POLITECNICO DI MILANO

Adaptive filtering: the Least Mean Squares algorithm

Antonio Canclini
Augusto Sarti

Introduction

- Method of Steepest Descent finds the optimum solution in Wiener sense through an iterative process

- The algorithm computes the gradient of the cost function

$$J(n) = E\{e(n)e^*(n)\}$$

- Adaptive algorithm in Steepest Descent:

$$\nabla J(n) = -2\mathbf{p} + 2\mathbf{R}\mathbf{w}(n)$$

$$\mathbf{w}(n+1) = \mathbf{w}(n) + \mu[\mathbf{p} - \mathbf{R}\mathbf{w}(n)]$$

- Definitions:

$$\mathbf{p} = E\{\mathbf{u}(n)d^*(n)\}$$

$$\mathbf{R} = E\left\{\mathbf{u}(n)\mathbf{u}^H(n)\right\}$$

- Replacing the expressions of $\mathbf{R}$ and $\mathbf{p}$ in that of the gradient, we obtain:

$$\mathbf{p} - \mathbf{R}\mathbf{w}(n) = E\left\{\mathbf{u}(n)[d^*(n) - \mathbf{u}^H(n)\mathbf{w}(n)]\right\} = E\left\{\mathbf{u}(n)[d^*(n) - \hat{d}^*(n)]\right\}$$

where $\hat{d}(n) = \mathbf{w}^H(n)\mathbf{u}(n)$ is the filter output (i.e., the estimation of the desired response given $\mathbf{u}(n) = [u(n), u(n-1), \ldots, u(n-M+1)]^H$

- Thus, the update equation of steepest descent method can be rewritten as

$$\begin{aligned}\mathbf{w}(n+1) &= \mathbf{w}(n) + \mu E\left\{\mathbf{u}(n)[d^*(n) - \hat{d}^*(n)]\right\} \\ &= \mathbf{w}(n) + \mu E\left\{\mathbf{u}(n)e^*(n)\right\}\end{aligned}$$

The LMS algorithm

- Problem: the expectation $E\left\{e(n)e^*(n)\right\}$ of the mean squared error is not available in most cases
  - it would require prior knowledge of $\mathbf{R}$ and $\mathbf{p}$ (need infinite data)
  - we can only estimate it from the available data
- Possible solution: use the stochastic gradient, i.e. replace the expectation with the current value (instantaneous estimation)

gradient: $-2E\left\{\mathbf{u}(n)e^*(n)\right\}$

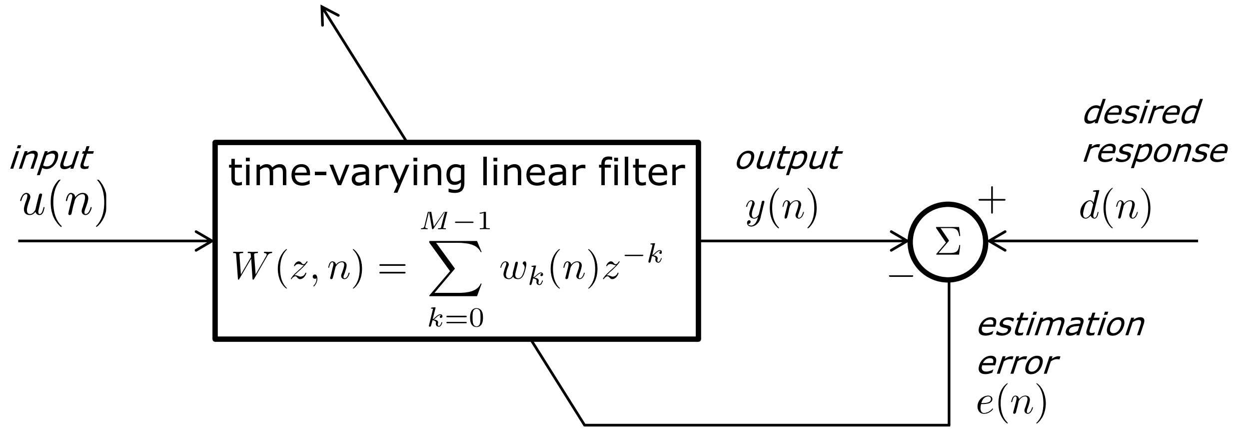
stochastic gradient: $-2\mathbf{u}(n)e^*(n)$

- The Least Mean Squares (LMS) algorithm is thus defined by the following update equation:

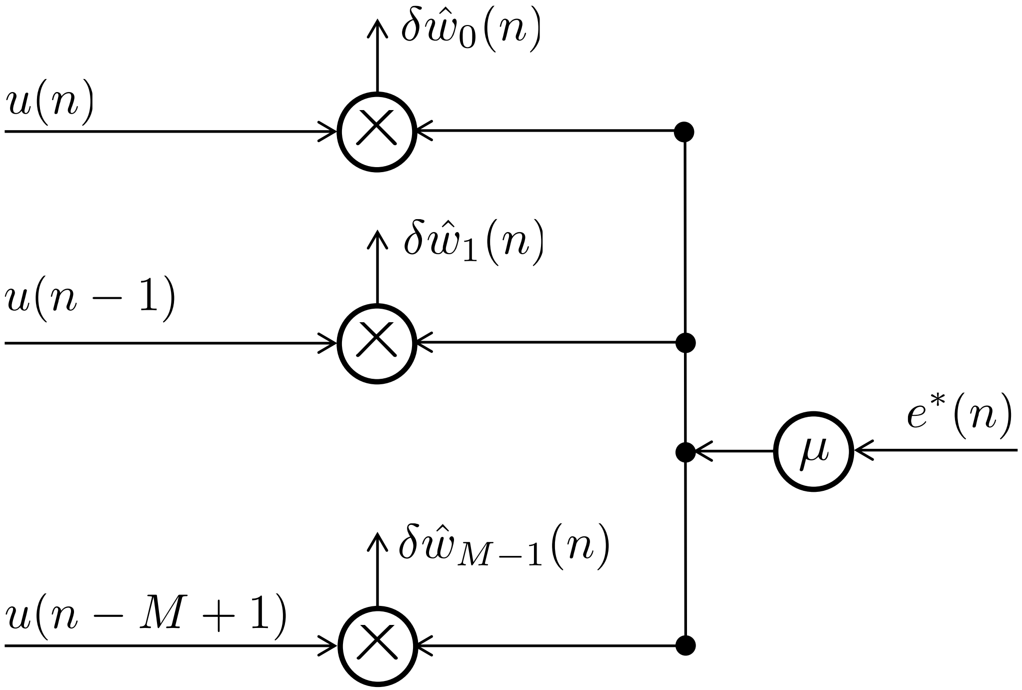
$$\boxed{\mathbf{w}(n+1) = \mathbf{w}(n) + \mu\mathbf{u}(n)e^*(n)}$$

The LMS algorithm

- The LMS adaptation algorithm includes the following steps:
  1. A filtering process which involves
     - the computation of the output of the transversal filter
     - the estimation of the error signal, through the comparison of the desired response with the output
  2. An adaptive process, which involves the automatic adjustment of the tap filters in accordance with the estimation error
- Thus, the block schematic is completely similar to that of the steepest descent method:



- However, LMS is substantially different from the steepest descent method as far as the adaptation step is concerned
- As no expectation must be computed, the stochastic gradient is obtained istantaneously using a bank of multipliers (**much more efficient** with respect to using a bank of correlators, as needed for the steepest descent method):



correction of the kth tap at time n:

$$\delta\hat{w}_k(n) \triangleq \mu\, u(n-k)e^*(n)$$

Examples of application: instantaneous frequency measurement

- Consider the problem of measuring the istantaneous frequency of a narrow-band signal characterized by a rapidly varying power spectrum

- We assume that the narrow-band signal is generated by a time varying Auto-Regressive (AR) process:

$$u(n) = -\sum_{k=1}^{M} a_k(n)u(n-k) + v(n)$$

- Idea: use LMS as a linear predictor for adaptively estimating the time varying AR parameters, i.e.

$$a_k(n) \quad k = 1, 2, \ldots, M$$

Examples of application: instantaneous frequency measurement

- The power spectrum of the time varying AR process is given by

$$S_{\mathrm{AR}}(\omega, n) = \frac{\sigma_v^2(n)}{\left|1 + \sum_{k=1}^{M} a_k(n) e^{-j\omega k}\right|^2} \qquad -\pi < \omega < \pi$$

- Using LMS as a linear predictor, the prediction error corresponds to the estimation error:

$$f(n) = u(n) - \sum_{k=1}^{M} w_k(n) u(n-k)$$

- The adaptation step is therefore given by:

$$w_k(n+1) = w_k(n) + \mu u(n-k) f(n)$$

Examples of application: instantaneous frequency measurement

- Let us ignore the estimation of $\sigma_v^2(n)$. We only use the tap weights of the adaptive predictor to define the following time-varying frequency function (the weights $w_k(n)$ are an estimate of $-a_k(n)$):

$$F(\omega, n) = \frac{1}{\left|1 - \sum_{k=1}^{M} w_k(n) e^{-j\omega k}\right|^2} \qquad -\pi < \omega < \pi$$

- We now assume that:
  1. The predictor has been in operation sufficently long
  2. The step-size parameter enables a good tracking of the input data
  3. The signal is slow varying in the time interval

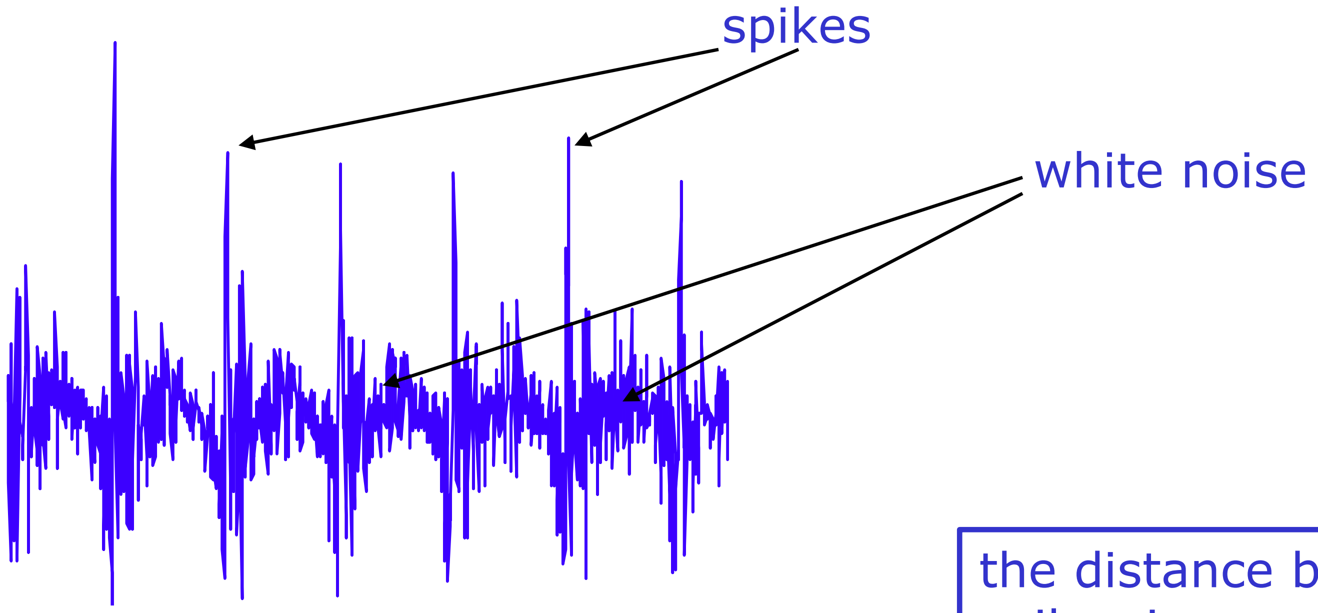
- Given the validity of the previous assumptions, we can find that the frequency function $F(\omega, n)$ has a peak at the instantaneous frequency of the input signal $u(n)$

- See the Matlab example

Examples of application: pitch extraction through long term prediction

- **Preamble**: speech data can be thought as the signal emitted by the vocal cords filtered by the glottis

- Typical waveform emitted by vocal cords:



the distance between spikes is generally referred as pitch

Examples of application:
pitch extraction through long term prediction

- Given an approximate estimation of the pitch (e.g., obtained from the auto-correlation method), this first estimation can be refined using the long term prediction

- **Idea**: estimate the time lag between two successive spikes

- **Assumption: the distance between two successive spikes is bigger than the length of the predictor**

- Given the validity of the previous hypothesis, the prediction filter fails the estimation when it meets a spike and produces a big estimation error

- **Method**: use the estimation error to obtain the position of the spikes

Examples of application: pitch extraction through long term prediction

- Realization:

1. Filtering step: $\hat{u}(n+\Delta) = \mathbf{w}^T(n)\mathbf{u}(n)$, with

$$\mathbf{w}(n) = [w_0(n), w_1(n), \ldots, w_{M-1}(n)]^T$$ → current filter taps

$$\mathbf{u}(n) = [u(n), u(n-1), \ldots, u(n-M+1)]^T$$ → input signal

2. Instantaneous pitch estimation: extract the peaks of the prediction error

$$e(n) = u(n) - \hat{u}(n)$$

3. Adaptation step:

$$\mathbf{w}(n+1) = \mathbf{w}(n) + \mu\mathbf{u}(n)[u(n+\Delta) - \hat{u}(n+\Delta)]$$

- Consider the following definitions:

Weight error set: $\boldsymbol{\varepsilon}(n) = \mathbf{w}(n) - \mathbf{w}_o$

Weight error set updating: $\boldsymbol{\varepsilon}(n+1) = [\mathbf{I} - \mu\mathbf{u}(n)\mathbf{u}^H(n)]\boldsymbol{\varepsilon}(n) + \mu\mathbf{u}(n)e_o^*(n)$

Optimum estimation error: $e_o(n) = d(n) - \mathbf{w}_o\mathbf{u}^H(n)$

- **Direct averaging method**: in order to study the convergence, under **the hypothesis of a small step-size parameter**, the solution of the stochastic equation is close to the solution of another stochastic system in which the system matrix is equal to the ensemble average

$$E\left\{\mathbf{I} - \mu\mathbf{u}(n)\mathbf{u}^H(n)\right\} = \mathbf{I} - \mu\mathbf{R}$$

- Thus we can write:

$$\boldsymbol{\varepsilon}(n+1) = (\mathbf{I} - \mu\mathbf{R})\boldsymbol{\varepsilon}(n) + \mu\mathbf{u}(n)e_o^*(n)$$

Independency theory

- In what follows, we will restric ourselves to a statistical analysis of the LMS algorithm under the *indipendence assumption*, consisting of four points:

1. $\mathbf{u}(1), \mathbf{u}(2), \ldots, \mathbf{u}(n)$ are statistically indipendent vectors;
2. at time n, the tap input vector $\mathbf{u}(n)$ is statistically indipendent of all previous samples of the desired response $d(1), d(2), \ldots, d(n-1)$
3. at time n, the desired response is dependent on the corresponding tap input vector $\mathbf{u}(n)$ and indipendent from the previous samples of the desired response
4. $d(n)$ and $\mathbf{u}(n)$ consist of mutually Gaussian distributed random variables for all n

Convergence criteria

- Two main convergence criteria can be adopted
  - $D(n) = E\left\{\|\boldsymbol{\varepsilon}(n)\|^2\right\} \quad \text{constant as } n \to \infty$
  - $J(n) = E\left\{|e(n)|^2\right\} \quad \text{constant as } n \to \infty$
- The two convergence criteria are somehow equivalent: in fact it can be proved that

$$\lambda_{\min} D(n) \leq J_{\text{ex}}(n) \leq \lambda_{\max} D(n)$$

where $J_{\text{ex}}(n) \triangleq J(n) - J_{\min}$ is called Excess Mean Squared Error

- It turns that it is sufficient to focus on the MSE criterion $J(n)$
- In practice, we are requiring the algorithm to converge "close enough" to the optimum solution, so that the excess MSE $J_{\text{ex}}(n)$ is constant after an infinite number of iterations

Weight-Error correlation matrix

- The correlation matrix of the weight-error vector is

$$\mathbf{K}(n) = E\left\{\boldsymbol{\varepsilon}(n)\boldsymbol{\varepsilon}^H(n)\right\}$$

- Applying this definition to the update equation and invoking the independence assumption, we get

$$\mathbf{K}(n+1) = (\mathbf{I} - \mu\mathbf{R})\mathbf{K}(n)(\mathbf{I} - \mu\mathbf{R}) + \mu^2 J_{\min}\mathbf{R}$$

  - The first term $(\mathbf{I} - \mu\mathbf{R})\mathbf{K}(n)(\mathbf{I} - \mu\mathbf{R})$ is the result of the outer product of $(\mathbf{I} - \mu\mathbf{R})\boldsymbol{\varepsilon}(n)$ with itself
  - The expectation of the cross-product term, $\mu e_o(n)(\mathbf{I} - \mu\mathbf{R})\boldsymbol{\varepsilon}(n)\mathbf{u}^H(n)$, is zero by virtue of the implied independence of $\boldsymbol{\varepsilon}(n)$ and $\mathbf{u}(n)$
  - The last term, $\mu^2 J_{\min}\mathbf{R}$, is obtained by applying the Gaussian factorization theorem to expression $\mu^2 e_o^*(n)\mathbf{u}(n)\mathbf{u}^H(n)e_o(n)$

- Since the auto-correlation matrix $\mathbf{R}$ is positive definite and μ is positive, the term $\mathbf{K}(n+1)$ is positive definite, provided that $\mathbf{K}(n)$ is positive definite (easy to prove by induction)

Excess Mean-Squared Error

- The previous matrix difference equation provides a useful tool for determining the transient behaviour of the MSE of LMS algorithm

- We can express the estimation error as follows:

$$\begin{aligned} e(n) &= d(n) - \mathbf{w}^H(n)\mathbf{u}(n) \\ &= d(n) - \mathbf{w}_o^H\mathbf{u}(n) - \boldsymbol{\varepsilon}^H(n)\mathbf{u}(n) \\ &= e_o(n) - \boldsymbol{\varepsilon}^H(n)\mathbf{u}(n) \end{aligned}$$

- The MSE of the estimation error is therefore

$$J(n) = E\left\{|e(n)|^2\right\} = J_{\min} + E\left\{\boldsymbol{\varepsilon}^H(n)\mathbf{u}(n)\mathbf{u}^H(n)\boldsymbol{\varepsilon}(n)\right\}$$

- Moreover, it can be proved that $E\left\{\boldsymbol{\varepsilon}^H(n)\mathbf{u}(n)\mathbf{u}^H(n)\boldsymbol{\varepsilon}(n)\right\} = \mathrm{tr}[\mathbf{RK}(n)]$

$$\Rightarrow \boxed{J_{\mathrm{ex}}(n) = J(n) - J_{\min} = \mathrm{tr}[\mathbf{RK}(n)]}$$

Excess Mean-Squared Error

- Consider now the eigenvalue decomposition of the autocorrelation matrix and of the weight error matrix:
  - $\mathbf{Q}^H\mathbf{R}\mathbf{Q} = \mathbf{\Lambda} = \mathrm{diag}[\lambda_1, \lambda_2, \ldots, \lambda_M]$
  - $\mathbf{Q}^H\mathbf{K}(n)\mathbf{Q} = \mathbf{X}(n)$
- It is easy to verify that $\mathrm{tr}[\mathbf{R}\mathbf{K}(n)] = \mathrm{tr}[\mathbf{\Lambda}\mathbf{X}(n)]$, and therefore:

$$\boxed{J_{\mathrm{ex}}(n) = \mathrm{tr}[\mathbf{\Lambda}\mathbf{X}(n)] = \sum_{i=1}^{M} \lambda_i x_i(n)}$$

- Moreover, we can write the update equation for the eigenvalues:

$$\mathbf{X}(n+1) = (\mathbf{I} - \mu\mathbf{\Lambda})\mathbf{X}(n)(\mathbf{I} - \mu\mathbf{\Lambda}) + \mu^2 J_{\min}\mathbf{\Lambda}$$

- The last equation can be rewritten in terms of the matrix elements:

$$x_i(n+1) = (1-\mu\lambda_i)^2 x_i(n) + \mu^2 J_{\min}\lambda_i$$

- Consider also the following definitions:
  - $\mathbf{x}(n) \triangleq [x_1(n), x_2(n), \ldots, x_M(n)]^T$
  - $\boldsymbol{\lambda} \triangleq [\lambda_1, \lambda_2, \ldots, \lambda_M]^T$
  - $\mathbf{B} \triangleq \begin{bmatrix} b_{11} & b_{12} & \ldots & b_{1M} \\ b_{21} & b_{22} & \ldots & b_{2M} \\ \vdots & \vdots & \ddots & \vdots \\ b_{M1} & b_{M2} & \ldots & b_{MM} \end{bmatrix}, \quad \text{with} \quad b_{ij} = \begin{cases} (1-\mu\lambda_i)^2 & \text{when } i=j \\ \mu^2\lambda_i\lambda_j & \text{otherwise} \end{cases}$

- We can finally write:

$$\boxed{\mathbf{x}(n+1) = \mathbf{B}\mathbf{x}(n) + \mu^2 J_{\min}\boldsymbol{\lambda}}$$

- The solution of the previous recursive equation is given by

$$\mathbf{x}(n) = \sum_{i=1}^{M} c_i^n \mathbf{g}_i \mathbf{g}_i^T [\mathbf{x}(0) - \mathbf{x}(\infty)] + \mathbf{x}(\infty)$$

where:

- c_i are the eigenvalues of the matrix $\mathbf{B}$
- $\mathbf{g}_i$ are the eigenvectors of $\mathbf{B}$ associated to c_i
- $\mathbf{x}(0)$ and $\mathbf{x}(\infty)$ are the initial and final value of $\mathbf{x}(n)$, respectively

- We can therefore write a final expression for the excess MSE:

$$\begin{aligned} J_{\text{ex}}(n) = \boldsymbol{\lambda}^T \mathbf{x}(n) &= \sum_{i=1}^{M} c_i^n \lambda_i \mathbf{g}_i \mathbf{g}_i^T [\mathbf{x}(0) - \mathbf{x}(\infty)] + \boldsymbol{\lambda}^T \mathbf{x}(\infty) \\ &= \sum_{i=1}^{M} c_i^n \lambda_i \mathbf{g}_i \mathbf{g}_i^T + J_{\text{ex}}(\infty) \\ &= \sum_{i=1}^{M} c_i^n \gamma_i + J_{\text{ex}}(\infty) \end{aligned}$$

- The MSE (i.e., the cost function) can be rewritten as

$$J(n) = J_{\min} + J_{\mathrm{ex}}(n) = J_{\min} + \sum_{i=1}^{M} c_i^n \gamma_i + J_{\mathrm{ex}}(\infty)$$

- It is now possible to analyze the transient behaviour of the MSE:

  - **Property 1**: the transient component of $J(n)$ does not exhibit oscillations, as the terms c_i are all positive real numbers

  - **Property 2**: if the eigenvalues of the matrix $\mathbf{B}$ are all smaller than 1, then the transient component dies out when

$$0 < \mu < \frac{2}{\lambda_{\max}}$$

  - **Property 3**: the excess MSE is smaller than the minimum MSE (i.e.) if $J_{\mathrm{ex}}(\infty) < J_{\min}$

$$\sum_{i=1}^{M} \frac{2\lambda_i}{2 - \mu\lambda_i} < 1$$

Block adaptive filter

- Block adaptive filter is an interesting variant of LMS algorithm

- The input signal is partitioned into L-point blocks

- The adaptation of the filter is performed on a block-by-block basis, i.e. the weights are fixed over a block of data

- Notation:

  - we use k to denote the block index
  - the tap weight vector for the kth block is

$$\mathbf{w}(k) = [w_0(k), w_1(k), \ldots, w_{M-1}(k)]^T, \quad k = 0, 1, \ldots$$

  - the index n is reserved to the original sample time

$$n = kL + i, \quad i = 0, 1, \ldots, M-1, \quad k = 0, 1, \ldots$$

  - the input vector is, as usual: $\mathbf{u}(n) = [u(n), u(n-1), \ldots, u(n-M+1)]^T$

Block adaptive filter

- The block-based filtering process is thus described by

$$\begin{aligned} y(kL+i) &= \mathbf{w}^H(k)\mathbf{u}(kL+i) \\ &= \sum_{l=0}^{M-1} w_l(k)u(kL+i-l)\,, \quad i=0,1,\ldots M-1 \end{aligned}$$

- The error signal is computed as

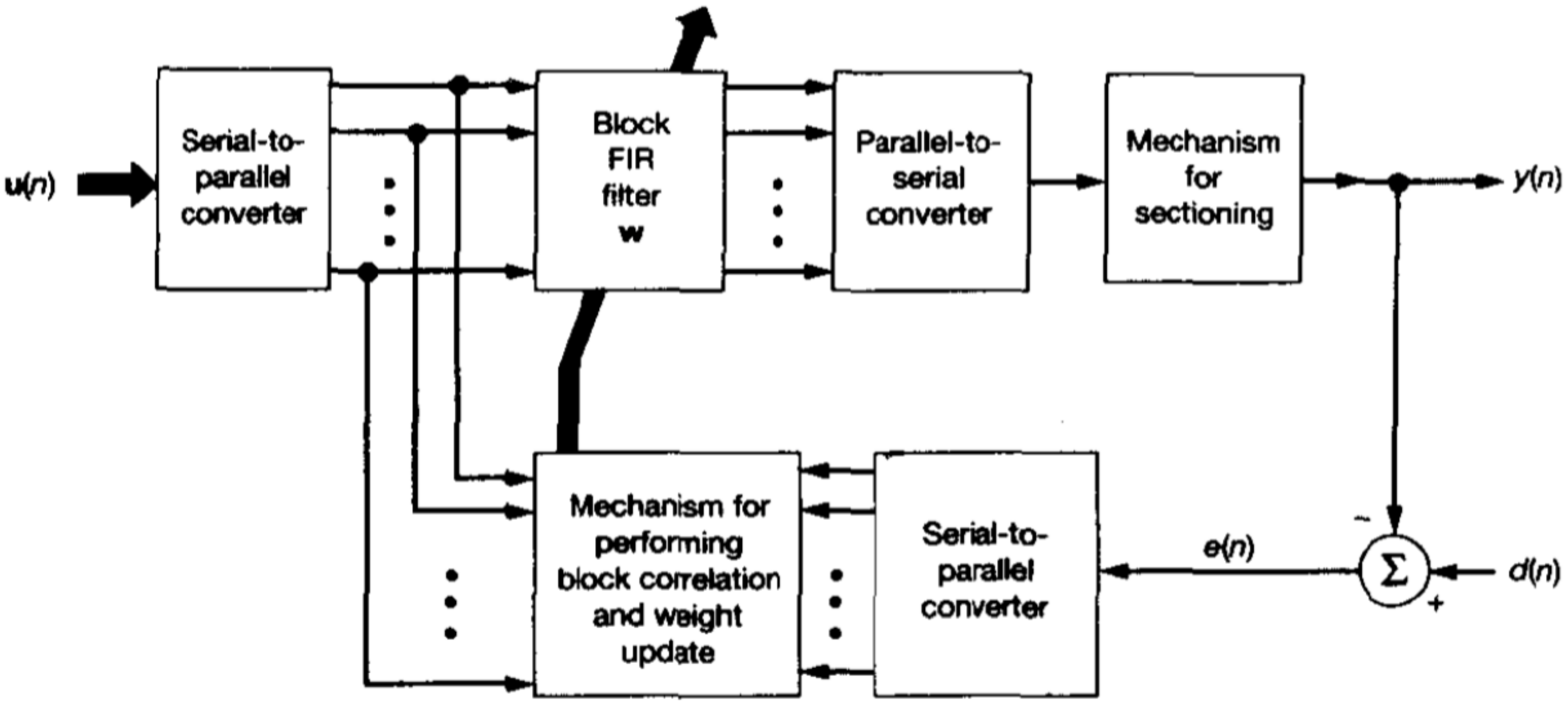
$$e(kL+i) = d(kL+i) - y(kL+i)$$

- The adptation step is performed for each block, as follows:

$$\mathbf{w}(k+1) = \mathbf{w}(k) + \mu \sum_{l=0}^{M-1} \mathbf{u}(kL+i)e^*(kL+i)$$

$$= \mathbf{w}(k) + \mu\boldsymbol{\Phi}(k)$$

Block Adaptive FIltering



Choice of the block size

- Example with $L = M = 3$

$$(k-1)\text{th block}\quad \begin{bmatrix} u(3k-3) & u(3k-4) & u(3k-5) \\ u(3k-2) & u(3k-3) & u(3k-4) \\ u(3k-1) & u(3k-2) & u(3k-3) \end{bmatrix} \begin{bmatrix} w_0(k-1) \\ w_1(k-1) \\ w_2(k-1) \end{bmatrix} = \begin{bmatrix} y(3k-3) \\ y(3k-2) \\ y(3k-1) \end{bmatrix}$$

$$k\text{th block}\quad \begin{bmatrix} u(3k) & u(3k-1) & u(3k-2) \\ u(3k+1) & u(3k) & u(3k-1) \\ u(3k+2) & u(3k+1) & u(3k) \end{bmatrix} \begin{bmatrix} w_0(k) \\ w_1(k) \\ w_2(k) \end{bmatrix} = \begin{bmatrix} y(3k) \\ y(3k+1) \\ y(3k+2) \end{bmatrix}$$

$$(k+1)\text{th block}\quad \begin{bmatrix} u(3k+3) & u(3k+2) & u(3k+1) \\ u(3k+4) & u(3k+3) & u(3k+2) \\ u(3k+5) & u(3k+4) & u(3k+3) \end{bmatrix} \begin{bmatrix} w_0(k+1) \\ w_1(k+1) \\ w_2(k+1) \end{bmatrix} = \begin{bmatrix} y(3k+3) \\ y(3k+4) \\ y(3k+5) \end{bmatrix}$$

Block adaptive filter

- As the adaptation is performed only every L samples, it turns that the stochastic gradient is time-averaged over L samples

- In other words, operating on a block-basis, the stochastic gradient is replaced by a better estimate of the gradient of the cost function, obtained as

$$\hat{\boldsymbol{\nabla}} J(k) = -\frac{2}{L} \sum_{i=0}^{L-1} \mathbf{u}(kL+i) e^{*}(kL+i)$$

- Then, in terms of the gradient, the block LMS algorithm can be reformulated as follows:

$$\mathbf{w}(k+1) = \mathbf{w}(n) - \frac{L}{2} \mu \hat{\boldsymbol{\nabla}} J(k)$$

Convergence properties of block LMS

- Convergence analysis of block LMS algorithm is similar to LMS algorithm, thus we will omit it

- Main results:
  - The tap weight vector converges to the Wiener optimum solution as the number of iterations approaches infinity (assuming stationary signals):

$$\lim_{k\to\infty} E\left\{\mathbf{w}(k)\right\} = \mathbf{R}^{-1}\mathbf{p} = \mathbf{w}_o$$

  - Condition to be satisfied by the step-size parameter:

$$0 < \mu < \frac{2}{\lambda_{\max}}$$

  - In order to satisfy the condition $J_{\mathrm{ex}}(\infty) < J_{\min}$, with respect to LMS a more stringent condition must be satisfied:

$$0 < \mu < \frac{2}{L\sum_{i=1}^{M}\lambda_i}$$

Choice of the block size

- The choice of the block size L is crucial for an efficient implementation of the algorithm
  - when $L > M$, redundant operations are involved in adaptive process
  - when $L < M$, some of the tap weights of the filter are wasted, because the sequence of the input is not sufficiently long the feed the whole filter
- As a consequence, the most practical solution is to choose $L = M$