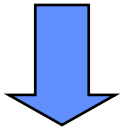
III. FIR Wiener filters - Adaptive Implementations

References:

• [Therrien]: Statistical Signal Processing, C.W. Therrien.
• [Manolakis]: Statistical and Adaptive Signal Processing, D. Manolakis, V. Ingle, S. Kogon, McGraw Hill, 2000.
• [Haykin]: Adaptive Filter Theory, S. Haykin, Prentice Hall 2002.
• [Haykin2]: Neural Networks, S. Haykin, Prentice Hall, 1999.
• [Orfanidis]: Optimum Signal Processing, J. J. Orfanidis, MacMillan, 1985.

• IIR Wiener filter requires an infinite number of weights to get lowest MSE.

• FIR Wiener filter require to solve the W-H equations and knowledge of $R_x(n)$ [$\rightarrow R_s(n)$], $R_{dx}(n)$ which may not be available.

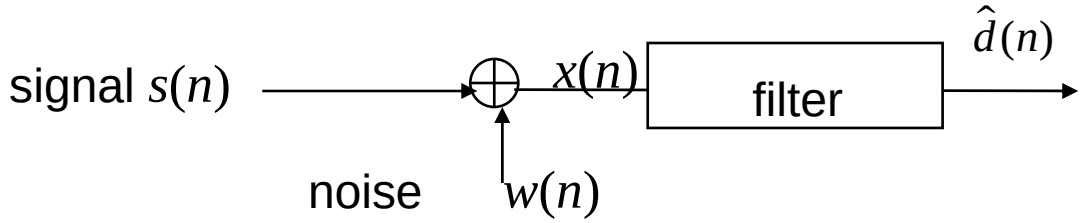


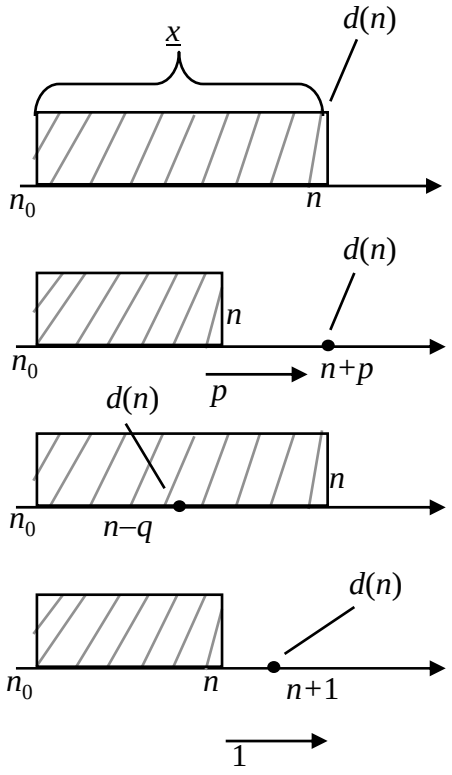
- W-H equations require to compute the inverse of a matrix.
- Solution may need to be recomputed if the signal input changes its behavior.
- Addressed via the method of *steepest descent* by solving W-H equations iteratively (leading to an adaptive scheme).
- Filter progressively learns the correlation and cross-correlation, and the filter coefficients. Goal: to converge to the optimum values given by the W-H equations.
- i.e., finds the minimum values of σ_e^2 iteratively.

Goal: Clean signal information by removing distortions from noisy signals

Examples of distortions can be found in
- air conditioning noise
- excessive reverberation and echoes
- wind and rain
- road vehicles and aircraft
- engine noise
- domestic appliances
- other conversations
- radio, TV and live music
- induced hum and/or buzz from lighting and other sources
- interference from nearby transmitters such as mobile telephones
- faulty microphones and recording equipment
- noise inherent to the recording medium

- **Recall FIR Wiener filter results**





Problem	Form of observations	Desired Signal
Filtering of noisy signal	x[n]=s[n]+w[n]	d[n]=s[n]
Prediction of signal in noise	x[n]=s[n]+w[n]	d[n]=s[n+p] p>0
Smoothing of signal in noise	x[n]=s[n]+w[n]	d[n]=s[n-q] q>0
Linear Prediction	x[n]=s[n]	d[n]=s[n+1]

- Criterion used to compute the filter coefficients

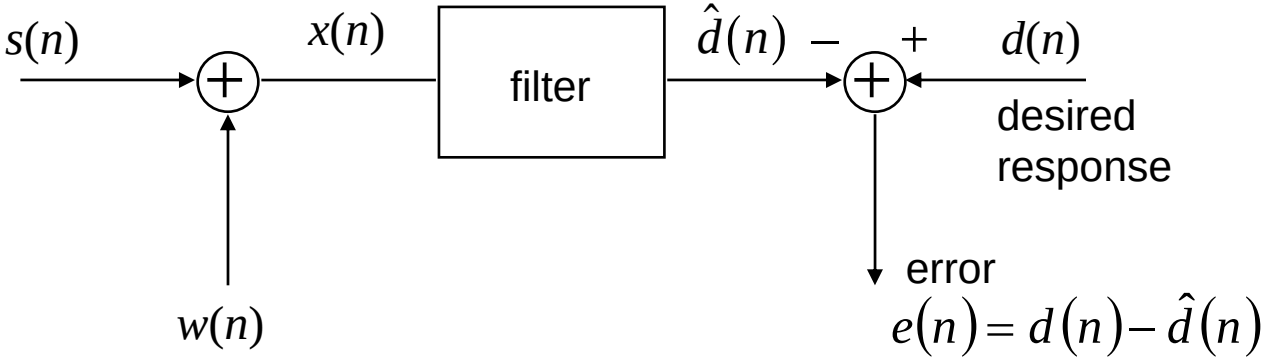
  minimization of mean square error between $d(n)$ and $\hat{d}(n)$

$$\hat{d}(n) = \sum_{k=0}^{P-1} h^*(k)x(n-k)$$

- Resulting Wiener-Hopf Equations

$$\begin{cases} R_x \underline{h} = \underline{r}_{dx}^* \Rightarrow \underline{h} = R_x^{-1} \underline{r}_{dx}^* \\ \sigma_{e_{\min}}^2 = R_d(0) - \underline{h}_{opt}^T \underline{r}_{dx} \end{cases}$$

FIR Wiener filter results



How $d(n)$ is defined specifies the operation done:

- **filtering: *d(n)=s(n)***
- **predicting: *d(n)=s(n+p)***
- **smoothing: *d(n)=s(n-p)***

- **Steepest Descent Iteration**

  Define an iterative scheme which take steps leading "downhill" for the MSE function:

A simple example: 1-coefficient filter (real filter)

- Assume we want to find *h* so that

$$\hat{d}(n) = hx(n)$$

- Define error σ_e^2 as:

$$\xi(h) = \sigma_e^2(h) = E\left[\left|d(n) - \hat{d}(n)\right|^2\right]$$

$$= E\left[\left|d(n) - hx(n)\right|^2\right]$$

$$= R_d(0) - 2hR_{dx}(0) + h^2 R_x(0)$$

$$\xi(h) = R_d(0) - 2hR_{dx}(0) + h^2 R_x(0)$$

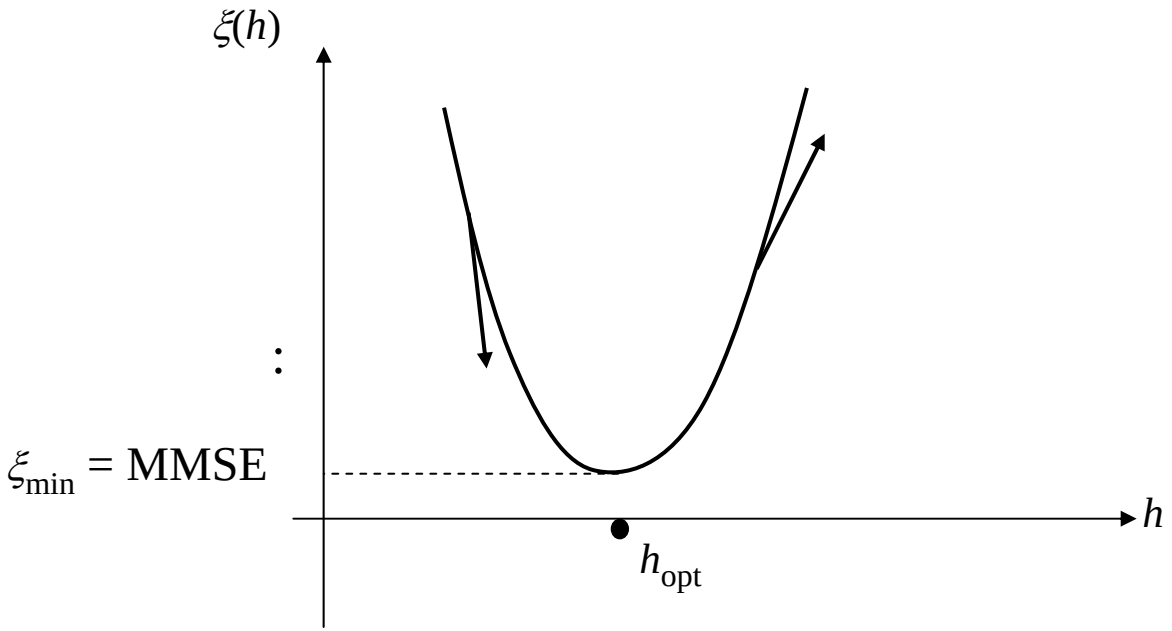
ξ is minimum when:

- Adaptive implementation is based on <u>solving</u> <u>iteratively</u> the W-H equation.

- Define $\boldsymbol{h\,(n+1) = h\,(n) + \Delta h\,(n)}$ (1)

  where $\Delta h\,(n)$ is chosen so that $\lim_{n \to +\infty} h(n) = h_{\text{opt}}$

- How to choose $\Delta h\,(n)$?

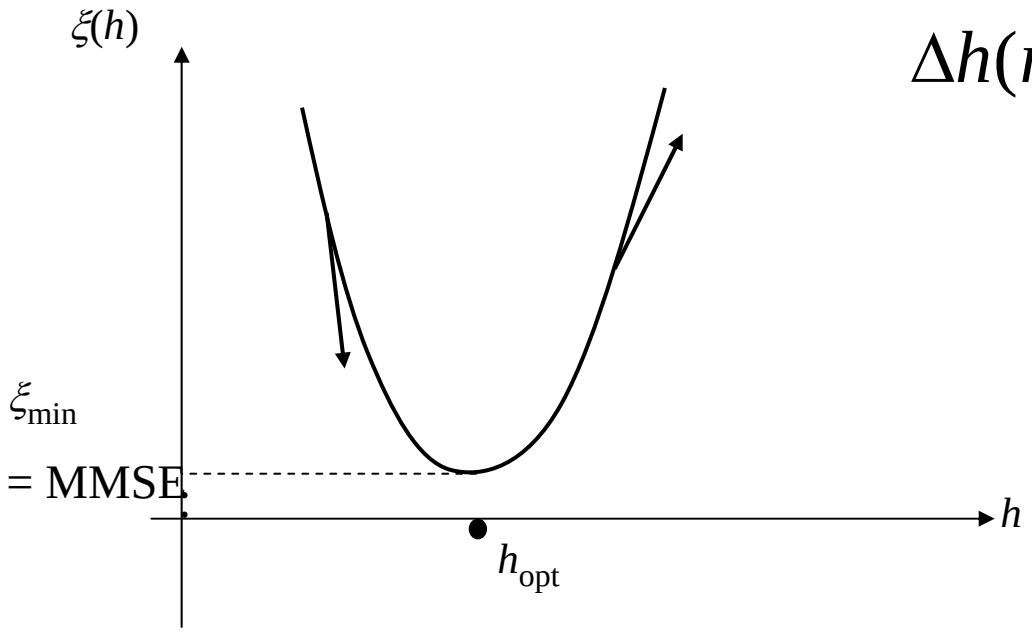


- How to choose $\Delta h\,(n)$? ➔ Use steepest descent (SD) scheme so that

$$\xi(h(n+1)) \le \xi(h(n))$$

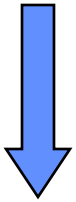
$$h(n+1) = h(n) + \Delta h(n)$$

$$\Delta h(n) = -\gamma \frac{\partial \xi(h)}{\partial h} =$$



$$h(n+1) = h(n) + \Delta h(n)$$

$$\Delta h(n) = -\gamma \frac{\partial \xi(h)}{\partial h} =$$



$$h(n+1) = h(n) - \gamma \left[\frac{\partial \xi(h)}{\partial h} \right]$$

$$=$$

- **Difference equation has a general solution:**

$$h(n) = h_{\text{opt}} + \left[1 - 2\gamma R_x(0)\right]^n \left(h(0) - h_{\text{opt}}\right)$$

where h_{opt} is the optimum value obtained with W-H eqs.

- **Algorithm converges to optimum value if:**

$$\lim_{n \to +\infty} h(n) = h_{\text{opt}} \Leftrightarrow$$

Conclusion: SD algorithm converges to optimum filter coefficient h_{opt} if we define a time-varying filter coefficient so that:

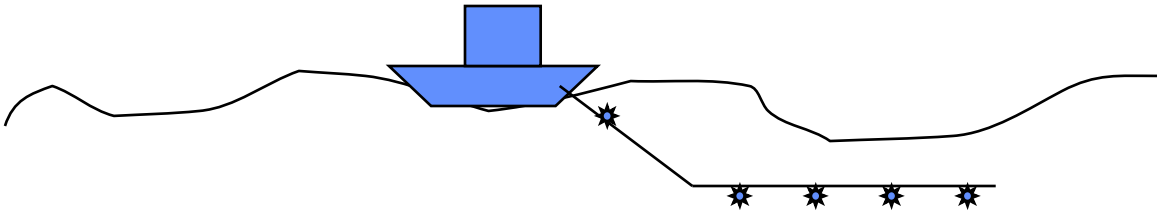
$$h(n+1) = h(n) - \gamma \bar{V}_n \left[\xi(h) \right]$$

$$= h(n)\left[1 - 2\gamma R_x(0)\right] + 2\gamma r_{dx}$$

- **And**

$$0 < \gamma < 1 / R_x(0)$$

- **Example:** Ocean-ship noise cancellation in a towed array system. (**Noise canceller [ANC]**)



- Assume noisy measurement at the array hydrophones is:

$$y(n) = s(n) + w(n)$$

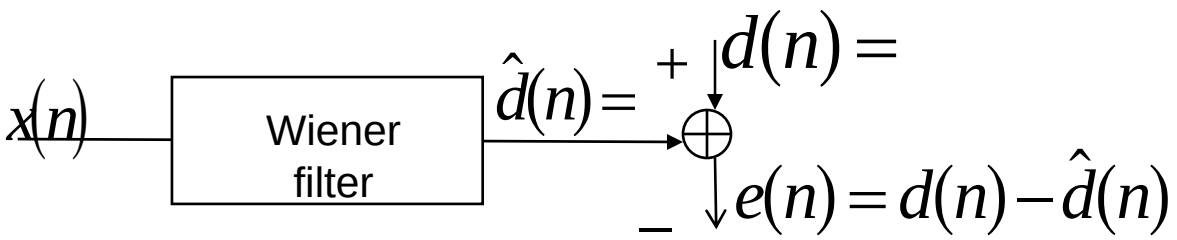
where $s(n)$: wideband (desired) signal, $w(n)$: engine noise (periodic)

$$w(n) = \sin\left(\frac{2\pi n}{16}\right)$$

- Assume signal received at the reference hydrophone is correlated with $w(n)$.

$$w'(n) = 1.25\,w(n)$$

• **Problem set-up:** Find the Wiener filter coefficient h which leads to the best approximation of *s*(*n*) using *w'*(*n*) and *y(n).*



- **For complex filter implementations:**

$$\sigma_e^2 = \xi(h) = E\left\{\left|d(n) - h^* x(n)\right|^2\right\}$$

$$= E\left\{\left(d(n) - h^* x(n)\right)\left(d(n) - h^* x(n)\right)^H\right\}$$

$$= R_d(0) + h^H E\left\{x(n)x(n)^H\right\}h - h^H R_{dx}^*(0) - \left(h^H R_{dx}^*(0)\right)^*$$

Different ways to define the gradient operation exist when dealing with complex data (see [Therrien, Appendix A]

Gradient expressions

$$h^H Rh \rightarrow 2Rh$$

$$h^H b \rightarrow 2b$$

$$b^H h \rightarrow 0$$

- **For complex filter implementations:**

$$\xi(h) = R_d(0) + h^H E\left\{x(n)x(n)^H\right\}h$$

$$-h^H R_{dx}^*(0) - \left(h^H R_{dx}^*(0)\right)^*$$

$$\nabla\xi(h) = 2(R_x(0)h - R_{dx}^*(0))$$

$$\Rightarrow h(n+1) = h(n) - \gamma\nabla\xi(h)$$

$$\Rightarrow h(n+1) = h(n) - 2\gamma(R_x(0)h - R_{dx}^*(0))$$

$$\boxed{h(n+1) = [1 - 2\gamma R_x(0)]h(n) + 2\gamma R_{dx}^*(0)}$$

- **Generalization to a P-coefficient filter (for complex data)**

exactly the same!

- W-H eqs give

$$\underline{h}_{\text{opt}} = R_x^{-1}\,\underline{r}_{dx}^{*} \quad \text{with} \quad \underline{r}_{dx} = \begin{bmatrix} R_{dx}(0) \\ \\ R_{dx}(P-1) \end{bmatrix}$$

- Adaptive implementation:

$$\underline{h}(n+1) = \underline{h}(n) - \gamma \bar{V}_h \left[\xi(\underline{h}) \right]$$

with $$\xi(\underline{h}) = E\left(\left| d(n) - \underline{h}^H(n)\,\underline{x}(n) \right|^2 \right)$$

$$\underline{h}(n) = \left[h_0(n), \cdots, h_{P-1}(n) \right]^T$$

$$\underline{x}(n) = \left[x(n), x(n-1), \cdots, x(n-P+1) \right]^T$$

1-coefficient case:

$$\nabla \xi(h) = 2(R_x(0)h - R_{dx}^*(0))$$

$$\Rightarrow h(n+1) = [1 - 2\gamma R_x(0)]h(n) + 2\gamma R_{dx}^*(0)$$

P-coefficient case:

$$\underline{h}(n+1) = \underline{h}(n) - \gamma \bar{V}_h \left[\xi(\underline{h}) \right]$$

with $\xi(\underline{h}) = \left| d(n) - \underline{h}^H(n)\underline{x}(n)^2 \right|$

$$\underline{h}(n) = \left[h_0(n), \cdots, h_{P-1}(n) \right]^T, \underline{x}(n) = \left[x(n), \cdots, x(n-P+1) \right]^T$$

as before $\bar{V}_h \xi \left[\underline{h}(n) \right] = 2(R_x \underline{h}(n) - \underline{r}_{dx}^*)$

$$\Rightarrow \underline{h}(n+1) = \left[I - 2\gamma R_x \right] \underline{h}(n) + 2\gamma \underline{r}_{dx}^*$$

P-coefficient case:

$$\underline{h}(n+1) = \left[I - 2\gamma R_x \right] \underline{h}(n) + 2\gamma \underline{r}_{dx}^{*}$$

Difference equation has the solution:

$$\boxed{\underline{h}(n) = \underline{h}_{\text{opt}} + \left[I - 2\gamma R_x \right]^n \left(\underline{h}(0) - \underline{h}_{\text{opt}} \right)}$$

- **Any restrictions in γ?** **Yes**

  Generalize 1-coefficient case findings:

$$\underline{h}(n) = \underline{h}_{\text{opt}} + \left[I - 2\gamma R_x \right]^n \left(\underline{h}(0) - \underline{h}_{\text{opt}} \right)$$

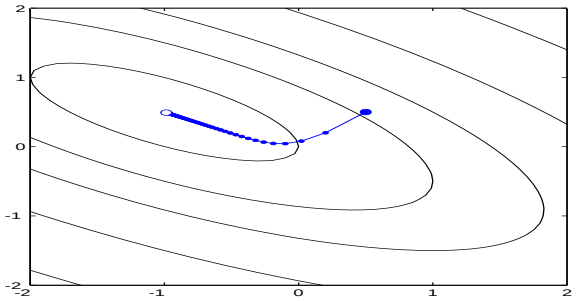
- In practice: use the fact that (for a filter of length P):

$$\lambda_{\max}\left[R_x\right] \le \text{trace}\left[R_x\right] \le \sum_{i=0}^{P-1} R_x(i,i) \le PR_x(0)$$

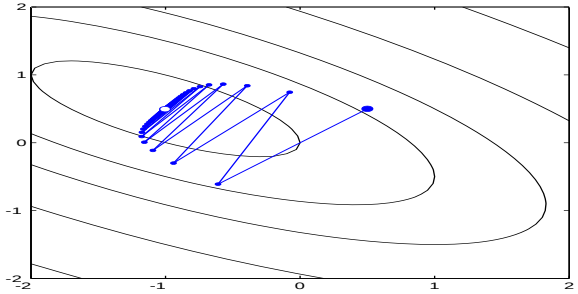
$$\boxed{0 < \gamma < 1/PR_x(0)}$$ P filter length

- **Note:** Value of γ affects the speed of convergence to $\underline{h}_{opt}$

➔ underdamped and overdamped behavior

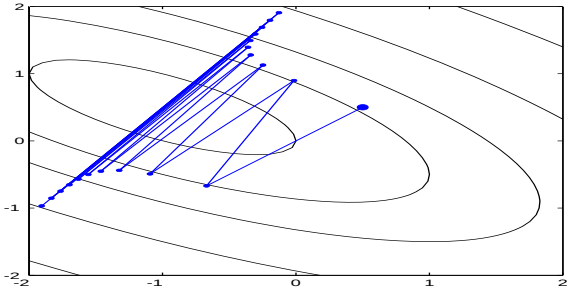


overdamped behavior



γ increases

underdamped behavior



unstable behavior

- **Overdamped/Underdamped Behavior**

$$\underline{h}(n+1) = \left(I - 2\gamma R_x\right)\underline{h}(n) + 2\gamma \underline{r}^*_{dx}$$

  - define

$$\underline{c}(n+1) = \underline{h}(n+1) - \underline{h}_{\text{opt}}$$

$$= \left[\left(I - 2\gamma R_x\right)\underline{h}(n) + 2\gamma \underline{r}^*_{dx}\right] - \left[R_x^{-1}\underline{r}^*_{dx}\right]$$

$$= \left(I - 2\gamma R_x\right)\underline{h}(n) - \left(I - 2\gamma R_x\right)R_x^{-1}\underline{r}^*_{dx}$$

$$= \left(I - 2\gamma R_x\right)\left(\underline{h}(n) - \underbrace{R_x^{-1}\underline{r}^*_{dx}}_{-\underline{h}_{\text{opt}}}\right)$$

$$\underline{c}(n+1) = \left(I - 2\gamma R_x\right)\underline{c}(n)$$

$$= U\left(I - 2\gamma\Sigma\right)U^H\underline{c}(n); \quad R_x = U\Sigma U^H$$

$$U^H\underline{c}(n+1) = \underbrace{U^H U}_{I}\left(I - 2\gamma\Sigma\right)U^H\underline{c}(n)$$

- **Overdamped/Underdamped Behavior**

$$U^H \underline{c}(n+1) = \underbrace{U^H U}_{I} \left(I - 2\gamma\Sigma \right) U^H \underline{c}(n)$$

$$\underline{V}(n+1) = \left(I - 2\gamma\Sigma \right) \cdot \underline{V}(n)$$

$$\underline{V}(n+1) = \left(I - 2\gamma\Sigma \right)^n \underline{V}_0$$

$$\underline{V}(n+1) = \left(I - 2\gamma\Sigma\right) \cdot \underline{V}(n)$$

$$\underline{V}(n+1) = \left(I - 2\gamma\Sigma\right)^{n} \underline{V}_0$$

$$\Rightarrow \underline{h}(n) = \underline{c}(n) + \underline{h}_{\text{opt}}$$

$$= U\underline{V}(n) + \underline{h}_{\text{opt}} = \sum \underline{q}_{\ell} V_{\ell}\left(n\right) + \underline{h}_{\text{opt}}$$

$$= \sum \underline{u}_{\ell} \underbrace{\left(1 - 2\gamma\Lambda_{\ell}\right)^{n}}_{\text{change of sign if } \left(1 - 2\gamma\Lambda_{\ell}\right) < 0} V_{\ell}\left(0\right) + \underline{h}_{\text{opt}}$$

To insure overdamped behavior

Select $\gamma < 1/2\,\Lambda_{\ell}$ for all *l* so that $\left(1 - 2\gamma\Lambda_{\ell}\right)^{n}$ doesn't flip sign depending if "*n*" is even or odd.

$$\boxed{\gamma < 1/2\,\Lambda\,(R_x)_{\max}}$$

- **Effects of varying the step size on the iterative scheme behavior (underdamped / overdamped)**

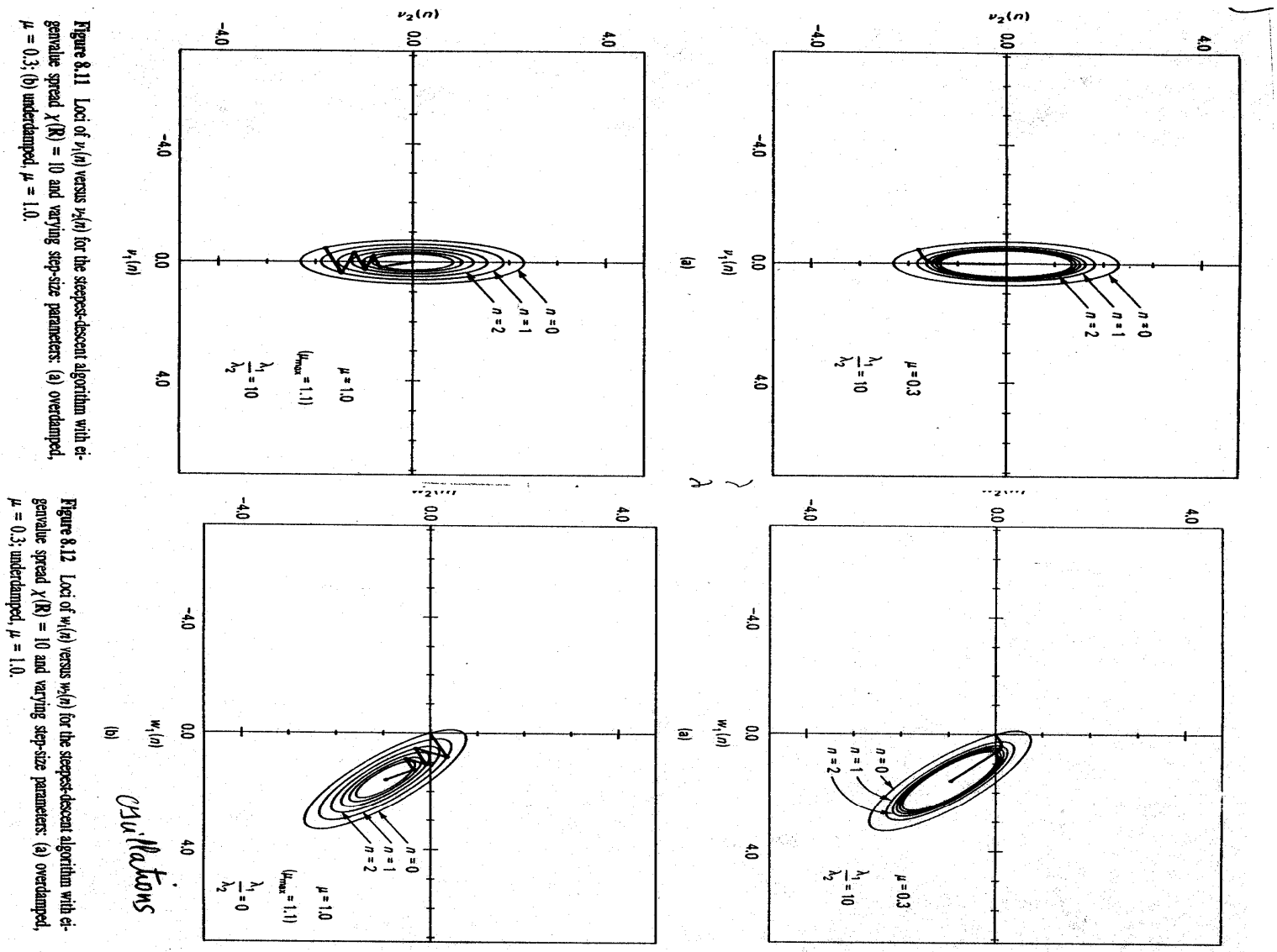


Figure 8.11 Loci of $v_1(n)$ versus $v_2(n)$ for the steepest-descent algorithm with eigenvalue spread $\chi(\mathbf{R}) = 10$ and varying step-size parameters: (a) overdamped, $\mu = 0.3$; (b) underdamped, $\mu = 1.0$.

Figure 8.12 Loci of $w_1(n)$ versus $w_2(n)$ for the steepest-descent algorithm with eigenvalue spread $\chi(\mathbf{R}) = 10$ and varying step-size parameters: (a) overdamped, $\mu = 0.3$; underdamped, $\mu = 1.0$.

Ref: [Haykin]

- **LMS derivation**
  - **Note:** Above SD derivation still needs

$$R_{dx}(0) = r$$

$$R_x(0)$$

  - **Solution:** Simplify $\bar{V}_n[\xi(h)]$

$$\bar{V}_n[\xi(h)] = \frac{\partial}{\partial h}\left[E\left[\left|d(n) - \hat{d}(n)\right|^2\right]\right]$$

$$= \frac{\partial}{\partial h}\left[E\left[\left|d(n) - h^* x(n)\right|^2\right]\right]$$

$$= \frac{\partial}{\partial h}\Big[R_d(0) + E[h^* x(n) x^*(n) h]$$

$$- E[h^* x(n) d^*(n) - h x^*(n) d(n)]$$

$$= \frac{\partial}{\partial h}\Big[R_d(0) + h^* E[x(n) x^*(n)] h$$

$$- h^* E[x(n) d^*(n)] - h E[x^*(n) d(n)]$$

$$\overline{V}_n\left[\xi(h)\right] = \frac{\partial}{\partial h}\Big[R_d(0) + h^* E[x(n)x^*(n)]h$$

$$- h^* E[x(n)d^*(n)] - hE[x^*(n)d(n)]$$

$$= 2E[x(n)x^*(n)]h - 2E[x(n)d^*(n)]$$

$$= 2E\Big[x(n)[\underbrace{hx^*(n) - d^*(n)}_{-e^*(n)}]\Big]$$

$$\overline{V}_n\left[\xi(h)\right] = -2E\Big[x(n)e^*(n)\Big]$$

$$\simeq -2x(n)e^*(n)$$

- **Least mean Square (LMS) Iteration (1-coef case)**
  - **LMS iteration (for complex filter/data):**

$$h(n+1) = h(n) + 2\gamma e^{*}(n)x(n)$$

$$\text{where} \quad e(n) = d(n) - h^{*}x(n)$$

$$\text{with} \quad \gamma \text{ bounded}$$

  - **Basic LMS algorithm:**
    - At time $n = 0$, pick initial estimate for $h(0)$
    - Compute $\hat{d}(n) = h^{*}(n)x(n)$
    - Compute $e(n) = d(n) - \hat{d}(n)$
    - Compute $h(n+1) = h(n) + 2\gamma e^{*}(n)x(n)$

    $n = n + 1$

• **Basic LMS implementation:**

Any restrictions in γ? Yes

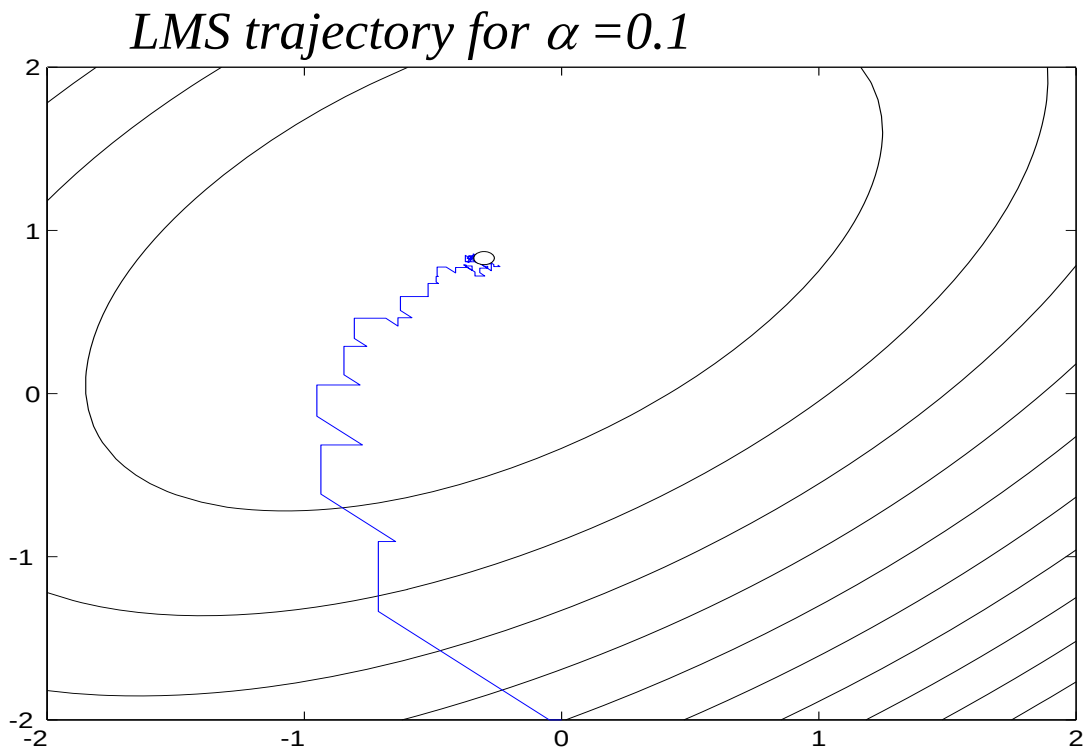
• Same initial type of restriction as for the SD exists

• Needs to have a stricter constraint because of the gradient approximation

• ➔ as a starting point, select:

$$0 < \gamma < 10\% \frac{1}{PR_x(0)}$$

- **Gradient Approximation Consequence in Convergence Behavior**



- **1-step ahead LMS predictor example**

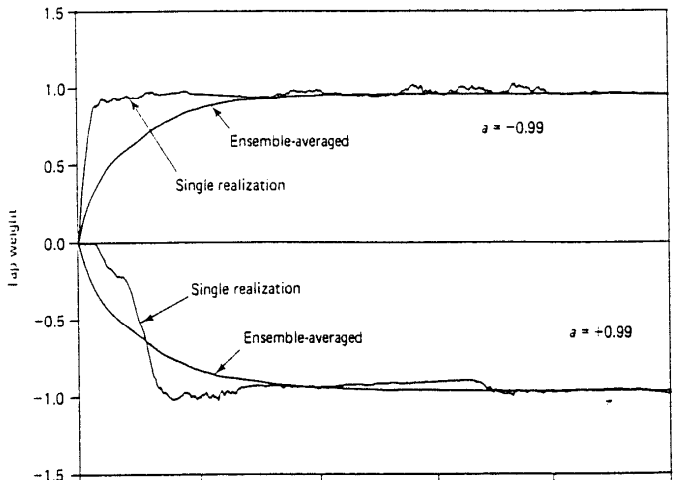
Signal is generated as an AR(1) process of the form:

$$u(n)=-au(n-1)+w(n),\ w(n)\ iid\ with\ pdf\ N(0,K^2)$$

1) Derive the 1-step ahead FIR predictor of length 1 to estimate the parameter *a*.

2) Implement the predictor using MATLAB-Simulink

3) Extend to adaptive implementation using the LMS algorithm



1. AR parameter: $a = -0.99$
   Variance of AR process $\{u(n)\}$: $\sigma_u^2 = 0.93627$
2. AR parameter: $a = +0.99$
   Variance of AR process $\{u(n)\}$: $\sigma_u^2 = 0.995$

J_{min}=0.01985

J_{min}-LMS=0.0222

Slight bias is present

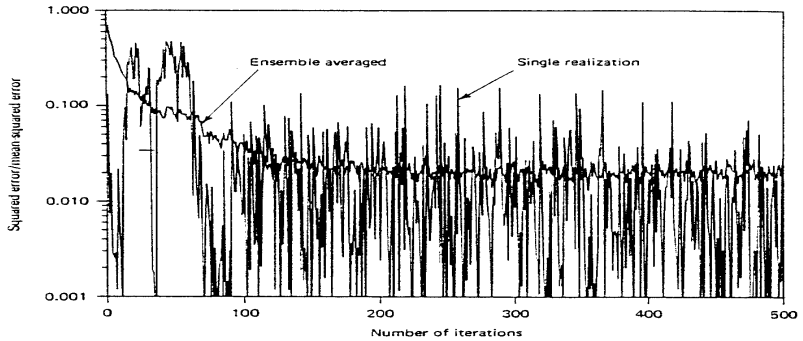


Figure 9.11 Transient behavior of squared prediction error in adaptive first-order predictor, for conditions (1) and $\mu = 0.05$.

Ref: [Haykin2]

- **LMS variants (time domain)**

1) Normalized LMS:

– handles changes in input signal power

2) Variable step size LMS:

– additional freedom in step size update

– increased misadjustment

3) Leaky LMS:

– insure weights go to 0 when input signal goes to 0

– additional bias in the solution

4) Sign LMS:

– cheaper to implement

– some performance degradations

5) Block LMS:

– reduction in computational load

– smoother behavior

6) Smoothed gradient LMS

– Smoother gradient approximation

LMS Variants (frequency domain)

7) Frequency domain LMS algorithm (FLMS):

- Improves algorithm convergence

Basic LMS implementation

$$\underline{h}(n+1) = \underline{h}(n) + 2\gamma \underline{x}(n) e^*(n), \; 0 < \gamma < 1/PR_x(0)$$

or

$$\underline{h}(n+1) = \underline{h}(n) + \gamma \underline{x}(n) e^*(n),$$

$$0 < \gamma < 2/PR_x(0)$$

- What happens when the filter input signal short time power varies?

1) Normalized LMS

$$\underline{h}(n+1) = \underline{h}(n) + \left[\qquad \right] \underline{x}(n) e^*(n)$$

Normalized LMS con't

$$\underline{h}(n+1) = \underline{h}(n) + \left[\qquad\qquad\qquad\qquad\right] \underline{x}(n) e^{*}(n)$$

Restrictions:

Convergence criterion:

2)Variable step size LMS

- Each coefficient of the update get updated separately

$$\underline{h}(n+1) = \underline{h}(n) + e^*(n) M_n \underline{x}(n)$$

$$with: \quad M_n = \begin{bmatrix} \gamma_0(n) & \cdots & 0 & 0 \\ & \gamma_1(n) & & \vdots \\ & & \ddots & \vdots \\ 0 & 0 & \cdots & \gamma_{N-1}(n) \end{bmatrix}$$

- How to update $\mu_k(n)$:
  monitor changes in $e(n)x(n-k)$
  if changes are small, increase $\gamma_k(n)$
  if changes are large, decrease $\gamma_k(n)$
- Advantages: $\gamma_k(n)$ adapted to signal variations

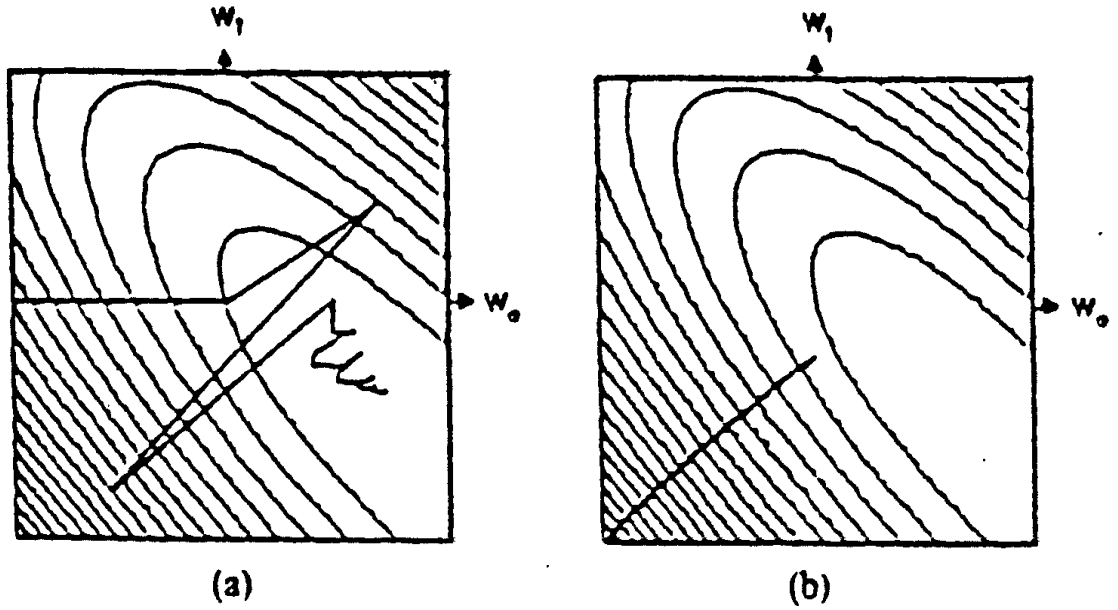
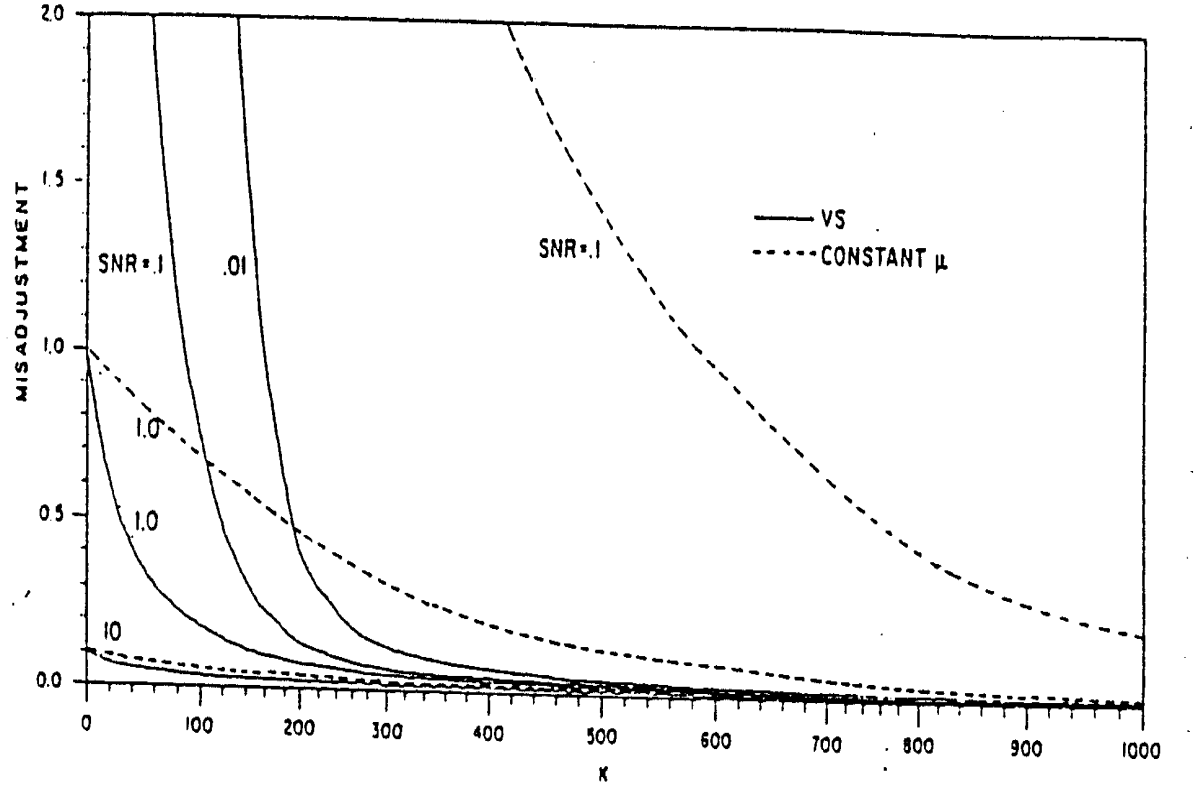


Fig. 4. Comparison of VS and LMS convergence behavior for two weights. (a) Variable step algorithm: $\mu_{max} = 0.5$, $W_{initial} = (-1, -1)$, $\mu_{min} = 0.000005$ $\alpha = 2$, $m_0 = m_1 = 3$. (b) LMS: $\mu = 0.005$, $W_{initial} = (-1, -1)$.

note $\gamma = \mu$

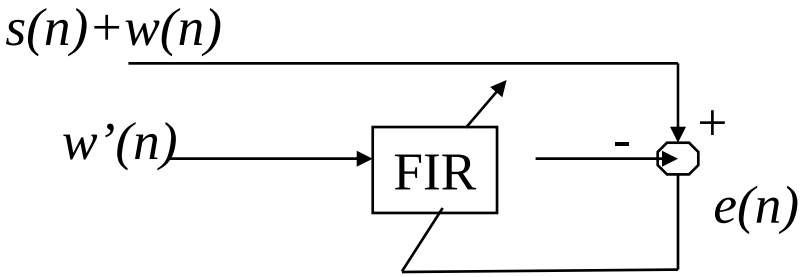
Ref: [Orfanidis], [Haykin]



Mis-adjustment comparisons of VS and constant μ algorithms for various SNR's (ANC case)

note $\gamma = \mu$

Adaptive Noise Canceler (ANC) example



Ref: [Haykin]

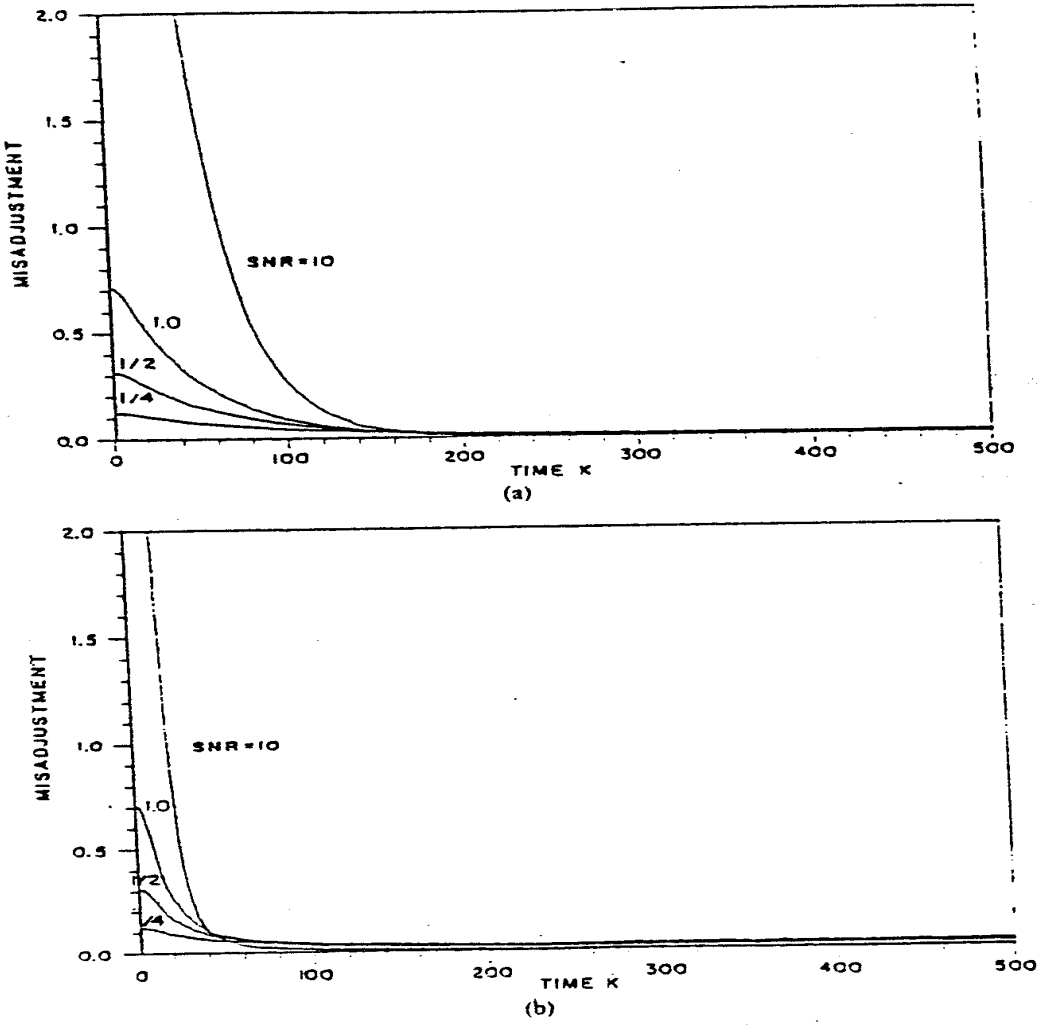


Fig. 6. FIR ALE comparisons of VS and constant μ(LMS) algorithm. (a) Misadjustment for constant μ algorithm. (b) Misadjustment for VS algorithm. $m_0 = 2$, $m_1 = 3$.

LMS

VLMS

Increased Mis-adjustment

Ref: [Haykin]

3) Leaky LMS

Basic idea: add a small amount of white noise to input to insure the weights return to zero when input is zero.

$$\underline{\tilde{h}}(n+1) = \underline{\tilde{h}}(n)(1-\mu\gamma) + \gamma e^{*}(n)\underline{x}(n)$$
$$with \quad 0 \le \mu < 1/\gamma$$

Note: addition of $(1-\mu\gamma)$ in recursion is equivalent to adding a white noise sequence to the input $\{x(n)\}$ and passing the resulting signal through the filter.

Leaky LMS con't

Justification:

$$\underline{\tilde{h}}(n+1) = \underline{\tilde{h}}(n)(1-\mu\gamma) + 2\gamma e^{*}(n)\underline{x}(n)$$

$$= \underline{\tilde{h}}(n)(1-\mu\gamma) + 2\gamma\left[d(n) - \underline{\tilde{h}}^{H}(n)\underline{x}(n)\right]^{*}\underline{x}(n)$$

$$= \underline{\tilde{h}}(n)(1-\mu\gamma) + 2\gamma\left[d^{*}(n) - \underline{x}^{H}(n)\underline{\tilde{h}}(n)\right]\underline{x}(n)$$

$$= \underline{\tilde{h}}(n)(1-\mu\gamma) + 2\gamma\left[d^{*}(n)\underline{x}(n) - \underline{x}(n)\underline{x}^{H}(n)\underline{\tilde{h}}(n)\right]$$

$$= \left[(1-\mu\gamma)I - 2\gamma\underline{x}(n)\underline{x}^{H}(n)\right]\underline{\tilde{h}}(n) + 2\gamma d^{*}(n)\underline{x}(n)$$

At convergence:

$$E\{\underline{\tilde{h}}(\infty)\} = E\{\left[(1-\mu\gamma)I - 2\gamma\,\underline{x}(\infty)\underline{x}^{H}(\infty)\right]\underline{\tilde{h}}(\infty) + 2\gamma d^{*}(\infty)\underline{x}(\infty)\}$$

$$\Rightarrow E\{\underline{\tilde{h}}(\infty)\} = \left[(1-\mu\gamma)I - 2\gamma E\{\underline{x}(\infty)\underline{x}^{H}(\infty)\}\right]\underline{\tilde{h}}(\infty) + 2\gamma E\{d^{*}(\infty)\underline{x}(\infty)\}$$

$$\Rightarrow \underline{\tilde{h}}_{opt} = \left[(1-\mu\gamma)I - 2\gamma R_x\right]\underline{\tilde{h}}_{opt} + 2\gamma \underline{r}^{*}_{dx}$$

At convergence:

$$\underline{\tilde{h}}_{opt} = \left[(1-\mu\gamma)I - 2\gamma R_x\right]\underline{\tilde{h}}_{opt} + 2\gamma \underline{r}^{*}_{dx}$$

$$\Rightarrow \left[(1-\mu\gamma)I - 2\gamma R_x - I\right]\underline{\tilde{h}}_{opt} + 2\gamma \underline{r}^{*}_{dx} = \underline{0}$$

$$-\left[2\gamma R_x + \mu\gamma I\right]\underline{\tilde{h}}_{opt} = -2\gamma \underline{r}^{*}_{dx}$$

$$\underline{\tilde{h}}_{opt} = \{2\gamma\left[R_x + \mu I/2\right]\}^{-1} 2\gamma \underline{r}^{*}_{dx}$$

$$\underline{\tilde{h}}_{opt} = \{\left[R_x + \mu I/2\right]\}^{-1} \underline{r}^{*}_{dx}$$

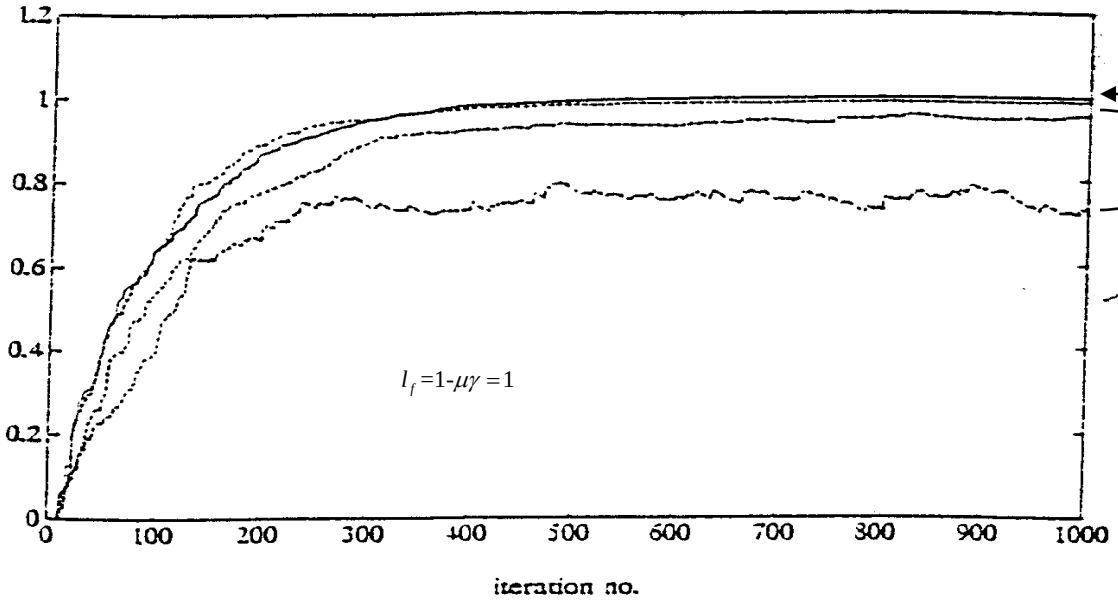
$$\underline{\tilde{h}}_{opt} = \{\left[R_x + \mu I/2\right]\}^{-1} R_x \underline{h}_{opt}$$

Bias coming from this term

Drawbacks:

- Leakage coefficient introduces a bias in the solution

  Define "leakage factor" as: $l_f = 1-\mu\gamma$



$l_f = 1-\mu\gamma = 1$

$l_f = 1-\mu\gamma < 1$

$\underline{h}_0(n)$ for leakage factor

$l_f = 1$(solid line)

$l_f = 0.9999$

$l_f = 0.9997$

$l_f = 0.9995$

Leaky LMS trial, plot

Note: Maximum step size value needs to be recomputed

Ref: [Haykin]

$$\underline{\tilde{h}}(n+1) = \underline{\tilde{h}}(n)(1-\mu\gamma) + 2\gamma e^{*}(n)\underline{x}(n)$$

$$\Rightarrow E\{\underline{\tilde{h}}(n+1)\} = \Big[(1-\mu\gamma)I - 2\gamma E\{\underline{x}^{*}(n)\underline{x}(n)\}\Big] e\{\underline{\tilde{h}}(n)\}$$

$$+ 2\gamma E\{d^{*}(n)\underline{x}(n)\}$$

$$E\{\underline{\tilde{h}}(n+1)\} = \big[(1-\mu\gamma)I - 2\gamma R_x\big] E\{\underline{\tilde{h}}(n)\}$$

$$+ 2\gamma \underline{r}^{*}_{dx}$$

Average solution of the type

$$E\{\underline{\tilde{h}}(n)\} = \big[I - \gamma(2R_x + \mu I)\big]^{n} (E\{\underline{\tilde{h}}(0)\} - \underline{\tilde{h}}_{opt}) - + \underline{\tilde{h}}_{opt}$$

Algorithm converges if:

$$\big|1-\gamma(2\lambda+\mu)\big| < 1 \Rightarrow \boxed{0 < \gamma < \frac{1}{\lambda_{\max} + \mu/2}}$$

4) Sign (Clipped) LMS algorithm

- Sometimes LMS is too expensive to compute

===> needs to decrease the computational load

$$\underline{h}(n+1) = \underline{h}(n) + \gamma \operatorname{sign}(\underline{x}(n)) e^{*}(n)$$

$$sign(y) = \begin{cases} 1 & if \quad y > 0 \\ -1 & if \quad y < 0 \\ 0 & if \quad y = 0 \end{cases}$$

- Some performance degradations (in CV rate & error) but not too much

5) Block LMS algorithm

- Designed to reduce the computational load
- weights get updated only once every N data points
- filter is usually updated using the average of the gradient estimates obtained for that data block

$$\underline{h}\left[(j+1)N\right] = \underline{h}\left[jN\right] + \frac{\gamma}{N}\sum_{i=0}^{N-1} e^{*}(jN+i)\,\underline{x}(jN+i)$$

$$\hat{d}(jN+i) = \underline{h}(jN)^{H}\,\underline{x}(jN+i)$$

$$e(jN+i) = d(jN+i) - \hat{d}(jN+i)$$

with:

$$x(jN+i) = [x(jN+i), \cdots, x(jN+i-L+1)]^{t}$$

$h(jN)$: obtained in j^{th} block of data

- Consequences:
  - on algorithm speed of convergence: slower CV speed
  - on MMSE values: MMSE decreases due to averaging operation
  - Takes fewer computations than LMS, takes advantage of parallel processing

6) Smoothed gradient LMS algorithm

- Main problem with LMS is that it uses instantaneous gradient estimates
- Possible improvements obtained by "smoothing" gradient using any LP filter

$$\underline{h}(n+1) = \underline{h}(n) + \gamma \underline{b}_n$$

where:

$$b_n(i) = LPF\left\{e^*(n)x(n-i), \cdots e^*(n)x(n-i-L+1)\right\},$$

$$i = 0, \ldots, P-1$$

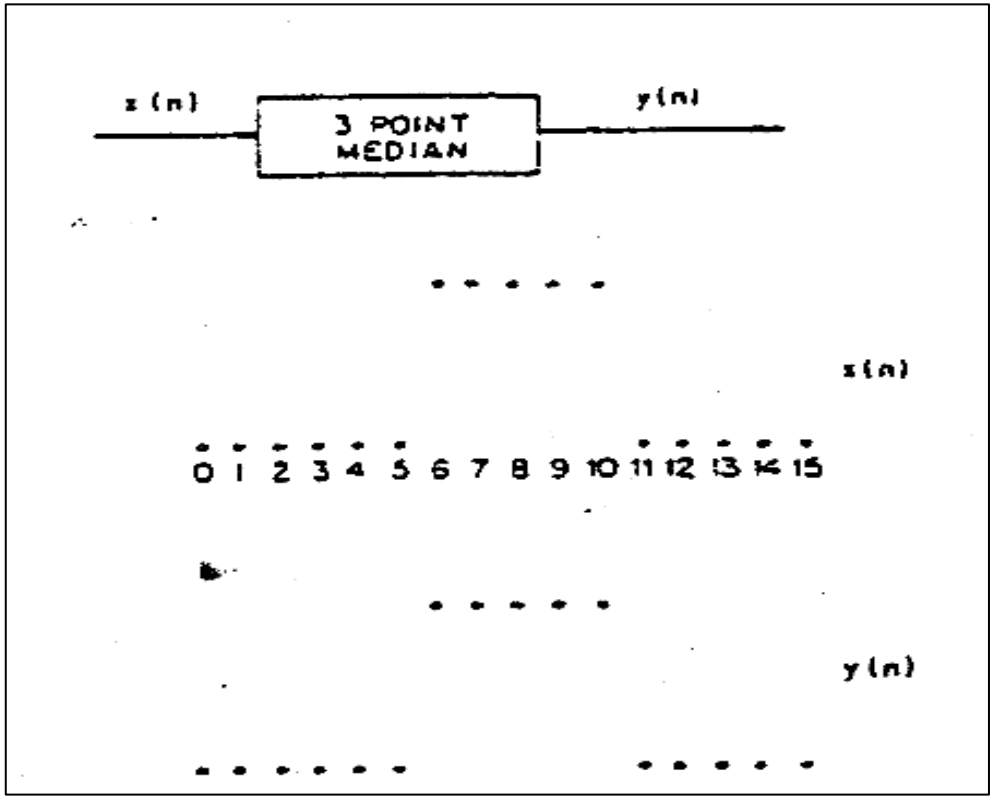
- **Advantages:** smoother convergence
- **Drawbacks:** slower convergence

  slower to react to sudden changes in input

Comments

- A nonlinear filter can be used instead of a a linear LPF to computes improved gradient estimates
- Median filtering can be used (see next pages)
- Set-up useful when impulsive interferences are present in input or desired signal, as median filter removes short duration impulses from the signal.
- CV may be slightly lower when inputs are non impulsive
- A few pathological cases where Median LMS (MLMS) becomes unstable

7) Median smoother LMS (MLMS)

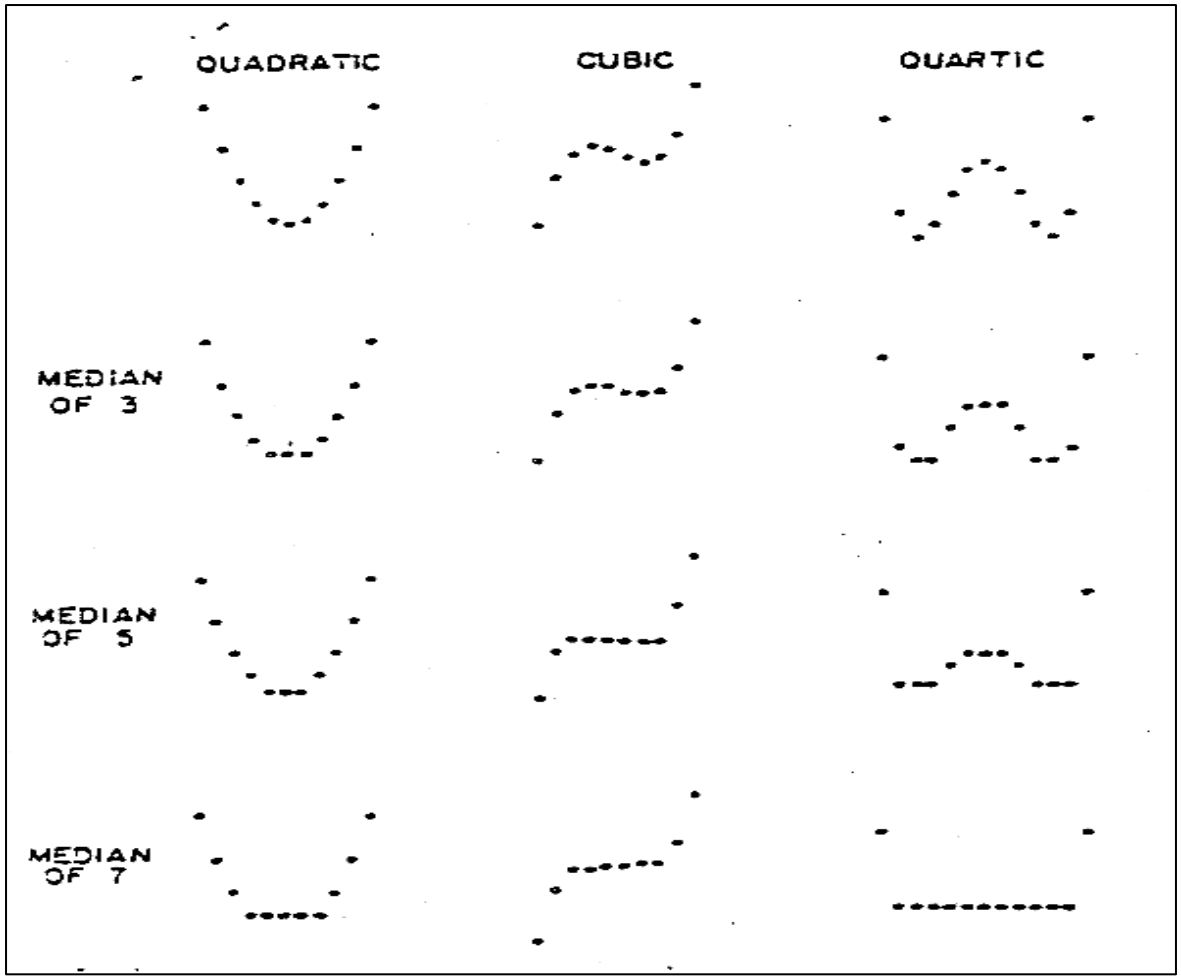


Median smoothing of a sequence with a discontinuity

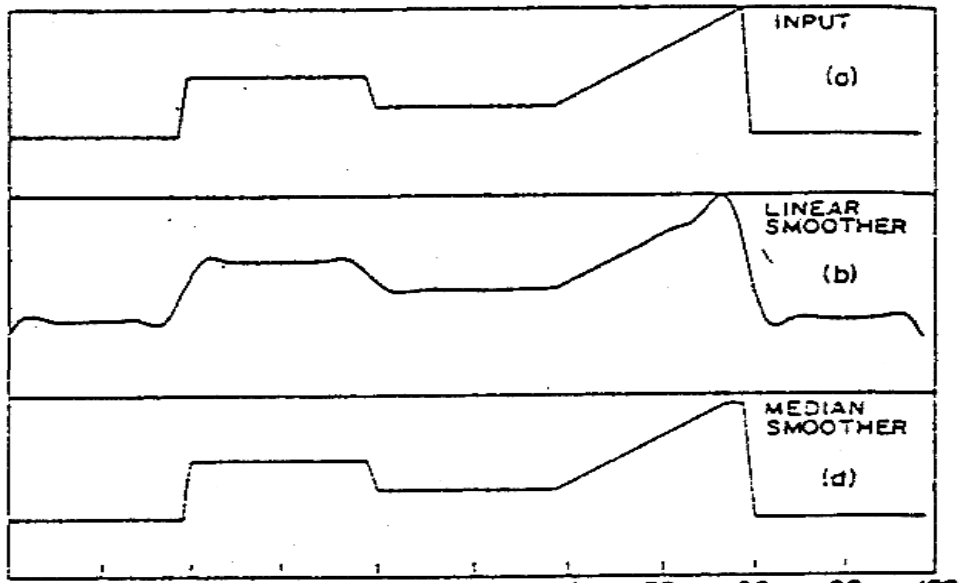
Median smoothing operation of length L, filter of length P

$$h_i(n+1) = h_i(n) + \gamma Med\{e(n)x(n), e(n)x(n-1), \ldots, e(n)x(n-L+1)\}, \quad i = 0, \ldots, P-1$$

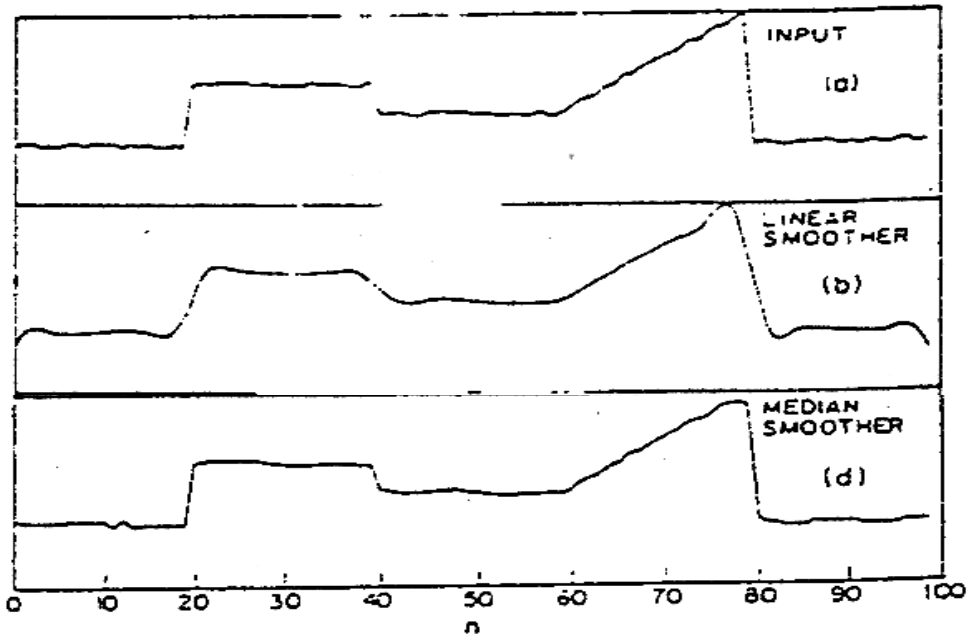
Comparisons median/linear smoothers



Effects of median smoother on low order polynomials



Median/linear smoother effects



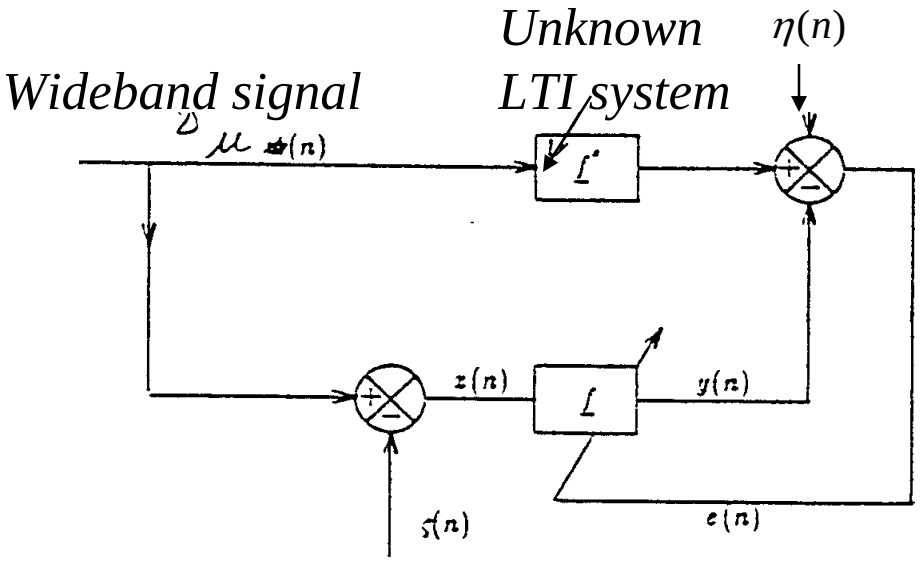
Median/linear smoother effects on noisy signals

- **Adaptive system identification application**

Data generated as:

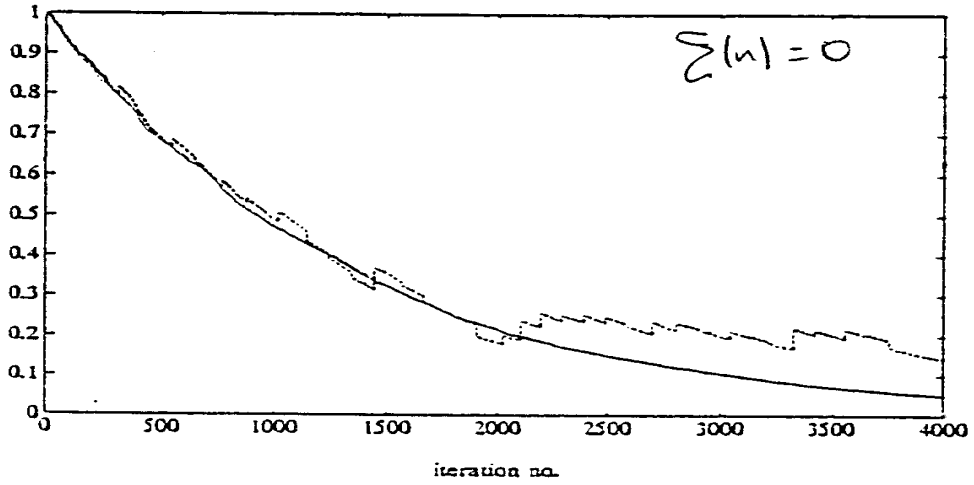
u(n) ~ N(0,K), iid; η(n), ζ(n) impulsive sequences with amplitude N(0,8) occurring at 100 samples interval

Adaptive system with additive impulsive noise occurring at input & desired signal locations



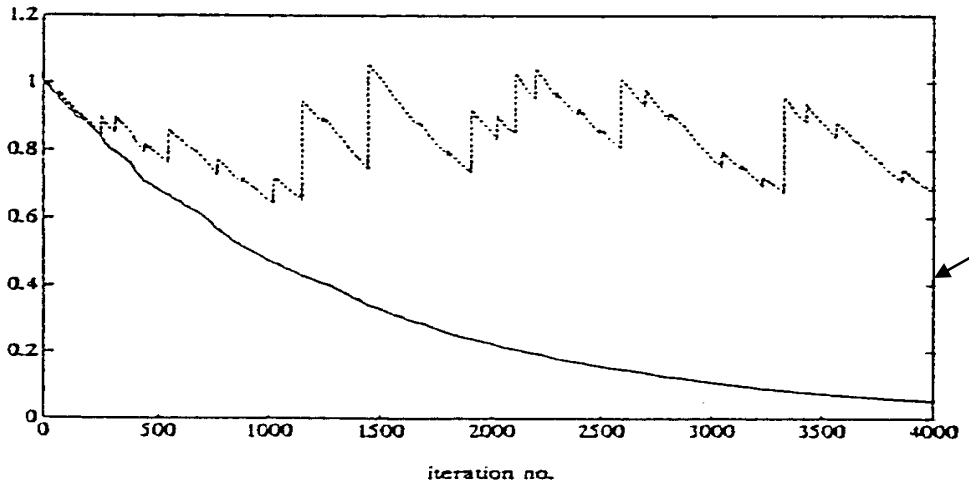
LMS filter of length P=5,
median smoother of length L=3
γ=0.01

Ref: [Haykin]



Impulsive noise present at $\eta(n)$ only, $\zeta(n)=0$.

1st filter coefficient error between true coefficient and adaptive coefficient for MLMS (solid line) and LMS (dashed line)



Impulsive noise present at $\zeta(n)$ only, $\eta(n)=0$.

Ref: [Haykin]

8) Frequency domain LMS

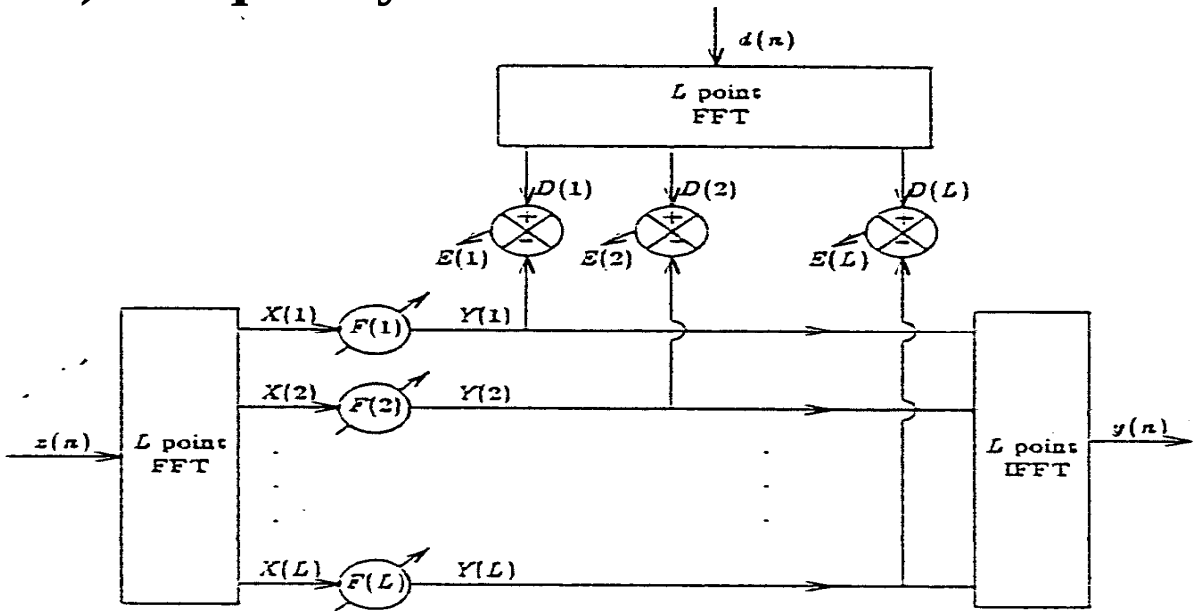


Figure 6.6.1 Frequency Domain Adaptive Filter (FDAF).

- Basic idea: used to improve the algorithm convergence by reducing the eigenvalue spread
- Note: This is a "block LMS" as incoming data is stored in blocks before FFT is computed
- DFT decouples signal into a set of frequency bins (bands). Gradients get computed over each independent band. As a result, one may select different values of the step size over different bands.
- Fast CV rate, approaches that of Recursive Least Square (RLS)

- Definition:

$$x_i(n) = x(iL+n) \quad n = 0, \cdots, L-1$$

$$d_i(n) = d(iL+n) \quad n = 0, \cdots, L-1$$

- Each block of data $\{x_i(n)\}$, $\{d_i(n)\}$, gets Fourier transformed to get the Fourier coefficients $\{X_i(k)\}$,& $\{D_i(k)\}$

- Error in each frequency bin k defined as:

$$E_i(k) = D_i(k) - Y_i(k)$$

$$= - F_i^*(k) X_i(k), k = 0, \ldots, L-1$$

- **Weight update equation**:

$$F_{i+1}(k) = F_i(k) + \gamma E_i^*(k) X_i(k),$$
$$k = 0, \ldots, L-1$$

$$\underline{F}_{i+1} = \underline{F}_i + \gamma \begin{bmatrix} X_i(0) & \cdots & 0 \\ & \ddots & \\ 0 & & X_i(L-1) \end{bmatrix} \begin{bmatrix} \underline{E}_i^* \end{bmatrix}$$

$$\underline{F}_i = \left[F_i(0), \ldots, F_i(L-1) \right]^T$$

$$\underline{E}_i = \left[E_i(0), \ldots, E_i(L-1) \right]^T$$

- FLMS algorithm convergence issues:

$$E_i^*(k)$$

$$F_{i+1}(k) = F_i(k) + \gamma \overbrace{\left[D_i(k) - F_i^*(k) X_i(k) \right]}^{*} X_i(k)$$

$$= \left[1 - \gamma |X_i(k)|^2 \right] F_i(k) + \gamma D^*_i(k) X_i(k)$$

$$\Rightarrow$$

$$\mathrm{E}[F_{i+1}(k)] = \left[1 - \gamma E[|X_i(k)|^2] \right] E[F_i(k)] +$$

$$+ \gamma E[D^*_i(k) X_i(k)]$$

- FLMS algorithm convergence issues:

$$\mathrm{E}[F_{i+1}(k)] = \left[1 - \gamma E[|X_i(k)|^2]\right] E[F_i(k)] + \gamma E[D^*_i(k) X_i(k)]$$

If $x(n)$ and $d(n)$ are stationary:

$$\mathrm{E}[|X_i(k)|^2] = R(k), \; G(k) = E[D^*_i(k) X_i(k)]$$

$$1)\ \mathrm{E}[F_\infty(k)] = \left[1 - \gamma |R(k)|^2\right] E[F_\infty(k)] + \gamma[G(k)]$$

$$\mathrm{E}[F_\infty(k)](1 + \gamma |R(k)|^2 - 1) = \gamma[G(k)]$$

$$\Rightarrow \mathrm{E}[F_\infty(k)] = \frac{G(k)}{R(k)}$$

$$2)\mathrm{E}[F_{i+1}(k)] = \left[1 - \gamma R(k)\right] E[F_i(k)] + \gamma G(k)$$

Error per band k

$$\xi_i(k) = E[F_i(k)] - E[F_\infty(k)]$$

$$= \left(1-\gamma R(k)\right) E[F_{i-1}(k)] + \gamma G(k) - \underbrace{E[F_\infty(k)]}_{G(k)/R(k)}$$

$$\frac{G(k)}{R(k)}[\gamma R(k) - 1]$$

$$= \left(1-\gamma R(k)\right)\left\{ E[F_{i-1}(k)] - \frac{G(k)}{R(k)} \right\}$$

$$\xi_i(k) = \left(1-\gamma R(k)\right)\xi_{i-1}(k)$$

Error per band k

$$\xi_i(k) = \left(1 - \gamma R(k)\right) \xi_{i-1}(k)$$

Algorithm converges if:

$$\left|1 - \gamma R(k)\right| < 1 \Rightarrow 0 < \gamma < 2 / R(k)$$

Note: CV rate is a function of the power in each band

Comments:

- Convergence rate is independent for each *k*, as it depends on *R(k)*, *k=0,…L-1*
- Convergence rate is different for each band (not good)
- How can we improve the alg. convergence?

  Normalize the algorithm convergence in each band

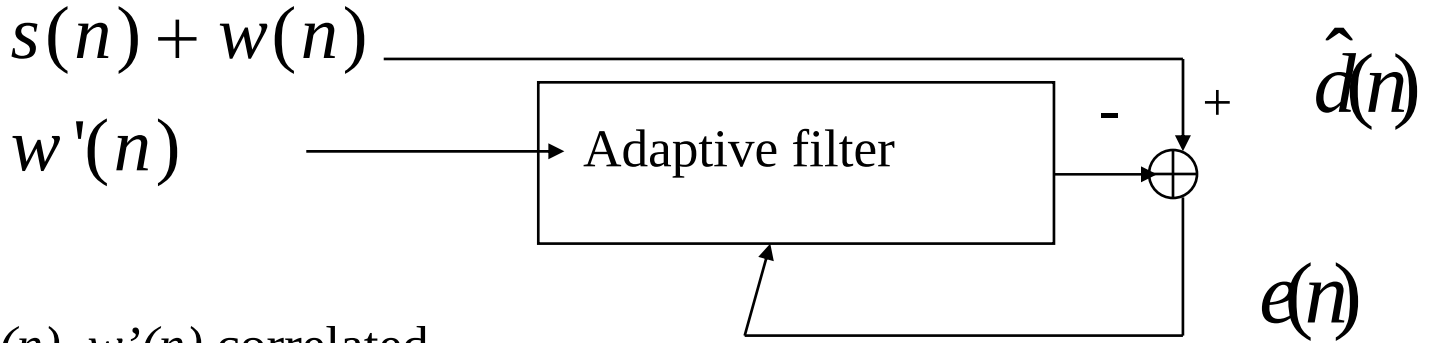
  Use a normalized algorithm

$$F_{i+1}(k) = F_i(k) + \gamma \frac{E_i^*(k) X_i(k)}{\left[\qquad\qquad\right]},$$

$$k = 0, \ldots, L-1$$

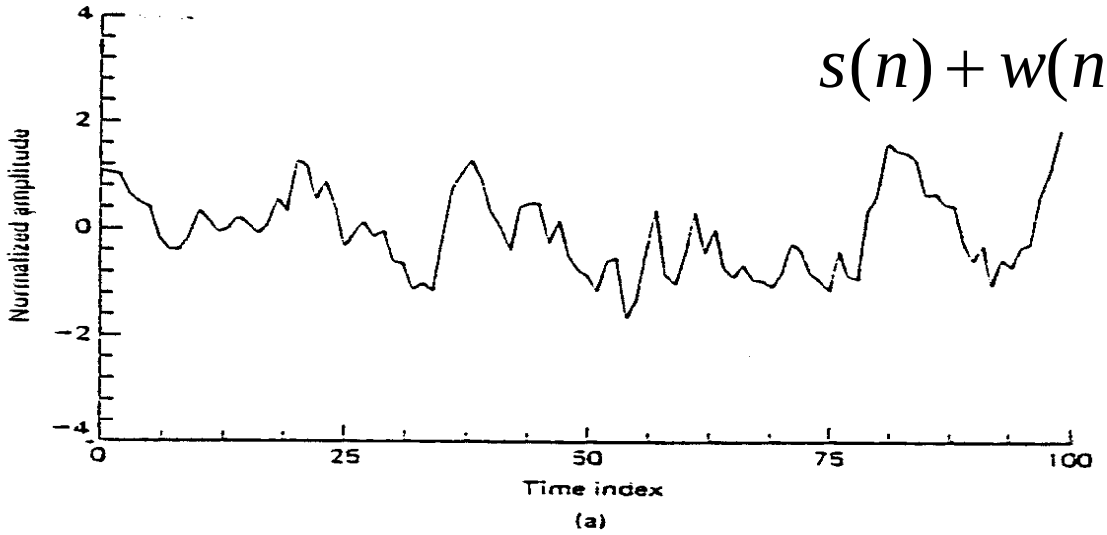
Applications of adaptive filtering

1. Adaptive noise cancellation (ANC) with external reference



w(n), w'(n) correlated

w(n), s(n) uncorrelated and zero mean



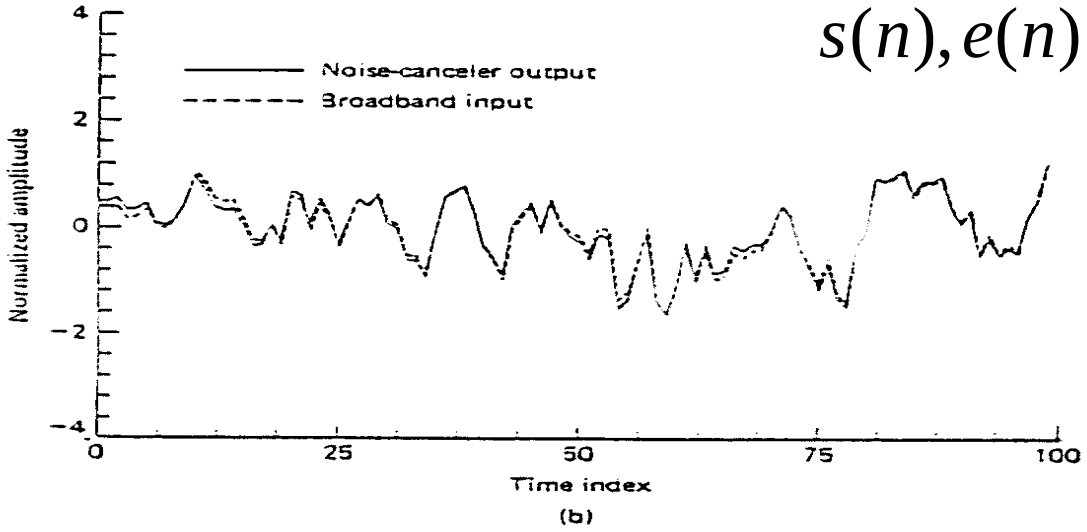
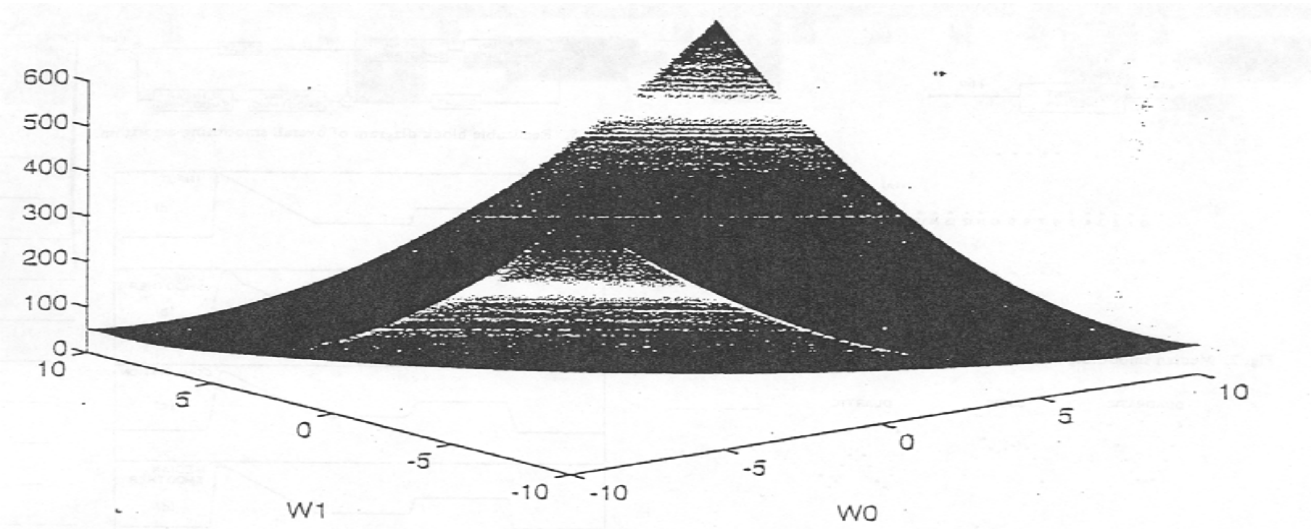
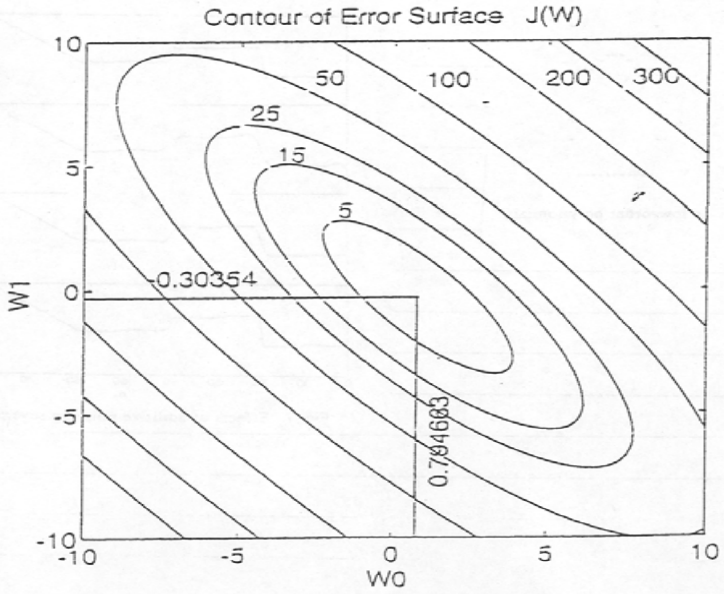
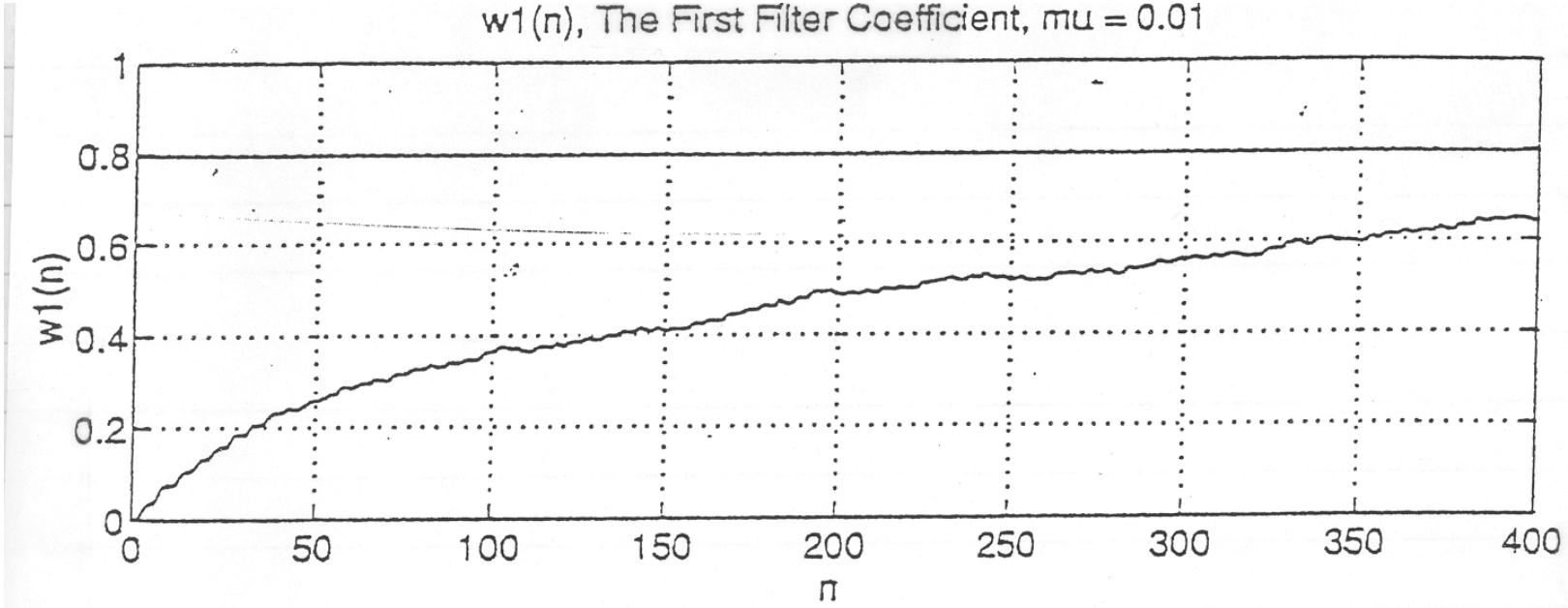


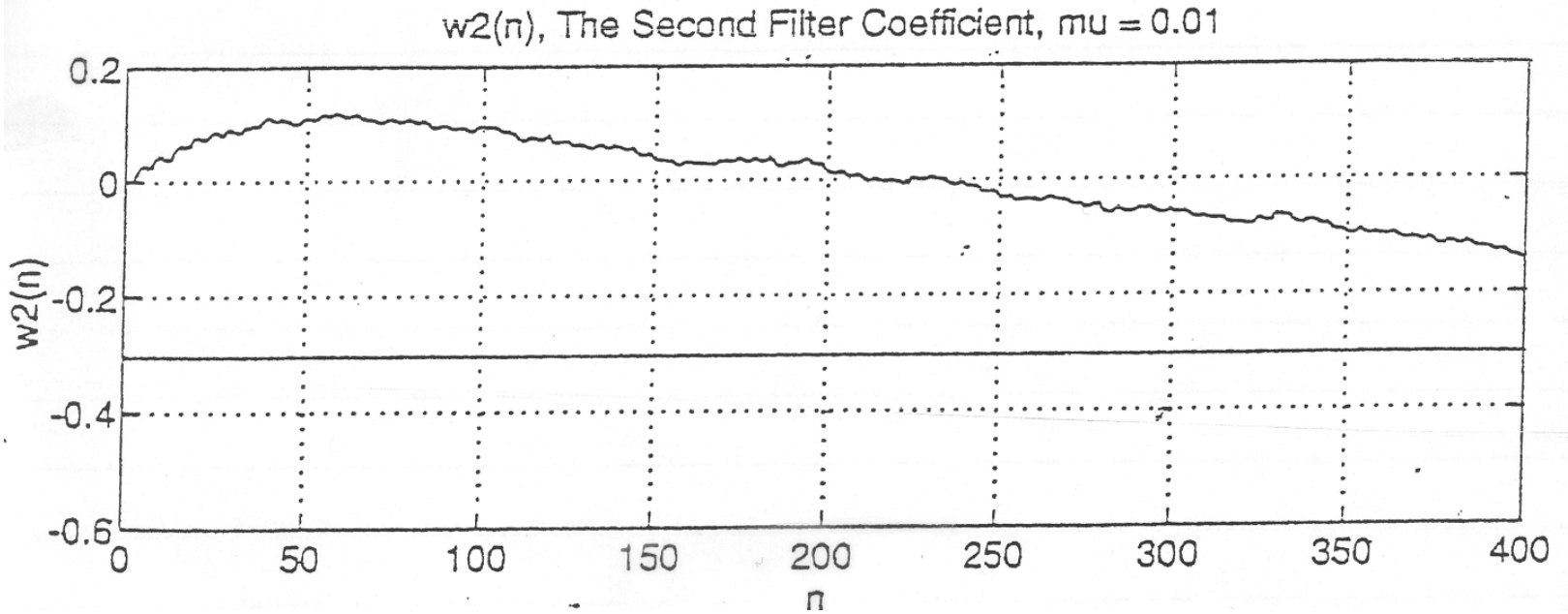
Figure 12.35 Result of periodic interference canceling experiment: (a) input signal (correlated Gaussian noise and sine wave); (b) noise-canceler output (correlated Gaussian noise). From B. Widrow et al., *Adaptive Noise Canceling: Principles and Applications*, © December 1975, IEEE.

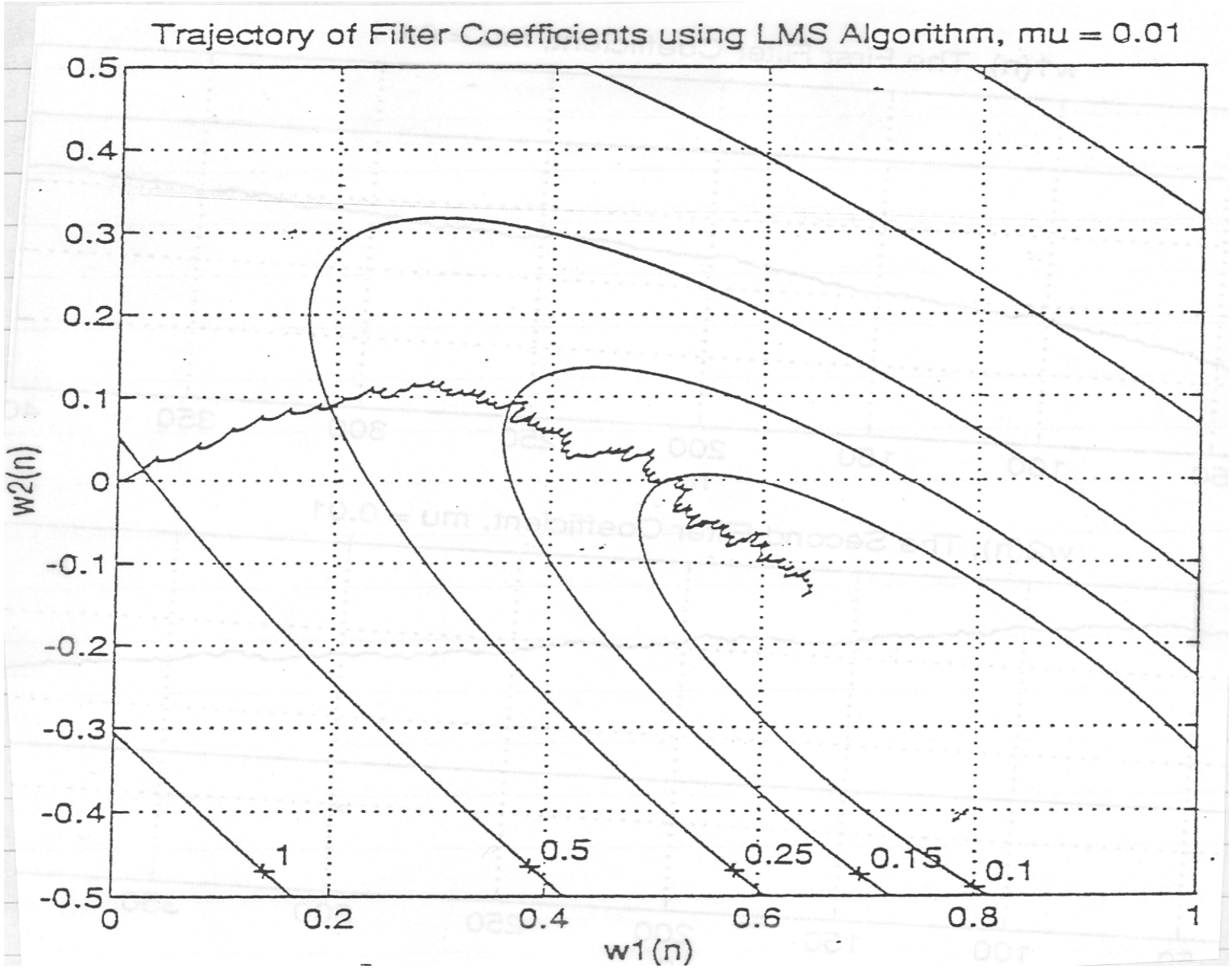
Error Surface for a FIR filter of length 2

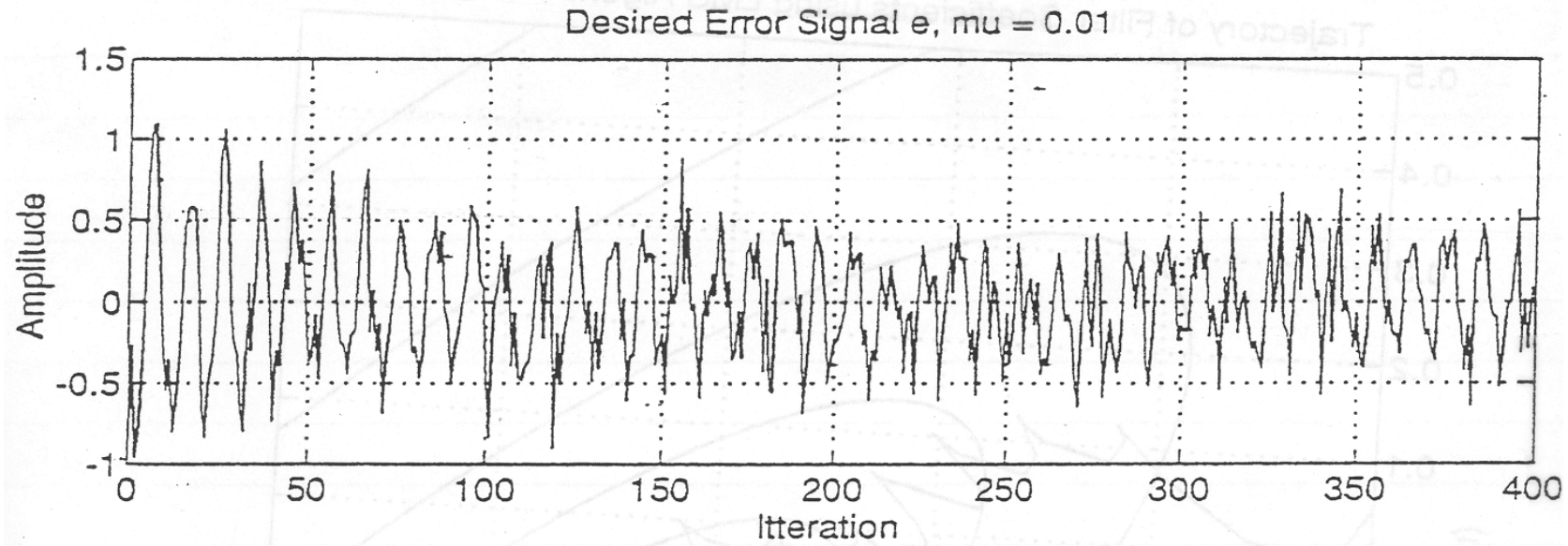


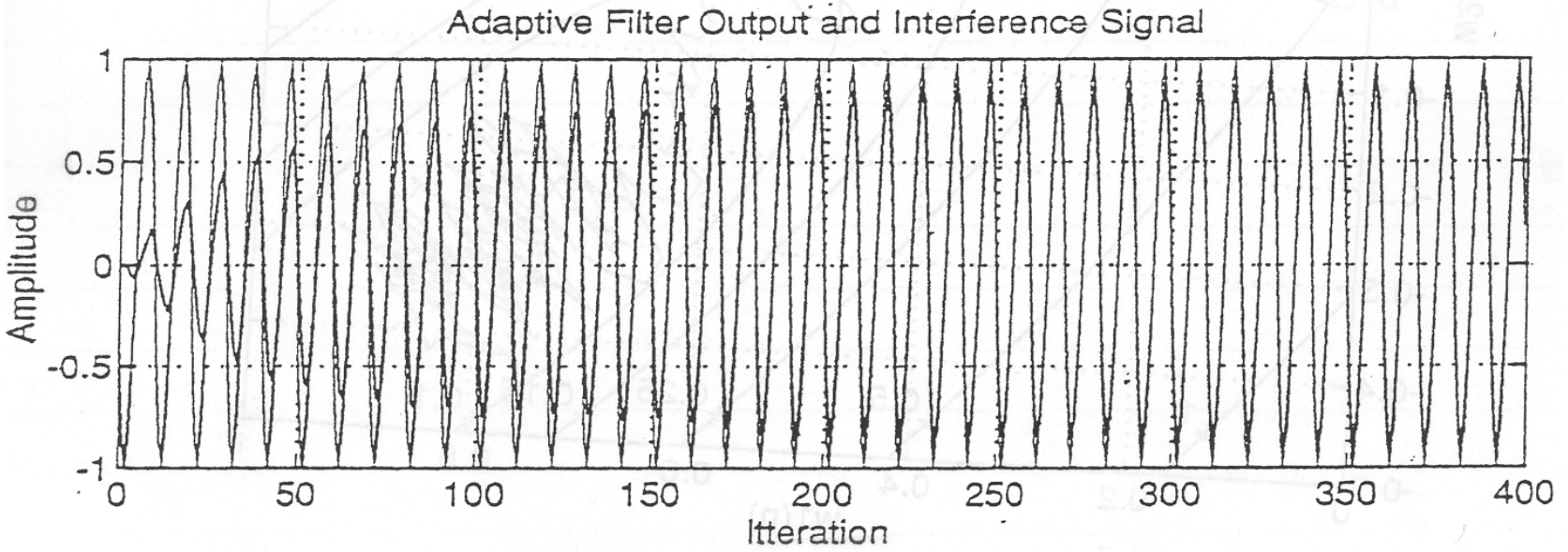


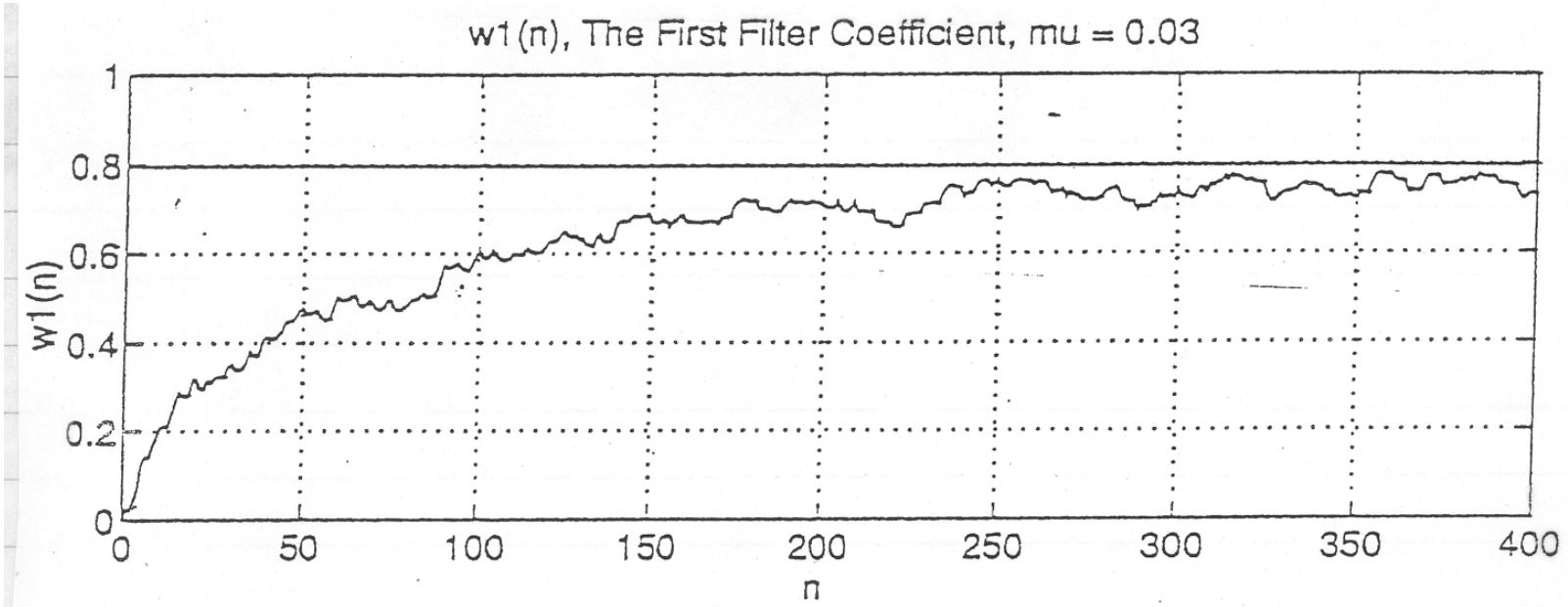

w1(n), The First Filter Coefficient, mu = 0.01
w1(n)
n
0
0.2
0.4
0.6
0.8
1
0
50
100
150
200
250
300
350
400

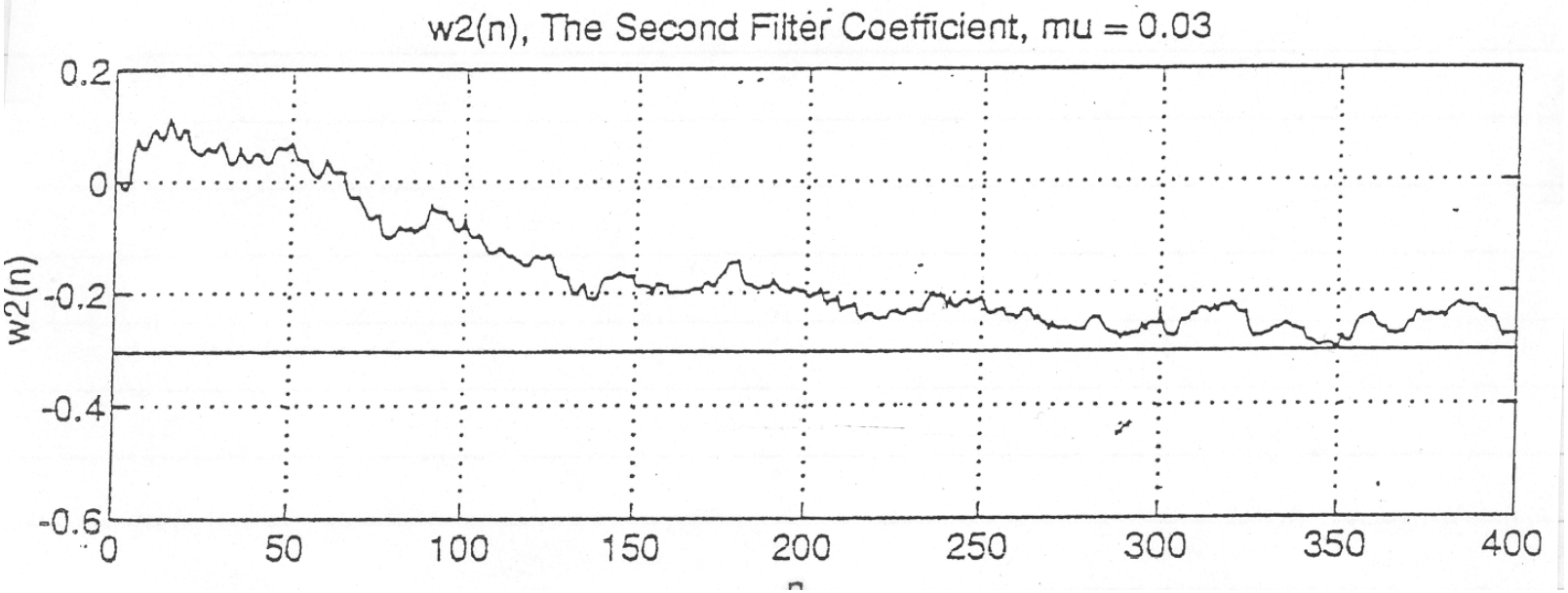

w2(n), The Second Filter Coefficient, mu = 0.01
w2(n)
n
0.2
0
-0.2
-0.4
-0.6
0
50
100
150
200
250
300
350
400

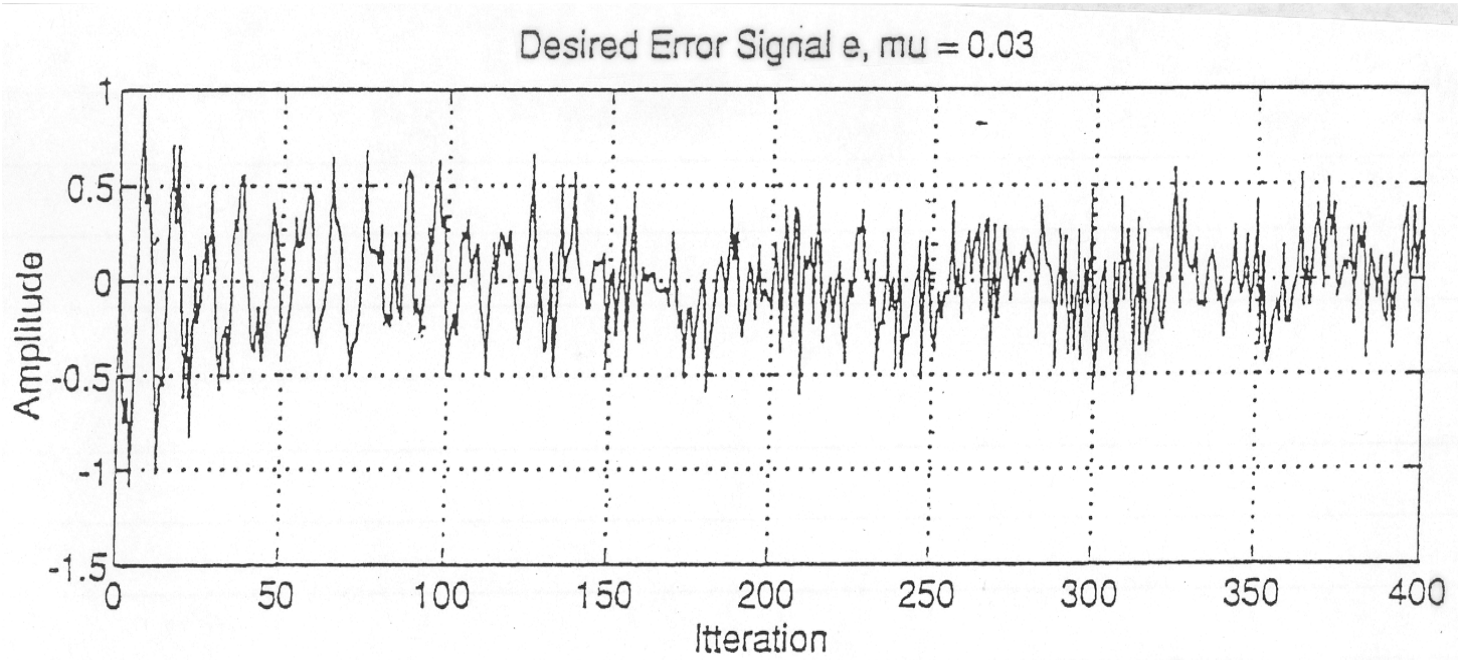

Trajectory of Filter Coefficients using LMS Algorithm, mu = 0.01
0.5
0.4
0.3
0.2
0.1
0
-0.1
-0.2
-0.3
-0.4
-0.5
w2(n)
0
0.2
0.4
0.6
0.8
1
w1(n)
1
0.5
0.25
0.15
0.1

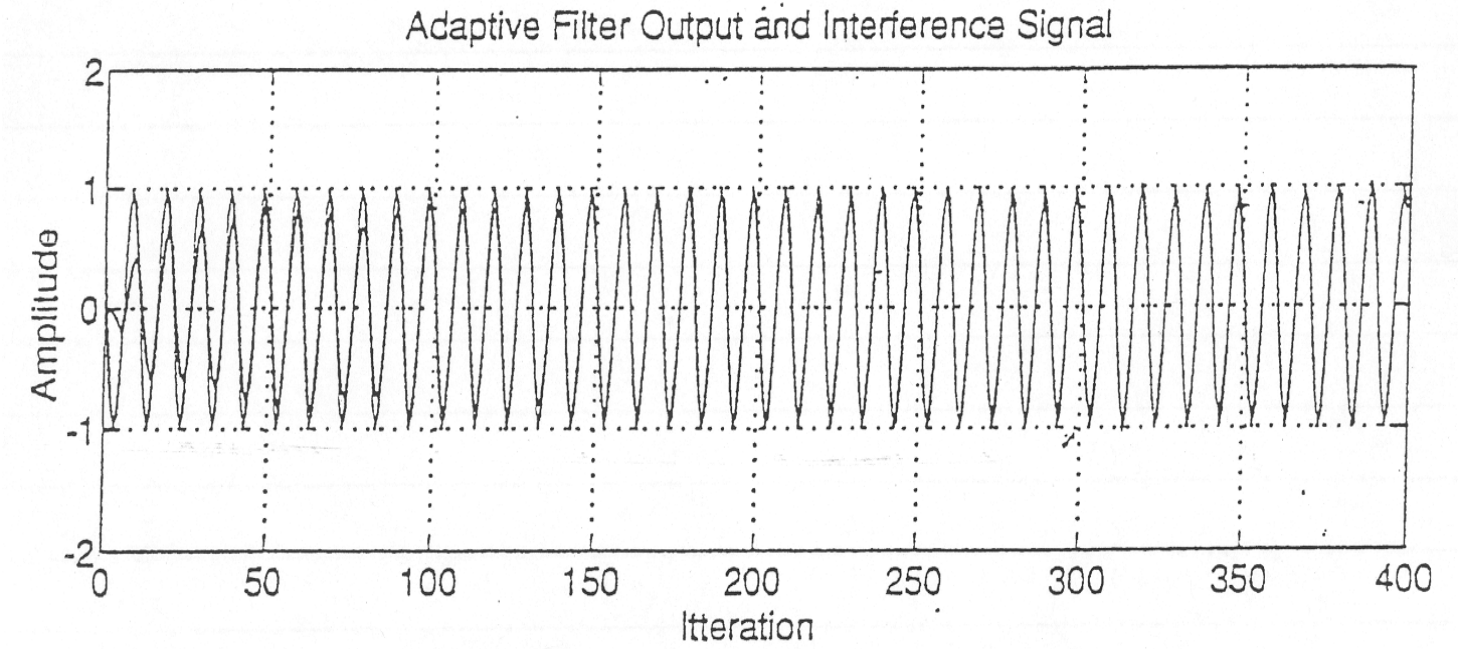

Desired Error Signal e, mu = 0.01
Amplitude
1.5
1
0.5
0
-0.5
-1
0
50
100
150
200
250
300
350
400
Itteration

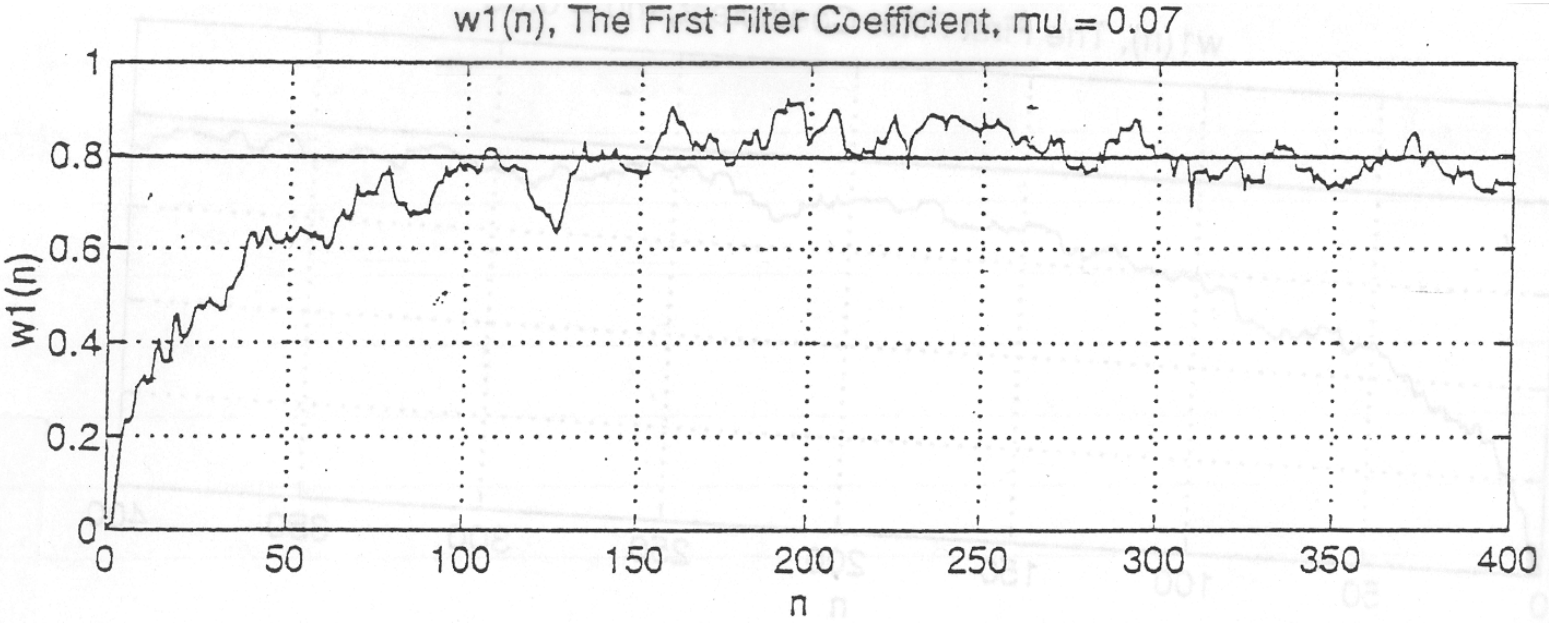

Adaptive Filter Output and Interference Signal
Amplitude
1
0.5
0
-0.5
-1
0
50
100
150
200
250
300
350
400
Itteration

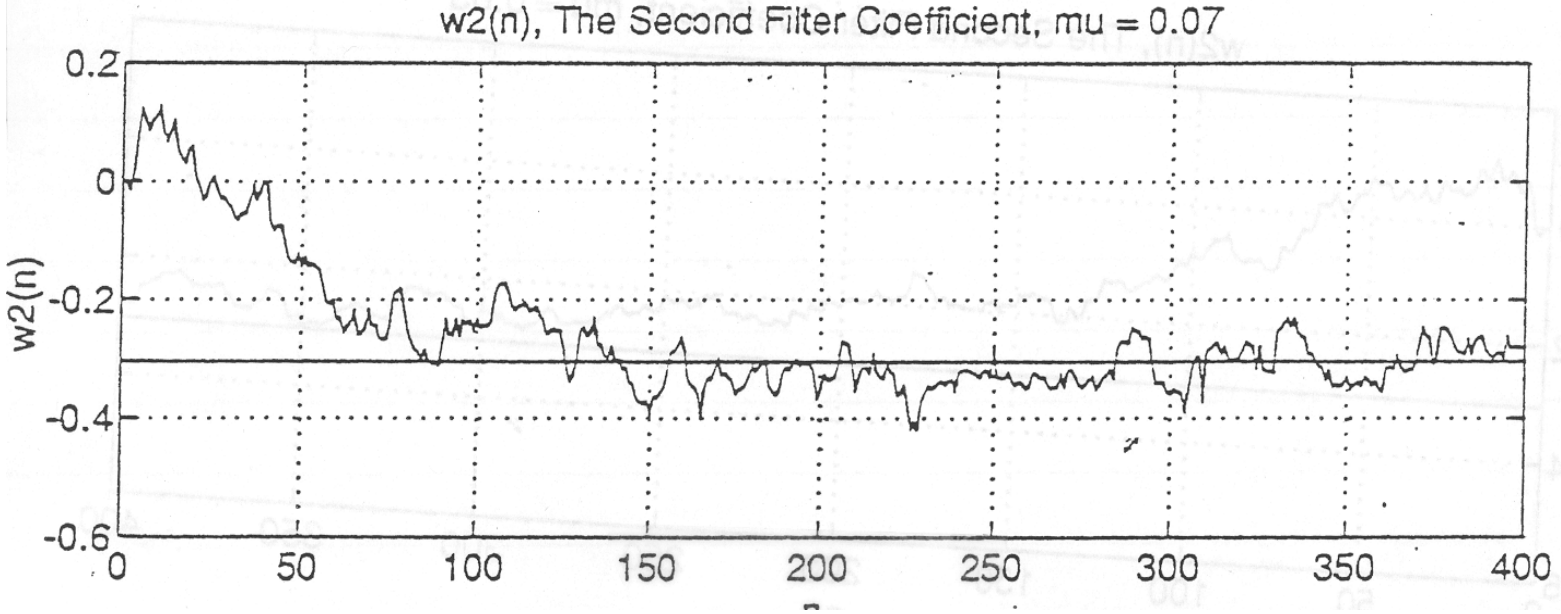

w1(n), The First Filter Coefficient, mu = 0.03
w1(n)
n
0
0.2
0.4
0.6
0.8
1
0
50
100
150
200
250
300
350
400

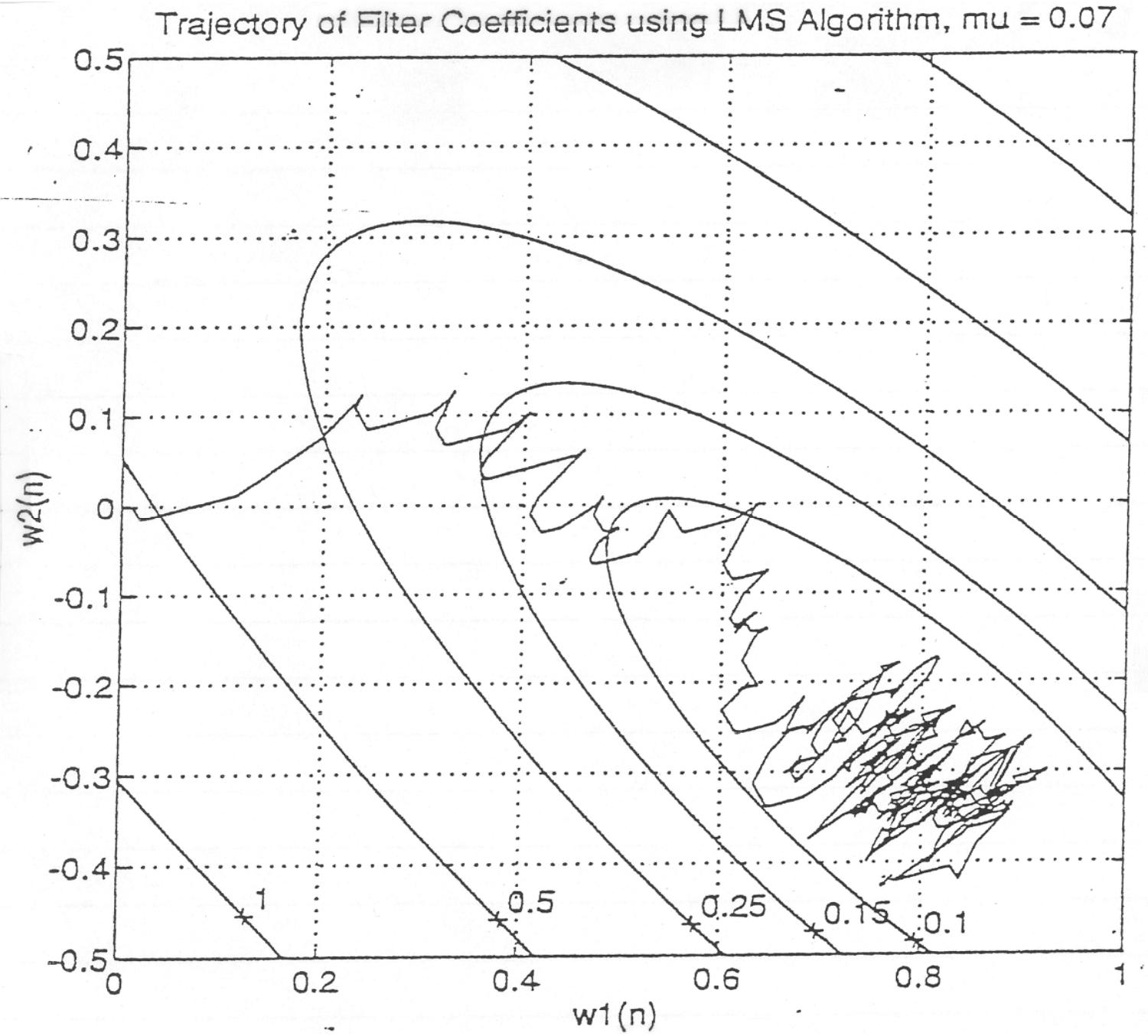

w2(n), The Second Filter Coefficient, mu = 0.03
w2(n)
n
0.2
0
-0.2
-0.4
-0.6
0
50
100
150
200
250
300
350
400

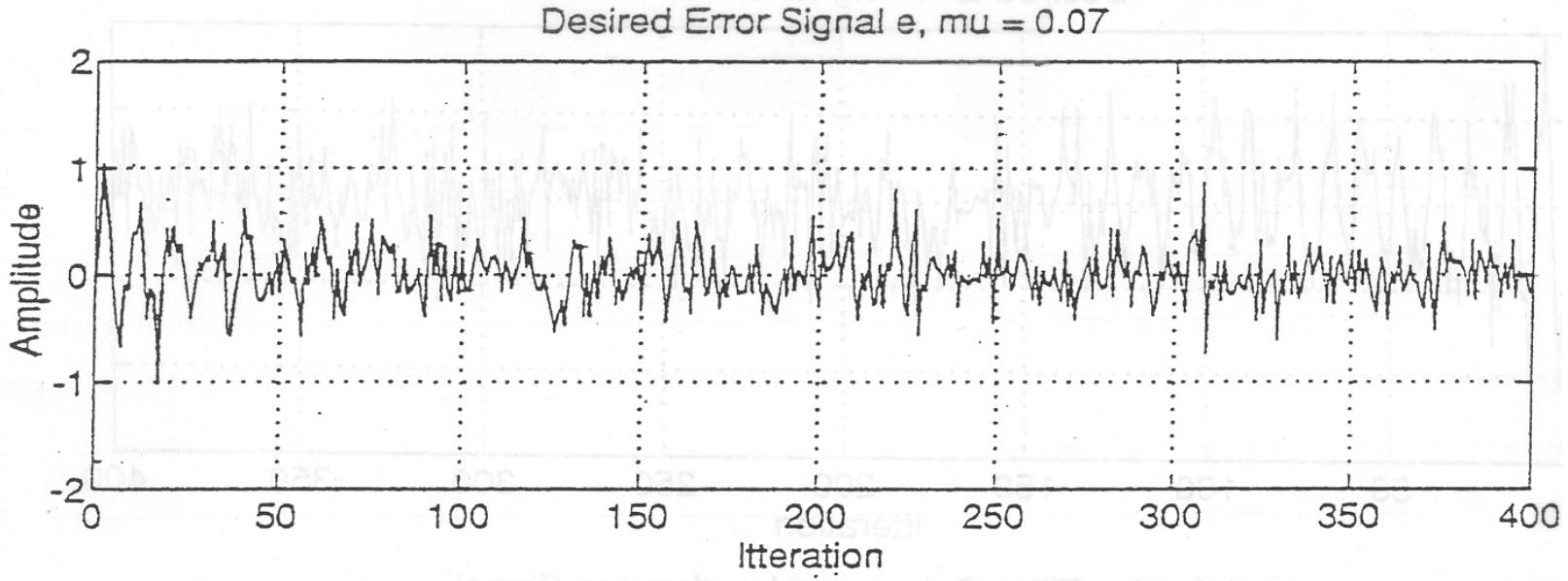

Desired Error Signal e, mu = 0.03
Amplitude
Itteration
1
0.5
0
-0.5
-1
-1.5
0
50
100
150
200
250
300
350
400

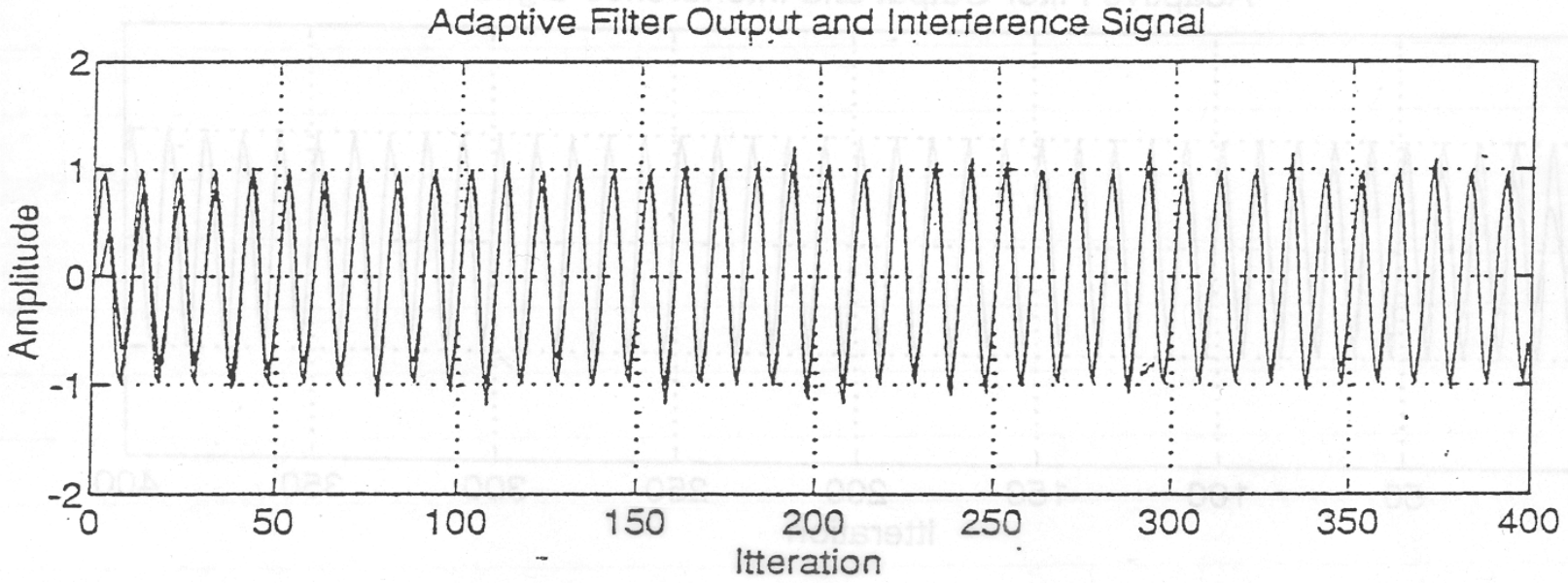

Adaptive Filter Output and Interference Signal
Amplitude
Itteration
2
1
0
-1
-2
0
50
100
150
200
250
300
350
400


w1(n), The First Filter Coefficient, mu = 0.07
w1(n)
n
1
0.8
0.6
0.4
0.2
0
0
50
100
150
200
250
300
350
400

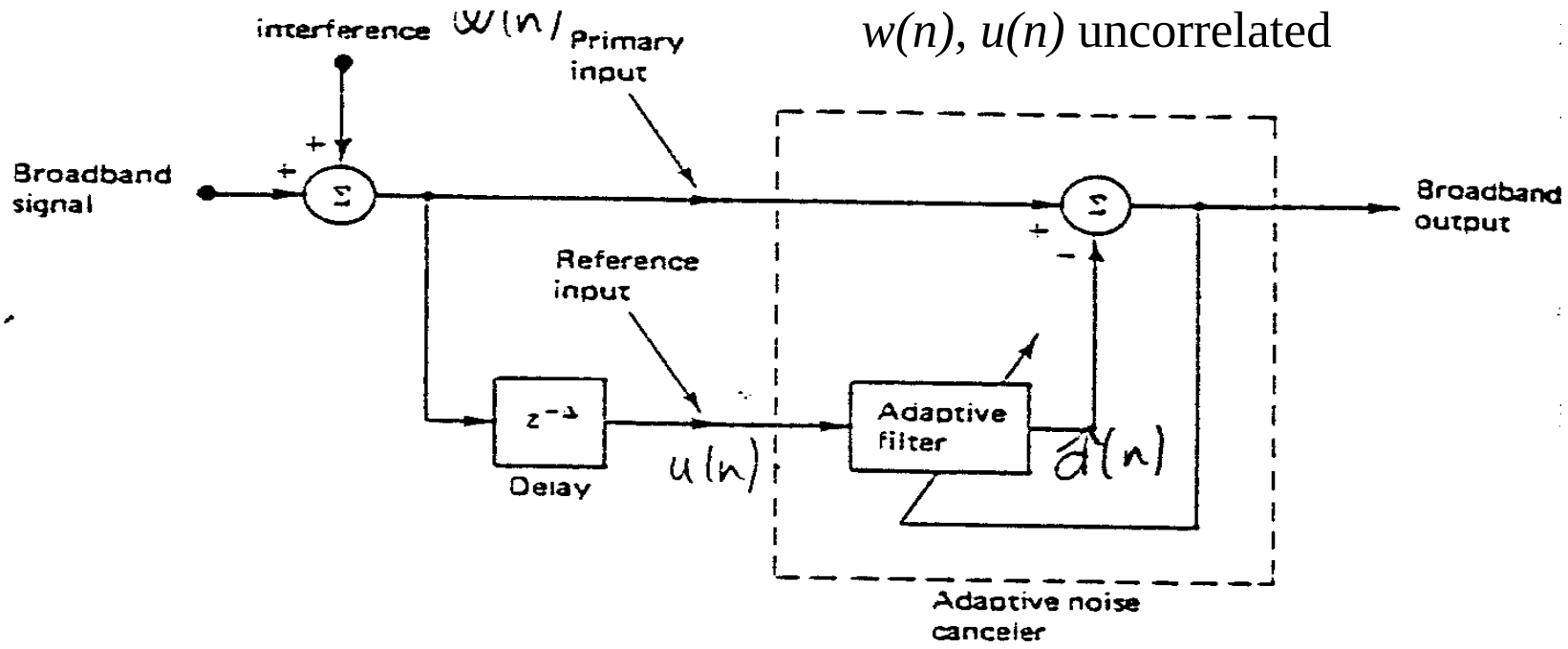

w2(n), The Second Filter Coefficient, mu = 0.07
w2(n)
0.2
0
-0.2
-0.4
-0.6
0
50
100
150
200
250
300
350
400


Trajectory of Filter Coefficients using LMS Algorithm, mu = 0.07
0.5
0.4
0.3
0.2
0.1
0
-0.1
-0.2
-0.3
-0.4
-0.5
w2(n)
0
0.2
0.4
0.6
0.8
1
w1(n)
1
0.5
0.25
0.15
0.1


Desired Error Signal e, mu = 0.07
Amplitude
Itteration
2
1
0
-1
-2
0
50
100
150
200
250
300
350
400


Adaptive Filter Output and Interference Signal
Amplitude
Itteration
2
1
0
-1
-2
0
50
100
150
200
250
300
350
400

2. Adaptive noise cancellation without external reference



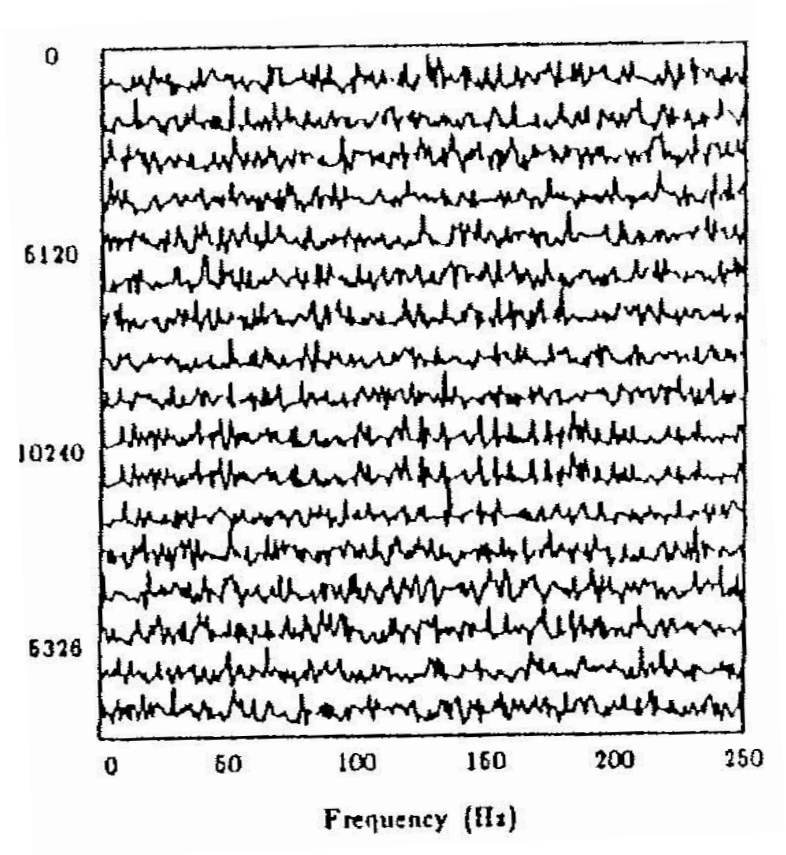
w(n), u(n) uncorrelated

Figure 12.34 Canceling periodic interference without an external reference source. From B. Widrow et al., *Adaptive Noise Canceling: Principles and Applications*. © December 1975, IEEE.

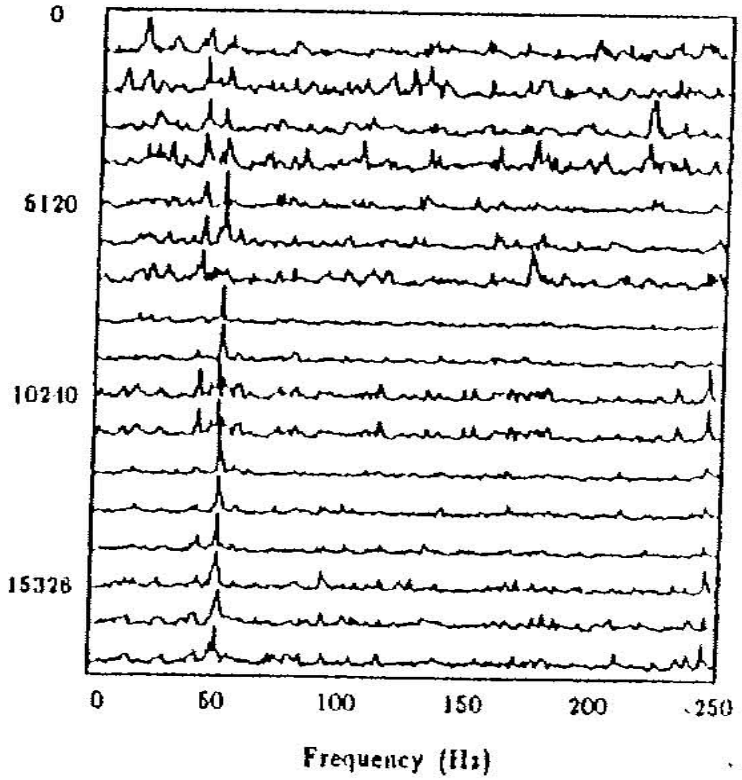
Assume interference is periodic (60Hz hum, engine noise…)

ALE example: 1 sine in noise,

SNR=-23.1dB, filter length P=200, step size=10^{-11},
Each line represents a 521-point amplitude spectrum



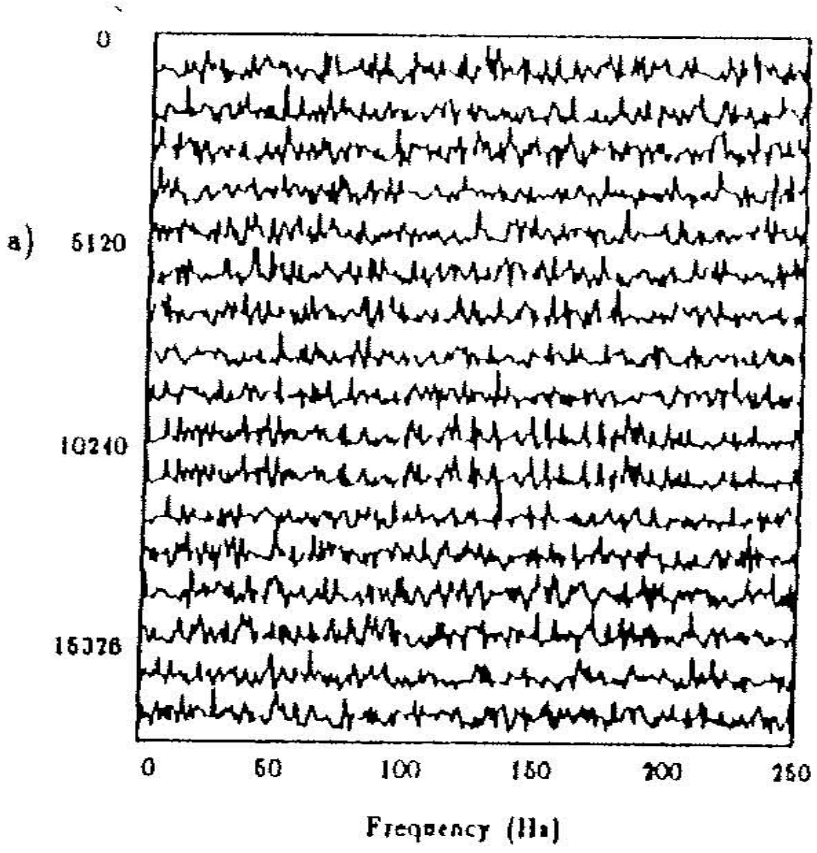
Filter input



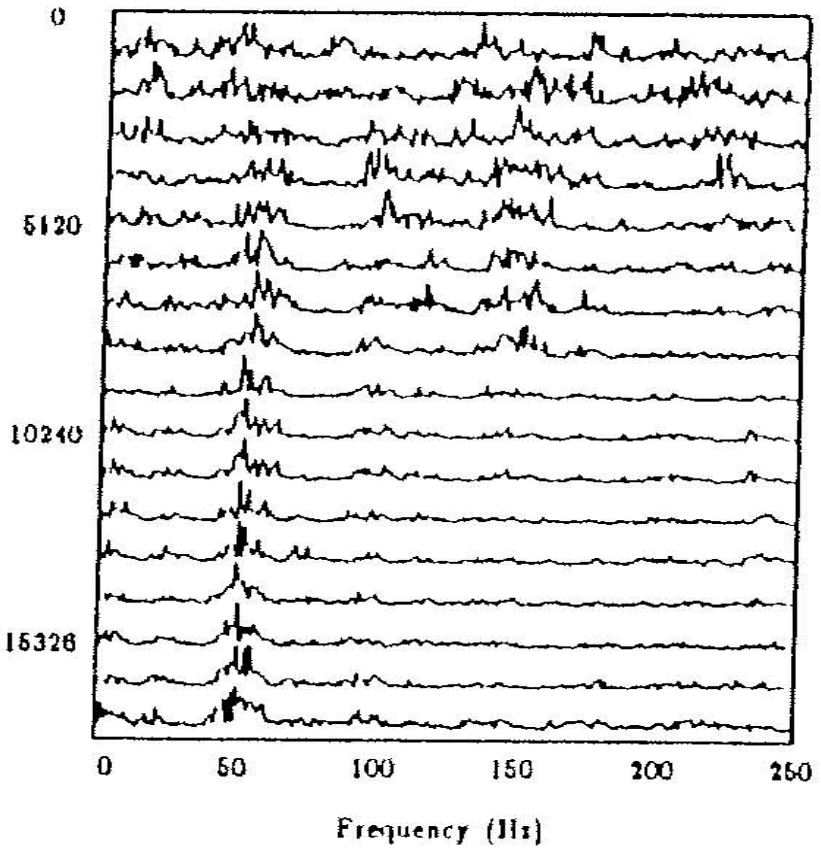
Filter output

ALE example: 1 sine in noise,

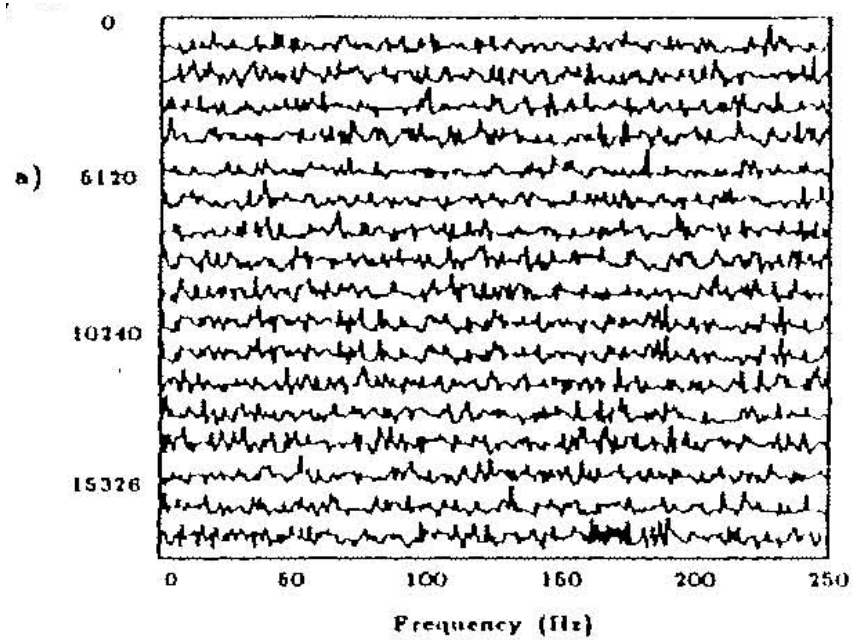
SNR=-23.1dB, filter length P=40, step size=2.510^{-11},
Each line represents a 521-point amplitude spectrum



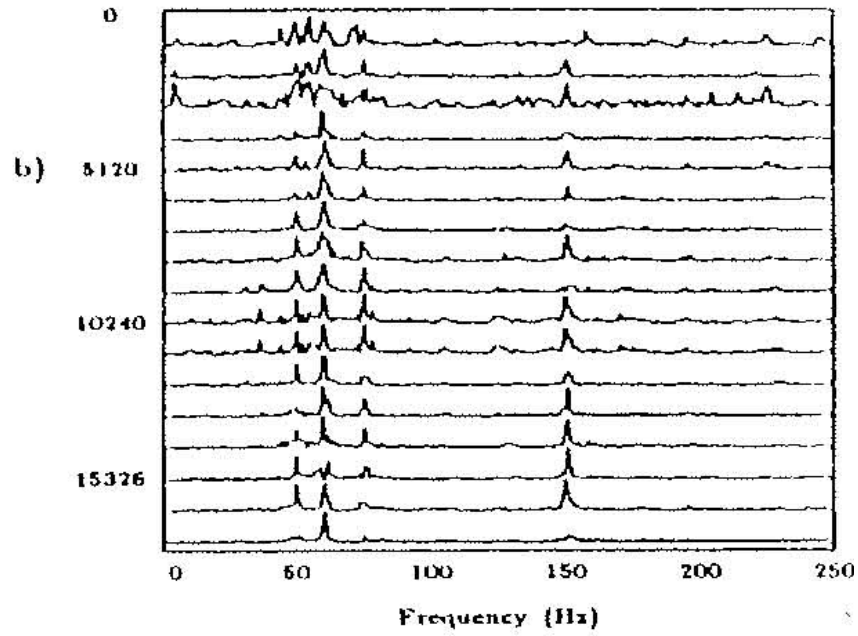
Filter input



Filter output



Filter input



Filter output

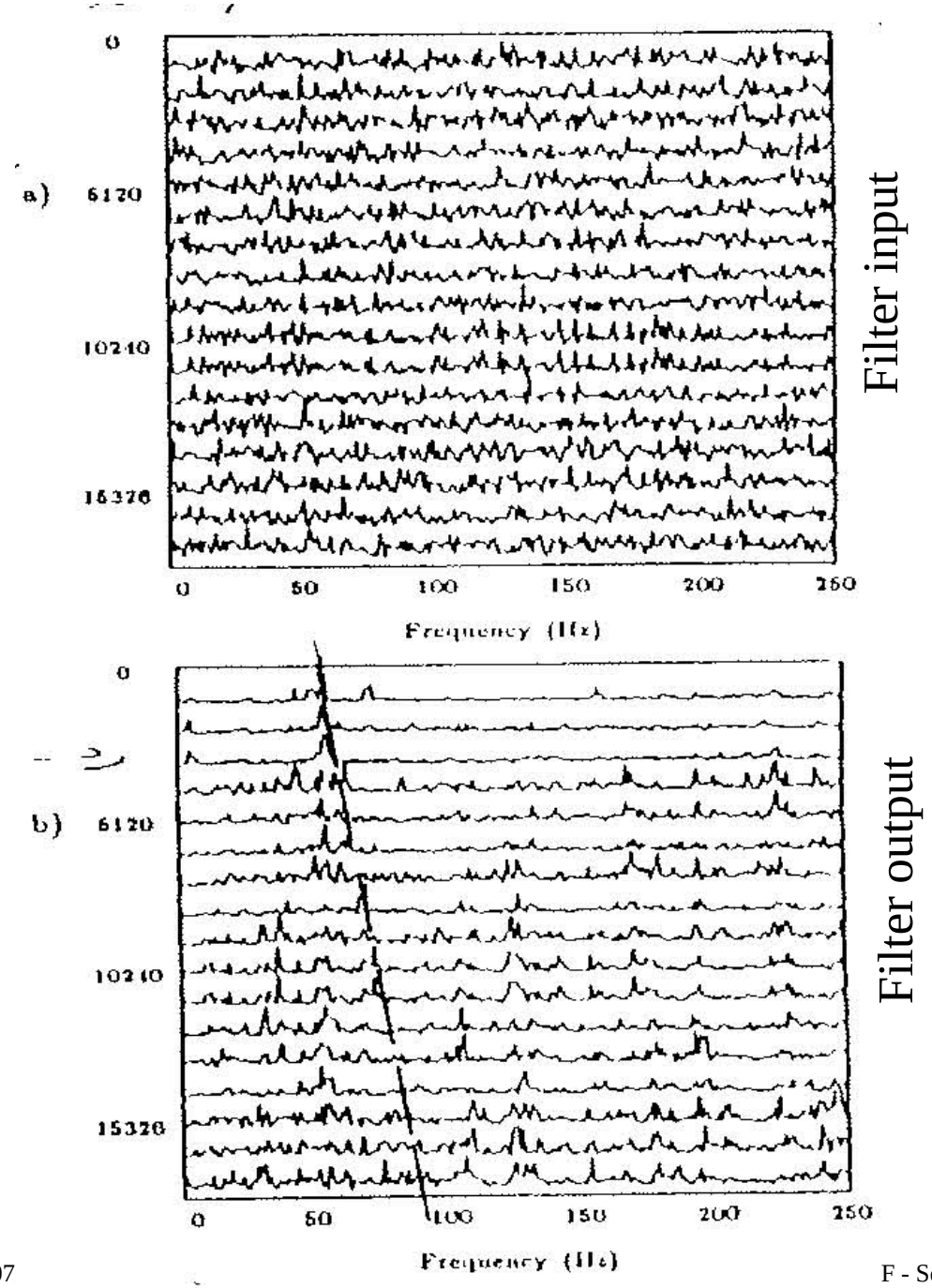
ALE, sines in noise

SNR=-15.4dB

Filter length P=200

Sine frequencies: 50, 60, 75, 150Hz

512-point amplitude spectrum



ALE, linear chirp in noise

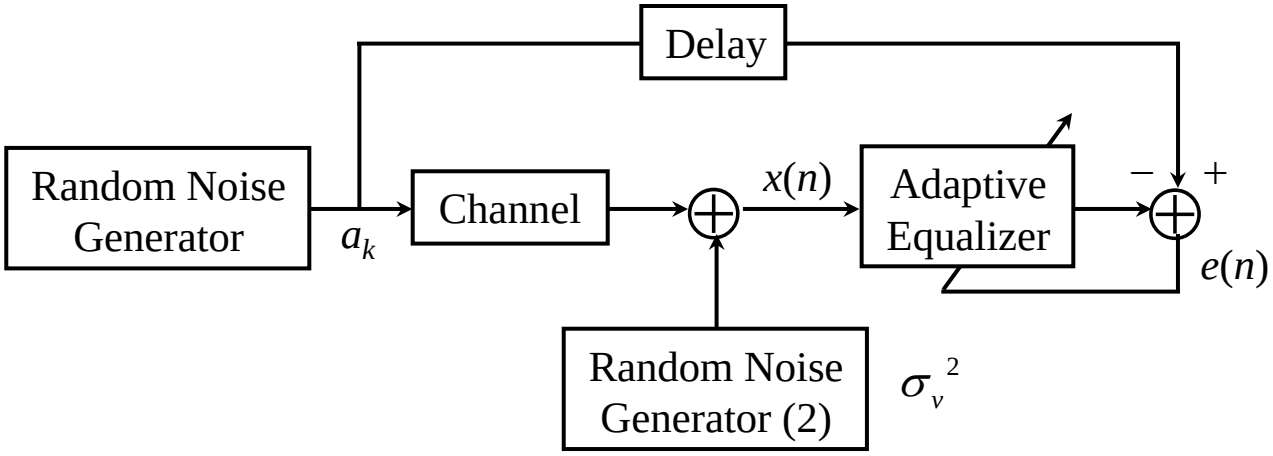
SNR=-15.4dB

Filter length P=200

Linear chirp: from 50 to 100Hz over 20,000 samples.

Step size: $2.5\ 10^{-2}$

• **Application to Adaptive Equalization** (same example as for FIR implementation done earlier)



Goal: Design the adaptive equalizer to correct distortion produced by channel in the presence of noise.

Assume: • $\{a_n\} = \pm 1$ with zero mean (BPSK sequence).

• channel impulse response:

$$c_n = \begin{cases} \frac{1}{2}\left[1+\cos\left(\frac{2\pi}{W}(n-2)\right)\right] & n=1,2,3 \\ 0 \quad ow \end{cases}$$

- channel impulse response:

$$c_n = \begin{cases} \frac{1}{2}\left[1+\cos\left(\frac{2\pi}{W}(n-2)\right)\right] & n=1,2,3 \\ 0 \quad ow \end{cases} \qquad \text{(BPSK sequence)}$$

$$H_c(z) = c_0 + c_1 z^{-1} + c_2 z^{-2}$$

W represents amount of amplitude distortion introduced by channel.

Assume W= 3.1➔c_0=0.2798,c_1=1,c_2=0.2798

(1) Assume $\sigma_v = 0$ and instantaneous transfer (delay = 0)

a) Design the filter needed to cancel transmission effects.

b) Identify whether the filer is causal and/or stable.

c) Can we use a FIR filter to cancel channel transmission effects?

(2) Assume $\sigma_v^2 = 0.001,\ P = 11, W = 3.1$

LMS
$\gamma = 0.075$

Figure 9.17 $\rightarrow$ eigenvalue spread increase causes:

- increased error
- algorithm to slow down

Figure 9.19

- faster convergence with larger step size
- average error increases with increasing step size

LMS

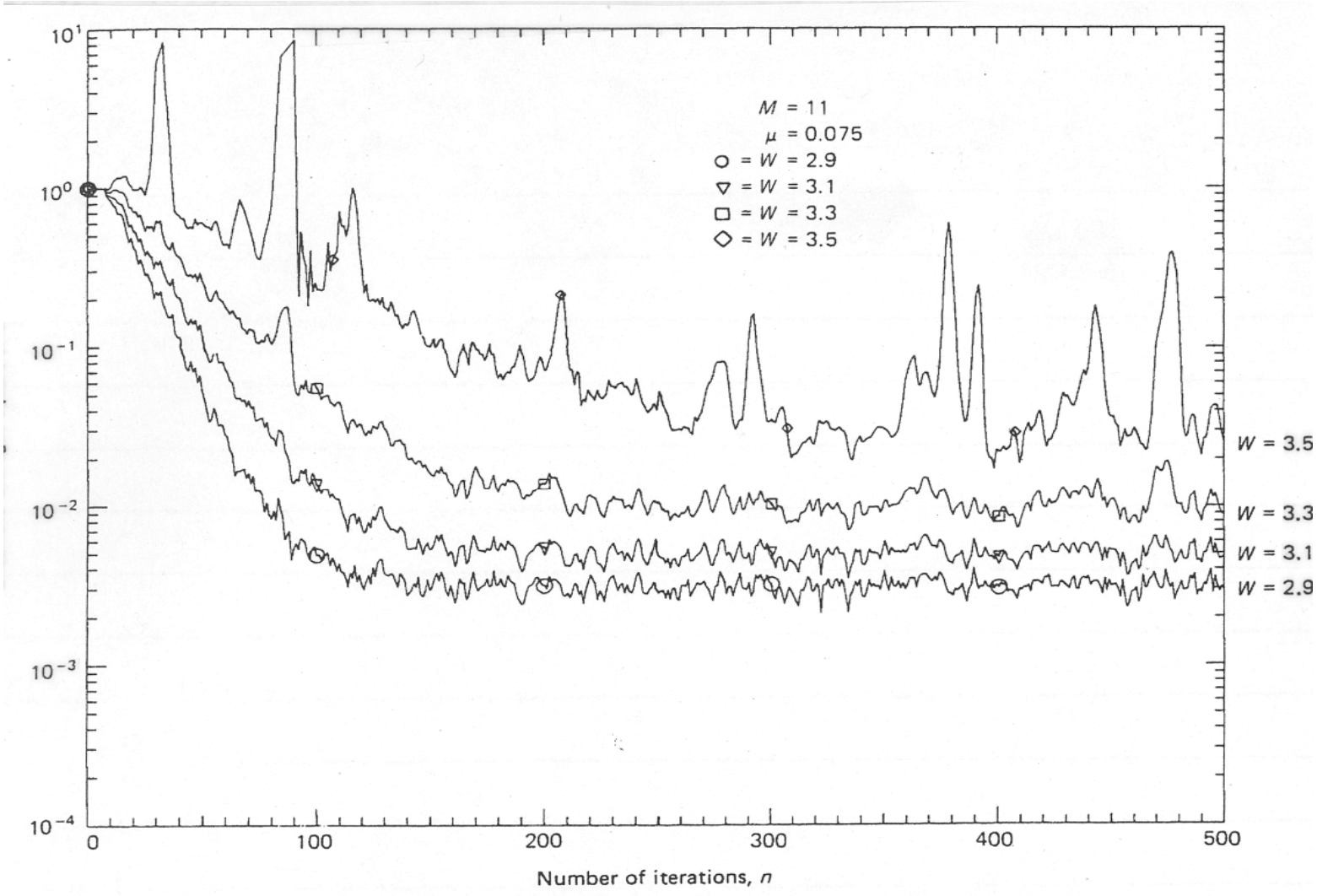


Figure 9.17 Learning curves of the LMS algorithm for adaptive equalizer with number of taps $M = 11$, step-size parameter $\mu = 0.075$, and varying eigenvalue spread $\chi(\mathbf{R})$.

[Haykin]

LMS

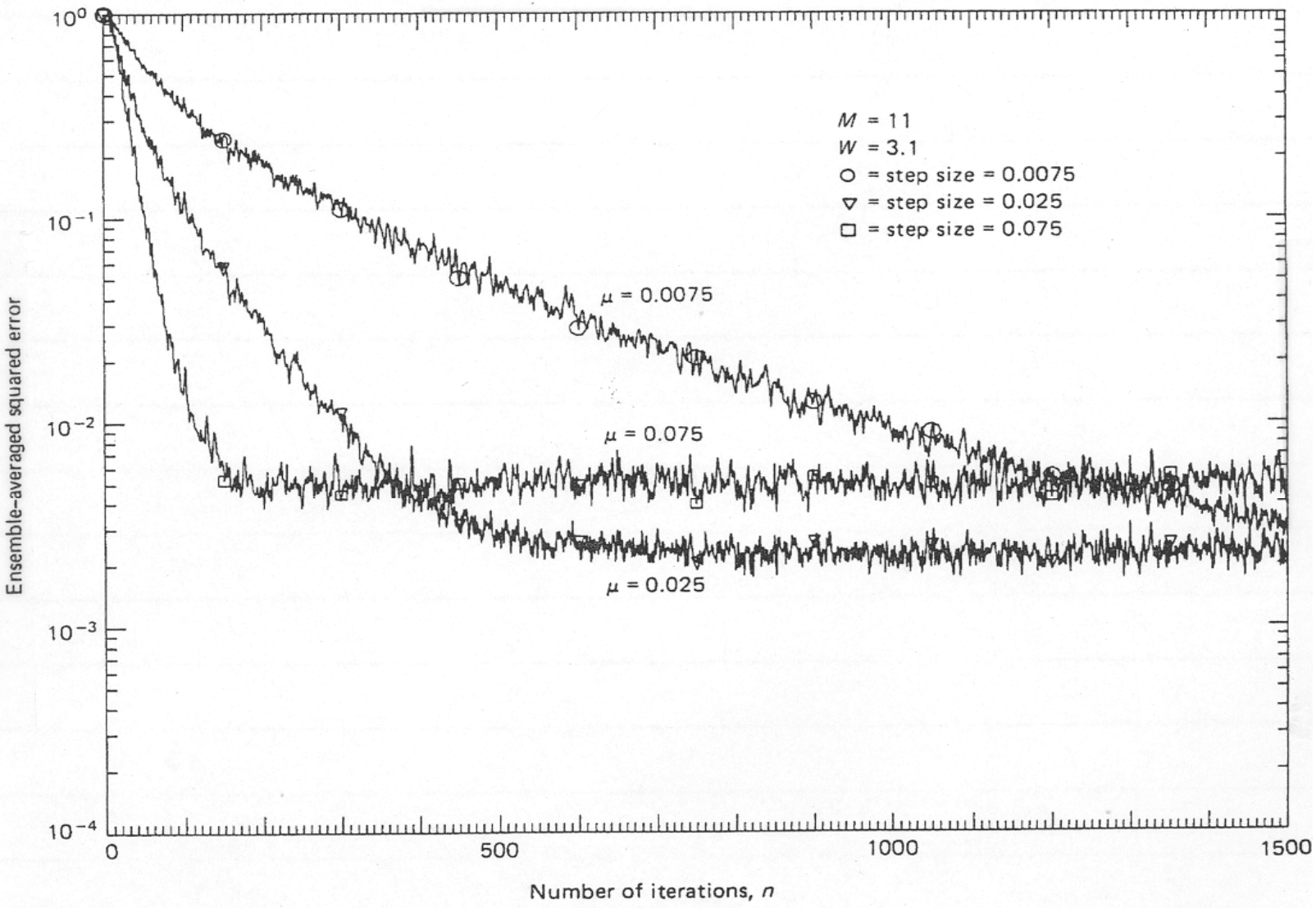


Figure 9.19 Learning curves of the LMS algorithm for adaptive equalizer with number of taps $M = 11$, fixed eigenvalue spread, and varying step-size parameter μ.

[Haykin]

- **Application to the Constant Modulus Algorithm** (CMA)

Communication signals have specific properties which can be further exploited

➔Complex FM, PM, FSK, PSK have constant amplitudes, i.e., constant modulus

When the desired signal is known to have constant amplitude, the quantity to be monitored is deviation from the constant modulus property.

$$\xi(\underline{h}(n)) = E\left\{\left|V(\underline{h}(n))\right|^2\right\},$$

$$\text{with } V(\underline{h}(n)) = |\hat{d}(n)|^2 - A^2$$

Filter output

Constant envelope value

Apply LMS iterative update approach for basic adaptive CMA algorithm

Recall LMS iteration

$$\xi(\underline{h}(n)) = \sigma_e^2(\underline{h}(n)) = E\left[\left|d(n) - \hat{d}(n)\right|^2\right]$$

$$\underline{h}(n+1) = \underline{h}(n) - \gamma V_{\underline{h}}(\xi(\underline{h}(n)))$$

$$\underline{h}(n+1) = \underline{h}(n) + 2\gamma \underline{x}(n) e^*(n), \quad 0 < \gamma < 1/PR_x(0)$$

Select a different error to monitor: deviation from the constant modulus behavior

CMA

$$\xi(\underline{h}(n)) = E\left\{\left(V(\underline{h}(n))\right)^2\right\},$$

$$\text{with } V(\underline{h}(n)) = |\hat{d}(n)|^2 - A^2$$

Need to compute $\nabla_{\underline{h}}(\xi(\underline{h}(n))$

$$\xi(\underline{h}(n)) = E\left\{\left(|\hat{d}(n)|^2 - A^2\right)^2\right\}$$

Use Instantaneous gradient

$$V(\underline{h}(n)) = \left(\hat{d}(n)|^2 - A^2\right)^2$$

Use the complex gradient property

$$\nabla_{\underline{h}}(\underline{h}^H B\underline{h}) = B\underline{h}$$

$$\nabla_{\underline{h}}(V(\underline{h}(n)) = \nabla_{\underline{h}}\left(\left(\hat{d}(n)|^2 - A^2\right)^2\right)$$