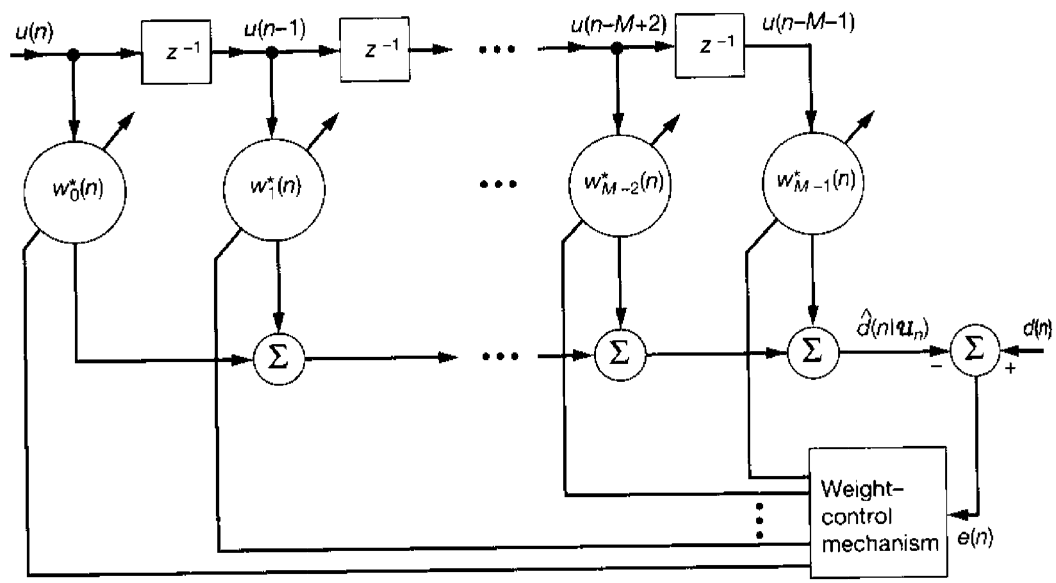
Adaptive Filtering Method of Steepest Descent

Steepest descent is an old, deterministic method, which is the basis for stochastic gradient based methods.

This is a feedback approach to finding the minimum of the error performance surface.

- error surface must be known
- adaptive approach converges to the optimal solution, $\mathbf{w}_0 = \mathbf{R}^{-1}\mathbf{p}$ without inverting a matrix



- $\{x(n)\}$ are the WSS input samples
- $\{d(n)\}$ is the WSS desired output
- $\{\hat{d}(n)\}$ is the estimate of the desired signal given by

$$\hat{d}(n) = \mathbf{w}^H(n)\mathbf{x}(n)$$

where $\mathbf{x}(n) = [x(n), x(n-1), \cdots, x(n-M+1)]^T$ and

$\mathbf{w}(n) = [w_0(n), w_1(n), \cdots, w_{\text{M-1}}(n)]^T$ is the filter weight vector at time n.

Then

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

Thus the MSE of time n is

$$\begin{aligned} J(n) &= E\{|e(n)|^2\} \\ &= \sigma_d^2 - \mathbf{w}^H\mathbf{p} - \mathbf{p}^H\mathbf{w}(n) + \mathbf{w}^H(n)\mathbf{R}\mathbf{w}(n) \end{aligned}$$

where

- σ_d^2 – variance of desired signal
- $\mathbf{p}$ – cross-correlation between $\mathbf{x}(n)$ and $d(n)$
- $\mathbf{R}$ – correlation matrix of $\mathbf{x}(n)$

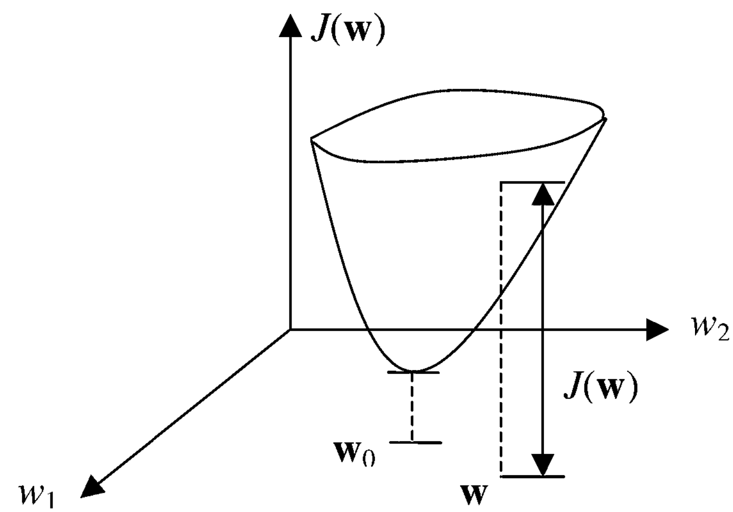
When $\mathbf{w}(n)$ is set to the (optimal) Wiener solution, then

$$\mathbf{w}(n) = \mathbf{w}_0 = \mathbf{R}^{-1}\mathbf{p}$$

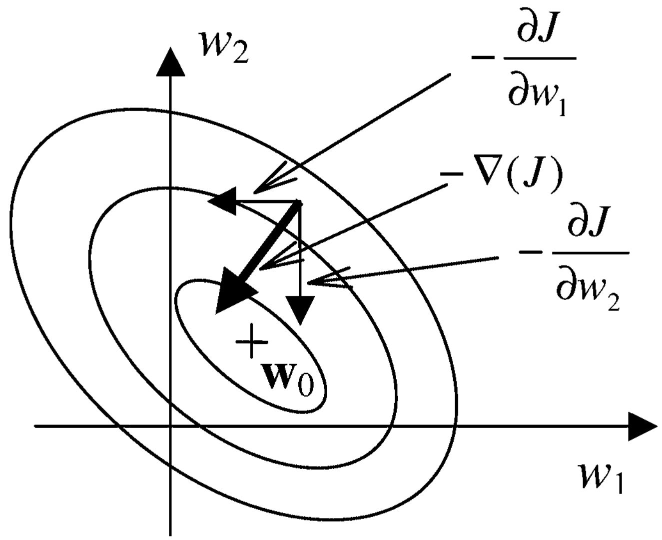
and

$$J(n) = J_{\min} = \sigma_d^2 - \mathbf{p}^H \mathbf{w}_0$$

Hence, in order to iteratively find $\mathbf{w}_0$, we use the method of steepest descent. To illustrate this concept, let $M = 2$, in the 2-D spaced $\mathbf{w}(n)$, the MSE forms a bowl-shaped function.



A contour of the MSE is given as



Thus, if we are at a specific point in the bowl, we can imagine dropping a marble. It would reach the minimum by going through the path of steepest descent.

Hence the direction in which we change the filter direction is $-\nabla J(n)$, or

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

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

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

for $n = 0, 1, 2, \cdots$ and where μ is called the step size and

$$\mathbf{w}(0) = \mathbf{0} \qquad \text{(in general)}$$

Stability: Since the SD method uses feedback, the system can go unstable

- bounds on the step size guaranteeing stability can be determined with respect to the eigenvalues of $\mathbf{R}$ (widrow, 1970)

Define the error vector for the tap weights as

$$\mathbf{c}(n) = \mathbf{w}(n) - \mathbf{w}_0$$

Then using $\mathbf{p} = \mathbf{R}\mathbf{w}_0$ in the update,

$$\begin{aligned} \mathbf{w}(n+1) &= \mathbf{w}(n) + \mu[\mathbf{p} - \mathbf{R}\mathbf{w}(n)] \\ &= \mathbf{w}(n) + \mu[\mathbf{R}\mathbf{w}_0 - \mathbf{R}\mathbf{w}(n)] \\ &= \mathbf{w}(n) - \mu\mathbf{R}\mathbf{c}(n) \end{aligned}$$

and

$$\mathbf{w}(n+1) - \mathbf{w}_0 = \mathbf{w}(n) - \mathbf{w}_0 - \mu\mathbf{R}\mathbf{c}(n)$$

or

$$\begin{aligned} \mathbf{c}(n+1) &= \mathbf{c}(n) - \mu\mathbf{R}\mathbf{c}(n) \\ &= [\mathbf{I} - \mu\mathbf{R}]\mathbf{c}(n) \end{aligned}$$

Using the Unitary Similarity Transform

$$\mathbf{R} = \mathbf{Q}\boldsymbol{\Omega}\mathbf{Q}^H$$

we have

$$\mathbf{c}(n+1) = [\mathbf{I} - \mu\mathbf{Q}\boldsymbol{\Omega}\mathbf{Q}^H]\mathbf{c}(n)$$

Pre-multiplying by $\mathbf{Q}^H$ gives

$$\begin{aligned} \mathbf{Q}^H\mathbf{c}(n+1) &= [\mathbf{Q}^H - \mu\mathbf{Q}^H\mathbf{Q}\boldsymbol{\Omega}\mathbf{Q}^H]\mathbf{c}(n) \\ &= [\mathbf{I} - \mu\boldsymbol{\Omega}]\mathbf{Q}^H\mathbf{c}(n) \end{aligned}$$

Define the transformed coefficients as

$$\begin{aligned} \mathbf{v}(n) &= \mathbf{Q}^H\mathbf{c}(n) \\ &= \mathbf{Q}^H(\mathbf{w}(n) - \mathbf{w}_0) \end{aligned}$$

Then

$$\mathbf{v}(n+1) = [\mathbf{I} - \mu\boldsymbol{\Omega}]\mathbf{v}(n)$$

with initial condition

$$\mathbf{v}(0) = \mathbf{Q}^H(\mathbf{w}(0) - \mathbf{w}_0) = -\mathbf{Q}^H\mathbf{w}_0$$

if $\mathbf{w}(0) = \mathbf{0}$.

The k^{th} term in $\mathbf{v}(n+1)$ (mode) is given by

$$v_k(n+1) = (1 - \mu\lambda_k)v_k(n) \quad k = 1, 2, \cdots, M$$

or using the recursion

$$v_k(n) = (1 - \mu\lambda_k)^n v_k(0)$$

Thus for all $\lim_{n\to\infty} v_k(n) = 0$ we must have

$$|1 - \mu\lambda_k| < 1 \quad \text{for all } k\text{, or} \quad 0 < \mu < \frac{2}{\lambda_{\max}}$$

The k^{th} mode has geometric decay

$$v_k(n) = (1 - \mu\lambda_k)^n v_k(0)$$

We can characterize the rate of decay by finding the time it takes to decay to e^{-1} of the initiative. Thus

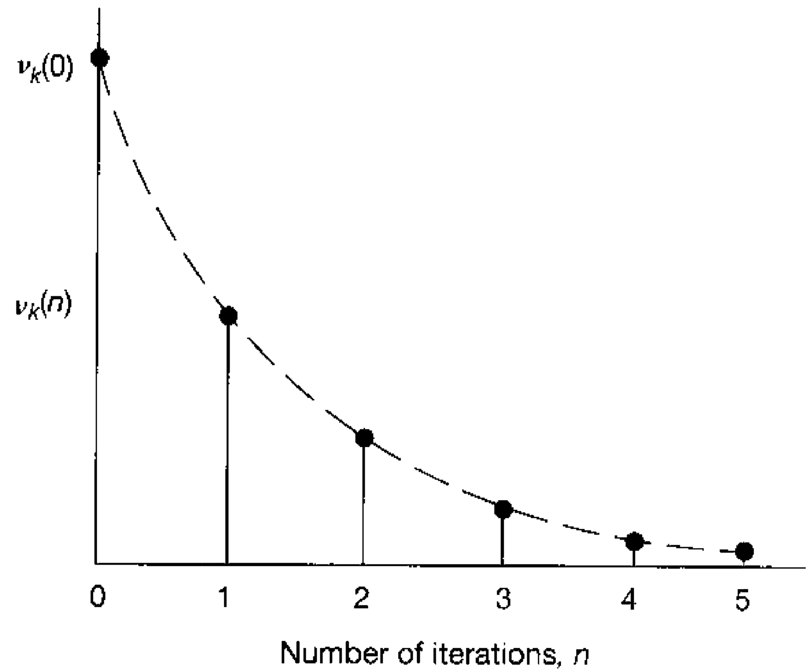
$$v_k(\tau_k) = (1 - \mu\lambda_k)^{\tau_k} v_k(0) = e^{-1} v_k(0)$$

$$\Downarrow$$

$$\tau_k = \frac{-1}{\ln(1 - \mu\lambda_k)} \approx \frac{1}{\mu\lambda_k} \quad \text{for} \quad \mu \ll 1$$

The overall rate of decay is

$$\frac{-1}{\ln(1 - \mu\lambda_{\max})} \leq \tau \leq \frac{-1}{\ln(1 - \mu\lambda_{\min})}$$



Recall that

$$
\begin{aligned}
J(n) &= J_{\min} + (\mathbf{w}(n) - \mathbf{w}_0)^H \mathbf{R} (\mathbf{w}(n) - \mathbf{w}_0) \\
&= J_{\min} + (\mathbf{w}(n) - \mathbf{w}_0)^H \mathbf{Q} \boldsymbol{\Omega} \mathbf{Q}^H (\mathbf{w}(n) - \mathbf{w}_0) \\
&= J_{\min} + \mathbf{v}(n)^H \boldsymbol{\Omega} \mathbf{v}(n)
\end{aligned}
$$

$$
\begin{aligned}
&= J_{\min} + \sum_{k=1}^{M} \lambda_k |v_k(n)|^2 \\
&= J_{\min} + \sum_{k=1}^{M} \lambda_k (1 - \mu\lambda_k)^{2n} |v_k(0)|^2
\end{aligned}
$$

Thus $\lim_{n\to\infty} J(n) = J_{\min}$.

Example:

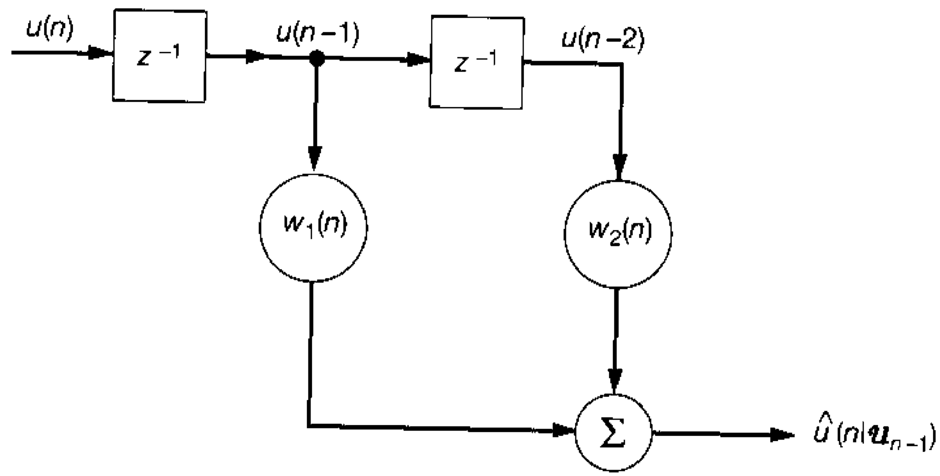
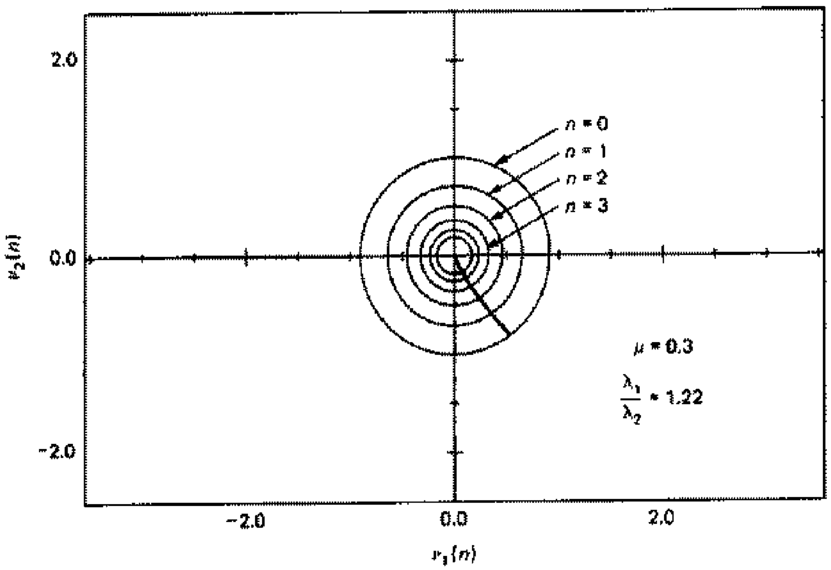


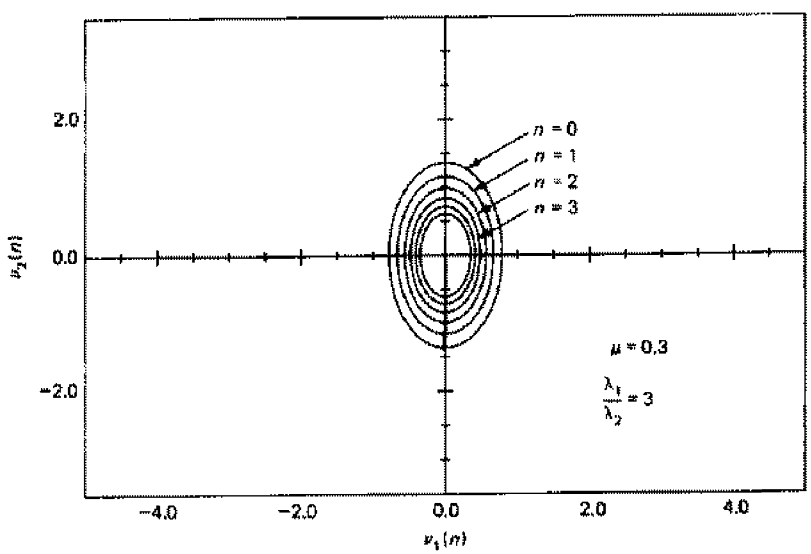
Figure 1: Two-tap predictor for real-valued input.

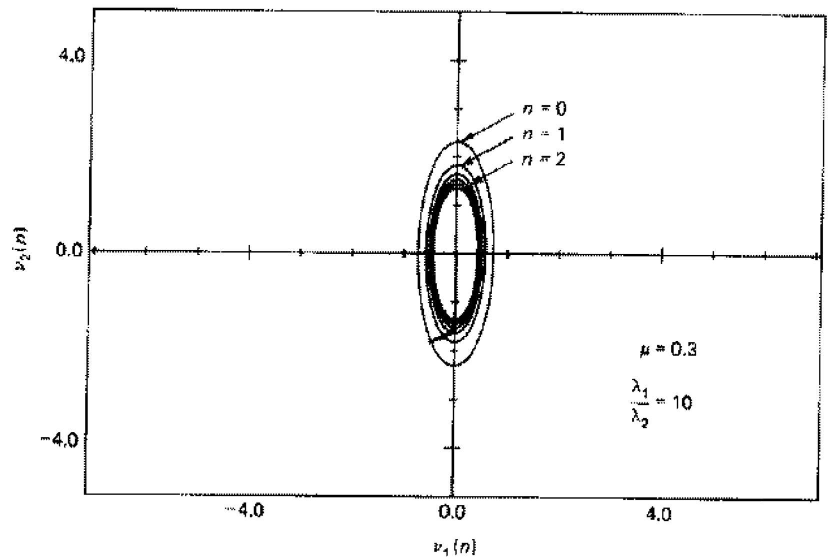
Consider the effects of the following cases

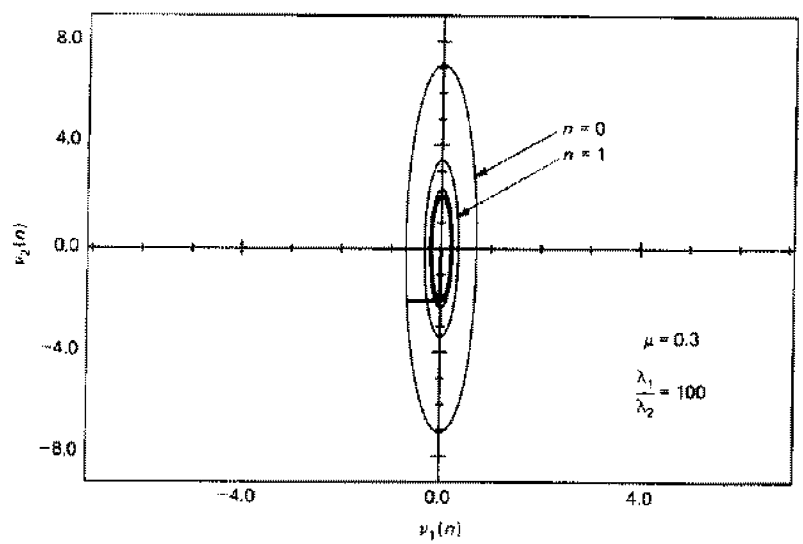
- Varying the eigenvalue spread $\chi(\mathbf{R}) = \frac{\lambda_{\max}}{\lambda_{\min}}$ and keeping μ fixed.
- Varying μ and keeping the eigenvalue spread $\chi(\mathbf{R})$ fixed.

The following 4 figures plot the loci of $v_1(n)$ versus $v_2(n)$ for the SD algorithm with step-size $\mu = 0.3$ and varying eigenvalue spread:
(a)$\chi(\mathbf{R}) = 1.22$; (b)$\chi(\mathbf{R}) = 3$; (c) $\chi(\mathbf{R}) = 10$; (d) $\chi(\mathbf{R}) = 100$.

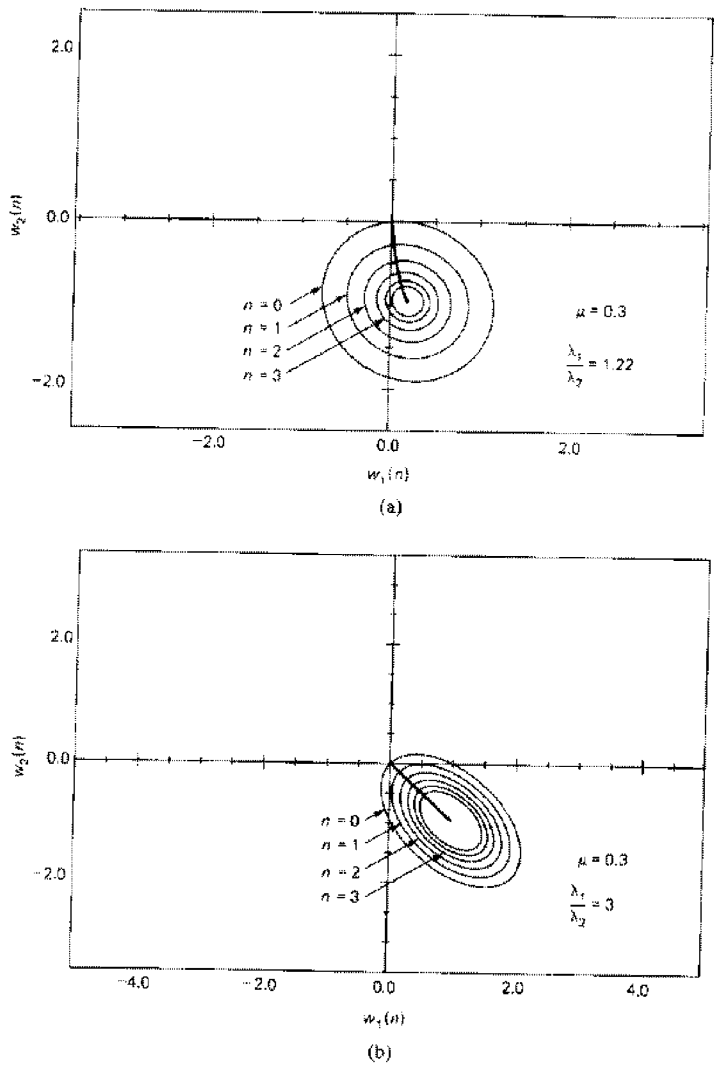

2.0
0.0
-2.0
$v_2(n)$
-2.0
0.0
2.0
$v_1(n)$
n = 0
n = 1
n = 2
n = 3
μ = 0.3
$\frac{\lambda_1}{\lambda_2}$ = 1.22

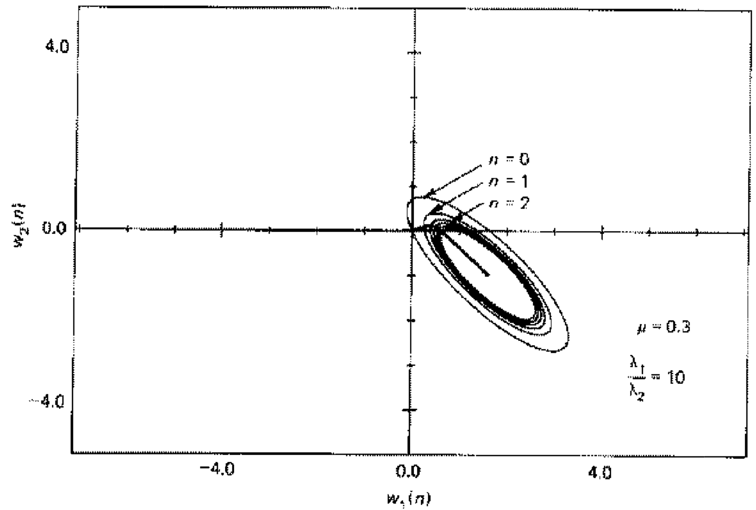

2.0
0.0
-2.0
$v_2(n)$
-4.0
-2.0
0.0
2.0
4.0
$v_1(n)$
n = 0
n = 1
n = 2
n = 3
μ = 0.3
$\frac{\lambda_1}{\lambda_2}$ = 3


4.0
0.0
−4.0
$\nu_2(n)$
−4.0
0.0
4.0
$\nu_1(n)$
$n = 0$
$n = 1$
$n = 2$
$\mu = 0.3$
$\frac{\lambda_1}{\lambda_2} = 10$

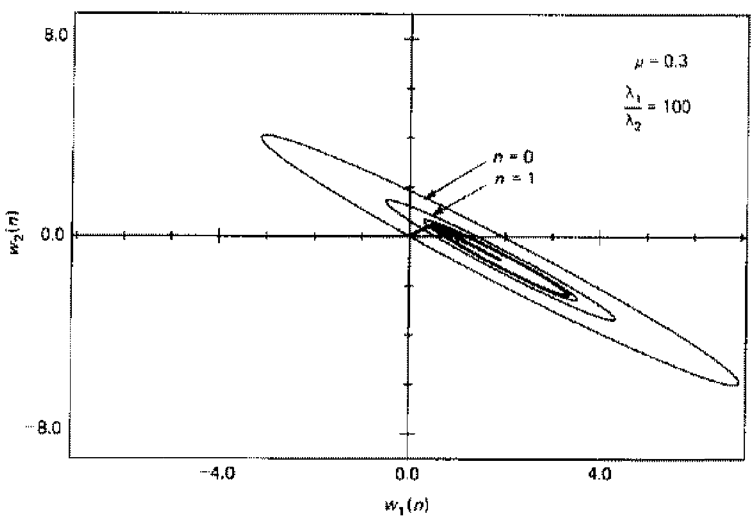

8.0
4.0
0.0
−4.0
−8.0
$\nu_2(n)$
−4.0
0.0
4.0
$\nu_1(n)$
$n = 0$
$n = 1$
$\mu = 0.3$
$\frac{\lambda_1}{\lambda_2} = 100$

The following 4 figures plot the loci of $w_1(n)$ versus $w_2(n)$ for the SD algorithm with step-size $\mu = 0.3$ and varying eigenvalue spread: (a)$\chi(\mathbf{R}) = 1.22$; (b)$\chi(\mathbf{R}) = 3$; (c) $\chi(\mathbf{R}) = 10$; (d) $\chi(\mathbf{R}) = 100$.


2.0
0.0
−2.0
$w_2(n)$
−2.0
0.0
2.0
$w_1(n)$
n = 0
n = 1
n = 2
n = 3
μ = 0.3
$\frac{\lambda_1}{\lambda_2}$ = 1.22
(a)
2.0
0.0
−2.0
$w_2(n)$
−4.0
−2.0
0.0
2.0
4.0
$w_1(n)$
n = 0
n = 1
n = 2
n = 3
μ = 0.3
$\frac{\lambda_1}{\lambda_2}$ = 3
(b)


4.0
0.0
-4.0
$w_2(n)$
$n = 0$
$n = 1$
$n = 2$
$\mu = 0.3$
$\frac{\lambda_1}{\lambda_2} = 10$
-4.0
0.0
4.0
$w_1(n)$

(c)


8.0
0.0
-8.0
$w_2(n)$
$\mu = 0.3$
$\frac{\lambda_1}{\lambda_2} = 100$
$n = 0$
$n = 1$
-4.0
0.0
4.0
$w_1(n)$

(d)

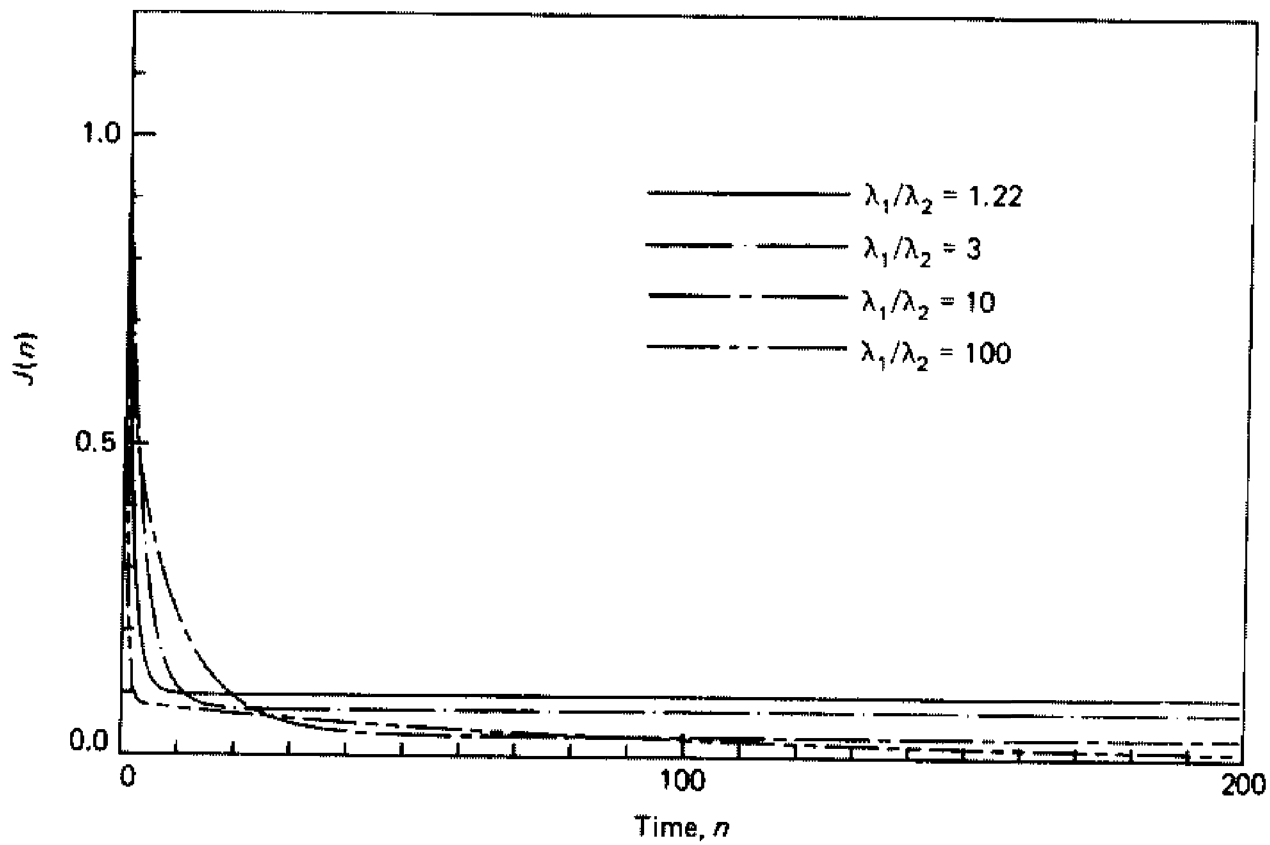


Figure 2: Learning curves of steepest-descent algorithm with step-size parameter $\mu = 0.3$ and varying eigenvalue spread.

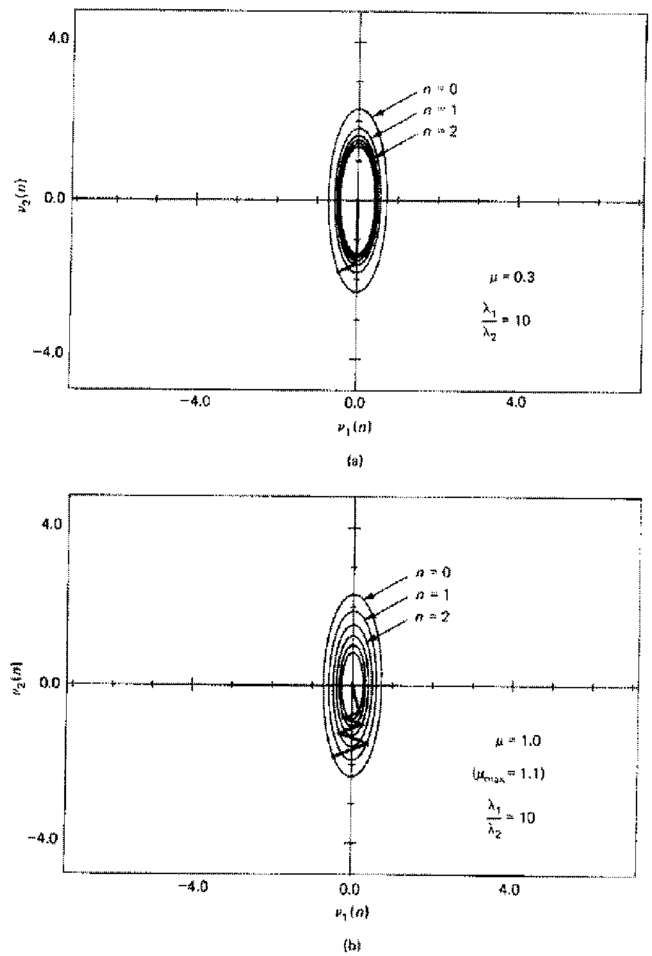


Figure 3: Loci of $v_1(n)$ versus $v_2(n)$ with $\chi(\mathbf{R}) = 10$ and varying step-sizes: (a) overdamped , $\mu = 0.3$ (b) underdamped, $\mu = 1.0$.

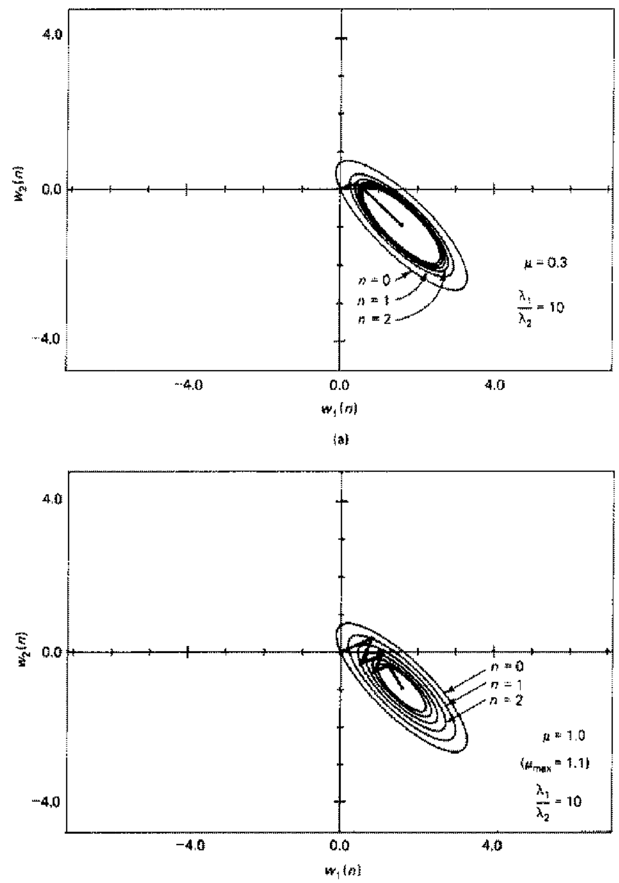
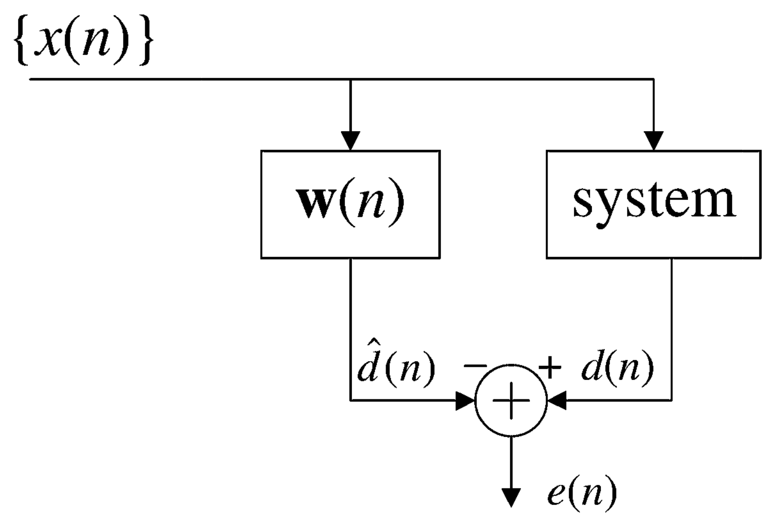


Figure 4: Loci of $w_1(n)$ versus $w_2(n)$ with $\chi(\mathbf{R}) = 10$ and varying step-sizes: (a) overdamped , $\mu = 0.3$ (b) underdamped, $\mu = 1.0$.

Example: Consider the system identification problem



For $M = 2$ suppose

$$\mathbf{R_x} = \begin{bmatrix} 1 & 0.8 \\ 0.8 & 1 \end{bmatrix} \quad \mathbf{P} = \begin{bmatrix} 0.8 \\ 0.5 \end{bmatrix}$$

From eigen analysis we have

$$\lambda_1 = 1.8, \lambda_2 = 0.2 \text{ and } \mu < \frac{2}{1.8}$$

also

$$\mathbf{q}_1 = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 \\ 1 \end{bmatrix} \quad \mathbf{q}_2 = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 \\ -1 \end{bmatrix}$$

and

$$\mathbf{Q} = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$$

Also,

$$\mathbf{w}_0 = \mathbf{R}^{-1}\mathbf{p} \begin{bmatrix} 1.11 \\ -0.389 \end{bmatrix}$$

Thus

$$\mathbf{v}(n) = \mathbf{Q}^H[\mathbf{w}(n) - \mathbf{w}_0]$$

Noting that

$$\mathbf{v}(0) = -\mathbf{Q}^H \mathbf{w}_0 = -\frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \begin{bmatrix} 1.11 \\ -0.389 \end{bmatrix} = \begin{bmatrix} 0.51 \\ 1.06 \end{bmatrix}$$

and

$$v_1(n) = (1 - \mu(1.8))^n 0.51$$
$$v_1(n) = (1 - \mu(0.2))^n 1.06$$

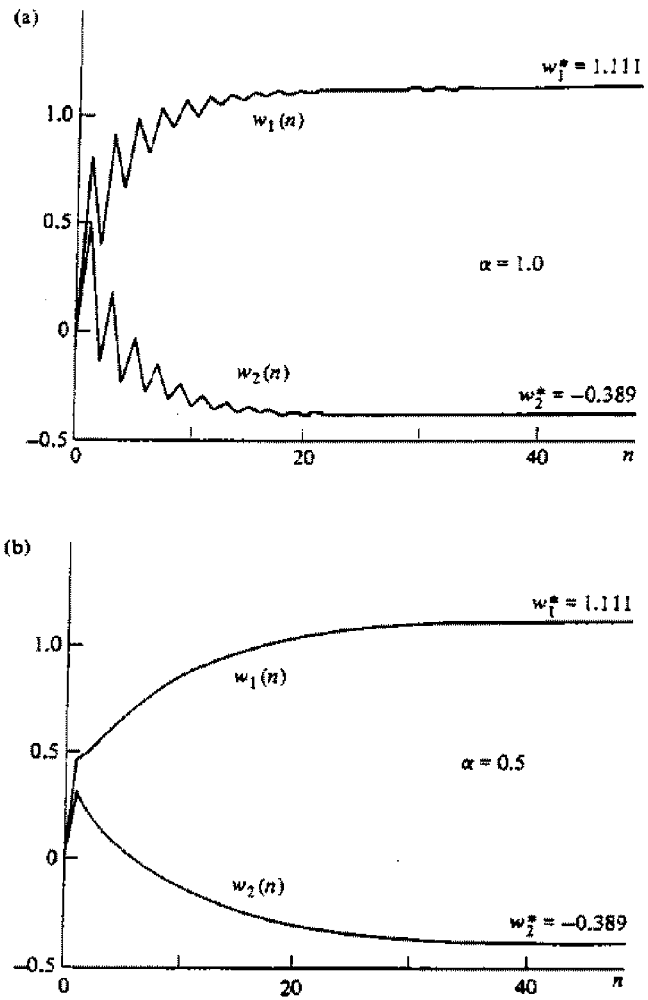


Figure 5: Convergence properties of steepest descent solution to normal equation for two α values (a) $\alpha = 1.0$, (b)$\alpha = 0.5$.

The Least Mean Square (LMS) Algorithm

The error performance surface used by the SD method is not always known a priori. We can use estimated values. The estimates are RVs and thus this leads to a stochastic approach.

We will use the following instantaneous estimates

$$\hat{\mathbf{R}}(n) = \mathbf{x}(n)\mathbf{x}^H(n)$$

$$\hat{\mathbf{p}}(n) = \mathbf{x}(n)d^*(n)$$

Recall the SD update

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

where the gradient of the error surface at $\mathbf{w}(n)$ was shown to be

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

Using the instantaneous estimates,

$$\begin{aligned}\hat{\nabla}(J(n)) &= -2\mathbf{x}(n)d^*(n) + 2\mathbf{x}(n)\mathbf{x}^H(n)\mathbf{w}(n) \\ &= -2\mathbf{x}(n)[d^*(n) - \mathbf{x}^H(n)\mathbf{w}(n)] \\ &= -2\mathbf{x}(n)[d^*(n) - \hat{d}^*(n)] \\ &= -2\mathbf{x}(n)e^*(n)\end{aligned}$$

where $e^*(n)$ is the complex conjugate of the estimate error.

Putting this in the update

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

Thus LMS algorithm belongs to the family of stochastic gradient algorithms.

The update is extremely simple while the instantaneous estimates may have large variance, the LMS algorithm is recursive and effectively averages these estimates.

The simplicity and good performance of the LMS algorithm make it the benchmark against which other optimization algorithms are judged.

The LMS algorithm can be analyzed by invoking the independence theory, which states

1. The vectors $\mathbf{x}(1), \mathbf{x}(2), \cdots, \mathbf{x}(n)$ are statistically independent.
2. $\mathbf{x}(n)$ is independent of $d(1), d(2), \cdots, d(n-1)$
3. $d(n)$ is statistically dependent on $\mathbf{x}(n)$, but is independent of $d(1), d(2), \cdots, d(n-1)$
4. $\mathbf{x}(n)$ and $d(n)$ are mutually Gaussian

The independence theorem is justified in some cases, e.g. beamforming where we receive independent vector observations. In other cases it is not well justified, but allows the analysis to proceeds.

Using the independence theory we can show that $\mathbf{w}(n)$ converges to the optimal solution in the mean

$$\lim_{n\to\infty} E\{\mathbf{w}(n)\} = \mathbf{w}_0$$

In certain cases, to show this, evaluate the update

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

$$\mathbf{w}(n+1) - \mathbf{w}_0 = \mathbf{w}(n) - \mathbf{w}_0 + \mu\mathbf{x}(n)e^*(n)$$

$$\begin{aligned}
\mathbf{c}(n+1) &= \mathbf{c}(n) + \mu\mathbf{x}(n)(d^*(n) - \mathbf{x}^H(n)\mathbf{w}(n)) \\
&= \mathbf{c}(n) + \mu\mathbf{x}(n)d^*(n) - \mu\mathbf{x}(n)\mathbf{x}^H(n)[\mathbf{w}(n) - \mathbf{w}_0 + \mathbf{w}_0] \\
&= \mathbf{c}(n) + \mu\mathbf{x}(n)d^*(n) - \mu\mathbf{x}(n)\mathbf{x}^H(n)\mathbf{c}(n) - \mu\mathbf{x}(n)\mathbf{x}^H(n)\mathbf{w}_0 \\
&= [\mathbf{I} - \mu\mathbf{x}(n)\mathbf{x}^H(n)]\mathbf{c}(n) + \mu\mathbf{x}(n)[d^*(n) - \mathbf{x}^H(n)\mathbf{w}_0]
\end{aligned}$$

Note that since $\mathbf{w}(n)$ is based on past inputs desired responses, $\mathbf{w}(n)$ (and $\mathbf{c}(n)$) is independent of $\mathbf{x}(n)$.

Thus

$$\mathbf{c}(n+1) = [\mathbf{I} - \mu\mathbf{x}(n)\mathbf{x}^H(n)]\mathbf{c}(n) + \mu\mathbf{x}(n)e_0^*(n)$$

$$\Downarrow$$

$$E\{\mathbf{c}(n+1)\} = (\mathbf{I} - \mu\mathbf{R})E\{\mathbf{c}(n)\} + \mu \underbrace{E\{\mathbf{x}(n)e_0^*(n)\}}_{\text{zero, why?}}$$

$$E\{\mathbf{c}(n+1)\} = (\mathbf{I} - \mu\mathbf{R})E\{\mathbf{c}(n)\}$$

Using arguments similar to the SD case we have

$$\lim_{n\to\infty} E\{\mathbf{c}(n)\} = 0 \quad \text{if} \quad 0 < \mu < \frac{2}{\lambda_{\max}}$$

or equivalently

$$\lim_{n\to\infty} E\{\mathbf{w}(n)\} = \mathbf{w}_0 \quad \text{if} \quad 0 < \mu < \frac{2}{\lambda_{\max}}$$

Noting that

$$\lambda_{\max} \leq \text{trace}[\mathbf{R}] = Mr(0) = M\sigma_x^2$$

a more conservative bound is

$$0 < \mu < \frac{2}{M\sigma_x^2}$$

Also, convergence in the mean

$$\lim_{n\to\infty} E\{\mathbf{w}(n)\} = \mathbf{w}_0$$

is a weak condition that says nothing about the variance, which may even grow.

A stronger condition is convergence in the mean square, which says

$$\lim_{n\to\infty} E\{|\mathbf{c}(n)|^2\} = \text{constant}$$

An equivalent condition is to show that

$$\lim_{n\to\infty} J(n) = \lim_{n\to\infty} E\{|\mathbf{e}(n)|^2\} = \text{constant}$$

write $e(n)$ as

$$\begin{aligned} e(n) &= d(n) - \hat{d}(n) = d(n) - \mathbf{w}^H(n)\mathbf{x}(n) \\ &= d(n) - \mathbf{w}_0^H\mathbf{x}(n) - \mathbf{c}^H(n)\mathbf{x}(n) \\ &= e_0(n) - \mathbf{c}^H(n)\mathbf{x}(n) \end{aligned}$$

Thus

$$\begin{aligned} J(n) &= E\{|e(n)|^2\} \\ &= E\{e_0(n) - \mathbf{c}^H(n)\mathbf{x}(n)(e_0^*(n) - \mathbf{x}^H(n)\mathbf{c}(n))\} \\ &= J_{\min} + \underbrace{E\{\mathbf{c}^H(n)\mathbf{x}(n)\mathbf{x}^H(n)\mathbf{c}(n)\}}_{J_{\text{ex}}(n)} \\ &= J_{\min} + J_{\text{ex}}(n) \end{aligned}$$

Since $J_{\text{ex}}(n)$ is a scalar

$$\begin{aligned} J_{\text{ex}}(n) &= E\{\mathbf{c}^H(n)\mathbf{x}(n)\mathbf{x}^H(n)\mathbf{c}(n)\} \\ &= E\{\text{trace}[\mathbf{c}^H(n)\mathbf{x}(n)\mathbf{x}^H(n)\mathbf{c}(n)]\} \\ &= E\{\text{trace}[\mathbf{x}(n)\mathbf{x}^H(n)\mathbf{c}(n)\mathbf{c}^H(n)]\} \\ &= \text{trace}[E\{\mathbf{x}(n)\mathbf{x}^H(n)\mathbf{c}(n)\mathbf{c}^H(n)\}] \end{aligned}$$

Invoking the independence theorem

$$\begin{aligned} J_{\text{ex}}(n) &= \text{trace}[E\{\mathbf{x}(n)\mathbf{x}^H(n)\}E\{\mathbf{c}(n)\mathbf{c}^H(n)\}] \\ &= \text{trace}[\mathbf{R}\mathbf{K}(n)] \end{aligned}$$

where

$$\mathbf{K}(n) = E\{\mathbf{c}(n)\mathbf{c}^H(n)\}$$

Thus

$$J(n) = J_{\min} + J_{\text{ex}}(n)$$

$$= \quad J_{\min} + \text{trace}[\mathbf{R}\mathbf{K}(n)]$$

Recall

$$\mathbf{Q}^H\mathbf{R}\mathbf{Q} = \mathbf{\Omega} \quad \text{or} \quad \mathbf{R} = \mathbf{Q}\mathbf{\Omega}\mathbf{Q}^H$$

Let

$$\mathbf{Q}^H\mathbf{K}(n)\mathbf{Q} \equiv \mathbf{S}(n)$$

where $\mathbf{S}(n)$ need not be diagonal. Then

$\mathbf{K}(n) = \mathbf{Q}\mathbf{S}(n)\mathbf{Q}^H$ and

$$\begin{aligned} J_{\text{ex}}(n) &= \text{trace}[\mathbf{R}\mathbf{K}(n)] \\ &= \text{trace}[\mathbf{Q}\mathbf{\Omega}\mathbf{Q}^H\mathbf{Q}\mathbf{S}(n)\mathbf{Q}^H] \\ &= \text{trace}[\mathbf{Q}\mathbf{\Omega}\mathbf{S}(n)\mathbf{Q}^H] \\ &= \text{trace}[\mathbf{Q}^H\mathbf{Q}\mathbf{\Omega}\mathbf{S}(n)] \\ &= \text{trace}[\mathbf{\Omega}\mathbf{S}(n)] \end{aligned}$$

Since $\mathbf{\Omega}$ is diagonal

$$J_{\text{ex}}(n) = \text{trace}[\mathbf{\Omega}\mathbf{S}(n)] = \sum_{i=1}^{M} \lambda_i s_i(n)$$

where $s_1(n), s_2(n), \cdots, s_M(n)$ are the diagonal elements of $\mathbf{S}(n)$.

The recursion expression can be modified to yield a recursion on $\mathbf{S}(n)$, which is

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

which for the diagonal elements is

$$s_i(n+1) = (1-\mu\lambda_i)^2 s_i(n) + \mu^2 J_{\min}\lambda_i \quad i = 1, 2, \cdots, M$$

Suppose $J_{\text{ex}}(n)$ converges, then $s_i(n+1) = s_i(n)$ and from the above

$$s_i(n) \quad = \quad \frac{\mu^2 J_{\min}\lambda_i}{1-(1-\mu\lambda_i)^2} = \frac{\mu^2 J_{\min}\lambda_i}{2\mu\lambda_i - \mu^2\lambda_i^2}$$

$$= \frac{\mu J_{\min}}{2 - \mu\lambda_i} \quad i = 1, 2, \cdots, M$$

Utilizing

$$J_{\mathrm{ex}}(n) = \mathrm{trace}[\mathbf{\Omega S}(n)] = \sum_{i=1}^{M} \lambda_i s_i(n)$$

we see

$$\lim_{n\to\infty} J_{\mathrm{ex}}(n) = J_{\min} \sum_{i=1}^{M} \frac{\mu\lambda_i}{2 - \mu\lambda_i}$$

The LMS misadjustment is defined

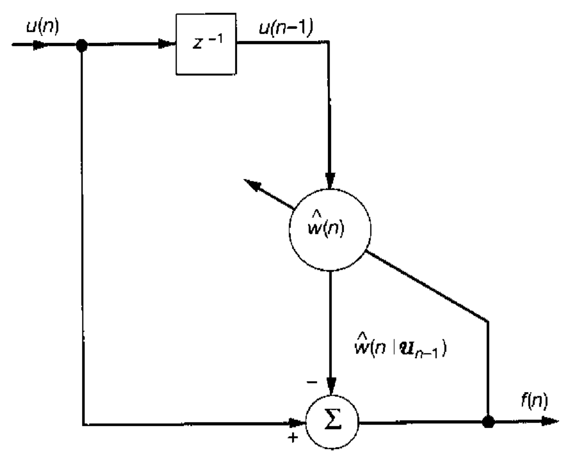
$$M = \frac{\lim_{n\to\infty} J_{\mathrm{ex}}(n)}{J_{\min}} = \sum_{i=1}^{M} \frac{\mu\lambda_i}{2 - \mu\lambda_i}$$

A misadjustment at 10% or less is generally considered acceptable.

Example: one tap predictor of order one AR process. Let

$$x(n) = -ax(n-1) + v(n)$$

and use a one tap predictor.



The weight update is

$$\begin{aligned} w(n+1) &= w(n) + \mu x(n+1)e(n) \\ &= w(n) + \mu x(n-1)[x(n) - w(n)x(n-1)] \end{aligned}$$

Note $w_0 = -a$ consider two cases and set $\mu = 0.05$.

a	σ_x^2
-0.99	0.93627
0.99	0.995

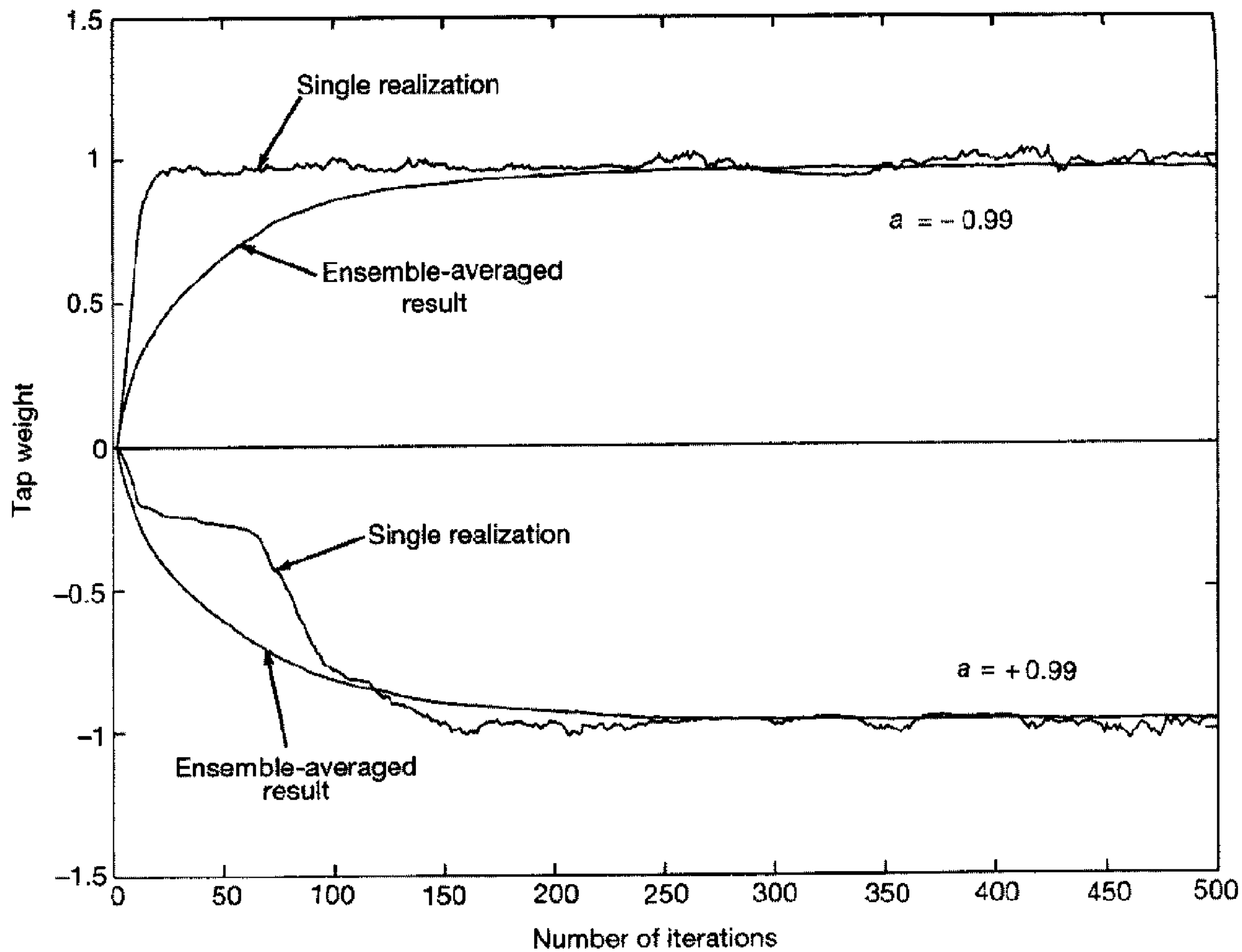


Figure 6: Transient behavior of weight $\hat{w}(n)$ of adaptive first-order predictor.

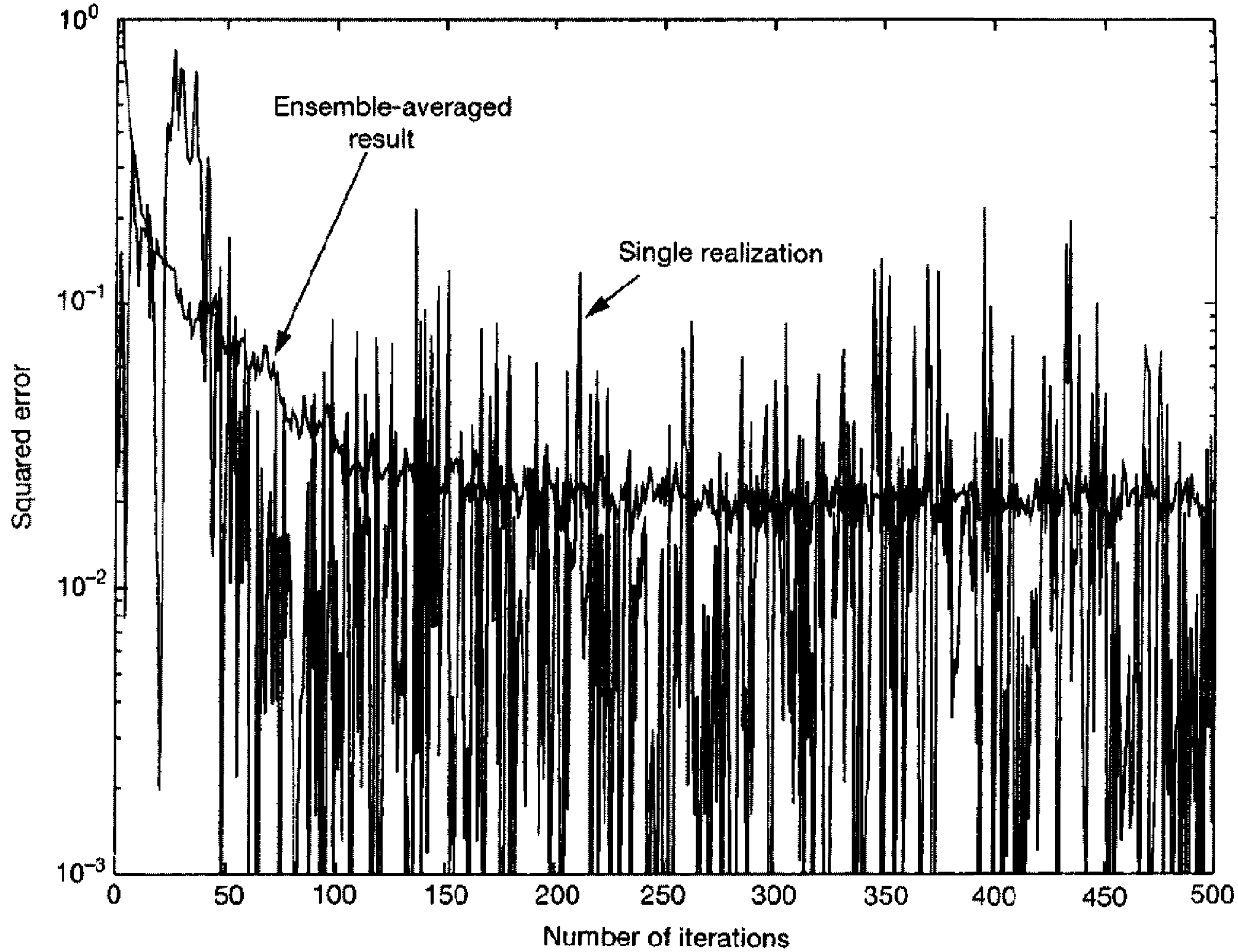


Figure 7: Transient behavior of squared prediction error in adaptive first-order predictor for $\mu = 0.05$.

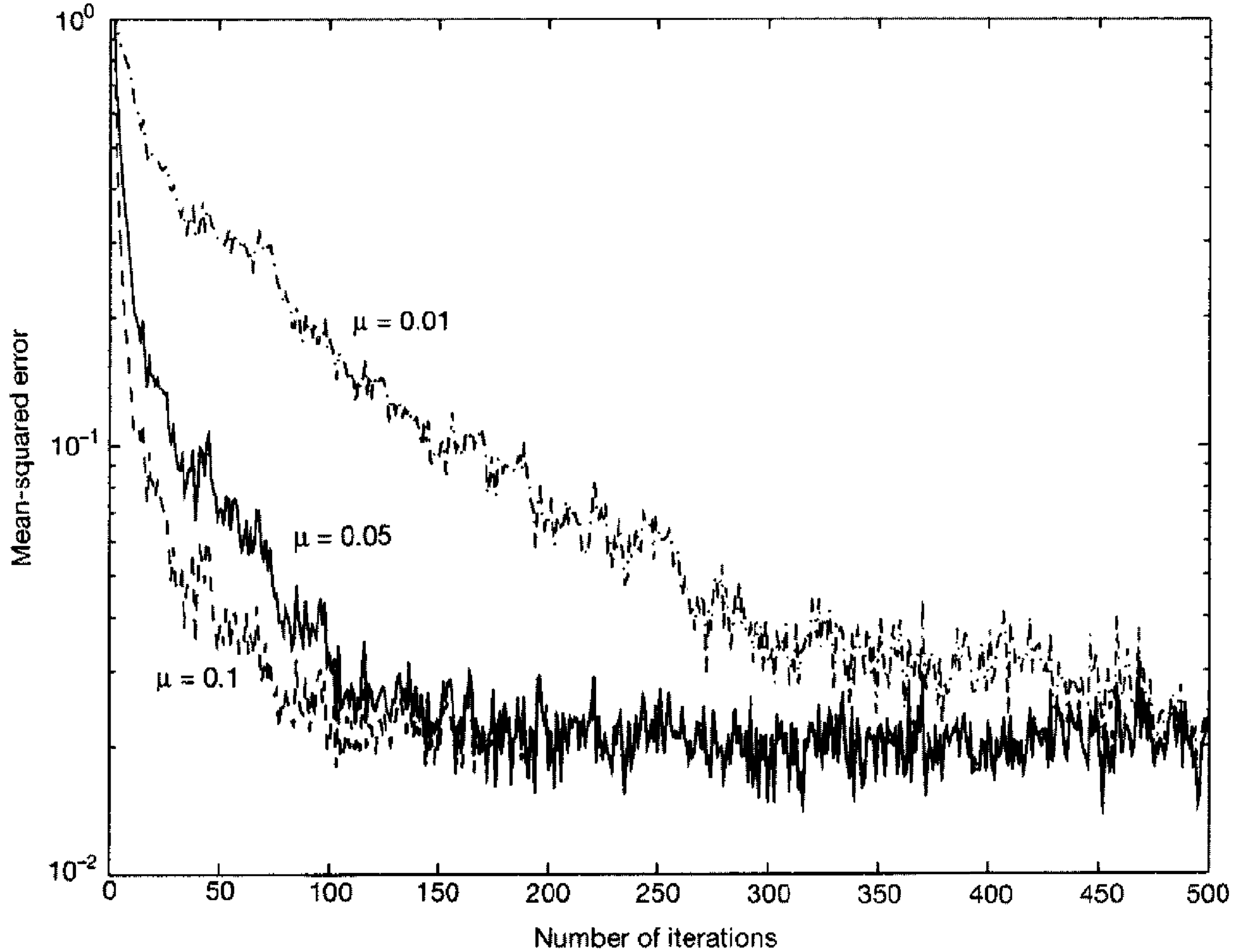


Figure 8: Experimental learning curves of adaptive first-order predictor for varying step-size parameter μ.

Consider the expected trajectory of $w(n)$.

Recall

$$\begin{aligned} w(n+1) &= w(n) + \mu x(n-1)e(n) \\ &= w(n) + \mu x(n-1)[x(n) - w(n)x(n-1)] \\ &= [1 - \mu x(n-1)x(n-1)]w(n) + \mu x(n-1)x(n) \end{aligned}$$

Since $x(n) = -ax(n-1) + v(n)$

$$\begin{aligned} w(n+1) &= [1 - \mu x(n-1)x(n-1)]w(n) + \mu x(n-1)[-ax(n-1) \\ &\quad +v(n)] \\ &= [1 - \mu x(n-1)x(n-1)]w(n) - \mu a x(n-1)x(n-1) \\ &\quad +\mu x(n-1)v(n) \end{aligned}$$

Taking the expectation and invoking the dependence theorem

$$E\{w(n+1)\} = (1 - \mu\sigma_x^2)E\{w(n)\} - \mu\sigma_x^2 a$$

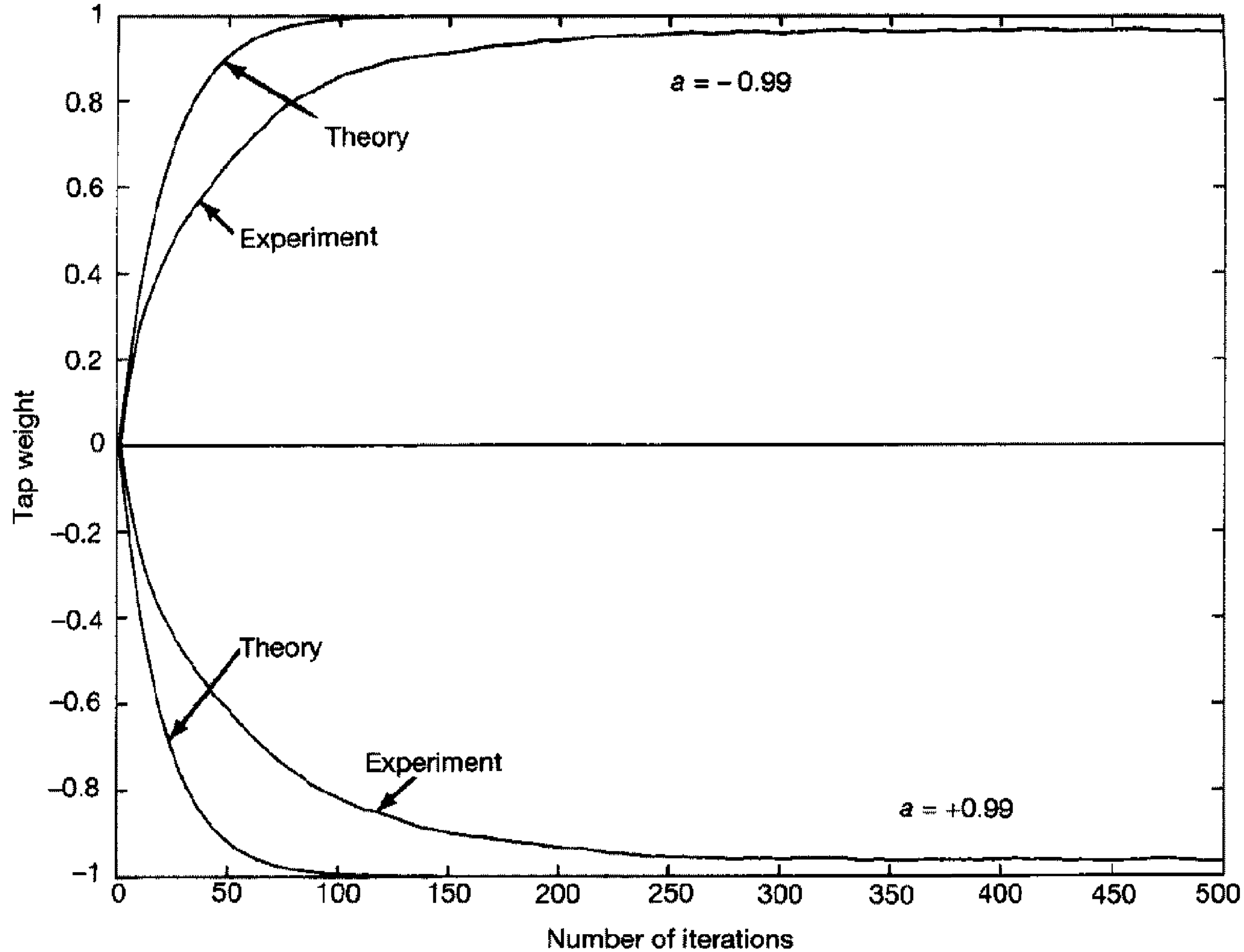


Figure 9: Comparison of experimental results with theory, based on $\hat{w}(n)$.

We can also derive a theoretical expression for $J(n)$.

Note that the initial value of $J(n)$ is

$$J(0) = \sigma_x^2$$

and the final value is

$$J(\infty) = J_{\min} + J_{\mathrm{ex}} = \sigma_v^2 + J_{\min}\frac{\mu\lambda_1}{2-\mu\lambda_1}$$

if μ is small

$$J(\infty) = \sigma_v^2 + \sigma_v^2\left(\frac{\mu\sigma_x^2}{2}\right) = \sigma_v^2\left(1+\frac{\mu\sigma_x^2}{2}\right)$$

Also, the time constant is

$$\tau = -\frac{1}{2\ln(1-\mu\lambda_1)} = -\frac{1}{2\ln(1-\mu\sigma_x^2)} \approx \frac{1}{2\mu\sigma_x^2}$$

$$J(n) = [\sigma_x^2 - \sigma_v^2(1+\frac{\mu}{2}\sigma_x^2)](1-\mu\sigma_x^2)^{2n} + \sigma_v^2(1+\frac{\mu}{2}\sigma_x^2)$$

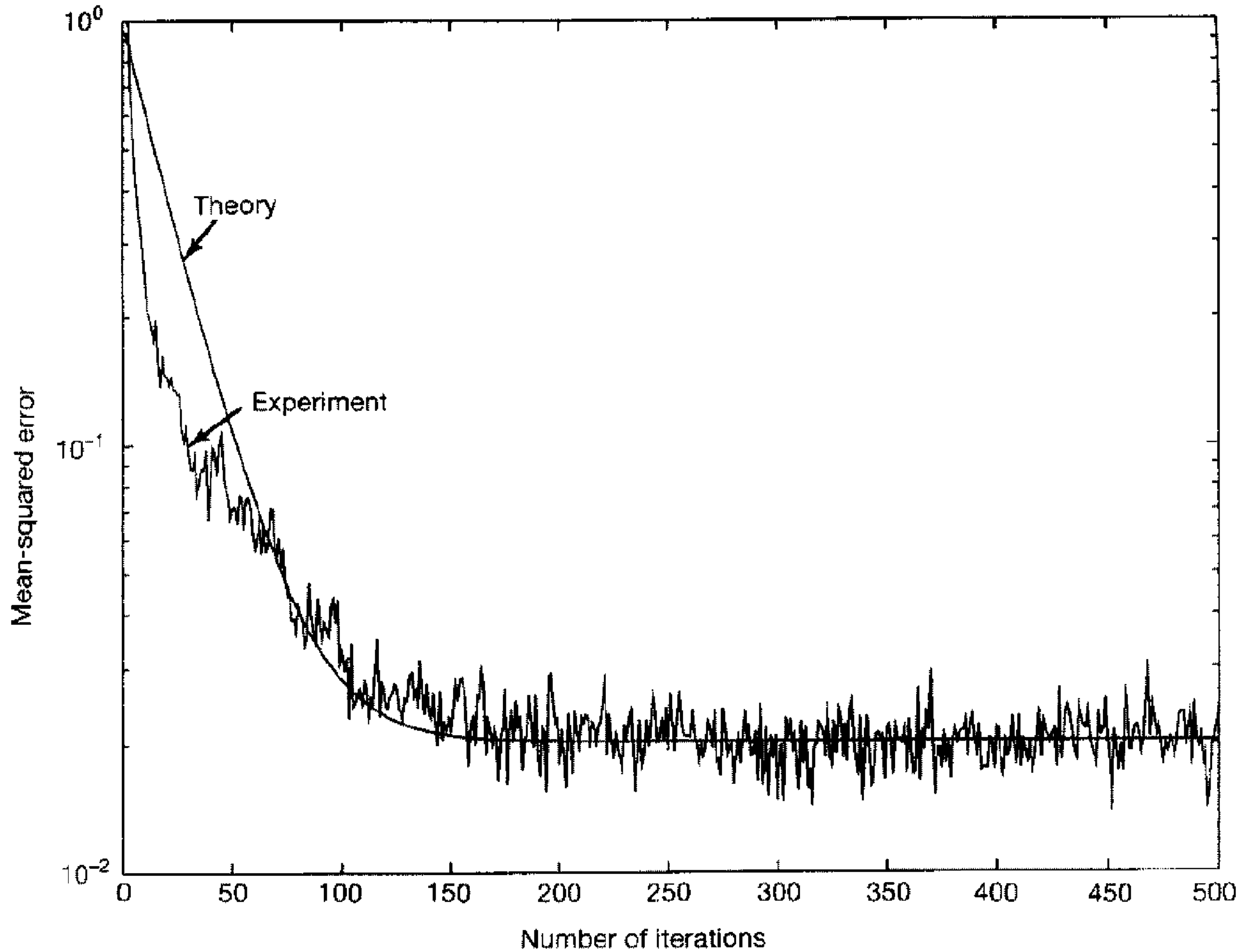
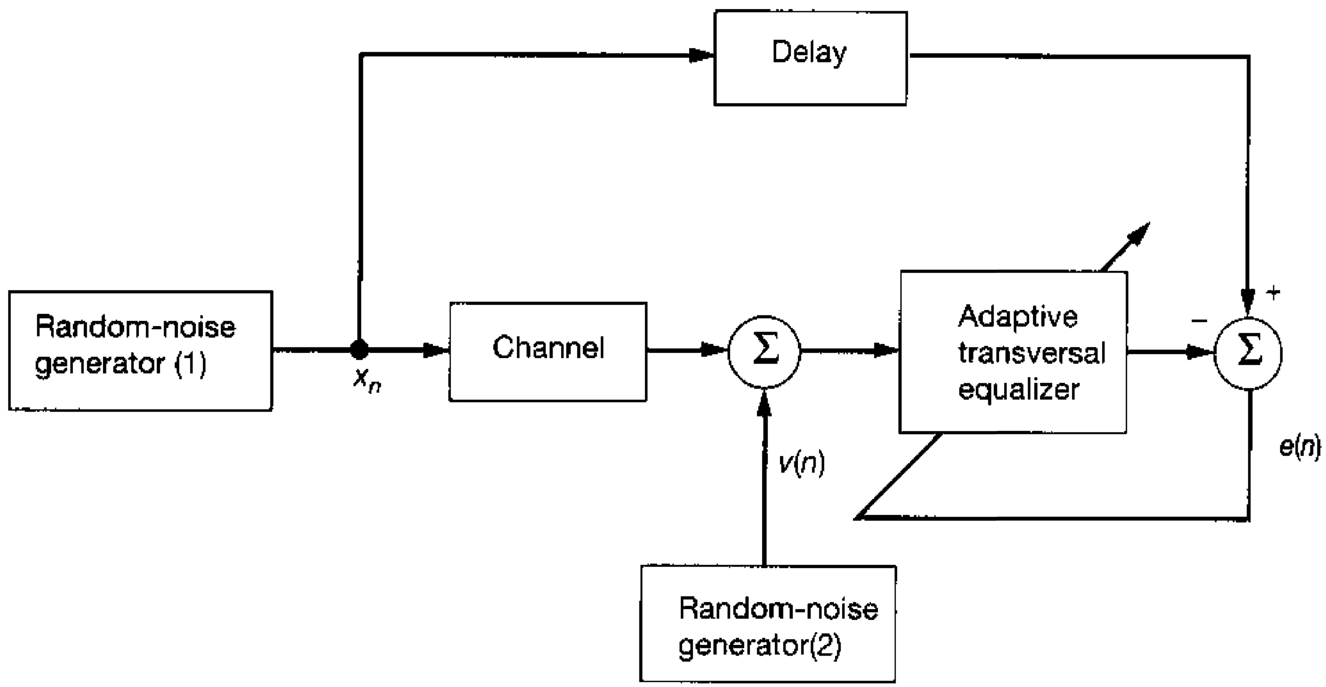


Figure 10: Comparison of experimental results with theory for the adaptive predictor, based on the mean-square error for $\mu = 0.001$.

Example: Adaptive equalization

Goal: Pass known signal through unknown channel to invert effects of channel and noise on signal.



The signal is a Bernoulli sequence

$$x_n = \begin{cases} +1 & \text{with probability } 1/2 \\ -1 & \text{with probability } 1/2 \end{cases}$$

The channel has a raised cosine response

$$h_n = \begin{cases} \frac{1}{2}\left[1 + \cos\left(\frac{2\pi}{w}(n-2)\right)\right] & n = 1, 2, 3 \\ 0 & \text{otherwise} \end{cases}$$

Note that w controls the eigenvalue spread $\chi(\mathbf{R})$.

Also the additive noise is $\sim N(0, 0.001)$.

Note that h_n is symmetric about $n = 2$ and thus introduces a delay of 2.
We will use an $M = 11$ tap filter, which will be symmetric about $n = 5$ and introduce a delay of 5.

Thus an overall delay of $\delta = 5 + 2 = 7$ is added to the system.

Channel response and Filter response

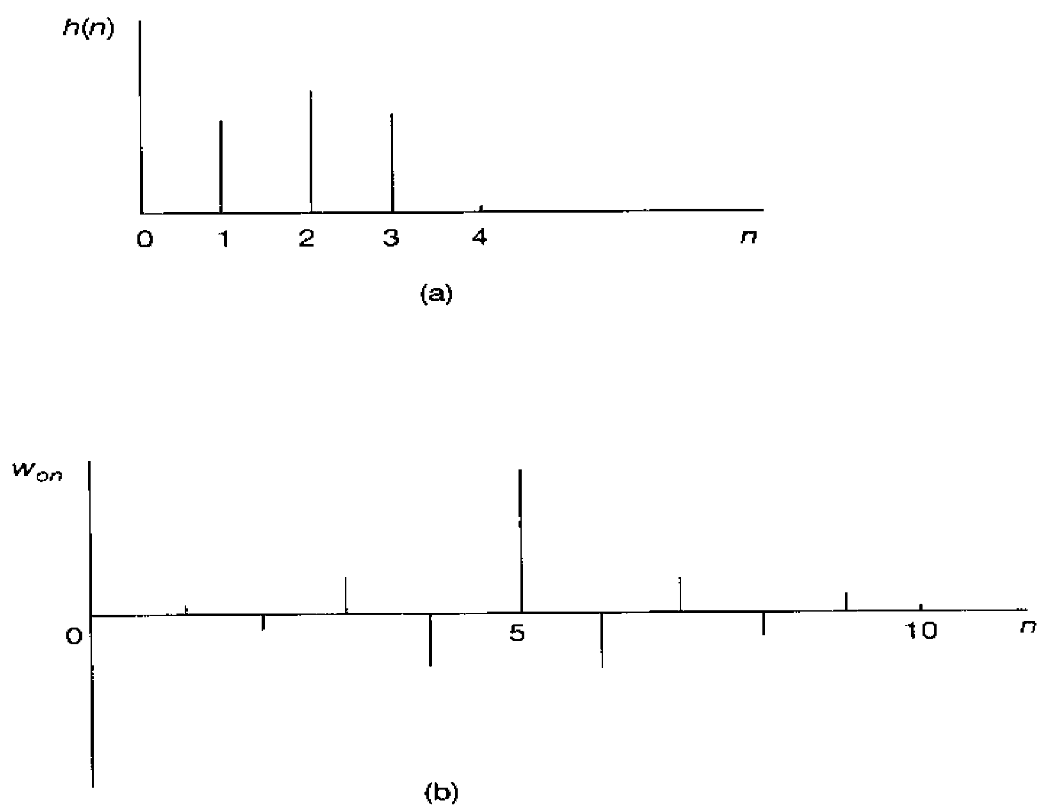


Figure 11: (a) Impulse response of channel; (b) impulse response of optimum transversal equalizer.

Consider three w values

TABLE 9.1 SUMMARY OF PARAMETERS FOR THE EXPERIMENT ON ADAPTIVE EQUALIZATION

W	2.9	3.1	3.3	3.5
$r(0)$	1.0963	1.1568	1.2264	1.3022
$r(1)$	0.4388	0.5596	0.6729	0.7774
$r(2)$	0.0481	0.0783	0.1132	0.1511
λ_{min}	0.3339	0.2136	0.1256	0.0656
λ_{max}	2.0295	2.3761	2.7263	3.0707
$\chi(\mathbf{R}) = \lambda_{max}/\lambda_{min}$	6.0782	11.1238	21.7132	46.8216

Note step size is bound by $w = 3.5$ case

$$\mu \leq \frac{2}{Mr(0)} = \frac{2}{11(1.3022)} = 0.14$$

Choose $\mu = 0.075$ in all cases.

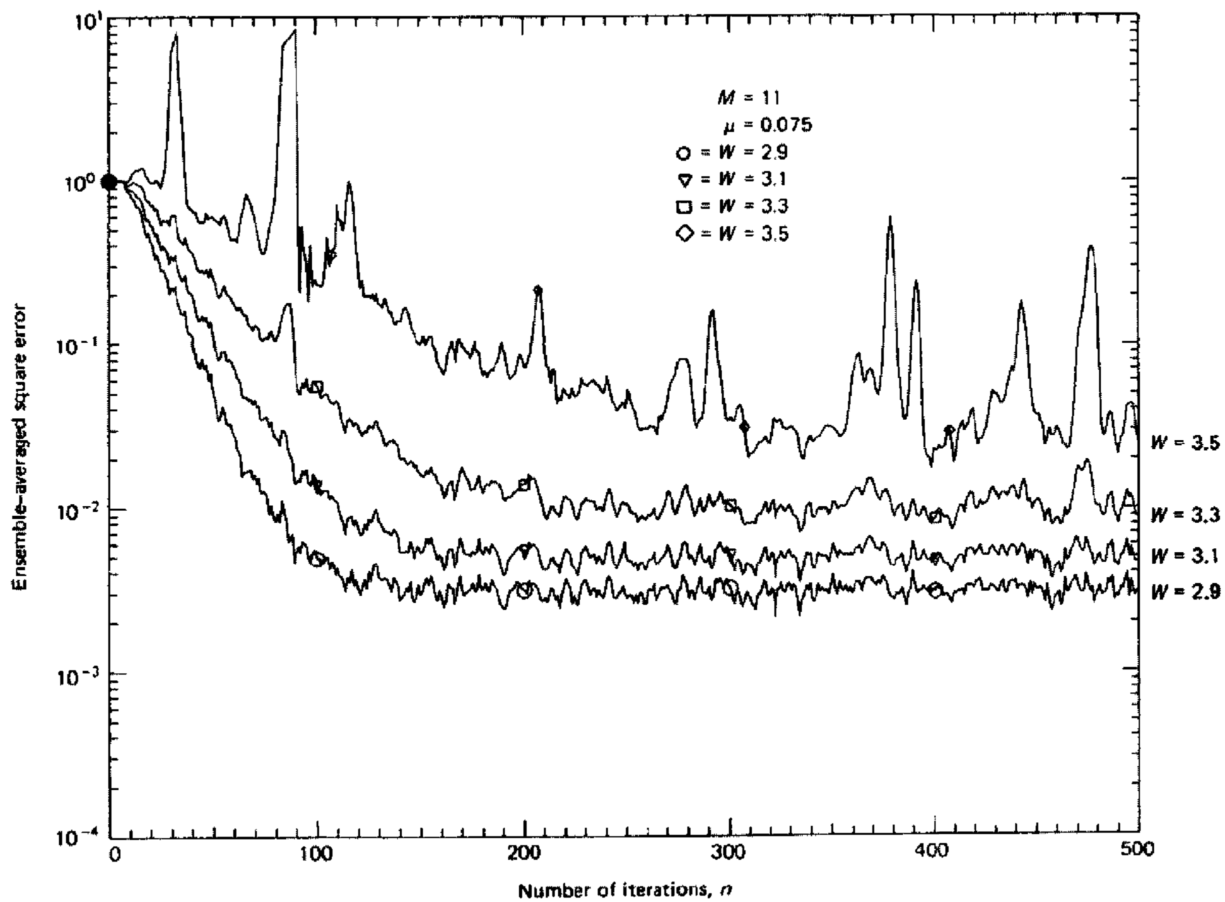


Figure 12: Learning curves of the LMS algorithm for an adaptive equalizer with number of taps $M = 11$, step-size parameter $\mu = 0.075$, and varying eigenvalue spread $\chi(\mathbf{R})$.

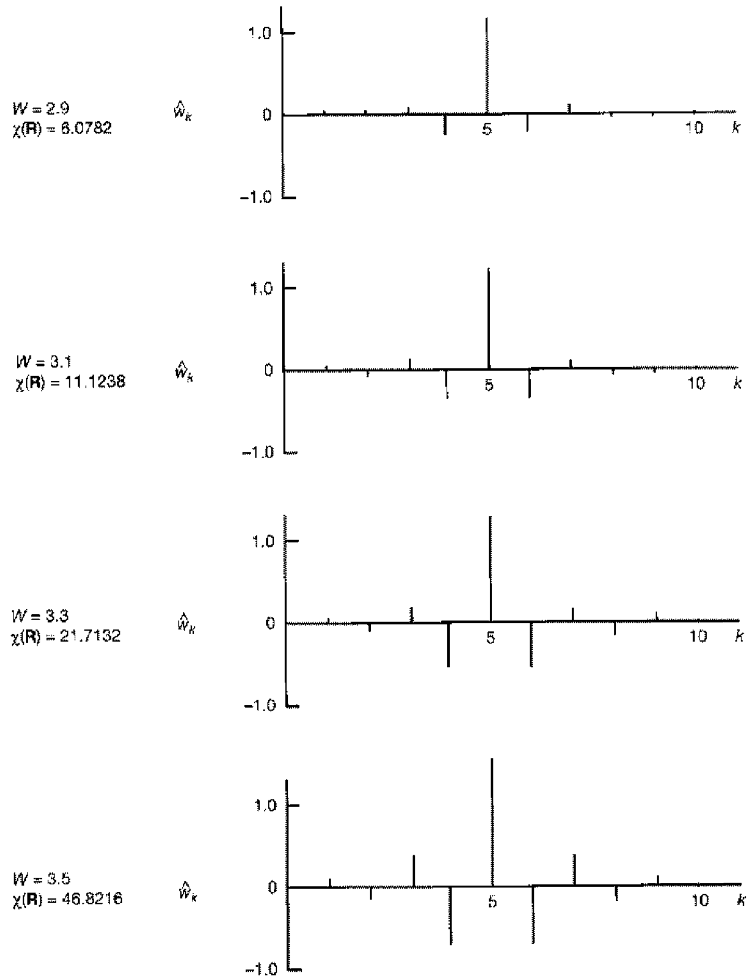


Figure 13: Ensemble-average impulse response of the adaptive equalizer (after 1000 iterations) for each of four different eigenvalue spreads.

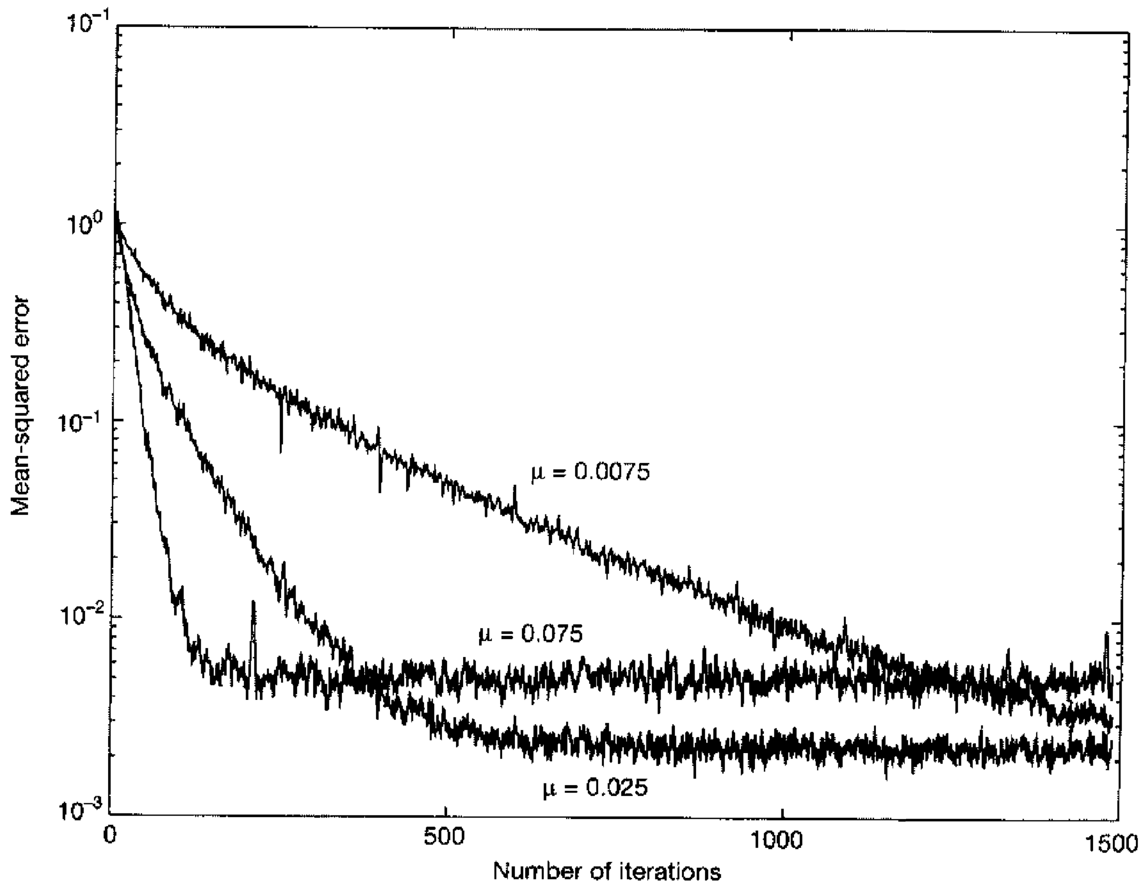


Figure 14: Learning curves of the LMS algorithm for an adaptive equalizer with the number of taps $M = 11$, fixed eigenvalue spread, and varying step-size parameter μ.

Example: Directionality of the LMS algorithm

- The speed of convergence of the LMS algorithm is faster in certain directions in the weight space.
- If the convergence is in the appropriate direction, the convergence can be accelerated by increased eigenvalue spread.

Consider the deterministic signal

$$x(n) = A_1 \cos(\omega_1 n) + A_2 \cos(\omega_2 n)$$

with

$$\mathbf{R} = \frac{1}{2} \begin{bmatrix} A_1^2 + A_2^2 & A_1^2 \cos(\omega_1) + A_2^2 \cos(\omega_2) \\ A_1^2 \cos(\omega_1) + A_2^2 \cos(\omega_2) & A_1^2 + A_2^2 \end{bmatrix}$$

which gives

$$\lambda_1 = \frac{1}{2} A_1^2 (1 + \cos(\omega_1)) + \frac{1}{2} A_2^2 (1 + \cos(\omega_2))$$

$$\lambda_2 = \frac{1}{2}A_1^2(1-\cos(\omega_1)) + \frac{1}{2}A_2^2(1-\cos(\omega_2))$$

and

$$\mathbf{q}_1 = \begin{bmatrix} 1 \\ 1 \end{bmatrix} \quad \mathbf{q}_2 = \begin{bmatrix} -1 \\ 1 \end{bmatrix}$$

Consider two cases:

$$x_a(n) = \cos(1.2n) + 0.5\cos(0.1n) \quad \text{and} \quad \chi(\mathbf{R}) = 2.9$$

$$x_b(n) = \cos(0.6n) + 0.5\cos(0.23n) \quad \text{and} \quad \chi(\mathbf{R}) = 12.9$$

In each case let

$$\mathbf{p} = \lambda_1\mathbf{q}_1 \Rightarrow \mathbf{R}\mathbf{w}_0 = \lambda_1\mathbf{q}_1 \Rightarrow \mathbf{w}_0 = \mathbf{q}_1 = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$$

$$\mathbf{p} = \lambda_2\mathbf{q}_2 \Rightarrow \mathbf{R}\mathbf{w}_0 = \lambda_2\mathbf{q}_2 \Rightarrow \mathbf{w}_0 = \mathbf{q}_2 = \begin{bmatrix} -1 \\ 1 \end{bmatrix}$$

Look at 200 iterations of the algorithm.

Look at minimum eigenfilter first, $\mathbf{w}_0 = \mathbf{q}_2 = \begin{bmatrix} -1 \\ 1 \end{bmatrix}$ Then maximum eigenfilter, $\mathbf{w}_0 = \mathbf{q}_1 = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$

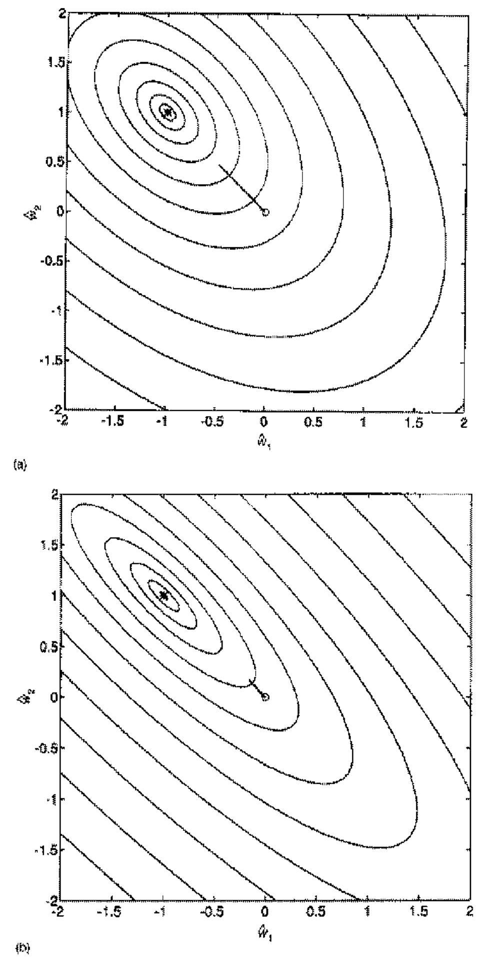


Figure 15: Convergence of the LMS algorithm, for a deterministic sinusoidal process, along "slow" eigenvector (i.e., minimum eigenfilter) for (a)input $u_a(n)$ and (b)input $u_b(n)$.

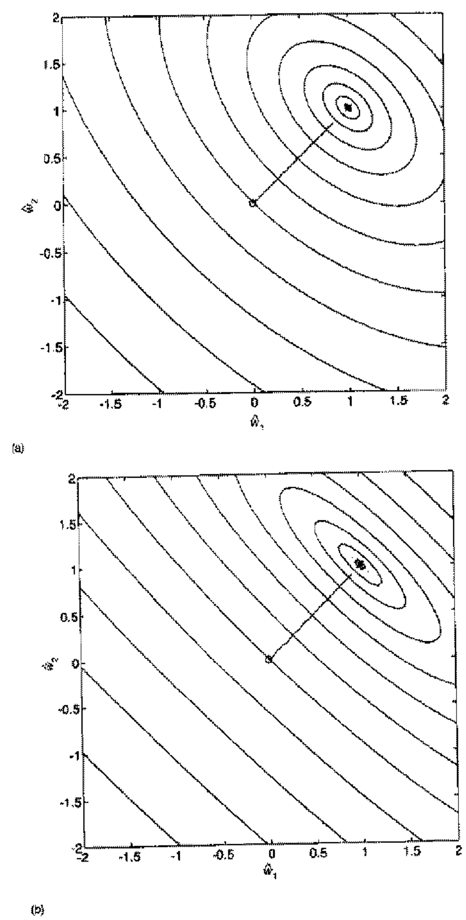


Figure 16: Convergence of the LMS algorithm, for a deterministic sinusoidal process, along "fast" eigenvector (i.e., minimum eigenfilter) for (a)input $u_a(n)$ and (b)input $u_b(n)$.

Normalized LMS Algorithm

In the standard LMS algorithm the correction is proportional to $\mu\mathbf{x}(n)e^*(n)$

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

If $\mathbf{x}(n)$ is large, the update suffers from gradient noise amplification. The normalized LMS algorithm seeks to avoid gradient noise amplification

- The step size is made time varying, $\mu(n)$, and optimized to minimize error.

Thus let

$$\begin{aligned}\mathbf{w}(n+1) &= \mathbf{w}(n) + \frac{1}{2}\mu(n)[-\nabla(n)] \\ &= \mathbf{w}(n) + \mu(n)[\mathbf{p} - \mathbf{R}\mathbf{w}(n)]\end{aligned}$$

Choose $\mu(n)$, such that the updated $\mathbf{w}(n+1)$ produces the minimum MSE,

$$J(n+1) = E\{|e(n+1)|^2\}$$

where

$$e(n+1) = d(n+1) - \mathbf{w}^H(n+1)\mathbf{x}(n+1)$$

Thus we choose $\mu(n)$ such that it minimizes $J(n+1)$.

The optimal step size, $\mu_0(n)$, will be a function of $\mathbf{R}$ and $\nabla(n)$. As before, we use instantaneous estimates of these values.

To determine $\mu_0(n)$, expand $J(n+1)$

$$\begin{aligned} J(n+1) &= E\{e(n+1)e^*(n+1)\} \\ &= E\{(d(n+1) - \mathbf{w}^H(n+1)\mathbf{x}(n+1)) \\ &\quad (d^*(n+1) - \mathbf{x}^H(n+1)\mathbf{w}(n+1))\} \\ &= \sigma_d^2 - \mathbf{w}^H(n+1)\mathbf{p} - \mathbf{p}^H\mathbf{w}(n+1) \\ &\quad +\mathbf{w}^H(n+1)\mathbf{R}\mathbf{w}(n+1) \end{aligned}$$

Now use the fact that $\mathbf{w}(n+1) = \mathbf{w}(n) - \frac{1}{2}\mu(n)\nabla(n)$

$$
\begin{aligned}
J(n+1) \quad = \quad & \sigma_d^2 - \left[\mathbf{w}(n) - \frac{1}{2}\mu(n)\nabla(n)\right]^H \mathbf{p} \\
& -\mathbf{p}^H \left[\mathbf{w}(n) - \frac{1}{2}\mu(n)\nabla(n)\right] \\
& + \underbrace{\left[\mathbf{w}(n) - \frac{1}{2}\mu(n)\nabla(n)\right]^H \mathbf{R} \left[\mathbf{w}(n) - \frac{1}{2}\mu(n)\nabla(n)\right]} \\
& \qquad = \mathbf{w}^H(n)\mathbf{R}\mathbf{w}(n) - \tfrac{1}{2}\mu(n)\mathbf{w}^H(n)\mathbf{R}\nabla(n) \\
& \qquad -\tfrac{1}{2}\mu(n)\nabla^H(n)\mathbf{R}\mathbf{w}(n) + \tfrac{1}{4}\mu^2(n)\nabla^H(n)\mathbf{R}\nabla(n)
\end{aligned}
$$

$$\begin{aligned}
J(n+1) &= \sigma_d^2 - \left[\mathbf{w}(n) - \frac{1}{2}\mu(n)\nabla(n)\right]^H \mathbf{p} \\
&\quad -\mathbf{p}^H \left[\mathbf{w}(n) - \frac{1}{2}\mu(n)\nabla(n)\right] \\
&\quad +\mathbf{w}^H(n)\mathbf{R}\mathbf{w}(n) - \frac{1}{2}\mu(n)\mathbf{w}^H(n)\mathbf{R}\nabla(n) \\
&\quad -\frac{1}{2}\mu(n)\nabla^H(n)\mathbf{R}\mathbf{w}(n) + \frac{1}{4}\mu^2(n)\nabla^H(n)\mathbf{R}\nabla(n)
\end{aligned}$$

Differentiating with respect to $\mu(n)$,

$$\begin{aligned}
\frac{\partial J(n+1)}{\partial \mu(n)} &= \frac{1}{2}\nabla^H(n)\mathbf{p} + \frac{1}{2}\mathbf{p}^H\nabla(n) - \frac{1}{2}\mathbf{w}^H\mathbf{R}\nabla(n) \\
&\quad -\frac{1}{2}\nabla^H(n)\mathbf{R}\mathbf{w}(n) + \frac{1}{2}\mu(n)\nabla^H(n)\mathbf{R}\nabla(n)
\end{aligned}$$

Setting it equal to 0

$$
\begin{aligned}
\mu_0(n)\nabla^H(n)\mathbf{R}\nabla(n) &= \mathbf{w}^H(n)\mathbf{R}\nabla(n) - \mathbf{p}^H\nabla(n) \\
&\quad + \nabla^H(n)\mathbf{R}\mathbf{w}(n) - \nabla^H(n)\mathbf{p}
\end{aligned}
$$

$$
\begin{aligned}
\mu_0(n) &= \frac{[\mathbf{w}^H(n)\mathbf{R} - \mathbf{p}^H]\nabla(n) + \nabla^H(n)[\mathbf{R}\mathbf{w}(n) - \mathbf{p}]}{\nabla^H(n)\mathbf{R}\nabla(n)} \\
&= \frac{[\mathbf{R}\mathbf{w} - \mathbf{p}]^H\nabla(n) + \nabla^H(n)[\mathbf{R}\mathbf{w}(n) - \mathbf{p}]}{\nabla^H(n)\mathbf{R}\nabla(n)} \\
&= \frac{\frac{1}{2}\nabla^H(n)\nabla(n) + \frac{1}{2}\nabla^H(n)\nabla^H(n)}{\nabla^H(n)\mathbf{R}\nabla(n)} \\
&= \frac{\nabla^H(n)\nabla(n)}{\nabla^H(n)\mathbf{R}\nabla(n)}
\end{aligned}
$$

Using instantaneous estimates

$$\hat{\mathbf{R}} = \mathbf{x}(n)\mathbf{x}^H(n)$$

$$\begin{aligned}\hat{\nabla}(n) &= 2[\mathbf{x}(n)\mathbf{x}^H(n)\mathbf{w}(n) - \mathbf{x}(n)d^*(n)] \\ &= 2[\mathbf{x}(n)(\hat{d}^*(n) - d^*(n))] \\ &= -2\mathbf{x}(n)e^*(n)\end{aligned}$$

Thus

$$\begin{aligned}\mu_0(n) &= \frac{4\mathbf{x}^H(n)e(n)\mathbf{x}(n)e^*(n)}{2\mathbf{x}^H(n)e(n)\mathbf{x}(n)\mathbf{x}^H(n)2\mathbf{x}(n)e^*(n)} \\ &= \frac{|e(n)|^2\mathbf{x}^H(n)\mathbf{x}(n)}{|e(n)|^2(\mathbf{x}^H(n)\mathbf{x}(n))^2} \\ &= \frac{1}{\mathbf{x}^H(n)\mathbf{x}(n)} = \frac{1}{||\mathbf{x}(n)||^2}\end{aligned}$$

Thus the NLMS update is

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

To avoid problems when $||\mathbf{x}(n)||^2 \approx 0$ we add an offset

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

where $a > 0$.

Consider now the convergence of the NLMS algorithm.

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

substituting $e(n) = d(n) - \mathbf{w}^H(n)\mathbf{x}(n)$

$$\mathbf{w}(n+1) \quad = \quad \mathbf{w}(n) + \frac{\tilde{\mu}}{||\mathbf{x}(n)||^2}\mathbf{x}(n)[d^*(n) - \mathbf{x}^H(n)\mathbf{w}(n)]$$

$$= \quad \left[\mathbf{I} - \tilde{\mu}\frac{\mathbf{x}(n)\mathbf{x}^H(n)}{||\mathbf{x}(n)||^2}\right]\mathbf{w}(n) + \tilde{\mu}\frac{\mathbf{x}(n)d^*(n)}{||\mathbf{x}(n)||^2}$$

Compare NLMS and LMS:

NLMS:

$$\mathbf{w}(n+1) = \left[\mathbf{I} - \tilde{\mu}\frac{\mathbf{x}(n)\mathbf{x}^H(n)}{||\mathbf{x}(n)||^2}\right]\mathbf{w}(n) + \tilde{\mu}\frac{\mathbf{x}(n)d^*(n)}{||\mathbf{x}(n)||^2}$$

LMS:

$$\mathbf{w}(n+1) = [\mathbf{I} - \mu\mathbf{x}(n)\mathbf{x}^H(n)]\mathbf{w}(n) + \mu\mathbf{x}(n)d^*(n)$$

Comparing them we see the following corresponding terms

LMS	NLMS
μ	$\tilde{\mu}$
$\mathbf{x}(n)\mathbf{x}^H(n)$	$\frac{\mathbf{x}(n)\mathbf{x}^H(n)}{\|\mathbf{x}(n)\|^2}$
$\mathbf{x}(n)d^*(n)$	$\frac{\mathbf{x}(n)d^*(n)}{\|\mathbf{x}(n)\|^2}$

Since in the LMS case

$$0 < \mu < \frac{2}{\text{trace}[E\{\mathbf{x}(n)\mathbf{x}^H(n)\}]} = \frac{2}{\text{trace}[\mathbf{R}]}$$

guarantees stability by analogy,

the NLMS condition is

$$0 < \tilde{\mu} < \frac{2}{\text{trace}\left[E\{\frac{\mathbf{x}(n)\mathbf{x}^H(n)}{||\mathbf{x}(n)||^2}\}\right]}$$

make the following approximation

$$E\left\{\frac{\mathbf{x}(n)\mathbf{x}^H(n)}{||\mathbf{x}(n)||^2}\right\} \approx \frac{\mathbf{x}(n)\mathbf{x}^H(n)}{E\{||\mathbf{x}(n)||^2\}}$$

Then

$$\begin{aligned}
\text{trace}\left[E\left\{\frac{\mathbf{x}(n)\mathbf{x}^H(n)}{||\mathbf{x}(n)||^2}\right\}\right] &= \frac{\text{trace}[E\{\mathbf{x}(n)\mathbf{x}^H(n)\}]}{||\mathbf{x}(n)||^2} \\
&= \frac{E\{\text{trace}[\mathbf{x}(n)\mathbf{x}^H(n)]\}}{||\mathbf{x}(n)||^2} \\
&= \frac{E\{\text{trace}[\mathbf{x}^H(n)\mathbf{x}(n)]\}}{||\mathbf{x}(n)||^2} \\
&= \frac{E\{\text{trace}[||\mathbf{x}(n)||^2]\}}{||\mathbf{x}(n)||^2} \\
&= 1
\end{aligned}$$

Thus the NLMS update

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

will converge if $0 < \tilde{\mu} < 2$

- The NLMS has a simpler convergence criterion than the LMS
- The NLMS generally converges faster than the LMS algorithm