INTRODUCTION TO ADAPTIVE SIGNAL PROCESSING

CONTENTS

REFERENCES:

Sections 1, 2, 3

R1. J. Lim Ed. Advanced topics in Signal Processing. Prentice Hall. 1988.

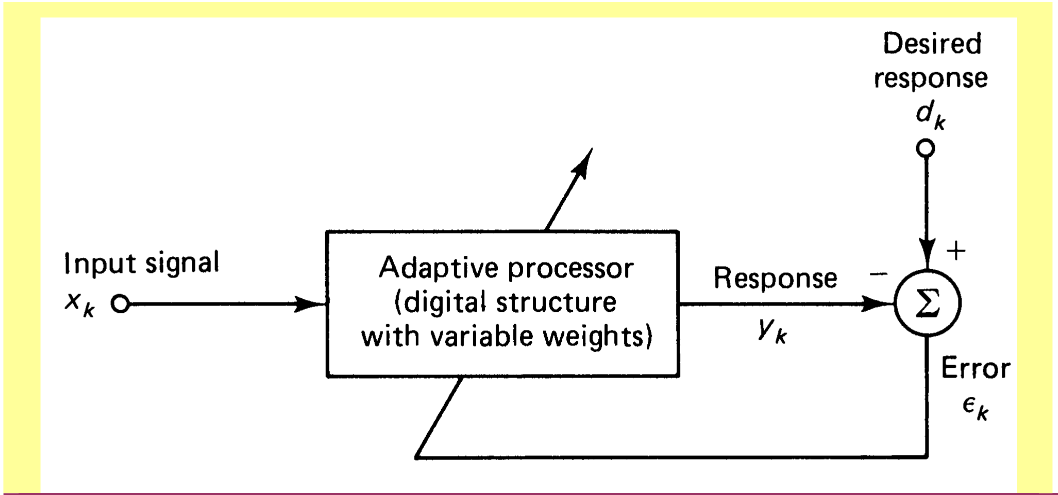
Sections 2,3, 4, 5, 6

R2. J. Proakis et al. Advanced Digital Signal Processing, Macmillan, 1992.

R3. J. Proakis. Digital Communications. 3rd Edition. McGraw-Hill International Editions. 1995.

1. INTRODUCTION

Adaptive signal processing attempts to design systems that adapt to the operating environments.



- **A commonly used structure consists of an adaptive linear combiner with input x(n) and output y(n).**
- **The weights or coefficients are variable.**

Fig. 1 Basic elements in a single-input, performance-feedback adaptive system.

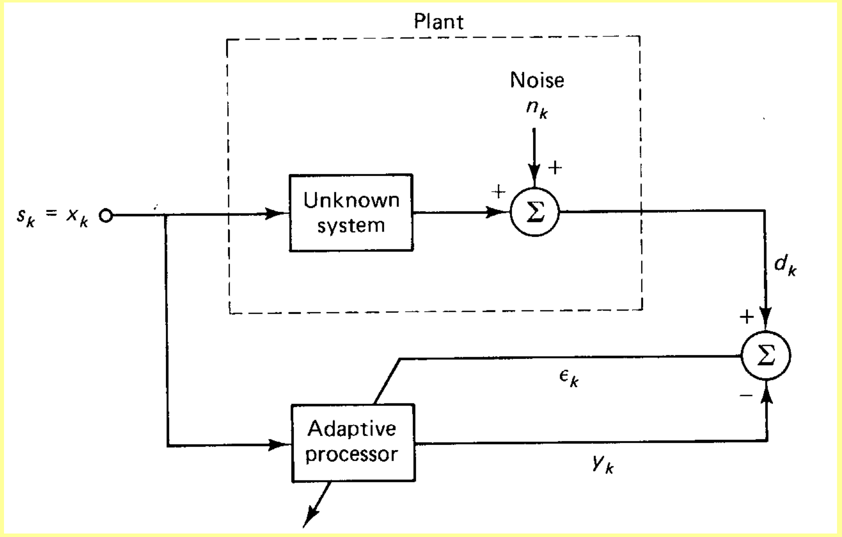
- **The weights are adjusted continuously, by means of an adaptive algorithm, to minimize some measure of the error,**

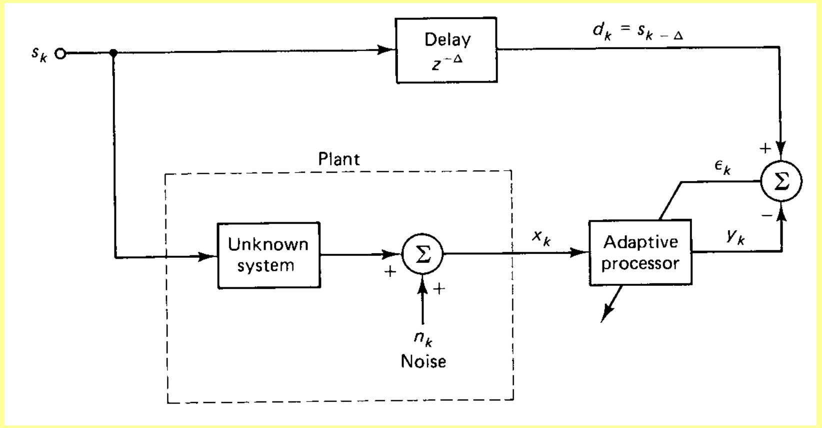
$$e(n) = d(n) - y(n), \tag{1-1}$$

between the desired response, $d(n)$, and the system response, $y(n)$.

The followings are some applications of the basic adaptive structure:

1. Adaptive Modeling or system Identification

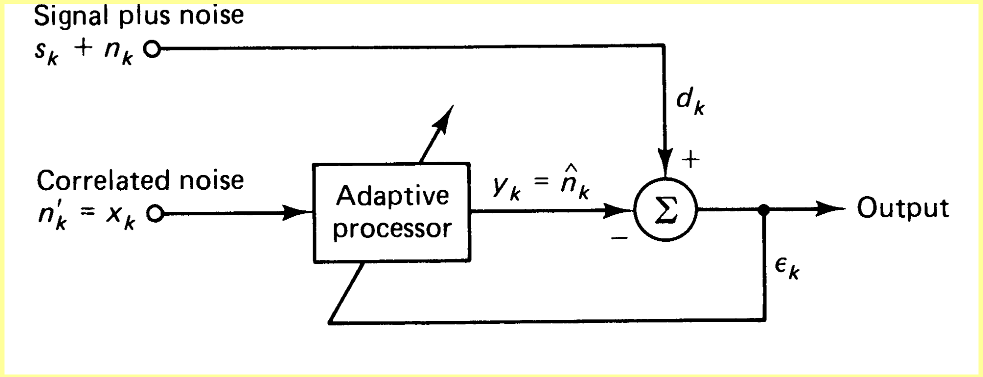




Forward modeling | **Backward modeling**

- By minimizing the averaged error between the outputs of the unknown system and the linear combiner, the linear combiner can be viewed as an approximation of the target system.

2. Adaptive Interference Cancellation.



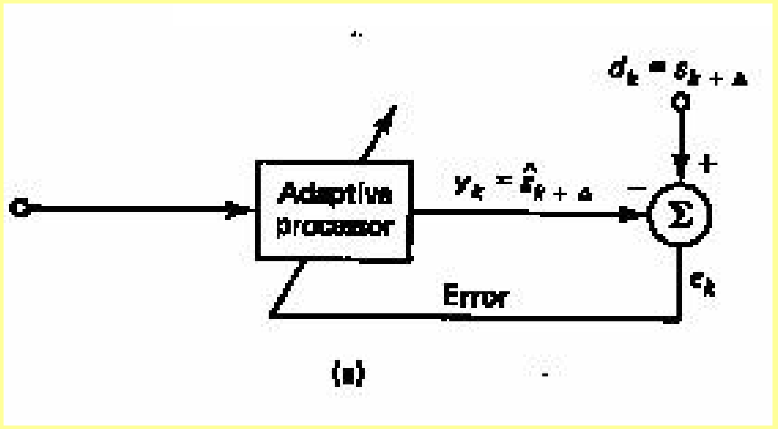
- to be corrupted by additive noise/interference n_k.

- If a correlated version of the noise/interference n'_k is available, the linear combiner can be used to transform n'_k to approximate and hence suppress n_k.

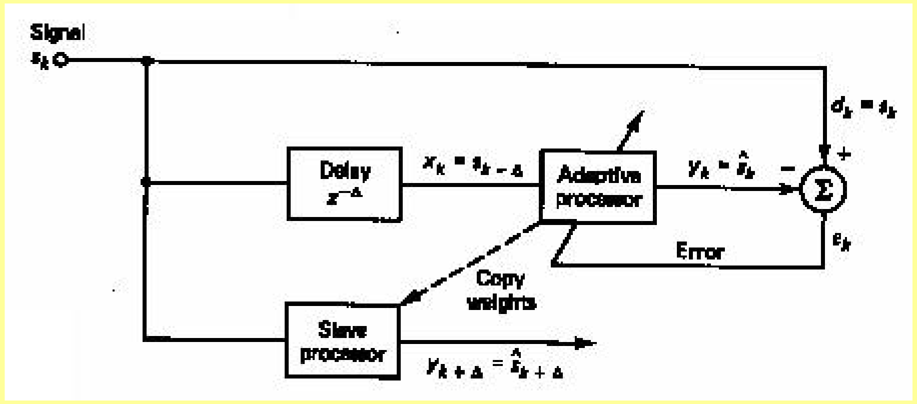
- The correlated noise n'_k must not contain a correlated component of s_k. Otherwise, the signal component will also be cancelled.

- Applications include suppressing interference in medical applications, adaptive beamformers, multiuser detection, etc.

3. Adaptive Prediction



(a) unrealisable form



(b) realisable form

- If *d*(*k*) is chosen as $s(k+\Delta)$, we are predicting the future inputs from the past inputs, i.e. forwards linear prediction.
- For causal implementation, the input is delayed by Δ so that d(n) is equal to $s(k)$.
- There are many other applications of these basis configurations.

2. MINIMIUM MEAN-SQUARE-ERROR CRITERION

- A commonly used error measure is the **Mean-Square-Error (MSE)**, $E[(e(n))^2]$
- Close form formula is available if the filter is linear. The expectation means that the averaged least squares error is minimized, which helps to tolerate additive noise and modelling errors.
- The signals involved in an adaptive system are usually **non-stationary (slowly time-varying)**, with the system adapting to changing signal conditions. Before developing the basic theory, it is useful to assume temporarily that the signals are stationary.
- The MSE is given by:

$$\begin{aligned} MSE &= E[(e(n))^2] = E[(d(n) - y(n))^2] \\ &= E[(d(n))^2] + E[(y(n))^2] - 2E[d(n)y(n)]. \end{aligned} \tag{2-1}$$

FIR/IIR FILTERS?

- *W*(z) is usually an FIR filter because adaptation using IIR filters are complicated by the two factors:
  1. Unconstrained poles can move outside the unit circle during adaptation, causing instability.
  2. The error surface can have flat areas and local minima, making gradient search techniques unreliable.

- For an FIR filter with length *L*,

$$W(z) = \sum_{i=0}^{L-1} w_i z^{-i}, \quad y(n) = \sum_{i=0}^{L-1} w_i x(n-i) \tag{2-2}$$

$$E[(y(n))^2] = \sum_{i=0}^{L-1}\sum_{m=0}^{L-1} w_i w_m E[x(n-i)x(n-m)] = \sum_{i=0}^{L-1}\sum_{m=0}^{L-1} w_i w_m r_{xx}(n, i-m)$$

$$E[d(n)y(n))] = \sum_{i=0}^{L-1} w_i E[x(n-i)d(n)] = \sum_{i=0}^{L-1} w_i r_{xd}(n-i,i).$$

where

$r_{xd}(n,i) = E[x(n)d(n+i)]$ **is the cross-correlation between** $x(n)$ **and** $d(n)$**,** $r_{xx}(n,i) = E[x(n)x(n+i)]$ **is the auto-correlation of** $x(n)$**.**

THE NORMAL EQUATION

- For wide-sense stationary signal, the second order statistics are independent of time shifts, i.e. independent of the variable *n*. Therefore,

$$r_{dx}(n-i,i) = r_{dx}(i), \text{ and } r_{xx}(n,i) = r_{xx}(i). \quad \text{(2-3)}$$

The MSE is a quadratic function in the filter coefficient.

$$MSE = r_{dd}(0) + \sum_{i=0}^{L-1}\sum_{m=0}^{L-1} w_i w_m r_{xx}(i-m) - 2\sum_{i=0}^{L-1} w_i r_{xd}(i). \quad \text{(2-4)}$$

The optimum filter coefficients are unique and are obtained by setting the gradient $\nabla_W(MSE)$ to zero.

$$\nabla_W(MSE) = \left[\frac{\partial(MSE)}{\partial w_0}\ \frac{\partial(MSE)}{\partial w_1} \cdots \frac{\partial(MSE)}{\partial w_{L-1}}\right] = 2\boldsymbol{RW} - 2\boldsymbol{P}\ , \quad \text{(2-5)}$$

where $\boldsymbol{W} = [w_0 \quad w_1 \quad \cdots \quad w_{L-1}]^T$,

$\boldsymbol{P} = [r_{dx}(0) \quad r_{dx}(1) \quad \cdots \quad r_{dx}(L-1)]^T$,

and $[\boldsymbol{R}]_{m,n} = r_{xx}(m-n)$,

$[\boldsymbol{A}]_{m,n}$ is the (m,n) element of matrix $\boldsymbol{A}$.

The optimal solution is governed by the "normal equation"

$$\boldsymbol{RW} = \boldsymbol{P}, \tag{2-6}$$

and the solution and minimum MSE are given respectively by

$$\boldsymbol{W}_{opt} = \boldsymbol{R}^{-1}\boldsymbol{P} \quad , \tag{2-7}$$

and $$(MSE)_{\min} = r_{dd}(0) - \boldsymbol{P}^{\mathrm{T}}\boldsymbol{W}_{opt}. \tag{2-8}$$

ESTIMATING *R* AND *P*

- $\boldsymbol{R}$ and $\boldsymbol{P}$ are usually unknown in prior and they have to be estimated from the input $x(n)$ and the desired signal $d(n)$.
- Since the ensemble averages

  $r_{xd}(n) = E[x(n)d(n+i)]$ and $r_{xx}(i) = E[x(n)x(n+i)]$,

  are difficult to estimate (the probability density function (pdf) of stochastic processes $x(n)$ and $d(n)$ are usually unknown in advance), they are usually replaced by time averages. That is, they are assumed to be ergodic.

$$\boldsymbol{R}(n) = \sum_{i=1}^{n} \lambda^{n-i} \boldsymbol{X}(i)\boldsymbol{X}^{T}(i) = \lambda \boldsymbol{R}(n-1) + \boldsymbol{X}(n)\boldsymbol{X}^{T}(n) \qquad (2.8a)$$

$$\boldsymbol{P}(n) = \sum_{i=1}^{n} \lambda^{n-i} d(i)\boldsymbol{X}(i) = \lambda \boldsymbol{P}(n-1) + d(n)\boldsymbol{X}(n) \qquad (2.8b)$$

where $\boldsymbol{X}(n) = [x(n) \quad x(n-1) \quad \cdots \quad x(n-L-1)]^T$, and λ is a positive number called forgetting factor having a value between 0 and 1.

- **The value of λ is usually chosen to be close to one, say 0.97, to allow the system to adapt to non-stationary conditions (e.g. when the system changes slowly with time).**
- **The weight vector at the (n+1) *th* time instant is**

$$\boldsymbol{W}(n) = \boldsymbol{R}^{-1}(n)\boldsymbol{P}(n) \tag{2.9}$$

Solving this equation requires $O(L^3)$ arithmetic operations and is prohibitively for real-time operations.

They are usually solved iteratively using different adaptive filtering algorithms (LMS, RLS algorithms). They differ in arithmetic complexity per iteration, initial convergence speed, steady state error, response to sudden system change, and tracking slowly varying systems.

3. THE LMS ALGORITHM

- **The LMS algorithm is based on the method of steepest descent.**

One begins with an arbitrarily chosen coefficient vector say $\boldsymbol{W}(0)$. Each tap coefficient is changed in the direction opposite to its corresponding gradient component in the gradient vector $\boldsymbol{g}(n)$, which is the derivative of the *MSE* with respect to the filter coefficients.

$$W(n+1) = W(n) - \mu' \cdot \boldsymbol{g}(n) \tag{3.1}$$

where μ' is a step-size parameter. In the LMS algorithm, $\boldsymbol{g}(n)$ is estimated continuously. A commonly used method is to approximate the MSE by the instantaneous error

$$MSE \approx (e(n))^2 = (d(n) - \boldsymbol{W}^T(n)\boldsymbol{X}(n))^2 \tag{3.2}$$

Thus, the estimated gradient is

$$\hat{\boldsymbol{g}}(n) = \nabla_{\boldsymbol{W}} (e(n))^2 = 2e(n) \cdot \nabla_{\boldsymbol{W}} e(n) = -2e(n)\boldsymbol{X}(n) \tag{3.3}$$

The LMS or stochastic gradient algorithm is given by

$$\boldsymbol{W}(n+1) = \boldsymbol{W}(n) + \mu \cdot e(n)\boldsymbol{X}(n) \tag{3.4}$$

The value of μ $(= 2\mu')$ controls the initial convergence rate (see Section 5) and the steady state error of the algorithm.

- It can be shown similarly that for complex signal, the LMS algorithm is given by

$$\boldsymbol{W}(n+1) = \boldsymbol{W}(n) + \mu \cdot e(n)\boldsymbol{X}^*(n) \tag{3.5}$$

where * denotes complex conjugate.

4. THE RECURSIVE LEAST SQUARE (RLS) ALGORITHM

In the RLS algorithm, instead of calculating the matrix inverse $\boldsymbol{R}^{-1}(n)$ of $\boldsymbol{R}(n)$ at each iteration, it is computed recursively from that of $\boldsymbol{R}(n-1)$, which is related to $\boldsymbol{R}(n)$ by

$$\boldsymbol{R}(n)=\sum_{i=1}^{n}\lambda^{n-i}\boldsymbol{X}(i)\boldsymbol{X}^{T}(i)=\lambda\boldsymbol{R}(n-1)+\boldsymbol{X}(n)\boldsymbol{X}^{T}(n). \qquad (4.1)$$

It makes use of the following matrix inversion formula

$$\left(\boldsymbol{A}+\beta\cdot\boldsymbol{x}\boldsymbol{y}^{T}\right)^{-1}=\boldsymbol{A}^{-1}\left(\boldsymbol{I}-\frac{\beta\cdot\boldsymbol{x}\boldsymbol{y}^{T}\boldsymbol{A}^{-1}}{1+\beta\cdot\boldsymbol{y}^{T}\boldsymbol{A}^{-1}\boldsymbol{x}}\right) \qquad (4.2)$$

where $\boldsymbol{x}$ and $\boldsymbol{y}$ are column vectors of dimension $(L\times 1)$, $\boldsymbol{A}$ is any non-singular matrix of dimension $(L\times L)$, and β is a constant.

Letting $\boldsymbol{R}(n-1)=\lambda\boldsymbol{A}$, $\boldsymbol{x}=\boldsymbol{y}=\boldsymbol{X}(n)$, and $\beta=1$, we have

$$\boldsymbol{R}^{-1}(n) = \lambda^{-1}(\boldsymbol{I} - \boldsymbol{K}(n)\boldsymbol{X}^T(n)) \cdot \boldsymbol{R}^{-1}(n-1) \qquad \textbf{(4.3)}$$

where $\boldsymbol{K}(n)$ is the Kalman gain vector given by

$$\boldsymbol{K}(n) = \frac{\boldsymbol{R}^{-1}(n-1)\boldsymbol{X}(n)}{\lambda + \boldsymbol{X}^T(n)\boldsymbol{R}^{-1}(n-1)\boldsymbol{X}(n)}. \qquad \textbf{(4.4)}$$

- **The filter coefficient at the (n+1) *th* time instant is**

$$\begin{aligned}
\boldsymbol{W}(n) &= \boldsymbol{R}^{-1}(n)\boldsymbol{P}(n) \\
&= \lambda^{-1}[\boldsymbol{I} - \boldsymbol{K}(n)\boldsymbol{X}^T(n)] \cdot \boldsymbol{R}^{-1}(n-1)[\lambda\boldsymbol{P}(n-1) + d(n)\boldsymbol{X}(n)] \\
&= \boldsymbol{R}^{-1}(n-1)\boldsymbol{P}(n-1) - \boldsymbol{K}(n)\boldsymbol{X}^T(n)\boldsymbol{R}^{-1}(n-1)\boldsymbol{P}(n-1) \\
&\qquad + \lambda^{-1}d(n)\boldsymbol{R}^{-1}(n-1)\boldsymbol{X}(n) \qquad \textbf{(4.5)} \\
&\qquad - \lambda^{-1}d(n)\boldsymbol{K}(n)\boldsymbol{X}^T(n)\boldsymbol{R}^{-1}(n-1)\boldsymbol{X}(n) \\
&= \boldsymbol{W}(n-1) - \boldsymbol{K}(n)\boldsymbol{X}^T(n)\boldsymbol{W}(n-1) \\
&\qquad + \lambda^{-1}d(n)[\boldsymbol{R}^{-1}(n-1)\boldsymbol{X}(n) - \boldsymbol{K}(n)\boldsymbol{X}^T(n)\boldsymbol{R}^{-1}(n-1)\boldsymbol{X}(n)].
\end{aligned}$$

Here we have used the fact that $\boldsymbol{W}(n-1)=\boldsymbol{R}^{-1}(n-1)\boldsymbol{P}(n-1)$. Using the identity in (4.4), we also have

$$\lambda\boldsymbol{K}(n)=\boldsymbol{R}^{-1}(n-1)\boldsymbol{X}(n)-\boldsymbol{K}(n)\boldsymbol{X}^{T}(n)\boldsymbol{R}^{-1}(n-1)\boldsymbol{X}(n). \tag{4.6}$$

Substituting (4.6) into (4.5), one gets

$$\boldsymbol{W}(n)=\boldsymbol{W}(n-1)+\boldsymbol{K}(n)[d(n)-\boldsymbol{X}^{T}(n)\boldsymbol{W}(n-1)]. \tag{4.7}$$

Since $\boldsymbol{X}^{T}(n)\boldsymbol{W}(n-1)$ is the output of the adaptive filter at time *n* based on the use of the filter coefficients at time *n-1*, the term $d(n)-\boldsymbol{X}^{T}(n)\boldsymbol{W}(n-1)$ inside the bracket of (4.7) is the error at time *n*.

$$e(n)=d(n)-\boldsymbol{X}^{T}(n)\boldsymbol{W}(n-1). \tag{4.8}$$

Consequently, (4.7) is simplified to

$$\boldsymbol{W}(n)=\boldsymbol{W}(n-1)+\boldsymbol{K}(n)e(n). \tag{4.9}$$

(4.3), (4.4) , (4.8), and (4.9) constitute the RLS algorithm.

SUMMARY OF RLS ALGORITHM

1. **Compute the filter output:**

$$y(n) = \boldsymbol{X}^T(n)\boldsymbol{W}(n-1).$$

2. **Compute the error**

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

3. **Compute the Kalman gain vector :**

$$\boldsymbol{K}(n) = \frac{\boldsymbol{R}^{-1}(n-1)\boldsymbol{X}(n)}{\lambda + \boldsymbol{X}^T(n)\boldsymbol{R}^{-1}(n-1)\boldsymbol{X}(n)}.$$

4. **Update the inverse of the correlation matrix**

$$\boldsymbol{R}^{-1}(n) = \lambda^{-1}\left(\boldsymbol{I} - \boldsymbol{K}(n)\boldsymbol{X}^T(n)\right)\cdot \boldsymbol{R}^{-1}(n-1).$$

5. **Update the coefficient vector of the filter**

$$\boldsymbol{W}(n) = \boldsymbol{W}(n-1) \; + \boldsymbol{K}(n)e(n).$$

Initial conditions and choice of forgetting factor

- The initial values of $\boldsymbol{R}(0)$ and $\boldsymbol{W}(0)$ are usually chosen to be a constant multiple of the identity matrix and the zero vector.
- The forgetting factor λ controls the effective amount of data used in the averaging and hence the degree to which the algorithms can track variations in signal characteristics.
- It also controls the variance of the long-term estimate. The closer the value of λ is to the value one, the lower will be the mis-adjustment factor of the RLS algorithm. Its tracking ability, however, will also be slower. This is known as the bias-variance trade-off problem.
- It can be shown that the RLS algorithm always converges for $0 < \lambda < 1$, if implemented exactly (numerical error can cause the algorithm to diverge).

Arithmetic complexity and numerical errors

- **The arithmetic complexity of the RLS algorithm is $O(L^2)$ instead of $O(L^3)$ for direct inversion.**
- **In addition to the time recursion mentioned above, it is also possible to perform an order recursion in the weight vector yielding very fast RLS algorithms with $O(L)$ arithmetic complexity.**
- **The basic RLS algorithm and the fast RLS, however, are very sensitive to numerically errors. Techniques such as error feedback or QR decomposition can be used to improve their numerical accuracies.**
- **Other commonly used algorithms include the transform domain LMS algorithm, gradient adaptive lattice, and the fast Newton algorithms.**

5. CONVERGENCE ANALYSIS OF THE LMS ALGORITHM

Assuming that the input and desired signals of the adaptive filter are wide-sense stochastic processes.

Taking the expectation of both sides of the weight update equation

$$\boldsymbol{W}(n+1) = \boldsymbol{W}(n) + \mu \cdot e(n)\boldsymbol{X}^*(n), \tag{5.1}$$

we have

$$\begin{aligned} E[\boldsymbol{W}(n+1)] &= E[\boldsymbol{W}(n)] + \mu \cdot E[e(n)\boldsymbol{X}^*(n)] \\ &= E[\boldsymbol{W}(n)] + \mu \cdot E[(d(n) - \boldsymbol{X}^T(n)\boldsymbol{W}(n)) \cdot \boldsymbol{X}^*(n)] \\ &= E[\boldsymbol{W}(n)] + \mu \cdot (E[d(n)\boldsymbol{X}^*(n)] - E[\boldsymbol{X}^T(n)\boldsymbol{X}^*(n)] \cdot E[\boldsymbol{W}(n)]) \\ &= E[\boldsymbol{W}(n)] + \mu \cdot (\boldsymbol{r}_{dX} - \boldsymbol{R}_{XX} \cdot E[\boldsymbol{W}(n)]). \\ &= (\boldsymbol{I} - \mu \boldsymbol{R}_{XX}) \cdot E[\boldsymbol{W}(n)] + \mu \boldsymbol{r}_{dX}\ , \end{aligned} \tag{5.2}$$

where $\boldsymbol{R}_{XX} = E[\boldsymbol{X}^T(n)\boldsymbol{X}^*(n)]$ **and** $\boldsymbol{r}_{dx} = E[d(n)\boldsymbol{X}^*(n)]$. **(5.3)**

Here we have assumed that $\boldsymbol{X}(n)$ and $\boldsymbol{W}(n)$ are independent. This assumption holds if μ is small, so called slow adaptation.

It can be shown that $\boldsymbol{R}_{XX}$ is Hermitian ($\boldsymbol{R}_{XX} = (\boldsymbol{R}^*_{XX})^T$ and it can be represented as

$$\boldsymbol{R}_{XX} = \boldsymbol{U}^H \Lambda \boldsymbol{U}, \tag{5.4}$$

where $\boldsymbol{U}$ is some unitary matrix with $\boldsymbol{U} \cdot \boldsymbol{U}^H = \boldsymbol{I}$, Λ is a diagonal matrix with diagonal elements λ_k (real), $0 \le k \le L-1$, equal to the eigenvalues of $\boldsymbol{R}_{XX}$, and $\boldsymbol{A}^H = (\boldsymbol{A}^*)^T$. Pre-multiply (5.2) by $\boldsymbol{U}$, one gets the following

$$\boldsymbol{U} \cdot E[\boldsymbol{W}(n+1)] = (\boldsymbol{I} - \mu\Lambda) \cdot \boldsymbol{U} \cdot E[\boldsymbol{W}(n)] + \mu \boldsymbol{U} \cdot \boldsymbol{r}_{dX},$$

or equivalently

$$\overline{\boldsymbol{W}}_{\boldsymbol{U}}(n+1) = (\boldsymbol{I} - \mu\Lambda) \cdot \overline{\boldsymbol{W}}_{\boldsymbol{U}}(n) + \mu \cdot \bar{\boldsymbol{r}}_{dX,U}, \tag{5.5}$$

where $\overline{\boldsymbol{W}}(n) = \boldsymbol{U} \cdot E[\boldsymbol{W}(n+1)]$ and $\bar{\boldsymbol{r}}_{dX,U} = \boldsymbol{U} \cdot \boldsymbol{r}_{dX}$.

Since $(\boldsymbol{I} - \mu\Lambda)$ is a diagonal matrix, (5.5) is equivalent to a set of scalar equations given by

$$[\overline{\boldsymbol{W}}_{\boldsymbol{U}}(n+1)]_k = (1-\mu\lambda_k)\cdot[\overline{\boldsymbol{W}}_{\boldsymbol{U}}(n)]_k + \mu\cdot[\overline{\boldsymbol{r}}_{d\boldsymbol{X},\boldsymbol{U}}]_k,\ 0 \le k \le M-1. \quad (5.6)$$

- This is a first order constant coefficient difference equation, similar to a first order IIR filter. It therefore convergences when

$$|1-\mu\lambda_k| < 1. \quad (5.7)$$

Hence, the range of values of μ that the LMS algorithm converges in the mean is

$$0 < \mu < \frac{2}{\lambda_{\max}}, \quad (5.8)$$

where $\lambda_{\max}$ is the largest eigenvalue of $\boldsymbol{R}_{XX}$. Since $\boldsymbol{R}_{XX}$ is an autocorrelation matrix, its eigenvalues are nonnegative. Hence an upper bound on $\lambda_{\max}$ is

$$\lambda_{\max} < \sum_{k=0}^{M-1} \lambda_k = trace(\boldsymbol{R}_{XX}) = L \cdot r_{xx}(0), \qquad (5.9)$$

where $r_{XX}(0)$ is the input signal power that is easily estimated, and $trace(\boldsymbol{A}) = \sum_{k=0}^{L-1} a_{kk}$ for any $(L \times L)$ matrix *A*. Therefore, an upper bound on the step size μ is

$$2/(L \cdot r_{xx}(0)).$$

- The smaller the value of $|1-\mu\lambda_k|$, the faster is its convergence rate. Even if we choose the stepsize to be

$$\mu = \frac{1}{\lambda_{\max}}, \qquad (5.10)$$

the convergence rate of the LMS algorithm will however depend on the decay of the mode corresponding to the smallest eigenvalue $\lambda_{\min}$ at a rate

$$[\overline{W}(n)]_{\lambda_{\min}} \approx C\left(1-\frac{\lambda_{\min}}{\lambda_{\max}}\right)^n u(n), \qquad (5.11)$$

where *C* is a constant and $u(n)$ is the unit step sequence.

- Consequently, the ratio $(\lambda_{\min}/\lambda_{\max})$ (called the eigenvalue spread) determines the convergence rate. If $(\lambda_{\min}/\lambda_{\max})$ is much smaller than unity, the convergence will be very slow. If $(\lambda_{\min}/\lambda_{\max})$ is close to unity, the convergence rate of the algorithm is fast.

比值接近于1，那么收敛较快

- It is also possible to perform a mean square convergence analysis of the LMS algorithm to determine its mean square error as a function of the stepsize. As it is rather involved, we only summary the results here:

The total MSE at the output of the adaptive filter is

$$MSE = MSE_{\min} + J_{\mu}, \qquad (5.12)$$

where $MSE_{\min}$ is the minimum MSE given by (2.8), and J_μ is called the excess error due to the noisy gradient estimate and is given by

$$J_\mu = \mu^2 MSE_{\min} \sum_{k=0}^{L-1} \frac{\lambda_k^2}{1-(1-\mu\lambda_k)^2}. \qquad (5.13)$$

Also, when the LMS algorithm is used to track slowly time-variant signal statistics, there will be an additional lagging error J_l due to its use of the estimated gradient.

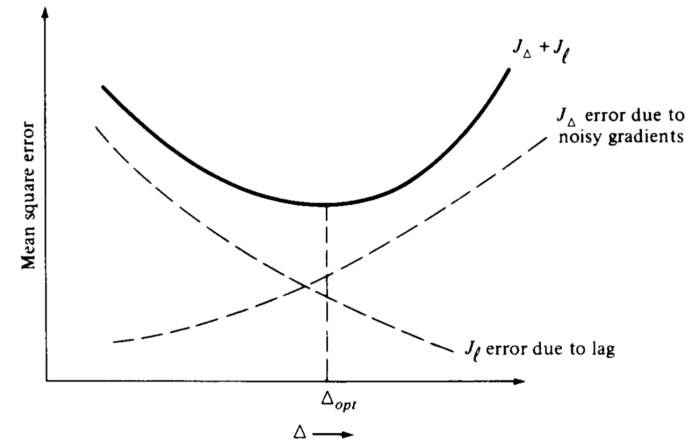


FIGURE 6.20 Excess mean-square error J_Δ and lag error J_ℓ as a function of the step size Δ.

Gradient noise increases with decreasing stepsize, while the lag error increases with decreasing stepsize (bias-variance tradeoff problem).

6. COMPARISON

Convergence Rate

The convergence rate of the RLS lattice-ladder algorithm is almost the same as the RLS direct-form FIR filter structure. The gradient lattice algorithm is slightly inferior but is considerably faster than the LMS when the eigenvalue spread of the autocorrelation matrix is large.

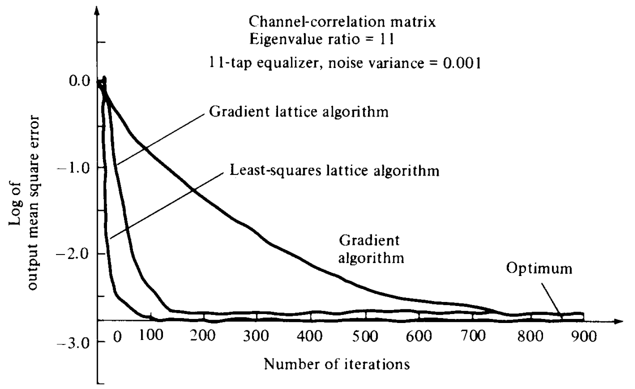


FIGURE 6.26 Learning curves for RLS lattice, gradient lattice, and LMS algorithm for adaptive equalizer of length $M = 11$. (From *Digital Communications* by John G. Proakis. © 1989 by McGraw-Hill Book Company. Reprinted with permission of the publisher.)

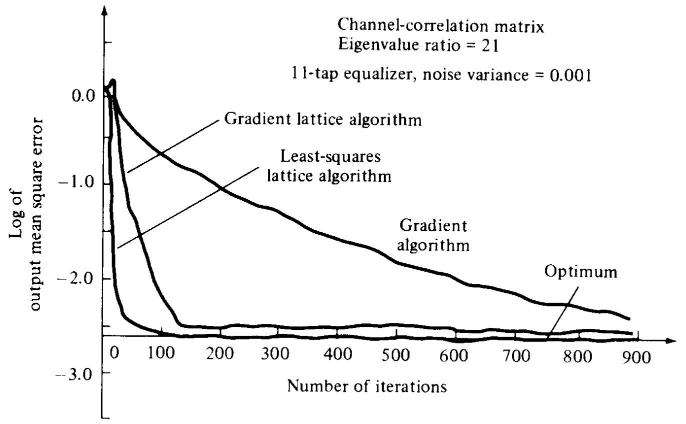


FIGURE 6.27 Learning curves for RLS lattice, gradient lattice and LMS algorithms for adaptive equalizer of length $M = 11$. (From *Digital Communications* by John G. Proakis. © 1989 by McGraw-Hill Book Company. Reprinted with permission of the publisher.)

Computational Requirements

LMS最少的计算量

- The LMS algorithm requires the smallest number of computations.
- The fast RLS algorithms, numerically unstable, are the most efficient among the RLS algorithms, closely followed by the gradient lattice algorithm, then the RLS lattice algorithms, and finally, the square-root algorithms (more stable than basic RLS with complexity $O(L^2)$.

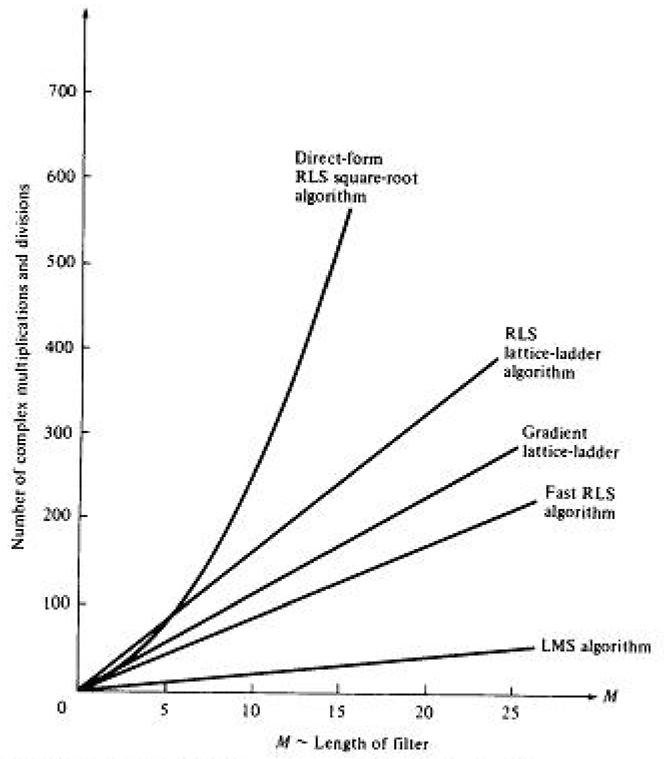


FIGURE 6.28 Computational complexity of adaptive filter algorithms.

Numerical Properties

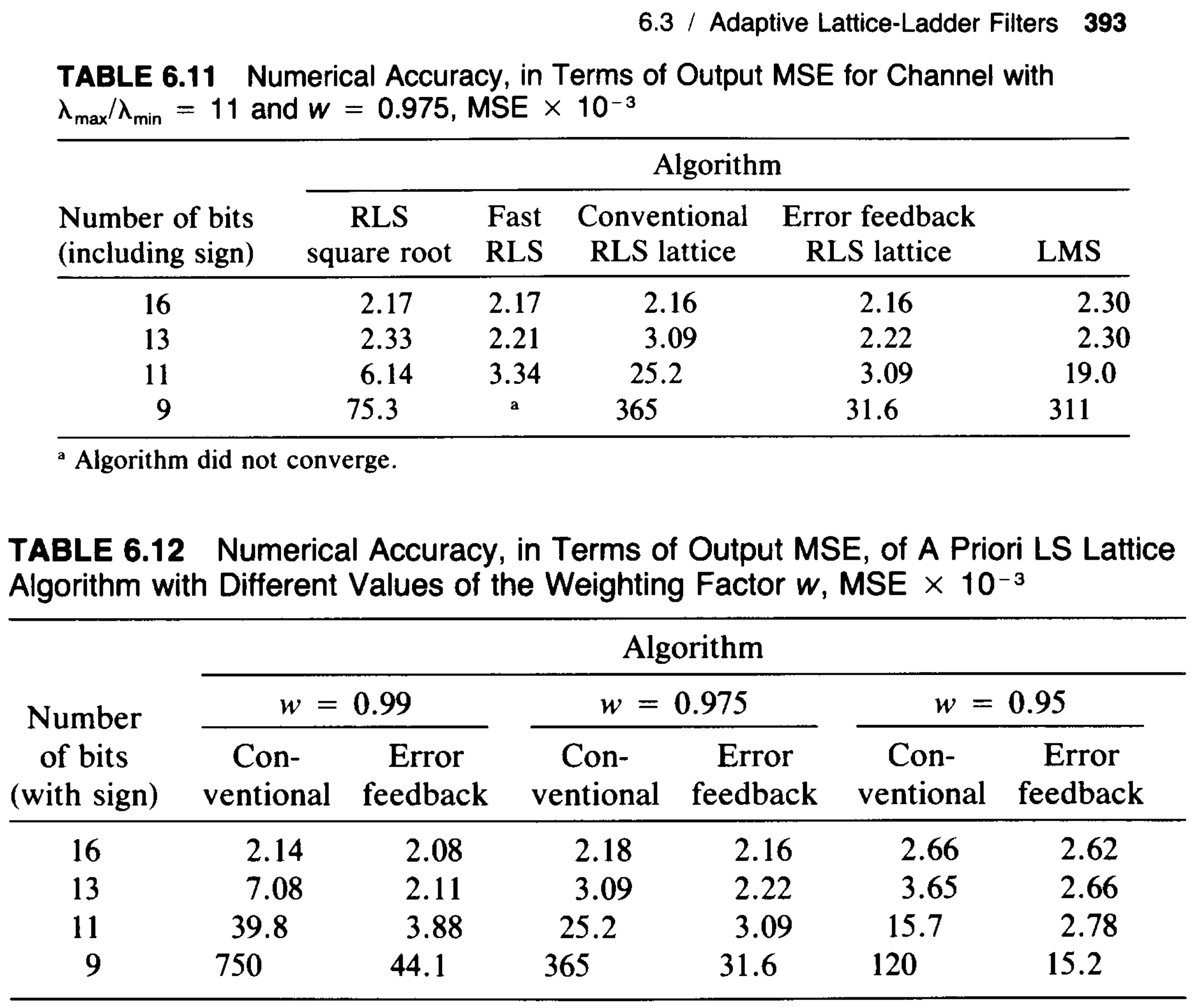


TABLE 6.11 Numerical Accuracy, in Terms of Output MSE for Channel with $\lambda_{max}/\lambda_{min} = 11$ and $w = 0.975$, MSE $\times 10^{-3}$

	Algorithm				
Number of bits (including sign)	RLS square root	Fast RLS	Conventional RLS lattice	Error feedback RLS lattice	LMS
16	2.17	2.17	2.16	2.16	2.30
13	2.33	2.21	3.09	2.22	2.30
11	6.14	3.34	25.2	3.09	19.0
9	75.3	[a]	365	31.6	311

[a] Algorithm did not converge.

TABLE 6.12 Numerical Accuracy, in Terms of Output MSE, of A Priori LS Lattice Algorithm with Different Values of the Weighting Factor w, MSE $\times 10^{-3}$

	Algorithm					
	$w = 0.99$		$w = 0.975$		$w = 0.95$	
Number of bits (with sign)	Con-ventional	Error feedback	Con-ventional	Error feedback	Con-ventional	Error feedback
16	2.14	2.08	2.18	2.16	2.66	2.62
13	7.08	2.11	3.09	2.22	3.65	2.66
11	39.8	3.88	25.2	3.09	15.7	2.78
9	750	44.1	365	31.6	120	15.2

TABLE 6.4 Numerical Accuracy of FIR Adaptive Filtering Algorithms (Least-Squares Error $\times 10^{-3}$)

	Algorithm		
Number of bits (including sign)	RLS square root	Fast RLS	LMS
16	2.17	2.17	2.30
13	2.33	2.21	2.30
11	6.14	3.34	19.0
10	17.6	[a]	77.2
9	75.3	[a]	311.0
8	[a]	[a]	1170.0

[a] Algorithm does not converge to optimum coefficients.