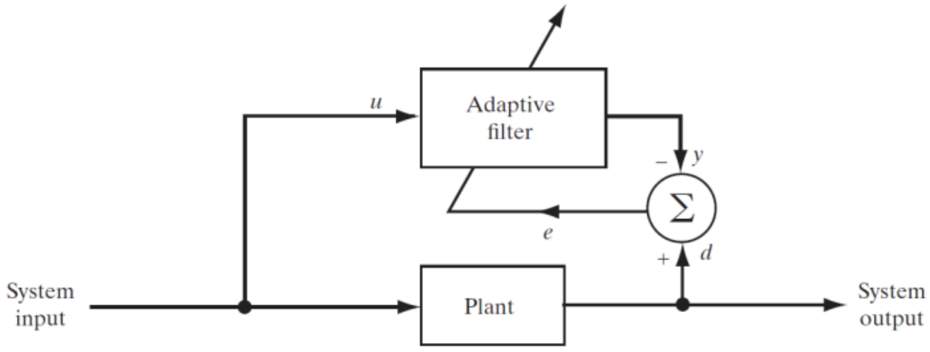
Steepest Descent and Least-Mean-Square (LMS) Adaptive Filters

Jesper Rindom Jensen, jrj@es.aau.dk

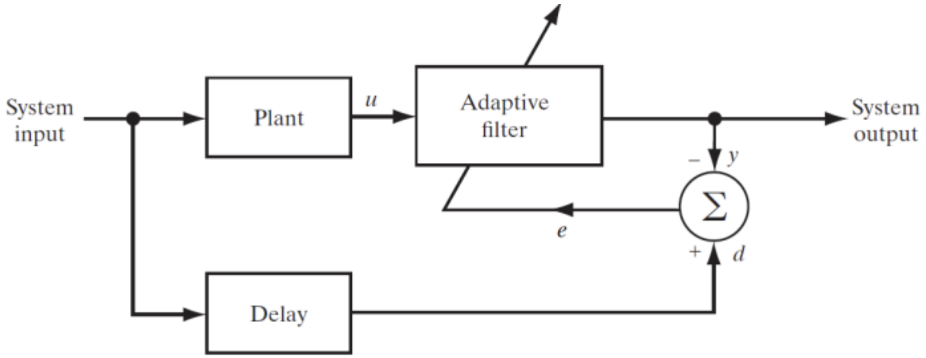
Advanced Signal Processing, Electronic Systems, Aalborg University

Why Adaptive Filtering?

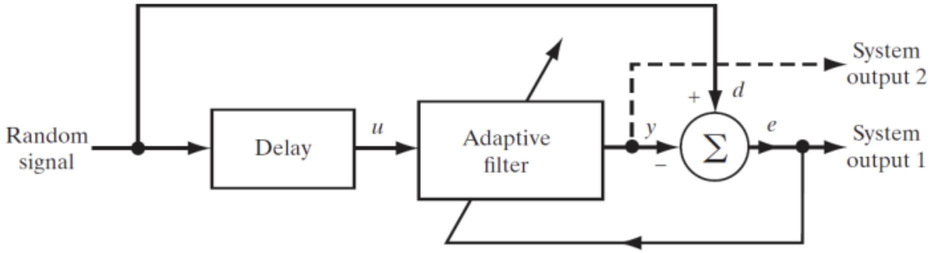
Applications



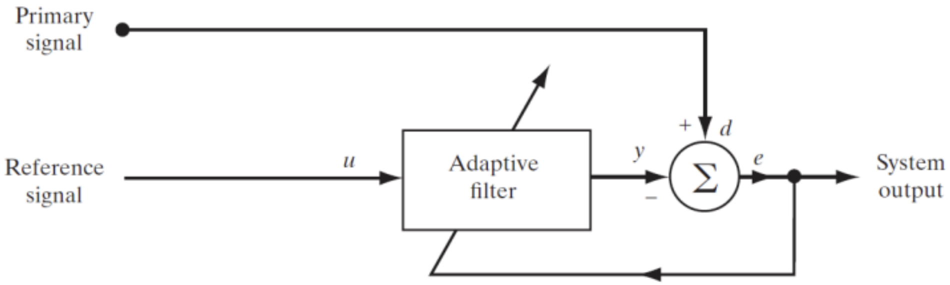
System identification



Inverse modeling (equalization)



Prediction



Noise cancellation

Why Adaptive Filtering?

- Imagine that $u(n)$ and $d(n)$ are not jointly stationary → the optimal filter weights are time-varying…

$$\mathbf{R}(n)\mathbf{w}_o(n) = \mathbf{p}(n)$$

$$\mathbf{R}(n) = \begin{bmatrix} r(n,n) & \cdots & r(n, n-M+1) \\ \vdots & \ddots & \vdots \\ r(n-M+1,n) & \cdots & r(n-M+1, n-M+1) \end{bmatrix}$$

$$\mathbf{p}(n) = \begin{bmatrix} p(n,n) & \cdots & p(n-M+1,n) \end{bmatrix}^T$$

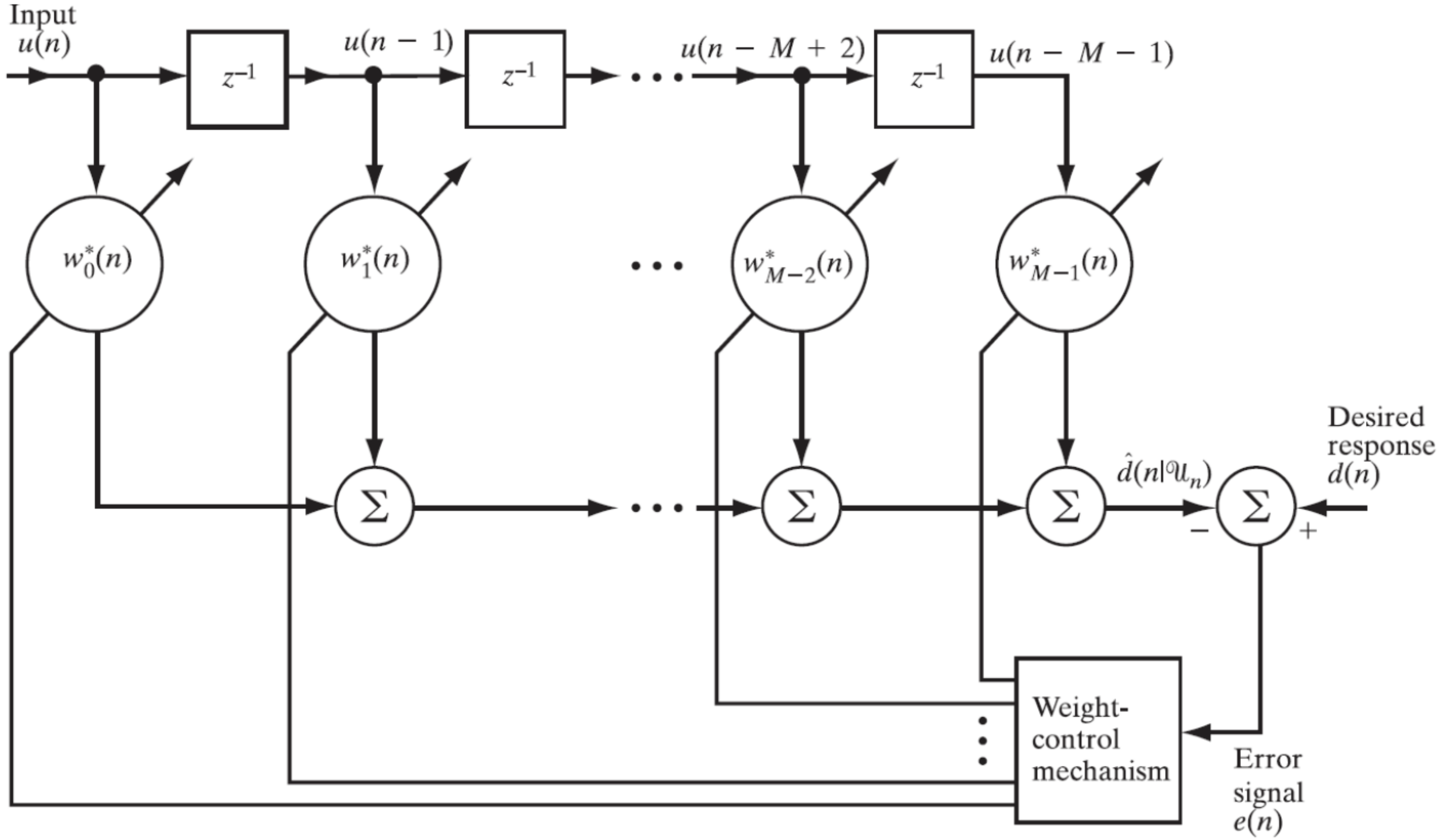
Why Adaptive Filtering?

- Two issues…
  - High computational complexity
  - Statistics might be unknown

- Adaptive filtering can solve these issues!

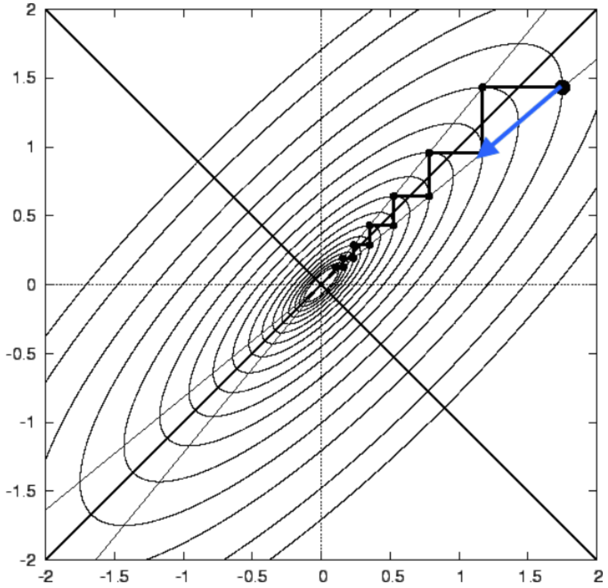
Why Adaptive Filtering?

- Adaptive FIR filtering



Steepest Descent

- **Steepest descent:** recursive method that allows us the iterative estimation of a filter ➔ tracking of time variations in signal's statistics.
- Steepest descent, when applied to Wiener filtering, converges to the Wiener optimal solution in stationary scenarios.



$$\mathbf{w}(n+1) = \mathbf{w}(n) - \frac{1}{2}\mu \frac{\partial J(\mathbf{w}(n))}{\partial \mathbf{w}(n)}$$

$$J(\mathbf{w}(n+1)) \approx J(\mathbf{w}(n)) - \frac{1}{2}\mu \left\| \frac{\partial J(\mathbf{w}(n))}{\partial \mathbf{w}(n)} \right\|^2$$

$\mu > 0$ Step-size parameter

Steepest Descent

Application of steepest descent to Wiener filtering

- Now, the filter weights are time-dependent: $w_i \rightarrow w_i(n), i = 0, \ldots, M-1$

$$e(n) = d(n) - \sum_{i=0}^{M-1} w_i^*(n) u(n-i) = d(n) - \mathbf{w}^H(n)\mathbf{u}(n)$$

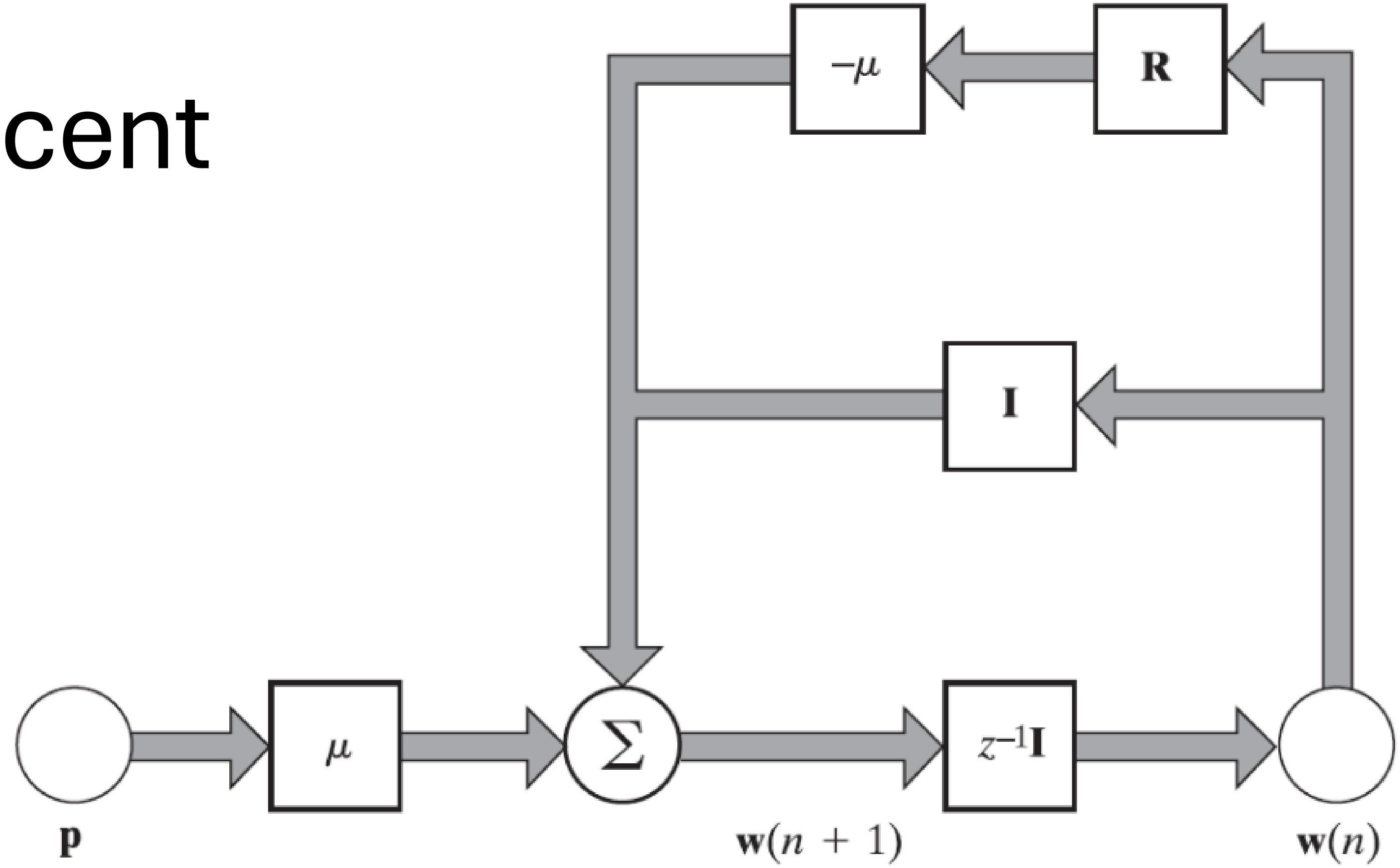
$$J(\mathbf{w}(n)) = E[e(n)e^*(n)] = \sigma_d^2 - \mathbf{w}^H(n)\mathbf{p} - \mathbf{p}^H\mathbf{w}(n) + \mathbf{w}^H(n)\mathbf{R}\mathbf{w}(n)$$

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

- And combining the above with the steepest-descent expression…

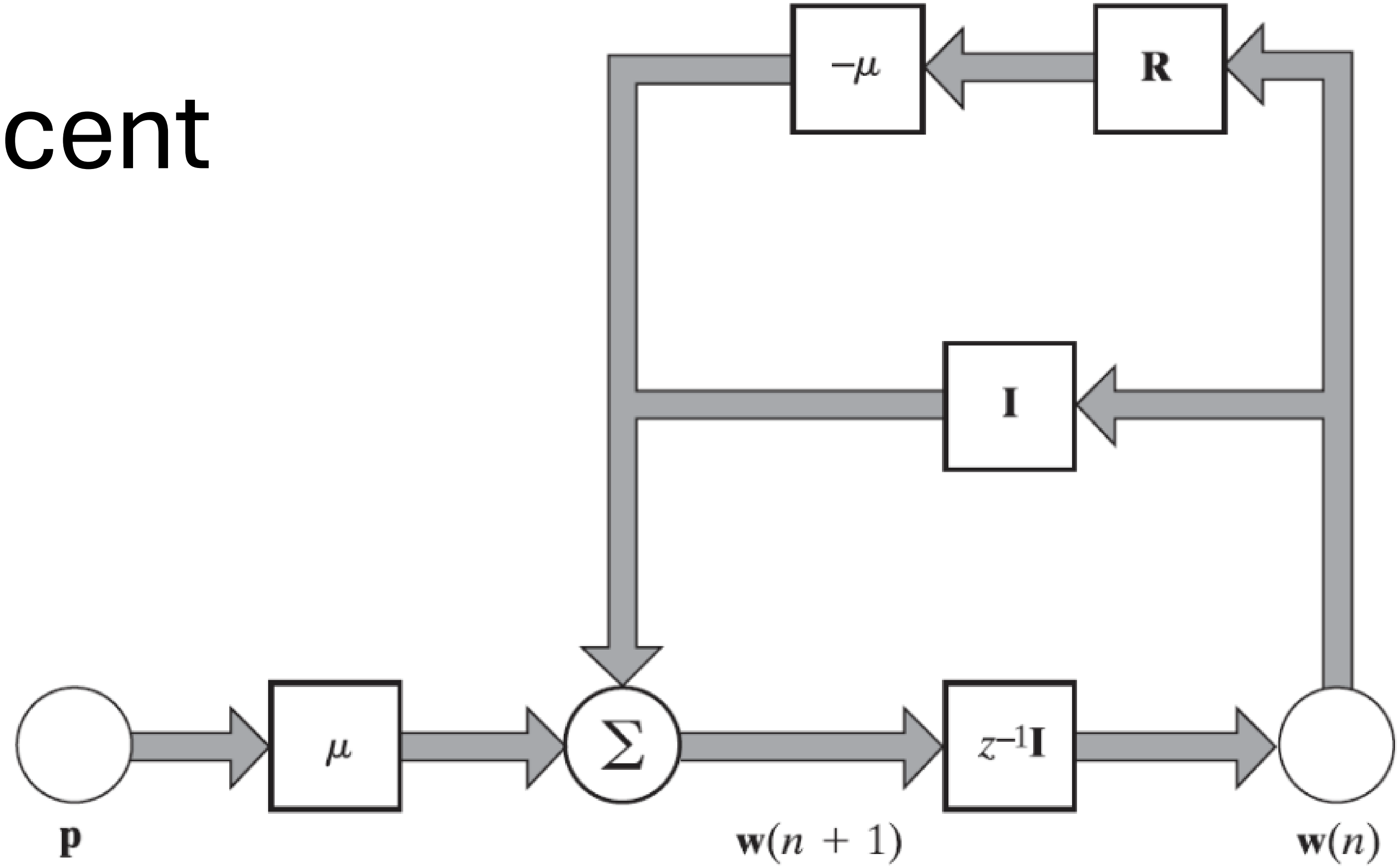
$$\mathbf{w}(n+1) = \mathbf{w}(n) - \frac{1}{2}\mu \frac{\partial J(\mathbf{w}(n))}{\partial \mathbf{w}(n)} = \mathbf{w}(n) + \mu(\mathbf{p} - \mathbf{R}\mathbf{w}(n))$$

Steepest Descent



- Convergence/stability of steepest descent depends on...
  - ...the step-size parameter μ
  - ...the correlation matrix **R**

Steepest Descent



- A necessary and sufficient condition for the convergence/stability of steepest descent is

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

λ_{max} the largest eigenvalue of **R**

Stochastic Gradient Descent

- In real-world applications, normally, the statistics **R** and **p** are unavailable

- We then need different *adaptive filtering* methods to track time variations in signal's statistics in an *online manner*:
    - **Least-mean-square (LMS) algorithm**
    - Recursive least-squares (RLS) algorithm

- LMS is based on **stochastic gradient descent** (**SGD**)

Least-Mean-Square Adaptive Filters

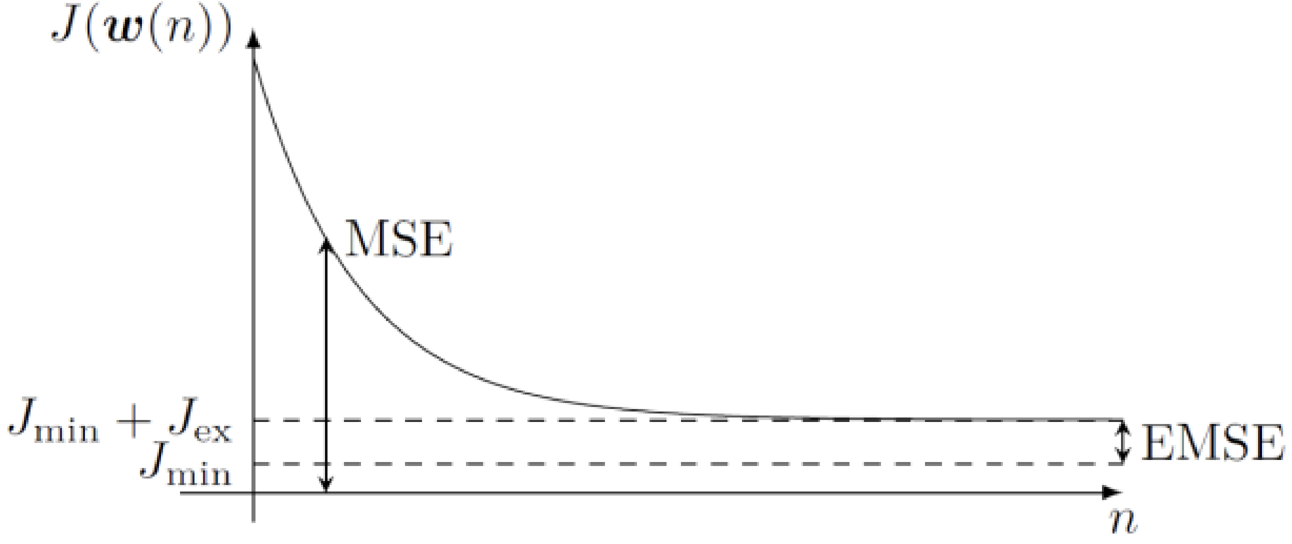
Some characteristics of LMS adaptive filtering:

- Computational complexity **linear** with the order of the FIR filter
- Knowledge on signal's statistics is not required
- LMS is robust against environmental noise/interference
- Unlike RLS, LMS does not require the inversion of the correlation matrix of the regressor

Least-Mean-Square Adaptive Filters

LMS (stochastic gradient descent, SGD) versus steepest descent (SD)

- In SD (SGD), the statistics are known (not known), so the gradient and filter weights are deterministic functions (stochastic processes)
- As a result, SGD is easier to implement in practice than SD
- SGD is an approximation to SD

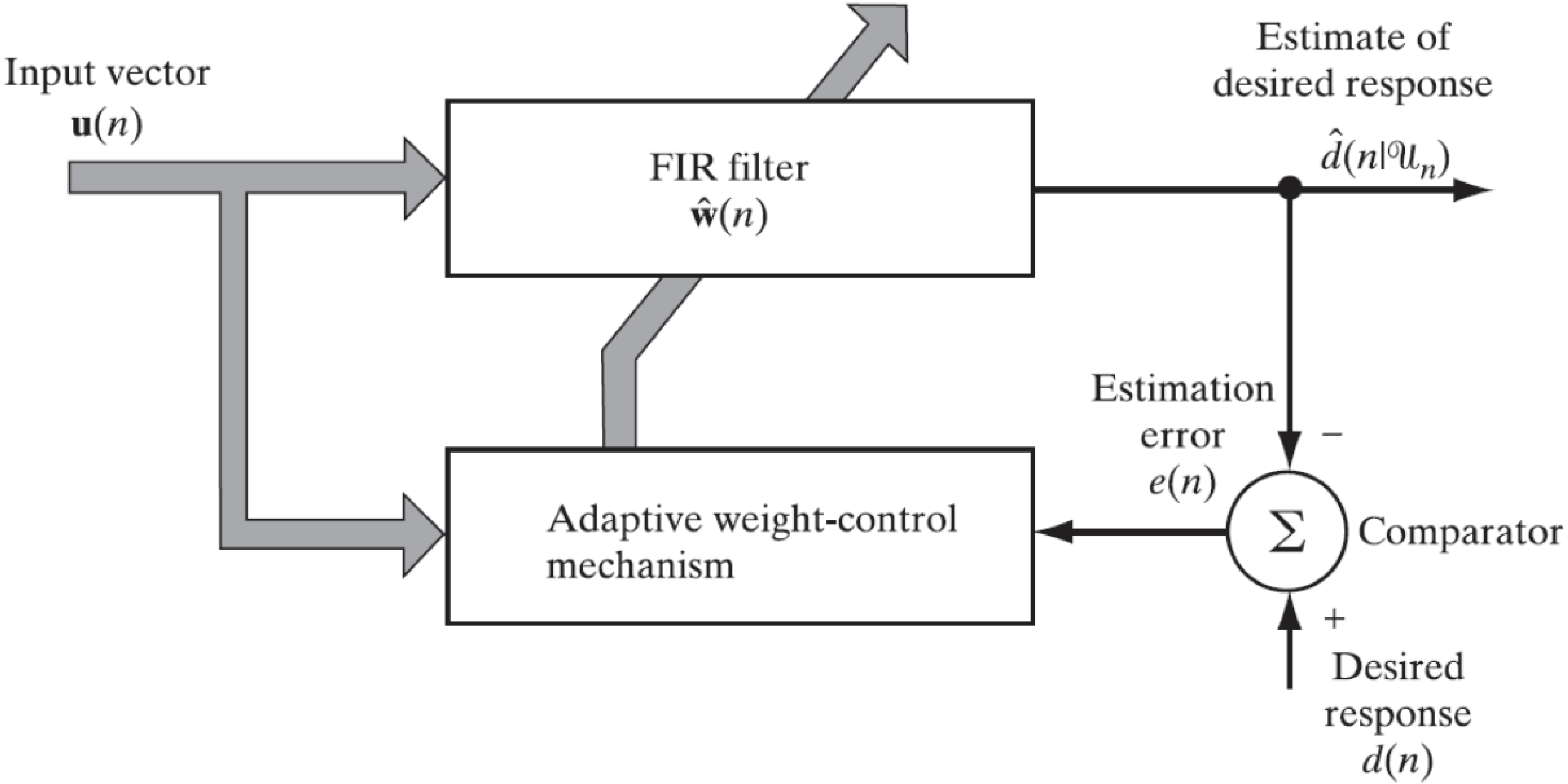


SD is optimal: it approaches to the Wiener solution as $n \rightarrow +\infty$

LMS is suboptimal: it randomly walks around the Wiener solution (*gradient noise*)

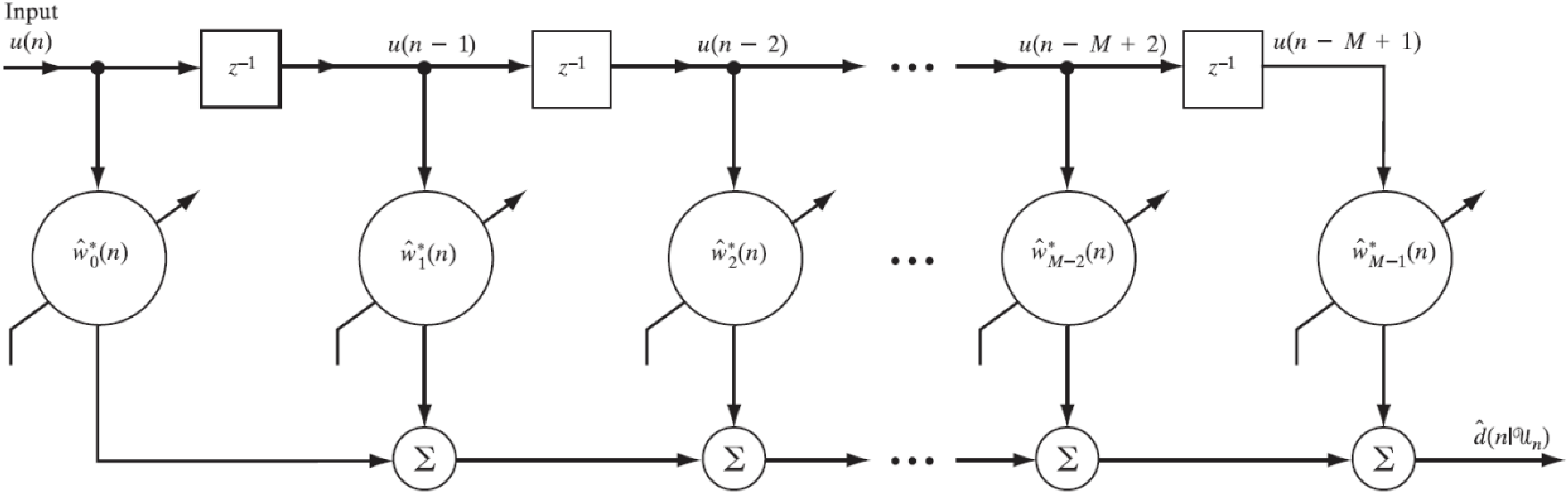
Least-Mean-Square Adaptive Filters

Overall block diagram of an (LMS) adaptive filter:



Least-Mean-Square Adaptive Filters

- **FIR filter:**



Least-Mean-Square Adaptive Filters

- **Reminder for Wiener filtering:** $J = E[|e(n)|^2]$
  - Adaptation based on the **instantaneous** error signal *e*(*n*) → **stochastic gradient descent**: we get rid of the expectation operator

$$J_s(n) = e(n)e^*(n) = |e(n)|^2$$ Sample value of a *stochastic* process

$$\nabla J_{s,k}(n) = \frac{\partial J_s(n)}{\partial w_k^*(n)} = -2u(n-k)e^*(n)$$

- **LMS updating rule:**

$$\hat{w}_k(n+1) = \hat{w}_k(n) - \frac{1}{2}\mu \nabla J_{s,k}(n)$$

$$\hat{w}_k(n+1) = \hat{w}_k(n) + \mu u(n-k)e^*(n)$$

Least-Mean-Square Adaptive Filters

- **LMS in vector form:**

$$\hat{\mathbf{w}}(n) = \begin{bmatrix} \hat{w}_0(n) & \hat{w}_1(n) & \cdots & \hat{w}_{M-1}(n) \end{bmatrix}^T$$
$$\mathbf{u}(n) = \begin{bmatrix} u(n) & u(n-1) & \cdots & u(n-M+1) \end{bmatrix}^T$$

- **LMS updating rule:**

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

Least-Mean-Square Adaptive Filters

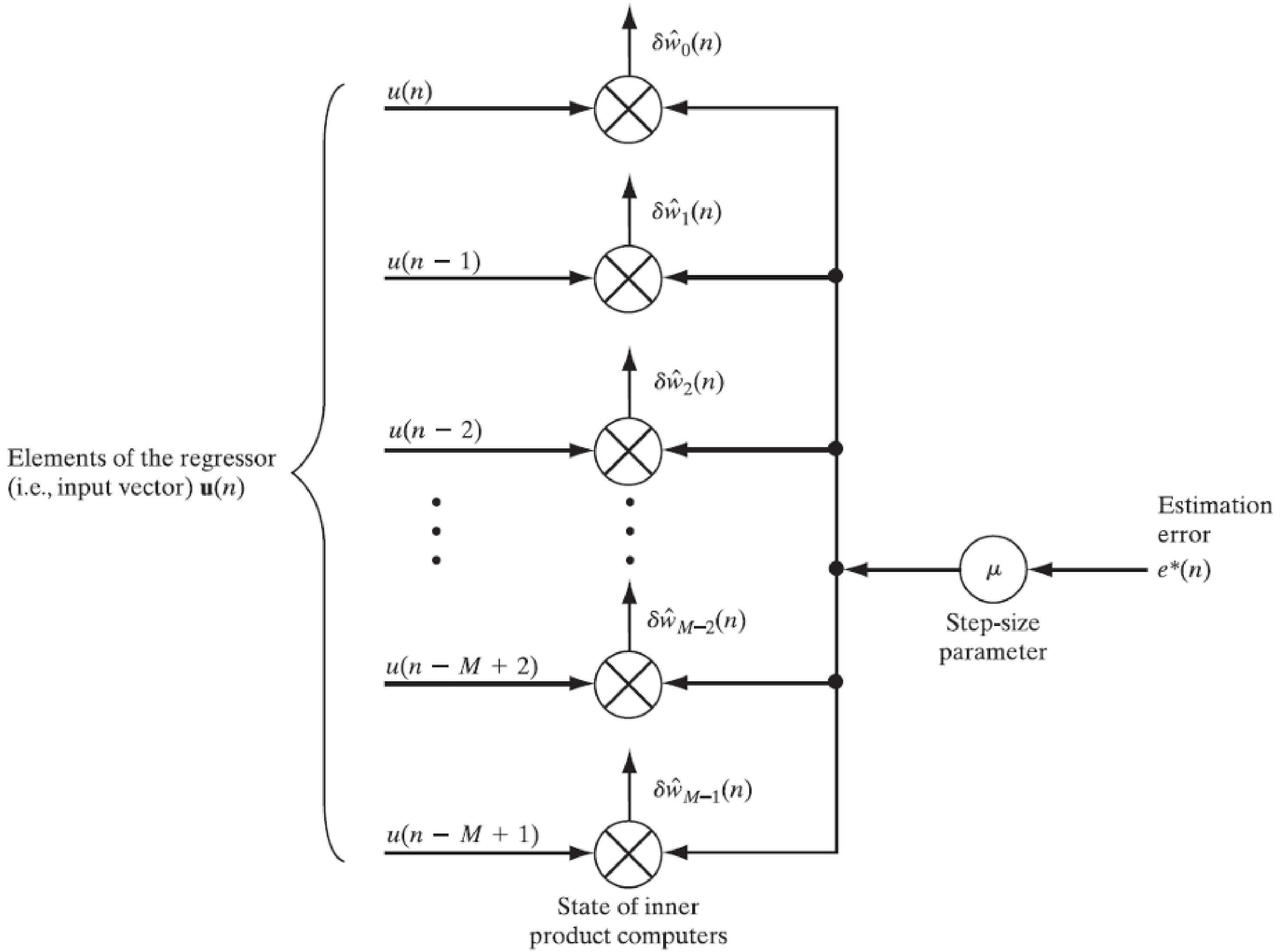
- Furthermore, bear in mind that…
  - $e(n) = d(n) - \hat{d}(n)$
  - $\hat{d}(n) = \hat{\mathbf{w}}^H(n)\mathbf{u}(n)$

- We can reach the same solution from steepest descent:
  - Instead of **R**: $\mathbf{u}(n)\mathbf{u}^H(n)$
  - Instead of **p**: $\mathbf{u}(n)d^*(n)$

- **LMS feedback**: the instantaneous estimates of **R** and **p** are averaged over time during adaptation

Least-Mean-Square Adaptive Filters

- **Adaptive weight-control mechanism of LMS:**



Least-Mean-Square Adaptive Filters

- As for steepest descent, convergence/stability of LMS is guaranteed when

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

- In practice, μ has to be sufficiently small for a robust behavior
- The excess mean-square error is defined as $J_{ex}(n) = J(n) - J_{min}(n)$
- **Misadjustment:**

$$\mathcal{M} = \frac{J_{\mathrm{ex}}(n \to \infty)}{J_{\min}} = \frac{\mu}{2} \sum_{k=1}^{M} \lambda_k$$

- We can give a power interpretation to the above equation

Summary of the LMS algorithm

Parameters: M = number of taps (i.e., filter length)
μ = step-size parameter

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

where $\lambda_{\max}$ is the maximum value of the correlation matrix of the tap inputs $u(n)$ and the filter length M is moderate to large.

Initialization: If prior knowledge of the tap-weight vector $\hat{\mathbf{w}}(n)$ is available, use it to select an appropriate value for $\hat{\mathbf{w}}(0)$. Otherwise, set $\hat{\mathbf{w}}(0) = \mathbf{0}$.

Data:

- Given $\mathbf{u}(n)$ = M-by-1 tap-input vector at time n
  $= [u\,((n), u(n\text{-}1)), \ldots, u(n-M+1)]^{\mathrm{T}}$
  $d(n)$ = desired response at time n.
- To be computed:
  $\hat{\mathbf{w}}(n+1) =$ estimate of tap-weight vector at time $n+1$.

Computation: For $n = 0, 1, 2, \ldots$, compute

$$e(n) = d(n) - \hat{\mathbf{w}}^{\mathrm{H}}(n)\mathbf{u}(n)$$

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

Normalized LMS

- In LMS, when **u**(*n*) is large, the adjustment is also large → *gradient noise amplification* problem

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

- **Solution**: employing **normalized least-mean-square (NLMS)** adaptive filters

- *Normalization factor*

$$\|\mathbf{u}(n)\|^2$$

- NLMS has the same structure as LMS

Normalized LMS

- *Principle of minimal disturbance*

- **Normalized LMS**: estimate $\hat{\mathbf{w}}(n+1)$ minimizing

$$\|\delta\hat{\mathbf{w}}(n+1)\|^2 = \|\hat{\mathbf{w}}(n+1) - \hat{\mathbf{w}}(n)\|^2$$

- subject to the constraint

$$\hat{\mathbf{w}}^H(n+1)\mathbf{u}(n) = d(n)$$

- We solve it by *Lagrange Multipliers*:

$$J(n) = \|\delta\hat{\mathbf{w}}(n+1)\|^2 + \mathrm{Re}(\lambda^*(d(n) - \hat{\mathbf{w}}^H(n+1)\mathbf{u}(n)))$$

Normalized LMS

- **Solving Lagrange Multipliers**:

$$\frac{\partial J(n)}{\partial \hat{\mathbf{w}}^H(n+1)} = 2(\hat{\mathbf{w}}(n+1) - \hat{\mathbf{w}}(n)) - \lambda^* \mathbf{u}(n) = 0$$

$$\hat{\mathbf{w}}(n+1) = \hat{\mathbf{w}}(n) + \tfrac{1}{2}\lambda^* \mathbf{u}(n)$$

- And the constraint…

$$d(n) = \hat{\mathbf{w}}^H(n+1)\mathbf{u}(n) = \left(\hat{\mathbf{w}}(n) + \tfrac{1}{2}\lambda^* \mathbf{u}(n)\right)^H \mathbf{u}(n)$$

$$\lambda = \frac{2(d(n) - \hat{\mathbf{w}}^H(n)\mathbf{u}(n))}{\|\mathbf{u}(n)\|^2} = \frac{2e(n)}{\|\mathbf{u}(n)\|^2}$$

- **NLMS updating rule:**

$$\hat{\mathbf{w}}(n+1) = \hat{\mathbf{w}}(n) + \frac{\tilde{\mu}}{\|\mathbf{u}(n)\|^2}\mathbf{u}(n)e^*(n)$$

LMS vs. Normalized LMS

- NLMS is equivalent to LMS with a *time-varying step-size parameter*

- NLMS has a rate of convergence potentially faster than that of LMS

- To avoid numerical instability in NLMS, we introduce $\delta \gtrsim 0$ as in

$$\hat{\mathbf{w}}(n+1) = \hat{\mathbf{w}}(n) + \frac{\tilde{\mu}}{\delta + \|\mathbf{u}(n)\|^2}\mathbf{u}(n)e^*(n)$$

Normalized LMS

- Defining some quantities...

$$e(n) = d(n) - \hat{\mathbf{w}}^H(n) + \mathbf{u}(n) \qquad d(n) = \mathbf{w}^H\mathbf{u}(n) + v(n)$$

$$\xi_u(n) = (\mathbf{w} - \hat{\mathbf{w}}(n))^H\mathbf{u}(n)$$ *Undisturbed error signal*

- Convergence/stability of NLMS is guaranteed if

$$0 < \tilde{\mu} < 2\frac{\mathrm{Re}(E[\xi_u(n)e^*(n)/\|\mathbf{u}(n)\|^2])}{E[|e(n)|^2/\|\mathbf{u}(n)\|^2]}$$

- And, for real-valued data...

$$\mathcal{D}(n) = E[\|\boldsymbol{\epsilon}(n)\|^2] = E[\|\mathbf{w} - \hat{\mathbf{w}}(n)\|^2]$$

$$0 < \tilde{\mu} < 2\frac{E[u^2(n)]\mathcal{D}(n)}{E[e^2(n)]}$$

Summary of the NLMS algorithm

Parameters: M = number of taps (i.e., filter length)

$\widetilde{\mu}$ = adaptation constant

$$0 < \widetilde{\mu} < 2\frac{\mathbb{E}[|u(n)|^2]\mathcal{D}(n)}{\mathbb{E}[|e(n)|^2]},$$

where

$\mathbb{E}[|e(n)|^2]$ = error signal power,

$\mathbb{E}[|u(n)|^2]$ = input signal power,

$\mathcal{D}(n)$ = mean-square deviation.

Initialization. If prior knowledge about the tap-weight vector $\hat{\mathbf{w}}(n)$ is available, use that knowledge to select an appropriate value for $\hat{\mathbf{w}}(0)$. Otherwise, set $\hat{\mathbf{w}}(0) = \mathbf{0}$.

Data

(a) Given: $\mathbf{u}(\mathrm{n})$ = M-by-1 tap input vector at time n.

$d(\mathrm{n})$ = desired response at time step n.

(b) To be computed: $\hat{\mathbf{w}}(n+1)$ = estimate of tap-weight vector at time step n + 1.

Computation: For $n = 0, 1, 2, \ldots,$ compute

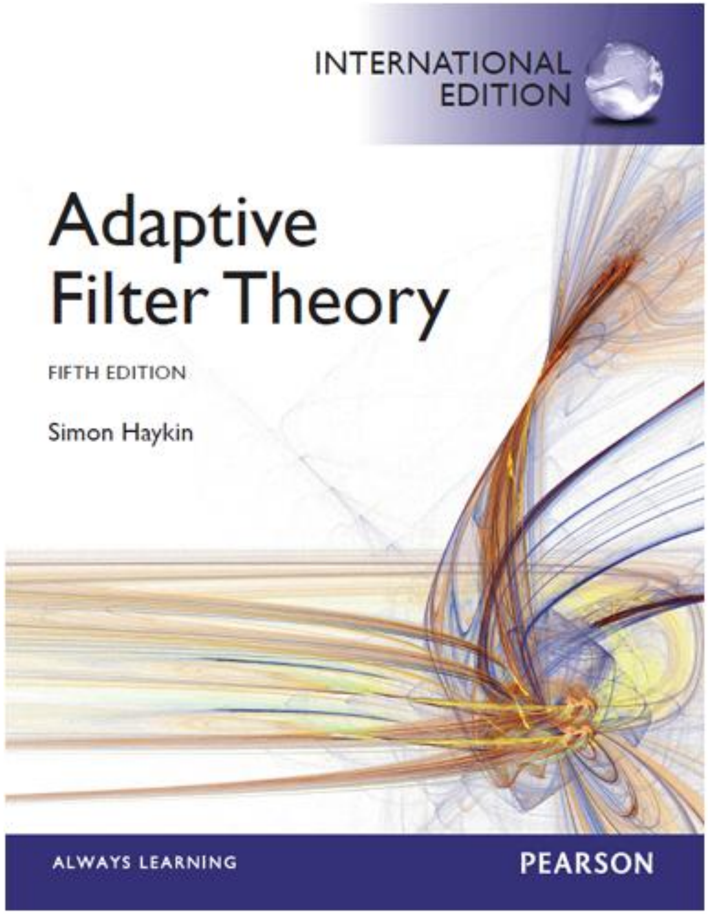
$$e(n) = d(n) - \hat{\mathbf{w}}^{\mathrm{H}}(n)\mathbf{u}(n),$$

$$\hat{\mathbf{w}}(n+1) = \hat{\mathbf{w}}(n) + \frac{\widetilde{\mu}}{\|\mathbf{u}(n)\|^2}\mathbf{u}(n)e^*(n).$$

Bibliography

Simon Haykin, “Adaptive Filter Theory (5th Edition)”. Pearson, 2014

- Steepest Descent: 4.1, 4.2 and 4.3
- Stochastic Gradient Descent: 5.1
- Least-Mean-Square Adaptive Filters: 5.2, 6.1, 6.2, 6.4 and 6.5
- Normalized LMS: 7.1, 7.2 and 7.3



Assignment: Echo Cancellation

We observe a primary signal *d*(*n*) (‘signal.asc’) that consists of a local signal (‘local.asc’, theoretically unknown) mixed with an interference. This interference consists of an echo, which is a hybrid-circuit-filtered version of a remote signal *u*(*n*) (‘remota.asc’). The sampling rate of these signals is 8 kHz. We want to design a **normalized LMS filter** canceling such an echo.

- 1. Draw the block diagram of the system
- 2. What is the a priori signal-to-noise ratio (SNR) of the observed primary signal *d*(*n*)? If *y*(*n*) = *x*(*n*) + *v*(*n*) is a noisy signal, where *x*(*n*) and *v*(*n*) are the target and noise signals, respectively, recall that the sample SNR is computed as

$$\mathrm{SNR} = 10\log_{10}\left(\frac{\sum_{n=0}^{N-1} x^2(n)}{\sum_{n=0}^{N-1} v^2(n)}\right)$$

Assignment: Echo Cancellation

- 3. Implement an echo canceler using NLMS, where $\hat{\mathbf{w}}(0) = \mathbf{0}$. Search for the step-size parameter $\hat{\tilde{\mu}}$ and filter order $\hat{M}$ maximizing the SNR of the signal after echo cancellation. The search ranges are $\tilde{\mu}$ ∈{0.0001, 0.0002, 0.0004, …, 0.0512} and *M* ∈ {1, 2, 3, …, 10}. What are the SNR, $\hat{\tilde{\mu}}$ and $\hat{M}$ values?
- 4. Plot the time evolution of the filter weights for the best SNR configuration. Also, plot both $d(n)$ and the signal after echo cancellation. Very briefly, comment the results
- 5. Repeat *3)* and *4)* by assuming that a *double talk detector* (DTD) spots local speech from sample 2,200, in such a manner that the filter adapts up to that sample only. Compare the results with those obtained in *3)* and *4)*. What kind of filter is obtained upon sample 2,200?