

# Block Adaptive Filters and Frequency Domain Adaptive Filters

## Overview

- Block Adaptive Filters
  - Iterating LMS under the assumption of small variations in  $\underline{w}(n)$
  - 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**

{

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

**1 Initialize the algorithm** with an arbitrary parameter vector  $\underline{w}(0)$ , for example  $\underline{w}(0) = 0$ .

**2 Iterate for**  $n = 0, 1, 2, 3, \dots, n_{max}$

**2.0** Read /generate a new data pair,  $(\underline{u}(n), d(n))$

**2.1** (Filter output)  $y(n) = \underline{w}(n)^T \underline{u}(n) = \sum_{i=0}^{M-1} w_i(n) u(n-i)$

**2.2** (Output error)  $e(n) = d(n) - y(n)$

**2.3** (Parameter adaptation)  $\underline{w}(n+1) = \underline{w}(n) + \mu \underline{u}(n) e(n)$

□

**Complexity of the algorithm:**  $2M + 1$  multiplications and  $2M$  additions per iteration

The error signal  $e(n)$  is computed using the parameters  $\underline{w}(n)$ , and we emphasize this by denoting  $e_{\underline{w}(n)}(n)$ .

## Iterating LMS under the assumption of small variations in $\underline{w}(n)$

The new parameters in LMS are evaluated at each time step

$$\begin{aligned}
 \underline{w}(n+L) &= \underline{w}(n+L-1) + \mu \underline{u}(n+L-1) e_{\underline{w}(n+L-1)}(n+L-1) \\
 &= \underline{w}(n+L-2) + \mu \underline{u}(n+L-2) e_{\underline{w}(n+L-2)}(n+L-2) + \mu \underline{u}(n+L-1) e_{\underline{w}(n+L-1)}(n+L-1) \\
 &= \underline{w}(n) + \sum_{i=0}^{L-1} \mu \underline{u}(n+i) e_{\underline{w}(n+i)}(n+i)
 \end{aligned}$$

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

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

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

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

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

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

The output of the filter is

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

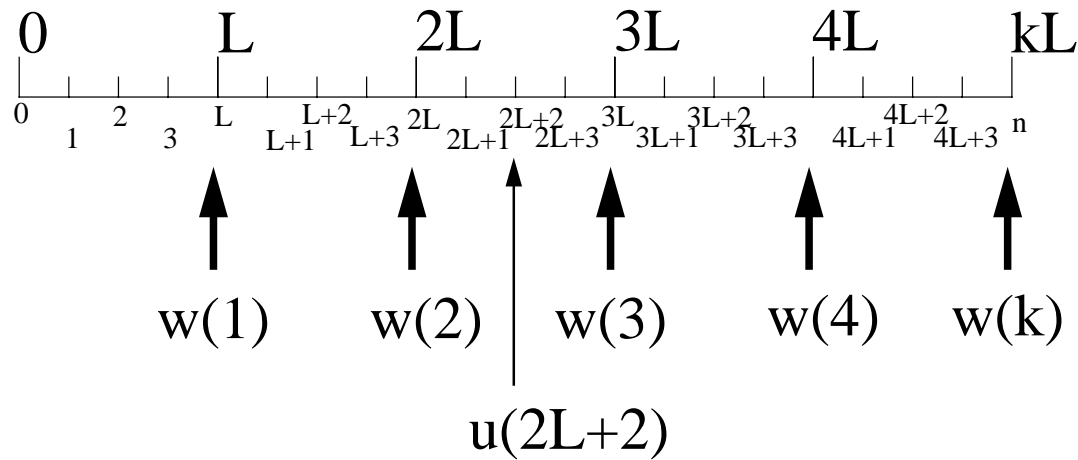
## Block processing

Data used for modifying the partameters 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  $y(kL + i) = \underline{w}^T(k)\underline{u}(kL + i)$
- the error signal  $e(kL + i)$

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



## Block LMS algorithm

Given

{

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

**1 Initialize the algorithm** with an arbitrary parameter vector  $\underline{w}(0)$ , for example  $\underline{w}(0) = 0$ .

**2 Iterate for**  $k = 0, 1, 2, 3, \dots, n_{max}$

**2.0 Initialize**  $\underline{\phi} = 0$

**2.1 Iterate for**  $i = 0, 1, 2, 3, \dots, (L - 1)$

**2.1.0** Read /generate a new data pair,  $(\underline{u}(kL + i), d(kL + i))$

**2.1.1** (Filter output)  $y(kL + i) = \underline{w}(k)^T \underline{u}(kL + i) = \sum_{j=0}^{M-1} w_j(k) u(kL + i - j)$

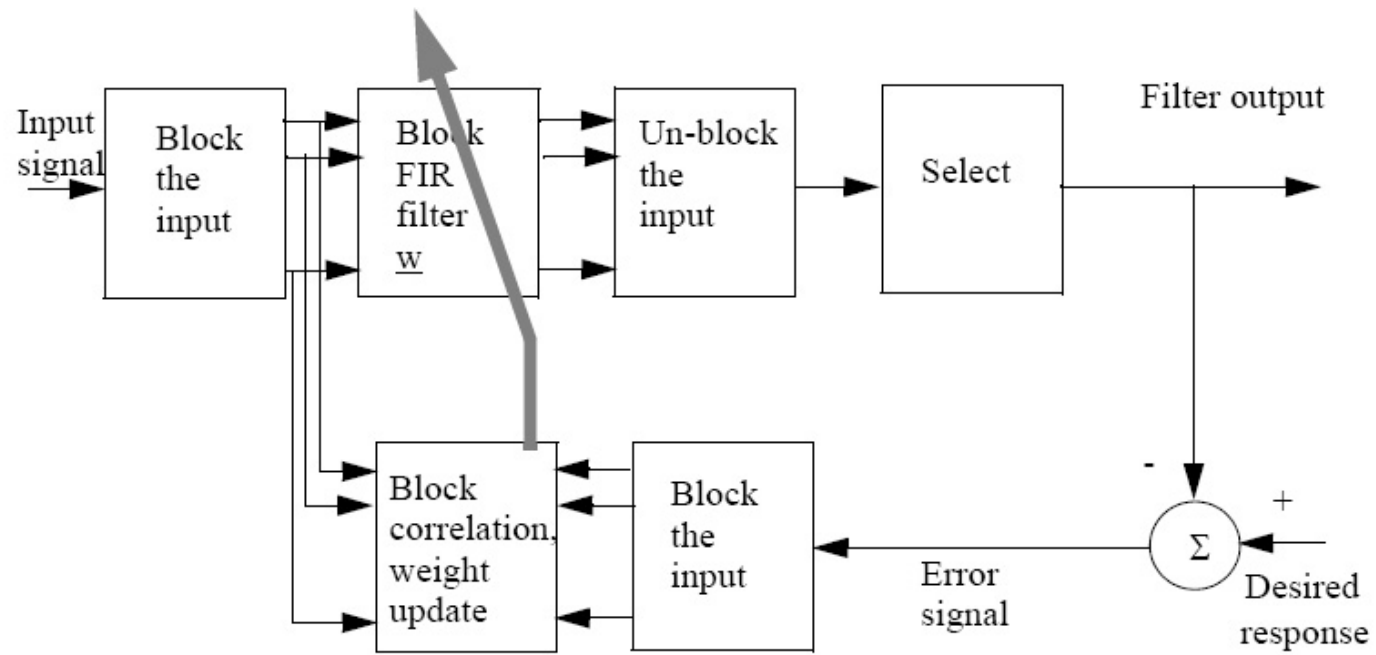
**2.1.2** (Output error)  $e(kL + i) = d(kL + i) - y(kL + i)$

**2.1.3** (Accumulate)  $\underline{\phi} \leftarrow \underline{\phi} + \mu e(kL + i) \underline{u}(kL + i)$

**2.2 (Parameter adaptation)**  $\underline{w}(k + 1) = \underline{w}(k) + \underline{\phi}$

□

**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

$$J = Ee^2(n) = E(d(n) - \underline{w}(n)^T \underline{u}(n))^2$$

has the gradient with respect to the parameter vector  $\underline{w}(n)$

$$\nabla_{\underline{w}(n)} J = -2e(n)\underline{u}(n)$$

The adaptation of parameters in the Block LMS algorithm is

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

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

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

where we denoted by

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

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  $E\mathbf{w}(n)$**

We will subtract the vector  $\mathbf{w}_o$  from the adaptation equation

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

and we will denote  $\mathbf{\varepsilon}(k) = \mathbf{w}(k) - \mathbf{w}_o$

$$\begin{aligned} \mathbf{w}(k+1) - \mathbf{w}_o &= \mathbf{w}(k) - \mathbf{w}_o + \mu \frac{1}{L} \sum_{i=0}^{L-1} \mathbf{u}(kL+i) (d(kL+i) - \mathbf{u}(kL+i)^T \mathbf{w}(k)) \\ \mathbf{\varepsilon}(k+1) &= \mathbf{\varepsilon}(k) + \mu \frac{1}{L} \sum_{i=0}^{L-1} \mathbf{u}(kL+i) (d(kL+i) - \mathbf{u}(kL+i)^T \mathbf{w}_o) + \\ &\quad + \mu \frac{1}{L} \sum_{i=0}^{L-1} (\mathbf{u}(kL+i) \mathbf{u}(kL+i)^T \mathbf{w}_o - \mathbf{u}(kL+i) \mathbf{u}(kL+i)^T \mathbf{w}(k)) \\ &= \mathbf{\varepsilon}(k) + \mu \frac{1}{L} \sum_{i=0}^{L-1} \mathbf{u}(kL+i) e_o(kL+i) - \mu \frac{1}{L} \sum_{i=0}^{L-1} \mathbf{u}(kL+i) \mathbf{u}(kL+i)^T \mathbf{\varepsilon}(k) \\ &= (I - \mu \frac{1}{L} \sum_{i=0}^{L-1} \mathbf{u}(kL+i) \mathbf{u}(kL+i)^T) \mathbf{\varepsilon}(k) + \mu \frac{1}{L} \sum_{i=0}^{L-1} \mathbf{u}(nkL+i) e_o(kL+i) \end{aligned}$$

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

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



and now using the statistical independence of  $\underline{u}(n)$  and  $\underline{w}(n)$ , which implies the statistical independence of  $\underline{u}(n)$  and  $\underline{\varepsilon}(n)$ ,

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

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

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

Reminding the equation

$$\underline{c}(n+1) = (I - \mu R)\underline{c}(n) \tag{1}$$

which was used in the analysis of SD algorithm stability, and identifying now  $\underline{c}(n)$  with  $E\underline{\varepsilon}(n)$ , we have the following result:

The mean  $E\underline{\varepsilon}(k)$  converges to zero, and consequently  $E\underline{w}(k)$   
converges to  $\underline{w}_o$   
iff  
 $0 < \mu < \frac{2}{\lambda_{max}}$  (*STABILITY CONDITION!*) where  $\lambda_{max}$  is the  
largest eigenvalue of the matrix  $R = E[\underline{u}(n)\underline{u}(n)^T]$ .

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

$$\tau_{mse,av} = \frac{L}{2\mu_B\lambda_{av}} \quad (2)$$

where  $\lambda_{av}$  is the average of the  $M$  eigenvalues of the correlation matrix

$$R = E[\underline{u}(n)\underline{u}^T(n)] \quad (3)$$

To compare, the average time constant for standard LMS is

$$\tau_{mse,av} = \frac{1}{2\mu\lambda_{av}} \quad (4)$$

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

- Misadjustment The misadjustment

$$\mathcal{M} \triangleq \frac{J(\infty) - J_{min}}{J_{min}} = \frac{\mu_B}{2L} tr[R] \quad (5)$$

(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.

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

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

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

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

...

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

Let us consider two FFT transformed sequences:

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

$$\underline{W} = FFT \left[ \begin{array}{c} w \\ 0 \end{array} \right]$$

or componentwise:

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

– the FFT transform of the vector  $\underline{u} = [u(kM-M) \ u(kM-M+1) \ \dots \ u(kM) \ u(kM+1) \ \dots \ u(kM+M-1)]$  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  $W_i U_i$  for  $i = 0, \dots, 2M-1$ :

$$\begin{aligned} W_i U_i &= \sum_{n=0}^{M-1} w(n) e^{-j \frac{2\pi i n}{2M}} \sum_{\ell=0}^{2M-1} u(kM-M+\ell) e^{-j \frac{2\pi i \ell}{2M}} = \sum_{n=0}^{M-1} \sum_{\ell=0}^{2M-1} w(n) u(kM-M+\ell) e^{-j \frac{2\pi i (n+\ell)}{2M}} \\ &= e^{-j \frac{2\pi i (M)}{2M}} \sum_{n=0}^{M-1} w(n) u(kM-n) + e^{-j \frac{2\pi i (M+1)}{2M}} \sum_{n=0}^{M-1} w(n) u(kM-n+1) + \dots + \\ &\quad + e^{-j \frac{2\pi i (M+M-1)}{2M}} \sum_{n=0}^{M-1} w(n) u(kM-n+M-1) + \left( e^{-j \frac{2\pi i (0)}{2M}} C_0 + \dots + e^{-j \frac{2\pi i (M-1)}{2M}} C_{M-1} \right) \\ &= e^{-j \frac{2\pi i (M)}{2M}} y(kM) + e^{-j \frac{2\pi i (M+1)}{2M}} y(kM+1) + \dots + e^{-j \frac{2\pi i (2M-1)}{2M}} y(kM+M-1) + \\ &\quad + \left( e^{-j \frac{2\pi i (0)}{2M}} C_0 + \dots + e^{-j \frac{2\pi i (M-1)}{2M}} C_{M-1} \right) = \text{the } i\text{th element of } FFT \left[ \begin{array}{c} \underline{C} \\ \underline{y}(kM) \end{array} \right] \end{aligned}$$

Denoting  $\underline{y} = [y(kM) \ y(kM+1) \ \dots \ y(kM+M-1)]^T$ , we obtain finally the identity:

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

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

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

$$\begin{aligned} \phi_\ell &= \sum_{i=0}^{M-1} e(kM + i) u(kM + i - \ell) \\ \phi_0 &= \sum_{i=0}^{M-1} e(kM + i) u(kM + i) = e(kM) u(kM) + \dots + e(kM + M - 1) u(kM + M - 1) \\ \dots & \\ \phi_{M-1} &= \sum_{i=0}^{M-1} e(kM + i) u(kM + i - (M - 1)) \end{aligned}$$

Let us consider the following FFT transformed sequence:

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

$$\underline{E} = FFT \begin{bmatrix} \underline{0} \\ \underline{e} \end{bmatrix}$$

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  $E_i \bar{U}_i$  for  $i = 0, \dots, 2M - 1$ :

$$\begin{aligned} E_i \bar{U}_i &= \sum_{n=0}^{M-1} e(kM + n) e^{-j \frac{2\pi i(n+M)}{2M}} \sum_{\ell=0}^{2M-1} u(kM - M + \ell) e^{j \frac{2\pi i \ell}{2M}} = \sum_{n=0}^{M-1} \sum_{\ell=0}^{2M-1} e(kM + n) u(kM - M + \ell) e^{-j \frac{2\pi i(n+M-\ell)}{2M}} \\ &= e^{-j \frac{2\pi i(M-1)}{2M}} \sum_{n=0}^{M-1} e(kM + n) u(kM + n - (M-1)) + e^{-j \frac{2\pi i(M-2)}{2M}} \sum_{n=0}^{M-1} e(kM + n) u(kM + n - (M-2)) + \dots + \\ &\quad + e^{-j \frac{2\pi i(0)}{2M}} \sum_{n=0}^{M-1} e(kM + n) u(kM + n) + \left( e^{-j \frac{2\pi i(M)}{2M}} D_M + \dots + e^{-j \frac{2\pi i(2M-1)}{2M}} D_{2M-1} \right) \\ &= e^{-j \frac{2\pi i(0)}{2M}} \phi_0 + e^{-j \frac{2\pi i(1)}{2M}} \phi_1 + \dots + e^{-j \frac{2\pi i(M-1)}{2M}} \phi_{M-1} + \left( e^{-j \frac{2\pi i(M)}{2M}} D_M + \dots + e^{-j \frac{2\pi i(2M-1)}{2M}} D_{2M-1} \right) \\ &= \text{the } i\text{th element of } FFT \left[ \begin{bmatrix} \phi \\ \underline{D} \end{bmatrix} \right] \end{aligned}$$

We obtained finally the identities:

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

where by  $\times$  we denoted the element-wise product of the vectors.

## The adaptation equation

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

Due to linearity of FFT, we can write

$$FFT \begin{bmatrix} \underline{w}(k+1) \\ \underline{0} \end{bmatrix} = FFT \begin{bmatrix} \underline{w}(k) \\ \underline{0} \end{bmatrix} + \mu FFT \begin{bmatrix} \underline{\phi} \\ \underline{0} \end{bmatrix}$$

## 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  $kM, \dots, kM + M - 1$

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

**2** Compute the correlation vector

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

**3** Update the parameters of the filter

$$FFT \begin{bmatrix} \underline{w}(k+1) \\ \underline{0} \end{bmatrix} = FFT \begin{bmatrix} \underline{w}(k) \\ \underline{0} \end{bmatrix} + \mu FFT \begin{bmatrix} \underline{\phi} \\ \underline{0} \end{bmatrix}$$

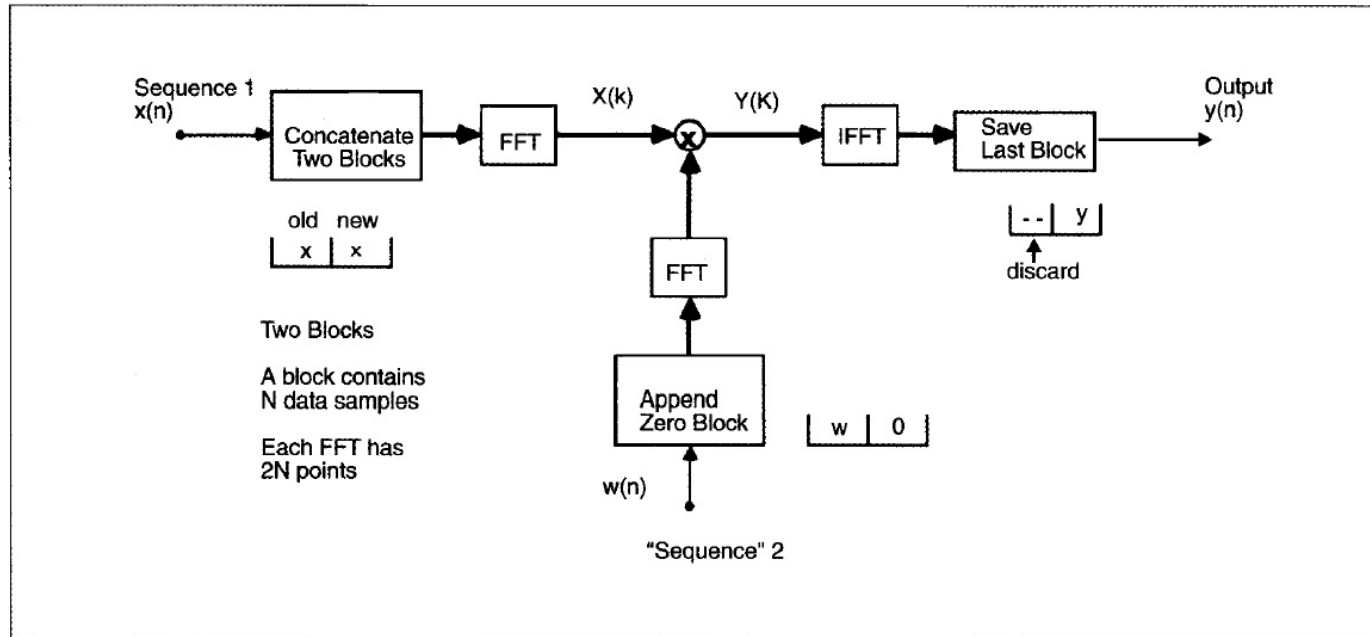


Fig. 4. Overlap-Save Sectioning. The overlap-save sectioning method performs a linear convolution between a finite-length sequence and an infinite-length sequence by appropriately partitioning the data. The finite-length "sequence"  $w(n)$  (in our case, the adaptive weights) has  $N$  elements; after 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.



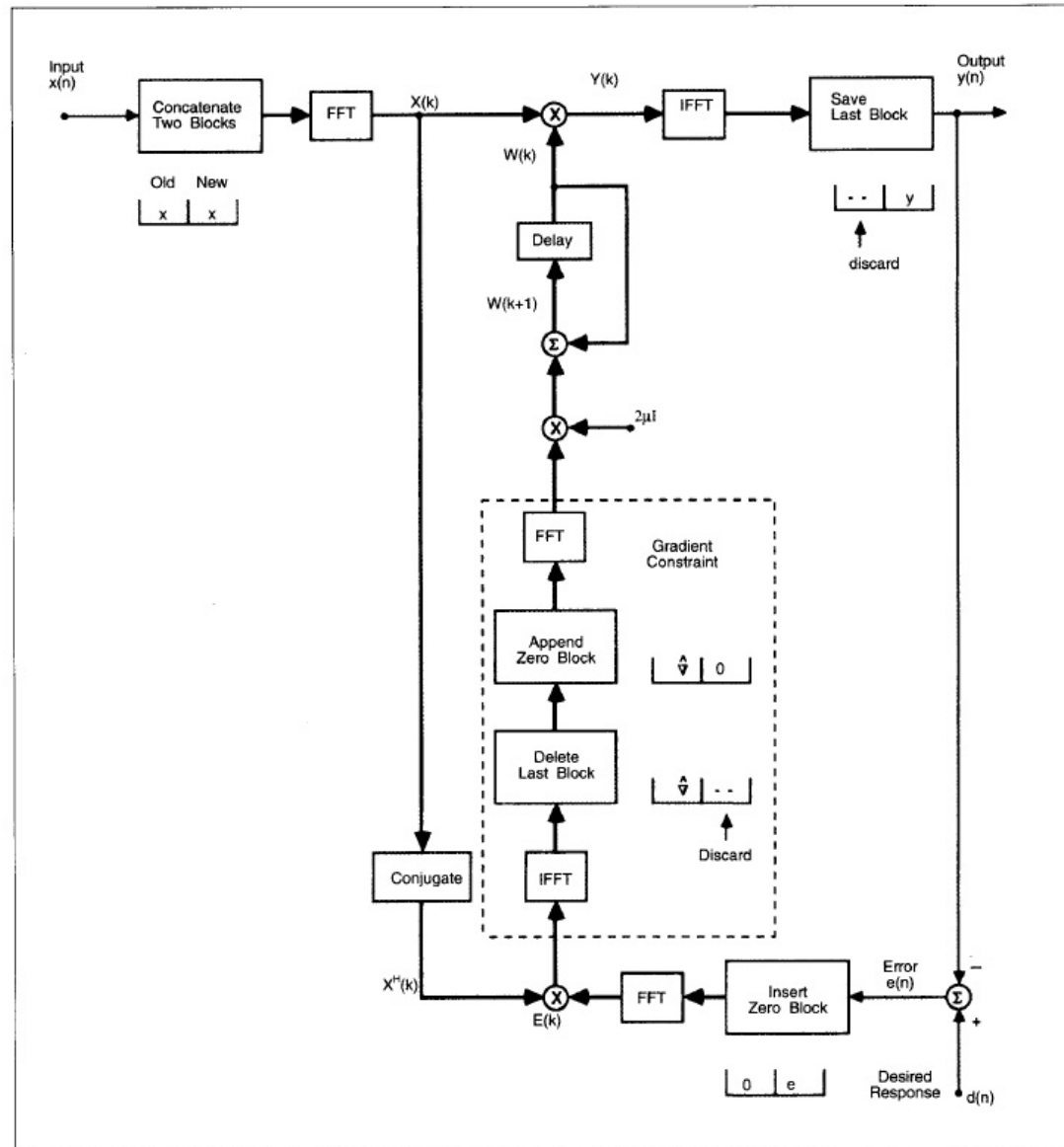


Fig. 5. Overlap-Save FDAF. This FDAF is based on the overlap-save sectioning procedure for implementing linear convolutions and linear correlations. The gradient constraint ensures that the IDFT of the  $2N$  frequency-domain weights yields only  $N$  non-zero time-domain weights. Because the DFTs are 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

$$P_i(k) = \gamma P_i(k-1) + (1-\gamma)|U_i(k)|^2, \quad i = 0, 1, \dots, 2M-1$$

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  $\underline{U}^H(k)\underline{E}(k)$
  - \* 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

$$\underline{W}(k+1) = \underline{W}(k) + \mu \underline{U}^H(k) \underline{E}(k) \quad (6)$$

- 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

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

- 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:

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

- From the previous example, a two stage procedure can be inferred:
  - \* Step I: Transform the input vector  $\underline{u}(n)$  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:

$$\nu_i(n) = \underline{q}_i^T \underline{u}(n), \quad , i = 0, \dots, M-1$$

where  $\underline{q}_i$  is the eigenvector associated with the  $i$ 'th eigenvalue  $\lambda_i$  of the correlation matrix  $R$  of the input vector  $\underline{u}(n)$ .

- 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

$$\underline{w}(n+1) = \underline{w}(n) + \alpha \Lambda^{-1} \underline{\nu}(n)e(n)$$

or written element-wise, for  $i = 0, 1, \dots, M-1$ :

$$w_i(n+1) = w_i(n) + \frac{\alpha}{\lambda_i} \nu_i(n)e(n)$$

- 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.