ESE 2023: Digital Signal Processing

Lecture 22: April 11, 2023

Adaptive Filters

Example 2:

A sequence $x = \{x[n], n = 0,1,\ldots,N-1\}$ is given; let $X(e^{j\omega})$ be its DTFT.

(a) Suppose $N = 10$. You want to evaluate both $X(e^{j2\pi 7/12})$ and $X(e^{j2\pi 3/8})$. The only computation you can perform is one DFT, on any one input sequence of your choice. Can you find the desired DTFT values? *(Show your analysis and explain clearly.)*

Example 2:

A sequence $x = \{x[n], n = 0,1,\ldots,N-1\}$ is given; let $X(e^{j\omega})$ be its DTFT.

(b) Suppose N is large. You want to obtain $X(e^{j\omega})$ at the following $2M$ frequencies:

$$\omega = \frac{2\pi}{M}m, \quad m = 0,1,\ldots,M-1 \quad \text{and} \quad \omega = \frac{2\pi}{M}m + \frac{2\pi}{N}, \quad m = 0,1,\ldots,M-1.$$

Here $M = 2^{\mu} \ll N = 2^{\nu}$

A standard radix-2 FFT algorithm is available. You may execute the FFT algorithm *once* or *more than once*, and *multiplications* and *additions* outside of the FFT are *allowed*, if necessary.

(i) You want to get the $2M$ DTFT values with as few *total multiplications* as possible (*including those in the FFT*). Give explicitly the best method you can find for this, with an estimate of the *total number of multiplications* needed in terms of M and N.

Example 2:

A sequence $x = \{x[n], n = 0,1,\ldots,N-1\}$ is given; let $X(e^{j\omega})$ be its DTFT.

(b) Suppose N is large. You want to obtain $X(e^{j\omega})$ at the following $2M$ frequencies:

$$\omega = \frac{2\pi}{M}m, \quad m = 0,1,\ldots,M-1 \quad \text{and} \quad \omega = \frac{2\pi}{M}m + \frac{2\pi}{N}, \quad m = 0,1,\ldots,M-1.$$

Here $M = 2^{\mu} \ll N = 2^{\nu}$

A standard radix-2 FFT algorithm is available. You may execute the FFT algorithm *once* or *more than once*, and *multiplications* and *additions* outside of the FFT are *allowed*, if necessary.

(i) You want to get the $2M$ DTFT values with as few *total multiplications* as possible (*including those in the FFT*). Give explicitly the best method you can find for this, with an estimate of the *total number of multiplications* needed in terms of M and N.

(ii) Does your result change if extra multiplications outside of FFTs are *not* allowed?

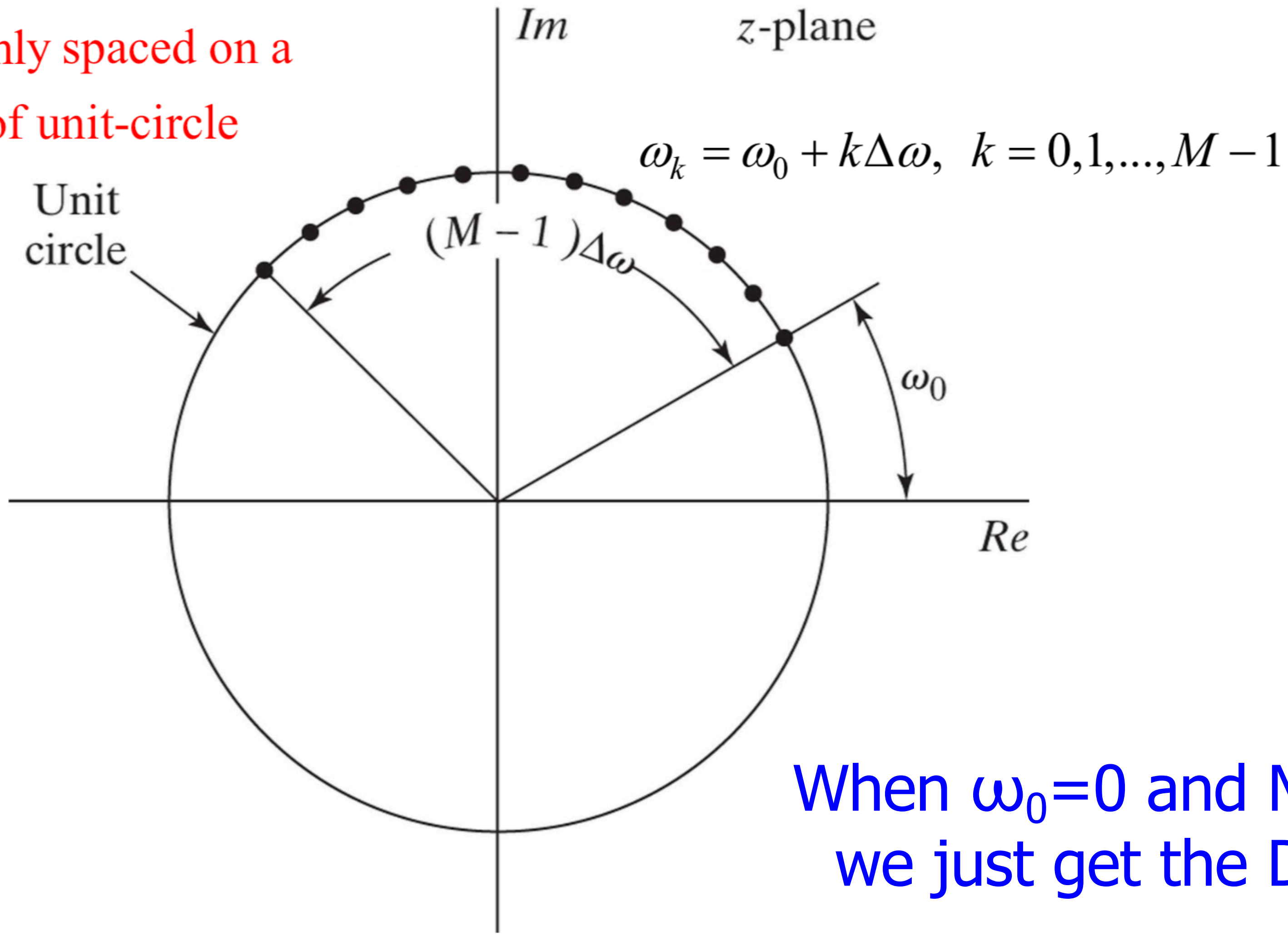
Chirp Transform Algorithm

Chirp Transform Algorithm

- Uses convolution to evaluate the DFT
- This algorithm is not optimal in minimizing any measure of computational complexity, but it has been useful in a variety of applications, particularly when implemented in technologies that are well suited to doing convolution with a fixed, pre-specified impulse response.
- The CTA is also more flexible than the FFT, since it can be used to compute *any* set of equally spaced samples of the Fourier transform on the unit circle.

Chirp Transform Algorithm

For M points of DTFT uniformly spaced on a sector of unit-circle



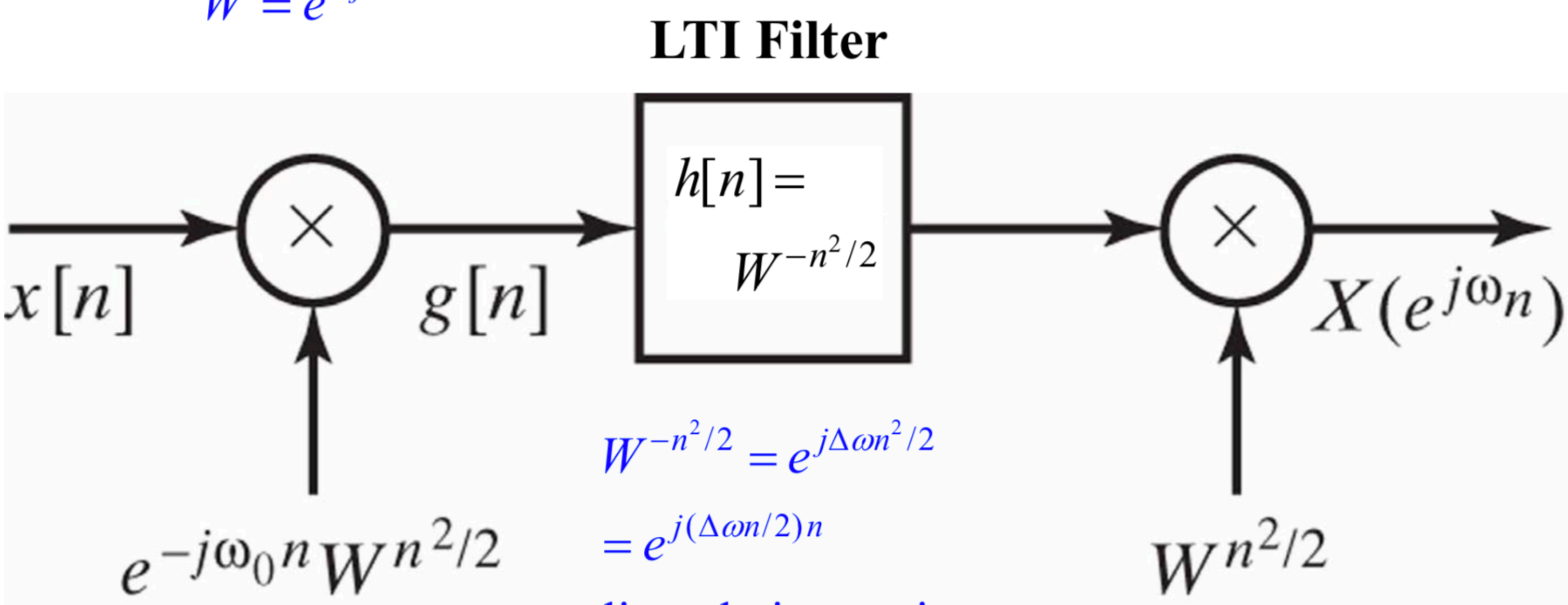
When ω_0=0 and M=N, we just get the DFT

Chirp Transform Algorithm

$$X(e^{j\omega_k}) = \sum_{n=0}^{N-1} x[n] e^{-j\omega_0 n} W^{nk} \qquad \forall\ k = 0, \ldots, M-1$$

Chirp Transform Algorithm

$W \triangleq e^{-j\Delta\omega}$



$W^{-n^2/2} = e^{j\Delta\omega n^2/2}$

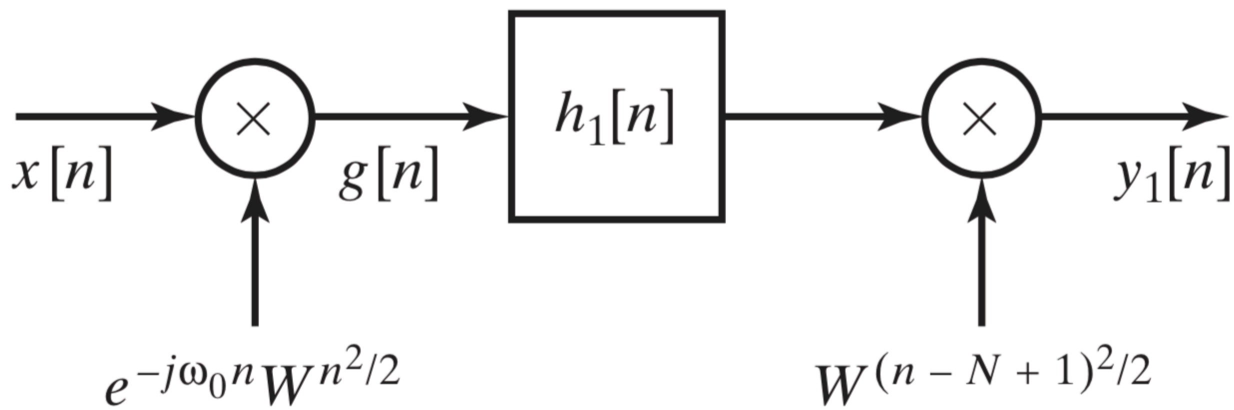
$= e^{j(\Delta\omega n/2)n}$

linearly increasing

frequency "chirp"

Causal FIR CTA

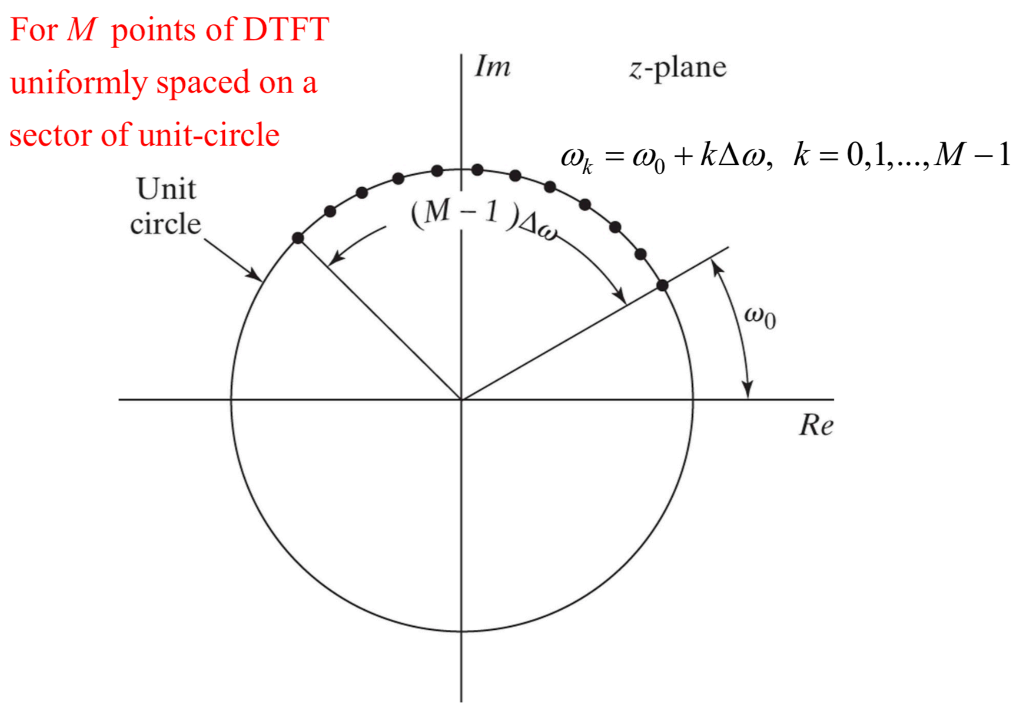
$$h_1[n] = \begin{cases} W^{-(n-N+1)^2/2}, & n = 0, 1, \ldots, M + N - 2, \\ 0, & \text{otherwise.} \end{cases}$$



$$X(e^{j\omega_n}) = y_1[n + N - 1], \qquad n = 0, 1, \ldots, M - 1.$$

Example: Chirp Transform Parameters

- We have a finite-length sequence x[n] that is nonzero only on the interval n = 0, ..., 25, (Length N=26) and we wish to compute 16 samples of the DTFT $X(e^{j\omega})$ at the frequencies $\omega_k = 2\pi/27 + 2\pi k/1024$ for k = 0, . . . , 15.

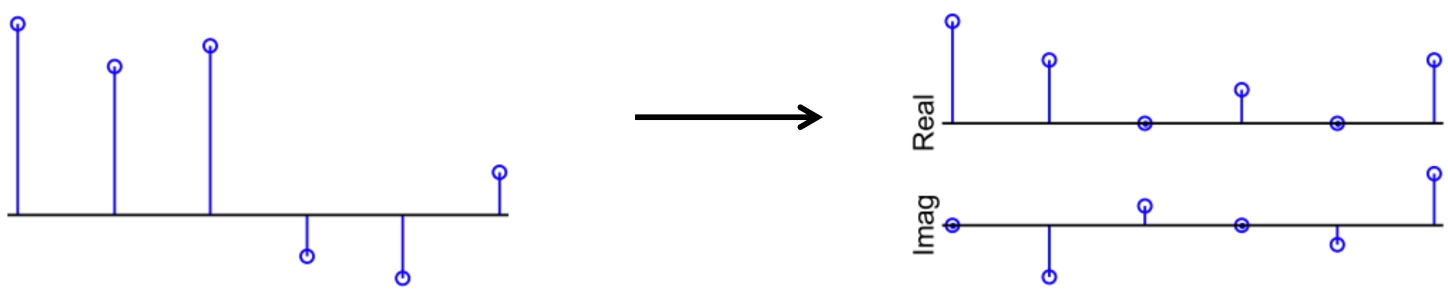


Discrete Cosine Transform

- Similar to the discrete Fourier transform (DFT), but using only real numbers
- Widely used in lossy compression applications (eg. Mp3, JPEG)
- Why use it?

DFT Problems

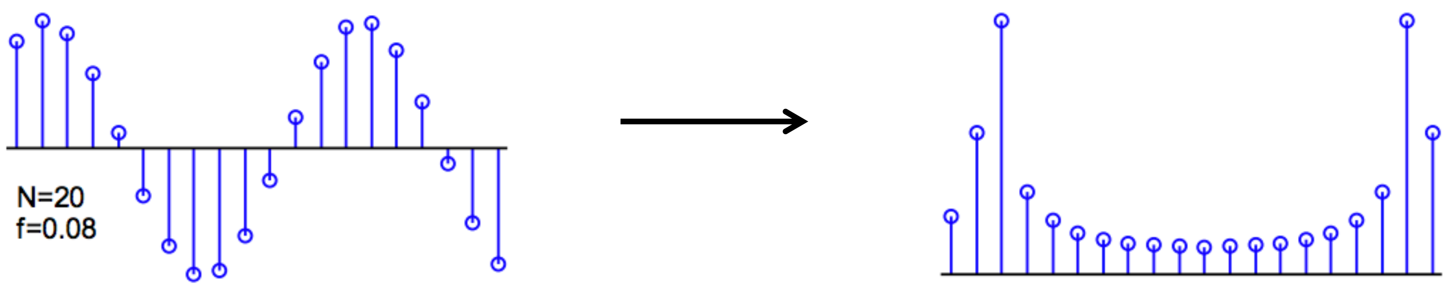
- For processing 1-D or 2-D signals (especially coding), a common method is to divide the signal into “frames” and then apply an invertible transform to each frame that compresses the information into few coefficients.
- The DFT has some problems when used for this purpose:
  - N real x[n] ↔ N complex X[k] : 2 real, N/2 − 1 conjugate pairs
  - DFT is of the periodic signal formed by replicating x[n]



DFT Problems

- For processing 1-D or 2-D signals (especially coding), a common method is to divide the signal into "frames" and then apply an invertible transform to each frame that compresses the information into few coefficients.
- The DFT has some problems when used for this purpose:
  - N real x[n] ↔ N complex X[k] : 2 real, $N/2 - 1$ conjugate pairs
  - DFT is of the periodic signal formed by replicating x[n]

    ⇒ Spurious frequency components from boundary discontinuity



The Discrete Cosine Transform (DCT) overcomes these problems.

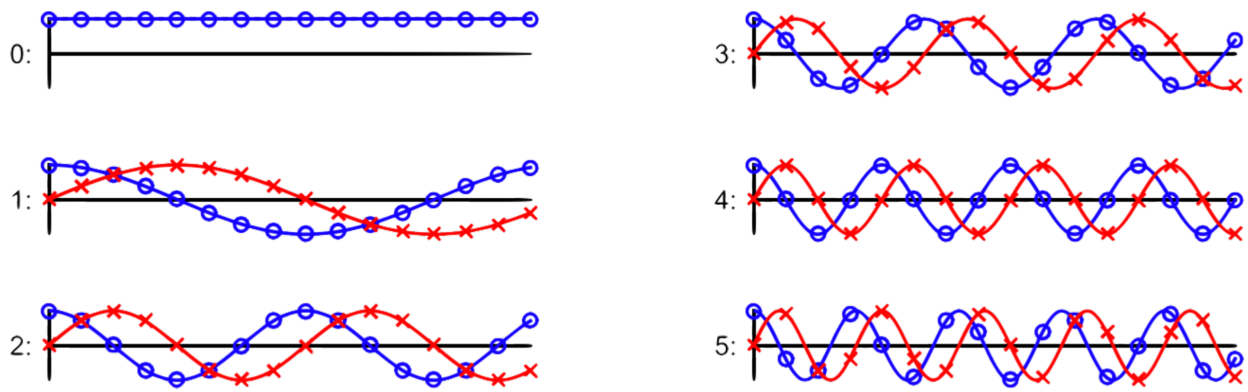
Discrete Cosine Transform

Forward DCT: $X_C[k] = \sum_{n=0}^{N-1} x[n] \cos \frac{2\pi(2n+1)k}{4N}$ for $k = 0 : N-1$

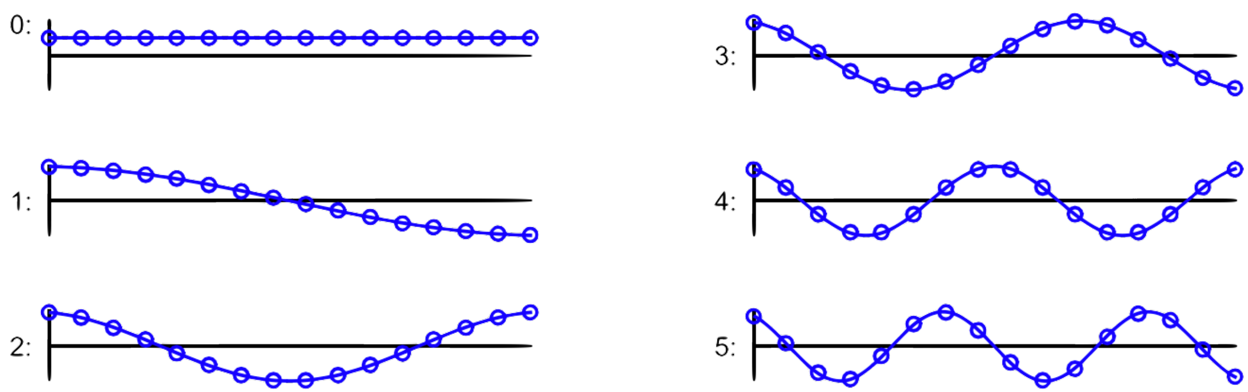
Inverse DCT: $x[n] = \frac{1}{N} X[0] + \frac{2}{N} \sum_{k=1}^{N-1} X[k] \cos \frac{2\pi(2n+1)k}{4N}$

Basis Functions

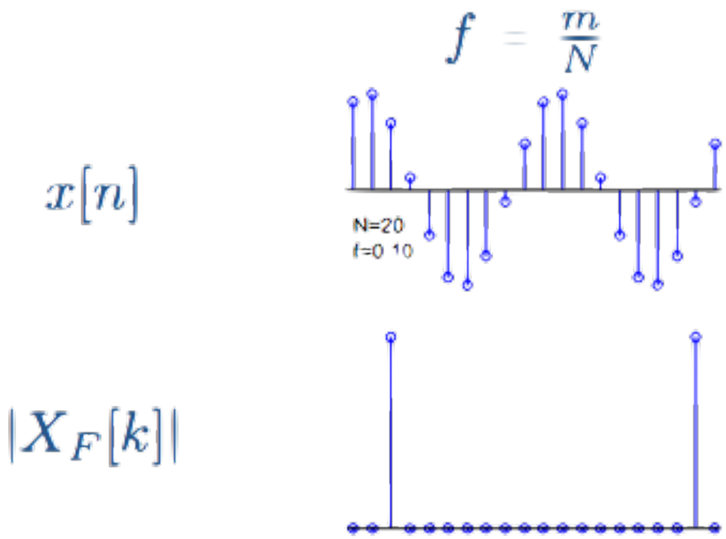
DFT basis functions: $x[n] = \frac{1}{N}\sum_{k=0}^{N-1} X[k]e^{j2\pi\frac{kn}{N}}$



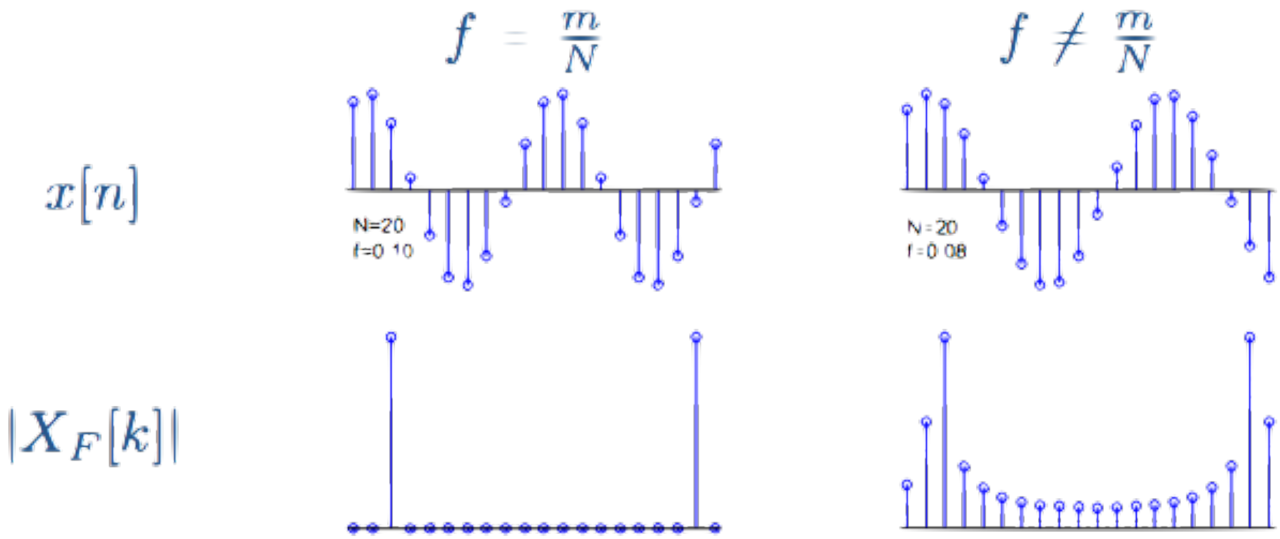
DCT basis functions: $x[n] = \frac{1}{N}X[0] + \frac{2}{N}\sum_{k=1}^{N-1} X[k]\cos\frac{2\pi(2n+1)k}{4N}$



DFT of Sine Wave

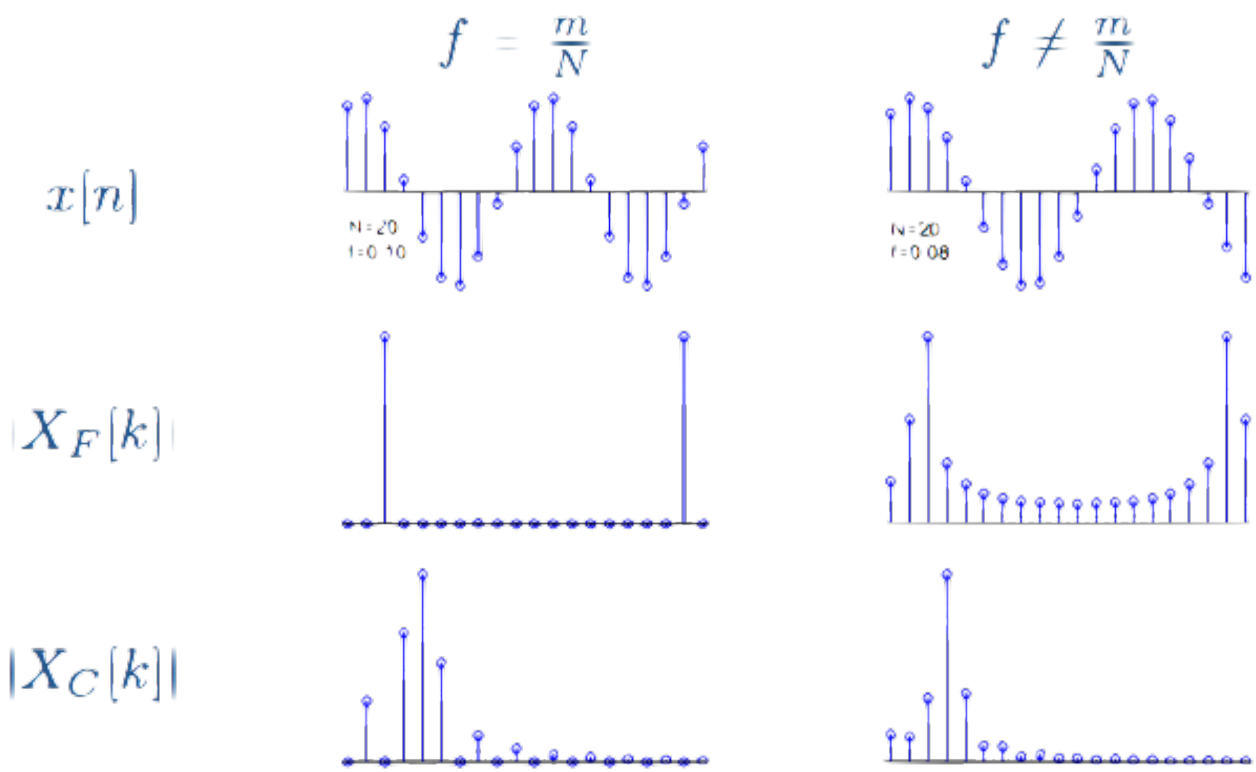


DFT of Sine Wave



DFT: Real→Complex; Freq range $[0, 1]$; Poorly localized unless $f = \frac{m}{N}$; $|X_F[k]| \propto k^{-1}$ for $Nf < k \ll \frac{N}{2}$

DCT of Sine Wave



DFT: Real→Complex; Freq range $[0, 1]$; Poorly localized unless $f = \frac{m}{N}$; $|X_F[k]| \propto k^{-1}$ for $Nf < k \ll \frac{N}{2}$

DCT: Real→Real; Freq range $[0, 0.5]$; Well localized $\forall f$; $|X_C[k]| \propto k^{-2}$ for $2Nf < k < N$

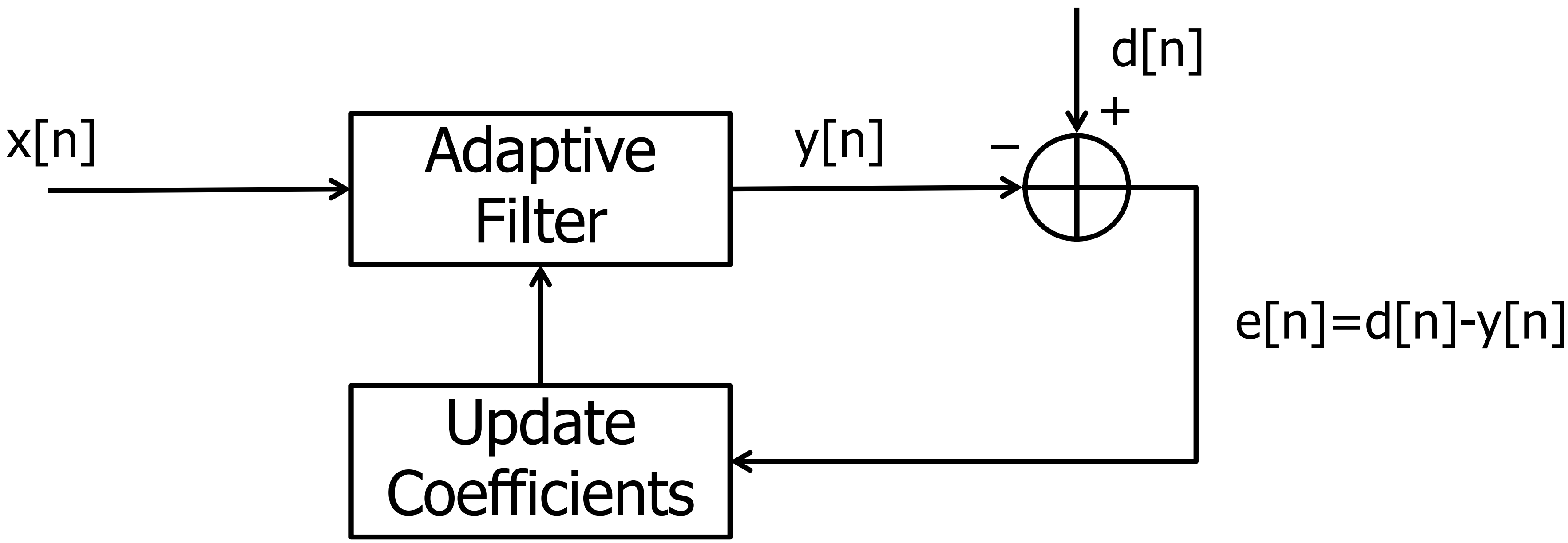
Adaptive Filters

Application Areas

- Speech coding
- Speech enhancement (hands-free systems, hearing aids, public address systems)
- Equalization (sending antennas, radar, loudspeakers)
- Anti-noise systems (cars and airplanes)
- Multi-channel signal processing (beamforming, submarine localization, layer of earth analysis)
- Missile control
- Medical applications (fetal heart rate monitoring, dialysis)
- Processing of video signals (cancellation of distortions, image analysis)
- Antenna arrays

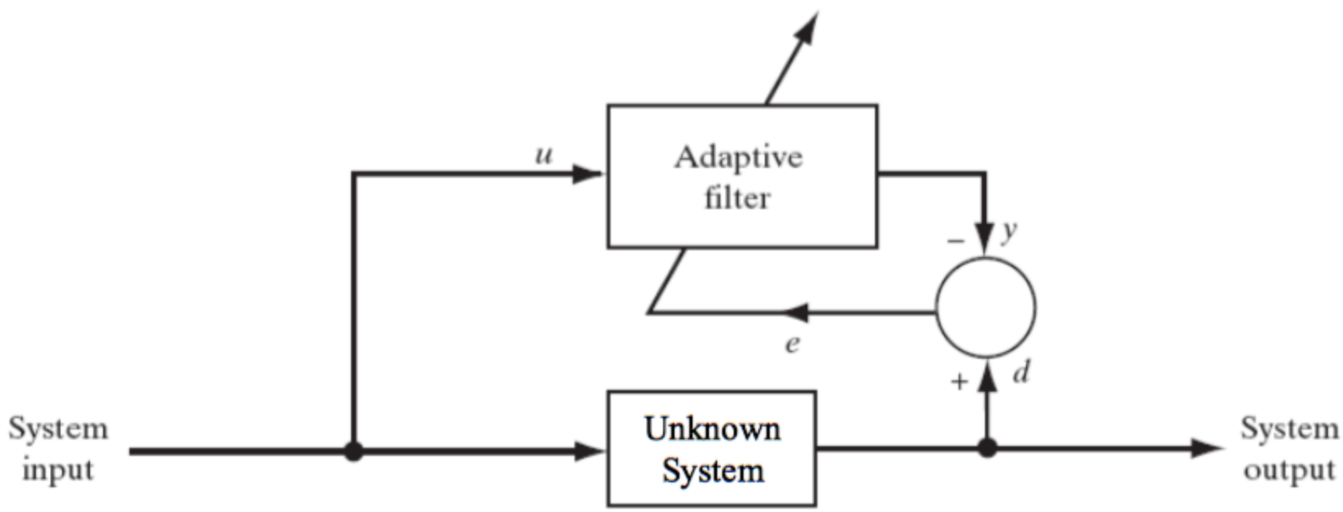
Adaptive Filters

- An adaptive filter is an adjustable filter that processes in time
  - It adapts…



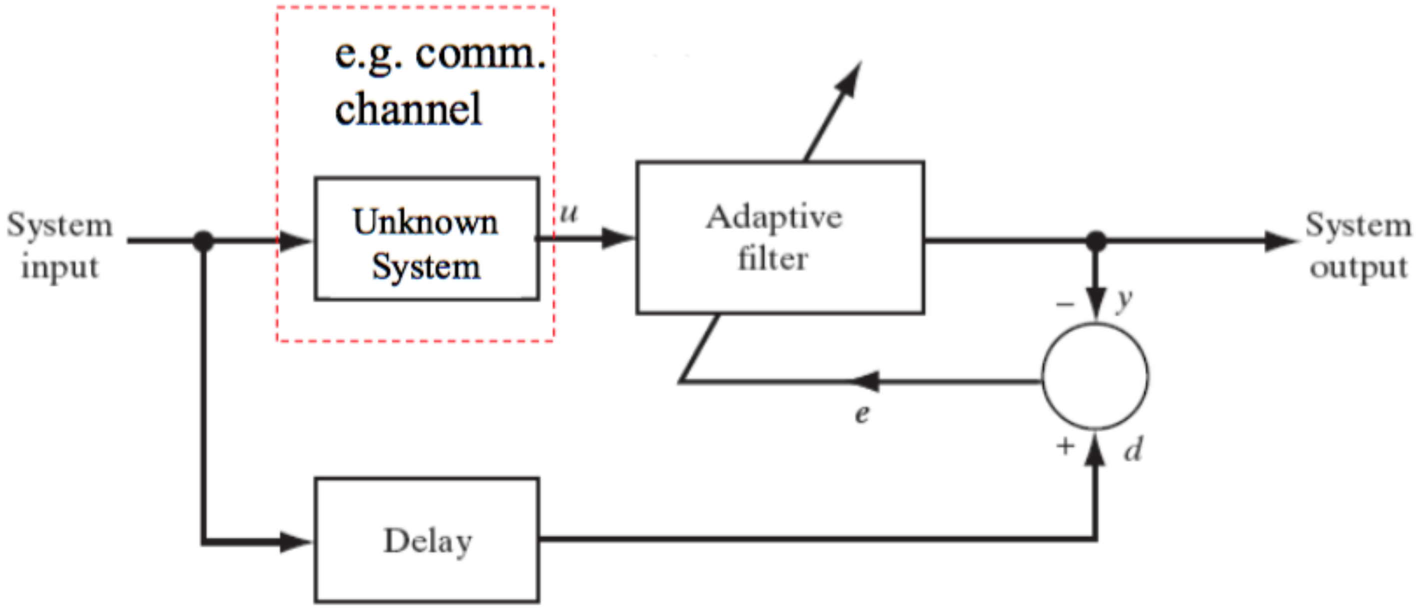
Adaptive Filter Applications

- System Identification



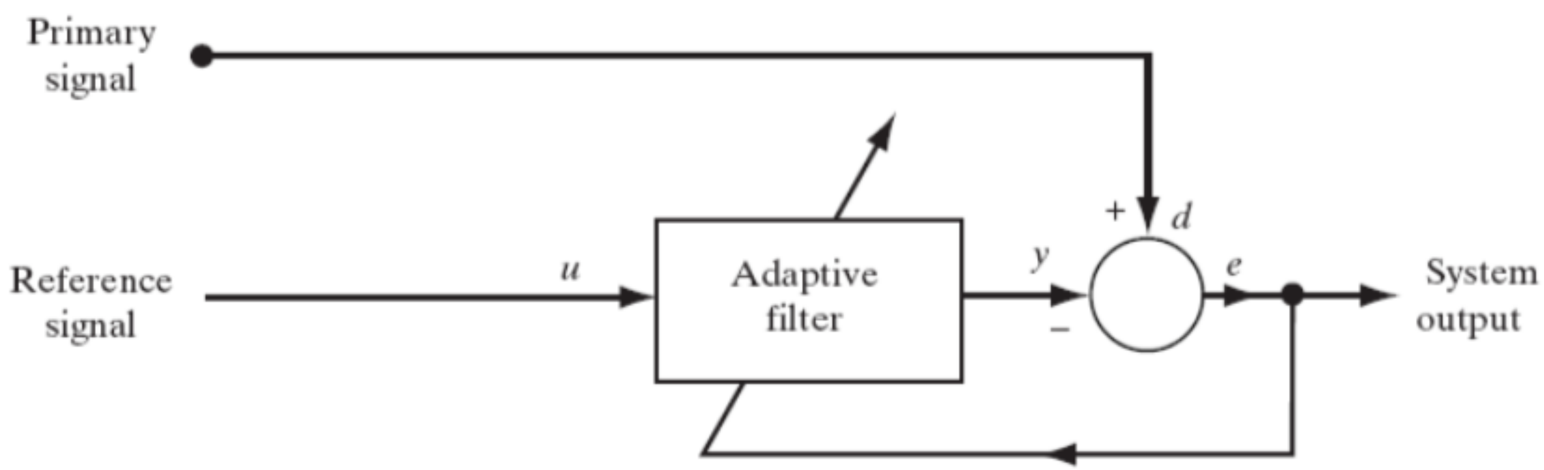
Adaptive Filter Applications

- Identification of inverse system

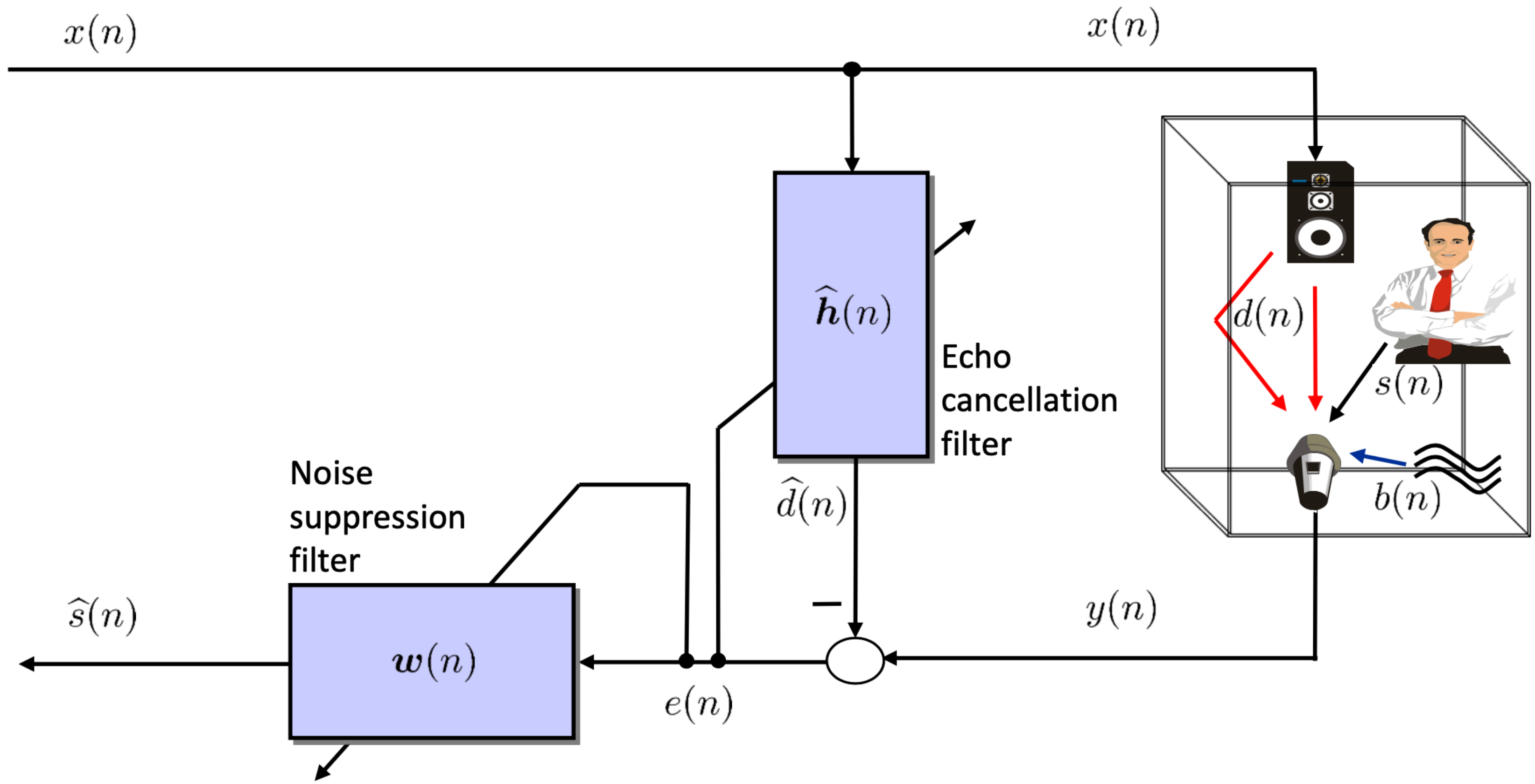


Adaptive Filter Applications

- Adaptive Interference Cancellation

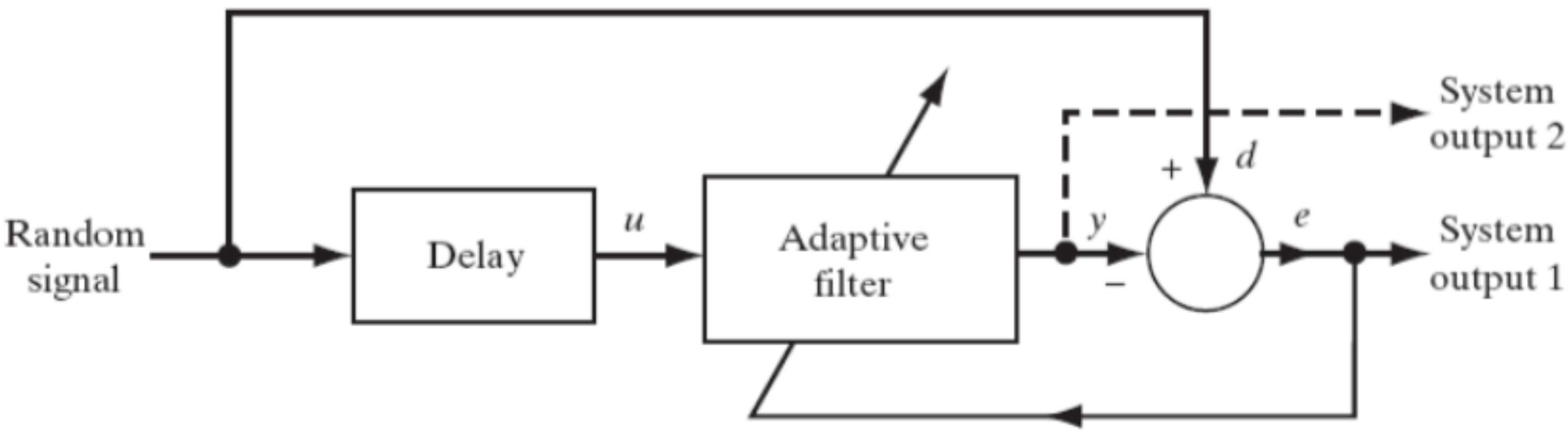


Automotive Hands-Free System



Adaptive Filter Applications

- Adaptive Prediction

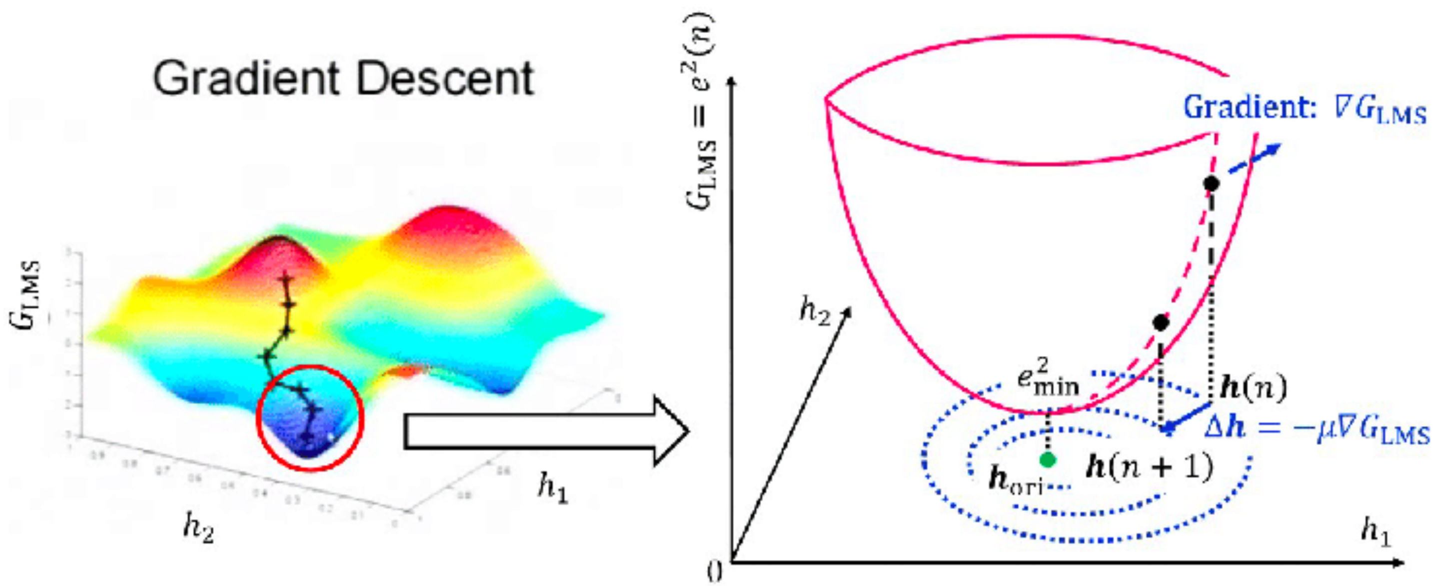


Stochastic Gradient Approach

- Most commonly used type of Adaptive Filters
- Define cost function as mean-squared error
    - Eg. Difference between filter output and desired response
- Based on the method of steepest descent
  - Move towards the minimum on the error surface to get to minimum
  - Requires the gradient of the error surface to be known

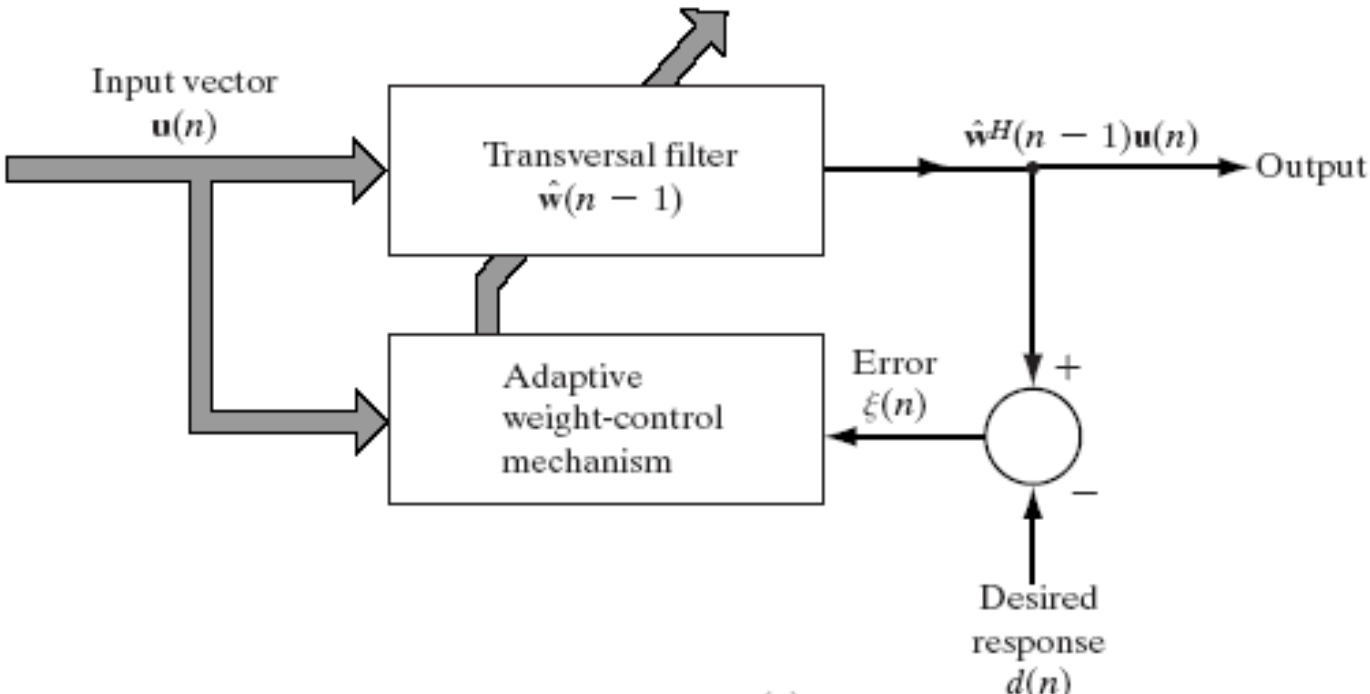
Stochastic Gradient Approach

- Most commonly used type of Adaptive Filters
- Define cost function as mean-squared error
    - Eg. Difference between filter output and desired response
- Based on the method of steepest descent
    - Move towards the minimum on the error surface to get to minimum
    - Requires the gradient of the error surface to be known



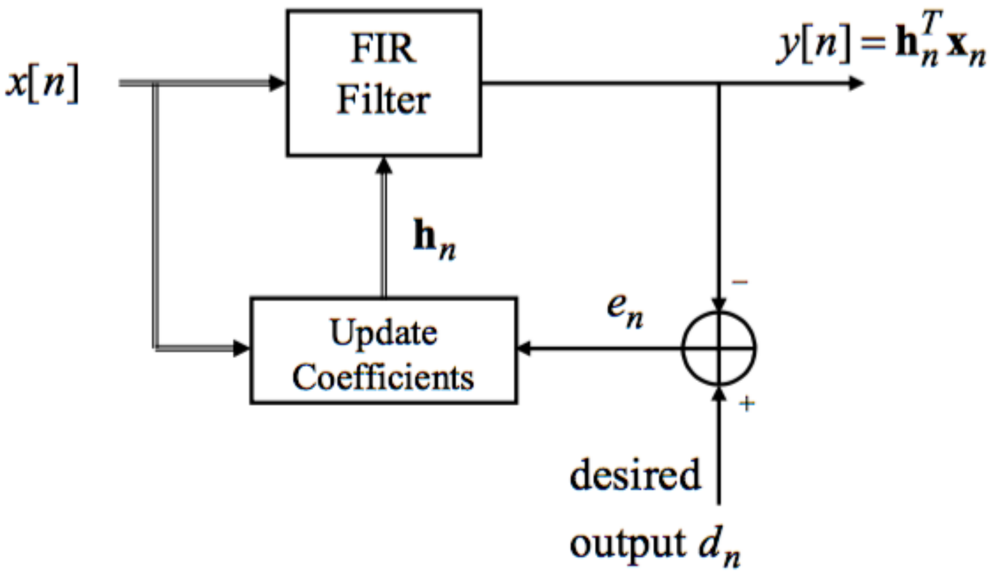
Least-Mean-Square (LMS) Algorithm

- The LMS Algorithm consists of two basic processes
  - Filtering process
    - Calculate the output of FIR filter by convolving input and taps
    - Calculate estimation error by comparing the output to desired signal
  - Adaptation process
    - Adjust tap weights based on the estimation error



(a)

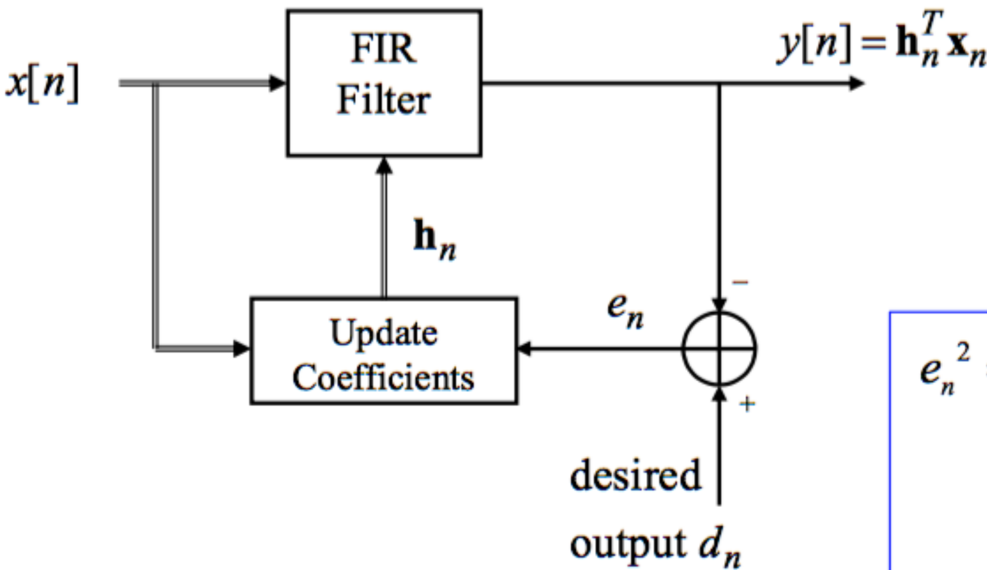
Adaptive FIR Filter: LMS



$$\mathbf{x}_n = (x[n], x[n-1]\ldots, x[n-M])^T$$

$$\mathbf{h}_n = (h_n[0], h_n[1], \ldots, h_n[M])^T$$

Adaptive FIR Filter: LMS

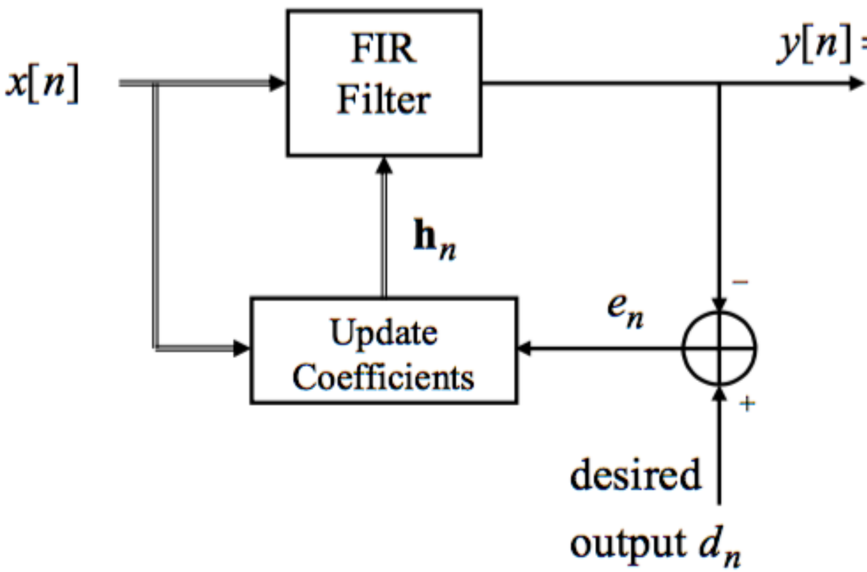


$$\mathbf{x}_n = (x[n], x[n-1] \ldots, x[n-M])^T$$

$$\mathbf{h}_n = (h_n[0], h_n[1], \ldots, h_n[M])^T$$

$$e_n^{\ 2} = (d[n] - y[n])^2 = (d[n] - \mathbf{h}_n^T \mathbf{x}_n)^2$$

Adaptive FIR Filter: LMS



$$\mathbf{x}_n = (x[n], x[n-1] \ldots, x[n-M])^T$$

$$\mathbf{h}_n = (h_n[0], h_n[1], \ldots, h_n[M])^T$$

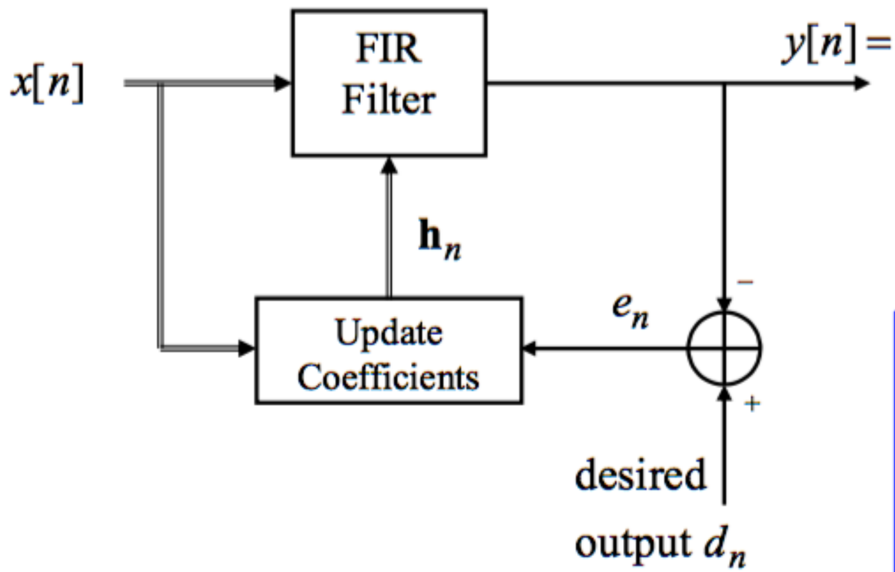
$$e_n^{\,2} = (d[n] - y[n])^2 = (d[n] - \mathbf{h}_n^T \mathbf{x}_n)^2$$

$$\frac{d\, e_n^{\,2}}{d\,\mathbf{h}_n} = -2(d[n] - \mathbf{h}_n^T \mathbf{x}_n)\mathbf{x}_n = -2 e_n \mathbf{x}_n$$

$$\bar{a}^T \bar{b} = \begin{bmatrix} a_1 & a_2 & a_3 & a_4 \end{bmatrix} \begin{bmatrix} b_1 \\ b_2 \\ b_3 \\ b_4 \end{bmatrix} = a_1 b_1 + a_2 b_2 + a_3 b_3 + a_4 b_4$$

$$\frac{\partial\left(a^T b\right)}{\partial \bar{a}} = \begin{bmatrix} \frac{\partial\left(a^T b\right)}{\partial a_1} & \frac{\partial\left(a^T b\right)}{\partial a_2} & \frac{\partial\left(a^T b\right)}{\partial a_3} & \frac{\partial\left(a^T b\right)}{\partial a_4} \end{bmatrix}^T = \begin{bmatrix} b_1 \\ b_2 \\ b_3 \\ b_4 \end{bmatrix} = \bar{b}$$

Adaptive FIR Filter: LMS



$y[n] = \mathbf{h}_n^T \mathbf{x}_n$

$\mathbf{x}_n = (x[n], x[n-1] \ldots, x[n-M])^T$

$\mathbf{h}_n = (h_n[0], h_n[1], \ldots, h_n[M])^T$

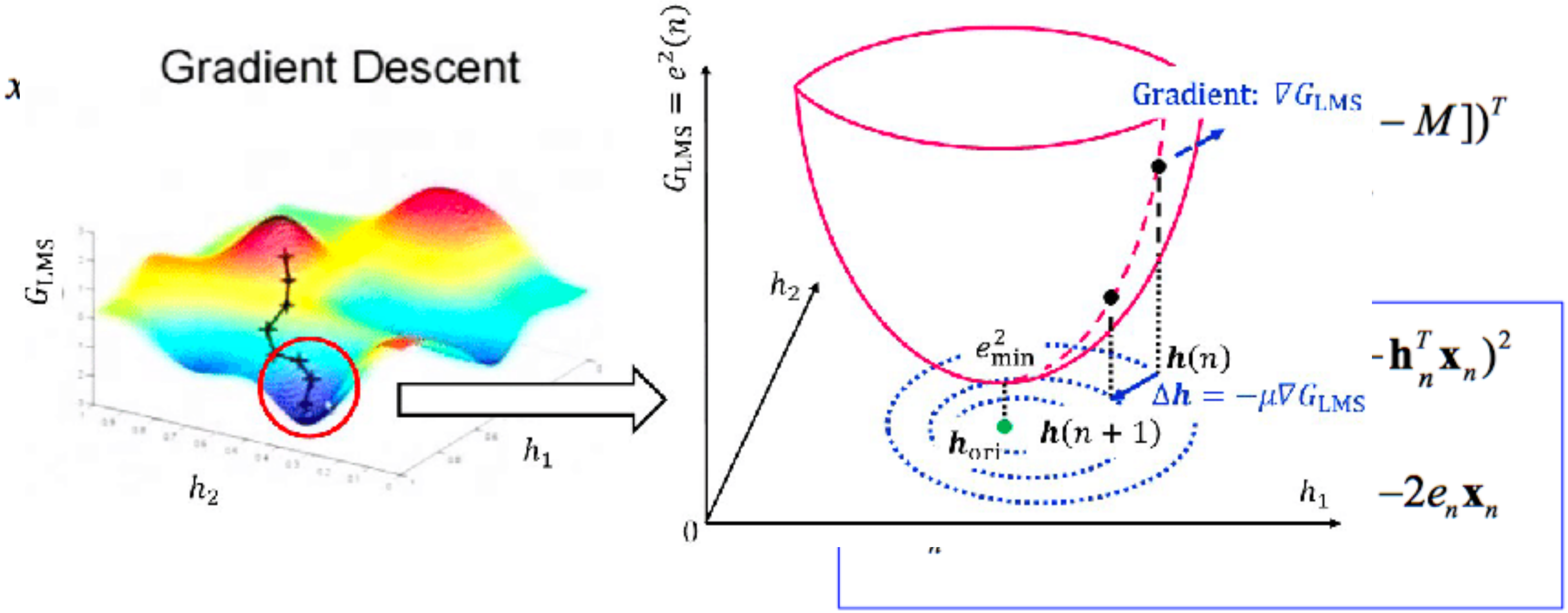
$$e_n^2 = (d[n] - y[n])^2 = (d[n] - \mathbf{h}_n^T \mathbf{x}_n)^2$$

$$\frac{d\, e_n^2}{d\,\mathbf{h}_n} = -2(d[n] - \mathbf{h}_n^T \mathbf{x}_n)\mathbf{x}_n = -2 e_n \mathbf{x}_n$$

Coefficient Update: Move in direction *opposite* to sign of gradient, proportional to magnitude of gradient $\mathbf{h}_{n+1} = \mathbf{h}_n + 2\mu e_n \mathbf{x}_n$

Stochastic Gradient Algorithm

Adaptive FIR Filter: LMS



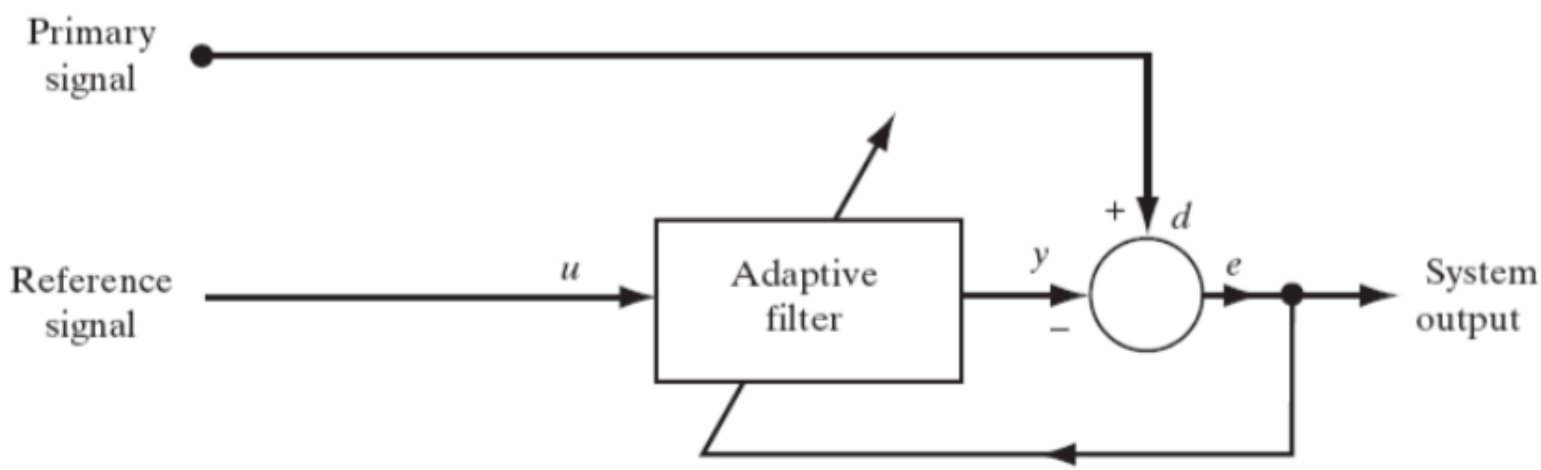
Coefficient Update: Move in direction *opposite* to sign of gradient, proportional to magnitude of gradient

$$\mathbf{h}_{n+1} = \mathbf{h}_n + 2\mu e_n \mathbf{x}_n$$

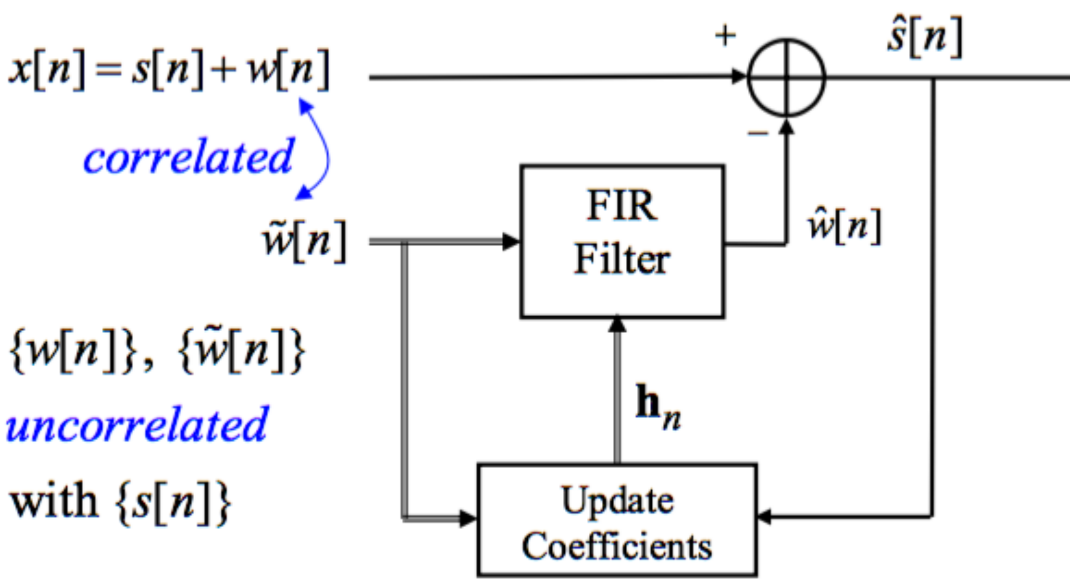
Stochastic Gradient Algorithm

Adaptive Filter Applications

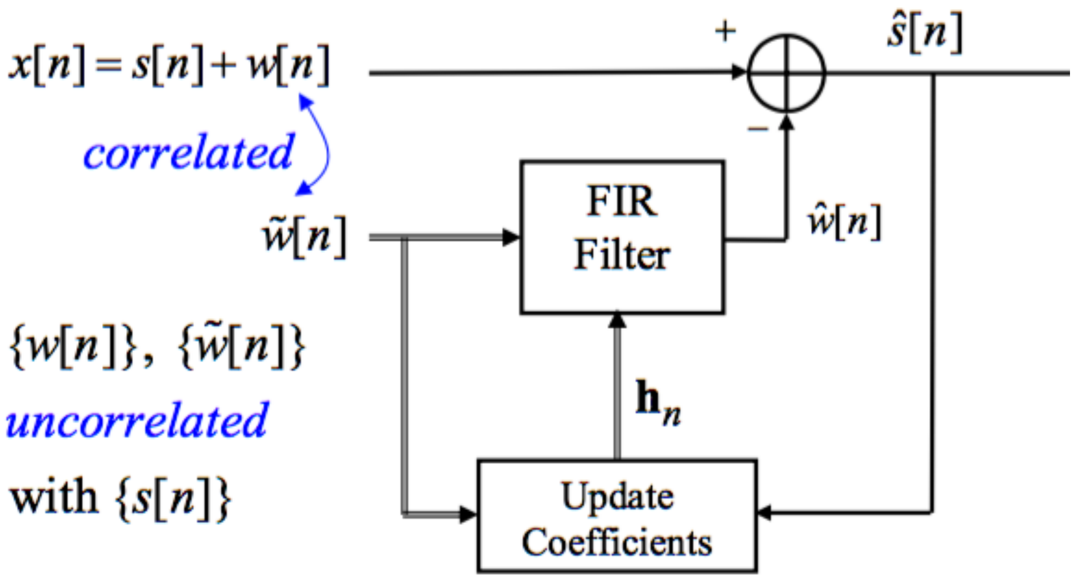
- Adaptive Interference Cancellation



Adaptive Interference Cancellation

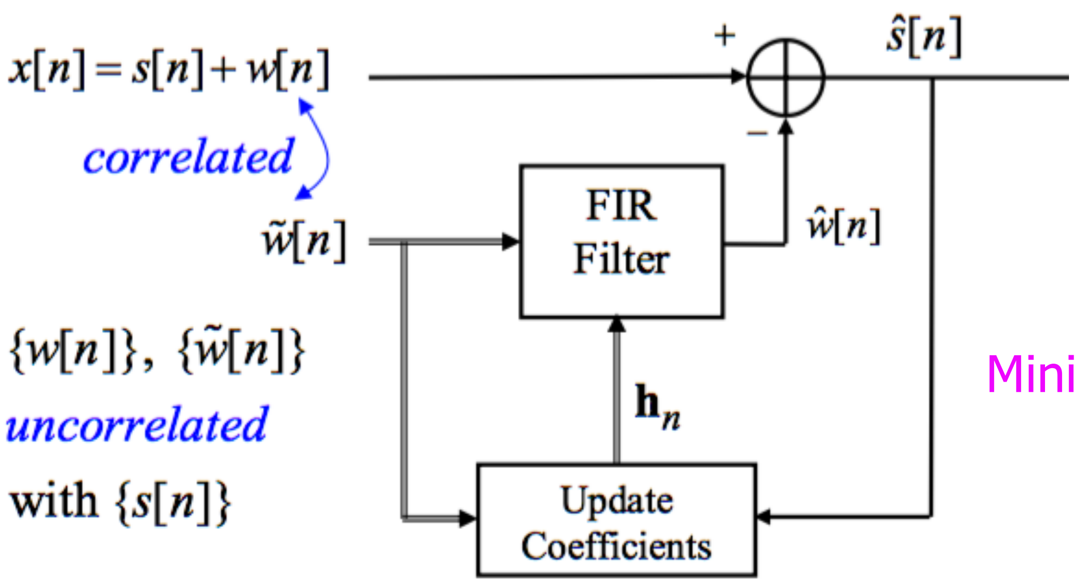


Adaptive Interference Cancellation



$$\hat{s}[n] = s[n] + w[n] - \hat{w}[n]$$
$$= s[n] + w[n] - h_n^T \tilde{w}_n$$

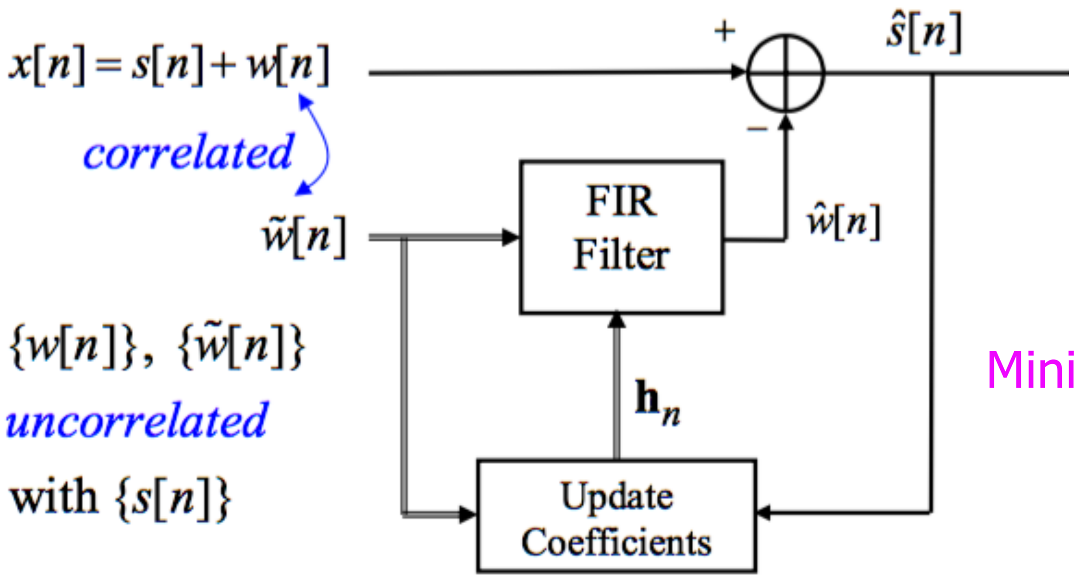
Adaptive Interference Cancellation



$$\hat{s}[n] = s[n] + w[n] - \hat{w}[n]$$
$$= s[n] + w[n] - h_n^T \tilde{w}_n$$

Minimizing $(\hat{s}[n])^2$ removes noise w[n]

Adaptive Interference Cancellation



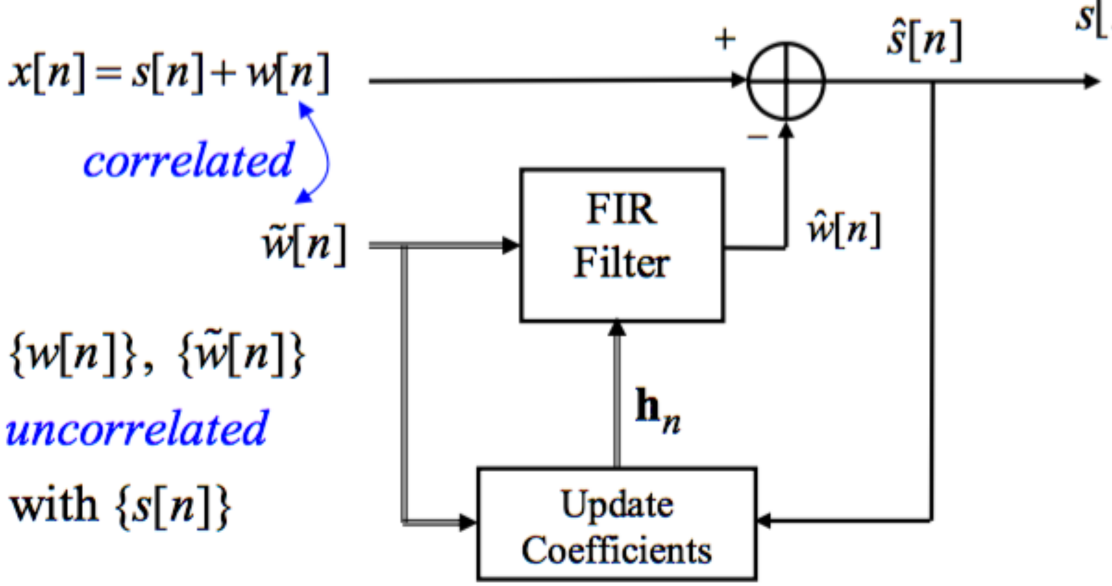
$$\hat{s}[n] = s[n] + w[n] - \hat{w}[n]$$
$$= s[n] + w[n] - h_n^T \tilde{w}_n$$

Minimizing $(\hat{s}[n])^2$ removes noise w[n]

$$\left(\hat{s}[n]\right)^2 = \left(s[n] + w[n] - h_n^T \tilde{w}_n\right)^2$$

$$\frac{\partial \hat{s}^2[n]}{\partial h_n} = 2\left(s[n] + w[n] - h_n^T \tilde{w}_n\right)(-\tilde{w}_n)$$
$$= 2\hat{s}[n](-\tilde{w}_n) = -2\hat{s}[n]\tilde{w}_n$$

Adaptive Interference Cancellation



$\{w[n]\}, \{\tilde{w}[n]\}$ *uncorrelated* with $\{s[n]\}$

$$\hat{s}[n] = x[n] - \mathbf{h}_n^T \tilde{\mathbf{w}}_n$$
$$= s[n] + \{w[n] - \mathbf{h}_n^T \tilde{\mathbf{w}}_n\}$$

uncorrelated

Minimize *instantaneous power* $\hat{s}^2[n]$ for best interference cancellation

$$\frac{d\left(\hat{s}[n]\right)^2}{d\,\mathbf{h}_n} = -2\hat{s}[n]\tilde{\mathbf{w}}_n$$

$$\boxed{\mathbf{h}_{n+1} = \mathbf{h}_n + 2\mu\,\hat{s}[n]\tilde{\mathbf{w}}_n}$$

Stability of LMS

- The LMS algorithm is convergent in the mean square if and only if the step-size parameter satisfy

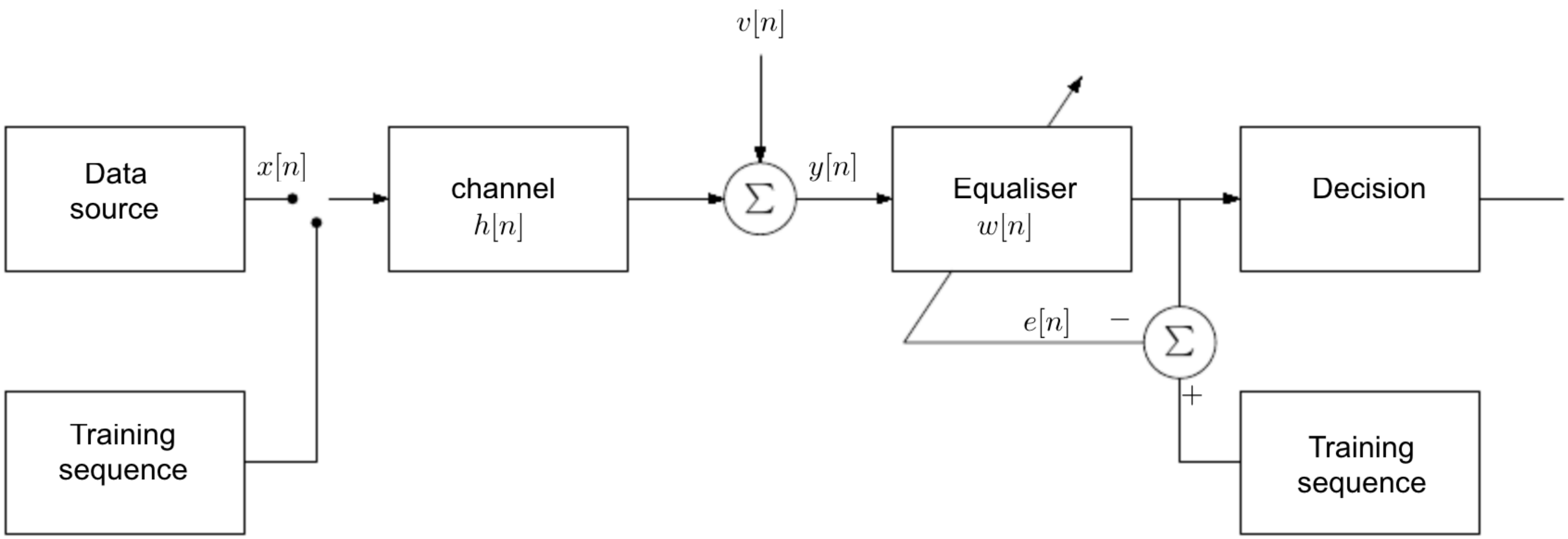
$$0 < \mu < \frac{2}{\lambda_{max}}$$

- Here λ_{max} is the largest eigenvalue of the correlation matrix of the input data
- More practical test for stability is

$$0 < \mu < \frac{2}{\text{input signal power}}$$

- Larger values for step size
  - Increases adaptation rate (faster adaptation)
  - Increases residual mean-squared error

Adaptive Equalization



Big Ideas

- Adaptive Filters
  - Use LMS algorithm to update filter coefficients
  - Applications like system ID, channel equalization, and signal prediction

Admin

- Project 2
  - Out after lecture
  - Due 4/26 (last day of classes)
- Final Exam – 5/1
  - 6-8pm
  - DRLB A2