# Lecture
# Adaptive Filters 1

# Let us note that

- We discussed Wiener filter and its application to linear prediction in last few lectures.
- Wiener filter is an LTI filter and it works on the assumption of WSS signals.
- The filter coefficients are determined from the knowledge of the autocorrelation and cross correlation functions.
- In practical situation, the signal is non-stationary. Under such circumstances, optimal filter should be time varying.

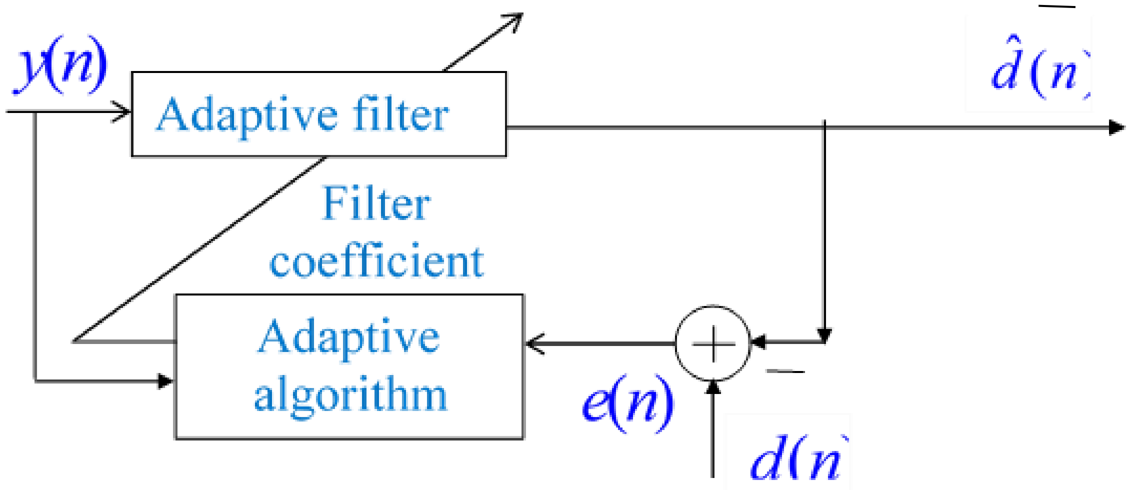
.

# How to tackle nonstationarity

- One way to tackle non-stationarity is to assume stationarity within certain data length. For example, in speech coding purpose, the signal is assumed to be WSS during a few milliseconds.
- The time-duration over which stationarity is a valid assumption, may be short so that accurate estimation of the model parameters is difficult.
- Another solution is ***adaptive filtering***. Here the filter coefficients are updated as a function of the filtering error using an adaptive algorithm.
- The adaptive algorithm updates filter coefficients based on the input signal and the other relevant information to obtain optimal performance
- This lecture will cover the basics of adaptive filters.

# General set-up for adaptive filtering

- The basic set-up for is as shown in the figure.




- The adaptation of filter coefficients is based on the error $e(n)$ between the filter output and a reference signal $d(n)$ usually called the *desired signal*. Choosing $d(n)$ is tricky- it depends on the specific application.

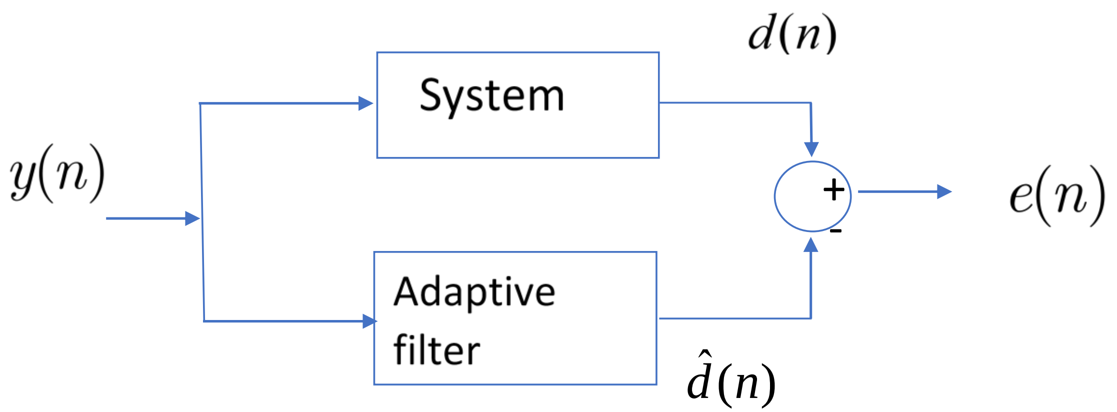
The adaptive filter may be FIR with a known filter length or IIR. The FIR filter ructure is normally used. The adaptive algorithm updates each filter coefficient dividually

# Applications

❖System identification

Used to obtain a linear model of the system

- A broadband signal $y(n)$, usually a white noise is input to both the system and adaptive filter. The output of the system is the desired signal $d(n)$.



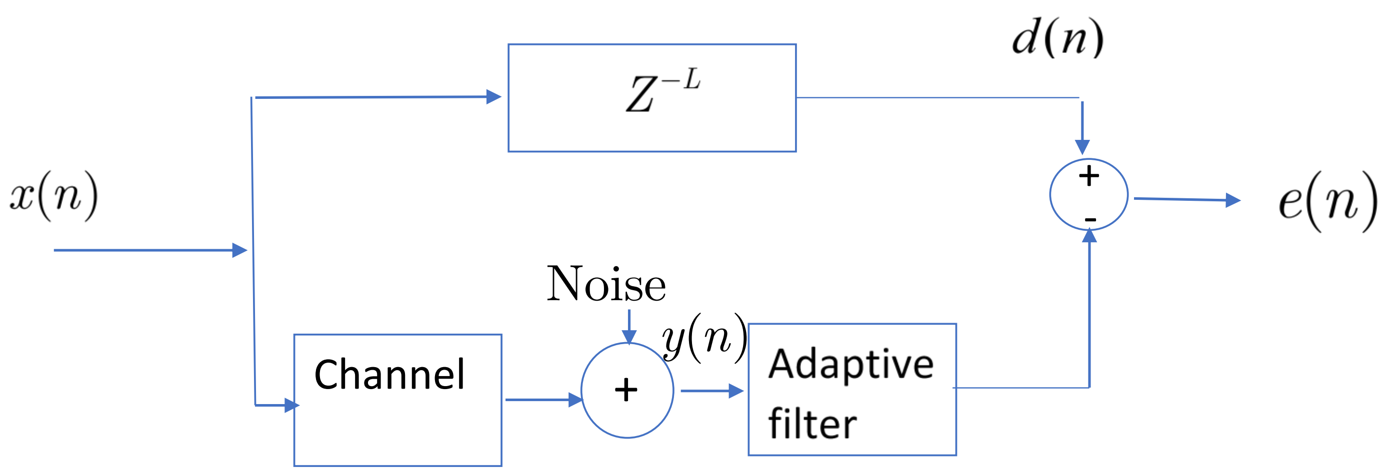

- At convergence, adaptive filter gives the linear model of the system

# Applications

❖ Channel equalization

Used to cancel the effect of the channel.

- A training signal is sent through the channel
- The delayed version of the training signal is the desired output *d*(*n*).




- At convergence, the adaptive filter cancels the effect of the channel.

# FIR Wiener filter and steepest descent

- Assume the signals to be WSS. Our goal is to estimate $d(n)$ using an FIR Wiener filter of length $M$ and the filter coefficients

  $h_i(n)$, i = 0,1, ... M-1.

- Represent the filter coefficients by the filter parameter vector

$$\boldsymbol{h}(n) = \begin{bmatrix} h_0(n) \\ h_1(n) \\ \vdots \\ h_{M-1}(n) \end{bmatrix}$$

- Our goal is to find $\boldsymbol{h}(n)$ by minimizing the mean-square error

$$Ee^2(n) = E(d(n) - \hat{d}(n))^2 = E(d(n) - \sum_{i=0}^{M-1} h_i(n) y(n-i))^2$$

# FIR Wiener filter and steepest descent …

- Representing the observed signals as a vector

$$\mathbf{y}(n) = \begin{bmatrix} y(n) \\ y(n-1) \\ \vdots \\ y(n-M+1) \end{bmatrix}, \text{ we get}$$

$$\begin{aligned} Ee^2(n) &= E(d(n) - \mathbf{h}'(n)\mathbf{y}(n))^2 \\ &= R_d(0) - 2\mathbf{h}'(n)\mathbf{r}_{d\mathbf{Y}} + \mathbf{h}'(n)\mathbf{R}_{\mathbf{Y}}\mathbf{h}(n) \end{aligned}$$

Where

$$\mathbf{r}_{d\mathbf{Y}} = \begin{bmatrix} R_{dY}(0) \\ R_{dY}(1) \\ \vdots \\ R_{dY}(M-1) \end{bmatrix} \text{ and } \mathbf{R}_{\mathbf{Y}} = \begin{bmatrix} R_Y(0) & R_Y(1) & \ldots. & R_Y(M-1) \\ R_Y(1) & R_Y(0) & \ldots. & R_Y(M-2) \\ \ldots & & & \\ R_Y(M-1) & R_Y(M-2) & \ldots. & R_Y(0) \end{bmatrix}$$

# FIR Wiener filter and steepest descent …

❖ The Wiener filtering problem can be written as

$$\text{Minimize } Ee^2(n) \qquad (1)$$

with respect to the filter coefficient vector $\boldsymbol{h}(n)$

❖ The cost function represented by $Ee^2(n)$ is a quadratic function in $\boldsymbol{h}(n)$ and a unique global minimum exists

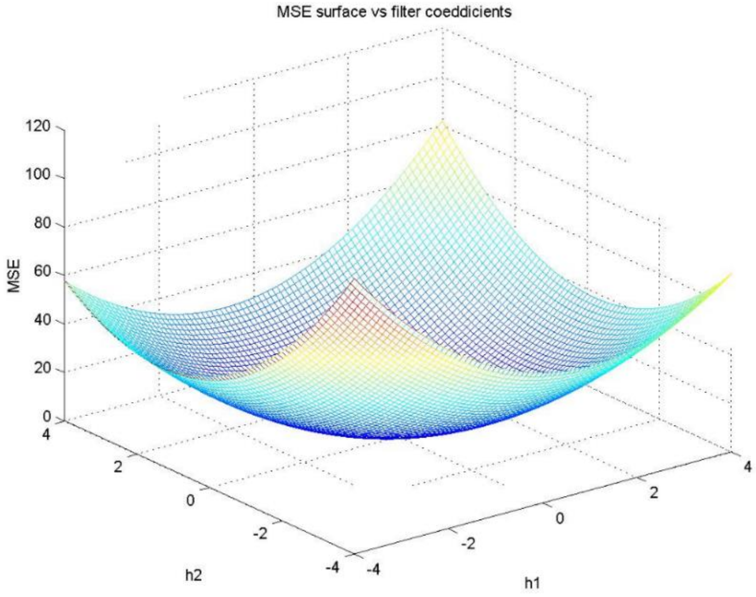



**Figure** - Cost Function $Ee^2(n)$ for a length 2 FIR Wiener filter

# FIR Wiener filter and steepest descent …

❖ The gradient of $Ee^2(n)$ is given by

$$\nabla Ee^2(n) = \begin{bmatrix} \frac{\partial Ee^2(n)}{\partial h_0} \\ \cdots\cdots\cdots \\ \cdots\cdots\cdots \\ \cdots\cdots \\ \frac{\partial Ee^2(n)}{\partial h_{M-1}} \end{bmatrix}$$

$$= -2\mathbf{r}_{d\mathbf{Y}} + +2\mathbf{R}_{\mathbf{Y}}\mathbf{h}(n)$$

❖ By setting $\nabla Ee^2(n) = 0$ we get the WH equations

$$\mathbf{R}_{\mathbf{Y}}\mathbf{h}_{\mathbf{opt}} = \mathbf{r}_{d\mathbf{Y}}$$

$$\therefore \mathbf{h}_{\mathbf{opt}} = \mathbf{R}_{\mathbf{Y}}^{-1}\mathbf{r}_{d\mathbf{Y}}$$

❖ Instead of analytical solution, the optimization problem in (1) can be solved iteratively. One of the iterative optimization algorithms is the ***steepest descent algorithm* (SDA)**. The most of the popular adaptation algorithms including machine learning, are based on the SDA.

# SDA iterations

- Since the gradient of a function points to the direction of maximum increase of the function, the negative of the gradient is the direction of maximum decrease of the function.
- Applying the SDA, the optimization problem in (1) can be solved by the following iterative relation:

$$\mathbf{h}(n+1)=\mathbf{h}(n)+\frac{\mu}{2}(-\nabla Ee^2(n))$$

where $\mu$ is the step-size parameter.

- So the steepest descent rule will now give

$$\mathbf{h}(n+1)=\mathbf{h}(n)+\mu(\mathbf{r}_{d\mathbf{Y}}-\mathbf{R}_{\mathbf{Y}}\mathbf{h}(n))$$

For a proper choice of $\mu$, the SDA solves the Wiener Hopf equation in a finite number of iterations.

## Convergence of the SDA

We have, $\mathbf{h}(n+1) = \mathbf{h}(n) + \mu(\mathbf{r}_{d\mathbf{Y}} - \mathbf{R}_{\mathbf{Y}}\mathbf{h}(n))$

$$= \mathbf{h}(n) - \mu \mathbf{R}_{\mathbf{Y}}\mathbf{h}(n) + \mu \mathbf{r}_{dY}$$

$$= (\mathbf{I} - \mu \mathbf{R}_{\mathbf{Y}})\mathbf{h}(n) + \mu \mathbf{r}_{dY}$$

where $\mathbf{I}$ is the $M \times M$ identity matrix.

$$\therefore \mathbf{h}(n+1) = (\mathbf{I} - \mu \mathbf{R}_{\mathbf{Y}})\mathbf{h}(n) + \mu \mathbf{r}_{dY} \qquad (2)$$

Expanding, we get

$$\therefore \begin{bmatrix} h_0(n+1) \\ h_1(n+1) \\ \vdots \\ h_{M-1}(n+1) \end{bmatrix} = \begin{bmatrix} 1-\mu R_Y(0) & \text{-}R_Y(1) & \ldots & \text{-}R_Y(M-1) \\ -R_Y(1) & 1\text{-}\mu R_Y(0) & \ldots & \text{-}R_Y(M-2) \\ \ldots & & & \\ -R_Y(M-1) & \text{-}R_Y(M-2) & \ldots & 1\text{-}\mu R_Y(0) \end{bmatrix} \begin{bmatrix} h_0(n) \\ h_1(n) \\ \vdots \\ h_{M-1}(n) \end{bmatrix} + \mu \begin{bmatrix} R_{dY}(0) \\ R_{dY}(1) \\ \ldots \\ R_{dY}(M-1) \end{bmatrix}$$

- Thus the SDA iteration is given by a coupled set of linear difference equations.

# Convergence of the SDA …

- $\mathbf{R_Y}$ is a symmetric non-singular matrix and can be diagonalized by the following similarity transform

$$\mathbf{R_Y} = \mathbf{Q\Lambda Q'}$$

where $\mathbf{Q}$ is the orthogonal matrix of the eigenvectors of $\mathbf{R_Y}$. $\mathbf{\Lambda}$ is a diagonal matrix with the corresponding eigen values as the diagonal elements.

- Also $\mathbf{I} = \mathbf{QQ'} = \mathbf{Q'Q}$

$$\therefore \mathbf{h}(n+1) = (\mathbf{QQ'} - \mu\mathbf{Q}\Lambda\mathbf{Q'})\mathbf{h}(n) + \mu\mathbf{r}_{dY}$$

- Multiply by $\mathbf{Q'}$

$$\mathbf{Q'h}(n+1) = (\mathbf{I} - \mu\mathbf{\Lambda})\mathbf{Q'h}(n) + \mu\mathbf{Q'r}_{d\mathbf{Y}}$$

# Convergence of the SDA …

- Define a new variable

$$\bar{\mathbf{h}}(n) = \mathbf{Q}'\mathbf{h}(n) \quad \text{and } \bar{\mathbf{r}}_{XY} = \mathbf{Q}'\mathbf{r}_{d\mathbf{Y}}$$

- Then

$$\bar{\mathbf{h}}(n+1) = (\mathbf{I} - \mu\mathbf{\Lambda})\bar{\mathbf{h}}(n) + \mu\bar{\mathbf{r}}_{d\mathbf{Y}}$$

$$= \begin{bmatrix} 1-\mu\lambda_1 & 0 \cdots\cdots & \cdots & .0 \\ 0 & & & \\ \vdots & & & \\ \vdots & & & \\ 0 & \ldots\ldots & \ldots & 1-\mu\lambda_M \end{bmatrix} \bar{\mathbf{h}}(n) + \mu\bar{\mathbf{r}}_{d\mathbf{Y}}$$

- This is a decoupled set of linear difference equations

$$\bar{h}_i(n+1) = (1-\mu\lambda_i)\bar{h}_i(n) + \mu\bar{r}_{dy}(i) \qquad i = 1, \ldots, M$$

and can be easily checked for convergence.

# Convergence of the SDA …

- The convergence condition is given by

$$|1-\mu\lambda_i| < 1$$

$$\Rightarrow -1 < 1-\mu\lambda_i < 1$$

$$\Rightarrow 0 < \mu < 2/\lambda_i, i = 1,\ldots,M$$

$$\Rightarrow 0 < \mu < 2/\lambda_{Max}$$

Thus, the condition for the convergence of the modified difference equation

$$\bar{h}_i(n+1) = (1-\mu\lambda_i)\bar{h}_i(n) + \mu\bar{r}_{dy}(i) \qquad i = 1,\ldots,M$$

is given by ,

$$0 < \mu < 2/\lambda_{Max}$$

Equivalently, the SDA iteration $\mathbf{h}(n+1) = (\mathbf{I}-\mu\mathbf{R}_\mathbf{Y})\mathbf{h}(n) + \mu\mathbf{r}_{dY}$ converges if

$$0 < \mu < 2/\lambda_{Max}$$

# Convergence of the SDA …

- **A simpler condition**

Note that all the eigen values of $\mathbf{R_Y}$ are positive.

- Let $\lambda_{\max}$ be the maximum eigen value. Then,

$$\lambda_{\max} < \lambda_1 + \lambda_2 + \ldots.. + \lambda_M$$
$$= \mathrm{Trace}(\mathbf{R_Y})$$

$$\therefore 0 < \mu < \frac{2}{Trace(\mathbf{R_{yy}})}$$
$$= \frac{2}{\mathrm{M.R_{YY}}(0)}$$

- The steepest decent algorithm converges to the corresponding Wiener filter

$$\lim_{n\to\infty} \mathbf{h}[n] = \mathbf{R}_{\mathbf{Y}}^{-1} \ \mathbf{r}_{d\mathbf{Y}}$$

if the step size $\mu$ is within the range of specified by the above relation.

# Rate of Convergence

- Considering the difference equation,

$$\mathbf{h}(n+1) = (\mathbf{I} - \mu\mathbf{R}_{\mathbf{Y}})\mathbf{h}(n) + \mu\mathbf{r}_{dY}$$

the rate of convergence depends on the eigen value spread for the autocorrelation matrix $\mathbf{R}_{\mathbf{Y}}$. This spread is expressed in terms of the condition number of $\mathbf{R}_{\mathbf{Y}}$, defined as $k = \dfrac{\lambda_{\max}}{\lambda_{\min}}$.

- The fastest convergence of this system of difference equations occurs when $k = 1$, corresponding to white noise.

# Example

- Suppose $\mathbf{R_Y} = \begin{bmatrix} 21 & 16 \\ 16 & 21 \end{bmatrix}$ and $r_{d\mathbf{Y}} = \begin{bmatrix} 20 \\ 16 \end{bmatrix}$. We want to determine a length-2 FIR Wiener filter using the SDA. Take $\mathbf{h}(0) = \begin{bmatrix} h_0(0) \\ h_1(0) \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$. The eigen values of $\mathbf{R_Y}$ are $\lambda_1 = 37$ and $\lambda_2 = 5$. We can choose $\mu = 0.02 < \dfrac{2}{37}$.

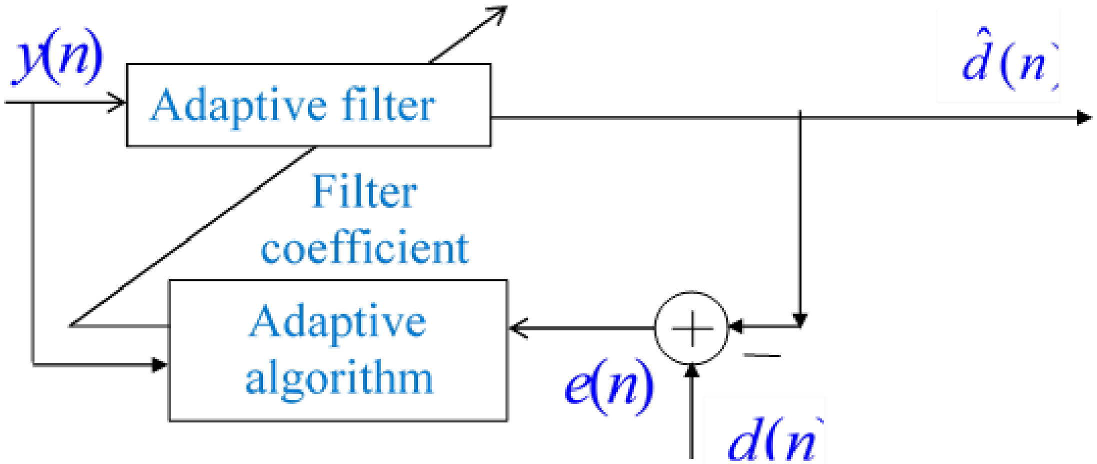
- Using $\mathbf{h}(n+1) = \mathbf{h}(n) + \mu(\mathbf{r}_{d\mathbf{Y}} - \mathbf{R_Y h}(n))$ , we get

$$\mathbf{h}(1) = \begin{bmatrix} h_0(1) \\ h_1(1) \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} + 0.02 \times \left( \begin{bmatrix} 20 \\ 16 \end{bmatrix} - \begin{bmatrix} 0 \\ 0 \end{bmatrix} \right) = \begin{bmatrix} 0.4 \\ 0.32 \end{bmatrix}$$

- Similarly $\mathbf{h}(3) = \begin{bmatrix} 0.5863 \\ 0.3695 \end{bmatrix}$, and after iterations we will get close to the WH solution $\mathbf{h} = \begin{bmatrix} 0.8865 \\ 0.0865 \end{bmatrix}$

# Summary

❖ The filter coefficients of an adaptive filter are updated based on the error $e(n)$ between the filter output and the desired signal $d(n)$ as shown in the figure.




❖**The cost function $Ee^2(n)$ for an FIR Wiener filter is a quadratic function in $\mathbf{h}(n)$ and a unique global minimum exists .**

The optimal set of filter parameters can be found by the SDA iteration:

$$\mathbf{h}(n+1) = \mathbf{h}(n) + \frac{\mu}{2}(-\nabla Ee^2(n))$$

Under WSS assumption

$$\mathbf{h}(n+1) = \mathbf{h}(n) + \mu(\mathbf{r}_{d\mathbf{Y}} - \mathbf{R}_{\mathbf{Y}}\mathbf{h}(n))$$

# LMS algorithm (Least Mean Square) algorithm

- Consider the steepest descent relation

$$\mathbf{h}(n+1) = \mathbf{h}(n) - \frac{\mu}{2}\nabla \mathbf{E}e^2(n)$$

where

$$\nabla \mathbf{E}e^2(n) = \begin{bmatrix} \frac{\partial E e^2(n)}{\partial h_0} \\ \cdots\cdots\cdots \\ \cdots\cdots\cdots \\ \cdots\cdots\cdots \\ \frac{\partial E e^2(n)}{\partial h_{M-1}} \end{bmatrix}$$

# LMS algorithm…

- In the LMS algorithm $Ee^2(n)$ is approximated by $e^2(n)$ to achieve a computationally simple algorithm.

$$\nabla \mathbf{E} e^2(n) \cong 2.e(n).\begin{bmatrix} \dfrac{\partial e(n)}{\partial h_0} \\ \dots\dots\dots \\ \dots\dots\dots \\ \dots\dots\dots \\ \dfrac{\partial e(n)}{\partial h_{M-1}} \end{bmatrix}$$

- Now consider

$$e(n) = d(n) - \sum_{i=0}^{M-1} h_i(i)\, y(n-i)$$

$$\frac{\partial e(n).}{\partial h_j} = -y(n-j),\ j = 0,1,\dots\dots M-1$$

# LMS algorithm…

$$\therefore \begin{bmatrix} \dfrac{\partial e(n)}{\partial h_0} \\ \cdots\cdots\cdots \\ \cdots\cdots\cdots \\ \dfrac{\partial e(n)}{\partial h_{M-1}} \end{bmatrix} = -\begin{bmatrix} y(n) \\ y(n-1) \\ \cdots\cdots\cdots\cdots\cdots \\ \cdots\cdots\cdots\cdots\cdots \\ \cdots\cdots\cdots\cdots \\ y(n-M+1) \end{bmatrix} = -\mathbf{y}(n)$$

$$\therefore \nabla \mathbf{Ee}^2(n) \cong -\mathbf{2}e(n)\mathbf{y}(n)$$

- The steepest descent update now becomes

$$\mathbf{h(n+1)} = \mathbf{h(n)} + \mu e(n)\mathbf{y(n)}$$

- This modification is due to Widrow and Hopf and the corresponding adaptive filter is known as the ***LMS filter*.**

# LMS algorithm steps

❖ Given the input signal $y[n]$, reference signal $x(n)$ and step size $\mu$

1. Initialization $h_i(0) = 0, i = 0,1,2.....M-1$

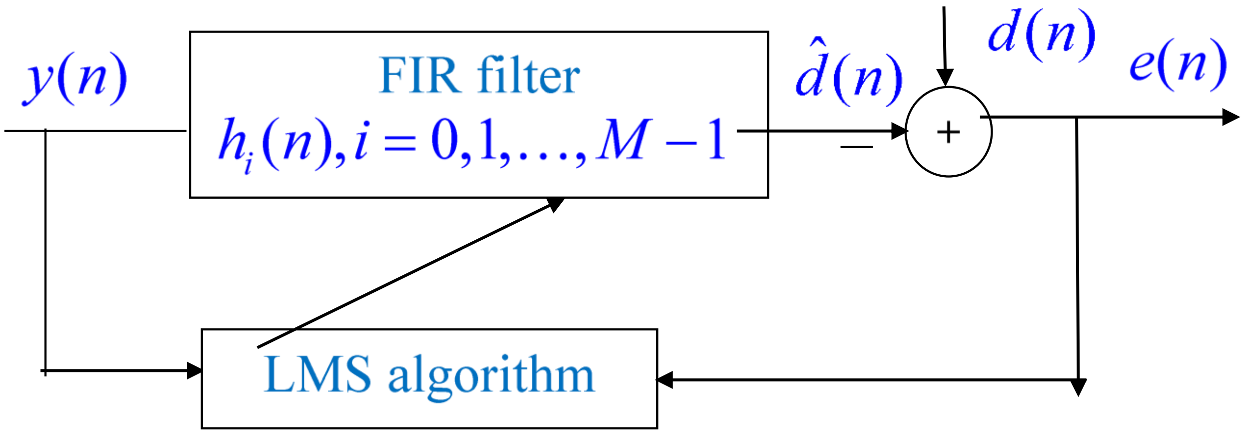
2. For $n > 0$

   Filter output $\hat{d}(n) = \mathbf{h}'(n)\mathbf{y}(n)$

   Estimation of the error $e(n) = d(n) - \hat{x}(n)$

3. Tap weight adaptation

$$\mathbf{h(n+1)} = \mathbf{h(n)} + \mu e(n)\mathbf{y(n)}$$



# THANK YOU