Block Adaptive Filters and Frequency Domain Adaptive Filters

Overview

• Block Adaptive Filters
  – Iterating LMS under the assumption of small variations in $w(n)$
  – Approximating the gradient by time averages
  – The structure of the Block adaptive filter
  – Convergence properties

• Frequency Domain Adaptive Filters
  – Frequency domain computation of linear convolution
  – Frequency domain computation of linear correlation
  – Fast LMS algorithm
  – Improvement of convergence rate
  – Unconstrained frequency domain adaptive filtering
  – Self-orthogonalizing adaptive filters

Reference: Chapter 7 from Haykin’s book *Adaptive Filter Theory* 2002
LMS algorithm

Given

\[
\begin{align*}
\begin{cases}
\quad & \text{the (correlated) input signal samples } \{u(1), u(2), u(3), \ldots\}, \\
\quad & \text{generated randomly;}
\end{cases} \\
\begin{cases}
\quad & \text{the desired signal samples } \{d(1), d(2), d(3), \ldots\} \text{ correlated} \\
\quad & \text{with } \{u(1), u(2), u(3), \ldots\}
\end{cases}
\end{align*}
\]

1. **Initialize the algorithm** with an arbitrary parameter vector \( \mathbf{w}(0) \), for example \( \mathbf{w}(0) = 0 \).
2. **Iterate for** \( n = 0, 1, 2, 3, \ldots, n_{\text{max}} \)
   2.0  Read /generate a new data pair, \( (u(n), d(n)) \)
   2.1  (Filter output) \( y(n) = \mathbf{w}(n) \mathbf{u}(n) = \sum_{i=0}^{M-1} w_i(n) u(n-i) \)
   2.2  (Output error) \( e(n) = d(n) - y(n) \)
   2.3  (Parameter adaptation) \( \mathbf{w}(n+1) = \mathbf{w}(n) + \mu u(n) e(n) \)

**Complexity of the algorithm:** \( 2M + 1 \) multiplications and \( 2M \) additions per iteration

The error signal \( e(n) \) is computed using the parameters \( \mathbf{w}(n) \), and we emphasize this by denoting \( e_{\mathbf{w}(n)}(n) \).
Iterating LMS under the assumption of small variations in $w(n)$

The new parameters in LMS are evaluated at each time step

$$w(n + L) = w(n + L - 1) + \mu u(n + L - 1)e_{w(n+L-1)}(n + L - 1)$$

$$= w(n + L - 2) + \mu u(n + L - 2)e_{w(n+L-2)}(n + L - 2) + \mu u(n + L - 1)e_{w(n+L-1)}(n + L - 1)$$

$$= w(n) + \sum_{i=0}^{L-1} \mu u(n + i)e_{w(n+i)}(n + i)$$

If the variations of parameters $w(n + L - i)$ during the $L$ steps of adaptation are small, $w(n + L - i) \approx w(n)$

$$w(n + L) \approx w(n) + \sum_{i=0}^{L-1} \mu u(n + i)e_{w(n)}(n + i)$$

Introduce a second time index $k$ such that $n = kL$ with a fixed integer $L$

$$w(kL + L) = w((k + 1)L) = w(kL) + \mu \sum_{i=0}^{L-1} u(n + i)e_{w(n)}(n + i)$$

If the parameters are changed only at moments $kL$, we may change the notation $w(k) \leftarrow w(kL)$

$$w(k + 1) = w(k) + \mu \sum_{i=0}^{L-1} u(kL + i)e_{w(k)}(kL + i)$$

The output of the filter is

$$y(kL + i) = w^T(k)u(kL + i) \quad i \in \{0, \ldots, L - 1\}$$
Block processing

Data used for modifying the parameters is grouped in blocks of length $L$.

The variables defined at time instants $n = kL + i$:

- the input signal $u(kL + i)$
- the output of the filter $y(kL + i) = w^T(k)u(kL + i)$
- the error signal $e(kL + i)$

The parameter vector, $w(k)$, is defined only at time instants $kL$. 

\[
\begin{array}{ccccccc}
0 & L & 2L & 3L & 4L & kL \\
0 & 1 & 2 & 3 & 4 & 5 & 6 & n \\
\end{array}
\]

\[
\begin{array}{ccccccc}
w(1) & w(2) & w(3) & w(4) & w(k) \\
\end{array}
\]

\[
u(2L+2)
\]
## Block LMS algorithm

Given

- the (correlated) input signal samples \( \{u(1), u(2), u(3), \ldots \} \), randomly generated;
- the desired signal samples \( \{d(1), d(2), d(3), \ldots \} \) correlated with \( \{u(1), u(2), u(3), \ldots \} \)

1. **Initialize the algorithm** with an arbitrary parameter vector \( w(0) \), for example \( w(0) = 0 \).
2. **Iterate** for \( k = 0, 1, 2, 3, \ldots, n_{\text{max}} \)
   
   2.0 **Initialize** \( \phi = 0 \)
   
   2.1 **Iterate** for \( i = 0, 1, 2, 3, \ldots, (L - 1) \)
      
      2.1.0 **Read** /generate a new data pair, \( \{u(kL + i), d(kL + i)\} \)
      
      2.1.1 **(Filter output)**
      \[
      y(kL + i) = w(k)^T u(kL + i) = \sum_{j=0}^{M-1} w_j(k) u(kL + i - j)
      \]
      
      2.1.2 **(Output error)**
      \[
      e(kL + i) = d(kL + i) - y(kL + i)
      \]
      
      2.1.3 **(Accumulate)**
      \[
      \phi \leftarrow \phi + \mu e(kL + i) u(kL + i)
      \]
      
   2.2 **(Parameter adaptation)**
   \[
   w(k + 1) = w(k) + \frac{\phi}{\bar{u}}
   \]

**Complexity of the algorithm**: \( 2M + 1 + \frac{1}{L} \) multiplications and \( 2M + \frac{1}{L} \) additions per iteration
Another way to introduce Block LMS algorithm: approximating the gradient by time averages

The criterion

\[ J = E e^2(n) = E (d(n) - w(n)^T u(n))^2 \]

has the gradient with respect to the parameter vector \( w(n) \)

\[ \nabla_{w(n)} J = -2e(n)u(n) \]

The adaptation of parameters in the Block LMS algorithm is

\[ w(k + 1) = w(k) + \mu \sum_{i=0}^{L-1} u(kL + i) e_{w(k)}(kL + i) \]

and denoting \( \mu_B = \mu L \), the adaptation can be rewritten

\[ w(k + 1) = w(k) + \mu_B \left[ \frac{1}{L} \sum_{i=0}^{L-1} u(kL + i) e_{w(k)}(kL + i) \right] = w(k) - \mu_B \frac{1}{2} \hat{\nabla}_{w(k)} J \]

where we denoted by

\[ \hat{\nabla}_{w(k)} J = -\frac{1}{L} \sum_{i=0}^{L-1} u(kL + i) e_{w(k)}(kL + i) \]

which shows that expectation in the expression of the gradient is replaced by time average.
Convergence properties of the Block LMS algorithm:

- Convergence of average parameter vector $Ew(n)$

We will subtract the vector $w_o$ from the adaptation equation

$$w(k + 1) = w(k) + \mu \frac{1}{L} \sum_{i=0}^{L-1} u(kL + i)e_{w(k)}(kL + i) = w(k) + \mu \frac{1}{L} \sum_{i=0}^{L-1} u(kL + i)(d(kL + i) - u(kL + i)^T w(k))$$

and we will denote $\varepsilon(k) = w(k) - w_o$

$$w(k + 1) - w_o = w(k) - w_o + \mu \frac{1}{L} \sum_{i=0}^{L-1} u(kL + i)(d(kL + i) - u(kL + i)^T w(k))$$

$$\varepsilon(k + 1) = \varepsilon(k) + \mu \frac{1}{L} \sum_{i=0}^{L-1} u(kL + i)(d(kL + i) - u(kL + i)^T w_o) +$$

$$+ \mu \frac{1}{L} \sum_{i=0}^{L-1} (u(kL + i)u(kL + i)^T w_o - u(kL + i)u(kL + i)^T w(k))$$

$$= \varepsilon(k) + \mu \frac{1}{L} \sum_{i=0}^{L-1} u(kL + i)e_o(kL + i) - \mu \frac{1}{L} \sum_{i=0}^{L-1} u(kL + i)u(kL + i)^T \varepsilon(k)$$

$$= (I - \mu \frac{1}{L} \sum_{i=0}^{L-1} u(kL + i)u(kL + i)^T)\varepsilon(k) + \mu \frac{1}{L} \sum_{i=0}^{L-1} u(nkL + i)e_o(kL + i)$$

Taking the expectation of $\varepsilon(k + 1)$ using the last equality we obtain

$$E\varepsilon(k + 1) = E(I - \mu \frac{1}{L} \sum_{i=0}^{L-1} u(kL + i)u(kL + i)^T)\varepsilon(k) + E\mu \frac{1}{L} \sum_{i=0}^{L-1} u(nkL + i)e_o(kL + i)$$
and now using the statistical independence of $u(n)$ and $w(n)$, which implies the statistical independence of $u(n)$ and $\varepsilon(n)$,

$$E[\varepsilon(k + 1)] = (I - \mu E[\frac{1}{L} \sum_{i=0}^{L-1} u(kL + i)u(kL + i)^T])E[\varepsilon(n)] + \mu E[\frac{1}{L} \sum_{i=0}^{L-1} u(nkL + i)e_o(kL + i)]$$

Using the principle of orthogonality which states that $E[u(kL + i)e_o(kL + i)] = 0$, the last equation becomes

$$E[\varepsilon(k + 1)] = (I - \mu E[u(kL + i)u(kL + i)^T])E[\varepsilon(k)] = (I - \mu R)E[\varepsilon(k)]$$

Reminding the equation

$$\zeta(n + 1) = (I - \mu R)\zeta(n)$$  \hspace{1cm} (1)

which was used in the analysis of SD algorithm stability, and identifying now $\zeta(n)$ with $E[\varepsilon(n)]$, we have the following result:

The mean $E[\varepsilon(k)]$ converges to zero, and consequently $E[w(k)]$ converges to $w_o$ iff

$$0 < \mu < \frac{2}{\lambda_{max}}$$ (STABILITY CONDITION) where $\lambda_{max}$ is the largest eigenvalue of the matrix $R = E[u(n)u(n)^T]$.

Stated in words, block LMS algorithm is convergent in mean, iff the stability condition is met.
**Study using small-step assumption**

- The average time constant is
  \[
  \tau_{mse,av} = \frac{L}{2\mu_B \lambda_{av}}
  \]  
  where \(\lambda_{av}\) is the average of the \(M\) eigenvalues of the correlation matrix
  \[
  R = E[u(n)u^T(n)]
  \]

To compare, the average time constant for standard LMS is
  \[
  \tau_{mse,av} = \frac{1}{2\mu \lambda_{av}}
  \]

therefore, the transients have the same convergence speed for block and standard LMS.

- **Misadjustment** The misadjustment
  \[
  M \triangleq \frac{J(\infty) - J_{min}}{J_{min}} = \frac{\mu_B}{2L} tr[R]
  \]
  (where \(J_{min}\) is the MSE of the optimal Wiener filter) is the same as for the standard LMS algorithm.

- **Choice of block size**
  In most application the block size is selected to be equal to the filter length \(L = M\). It is a tradeoff of the following drawbacks:
  - For \(L > M\) the gradient is estimated using more data than the filter itself.
  - For \(L < M\) the data in the current block is not enough to feed the whole tap vector, and consequently some weights are not used.
Frequency Domain Adaptive Filters

- FFT domain computation of the linear convolution with *Overlap-Save* method

We want to compute simultaneously all the outputs of the block filter, corresponding to one block of data. Note that the filter parameters are kept constant during a block processing.

\[
y(kM + m) = \sum_{i=0}^{M-1} w_i (kM + m - i)
\]

\[
y(kM) = \sum_{i=0}^{M-1} w_i (kM - i) = w_0 u(kM) + w_1 u(kM - 1) + \ldots + w_{M-1} u(kM - M + 1)
\]

\[
y(kM + 1) = \sum_{i=0}^{M-1} w_i (kM - i + 1) = w_0 u(kM + 1) + w_1 u(kM) + \ldots + w_{M-1} u(kM - M + 2)
\]

\[
y(kM + 2) = \sum_{i=0}^{M-1} w_i (kM - i + 2) = w_0 u(kM + 2) + w_1 u(kM + 1) + \ldots + w_{M-1} u(kM - M + 3)
\]

\[\ldots\]

\[
y(kM + (M - 1)) = \sum_{i=0}^{M-1} w_i (kM - i + (M - 1)) = w_0 u(kM + (M - 1)) + w_1 u(kM + (M - 2)) + \ldots + w_{M-1} u(kM + 0)
\]

Let us consider two FFT transformed sequences:

- the *M*-length weight vector is padded at the end with *M* zeros and then a 2*M*-length FFT is computed

\[
W = FFT \begin{bmatrix} w \\ 0 \end{bmatrix}
\]
or componentwise:

\[ W_i = \sum_{n=0}^{M-1} w(n)e^{-j\frac{2\pi in}{2M}} \]

- the FFT transform of the vector \( u = [u(kM-M) \ u(kM-M+1) \ldots \ u(kM) \ u(kM+1) \ldots \ u(kM+M-1)] \) is then computed

\[ U_i = \sum_{\ell=0}^{2M-1} u(kM-M+\ell)e^{-j\frac{2\pi i\ell}{2M}} \]

We try to rewrite in a different form the product of the terms \( W_iU_i \) for \( i = 0, \ldots, 2M-1 \):

\[
W_iU_i = \sum_{n=0}^{M-1} w(n)e^{-j\frac{2\pi in}{2M}} \sum_{\ell=0}^{2M-1} u(kM-M+\ell)e^{-j\frac{2\pi i\ell}{2M}} = \sum_{n=0}^{M-1} \sum_{\ell=0}^{2M-1} w(n)u(kM-M+\ell)e^{-j\frac{2\pi i(n+\ell)}{2M}}
\]

\[
= e^{-j\frac{2\pi i(M)}{2M}} \sum_{n=0}^{M-1} w(n)u(kM-n) + e^{-j\frac{2\pi i(M+1)}{2M}} \sum_{n=0}^{M-1} w(n)u(kM-n+1) + \ldots +
\]

\[
+ e^{-j\frac{2\pi i(M+M-1)}{2M}} \sum_{n=0}^{M-1} w(n)u(kM-n+M-1) + \left( e^{-j\frac{2\pi i(0)}{2M}} C_0 + \ldots + e^{-j\frac{2\pi i(M-1)}{2M}} C_{M-1} \right)
\]

\[
= e^{-j\frac{2\pi i(M)}{2M}} y(kM) + e^{-j\frac{2\pi i(M+1)}{2M}} y(kM+1) + \ldots + e^{-j\frac{2\pi i(2M-1)}{2M}} y(kM+M-1) +
\]

\[
+ \left( e^{-j\frac{2\pi i(0)}{2M}} C_0 + \ldots + e^{-j\frac{2\pi i(M-1)}{2M}} C_{M-1} \right) \text{ the } i\text{th element of } FFT \left[ \begin{bmatrix} C \\ y(kM) \end{bmatrix} \right]
\]

Denoting \( y = [y(kM) \ y(kM+1) \ldots \ y(kM+M-1)]^T \), we obtain finally the identity:

\[
\begin{bmatrix} C \\ y \end{bmatrix} = IFFT \left( FFT \left( \begin{bmatrix} w \\ 0 \end{bmatrix} \right) \times FFT (\begin{bmatrix} u \end{bmatrix}) \right)
\]

where by \( \times \) we denoted the element-wise product of the vectors.
• FFT domain computation of the linear correlation

We want to compute simultaneously all entries in the correlation vector needed in the adaptation equation

\[
\phi = \sum_{i=0}^{M-1} e(kM + i)u(kM + i) = \sum_{i=0}^{M-1} \begin{bmatrix} u(kM + i) \\ u(kM + i - 1) \\ . \\ . \\ u(kM + i - (M - 1)) \end{bmatrix} e(kM + i)
\]

\[
\phi_\ell = \sum_{i=0}^{M-1} e(kM + i)u(kM + i - \ell)
\]

\[
\phi_0 = \sum_{i=0}^{M-1} e(kM + i)u(kM + i) = e(kM)u(kM) + \ldots + e(kM + M - 1)u(kM + M - 1)
\]

\[
\ldots
\]

\[
\phi_{M-1} = \sum_{i=0}^{M-1} e(kM + i)u(kM + i - (M - 1))
\]

Let us consider the following FFT transformed sequence:

– the \( M \)-length error vector \( e = [e(kM)\ e(kM+1)\ \ldots\ e(kM+(M-1))]^T \) is padded at the beginning with \( M \) zeros and then a \( 2M \)-length FFT is computed

\[
E = FFT \begin{bmatrix} 0 \\ e \end{bmatrix}
\]
or componentwise:

\[
E_i = \sum_{n=0}^{M-1} e(kM + n)e^{-j\frac{2\pi i(n+M)}{2M}} \quad U_i = \sum_{\ell=0}^{2M-1} u(kM - M + \ell)e^{-j\frac{2\pi i\ell}{2M}}
\]

We try to rewrite in a different form the product of the terms \(E_i U_i\) for \(i = 0, \ldots, 2M - 1\):

\[
E_i U_i = \sum_{n=0}^{M-1} e(kM + n)e^{-j\frac{2\pi i(n+M)}{2M}} \sum_{\ell=0}^{2M-1} u(kM - M + \ell)e^{-j\frac{2\pi i\ell}{2M}} = \sum_{n=0}^{M-1} \sum_{\ell=0}^{2M-1} e(kM + n)u(kM - M + \ell)e^{-j\frac{2\pi i(n+M-\ell)}{2M}}
\]

\[
= e^{-j\frac{2\pi i(M-1)}{2M}} \sum_{n=0}^{M-1} e(kM + n)u(kM + n - (M - 1)) + e^{-j\frac{2\pi i(M-2)}{2M}} \sum_{n=0}^{M-1} e(kM + n)u(kM + n - (M - 2)) + \ldots +
\]

\[
+ e^{-j\frac{2\pi i(0)}{2M}} \sum_{n=0}^{M-1} e(kM + n)u(kM + n) + \left(e^{-j\frac{2\pi i(M)}{2M}} D_M + \ldots + e^{-j\frac{2\pi i(2M-1)}{2M}} D_{2M-1}\right)
\]

\[
= e^{-j\frac{2\pi i(0)}{2M}} \phi_0 + e^{-j\frac{2\pi i(1)}{2M}} \phi_1 + \ldots + e^{-j\frac{2\pi i(M-1)}{2M}} \phi_{M-1} + \left(e^{-j\frac{2\pi i(M)}{2M}} D_M + \ldots + e^{-j\frac{2\pi i(2M-1)}{2M}} D_{2M-1}\right)
\]

\[
= \text{the } i\text{th element of } FFT \begin{bmatrix} \phi \\ D \end{bmatrix}
\]

We obtained finally the identities:

\[
FFT \begin{bmatrix} \phi \\ D \end{bmatrix} = FFT \begin{bmatrix} 0 \\ e \end{bmatrix} \times FFT \begin{bmatrix} u \end{bmatrix} \quad \text{and} \quad \begin{bmatrix} \phi \\ D \end{bmatrix} = IFFT \left( FFT \left( \begin{bmatrix} 0 \\ e \end{bmatrix} \right) \times FFT \left( \begin{bmatrix} u \end{bmatrix} \right) \right)
\]

where by \(\times\) we denoted the element-wise product of the vectors.
The adaptation equation

\[ w(k + 1) = w(k) + \mu \sum_{i=0}^{M-1} u(kM + i)e_{w(k)}(kM + i) = w(k) + \mu \phi \]

Due to linearity of FFT, we can write

\[ \text{FFT} \left[ \begin{bmatrix} w(k+1) \\ 0 \end{bmatrix} \right] = \text{FFT} \left[ \begin{bmatrix} w(k) \\ 0 \end{bmatrix} \right] + \mu \text{FFT} \left[ \begin{bmatrix} \phi \\ 0 \end{bmatrix} \right] \]

The fast LMS algorithm (Frequency Domain Adaptive Filter=FDAF)

For each block of \( M \) data samples do the following:

1. Compute the output of the filter for the block \( kM, \ldots, kM + M - 1 \)

\[ \begin{bmatrix} C \\ y \end{bmatrix} = \text{IFFT} \left( \text{FFT} \left( \begin{bmatrix} w(k) \\ 0 \end{bmatrix} \right) \times \text{FFT} \left( \begin{bmatrix} u \end{bmatrix} \right) \right) \]

2. Compute the correlation vector

\[ \begin{bmatrix} \phi \\ D \end{bmatrix} = \text{IFFT} \left( \text{FFT} \left( \begin{bmatrix} 0 \\ e \end{bmatrix} \right) \times \overline{\text{FFT} \left( \begin{bmatrix} u \end{bmatrix} \right)} \right) \]

3. Update the parameters of the filter

\[ \text{FFT} \left[ \begin{bmatrix} w(k + 1) \\ 0 \end{bmatrix} \right] = \text{FFT} \left[ \begin{bmatrix} w(k) \\ 0 \end{bmatrix} \right] + \mu \text{FFT} \left[ \begin{bmatrix} \phi \\ 0 \end{bmatrix} \right] \]
Fig. 4. Overlap-Save Sectioning. The overlap-save sectioning method performs a linear convolution between a finite-length sequence and an infinite-length sequence by appropriately partitioning the data. The finite-length "sequence" \( w(n) \) (in our case, the adaptive weights) has \( N \) elements; after appending \( N \) zeros, a 2N-point FFT is computed. For the infinite-length input sequence \( x(n) \), the most recent \( N \) data samples are concatenated with the previous block of \( N \) samples; a 2N-point DFT of this extended data vector is then computed. The product of the transformed sequences (i.e., \( Y(k) = X(k)W(k) \)) is processed by a 2N-point inverse FFT (IFFT), yielding a block of output samples. The first \( N \) points of this output frame are discarded, while the last \( N \) points are the desired output samples of a linear convolution.
Fig. 5. Overlap-Save FDAR. This FDAR is based on the overlap-save sectioning procedure for implementing linear convolutions and linear correlations. The gradient constraint ensures that the IDFT of the 2N frequency domain weights yields only N non-zero time-domain weights. Because the DFTs are computed only once for each block of data, there is an end-to-end delay of N samples.
Computational Complexity of the fast LMS algorithm

1 Classical LMS requires $2M$ multiplications per sample, so for a block of $M$ samples there is a need of $2M^2$ multiplications.

2 In the fast LMS algorithm there are 5 FFT transforms, requiring approximately $2M \log(2M)$ real multiplications each, and also other $16M$ operations (when updating the parameters, computing the errors, element-wise multiplications of FFT transformed vectors) so the total is

$$10M \log(2M) + 16M = 10M \log(M) + 26M$$

3 The complexity ratio for the fast LMS to standard LMS is

$$\text{Complexity ratio} = \frac{10M \log(M) + 26M}{2M^2} = \frac{5 \log_2(M) + 13}{M}$$

For $M = 20$ Complexity ratio=0.1
For $M = 32$ Complexity ratio=0.84
For $M = 64$ Complexity ratio=1.49
For $M = 1024$ Complexity ratio=16
For $M = 2048$ Complexity ratio=30
Convergence rate improvement

- In fast LMS, since the weights are adapted in the frequency domain, they can be associated to one mode of the adaptive process. The individual convergence rate may be varied in a straightforward manner. This is different of the mixture of modes type of adaptation, which was found in LMS.

- The convergence time for the \( i \)'th mode is inversely proportional to \( \mu \lambda_i \), where \( \lambda_i \) is the eigenvalue of the correlation matrix \( R \) of the input vector, and \( \lambda_i \) is a measure of the average input power in the \( i \)'th frequency bin.

- All the modes will converge at the same rate by assigning to each weight a different step-size

\[
\mu_i = \frac{\alpha}{P_i}
\]

where \( P_i \) is an estimate of the average power in the \( i \)'th bin, and \( \alpha \) controls the overall time constant of the convergence process

\[
\tau = \frac{2M}{\alpha} \text{ samples}
\]

If the environment is non-stationary, the estimation of \( P_i \) can be carried out by

\[
P_i(k) = \gamma P_i(k - 1) + (1 - \gamma) |U_i(k)|^2, \quad i = 0, 1, \ldots, 2M - 1
\]

where \( \gamma \) is a forgetting factor
Unconstrained frequency-domain adaptive filtering

- In the computation of the gradient, some constraints are imposed in order to achieve a linear correlation, (as opposed to a circular correlation). These constraints are:
  - Discard the last $M$ elements of the inverse FFT of $U^H(k)E(k)$
  - Replace the elements discarded by an appended block of zeros.

- If from the flow-graph of the LMS algorithm the gradient constraints are removed (a FFT block, a IFFT block, the delete block, and the append block), the algorithm is no longer equivalent to block LMS block

\[
W(k + 1) = W(k) + \mu U^H(k)E(k)
\]  

(6)

- The resulting algorithm has a lower complexity (only three FFTs are required).
- The drawbacks:
  - when the number of processed blocks increases, the weight vector no longer converges to the Wiener solution.
  - the steady state error of the unconstrained algorithm is increased compared to the fast LMS algorithm.
Self-orthogonalizing adaptive filters

The self-orthogonalizing adaptive filter was introduced to guarantee a constant convergence rate, not dependent on the input statistics.

- The updating equation is

\[ w(n + 1) = w(n) + \alpha R^{-1}u(n)e(n) \]

- the step size must satisfy \( 0 < \alpha < 1 \) and it was recommended to be selected as

\[ \alpha = \frac{1}{2M} \]

- Example: for white Gaussian input, with variance \( \sigma^2 \),

\[ R = \sigma^2 I \]

and the adaptation becomes the one from the standard LMS algorithm:

\[ w(n + 1) = w(n) + \frac{1}{2M\sigma^2}u(n)e(n) \]

- From the previous example, a two stage procedure can be inferred:

* Step I: Transform the input vector \( u(n) \) into a corresponding vector of uncorrelated variables.
* Step II: use the transformed vector into an LMS algorithm

- Consider first as uncorrelating transformation the Karhunen-Loeve transform:

\[ \nu_i(n) = q^T_i u(n), \quad i = 0, \ldots, M - 1 \]
where \( q_i \) is the eigenvector associated with the \( i \)’th eigenvalue \( \lambda_i \) of the correlation matrix \( R \) of the input vector \( \mathbf{u}(n) \).

- The individual outputs of the KLT are uncorrelated:
  \[
  E\nu_i(n)\nu_j(n) = \begin{cases} 
  \lambda_i, & j = i \\
  0, & j \neq i
  \end{cases}
  \]

- The adaptation equation (Step II) becomes
  \[
  \mathbf{w}(n+1) = \mathbf{w}(n) + \alpha \Lambda^{-1}\nu(n)e(n)
  \]
  or written element-wise, for \( i = 0, 1, \ldots, M-1 \):
  \[
  w_i(n+1) = w_i(n) + \frac{\alpha}{\lambda_i} \nu_i(n)e(n)
  \]

- Replacing the optimal KLT with the (sub)optimal DCT (discrete cosine transform) one obtains the DCT-LMS algorithm.

- The DCT is performed at each sample (the algorithm is no longer equivalent to a block LMS. Advantage: better convergence. Disadvantage: not so computationally efficient.)