note that the numerator and denominator are causal and that \( A(\infty) = 1 \).
Filtering a WSS random process

\[ x(i) \xrightarrow{} H(z) \xrightarrow{} y(i) \]

\[ S_{yy}(z) = H(z) \cdot S_{xx}(z) \cdot H\left(\frac{1}{z^*}\right)^* \]

For scalar systems, you may be more used to

\[ S_{yy}(z) = |H(z)|^2 \cdot S_{xx}(z) \]

Now we are free to pick any arbitrary filter … such as

\[ H(z) = \frac{1}{\sigma_u \cdot A(z)} \]

applying this filter, we have

\[ S_{yy}(z) = \frac{1}{\sigma_u \cdot A(z)} \cdot S_{xx}(z) \cdot \frac{1}{\sigma_u \cdot A\left(\frac{1}{z^*}\right)^*} \]

but we have a known \( S_{xx}(z) \)

\[ S_{yy}(z) = \frac{1}{\sigma_u \cdot A(z)} \cdot \left[ \sigma_u^2 \cdot A(z) \cdot A\left(\frac{1}{z^*}\right) \right] \cdot \frac{1}{\sigma_u \cdot A\left(\frac{1}{z^*}\right)^*} \]

The expected result is then

\[ S_{yy}(z) = 1 \]

The filter signal \( \{\tilde{u}(\cdot)\} \) has been whitened with a known auto-correlation

\[ r(k) = E[\tilde{u}(i) \cdot \tilde{u}(i-k)^H] = \begin{cases} 1, & \text{for } k = 0 \\ 0, & \text{for } k = \pm 1, \pm 2, \ldots \end{cases} \]

We also have the whitened covariance matrix

\[ R_{uu} = E[\tilde{u}_i^H \cdot \tilde{u}_i] = I \]
Adaptive filter implementation

For the LMS algorithm

\[
\tilde{w}_i = \tilde{w}_{i-1} + \mu \cdot \tilde{u}_i \cdot (\tilde{d}(i) - \tilde{u}_i \cdot \tilde{w}_{i-1})
\]

Of note is the requirement to form a filtered version of \(d(i)\). An example of this algorithm and other to come is seen in Figure 10.5.
Dr. Bazuin’s Aside
If the input signals were based on a spatial vector, as is typical in a multiple antenna input process. The “spatial covariance matrix”

\[
R_{uu} = E[u_i \cdot u_i^H]
\]

an estimate can be defined as

\[
\hat{R}_{uu} = \frac{1}{i+1} \sum_{j=0}^{i} \lambda^{i-j} \cdot u_i^H \cdot u
\]

or for \( \lambda = 1 \)

\[
\hat{R}_{uu} = \frac{1}{i+1} \sum_{j=0}^{i} u_i^H \cdot u
\]

We can form the Cholesky factorization of this matrix

\[
\hat{R}_{uu} = R_u \cdot R_u^H
\]

Now if the “system” employs a pre-filter as

\[
\begin{align*}
&u_i \quad \rightarrow \quad R_u^{-1} \\
&\quad \rightarrow \quad \bar{u}_i
\end{align*}
\]

or

\[
\bar{u}_i = R_u^{-1} \cdot u_i
\]

Consider now the output of the vector-matrix multiplication

\[
\begin{align*}
R_{\pi\pi} &= E[\bar{u}_i \cdot \bar{u}_i^H] \\
R_{\pi\pi} &= E[R_u^{-1} \cdot u_i \cdot u_i^H \cdot R_u^{-H}] \\
R_{\pi\pi} &= R_u^{-1} \cdot E[u_i \cdot u_i^H] \cdot R_u^{-H} \\
R_{\pi\pi} &= R_u^{-1} \cdot R_u \cdot R_u^{-H} = I
\end{align*}
\]

Imagine this being applied for a blind-adaptive algorithm. The \( d(i) \) reference need not be generated, all focus is on the error function constant, gamma \( \gamma \).
A final consideration, the adaptive MIMO system would appear to require two matrix multiplications.

\[ u_i \quad R_u^{-1} \quad \bar{u}_i \quad W_i \quad x_i = W_i \cdot \bar{u}_i \]

but as the estimate of the covariance is being formed and the weight is being formed on a sample basis, the two computations can be performed as one. The structure would then implement

\[ u_i \quad R_u^{-1} \cdot W_i \quad x_i = \left[R_u^{-1} \cdot W_i\right] \cdot \bar{u}_i \]

**Interpretation of an optimal filter (Chap 2).**
Remember that we defined the optimal filter as

\[ K_{opt} = R_{yy}^{-1} \cdot R_{xy} \]

with the input y. If this were rewritten in the form of a Cholesky factorization of the aut0-covariance matrix

\[ K_{opt} = R_y^{-1} \cdot \left[R_y^{-H} \cdot R_{xy}\right] \]

One interpretation is that the optimal weights have an embedded pre-whitening multiplication for the environment combined with a “y normalized” weight element.
10.1.2 Unitary Transformations

The design of a pre-whitening filter can be difficult. So are there other transformations that can be applied.

Unitary transformations have the characteristics that
\[ T \cdot T^T = T^T \cdot T = I \]

The most recognized examples of such transformations are the discrete Fourier transform (DFT) and discrete cosine transform (DCT). For the DFT the elements of the T matrix are defined as
\[ [F]_{km} = \frac{1}{\sqrt{M}} \cdot \exp \left( -j \cdot 2 \cdot \pi \cdot \frac{m \cdot k}{M} \right) \]

For the DCT the elements of the T matrix are defined as
\[ [C]_{km} = \alpha(k) \cdot \frac{1}{\sqrt{M}} \cdot \cos \left( \frac{2 \cdot \pi \cdot \frac{m + \frac{1}{2}}{M} \cdot k}{2} \right) \]

where
\[ \alpha(k) = \begin{cases} 1, & \text{for } k = 0 \\ \sqrt{2}, & \text{for } k \neq 0 \end{cases} \]

In these equations, k denotes the row index and m the column index.

Aside: Other forms involve “odd-frequency” transformation.

The odd-frequency DFT
\[ [OF]_{km} = \frac{1}{\sqrt{M}} \cdot \exp \left( -j \cdot 2 \cdot \pi \cdot \frac{m \cdot k + 1}{M} \right) \]

The odd-frequency form of the DCT
\[ [OC]_{km} = \frac{\sqrt{2}}{\sqrt{M}} \cdot \cos \left( \frac{2 \cdot \pi \cdot \frac{m + \frac{1}{2}}{M} \cdot (k + \frac{1}{2})}{2} \right) \]

Applying the unitary matrices
\[ \bar{u}_i = u_i \cdot T \]
then the covariance becomes
\[ R_{\bar{u}\bar{u}} = E[\bar{u}_i^H \cdot \bar{u}_i] \]
\[ R_{uu} = E[T^H \cdot u_i^H \cdot u_i \cdot T] \]
\[ R_{\bar{u}u} = T^H \cdot E[u_i^H \cdot u_i] \cdot T \]
\[ R_{uu} = T^H \cdot R_{uu} \cdot T \]
Forming for the LMS algorithm
\[ w_i = w_{i-1} + \mu \cdot u_i^H \cdot (d(i) - u_i \cdot w_{i-1}) \]
\[ w_i = w_{i-1} + \mu \cdot u_i^H \cdot (d(i) - u_i \cdot T \cdot T^H \cdot w_{i-1}) \]

pre-multiply
\[ T^H \cdot w_i = T^H \cdot w_{i-1} + \mu \cdot T^H \cdot u_i^H \cdot (d(i) - u_i \cdot T \cdot T^H \cdot w_{i-1}) \]

Recognizing the transformed terms
\[ \bar{u}_i = u_i \cdot T \quad \text{and} \quad \bar{w}_i = T^H \cdot w_i \]

Results in
\[ \bar{w}_i = \bar{w}_{i-1} + \mu \cdot \bar{u}_i^H \cdot (d(i) - \bar{u}_i \cdot \bar{w}_{i-1}) \]

In block diagram terms

![Block diagram](image)

Notice that the reference sequence does not change and the error, \( e(i) \), does not change. Further, the weights are computed in the transform domain and never need to be computed for the non-transformed domain. Finally, if the systems outputs are to be \( e(i) \) (signal cancellation) or \( u_i \cdot w_{i-1} = \bar{u}_i \cdot \bar{w}_{i-1} \) (where signal equalization or enhancement is applied) nothing more needs to be done.

Intermediate comments:
While the transformed domain is different, the eigenvalues and eigenvalue spread of the similarity transformation performed remain the same. (HW problem 10.10).

Therefore, we may still need to provide some form of normalization.
Applying Normalization

First, perform a factorization of the covariance matrix

\[ R_{uu} = U \cdot D \cdot U^H \]

the Hermitian symmetric covariance matrix can be decomposed into an upper-triangular unitary matrix and a diagonal matrix as shown. Now, \( D \) is composed of the real-constants and can therefore be modified to

\[ R_{uu} = U \cdot D^{1/2} \cdot D^{H/2} \cdot U^H \]

The \( 1/2 \) represents that the matrix may be considered a Hermitian transpose, but it is actually a real matrix such that

\[ D^{1/2} = D^{H/2} \]

Now, if we normalize the LMS algorithm (not exactly the e-NLMS approach) as

\[ \bar{w}_i = \bar{w}_{i-1} + \mu \cdot D^{-1} \cdot \bar{u}_i^H \cdot (d(i) - \bar{u}_i \cdot \bar{w}_{i-1}) \]

and premultiply by the square root of the diagonal matrix

\[ D^{1/2} \cdot \bar{w}_i = D^{1/2} \cdot \bar{w}_{i-1} + \mu \cdot D^{H/2} \cdot D^{-1} \cdot \bar{u}_i^H \cdot (d(i) - \bar{u}_i \cdot \bar{w}_{i-1}) \]

Using another change in variable, define

\[ u_i' = \bar{u}_i \cdot D^{-1/2} = u_i \cdot T \cdot D^{-1/2} \quad \text{and} \quad w_i' = D^{1/2} \cdot \bar{w}_i = D^{1/2} \cdot T^H \cdot w_i \]

Manipulating the regressor-weight multiplication

\[ D^{1/2} \cdot \bar{w}_i = D^{1/2} \cdot \bar{w}_{i-1} + \mu \cdot D^{H/2} \cdot \bar{u}_i^H \cdot (d(i) - \bar{u}_i \cdot D^{-1/2} \cdot D^{1/2} \cdot \bar{w}_{i-1}) \]

and substituting

\[ w_i' = w_{i-1}' + \mu \cdot u_i'^H \cdot (d(i) - u_i'^T \cdot w_{i-1}') \]

The regressor covariance becomes

\[
\begin{align*}
R_{w'w'} &= E[u_i'^H \cdot u_i'^T] \\
R_{w'u'} &= E[D^{1/2} \cdot T^H \cdot u_i'^H \cdot u_i \cdot T \cdot D^{-1/2}] \\
R_{w'u'} &= D^{-1/2} \cdot T^H \cdot E[u_i'^H \cdot u_i] \cdot T \cdot D^{-1/2} \\
R_{w'u'} &= D^{-1/2} \cdot T^H \cdot R_{uu} \cdot T \cdot D^{-1/2}
\end{align*}
\]

If the transformation matrix \( T \) were selected as the unitary transformation matrix \( U \), such that \( T=U \), we would have

\[
\begin{align*}
R_{w'w'} &= D^{-1/2} \cdot [T^H \cdot U] \cdot D \cdot [U^H \cdot T] \cdot D^{-1/2} \\
R_{w'w'} &= D^{-1/2} \cdot U^H \cdot U \cdot D \cdot U^H \cdot U \cdot D^{-1/2} = D^{-1/2} \cdot D \cdot D^{-1/2} = I
\end{align*}
\]

The regressor is normalized and diagonally independent. For this choice of \( T=U \), the relationship is known as the Karhunen Loeve transformation (KLT).
The goal in using a transformed or normalized transform domain is to eliminate any correlation between the regressor elements with a resulting independence of the weight element calculations.

\[ R_{mm} = T^H \cdot R_{uu} \cdot T \approx \text{diag}([\lambda_0 \quad \lambda_1 \quad \cdots \quad \lambda_{M-1}]) \approx D \]

How well this is accomplished can be seen in the example shown on p. 580 of the textbook for a 4 x 4 auto-regressive process.

If near diagonalization is achieved, the elements of the transformed domain covariance matrix can be computed as

\[ \lambda_k(i) = \beta \cdot \lambda_k(i-1) + (1-\beta) \cdot ||u_i(k)||^2, \quad \text{for } k=0:M-1 \text{ and } 0<\beta<1 \]

with this estimate, the desired normalized diagonal matrix D can be computed.

With this the following implementation can be performed,

**Algorithm 10.1.1 General transform-domain LMS**

\[ \bar{u}_i = u_i \cdot T = [\bar{u}_i(0) \quad \bar{u}_i(1) \quad \cdots \quad \bar{u}_i(M-1)] \]

For each element k compute, k=0:M-1

\[ \lambda_k(i) = \beta \cdot \lambda_k(i-1) + (1-\beta) \cdot ||u_i(k)||^2, \quad \text{for } 0<\beta<1 \]

\[ D_i = \text{diag}([\lambda_0(i) \quad \lambda_i(i) \quad \cdots \quad \lambda_{M-1}(i)]) \]

\[ e(i) = d(i) - \bar{u}_i \cdot \bar{w}_{i-1} \]

\[ \bar{w}_i = \bar{w}_{i-1} + \mu \cdot D^{-1} \cdot \bar{u}_i^H \cdot e(i) \]

Initial conditions

\[ \lambda_k(-1) = \varepsilon \quad \text{and} \quad \bar{w}_0 = 0 \]
10.1.3 DFT-Domain LMS

Computations complexity versus performance may not appear to favor this routine at the moment, but in the future it could happen!

The DFT of the data adds an order $M \times \log_2(M)$ operations to the LMS and the $\text{inv}(D)$ adds more. Can the number of computations be reduced?

Minimizing operations required by the DFT.

$$[F]_{km} = \frac{1}{\sqrt{M}} \cdot \exp\left(-j \cdot 2 \cdot \pi \cdot \frac{m \cdot k}{M}\right)$$

The regressors are based on a taped delay line; therefore, $M-1$ of the elements are the same in successive processes.

$$\bar{u}_i(k) = \frac{1}{\sqrt{M}} \cdot \sum_{m=0}^{M-1} u(i-m) \cdot \exp\left(-j \cdot 2 \cdot \pi \cdot \frac{m \cdot k}{M}\right)$$

and

$$\bar{u}_{i-1}(k) = \frac{1}{\sqrt{M}} \cdot \sum_{m=0}^{M-1} u(i-1-m) \cdot \exp\left(-j \cdot 2 \cdot \pi \cdot \frac{m \cdot k}{M}\right)$$

Applying a change of variables $n = m + 1$

$$\bar{u}_{i-1}(k) = \frac{1}{\sqrt{M}} \cdot \sum_{n=1}^{M} u(i-n) \cdot \exp\left(-j \cdot 2 \cdot \pi \cdot \frac{(n-1) \cdot k}{M}\right)$$

Isolating terms, looking toward the $i^{th}$ regressor computation

$$\bar{u}_{i-1}(k) = \exp\left(j \cdot 2 \cdot \pi \cdot \frac{k}{M}\right) \cdot \frac{1}{\sqrt{M}} \cdot \sum_{n=1}^{M} u(i-n) \cdot \exp\left(-j \cdot 2 \cdot \pi \cdot \frac{n \cdot k}{M}\right)$$

$$\bar{u}_{i-1}(k) = e^{\frac{j2\pi k}{M}} \cdot \left[ \frac{1}{\sqrt{M}} \cdot \sum_{n=1}^{M-1} u(i-n) \cdot e^{-\frac{j2\pi n k}{M}} + \frac{1}{\sqrt{M}} \cdot u(i-M) \cdot e^{-\frac{j2\pi M k}{M}} \right]$$

The summation is composed of $M-1$ terms needed for computation of the $i^{th}$ regressor. Therefore, including the $i^{th}$ regressor (adding and subtracting)

$$\bar{u}_{i-1}(k) = e^{\frac{j2\pi k}{M}} \cdot \left\{ \left[\bar{u}_i(k) - \frac{1}{\sqrt{M}} \cdot u(i) \right] + \frac{1}{\sqrt{M}} \cdot u(i-M) \right\}$$
Rearranging the terms to allow computation of the $i^{th}$ regressor

$$e^{-j\frac{2\pi k}{M}} \cdot \bar{u}_{i-1}(k) = \bar{u}_i(k) - \frac{1}{\sqrt{M}} \cdot u(i) + \frac{1}{\sqrt{M}} \cdot u(i - M)$$

$$\bar{u}_i(k) = e^{-j\frac{2\pi k}{M}} \cdot \bar{u}_{i-1}(k) + \frac{1}{\sqrt{M}} \cdot [u(i) - u(i - M)]$$

This reduced the computations of the $M$ element regressor to order ($M$) operations!

Returning to a vector notation,

$$\bar{u}_i = \bar{u}_{i-1} \cdot S + \frac{1}{\sqrt{M}} \cdot [u(i) - u(i - M)] \cdot [1 \ 1 \ \cdots \ 1]$$

where $S$ is a diagonal matrix of the DFT coefficients

$$S = \text{diag} \left[ e^{-j\frac{2\pi 1}{M}} \ e^{-j\frac{2\pi 2}{M}} \ \cdots \ e^{-j\frac{2\pi (M-1)}{M}} \right]$$

See Algorithm 10.1.2 DFT-domain LMS on p. 582 for the implementation summary.

### 10.1.4 DCT-Domain LMS

Algorithm 10.1.3 DCT-domain LMS is summarized without derivation in this section.
10.2 to 10.4  **Block Adaptive Filters**

Processing on a block-by-block basis instead of sample-by-sample.

This section describes in a mixture of elements and notations the rudiments of polyphase filtering, where data is input by fixed size vectors, a matrix of data vectors is processed.

Processing is performed in the frequency domain where convolution is performed by multiplication. In general the description is of the “overlap-save” DFT implementation of frequency domain filtering.

The data is returned to the time domain where the output vectors of the filtering process are available.

This is a difficult way to describe some advanced concepts (frequency domain filtering) from digital signal processing. The key figure for recognizing this is Figure 10.10 and 10.13.

![Diagram of Block Adaptive Filters](image)

**FIGURE 27.5** Equivalent implementation of the mapping in Fig. 27.4 for block convolution in terms of the DFT matrix.
The processes here strictly define block based frequency domain filtering. The adaptive part comes with changing the filter elements L as shown in Figure 10.15.
10.5 Subband Adaptive Filters

From the structures of overlap-save DFT using polyphase elements, it is not a big leap to envision performing the filtering multiplication at a decimated rate and then interpolating back to the original sample rate. This approach uses an analysis filter and synthesis filter and is shown in Figure 10.23 (an open-loop structure) and 10.24 (a closed loop structure).

As before, if you need $e(i)$ use the closed loop form (next page). If you want the weighted regressor estimate of $d(i)$, either approach can be used.
FIGURE 28.8 Closed-loop structure for subband adaptive filtering.
**MATLAB Simulation**

(Acoustic echo cancellation) Acoustic echo cancellation is a common problem in hands-free telephony, where a person moves freely in a room while talking and listening to a remote speaker. With a loudspeaker and a microphone installed in the same room, as indicated by Fig. VI.3, undesired echoes over the walls interfere with the signal from the local speaker at the microphone.

The task of an acoustic echo canceller is to estimate the echo path, reconstruct the echoes, and subtract them from the microphone signal; thus leaving only the signal by the local speaker. Acoustic echo cancellation shares many commonalities with line echo cancellation, as studied in Computer Project IV.1. One main difference is that the length of the acoustic echo path tends to be considerably longer than the length of the line echo path. For this reason, block and subband adaptive filters are useful for applications involving acoustic echo cancellation.

![Diagram of adaptive acoustic echo cancellation](image)

**FIGURE VI.3** Adaptive acoustic echo reflections in a room containing a loudspeaker and a microphone.

Part B: Conventional e-NLMS Algorithm
Part C: DFT Block adaptive
Implement the block diagram shown in Figure 10.35

The implementation is based on Section 10.B and 10.C, the particular algorithm is Algorithm 10.C.4 on p. 637.

Configuration modification:
If a person moves freely in a room while talking and listening to a remote speaker, there is another audio input to the microphone, the speech of the speaker.

PartC_bjb_aug does this for a range of MATLAB test signals.