Chapter 4

- **Adaptive Filter Theory and Applications**

References:

- B.Widrow and M.E.Hoff, “Adaptive switching circuits,” *Proc. Of WESCON Conv. Rec.*, part 4, pp.96-140, 1960
- B.Widrow and S.D.Stearns, *Adaptive Signal Processing*, Prentice-Hall, 1985
- O.Macchi, *Adaptive Processing: The Least Mean Squares Approach with Applications in Transmission*, Wiley, 1995
- P.M.Clarkson, *Optimal and Adaptive Signal Processing*, CRC Press, 1993
- S.Haykin, *Adaptive Filter Theory*, Prentice-Hall, 2002
- D.F.Marshall, W.K.Jenkins and J.J.Murphy, "The use of orthogonal transforms for improving performance of adaptive filters", *IEEE Trans. Circuits & Systems*, vol.36, April 1989, pp.474-483

Adaptive Signal Processing is concerned with the design, analysis, and implementation of systems whose structure changes in response to the incoming data.

Application areas are similar to those of optimal signal processing but now the environment is changing, the signals are nonstationary and/or the parameters to be estimated are time-varying. For example,

- Echo cancellation for Hand-Free Telephones (The speech echo is a nonstationary signal)
- Equalization of Data Communication Channels (The channel impulse response is changing, particularly in mobile communications)
- Time-Varying System Identification (the system transfer function to be estimated is non-stationary in some control applications)

Adaptive Filter Development

Year	Application	Developer(s)
1959	Adaptive pattern recognition system	Widrow *et al*
1960	Adaptive waveform recognition	Jacowatz
1965	Adaptive equalizer for telephone channel	Lucky
1967	Adaptive antenna system	Widrow *et al*
1970	Linear prediction for speech analysis	Atal
Present	numerous applications, structures, algorithms	

Adaptive Filter Definition

An adaptive filter is a *time-variant* filter whose coefficients are adjusted in a way to optimize a *cost function* or to satisfy some predetermined optimization criterion.

Characteristics of adaptive filters:

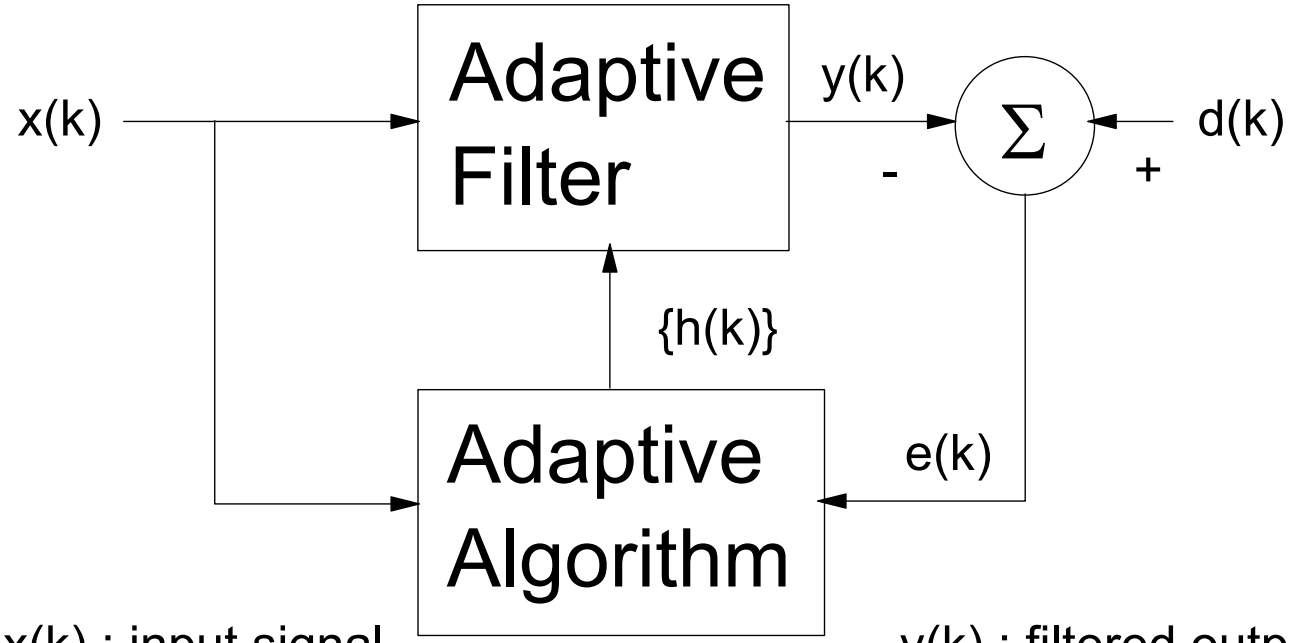
- They can automatically adapt (self-optimize) in the face of changing environments and changing system requirements
- They can be trained to perform specific filtering and decision-making tasks according to some updating equations (training rules)

Why adaptive?

It can *automatically* operate in

- *changing environments* (e.g. signal detection in wireless channel)
- *nonstationary signal/noise* conditions (e.g. LPC of a speech signal)
- *time-varying parameter* estimation (e.g. position tracking of a moving source)

Block diagram of a typical adaptive filter is shown below:



x(k) : input signal

y(k) : filtered output

d(k) : desired response

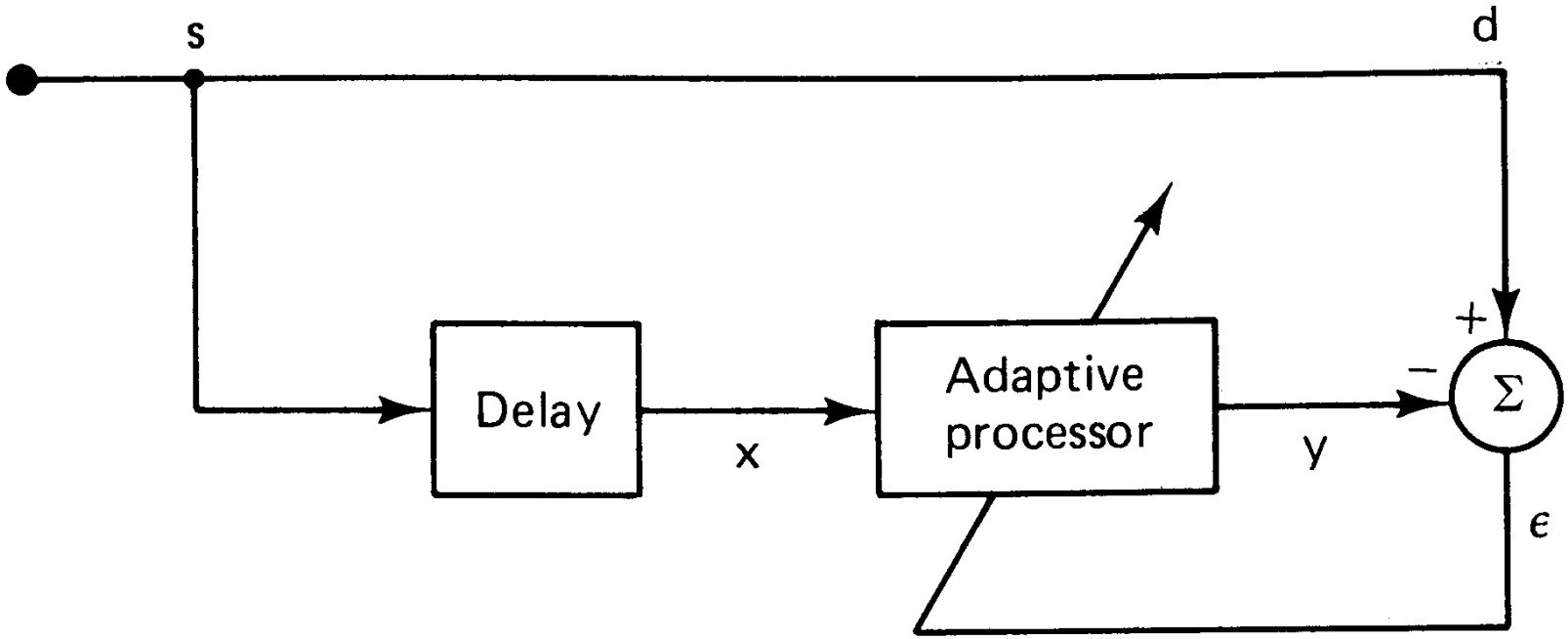
h(k) : impulse response of adaptive filter

The cost function may be $E\{e^2(k)\}$ or $\sum_{k=0}^{N-1} e^2(k)$

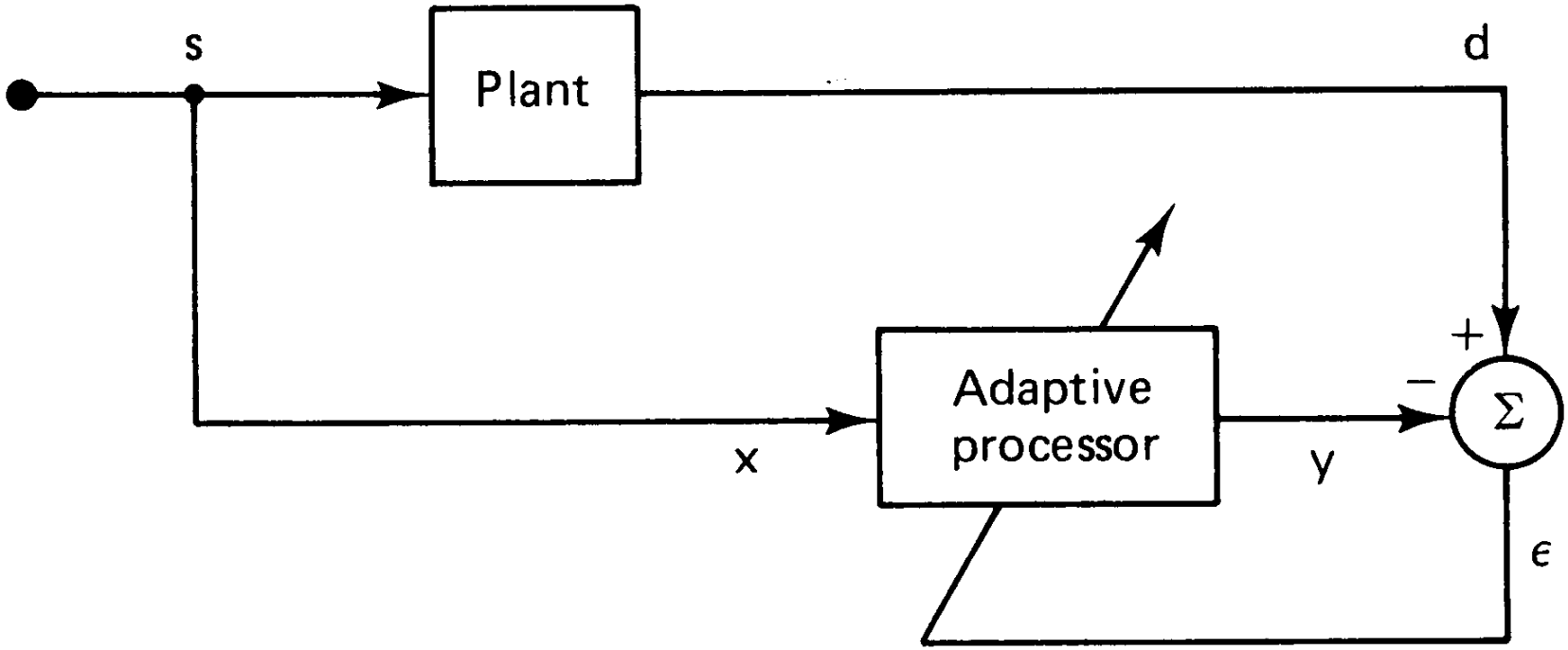
- FIR or IIR adaptive filter
- filter can be realized in various structures
- adaptive algorithm depends on the optimization criterion

Basic Classes of Adaptive Filtering Applications

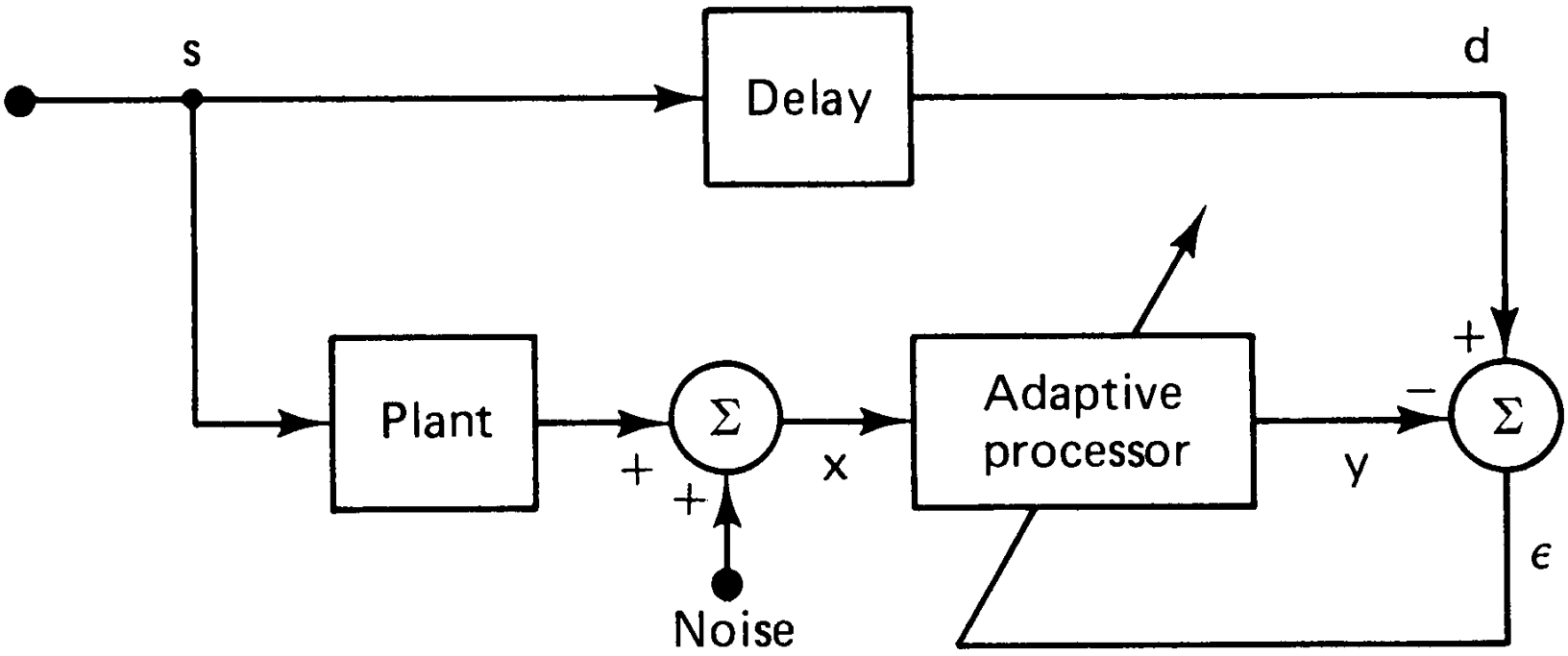
1.**Prediction** : signal encoding, linear prediction coding, spectral analysis



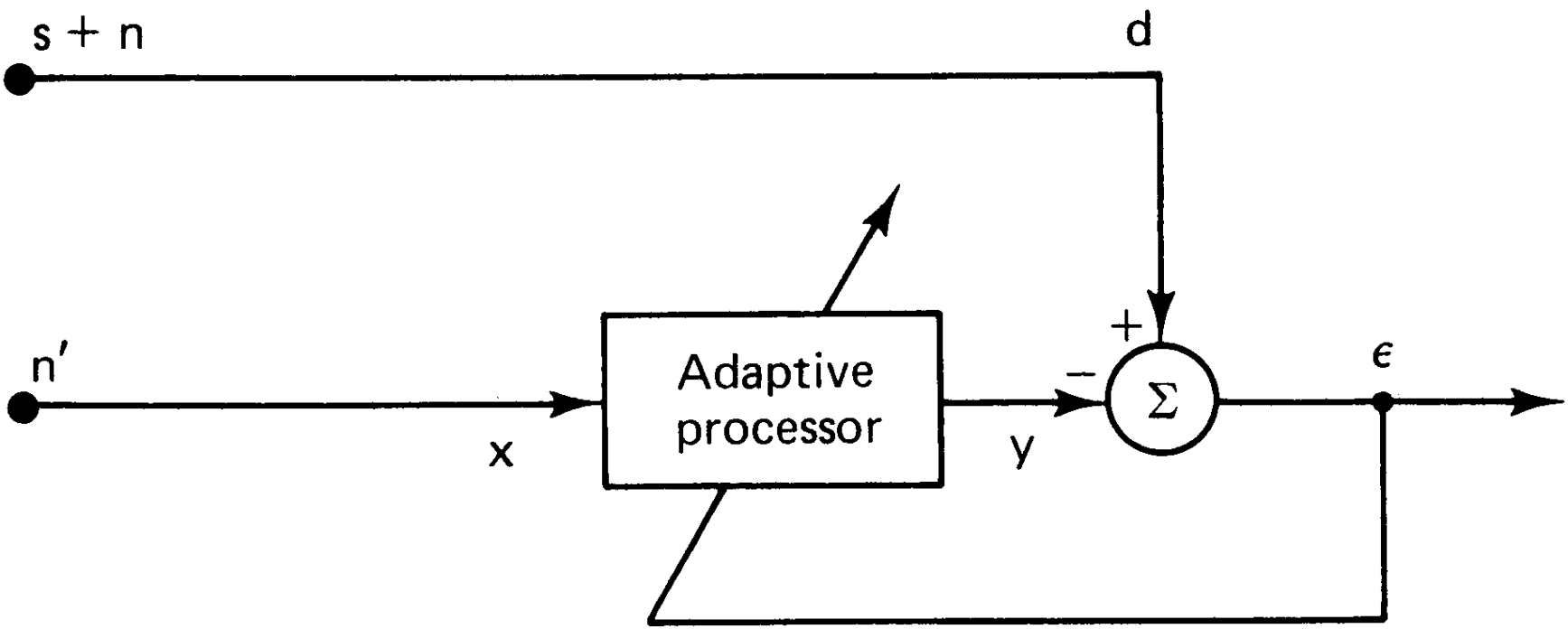
2.**Identification** : adaptive control, layered earth modeling, vibration studies of mechanical system



3.**Inverse Filtering** : adaptive equalization for communication channel, deconvolution



4.**Interference Canceling** : adaptive noise canceling, echo cancellation



Design Considerations

1. Cost Function

- choice of cost functions depends on the approach used and the application of interest
- some commonly used cost functions are

mean square error (MSE) criterion : minimizes $E\{e^2(k)\}$
where E denotes expectation operation, $e(k) = d(k) - y(k)$ is the estimation error, $d(k)$ is the desired response and $y(k)$ is the actual filter output

exponentially weighted least squares criterion : minimizes $\sum_{k=0}^{N-1} \lambda^{N-1-k} e^2(k)$
where N is the total number of samples and λ denotes the exponentially weighting factor whose value is positive close to 1.

2. **Algorithm**

- depends on the cost function used

- *convergence of the algorithm* : Will the coefficients of the adaptive filter converge to the desired values? Is the algorithm stable? Global convergence or local convergence?

- *rate of convergence* : This corresponds to the time required for the algorithm to converge to the optimum least squares/Wiener solution.
- *misadjustment* : excess mean square error (MSE) over the minimum MSE produced by the Wiener filter, mathematically it is defined as

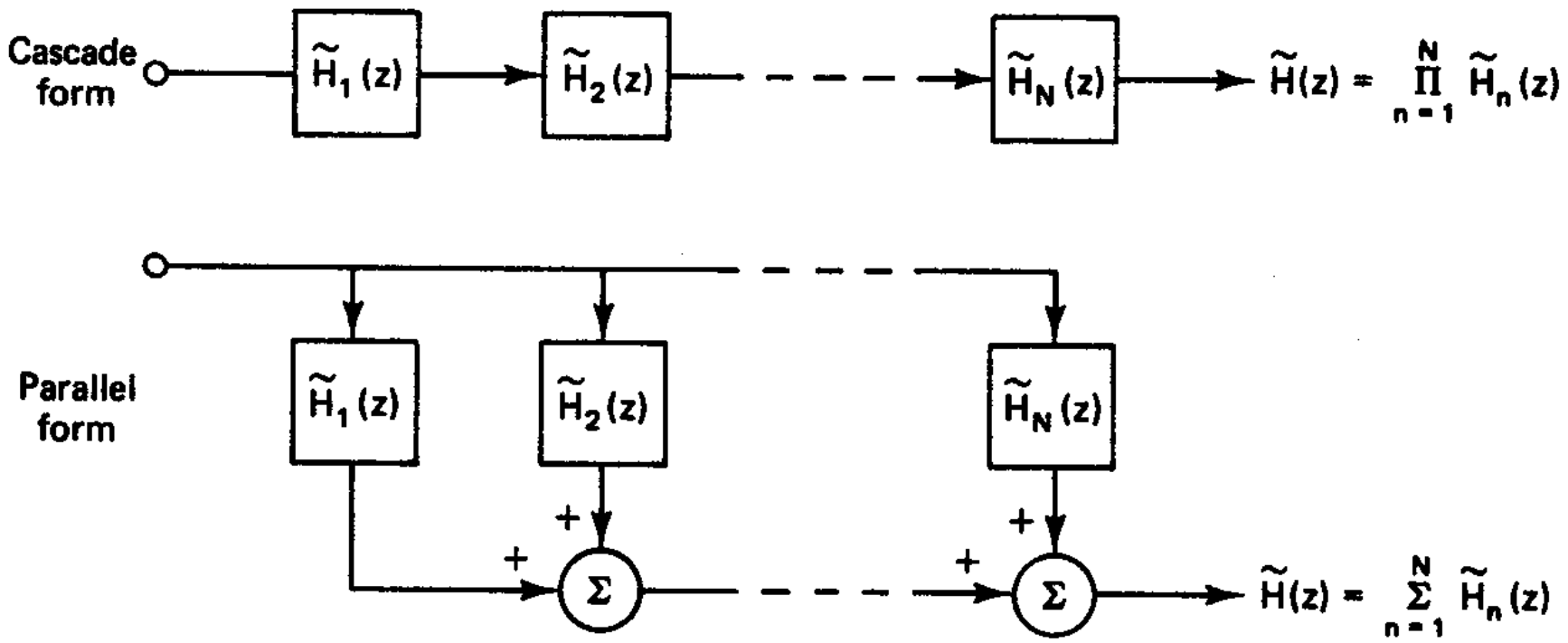
$$M = \frac{\lim_{k\to\infty} E\{e^2(k)\} - \varepsilon_{\min}}{\varepsilon_{\min}} \qquad (4.1)$$

(This is a performance measure for algorithms that use the minimum MSE criterion)

- *tracking capability* : This refers to the ability of the algorithm to track statistical variations in a nonstationary environment.
- *computational requirement* : number of operations, memory size, investment required to program the algorithm on a computer.
- *robustness* : This refers to the ability of the algorithm to operate satisfactorily with ill-conditioned data, e.g. very noisy environment, change in signal and/or noise models

3. Structure

- structure and algorithm are inter-related, choice of structures is based on quantization errors, ease of implementation, computational complexity, etc.
- four commonly used structures are direct form, cascade form, parallel form, and lattice structure. Advantages of lattice structures include simple test for filter stability, modular structure and low sensitivity to quantization effects.



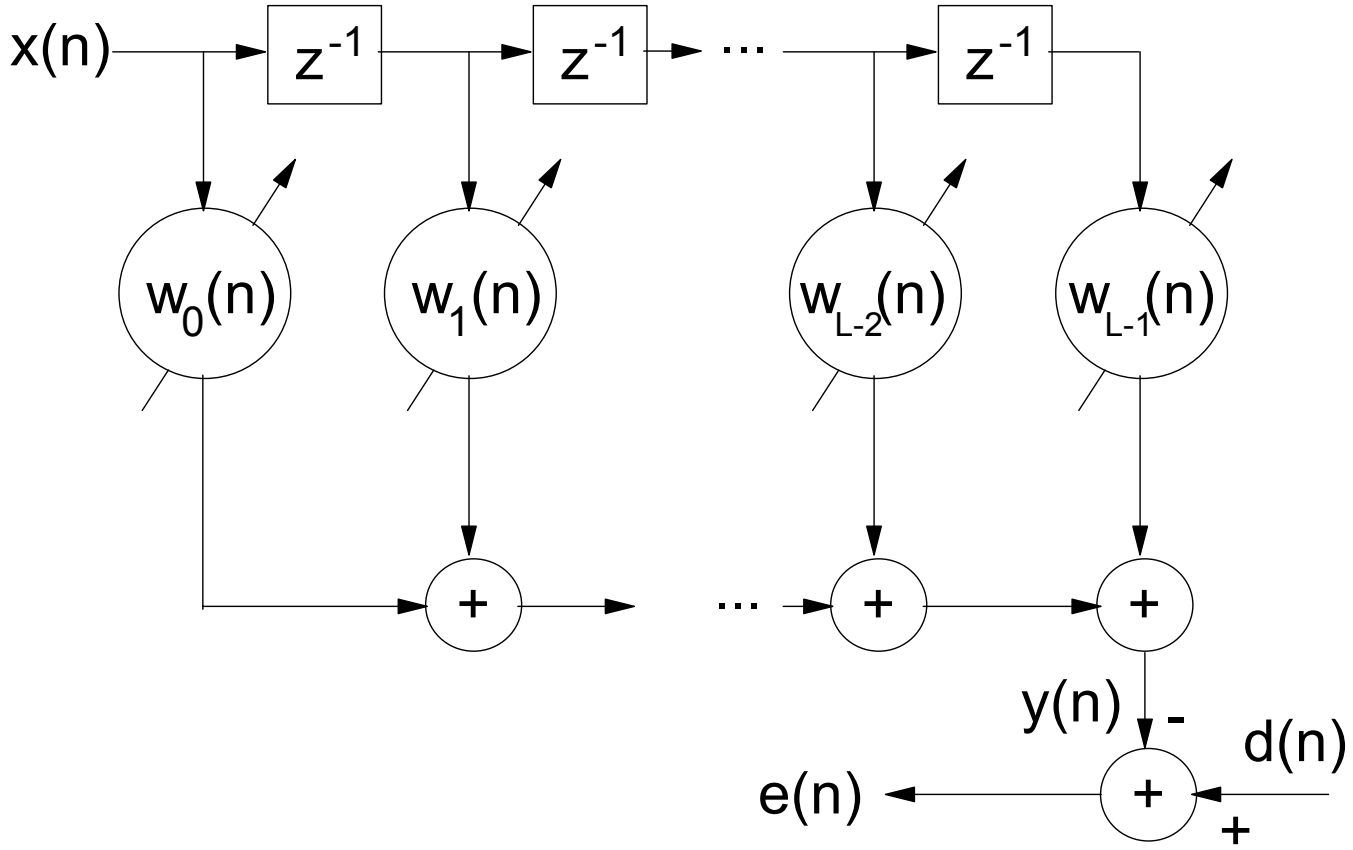
e.g.,

$$H(z) = \frac{B_2 z^2}{z^2 + A_1 z + A_0} = \frac{C_1 z}{(z - p_1)} \cdot \frac{C_2 z}{(z - p_2)} = \frac{D_1 z + E_1}{z - p_1} + \frac{D_2 z + E_2}{z - p_2}$$

Q. Can you see an advantage of using cascade or parallel form?

Commonly Used Methods for Minimizing MSE

For simplicity, it is assumed that the adaptive filter is of causal FIR type and is implemented in direct form. Therefore, its system block diagram is



The error signal at time n is given by

$$e(n) = d(n) - y(n) \tag{4.2}$$

where

$$y(n) = \sum_{i=0}^{L-1} w_i(n)x(n-i) = \underline{W}(n)^T \underline{X}(n),$$

$$\underline{W}(n) = [w_0(n) \quad w_1(n) \quad \cdots w_{L-2}(n) \quad w_{L-1}(n)]^T$$

$$\underline{X}(n) = [x(n) \quad x(n-1) \quad \cdots x(n-L+2) \quad x(n-L+1)]^T$$

Recall that minimizing the $E\{e^2(n)\}$ will give the Wiener solution in optimal filtering, it is desired that

$$\lim_{n\to\infty} \underline{W}(n) = \underline{W}_{MMSE} = (\underline{R}_{xx})^{-1} \cdot \underline{R}_{dx} \tag{4.3}$$

In adaptive filtering, the Wiener solution is found through an iterative procedure,

$$\underline{W}(n+1) = \underline{W}(n) + \Delta\underline{W}(n) \tag{4.4}$$

where $\Delta\underline{W}(n)$ is an incrementing vector.

Two common gradient searching approaches for obtaining the Wiener filter are

1. **Newton Method**

$$\Delta \underline{W}(n) = \mu \underline{R}_{xx}^{-1} \cdot -\left(\frac{\partial E\{e^2(n)\}}{\partial \underline{W}(n)} \right) \tag{4.5}$$

where μ is called the *step size*. It is a positive number that controls the convergence rate and stability of the algorithm. The adaptive algorithm becomes

$$\begin{aligned} \underline{W}(n+1) &= \underline{W}(n) - \mu \underline{R}_{xx}^{-1} \cdot \frac{\partial E\{e^2(n)\}}{\partial \underline{W}(n)} \\ &= \underline{W}(n) - \mu \underline{R}_{xx}^{-1} \cdot 2\left(\underline{R}_{xx} \underline{W}(n) - \underline{R}_{dx}\right) \\ &= (1-2\mu)\underline{W}(n) + 2\mu \underline{R}_{xx}^{-1} \underline{R}_{dx} \\ &= (1-2\mu)\underline{W}(n) + 2\mu \underline{W}_{MMSE} \end{aligned} \tag{4.6}$$

Solving the equation, we have

$$\underline{W}(n) = \underline{W}_{MMSE} + (1-2\mu)^n (\underline{W}(0) - \underline{W}_{MMSE}) \tag{4.7}$$

where $\underline{W}(0)$ is the initial value of $\underline{W}(n)$. To ensure

$$\lim_{n\to\infty} \underline{W}(n) = \underline{W}_{MMSE} \tag{4.8}$$

the choice of μ should be

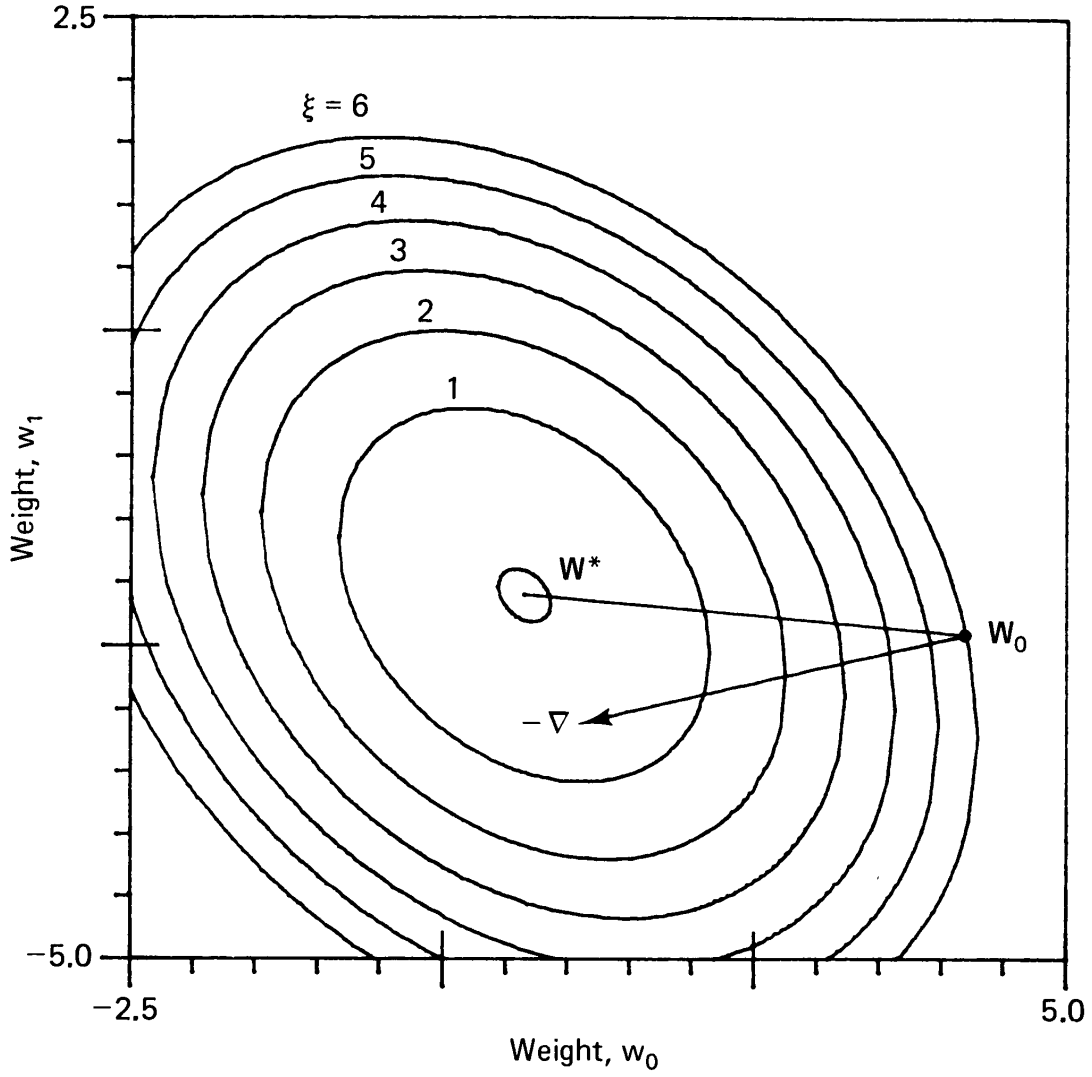
$$-1 < |\, 1-2\mu \,| < 1 \Rightarrow 0 < \mu < 1 \tag{4.9}$$

In particular, when $\mu = 0.5$, we have

$$\underline{W}(1) = \underline{W}_{MMSE} + (1-2\cdot 0.5)^1 (\underline{W}(0) - \underline{W}_{MMSE}) = \underline{W}_{MMSE} \tag{4.10}$$

The weights jump form any initial $\underline{W}(0)$ to the optimum setting $\underline{W}_{MMSE}$ in a single step.

An example of the Newton method with $\mu = 0.5$ and 2 weights is illustrated below.



2. Steepest Descent Method

$$\Delta \underline{W}(n) = -\mu \left(\frac{\partial E\{e^2(n)\}}{\partial \underline{W}(n)} \right) \tag{4.11}$$

Thus

$$\begin{aligned} \underline{W}(n+1) &= \underline{W}(n) - \mu \frac{\partial E\{e^2(n)\}}{\partial \underline{W}(n)} \\ &= \underline{W}(n) - \mu \cdot 2\left(\underline{R}_{xx}\underline{W}(n) - \underline{R}_{dx}\right) \\ &= (\underline{I} - 2\mu \underline{R}_{xx})\underline{W}(n) + 2\mu \underline{R}_{xx}\underline{W}_{MMSE} \\ &= (\underline{I} - 2\mu \underline{R}_{xx})(\underline{W}(n) - \underline{W}_{MMSE}) + \underline{W}_{MMSE} \end{aligned} \tag{4.12}$$

where $\underline{I}$ is the LxL identity matrix. Denote

$$\underline{V}(n) = \underline{W}(n) - \underline{W}_{MMSE} \tag{4.13}$$

We have

$$\underline{V}(n+1) = (\underline{I} - 2\mu \underline{R}_{xx})\underline{V}(n) \tag{4.14}$$

Using the fact that $\underline{R}_{xx}$ is symmetric and real, it can be shown that

$$\underline{R}_{xx} = \underline{Q} \cdot \underline{\Lambda} \cdot \underline{Q}^{-1} = \underline{Q} \cdot \underline{\Lambda} \cdot \underline{Q}^{T} \tag{4.15}$$

where the *modal matrix* $\underline{Q}$ is orthonormal. The columns of $\underline{Q}$, which are the L eigenvectors of $\underline{R}_{xx}$, are mutually orthogonal and normalized. Notice that $\underline{Q}^{-1} = \underline{Q}^{T}$. While $\underline{\Lambda}$ is the so-called *spectral matrix* and all its elements are zero except for the main diagonal, whose elements are the set of eigenvalues of $\underline{R}_{xx}$, $\lambda_1, \lambda_2, \cdots, \lambda_L$. It has the form

$$\underline{\Lambda} = \begin{bmatrix} \lambda_1 & 0 & \cdots & & 0 \\ 0 & \lambda_2 & & & \\ \vdots & & \ddots & & \vdots \\ & & & & 0 \\ 0 & & \cdots & 0 & \lambda_L \end{bmatrix} \tag{4.16}$$

It can be proved that the eigenvalues of $\underline{R}_{xx}$ are all real and greater or equal to zero. Using these results and let

$$\underline{V}(n) = \underline{Q} \cdot \underline{U}(n) \tag{4.17}$$

We have

$$\begin{aligned} \underline{Q} \cdot \underline{U}(n+1) &= (\underline{I} - 2\mu \underline{R}_{xx}) \underline{Q} \cdot \underline{U}(n) \Rightarrow \\ \underline{U}(n+1) &= \underline{Q}^{-1} (\underline{I} - 2\mu \underline{R}_{xx}) \underline{Q} \cdot \underline{U}(n) \\ &= \left(\underline{Q}^{-1} \cdot \underline{I} \cdot \underline{Q} - 2\mu \underline{Q}^{-1} \cdot \underline{R}_{xx} \cdot \underline{Q} \right) \underline{U}(n) \\ &= \left(\underline{I} - 2\mu \underline{\Lambda} \right) \underline{U}(n) \end{aligned} \tag{4.18}$$

The solution is

$$\underline{U}(n) = \left(\underline{I} - 2\mu \underline{\Lambda} \right)^n \underline{U}(0) \tag{4.19}$$

where $\underline{U}(0)$ is the initial value of $\underline{U}(n)$. Thus the steepest descent algorithm is stable and convergent if

$$\lim_{n \to \infty} \left(\underline{I} - 2\mu \underline{\Lambda} \right)^n = \underline{0}$$

or

$$\begin{bmatrix} \lim_{n\to\infty}(1-2\mu\lambda_1)^n & 0 & \cdots & 0 \\ 0 & \lim_{n\to\infty}(1-2\mu\lambda_2)^n & & \vdots \\ \vdots & & \ddots & \\ & & & 0 \\ 0 & \cdots & 0 & \lim_{n\to\infty}(1-2\mu\lambda_L)^n \end{bmatrix} = \underline{0} \qquad (4.20)$$

which implies

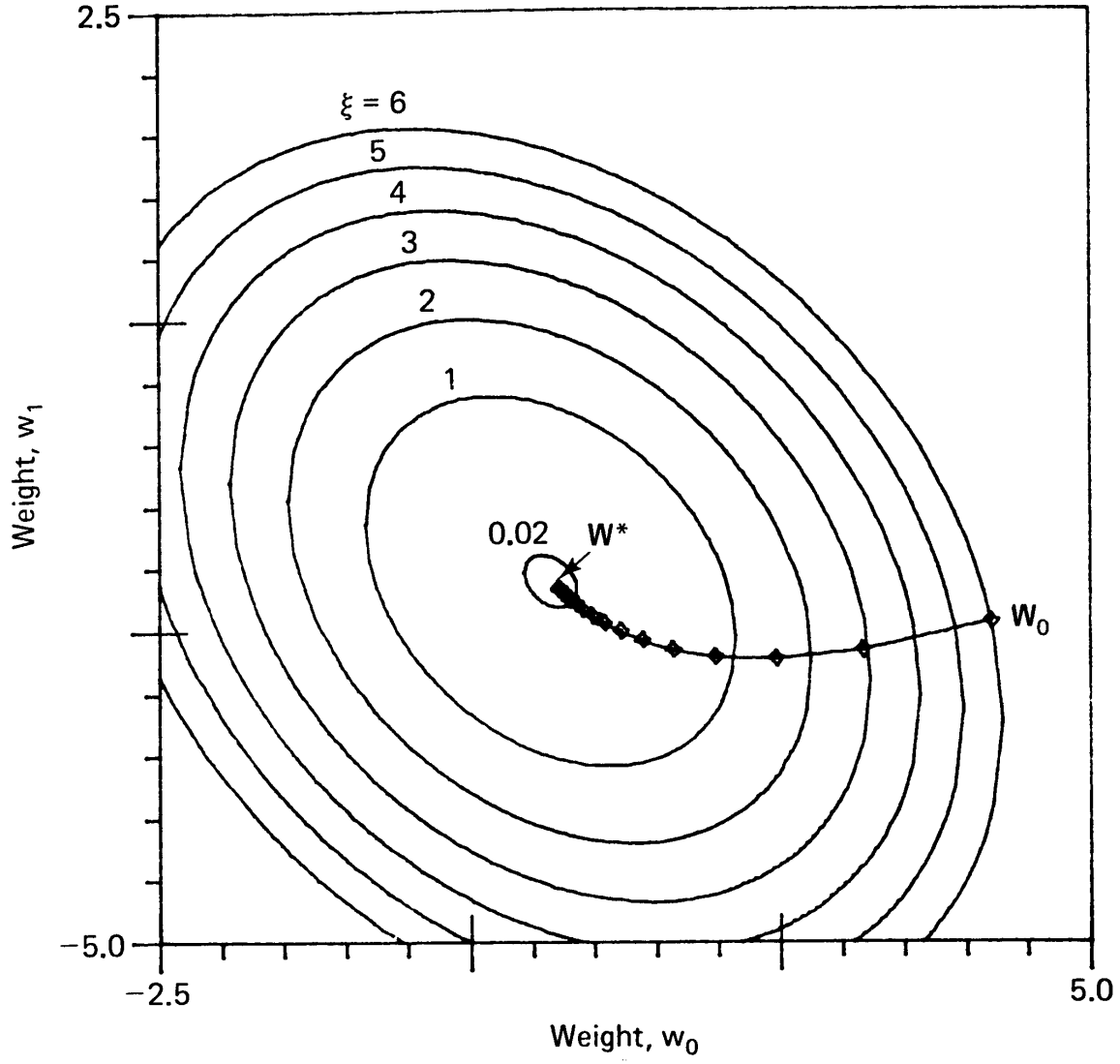
$$|1-2\mu\lambda_{\max}| < 1 \Rightarrow 0 < \mu < \frac{1}{\lambda_{\max}} \qquad (4.21)$$

where $\lambda_{\max}$ is the largest eigenvalue of $\underline{R}_{xx}$.

If this condition is satisfied, it follows that

$$\lim_{n\to\infty} \underline{U}(n) = \underline{0} \Rightarrow \lim_{n\to\infty} \underline{Q}^{-1} \cdot \underline{V}(n) = \lim_{n\to\infty} \underline{Q}^{-1} \cdot \left(\underline{W}(n) - \underline{W}_{MMSE}\right) \Rightarrow \underline{0}$$
$$\Rightarrow \lim_{n\to\infty} \underline{W}(n) = \underline{W}_{MMSE} \qquad (4.22)$$

An illustration of the steepest descent method with two weights and $\mu = 0.3$ is given as below.



Remarks:

- Steepest descent method is simpler than the Newton method since no matrix inversion is required.
- The convergence rate of Newton method is much faster than that of the steepest descent method.
- When the performance surface is unimodal, $\underline{W}(0)$ can be arbitrarily chosen. If it is multimodal, good initial values of $\underline{W}(0)$ is necessary in order for global minimization.
- However, both methods require exact values of $\underline{R}_{xx}$ and $\underline{R}_{dx}$ which are not commonly available in practical applications.

Widrow's Least Mean Square (LMS) Algorithm

A. Optimization Criterion

To minimize the mean square error $E\{e^2(n)\}$

B. Adaptation Procedure

It is an approximation of the steepest descent method where the expectation operator is ignored, i.e.,

$$\frac{\partial E\{e^2(n)\}}{\partial \underline{W}(n)} \text{ is replaced by } \frac{\partial e^2(n)}{\partial \underline{W}(n)}$$

The LMS algorithm is therefore:

$$\underline{W}(n+1) = \underline{W}(n) - \mu \frac{\partial e^2(n)}{\partial \underline{W}(n)}$$

$$= \underline{W}(n) - \mu \frac{\partial e^2(n)}{\partial e(n)} \cdot \frac{\partial e(n)}{\partial \underline{W}(n)}$$

$$= \underline{W}(n) - 2\mu e(n) \cdot \frac{\partial \left[d(n) - \underline{W}^T(n) \cdot \underline{X}(n) \right]}{\partial \underline{W}(n)}, \qquad \frac{\partial \underline{A}^T \cdot \underline{B}}{\partial \underline{A}} = \underline{B}$$

$$= \underline{W}(n) + 2\mu e(n) \underline{X}(n)$$

or

$$w_i(n+1) = w_i(n) + 2\mu e(n) x(n-i), \qquad i = 0,1,\cdots,L-1 \qquad (4.23)$$

C. **Advantages**

- low computational complexity
- simple to implement
- allow real-time operation
- does not need statistics of signals, i.e., $\underline{R}_{xx}$ and $\underline{R}_{dx}$

D. **Performance Surface**

The mean square error function or performance surface is identical to that in the Wiener filtering:

$$E\{e^2(n)\} = \varepsilon_{\min} + \left(\underline{W}(n) - \underline{W}_{MMSE}\right)^T \underline{R}_{xx} \left(\underline{W}(n) - \underline{W}_{MMSE}\right) \qquad (4.24)$$

where $\underline{W}(n)$ is the adaptive filter coefficient vector at time n.

E. **Performance Analysis**

Two important performance measures in LMS algorithms are *rate of convergence* & *misadjustment* (relates to steady state filter weight variance).

1. **Convergence Analysis**

For ease of analysis, it is assumed that $\underline{W}(n)$ is independent of $\underline{X}(n)$. Taking expectation on both sides of the LMS algorithm, we have

$$\begin{aligned} E\{\underline{W}(n+1)\} &= E\{\underline{W}(n)\} + 2\mu E\{e(n)\underline{X}(n)\} \\ &= E\{\underline{W}(n)\} + 2\mu E\left\{d(n)\underline{X}(n) - \underline{X}(n)\cdot(\underline{X}^T(n)\underline{W}(n))\right\} \\ &= E\{\underline{W}(n)\} + 2\mu\underline{R}_{dx} - 2\mu\underline{R}_{xx}E\{\underline{W}(n)\} \\ &= (\underline{I} - 2\mu\underline{R}_{xx})E\{\underline{W}(n)\} + 2\mu\underline{R}_{xx}\underline{W}_{MMSE} \end{aligned} \qquad (4.25)$$

which is very similar to the adaptive equation (4.12) in the steepest descent method.

Following the previous derivation, $\underline{W}(n)$ will converge to the Wiener filter weights in the *mean* sense if

$$\begin{bmatrix} \lim_{n\to\infty}(1-2\mu\lambda_1)^n & 0 & \cdots & 0 \\ 0 & \lim_{n\to\infty}(1-2\mu\lambda_2)^n & & \vdots \\ \vdots & & \ddots & \\ & & & 0 \\ 0 & \cdots & 0 & \lim_{n\to\infty}(1-2\mu\lambda_L)^n \end{bmatrix} = \underline{0}$$

$$\Rightarrow |\,1-2\mu\lambda_i\,| < 1, \qquad i = 1,2,\cdots,L$$

$$\Rightarrow 0 < \mu < \frac{1}{\lambda_{\max}} \tag{4.26}$$

Define geometric ratio of the pth term as

$$r_p = 1-2\mu\lambda_p, \qquad p = 1,2,\cdots,L \tag{4.27}$$

It is observed that each term in the main diagonal forms a geometric series $\{1, r_p^1, r_p^2, \cdots, r_p^{n-1}, r_p^n, r_p^{n+1}, \cdots\}$.

Exponential function can be fitted to approximate each geometric series:

$$r_p \approx \exp\left(-\frac{1}{\tau_p}\right) \Rightarrow \left\{r_p^n\right\} \approx \left\{\exp\left(-\frac{n}{\tau_p}\right)\right\} \tag{4.28}$$

where τ_p is called the pth *time constant* .

For slow adaptation, i.e., $2\mu\lambda_p << 1$, τ_p is approximated as

$$\begin{aligned} -\frac{1}{\tau_p} &= \ln(1-2\mu\lambda_p) = -2\mu\lambda_p - \frac{(-2\mu\lambda_p)^2}{2} + \frac{(-2\mu\lambda_p)^3}{3} - \cdots \approx -2\mu\lambda_p \\ \Rightarrow \tau_p &\approx \frac{1}{2\mu\lambda_p} \end{aligned} \tag{4.29}$$

Notice that the smaller the time constant the faster the convergence rate. Moreover, the overall convergence is limited by the slowest mode of convergence which in turns stems from the smallest eigenvalue of $\underline{R}_{xx}$, $\lambda_{\min}$.

That is,

$$\tau_{\max} \approx \frac{1}{2\mu\lambda_{\min}} \tag{4.30}$$

In general, the rate of convergence depends on two factors:

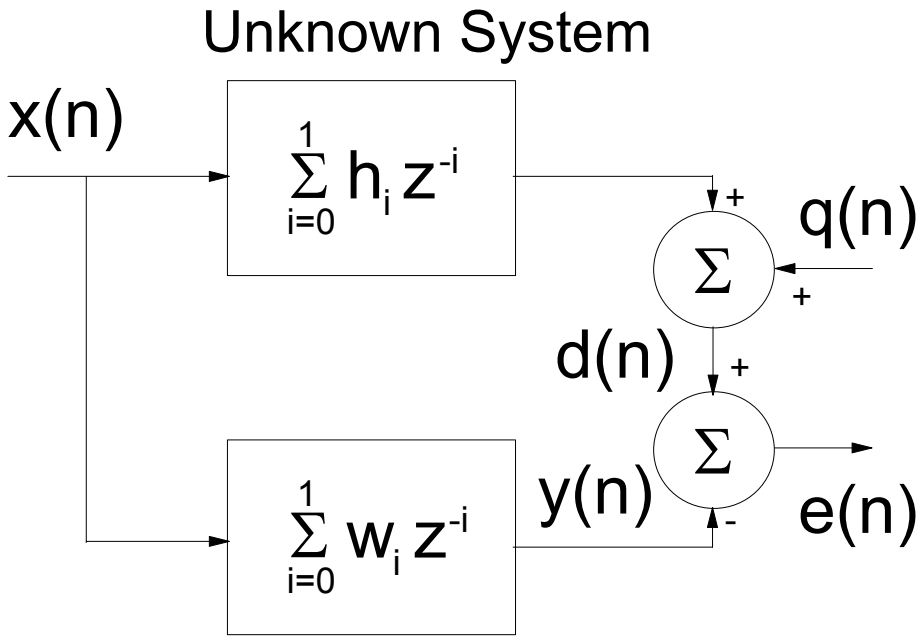
- the *step size* μ : the larger the μ, the faster the convergence rate
- the *eigenvalue spread* of $\underline{R}_{xx}$, $\chi(\underline{R}_{xx})$: the smaller $\chi(\underline{R}_{xx})$, the faster the convergence rate. $\chi(\underline{R}_{xx})$ is defined as

$$\chi(\underline{R}_{xx}) = \frac{\lambda_{\max}}{\lambda_{\min}} \tag{4.31}$$

Notice that $1 \le \chi(\underline{R}_{xx}) < \infty$. It is worthy to note that although $\chi(\underline{R}_{xx})$ cannot be changed, the rate of convergence will be increased if we transform $x(n)$ to another sequence, say, $y(n)$, such that $\chi(\underline{R}_{yy})$ is close to 1.

Example 4.1

An Illustration of eigenvalue spread for LMS algorithm is shown as follows.



$d(n) = h_0x(n) + h_1x(n-1) + q(n)$

$y(n) = w_0(n)x(n) + w_1(n)x(n-1)$

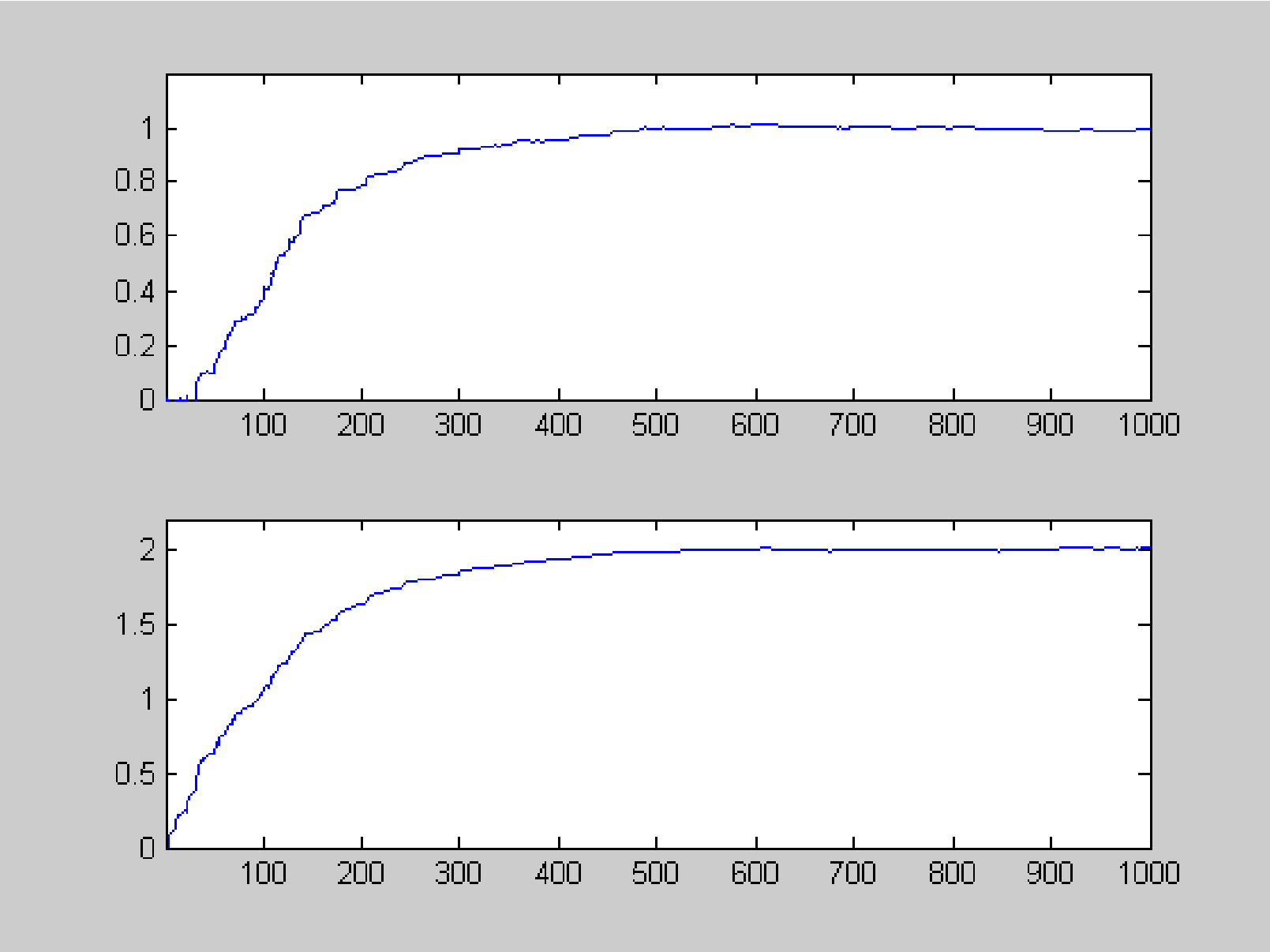
$e(n) = d(n) – y(n) = d(n) – w_0(n)x(n) – w_1(n)x(n-1)$

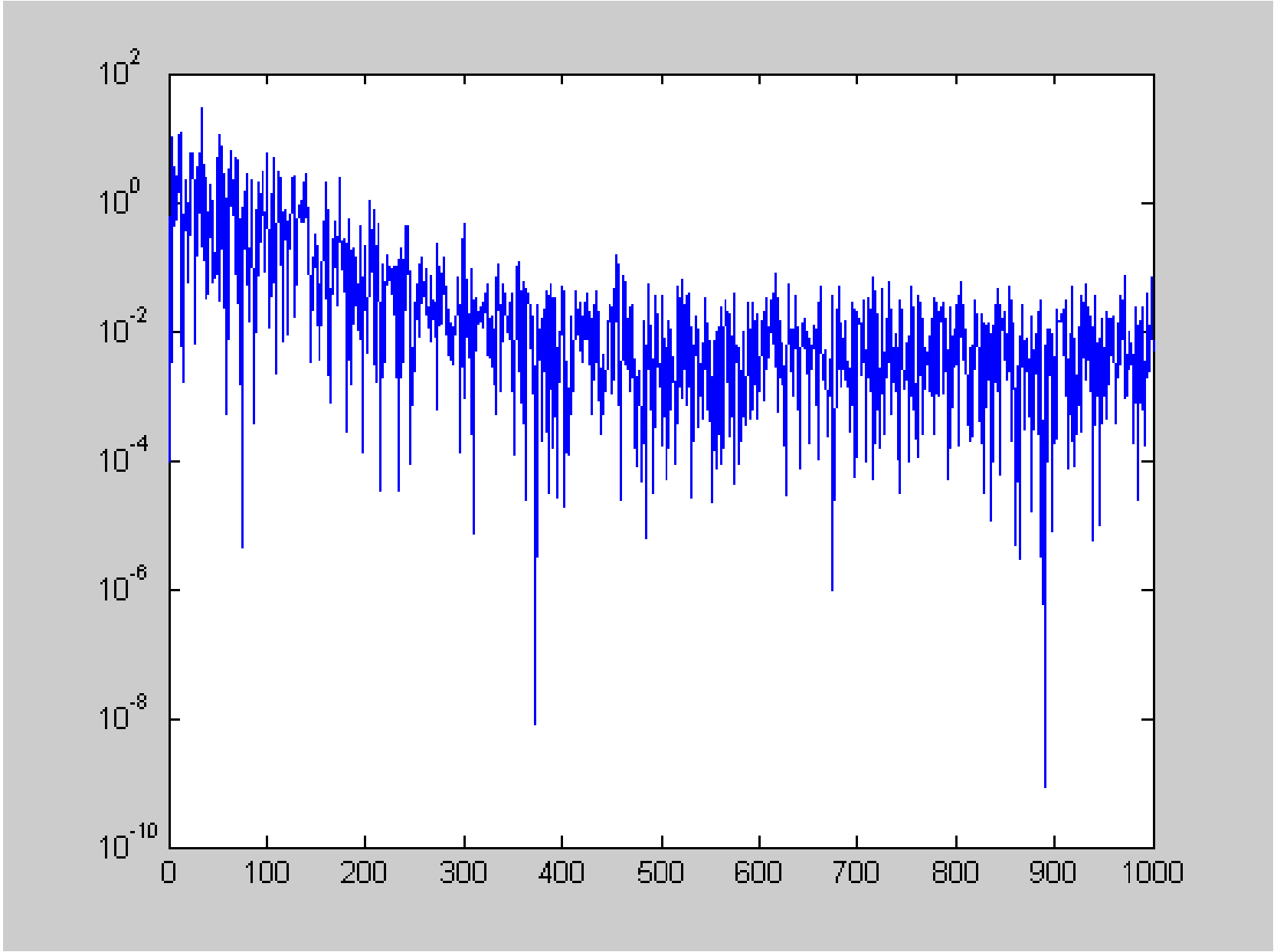
$w_0(n+1) = w_0(n) + 2ue(n)x(n)$

$w_1(n+1) = w_1(n) + 2ue(n)x(n-1)$

```
; file name is es.m
clear all
N=1000;                          % number of sample is 1000
np = 0.01;                       % noise power is 0.01
sp = 1;                          % signal power is 1 which implies SNR = 20dB
h=[1 2];                         % unknown impulse response
x = sqrt(sp).*randn(1,N);
d = conv(x,h);
d = d(1:N) + sqrt(np).*randn(1,N);

w0(1) = 0;                       % initial filter weights are 0
w1(1) = 0;

mu = 0.005;                      % step size is fixed at 0.005


y(1) = w0(1)*x(1);               % iteration at “n=0”
e(1) = d(1) - y(1);              % separate because “x(0)” is not defined
w0(2) = w0(1) + 2*mu*e(1)*x(1);
w1(2) = w1(1);
```

```
for n=2:N                          % the LMS algorithm
    y(n) = w0(n)*x(n) + w1(n)*x(n-1);
    e(n) = d(n) - y(n);
    w0(n+1) = w0(n) + 2*mu*e(n)*x(n);
    w1(n+1) = w1(n) + 2*mu*e(n)*x(n-1);
end

n = 1:N+1;
subplot(2,1,1)
plot(n,w0)                         % plot filter weight estimate versus time
axis([1 1000 0 1.2])
subplot(2,1,2)
plot(n,w1)
axis([1 1000 0 2.2])
figure(2)
subplot(1,1,1)
n = 1:N;
semilogy(n,e.*e);                  % plot square error versus time
```


1
0.8
0.6
0.4
0.2
0
100
200
300
400
500
600
700
800
900
1000
2
1.5
1
0.5
0
100
200
300
400
500
600
700
800
900
1000


10^{2}
10^{0}
10^{-2}
10^{-4}
10^{-6}
10^{-8}
10^{-10}
0
100
200
300
400
500
600
700
800
900
1000

Note that both filter weights converge at similar speed because the eigenvalues of the R_{xx} are identical:

Recall

$$R_{xx} = \begin{bmatrix} R_{xx}(0) & R_{xx}(1) \\ R_{xx}(1) & R_{xx}(0) \end{bmatrix}$$

For white process with unity power, we have

$$R_{xx}(0) = E\{x(n).x(n)\} = 1$$
$$R_{xx}(1) = E\{x(n).x(n-1)\} = 0$$

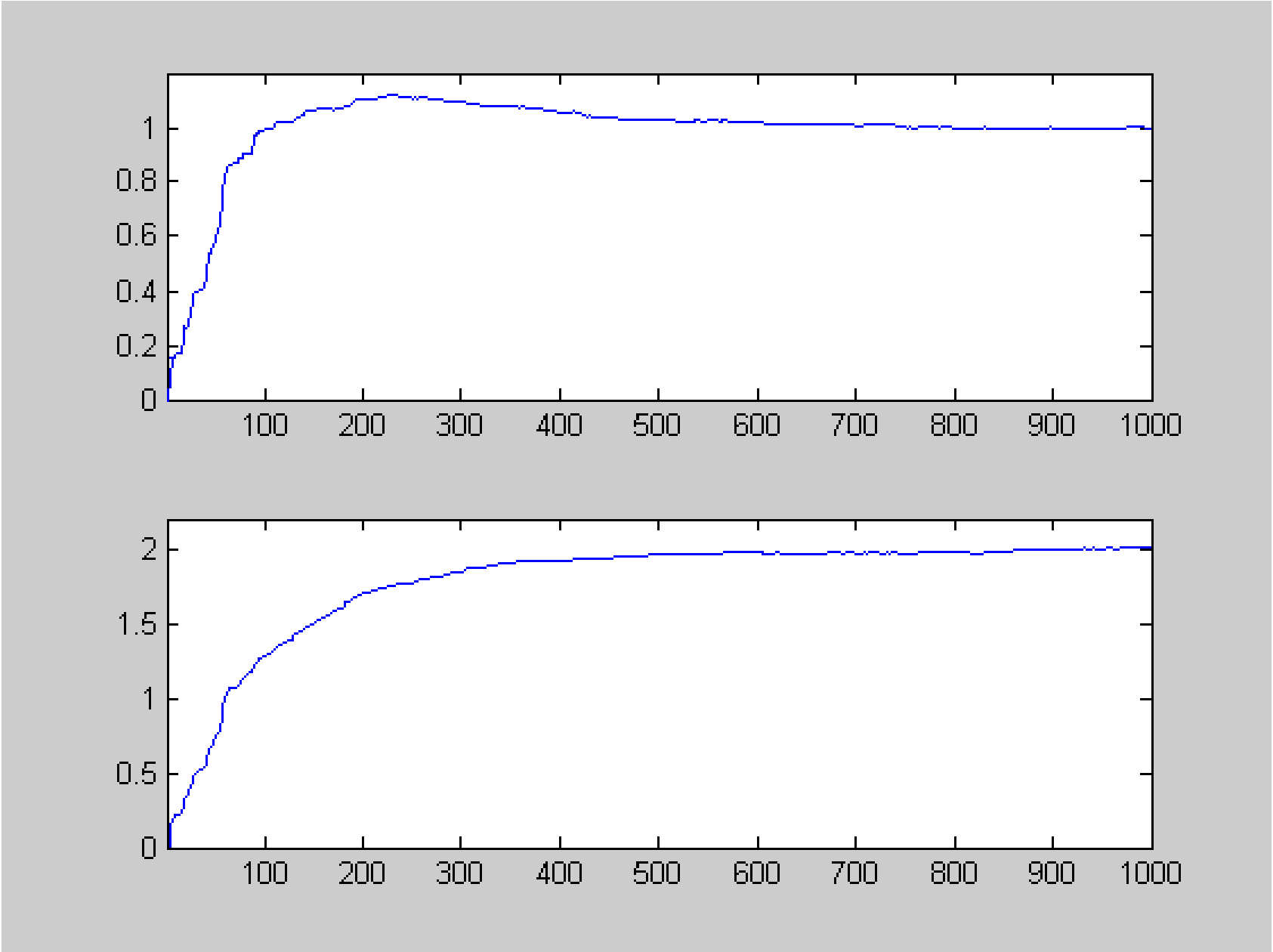
As a result,

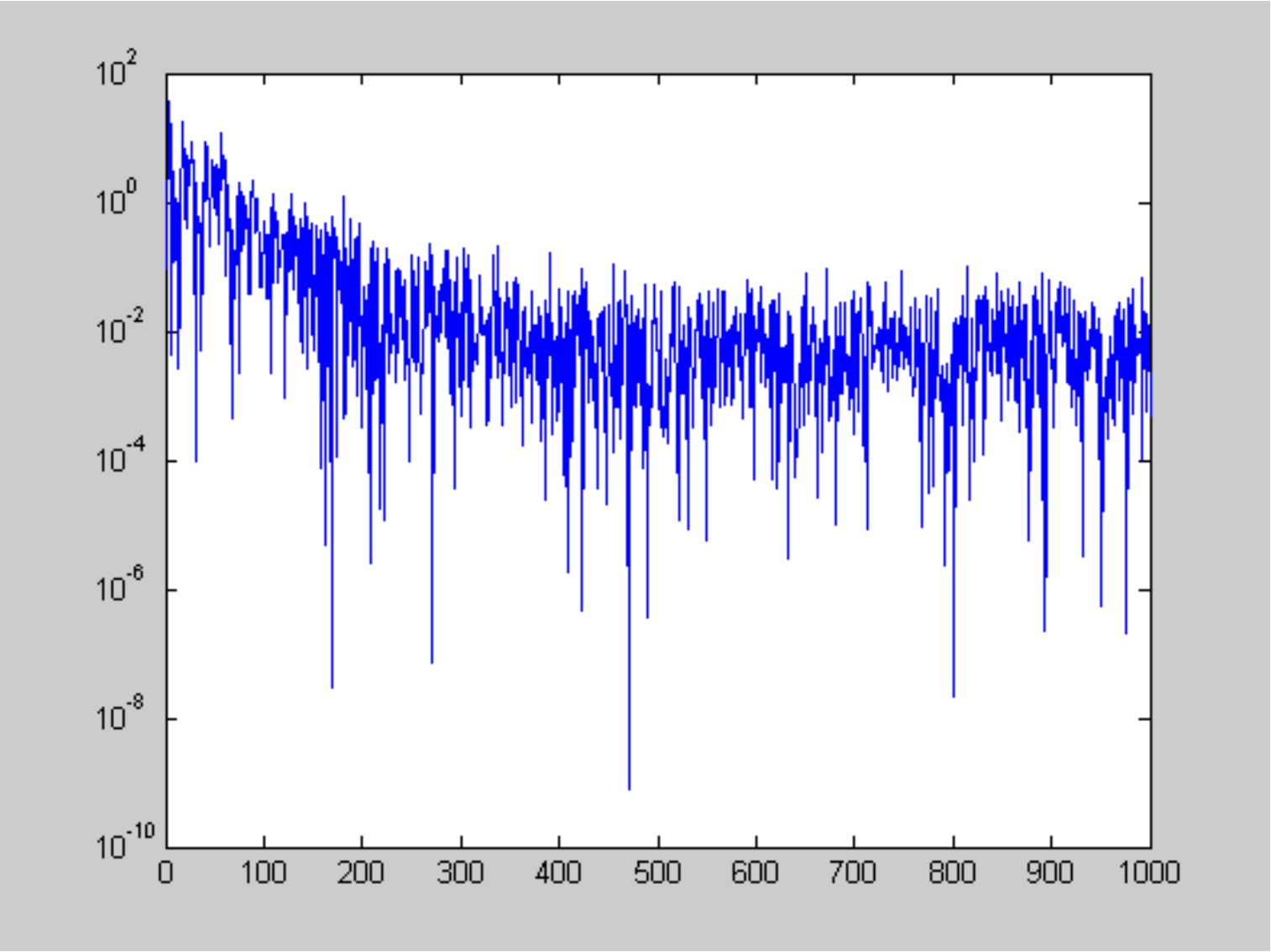
$$R_{xx} = \begin{bmatrix} R_{xx}(0) & R_{xx}(1) \\ R_{xx}(1) & R_{xx}(0) \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

$$\Rightarrow \quad \chi(\underline{R}_{xx}) = 1$$

```
; file name is es1.m
clear all
N=1000;
np = 0.01;
sp = 1;
h=[1 2];
u = sqrt(sp/2).*randn(1,N+1);
x = u(1:N) + u(2:N+1);                    % x(n) is now a MA process with power 1
d = conv(x,h);
d = d(1:N) + sqrt(np).*randn(1,N);

w0(1) = 0;
w1(1) = 0;

mu = 0.005;

y(1) = w0(1)*x(1);
e(1) = d(1) - y(1);
w0(2) = w0(1) + 2*mu*e(1)*x(1);
w1(2) = w1(1);
```

```
for n=2:N
    y(n) = w0(n)*x(n) + w1(n)*x(n-1);
    e(n) = d(n) - y(n);
    w0(n+1) = w0(n) + 2*mu*e(n)*x(n);
    w1(n+1) = w1(n) + 2*mu*e(n)*x(n-1);
end

n = 1:N+1;
subplot(2,1,1)
plot(n,w0)
axis([1 1000 0 1.2])
subplot(2,1,2)
plot(n,w1)
axis([1 1000 0 2.2])
figure(2)
subplot(1,1,1)
n = 1:N;
semilogy(n,e.*e);
```


1
0.8
0.6
0.4
0.2
0
100
200
300
400
500
600
700
800
900
1000
2
1.5
1
0.5
0
100
200
300
400
500
600
700
800
900
1000


10^{2}
10^{0}
10^{-2}
10^{-4}
10^{-6}
10^{-8}
10^{-10}
0
100
200
300
400
500
600
700
800
900
1000

Note that the convergence speed of $w_0(n)$ is slower than that of $w_1(n)$

Investigating the R_{xx} :

$$\begin{aligned} R_{xx}(0) &= E\{x(n).x(n)\} \\ &= E\{(u(n)+u(n-1))\cdot(u(n)+u(n-1))\} \\ &= E\{u^2(n)+u^2(n-1)\} \\ &= 0.5+0.5=1 \end{aligned}$$

$$\begin{aligned} R_{xx}(1) &= E\{x(n).x(n-1)\} \\ &= E\{(u(n)+u(n-1))\cdot(u(n-1)+u(n-2))\} \\ &= E\{u^2(n-1)\}=0.5 \end{aligned}$$

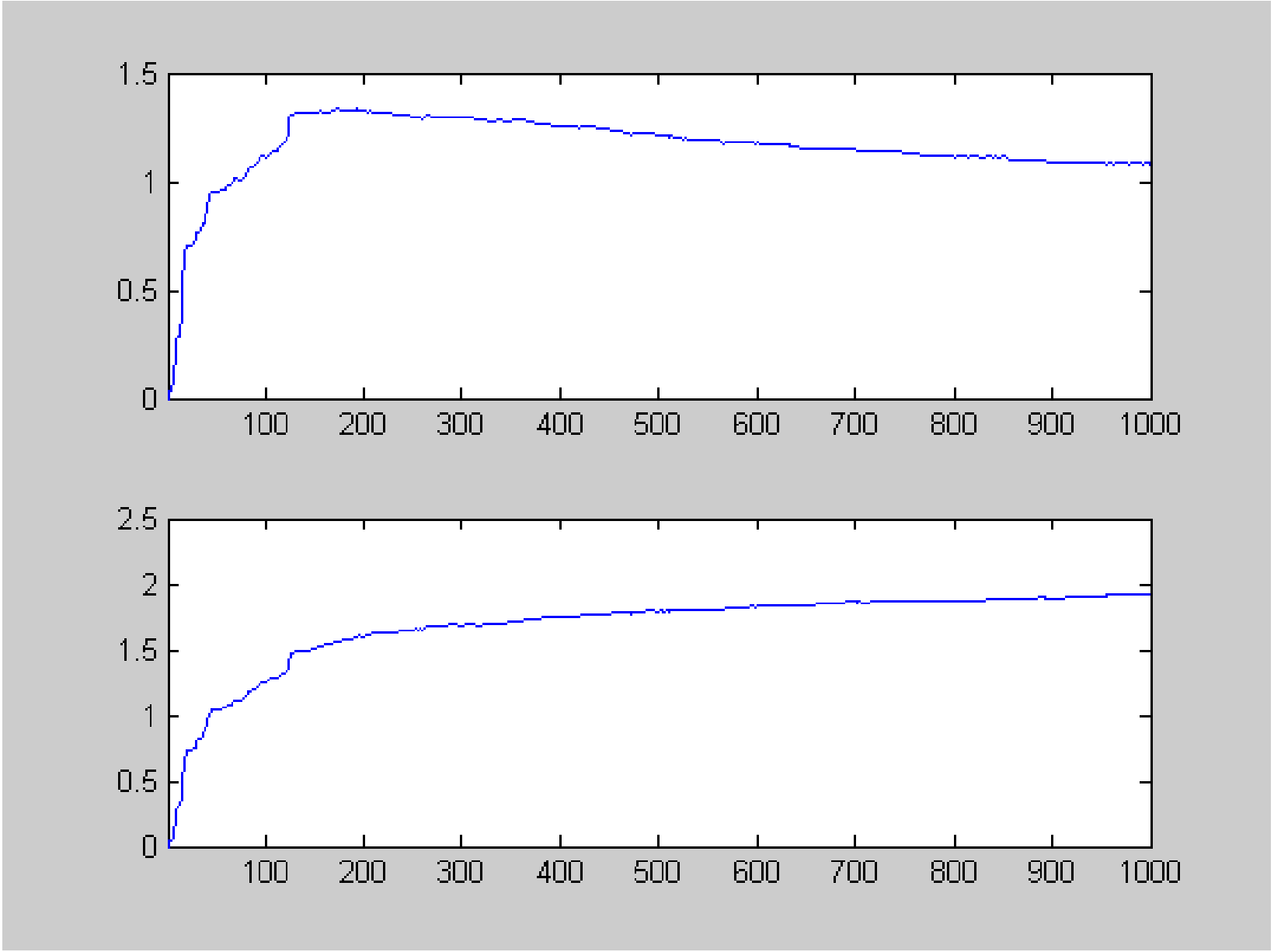
As a result,

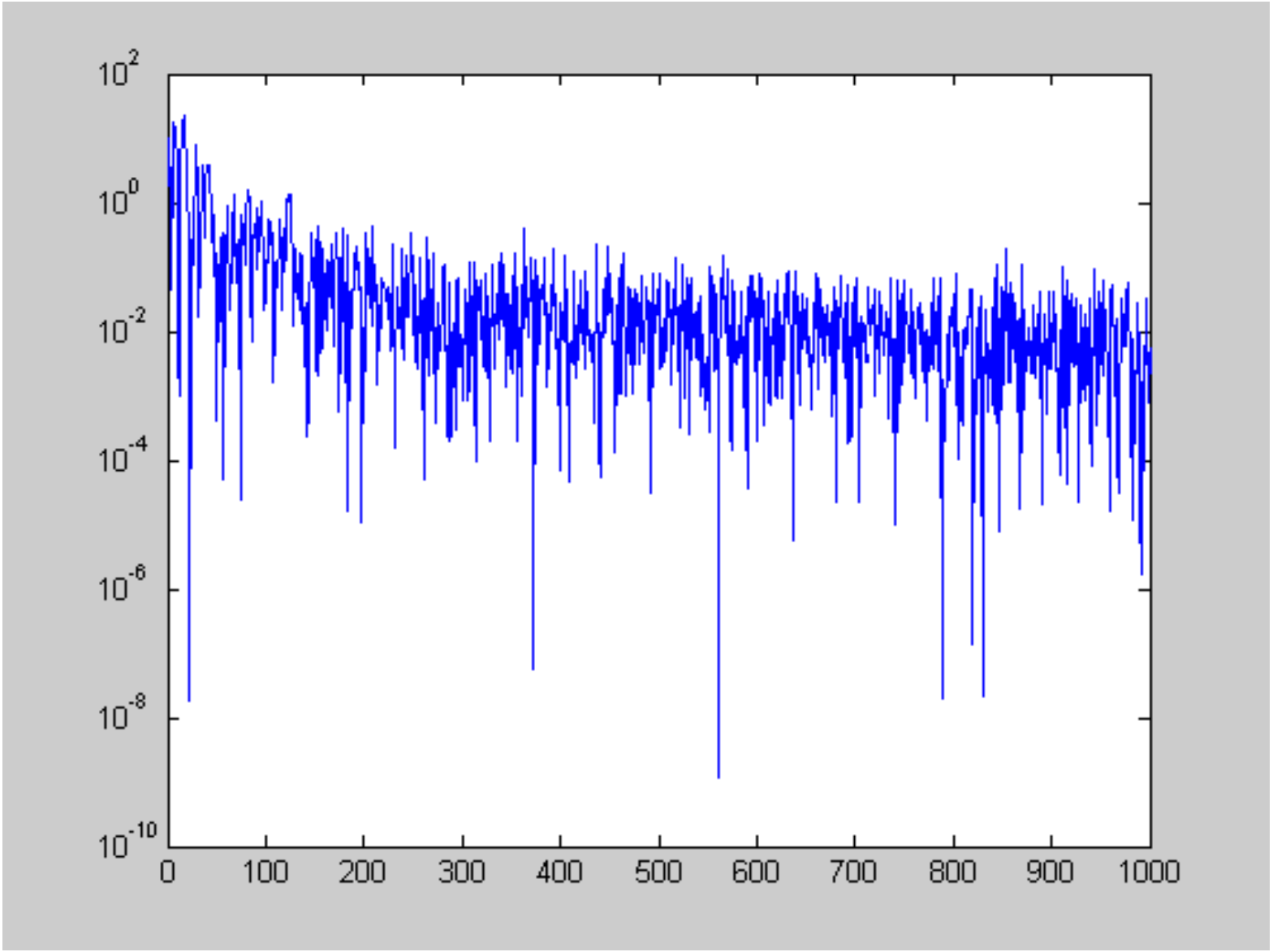
$$R_{xx}=\begin{bmatrix} R_{xx}(0) & R_{xx}(1) \\ R_{xx}(1) & R_{xx}(0) \end{bmatrix}=\begin{bmatrix} 1 & 0.5 \\ 0.5 & 1 \end{bmatrix}$$

$\Rightarrow \ \lambda_{\min}=0.5$ and $\lambda_{\max}=1.5 \ \Rightarrow \chi(\underline{R}_{xx})=3$ (MATLAB command: eig)

```
; file name is es2.m
clear all
N=1000;
np = 0.01;
sp = 1;
h=[1 2];
u = sqrt(sp/5).*randn(1,N+4);
x = u(1:N) + u(2:N+1) + u(3:N+2) + u(4:N+3) +u(5:N+4);     % x(n) is 5th order MA process
d = conv(x,h);
d = d(1:N) + sqrt(np).*randn(1,N);

w0(1) = 0;
w1(1) = 0;

mu = 0.005;

y(1) = w0(1)*x(1);
e(1) = d(1) - y(1);
w0(2) = w0(1) + 2*mu*e(1)*x(1);
w1(2) = w1(1);
```

```
for n=2:N
    y(n) = w0(n)*x(n) + w1(n)*x(n-1);
    e(n) = d(n) - y(n);
    w0(n+1) = w0(n) + 2*mu*e(n)*x(n);
    w1(n+1) = w1(n) + 2*mu*e(n)*x(n-1);
end

n = 1:N+1;
subplot(2,1,1)
plot(n,w0)
axis([1 1000 0 1.5])
subplot(2,1,2)
plot(n,w1)
axis([1 1000 0 2.5])
figure(2)
subplot(1,1,1)
n = 1:N;
semilogy(n,e.*e);
```


1.5
1
0.5
0
100
200
300
400
500
600
700
800
900
1000
2.5
2
1.5
1
0.5
0
100
200
300
400
500
600
700
800
900
1000


10^{2}
10^{0}
10^{-2}
10^{-4}
10^{-6}
10^{-8}
10^{-10}
0
100
200
300
400
500
600
700
800
900
1000

We see that the convergence speeds of both weights are very slow, although that of $w_1(n)$ is faster.

Investigating the R_{xx} :

$$\begin{aligned} R_{xx}(0) &= E\{x(n).x(n)\} \\ &= E\{u^2(n)+u^2(n-1)+u^2(n-2)+u^2(n-3)+u^2(n-4)\} \\ &= 0.2+0.2+0.2+0.2+0.2=1 \\ R_{xx}(1) &= E\{x(n).x(n-1)\}\} \\ &= E\{u^2(n-1)\}+E\{u^2(n-2)\}+E\{u^2(n-3)\}+E\{u^2(n-4)\} \\ &= 0.8 \end{aligned}$$

As a result,

$$R_{xx}=\begin{bmatrix} R_{xx}(0) & R_{xx}(1) \\ R_{xx}(1) & R_{xx}(0) \end{bmatrix}=\begin{bmatrix} 1 & 0.8 \\ 0.8 & 1 \end{bmatrix}$$

$\Rightarrow \ \lambda_{\min}=0.2$ and $\lambda_{\max}=1.8 \ \Rightarrow \chi(\underline{R}_{xx})=9$

2. Misadjustment

Upon convergence, if $\lim_{n\to\infty} \underline{W}(n) = \underline{W}_{MMSE}$, then the minimum MSE will be equal to

$$\varepsilon_{\min} = E\left\{d^2(n)\right\} - \underline{R}_{dx}^T \underline{W}_{MMSE} \tag{4.32}$$

However, this will not occur in practice due to random noise in the weight vector $\underline{W}(n)$. Notice that we have $\lim_{n\to\infty} E\{\underline{W}(n)\} = \underline{W}_{MMSE}$ but not $\lim_{n\to\infty} \underline{W}(n) = \underline{W}_{MMSE}$. The MSE of the LMS algorithm is computed as

$$\begin{aligned} E\{e^2(n)\} &= E\left\{\left(d(n) - \underline{W}(n)^T \underline{X}(n)\right)^2\right\} \\ &= \varepsilon_{\min} + E\left\{\left(\underline{W}(n) - \underline{W}_{MMSE}\right)^T \cdot \left(\underline{X}(n) \cdot \underline{X}(n)^T\right) \cdot \left(\underline{W}(n) - \underline{W}_{MMSE}\right)\right\} \quad (4.33) \\ &= \varepsilon_{\min} + E\left\{\left(\underline{W}(n) - \underline{W}_{MMSE}\right)^T \cdot \underline{R}_{xx} \cdot \left(\underline{W}(n) - \underline{W}_{MMSE}\right)\right\} \end{aligned}$$

The second term of the right hand side at $n \to \infty$ is known as the excess MSE and it is given by

$$\begin{aligned}
\text{excess} \quad \text{MSE} &= \lim_{n\to\infty} E\left\{\left(\underline{W}(n) - \underline{W}_{MMSE}\right)^T \cdot \underline{R}_{xx} \cdot \left(\underline{W}(n) - \underline{W}_{MMSE}\right)\right\} \\
&= \lim_{n\to\infty} E\left\{\underline{V}(n)^T \cdot \underline{R}_{xx} \cdot \underline{V}(n)\right\} \\
&= \lim_{n\to\infty} E\left\{\underline{U}(n)^T \cdot \underline{\Lambda} \cdot \underline{U}(n)\right\} \\
&= \mu\varepsilon_{\min} \sum_{i=0}^{L-1} \lambda_i = \mu\varepsilon_{\min} tr\left[\underline{R}_{xx}\right]
\end{aligned} \tag{4.34}$$

where $tr\left[\underline{R}_{xx}\right]$ is the trace of $\underline{R}_{xx}$ which is equal to the sum of all elements of the principle diagonal:

$$tr\left[\underline{R}_{xx}\right] = L \cdot R_{xx}(0) = L \cdot E\{x^2(n)\} \tag{4.35}$$

As a result, the misadjustment M is given by

$$M = \frac{\lim_{k\to\infty} E\{e^2(k)\} - \varepsilon_{\min}}{\varepsilon_{\min}} = \frac{\mu\varepsilon_{\min} L \cdot E\{x^2(n)\}}{\varepsilon_{\min}} = \mu \cdot L \cdot E\{x^2(n)\} \tag{4.36}$$

which is proportional to the step size, filter length and signal power.

Remarks:

1. There is a tradeoff between fast convergence rate and small mean square error or misadjustment. When μ increases, both the convergence rate and M increase; if μ decreases, both the convergence rate and M decrease.

2.The bound for μ is

$$0 < \mu < \frac{1}{\lambda_{\max}} \tag{4.37}$$

In practice, the signal power of $x(n)$ can generally be estimated more easily than the eigenvalue of $\underline{R}_{xx}$. We also note that

$$\lambda_{\max} \le \sum_{i=1}^{L} \lambda_i = tr[\underline{R}_{xx}] = L \cdot E\{x^2(n)\} \tag{4.38}$$

A more restrictive bound for μ which is much easier to apply thus is

$$0 < \mu < \frac{1}{L \cdot E\{x^2(n)\}} \tag{4.39}$$

Moreover, instead of a fixed value of μ, we can make it time-varying as $\mu(n)$. A design idea of a good $\mu(n)$ is

$$\mu(n) = \begin{cases} \text{large value initially,} & \text{to ensure fast inital convergence rate} \\ \text{small value finally,} & \text{to ensure small misadjustment upon convergence} \end{cases}$$

LMS Variants

1. Normalized LMS (NLMS) algorithm

- the product vector $e(n)\underline{X}(n)$ is modified with respect to the squared Euclidean norm of the tap-input vector $\underline{X}(n)$:

$$\underline{W}(n+1) = \underline{W}(n) + \frac{2\mu}{c + \underline{X}^T(n) \cdot \underline{X}(n)} e(n)\underline{X}(n) \tag{4.40}$$

  where c is a small positive constant to avoid division by zero.

- can also be considered as an LMS algorithm with a time-varying step size:

$$\mu(n) = \frac{\mu}{c + \underline{X}^T(n) \cdot \underline{X}(n)} \tag{4.41}$$

- substituting $c = 0$, it can be shown that the NLMS algorithm converges if $0 < \mu < 0.5 \Rightarrow$ selection of step size in the NLMS is much easier than that of LMS algorithm

2. Sign algorithms

- pilot LMS or signed error or signed algorithm:

$$\underline{W}(n+1) = \underline{W}(n) + 2\mu\,\mathrm{sgn}[e(n)]\underline{X}(n) \tag{4.42}$$

- clipped LMS or signed regressor:

$$\underline{W}(n+1) = \underline{W}(n) + 2\mu e(n)\,\mathrm{sgn}[\underline{X}(n)] \tag{4.43}$$

- zero-forcing LMS or sign-sign:

$$\underline{W}(n+1) = \underline{W}(n) + 2\mu\,\mathrm{sgn}[e(n)]\,\mathrm{sgn}[\underline{X}(n)] \tag{4.44}$$

- their computational complexity is simpler than the LMS algorithm but they are relatively difficult to analyze

3. Leaky LMS algorithm

- the LMS update is modified by the presence of a constant leakage factor γ:

$$\underline{W}(n+1) = \gamma \cdot \underline{W}(n) + 2\mu e(n)\underline{X}(n) \tag{4.45}$$

where $0 < \gamma < 1$.

- operates when $\underline{R}_{xx}$ has zero eigenvalues.

4. Least mean fourth algorithm

- instead of minimizing $E\{e^2(n)\}$, $E\{e^4(n)\}$ is minimized based on LMS approach:

$$\underline{W}(n+1) = \underline{W}(n) + 4\mu e^3(n)\underline{X}(n) \tag{4.46}$$

- can outperform LMS algorithm in non-Gaussian signal and noise conditions

Application Examples

Example 4.2

1. Linear Prediction

Suppose a signal $x(n)$ is a second-order autoregressive (AR) process that satisfies the following difference equation:

$$x(n) = 1.558x(n-1) - 0.81x(n-2) + v(n)$$

where $v(n)$ is a white noise process such that

$$R_{vv}(m) = E\{v(n)v(n+m)\} = \begin{cases} \sigma_v^2, & m = 0 \\ 0, & \text{otherwise} \end{cases}$$

We want to use a two-coefficient LMS filter to predict $x(n)$ by

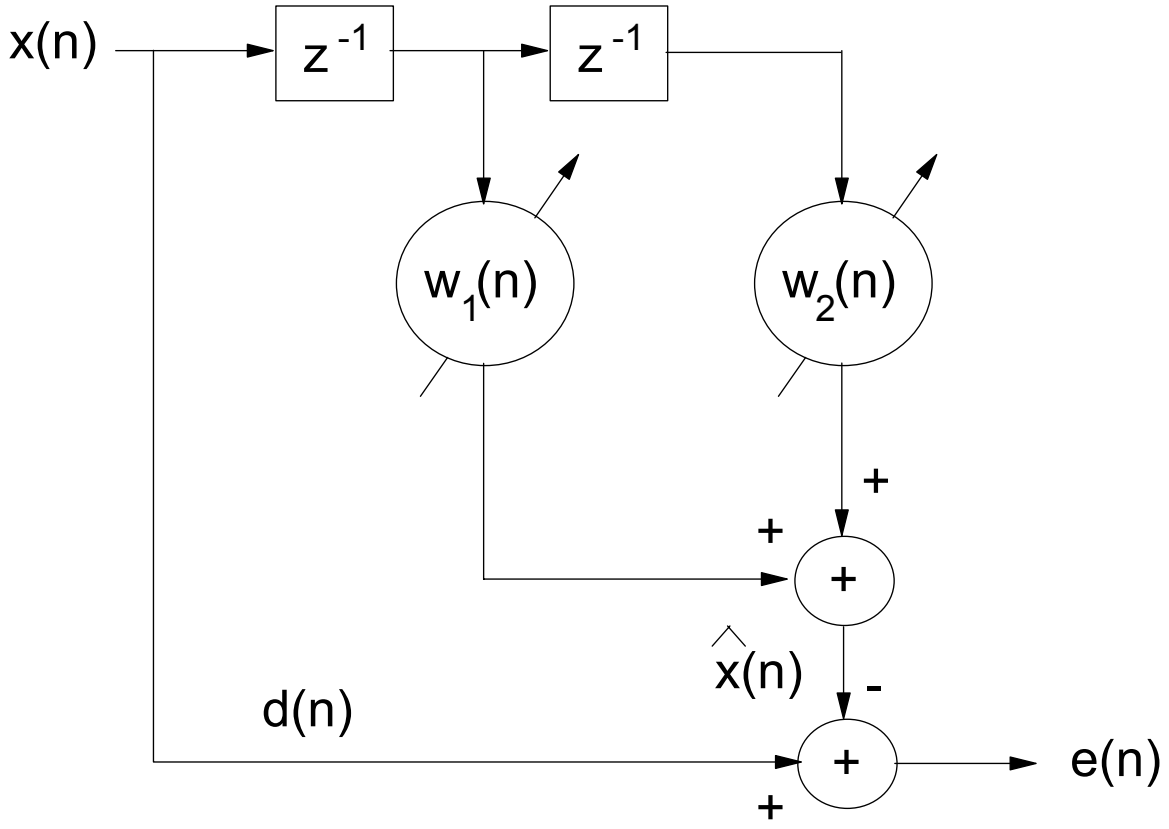
$$\hat{x}(n) = \sum_{i=1}^{2} w_i(n)x(n-i) = w_1(n)x(n-1) + w_2(n)x(n-2)$$

Upon convergence, we desire

$$E\{w_1(n)\} \rightarrow 1.558$$

and

$$E\{w_2(n)\} \rightarrow -0.81$$



The error function or prediction error $e(n)$ is given by

$$e(n) = d(n) - \sum_{i=1}^{2} w_i(n)x(n-i)$$
$$= x(n) - w_1(n)x(n-1) - w_2(n)x(n-2)$$

Thus the LMS algorithm for this problem is

$$w_1(n+1) = w_1(n) - \frac{\mu}{2} \cdot \frac{\partial e^2(n)}{\partial w_1(n)} = w_1(n) - \frac{\mu}{2} \cdot \frac{\partial e^2(n)}{\partial e(n)} \cdot \frac{\partial e(n)}{\partial w_1(n)}$$
$$= w_1(n) + \mu e(n)x(n-1)$$

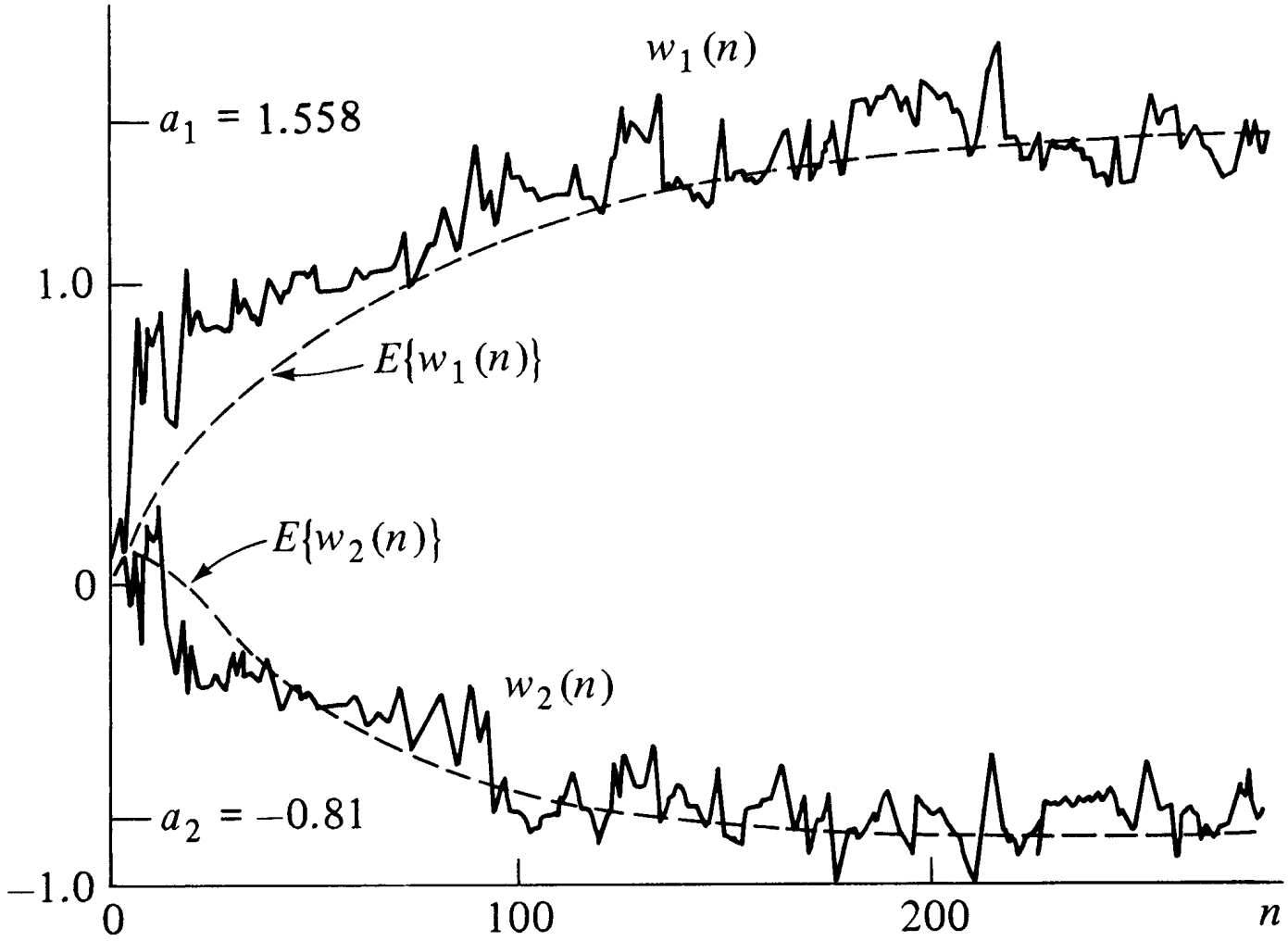
and

$$w_2(n+1) = w_2(n) - \frac{\mu}{2} \cdot \frac{\partial e^2(n)}{\partial w_2(n)}$$
$$= w_2(n) + \mu e(n)x(n-2)$$

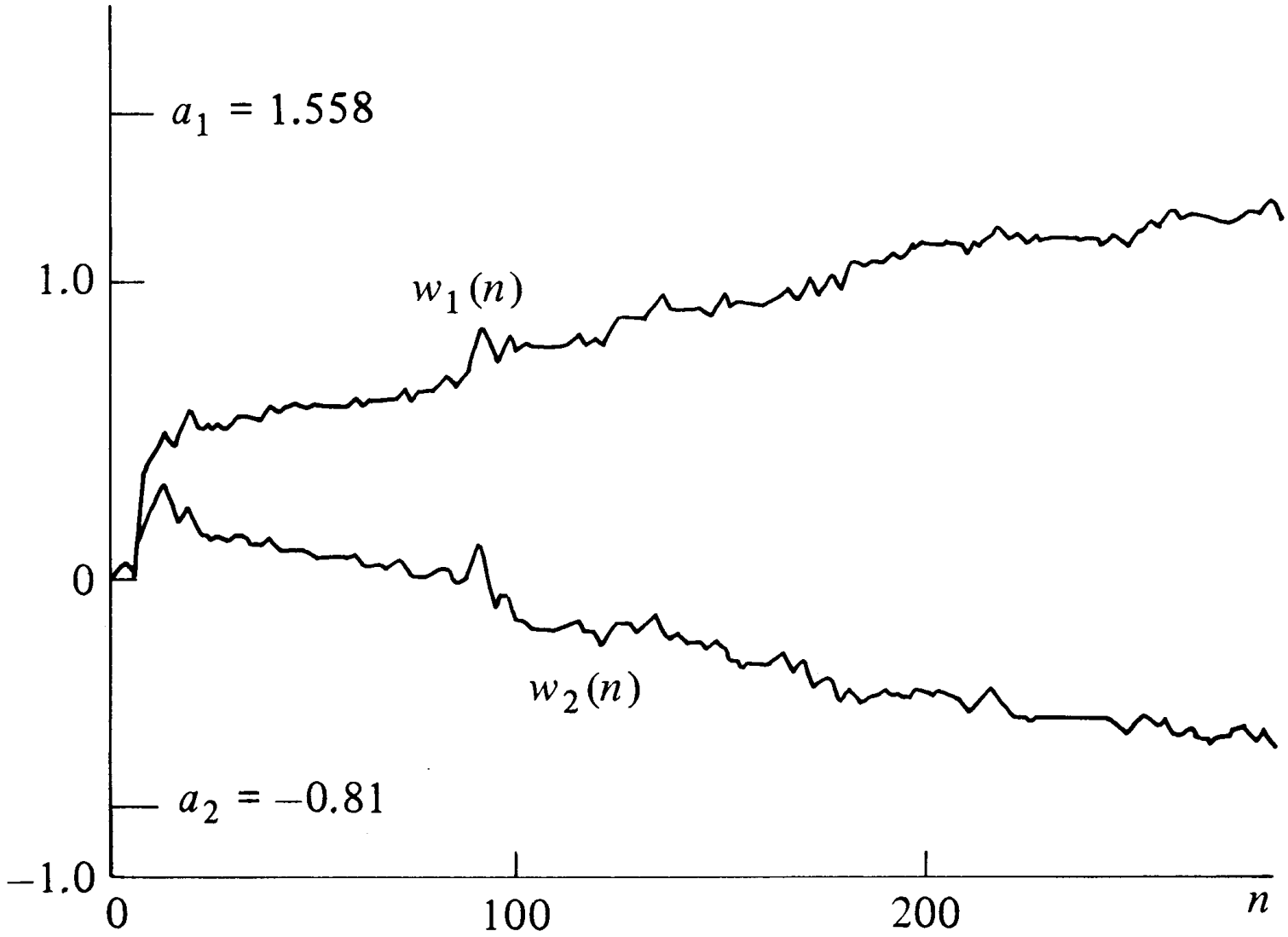
The computational requirement for each sampling interval is

- multiplications : 5
- addition/subtraction : 4

Two values of μ, 0.02 and 0.004, are investigated:



Convergence characteristics for the LMS predictor with $\mu = 0.02$



Convergence characteristics for the LMS predictor with $\mu = 0.004$

Observations:

1. When $\mu = 0.02$, we had a fast convergence rate (the parameters converged to the desired values in approximately 200 iterations) but large fluctuation existed in $w_1(n)$ and $w_2(n)$.
2. When $\mu = 0.004$, small fluctuation in $w_1(n)$ and $w_2(n)$ but the filter coefficients did not converge to the desired values of 1.558 and -0.81 respectively after the 300th iteration.
3. The learning behaviours of $w_1(n)$ and $w_2(n)$ agreed with those of $E\{w_1(n)\}$ and $E\{w_2(n)\}$. Notice that $E\{w_1(n)\}$ and $E\{w_2(n)\}$ can be derived by taking expectation on the LMS algorithm.

Example 4.3

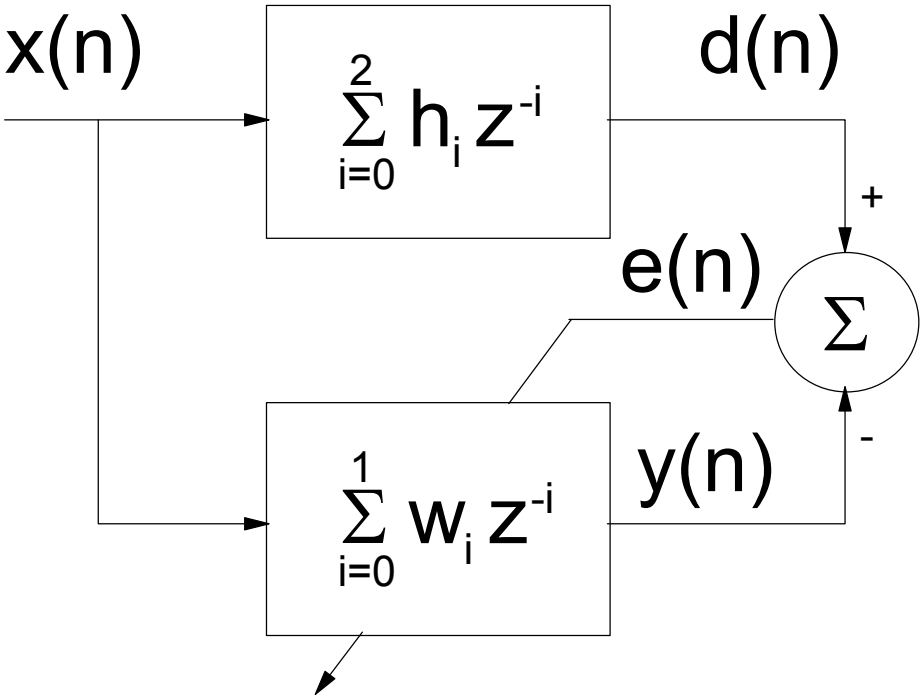
2. System Identification

Given the input signal $x(n)$ and output signal $d(n)$, we can estimate the impulse response of the system or plant using the LMS algorithm.

Suppose the transfer function of the plant is $\sum_{i=0}^{2} h_i z^{-i}$ which is a causal FIR unknown system, then $d(n)$ can be represented as

$$d(n) = \sum_{i=0}^{2} h_i x(n-i)$$

Assuming that the order the transfer function is unknown and we use a 2-coefficient LMS filter to model the system function as follows,



The error function is computed as

$$e(n) = d(n) - y(n) = d(n) - \sum_{i=0}^{1} w_i(n)x(n-i)$$

$$= d(n) - w_0(n)x(n) - w_1(n)x(n-1)$$

Thus the LMS algorithm for this problem is

$$w_0(n+1) = w_0(n) - \frac{\mu}{2} \cdot \frac{\partial e^2(n)}{\partial w_0(n)} = w_0(n) - \frac{\mu}{2} \cdot \frac{\partial e^2(n)}{\partial e(n)} \cdot \frac{\partial e(n)}{\partial w_0(n)}$$
$$= w_0(n) + \mu e(n)x(n)$$

and

$$w_1(n+1) = w_1(n) - \frac{\mu}{2} \cdot \frac{\partial e^2(n)}{\partial w_1(n)}$$
$$= w_1(n) + \mu e(n)x(n-1)$$

The learning behaviours of the filter weights $w_0(n)$ and $w_1(n)$ can be obtained by taking expectation on the LMS algorithm. To simplify the analysis, we assume that $x(n)$ is a stationary white noise process such that

$$R_{xx}(m) = E\{x(n)x(n+m)\} = \begin{cases} \sigma_x^2, & m = 0 \\ 0, & \text{otherwise} \end{cases}$$

Assume the filter weights are independent of $x(n)$ and apply expectation on the first updating rule gives

$$
\begin{aligned}
& E\{w_0(n+1)\} - E\{w_0(n)\} \\
&= \mu E\{e(n)x(n)\} \\
&= \mu E\{(d(n) - y(n))x(n)\} \\
&= \mu E\left\{\left(\sum_{i=0}^{2} h_i x(n-i) - \sum_{i=0}^{1} w_i(n)x(n-i)\right)x(n)\right\} \\
&= \mu E\{(h_0 x(n) + h_1 x(n-1) + h_2 x(n-2) - w_0(n)x(n) - w_1(n)x(n-1))x(n)\} \\
&= \mu h_0 E\{x^2(n)\} + \mu h_1 E\{x(n-1)x(n)\} + \mu h_2 E\{x(n-2)x(n)\} - \\
&\qquad \mu E\{w_0(n)\}E\{x^2(n)\} - \mu E\{w_1(n)\}E\{x(n-1)x(n)\} \\
&= \mu h_0 \sigma_x^2 - \mu E\{w_0(n)\}\sigma_x^2
\end{aligned}
$$

$$\Rightarrow E\{w_0(n+1)\} = E\{w_0(n)\}(1-\mu\sigma_x^2) + \mu h_0 \sigma_x^2$$

$$\Rightarrow E\{w_0(n)\} = E\{w_0(n-1)\}(1-\mu\sigma_x^2) + \mu h_0 \sigma_x^2$$

$$\Rightarrow E\{w_0(n-1)\} = E\{w_0(n-2)\}(1-\mu\sigma_x^2) + \mu h_0 \sigma_x^2$$

$$\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots$$

$$\Rightarrow E\{w_0(1)\} = E\{w_0(0)\}(1-\mu\sigma_x^2) + \mu h_0 \sigma_x^2$$

Multiplying the second equation by $(1-\mu\sigma_x^2)$ on both sides, the third equation by $(1-\mu\sigma_x^2)^2$, etc., and summing all the resultant equations, we have

$$E\{w_0(n+1)\} = E\{w_0(0)\}(1-\mu\sigma_x^2)^{n+1} + \mu h_0 \sigma_x^2 (1 + (1-\mu\sigma_x^2) + \cdots + (1-\mu\sigma_x^2)^n)$$

$$\Rightarrow E\{w_0(n+1)\} = E\{w_0(0)\}(1-\mu\sigma_x^2)^{n+1} + \mu h_0 \sigma_x^2 \frac{1-(1-\mu\sigma_x^2)^{n+1}}{1-(1-\mu\sigma_x^2)}$$

$$\Rightarrow E\{w_0(n+1)\} = E\{w_0(0)\}(1-\mu\sigma_x^2)^{n+1} + h_0(1-(1-\mu\sigma_x^2)^{n+1})$$

$$\Rightarrow E\{w_0(n+1)\} = (E\{w_0(0)\} - h_0)(1-\mu\sigma_x^2)^{n+1} + h_0$$

Hence

$$\lim_{n\to\infty} E\{w_0(n)\} = h_0$$

provided that

$$|\,1-\mu\sigma_x^2\,| < 1 \Rightarrow -1 < 1-\mu\sigma_x^2 < 1 \Rightarrow 0 < \mu < \frac{2}{\sigma_x^2}$$

Similarly, we can show that the expected value of $E\{w_1(n)\}$ is

$$E\{w_1(n)\} = (E\{w_1(0)\} - h_1)(1-\mu\sigma_x^2)^n + h_1$$

provided that

$$|\,1-\mu\sigma_x^2\,| < 1 \Rightarrow -1 < 1-\mu\sigma_x^2 < 1 \Rightarrow 0 < \mu < \frac{2}{\sigma_x^2}$$

It is worthy to note that the choice of the initial filter weights $E\{w_0(0)\}$ and $E\{w_1(0)\}$ do not affect the convergence of the LMS algorithm because the performance surface is unimodal.

Discussion:

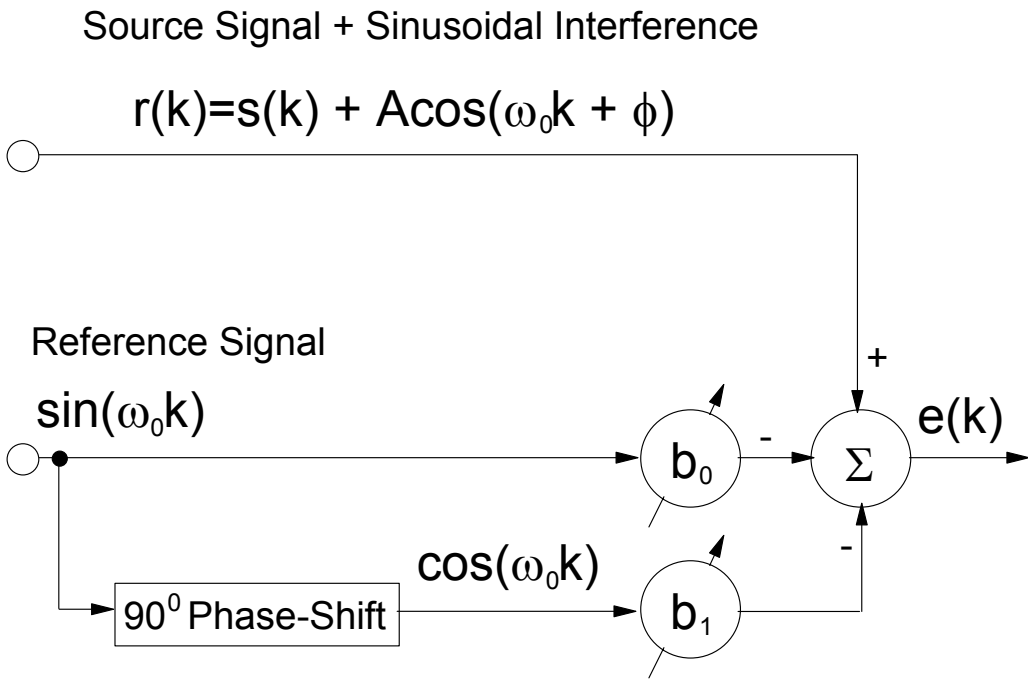
Since the LMS filter consists of two weights but the actual transfer function comprises three coefficients. The plant cannot be exactly modeled in this case. This refers to under-modeling. If we use a 3-weight LMS filter with transfer function $\sum_{i=0}^{2} w_i z^{-i}$, then the plant can be modeled exactly. If we use more than 3 coefficients in the LMS filter, we still estimate the transfer function accurately. However, in this case, the misadjustment will be increased with the filter length used.

Notice that we can also use the Wiener filter to find the impulse response of the plant if the signal statistics, $R_{xx}(0)$, $R_{xx}(1)$, $R_{dx}(0)$ and $R_{dx}(-1)$ are available. However, we do not have $R_{dx}(0)$ and $R_{dx}(-1)$ although $R_{xx}(0)=\sigma_x^2$ and $R_{xx}(1)=0$ are known. Therefore, the LMS adaptive filter can be considered as an adaptive realization of the Wiener filter and it is used when the signal statistics are not (completely) known.

Example 4.4

3. **Interference Cancellation**

Given a received signal $r(k)$ which consist of a source signal $s(k)$ and a sinusoidal interference with known frequency. The task is to extract $s(k)$ from $r(k)$. Notice that the amplitude and phase of the sinusoid is unknown. A well-known application is to remove 50/60 Hz power line interference in the recording of the electrocardiogram (ECG).



The interference cancellation system consists of a 90^0 phase-shifter and a two-weight adaptive filter. By properly adjusting the weights, the reference waveform can be changed in magnitude and phase in any way to model the interfering sinusoid. The filtered output is of the form

$$e(k) = r(k) - b_0(k)\sin(\omega_0 k) - b_1(k)\cos(\omega_0 k)$$

The LMS algorithm is

$$\begin{aligned} b_0(k+1) &= b_0(k) - \frac{\mu}{2}\cdot\frac{\partial e^2(k)}{\partial b_0(k)} = b_0(k) - \frac{\mu}{2}\cdot\frac{\partial e^2(k)}{\partial e(k)}\cdot\frac{\partial e(k)}{\partial b_0(k)} \\ &= w_0(k) + \mu e(k)\sin(\omega_0 k) \end{aligned}$$

and

$$\begin{aligned} b_1(k+1) &= b_1(k) - \frac{\mu}{2}\cdot\frac{\partial e^2(k)}{\partial b_1(k)} \\ &= b_1(k) + \mu e(k)\cos(\omega_0 k) \end{aligned}$$

Taking the expected value of $b_0(k)$, we have

$$
\begin{aligned}
& E\{b_0(k+1)\} \\
&= E\{b_0(k)\} + \mu E\{e(k)\sin(\omega_0 k)\} \\
&= E\{b_0(k)\} + \mu E\{[s(k) + A\cos(\omega_0 k + \phi) - b_0(k)\sin(\omega_0 k) - b_1(k)\cos(\omega_0 k)] \cdot \sin(\omega_0 k)\} \\
&= E\{b_0(k)\} + \mu E\{[A\cos(\omega_0 k)\cos(\phi) - A\sin(\omega_0 k)\sin(\phi) - b_0(k)\sin(\omega_0 k) \\
& - b_1(k)\cos(\omega_0 k)] \cdot \sin(\omega_0 k)\} \\
&= E\{b_0(k)\} + \mu E\left\{\frac{1}{2}\left(A\cos(\phi) - b_1(k)\right)\sin(2\omega_0 k)\right\} - \mu E\left\{\left(A\sin(\phi) + b_0(k)\right)\sin^2(\omega_0 k)\right\} \\
&= E\{b_0(k)\} - \mu E\left\{\left(A\sin(\phi) + b_0(k)\right) \cdot \frac{1 - \cos(2\omega_0 k)}{2}\right\} \\
&= E\{b_0(k)\} - \frac{\mu}{2} A\sin(\phi) - \frac{\mu}{2} E\{b_0(k)\} \\
&= \left(1 - \frac{\mu}{2}\right) E\{b_0(k)\} - \frac{\mu A}{2}\sin(\phi)
\end{aligned}
$$

Following the derivation in Example 4.3, provided that $0 < \mu < 4$, the learning curve of $E\{b_0(k)\}$ can be obtained as

$$E\{b_0(k)\} = -A\sin(\phi) + \left(E\{b_0(0)\} + A\sin(\phi)\right)\cdot\left(1-\frac{\mu}{2}\right)^k$$

Similarly, $E\{b_1(k)\}$ is calculated as

$$E\{b_1(k)\} = A\cos(\phi) + \left(E\{b_1(0)\} - A\cos(\phi)\right)\cdot\left(1-\frac{\mu}{2}\right)^k$$

When $k \to \infty$, we have

$$\lim_{k\to\infty} E\{b_0(k)\} = -A\sin(\phi)$$

and

$$\lim_{k\to\infty} E\{b_1(k)\} = A\cos(\phi)$$

The filtered output is then approximated as

$$\begin{aligned} e(k) &\approx r(k) + A\sin(\phi)\sin(\omega_0 k) - A\cos(\phi)\cos(\omega_0 k) \\ &= s(k) \end{aligned}$$

which means that $s(k)$ can be recovered accurately upon convergence.

Suppose $E\{b_0(0)\} = E\{b_1(0)\} = 0$, $\mu = 0.02$ and we want to find the number of iterations required for $E\{b_1(k)\}$ to reach 90% of its steady state value. Let the required number of iterations be k_0 and it can be calculated from

$$E\{b_1(k)\} = 0.9A\cos(\phi) = A\cos(\phi) - A\cos(\phi)\left(1 - \frac{\mu}{2}\right)^{k_0}$$

$$\Rightarrow \left(1 - \frac{0.02}{2}\right)^{k_0} = 0.1$$

$$\Rightarrow k_0 = \frac{\log(0.1)}{\log(0.99)} = 229.1$$

Hence 300 iterations are required.

If we use Wiener filter with filter weights b_0 and b_1, the mean square error function can be computed as

$$E\{e^2(k)\} = \frac{1}{2}\left(b_0^2 + b_1^2\right) + A\sin(\phi)\cdot b_0 - A\cos(\phi)\cdot b_1 + E\{s^2(k)\} + \frac{A^2}{2}$$

The Wiener coefficients are found by differentiating $E\{e^2(k)\}$ with respect to b_0 and b_1 and then set the resultant expression to zeros. We have

$$\frac{\partial E\{e^2(k)\}}{\partial b_0} = b_0 + A\sin(\phi) = 0 \Rightarrow \tilde{b}_0 = -A\sin(\phi)$$

and

$$\frac{\partial E\{e^2(k)\}}{\partial b_1} = b_1 - A\cos(\phi) = 0 \Rightarrow \tilde{b}_1 = A\cos(\phi)$$

Example 4.5

4. Time Delay Estimation

Estimation of the time delay between two measured signals is a problem which occurs in a diverse range of applications including radar, sonar, geophysics and biomedical signal analysis. A simple model for the received signals is

$$r_1(k) = s(k) + n_1(k)$$
$$r_2(k) = \alpha s(k - D) + n_1(k)$$

where $s(k)$ is the signal of interest while $n_1(k)$ and $n_2(k)$ are additive noises. The α is the attenuation and D is the time delay to be determined. In general, D is not an integral multiple of the sampling period.

Suppose the sampling period is 1 second and $s(k)$ is bandlimited between -0.5 Hz and 0.5 Hz ($-\pi$ rad/s and π rad/s). We can derive the system which can produce a delay of D as follows.

Taking the Fourier transform of $s_D(k) = s(k - D)$ yields

$$\hat{S}(\omega) = e^{-j\omega D} \cdot S(\omega)$$

This means that a system of transfer function $e^{-j\omega D}$ can generate a delay of D for $s(k)$. Using the inverse DTFT formula of (I.9), the impulse response of $e^{-j\omega D}$ is calculated as

$$\begin{aligned} h(n) &= \frac{1}{2\pi} \int_{-\pi}^{\pi} e^{-j\omega nD} \cdot e^{j\omega n} d\omega \\ &= \frac{1}{2\pi} \int_{-\pi}^{\pi} e^{j\omega(n-D)} d\omega \\ &= \text{sinc}(n - D) \end{aligned}$$

where

$$\text{sinc}(v) = \frac{\sin(\pi v)}{\pi v}$$

As a result, $s(k-D)$ can be represented as

$$\begin{aligned} s(k-D) &= s(k) \otimes h(k) \\ &= \sum_{i=-\infty}^{\infty} s(k-i)\mathrm{sinc}(i-D) \\ &\approx \sum_{i=-P}^{P} s(k-i)\mathrm{sinc}(i-D) \end{aligned}$$

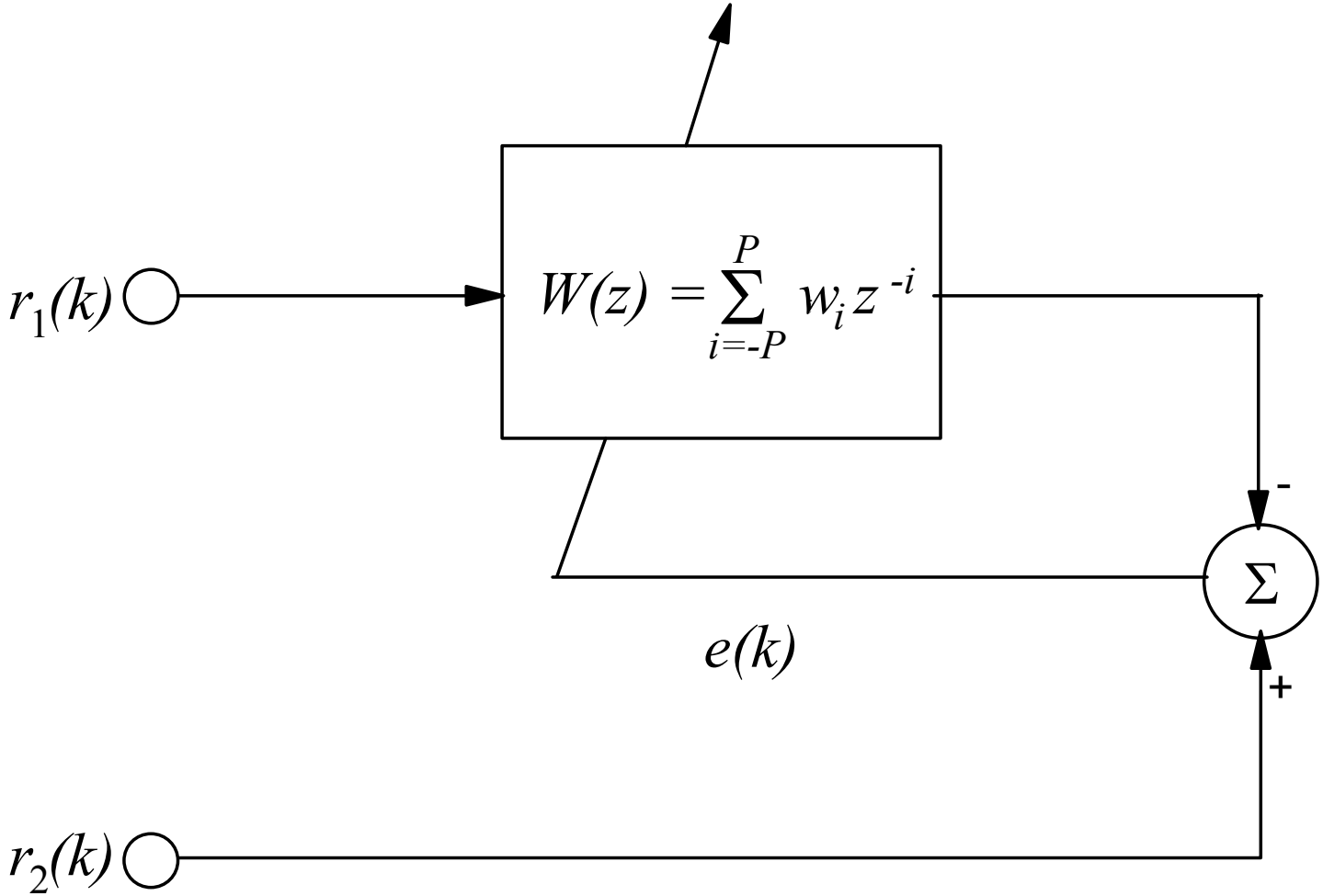
for sufficiently large P.

This means that we can use a non-casual FIR filter to model the time delay and it has the form:

$$W(z) = \sum_{i=-P}^{P} w_i z^{-i}$$

It can be shown that $w_i \to \beta \cdot \mathrm{sinc}(i-D)$ for $i = -P, -P+1, \cdots, P$ using the minimum mean square error approach. The time delay can be estimated from $\{w_i\}$ using the following interpolation:

$$\hat{D} = \arg\max_t \left\{ \sum_{i=-P}^{P} w_i \mathrm{sinc}(i-t) \right\}$$



$$e(k) = r_2(k) - \sum_{i=-P}^{P} r_1(k-i) w_i(k)$$

The LMS algorithm for the time delay estimation problem is thus

$$w_j(k+1) = w_j(k) - \mu \frac{\partial e^2(k)}{\partial w_j(k)}$$

$$= w_j(k) - \mu \frac{\partial e^2(k)}{\partial e(k)} \cdot \frac{\partial e(k)}{\partial w_j(k)}$$

$$= w_j(k) + 2\mu e(k) r_1(k-j), \qquad j = -P, -P+1, \cdots, P$$

The time delay estimate at time k is:

$$\hat{D}(k) = \arg\max_t \left\{ \sum_{i=-P}^{P} w_i(k) \mathrm{sinc}(i-t) \right\}$$

Exponentially Weighted Recursive Least-Squares

A. Optimization Criterion

To minimize the weighted sum of squares $J(n) = \sum_{l=0}^{n} \lambda^{n-l} e^2(l)$ for each time n where λ is a weighting factor such that $0 < \lambda \leq 1$.

When $\lambda = 1$, the optimization criterion is identical to that of least squaring filtering and this value of λ should not be used in a changing environment because all squared errors (current value and past values) have the same weighting factor of 1.

To smooth out the effect of the old samples, λ should be chosen less than 1 for operating in nonstationary conditions.

B. Derivation

Assume FIR filter for simplicity. Following the derivation of the least squares filter, we differentiate $J(n)$ with respect to the filter weight vector at time n, i.e., $\underline{W}(n)$, and then set the L resultant equations to zero.

By so doing, we have

$$\underline{R}(n) \cdot \underline{W}(n) = \underline{G}(n) \tag{4.47}$$

where

$$\underline{R}(n) = \sum_{l=0}^{n} \lambda^{n-l} \underline{X}(l) \underline{X}(l)^T$$

$$\underline{G}(n) = \sum_{l=0}^{n} \lambda^{n-l} d(l) \underline{X}(l)$$

$$\underline{X}(l) = [x(l) \quad x(l-1) \cdots x(l-L+2) \quad x(l-L+1)]^T$$

Notice that $\underline{R}(n)$ and $\underline{G}(n)$ can be computed recursively from

$$\underline{R}(n) = \sum_{l=0}^{n-1} \lambda^{n-l} \underline{X}(l) \underline{X}(l)^T + \underline{X}(n) \underline{X}(n)^T = \lambda \underline{R}(n-1) + \underline{X}(n) \underline{X}(n)^T \tag{4.48}$$

$$\underline{G}(n) = \sum_{l=0}^{n-1} \lambda^{n-l} d(l) \underline{X}(l) + d(n) \underline{X}(n) = \lambda \underline{G}(n-1) + d(n) \underline{X}(n) \tag{4.49}$$

Using the well-known matrix inversion lemma:

If

$$\underline{A} = \underline{B} + \underline{C} \cdot \underline{C}^T \tag{4.50}$$

where $\underline{A}$ and $\underline{B}$ are $N \times N$ matrix and $\underline{C}$ is a vector of length N, then

$$\underline{A}^{-1} = \underline{B}^{-1} - \underline{B}^{-1}\underline{C}(1 + \underline{C}^T \underline{B}^{-1}\underline{C})^{-1}\underline{C}^T \underline{B}^{-1} \tag{4.51}$$

Thus $\underline{R}(n)^{-1}$ can be written as

$$\underline{R}(n)^{-1} = \frac{1}{\lambda}\left[\underline{R}(n-1)^{-1} - \frac{\underline{R}(n-1)^{-1}\underline{X}(n)\underline{X}(n)^T \underline{R}(n-1)^{-1}}{\lambda + X(n)^T \underline{R}(n-1)^{-1}\underline{X}(n)}\right] \tag{4.52}$$

The filter weight $\underline{W}(n)$ is calculated as

$$\underline{W}(n) = \underline{R}(n)^{-1}\underline{G}(n)$$

$$= \frac{1}{\lambda}\left[\underline{R}(n-1)^{-1} - \frac{\underline{R}(n-1)^{-1}\underline{X}(n)\underline{X}(n)^T\underline{R}(n-1)^{-1}}{\lambda + X(n)^T\underline{R}(n-1)^{-1}\underline{X}(n)}\right]\left[\lambda\underline{G}(n-1) + d(n)\underline{X}(n)\right]$$

$$= \underline{R}(n-1)^{-1}\underline{G}(n-1) + \frac{1}{\lambda}d(n)\underline{R}(n-1)^{-1}\underline{X}(n) -$$

$$\frac{\lambda\underline{R}(n-1)^{-1}\underline{X}(n)\underline{X}(n)^T\underline{R}(n-1)^{-1}\underline{G}(n-1) - d(n)\underline{R}(n-1)^{-1}\underline{X}(n)\underline{X}(n)^T\underline{R}(n-1)^{-1}\underline{X}(n)}{\lambda(\lambda + X(n)^T\underline{R}(n-1)^{-1}\underline{X}(n))}$$

$$= \underline{W}(n-1) + \frac{\lambda d(n)\underline{R}(n-1)^{-1}\underline{X}(n) + d(n)\underline{R}(n-1)^{-1}\underline{X}(n)X(n)^T\underline{R}(n-1)^{-1}\underline{X}(n)}{\lambda(\lambda + X(n)^T\underline{R}(n-1)^{-1}\underline{X}(n))} -$$

$$\frac{\lambda\underline{R}(n-1)^{-1}\underline{X}(n)\underline{X}(n)^T\underline{W}(n-1) - d(n)\underline{R}(n-1)^{-1}\underline{X}(n)\underline{X}(n)^T\underline{R}(n-1)^{-1}\underline{X}(n)}{\lambda(\lambda + X(n)^T\underline{R}(n-1)^{-1}\underline{X}(n))}$$

$$= \underline{W}(n-1) + \frac{(d(n) - \underline{X}(n)^T\underline{W}(n-1))\underline{R}(n-1)^{-1}\underline{X}(n)}{\lambda + X(n)^T\underline{R}(n-1)^{-1}\underline{X}(n)}$$

As a result, the exponentially weighted recursive least squares (RLS) algorithm is summarized as follows,

1. Initialize $\underline{W}(0)$ and $\underline{R}(0)^{-1}$
2. For $n = 1,2,\cdots$, compute

$$e(n) = d(n) - \underline{X}(n)^T \underline{W}(n-1) \tag{4.53}$$

$$\alpha(n) = \frac{1}{\lambda + \underline{X}(n)^T \underline{R}(n-1)^{-1} \underline{X}(n)} \tag{4.54}$$

$$\underline{W}(n) = \underline{W}(n-1) + \alpha(n) e(n) \underline{R}(n-1)^{-1} \underline{X}(n) \tag{4.55}$$

$$\underline{R}(n)^{-1} = \frac{1}{\lambda}\left[\underline{R}(n-1)^{-1} - \alpha(n)\underline{R}(n-1)^{-1}\underline{X}(n)\underline{X}(n)^T \underline{R}(n-1)^{-1}\right] \tag{4.56}$$

Remarks:

1. When $\lambda = 1$, the algorithm reduces to the standard RLS algorithm that minimizes $\sum_{l=0}^{n} e^2(l)$.

2. For nonstationary data, $0.95 < \lambda < 0.9995$ has been suggested.

3. Simple choices of $\underline{\underline{W}}(0)$ and $\underline{\underline{R}}(0)^{-1}$ are $\underline{\underline{0}}$ and $\sigma^2 \underline{\underline{I}}$, respectively, where σ^2 is a small positive constant.

C. **Comparison with the LMS algorithm**

1. Computational Complexity

RLS is more computationally expensive than the LMS. Assume there are L filter taps, LMS requires $(4L+1)$ additions and $(4L+3)$ multiplications per update while the exponentially weighted RLS needs a total of $(3L^2 + L - 1)$ additions/subtractions and $(4L^2 + 4L)$ multiplications/divisions.

2. Rate of Convergence

RLS provides a faster convergence speed than the LMS because

- RLS is an approximation of the Newton method while LMS is an approximation of the steepest descent method.
- the pre-multiplication of $\underline{R}(n)^{-1}$ in the RLS algorithm makes the resultant eigenvalue spread becomes unity.

Improvement of LMS algorithm with the use of Orthogonal Transform

A. Motivation

When the input signal is white, the eigenvalue spread has a minimum value of 1. In this case, the LMS algorithm can provide optimum rate of convergence.

However, many practical signals are nonwhite, how can we improve the rate of convergence using the LMS algorithm?

B. **Idea**

To transform the input $x(n)$ to another signal $v(n)$ so that the modified eigenvalue spread is 1. Two steps are involved:

1. Transform $x(n)$ to $v(n)$ using an $N \times N$ orthogonal transform $\underline{T}$ so that

$$\underline{R}_{vv} = \underline{T} \cdot \underline{R}_{xx} \cdot \underline{T}^T = \begin{bmatrix} \sigma_1^2 & 0 & \cdots & & 0 \\ 0 & \sigma_2^2 & & & \vdots \\ \vdots & 0 & \ddots & & \\ & & & \sigma_{N-1}^2 & 0 \\ 0 & \cdots & & 0 & \sigma_N^2 \end{bmatrix}$$

where

$$\underline{V}(n) = \underline{T} \cdot \underline{X}(n)$$

$$\underline{V}(n) = [v_1(n) \quad v_2(n) \quad \cdots \quad v_{N-1}(n) \quad v_N(n)]^T$$

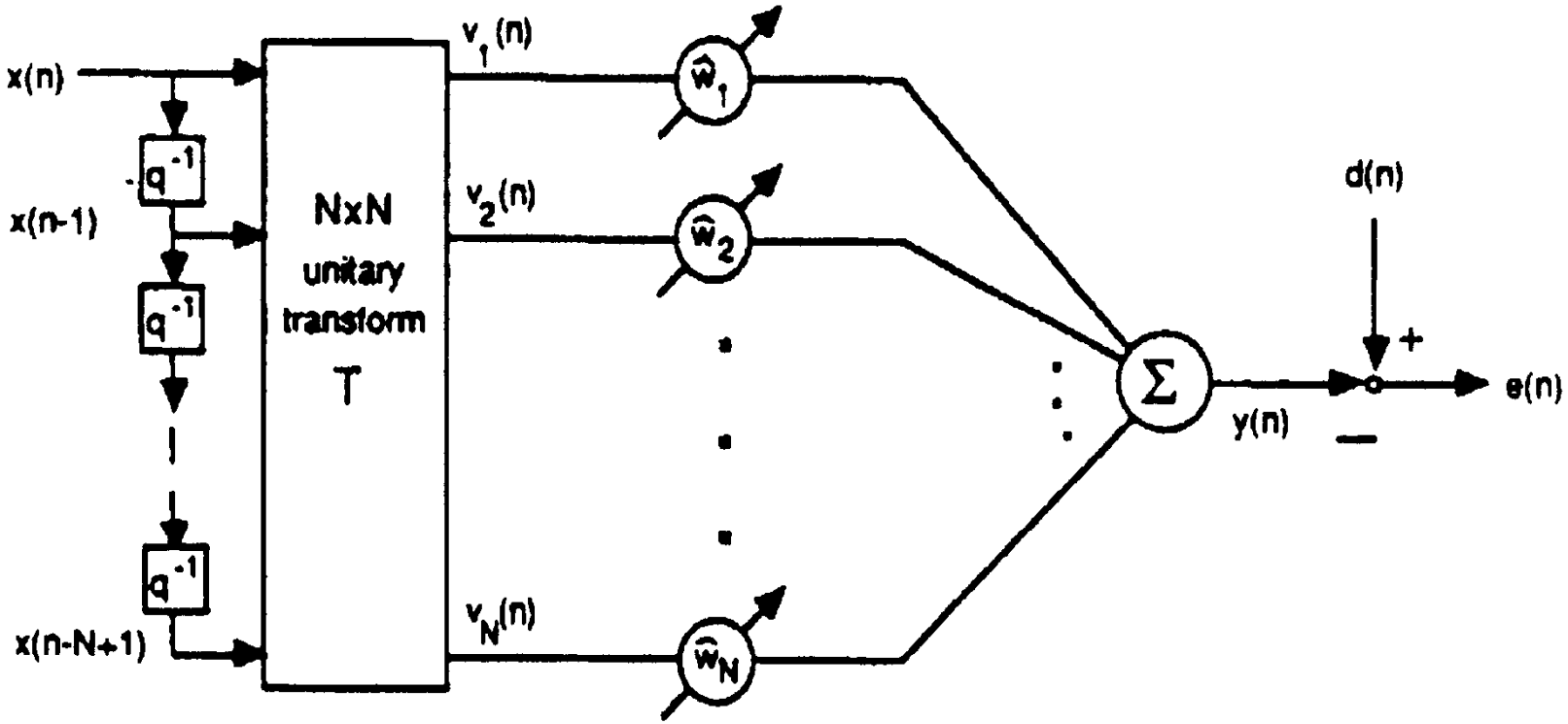
$$\underline{X}(n) = [x(n) \quad x(n-1) \quad \cdots \quad x(n-N+2) \quad x(n-N+1)]^T$$

$$\underline{R}_{xx} = E\left\{\underline{X}(n)\underline{X}(n)^T\right\}$$

$$\underline{R}_{vv} = E\left\{\underline{V}(n)\underline{V}(n)^T\right\}$$

2. Modify the eigenvalues of $\underline{R}_{vv}$ so that the resultant matrix has identical eigenvalues:

$$\underline{R}_{vv} \xrightarrow{\text{power normalizatoin}} \underline{R}'_{vv} = \begin{bmatrix} \sigma^2 & 0 & \cdots & & 0 \\ 0 & \sigma^2 & & & \vdots \\ \vdots & & \ddots & & \\ & & & \sigma^2 & 0 \\ 0 & \cdots & & 0 & \sigma^2 \end{bmatrix}$$



Block diagram of the transform domain adaptive filter

C. Algorithm

The modified LMS algorithm is given by

$$\underline{W}(n+1) = \underline{W}(n) + 2\mu e(n)\underline{\Lambda}^{-2}\underline{V}(n)$$

where

$$\underline{\Lambda}^{-2} = \begin{bmatrix} 1/\sigma_1^2 & 0 & \cdots & & 0 \\ 0 & 1/\sigma_2^2 & & & \vdots \\ \vdots & & \ddots & & \\ & & & 1/\sigma_{N-1}^2 & 0 \\ 0 & \cdots & & 0 & 1/\sigma_N^2 \end{bmatrix}$$

$$e(n) = d(n) - y(n)$$

$$y(n) = \underline{W}(n)^T \underline{V}(n) = \underline{W}(n)^T \cdot \underline{T} \cdot \underline{X}(n)$$

$$\underline{W}(n) = [w_1(n) \quad w_2(n) \quad \cdots \quad w_{N-1}(n) \quad w_N(n)]^T$$

Writing in scalar form, we have

$$w_i(n+1) = w_i(n) + \frac{2\mu e(n) v_i(n)}{\sigma_i^2}, \qquad i = 1,2,\cdots,N$$

Since σ_i^2 is the power of $v_i(n)$ and it is not known *a priori* and should be estimated. A common estimation procedure for $E\{v_i^2(n)\}$ is

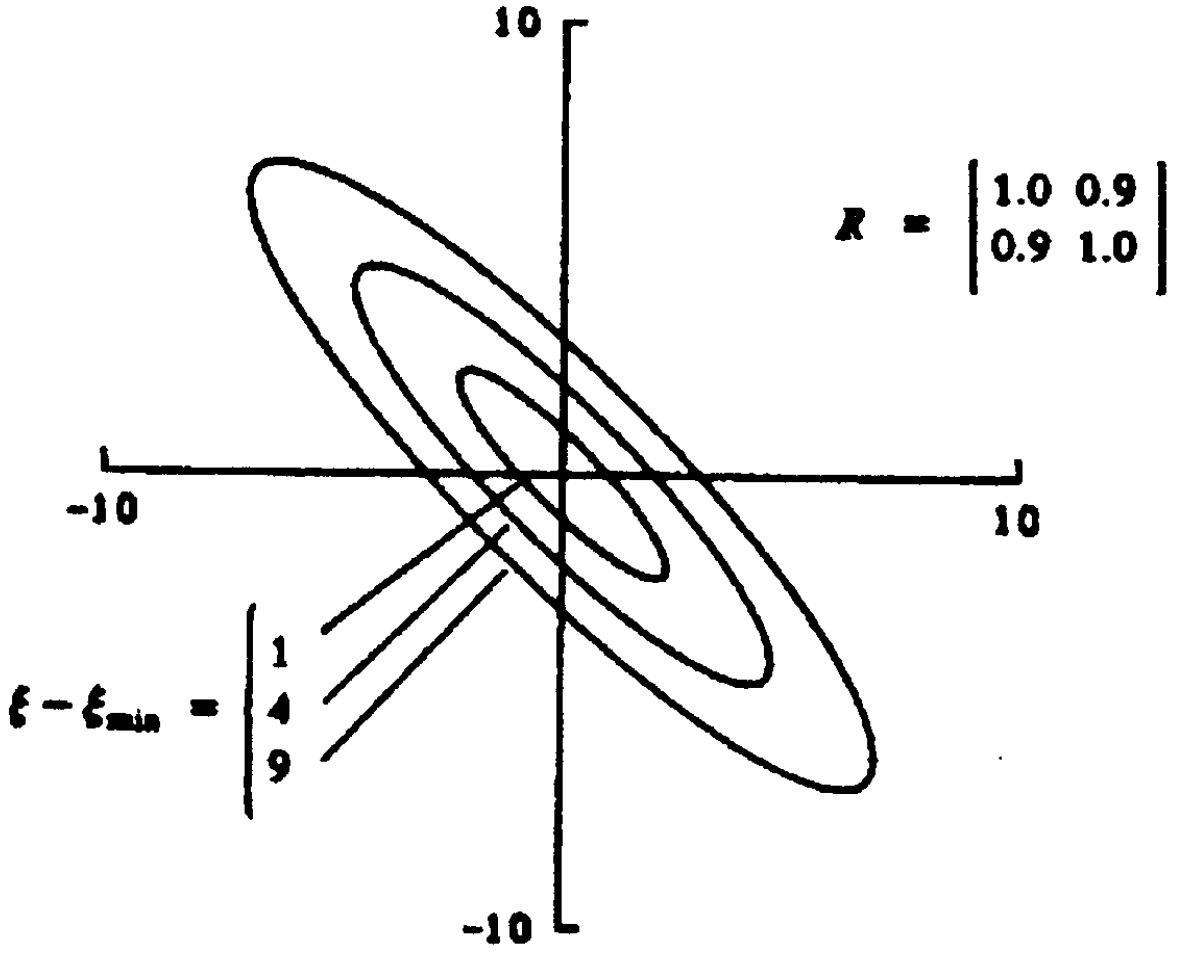
$$\sigma_i^2(n) = \alpha\sigma_i^2(n-1) + |v_i(n)|^2$$

where

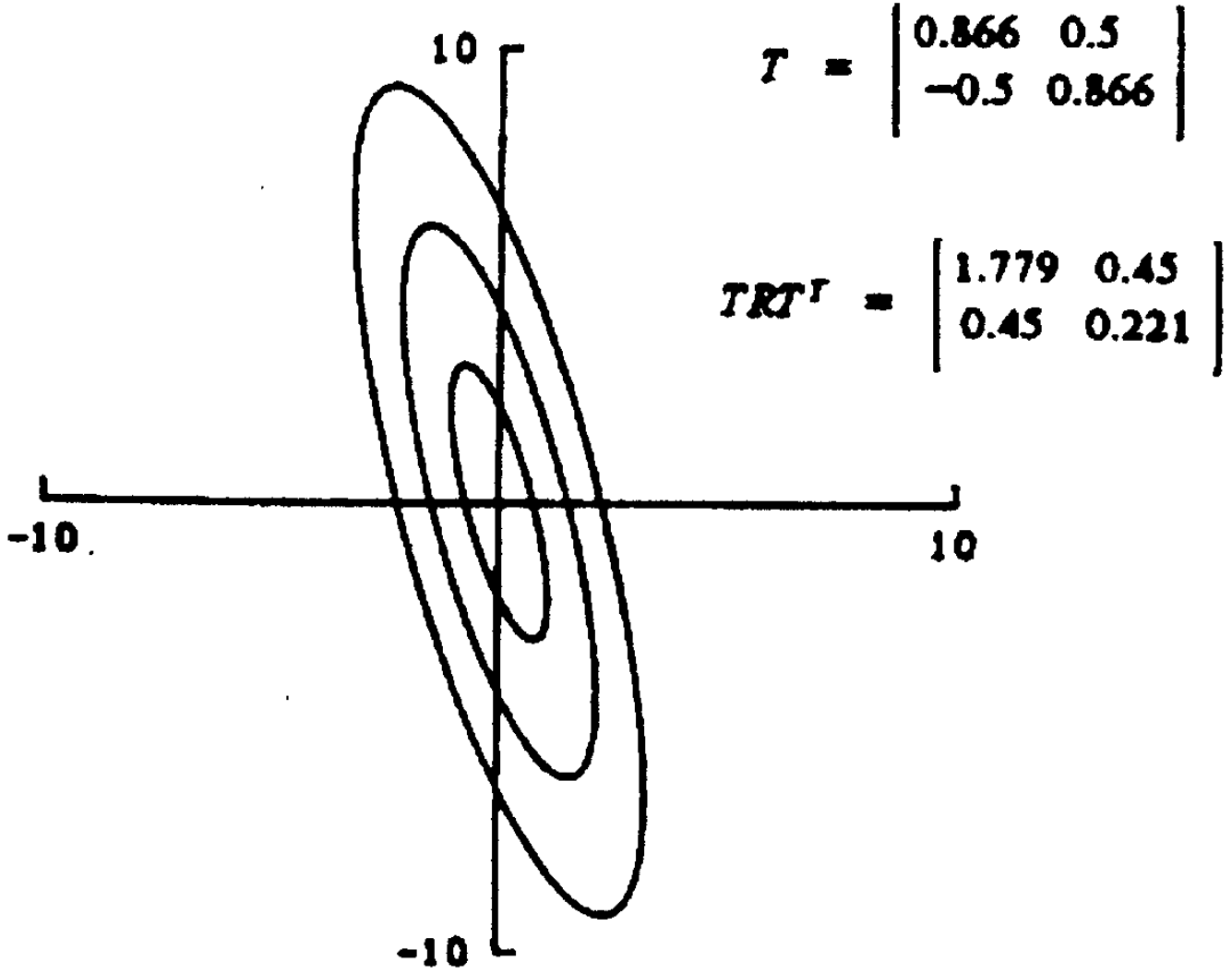
$$0 < \alpha < 1$$

In practice, α should be chosen close to 1, say, $\alpha = 0.9$.

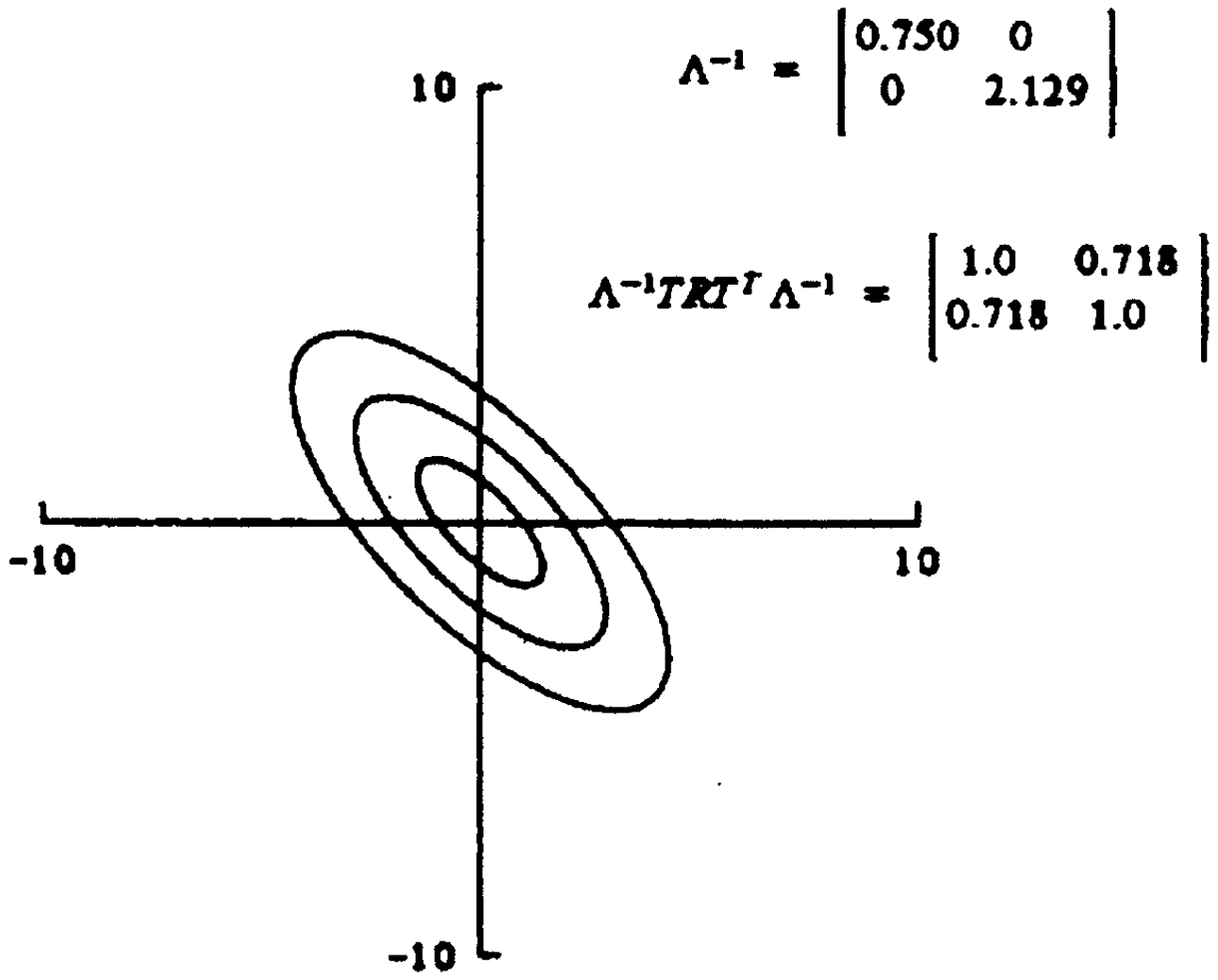
Using a 2-coefficient adaptive filter as an example:



A 2-D error surface without transform



Error surface with discrete cosine transform (DCT)



Error surface with transform and power normalization

Remarks:

1. The lengths of the principle axes of the hyperellipses are proportional to the eigenvalues of $\underline{R}$.
2. Without power normalization, no convergence rate improvement of using transform can be achieved.
3. The best choice for $\underline{T}$ should be Karhunen-Loeve (KL) transform which is signal dependent. This transform can make $\underline{R}_{vv}$ to a diagonal matrix but the signal statistics are required for its computation.
4. Considerations in choosing a transform:
   - fast algorithm exists?
   - complex or real transform?
   - elements of the transform are all power of 2?
5. Examples of orthogonal transforms are discrete sine transform (DST), discrete Fourier transform (DFT), discrete cosine transform (DCT), Walsh-Hadamard transform (WHT), discrete Hartley transform (DHT) and power-of-2 (PO2) transform.

Improvement of LMS algorithm using Newton's method

Since the eigenvalue spread of Newton based approach is 1, we can combine the LMS algorithm and Newton's method to form the "LMS/Newton" algorithm as follows,

$$\underline{W}(n+1) = \underline{W}(n) - \frac{\mu}{2}\underline{R}_{xx}^{-1}\frac{\partial e^2(n)}{\partial \underline{W}(n)}$$

$$= \underline{W}(n) + \mu \underline{R}_{xx}^{-1} e(n)\underline{X}(n)$$

Remarks:

1. The computational complexity of the LMS/Newton algorithm is smaller than the RLS algorithm but greater than the LMS algorithm.

2. When $\underline{R}_{xx}$ is not available, it can be estimated as follows,

$$\hat{R}_{xx}(l,n) = \alpha \hat{R}_{xx}(l,n-1) + x(n+l)\cdot x(n), \qquad l = 0,1,\cdots,L-1$$

where $\hat{R}_{xx}(l,n)$ represents the estimate of $R_{xx}(l)$ at time n and $0 < \alpha < 1$

Possible Research Directions for Adaptive Signal Processing

1. Adaptive modeling of non-linear systems

For example, second-order Volterra system is a simple non-linear system. The output $y(n)$ is related to the input $x(n)$ by

$$y(n) = \sum_{j=0}^{L-1} w^{(1)}(j)x(n-j) + \sum_{j_1=0}^{L-1}\sum_{j_2=0}^{L-1} w^{(2)}(j_1, j_2)x(n-j_1)x(n-j_2)$$

Another related research direction is to analyze non-linear adaptive filters, for example, neural networks, which are generally more difficult to analyze its performance.

2. New optimization criterion for Non-Gaussian signals/noises

For example, LMF algorithm minimizes $E\{e^4(n)\}$.

In fact, a class of steepest descend algorithms can be generalized by the least-mean-p (LMP) norm. The cost function to be minimized is given by

$$J = E\{|e(k)|^p\}$$

Some remarks:

- When p=1, it becomes least-mean-deviation (LMD), when p=2, it is least-mean-square (LMS) and if p=4, it becomes the least-mean-fourth (LMF).
- The LMS is optimum for Gaussian noises and it may not be true for noises of other probability density functions (PDFs). For example, if the noise is impulsive such as a α-stable process with $1 \le \alpha < 2$, LMD performs better than LMS; if the noise is of uniform distribution or if it is a sinusoidal signal, then LMF outperforms LMS. Therefore, the optimum p depends on the signal/noise models.
- The parameter p can be any real number but it will be difficult to analyze, particularly for non-integer p.
- Combination of different norms can be used to achieve better performance.
- Some suggests mixed norm criterion, e.g. $a \cdot E\{e^2(n)\} + b \cdot E\{e^4(n)\}$

- Median operation can be employed in the LMP algorithm for operating in the presence of impulsive noise. For example, the median LMS belongs to the family of order-statistics-least-mean-square (OSLMS) adaptive filter algorithms.

3. **Adaptive algorithms with fast convergence rate and small computation**

For example, design of optimal step size in LMS algorithms

4. **Adaptive IIR filters**

Adaptive IIR filters have 2 advantages over adaptive FIR filters:

- It generalizes FIR filter and it can model IIR system more accurately
- Less filter coefficients are generally required

However, development of adaptive IIR filters are generally more difficult than the FIR filters because

- The performance surface is multimodal $\Rightarrow$ the algorithm may lock at an undesired local minimum
- It may lead to biased solution
- It can be unstable

5. Unsupervised adaptive signal processing (blind signal processing)

What we have discussed previously refers to supervised adaptive signal processing where there is always a desired signal or reference signal or training signal.

In some applications, such signals are not available. Two important application areas of unsupervised adaptive signal processing are:

- Blind source separation
  - e.g. speaker identification in the noisy environment of a cocktail party
  - e.g. separation of signals overlapped in time and frequency in wireless communications
- Blind deconvolution (= inverse of convolution)
  - e.g. restoration of a source signal after propagating through an unknown wireless channel

6. New applications

For example, echo cancellation for hand-free telephone systems and signal estimation in wireless channels using space-time processing.

Questions for Discussion

1. The LMS algorithm is given by (4.23):

$$w_i(n+1) = w_i(n) + 2\mu e(n)x(n-i), \qquad i = 0,1,\cdots,L-1$$

where

$$e(n) = d(n) - y(n)$$

$$y(n) = \sum_{i=0}^{L-1} w_i(n)x(n-i) = \underline{W}(n)^T \underline{X}(n),$$

Based on the idea of LMS algorithm, derive the adaptive algorithm that minimizes $E\{|e(n)|\}$.

(Hint: $\dfrac{\partial |v|}{\partial v} = \mathrm{sgn}(v)$ where $\mathrm{sgn}(v) = 1$ if $v > 1$ and $\mathrm{sgn}(v) = -1$ otherwise)

2. For adaptive IIR filtering, there are basically two approaches, namely, output-error and equation-error. Let the unknown IIR system be

$$H(z) = \frac{B(z)}{A(z)} = \frac{\sum_{j=0}^{N-1} b_j z^{-j}}{1 + \sum_{i=1}^{M-1} a_i z^{-i}}$$

Using minimizing mean square error as the performance criterion, the output-error scheme is a direct approach which minimizes $E\{e^2(n)\}$ where

$$e(n) = d(n) - y(n)$$

with

$$\frac{Y(z)}{X(z)} = \frac{\hat{B}(z)}{\hat{A}(z)} = \frac{\sum_{j=0}^{N-1} \hat{b}_j z^{-j}}{1 + \sum_{i=1}^{M-1} \hat{a}_i z^{-i}}$$

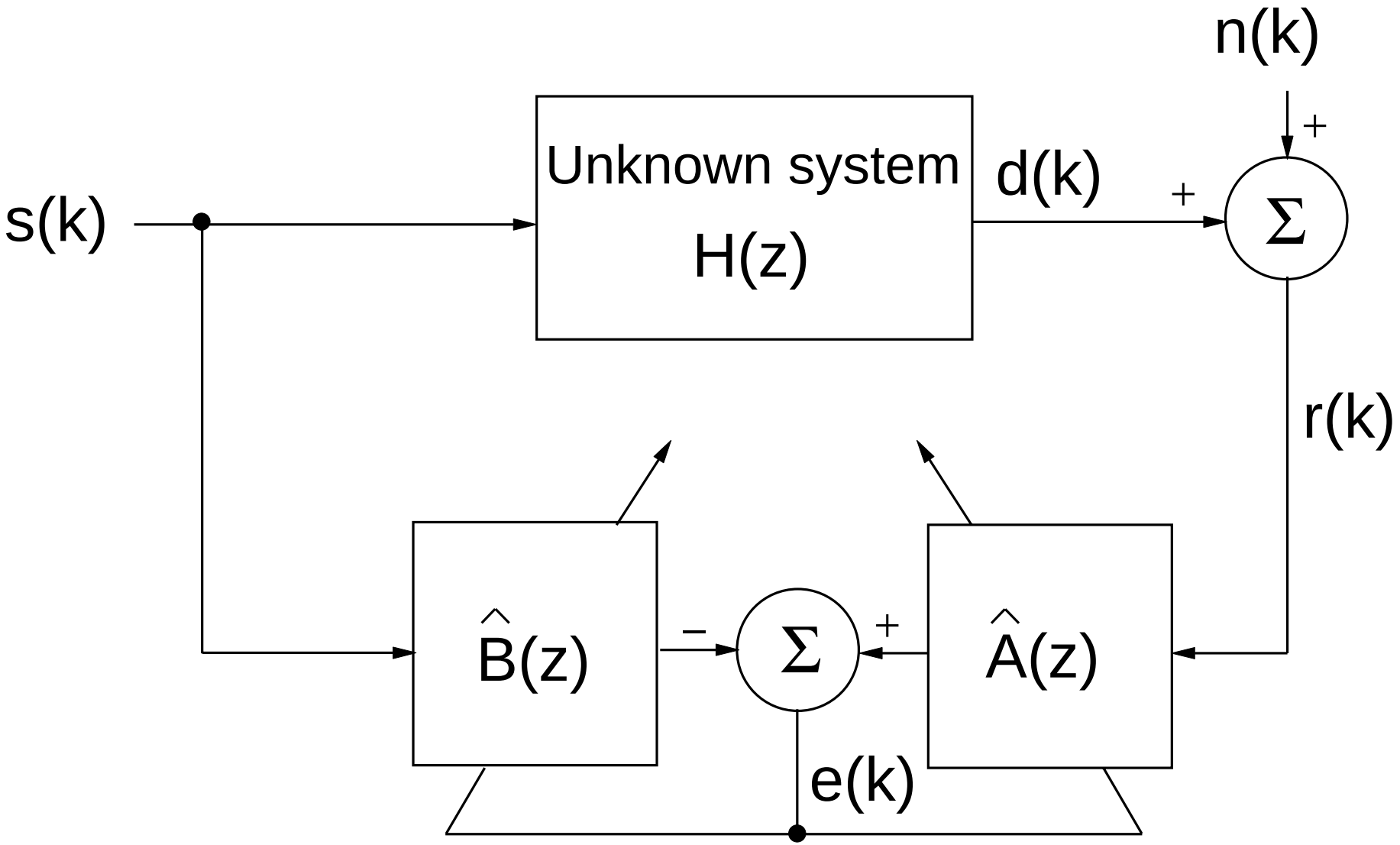
$$\Rightarrow y(n) = \sum_{j=0}^{N-1} \hat{b}_j x(n-j) - \sum_{i=1}^{M-1} \hat{a}_i y(n-i)$$

However, as in Q.2 of Chapter 3, this approach has two problems, namely, stability and multimodal performance surface.

On the other hand, the equation-error approach is always stable and has a unimodal surface. Its system block diagram is shown in the next page.

Can you see the main problem of it and suggest a solution?

(Hint: Assume $n(k)$ is white and examine $E\{e^2(k)\}$)


n(k)
+
Unknown system
H(z)
d(k)
+
Σ
s(k)
r(k)
B̂(z)
−
Σ
+
Â(z)
e(k)