

Adaptive Filters

By:

Douglas L. Jones

Adaptive Filters

By:

Douglas L. Jones

Online:

<<http://cnx.org/content/col10280/1.1/>>

C O N N E X I O N S

Rice University, Houston, Texas

©2008 Douglas L. Jones

This selection and arrangement of content is licensed under the Creative Commons Attribution License:
<http://creativecommons.org/licenses/by/1.0>

Table of Contents

Introduction to Adaptive Filters	1
1 Wiener Filter Algorithm	
1.1 Discrete-Time, Causal Wiener Filter	3
1.2 Practical Issues in Wiener Filter Implementation	6
1.3 Quadratic Minimization and Gradient Descent	7
Solutions	9
2 The LMS Adaptive Filter Algorithm	
2.1 The LMS Adaptive Filter Algorithm	11
2.2 First Order Convergence Analysis of the LMS Algorithm	13
2.3 Second-order Convergence Analysis of the LMS Algorithm and Misadjustment Error	16
Solutions	??
3 Applications of Adaptive Filters	
3.1 Applications of Adaptive Filters	19
3.2 Adaptive System Identification	19
3.3 Adaptive Equalization	20
3.4 Adaptive Interference (Noise) Cancellation	22
3.5 Adaptive Echo Cancellation	25
Solutions	??
4 Other Adaptive Filter Algorithms	
4.1 Beyond LMS: an overview of other adaptive filter algorithms	29
4.2 Adaptive IIR filters	30
4.3 The Constant-Modulus Algorithm and the Property-Restoral Principle	31
4.4 Complex LMS	32
4.5 Normalized LMS	33
Solutions	??
5 Summary of Adaptive Filtering Methods	35
Bibliography	36
Index	38
Attributions	39

Introduction to Adaptive Filters¹

In many applications requiring filtering, the necessary frequency response may not be known beforehand, or it may vary with time. (Example; suppression of engine harmonics in a car stereo.) In such applications, an adaptive filter which can automatically design itself and which can track system variations in time is extremely useful. Adaptive filters are used extensively in a wide variety of applications, particularly in telecommunications.

Outline of adaptive filter material

1. **Wiener Filters** - L_2 optimal (FIR) filter design in a statistical context
2. **LMS algorithm** - simplest and by-far-the-most-commonly-used adaptive filter algorithm
3. **Stability and performance of the LMS algorithm** - When and how well it works
4. **Applications of adaptive filters** - Overview of important applications
5. **Introduction to advanced adaptive filter algorithms** - Techniques for special situations or faster convergence

¹This content is available online at <<http://cnx.org/content/m11535/1.3/>>.

Chapter 1

Wiener Filter Algorithm

1.1 Discrete-Time, Causal Wiener Filter¹

Stochastic L_2 optimal (least squares) FIR filter design problem: Given a wide-sense stationary (WSS) input signal x_k and desired signal d_k (WSS $\Leftrightarrow E[y_k] = E[y_{k+d}], r_{yz}(l) = E[y_k z_{k+l}], \forall k, l : (r_{yy}(0) < \infty)$)

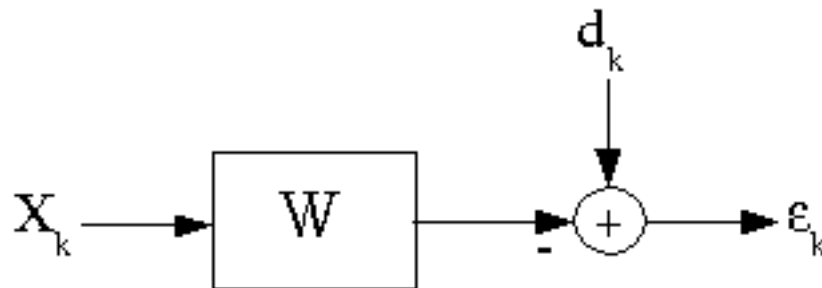


Figure 1.1

The Wiener filter is the linear, time-invariant filter minimizing $E[\epsilon^2]$, the variance of the error.

As posed, this problem seems slightly silly, since d_k is already available! However, this idea is useful in a wide variety of applications.

Example 1.1
active suspension system design

¹This content is available online at <<http://cnx.org/content/m11825/1.1/>>.

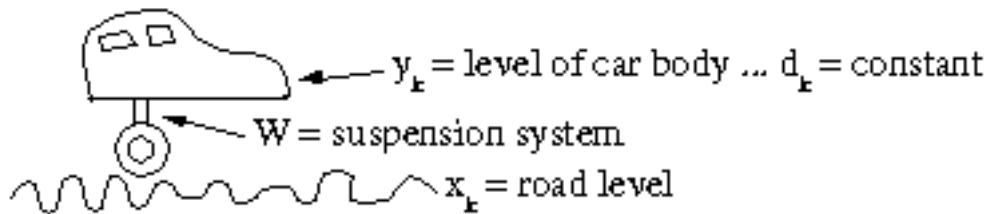


Figure 1.2

NOTE: optimal system may change with different road conditions or mass in car, so an **adaptive** system might be desirable.

Example 1.2

System identification (radar, non-destructive testing, adaptive control systems)

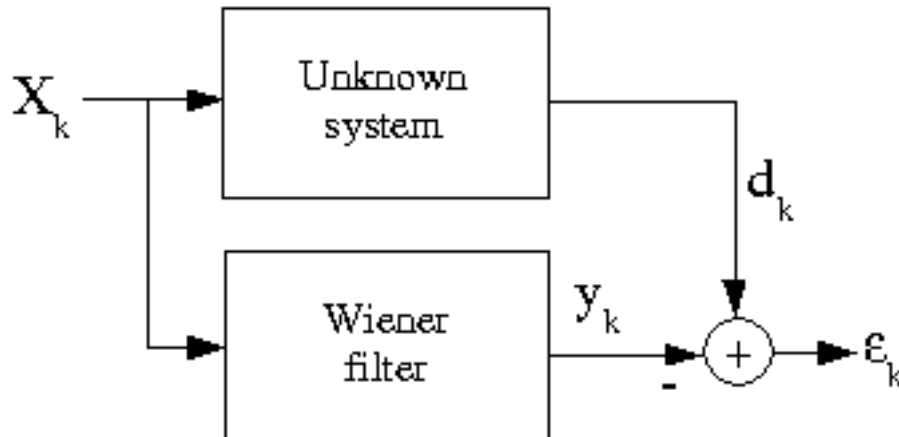


Figure 1.3

Exercise 1.1

Usually one desires that the input signal x_k be "persistently exciting," which, among other things, implies non-zero energy in all frequency bands. Why is this desirable?

1.1.1 Determining the optimal length-N causal FIR Wiener filter

NOTE: for convenience, we will analyze only the causal, real-data case; extensions are straightforward.

$$y_k = \sum_{l=0}^{M-1} (w_l x_{k-l})$$

$$\underset{w_l}{\operatorname{argmin}} E[\epsilon^2] = E[(d_k - y_k)^2] = E\left[\left(d_k - \sum_{l=0}^{M-1} (w_l x_{k-l})\right)^2\right] = E[d_k^2] - 2 \sum_{l=0}^{M-1} (w_l E[d_k x_{k-l}]) + \sum_{l=0}^{M-1} \left(\sum_{m=0}^{M-1} ((w_l w_m E[x_{k-l} x_{k-m}]))\right)$$

$$E[\epsilon^2] = r_{dd}(0) - 2 \sum_{l=0}^{M-1} (w_l r_{dx}(l)) + \sum_{l=0}^{M-1} \left(\sum_{m=0}^{M-1} (w_l w_m r_{xx}(l-m))\right)$$

where

$$r_{dd}(0) = E[d_k^2]$$

$$r_{dx}(l) = E[d_k x_{k-l}]$$

$$r_{xx}(l-m) = E[x_k x_{k+l-m}]$$

This can be written in matrix form as

$$E[\epsilon^2] = r_{dd}(0) - 2\mathbf{P}\mathbf{W}^T + \mathbf{W}^T R \mathbf{W}$$

where

$$\mathbf{P} = \begin{pmatrix} r_{dx}(0) \\ r_{dx}(1) \\ \vdots \\ r_{dx}(M-1) \end{pmatrix}$$

$$R = \begin{pmatrix} r_{xx}(0) & r_{xx}(1) & \dots & \dots & r_{xx}(M-1) \\ r_{xx}(1) & r_{xx}(0) & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & r_{xx}(0) & r_{xx}(1) \\ r_{xx}(M-1) & \dots & \dots & r_{xx}(1) & r_{xx}(0) \end{pmatrix}$$

To solve for the optimum filter, compute the gradient with respect to the top weights vector W

$$\left(\nabla \doteq \begin{pmatrix} \frac{\partial}{\partial w_0} (\epsilon^2) \\ \frac{\partial}{\partial w_1} (\epsilon^2) \\ \vdots \\ \frac{\partial}{\partial w_{M-1}} (\epsilon^2) \end{pmatrix} \right)$$

$$\nabla = -(2\mathbf{P}) + 2R\mathbf{W}$$

(recall $\frac{d}{d\mathbf{W}} (\mathbf{A}^T \mathbf{W}) = \mathbf{A}^T$, $\frac{d}{d\mathbf{W}} (\mathbf{W} M \mathbf{W}) = 2M\mathbf{W}$ for symmetric M) setting the gradient equal to zero \Rightarrow

$$(W_{\text{opt}} R = \mathbf{P} \Rightarrow W_{\text{opt}} = R^{-1} \mathbf{P})$$

Since R is a correlation matrix, it must be non-negative definite, so this is a minimizer. For R positive definite, the minimizer is unique.

1.2 Practical Issues in Wiener Filter Implementation²

The Wiener filter, $W_{\text{opt}} = R^{-1}\mathbf{P}$, is ideal for many applications. But several issues must be addressed to use it in practice.

Exercise 1.2 *(Solution on p. 9.)*

In practice one usually won't know exactly the statistics of x_k and d_k (i.e. R and \mathbf{P}) needed to compute the Wiener filter.

How do we surmount this problem?

Exercise 1.3 *(Solution on p. 9.)*

In many applications, the statistics of x_k , d_k vary slowly with time.

How does one develop an **adaptive** system which tracks these changes over time to keep the system near optimal at all times?

Exercise 1.4 *(Solution on p. 9.)*

How can $\widehat{r_{xx}^k}(l)$ be computed efficiently?

Exercise 1.5

how does one choose N ?

1.2.1 Tradeoffs

Larger $N \rightarrow$ more accurate estimates of the correlation values \rightarrow better \hat{W}_{opt} . However, larger N leads to slower adaptation.

NOTE: The success of adaptive systems depends on x , d being roughly stationary over at least N samples, $N > M$. That is, all adaptive filtering algorithms require that the underlying system varies slowly with respect to the sampling rate and the filter length (although they can tolerate occasional step discontinuities in the underlying system).

1.2.2 Computational Considerations

As presented here, an adaptive filter requires computing a matrix inverse at each sample. Actually, since the matrix R is Toeplitz, the linear system of equations can be solved with $O(M^2)$ computations using Levinson's algorithm, where M is the filter length. However, in many applications this may be too expensive, especially since computing the filter output itself requires $O(M)$ computations. There are two main approaches to resolving the computation problem

1. Take advantage of the fact that R^{k+1} is only slightly changed from R^k to reduce the computation to $O(M)$; these algorithms are called Fast Recursive Least Squares algorithms; all methods proposed so far have stability problems and are dangerous to use.
2. Find a different approach to solving the optimization problem that doesn't require explicit inversion of the correlation matrix.

NOTE: Adaptive algorithms involving the correlation matrix are called **Recursive least Squares** (RLS) algorithms. Historically, they were developed after the LMS algorithm, which is the simplest and most widely used approach $O(M)$. $O(M^2)$ RLS algorithms are used in applications requiring very fast adaptation.

²This content is available online at <http://cnx.org/content/m11824/1.1/>.

1.3 Quadratic Minimization and Gradient Descent³

1.3.1 Quadratic minimization problems

The least squares optimal filter design problem is quadratic in the filter coefficients:

$$E[\epsilon^2] = r_{\text{dd}}(0) - 2\mathbf{P}^T\mathbf{W} + \mathbf{W}^T R \mathbf{W}$$

If R is positive definite, the error surface $E[\epsilon^2](w_0, w_1, \dots, w_{M-1})$ is a unimodal "bowl" in \mathbb{R}^N .

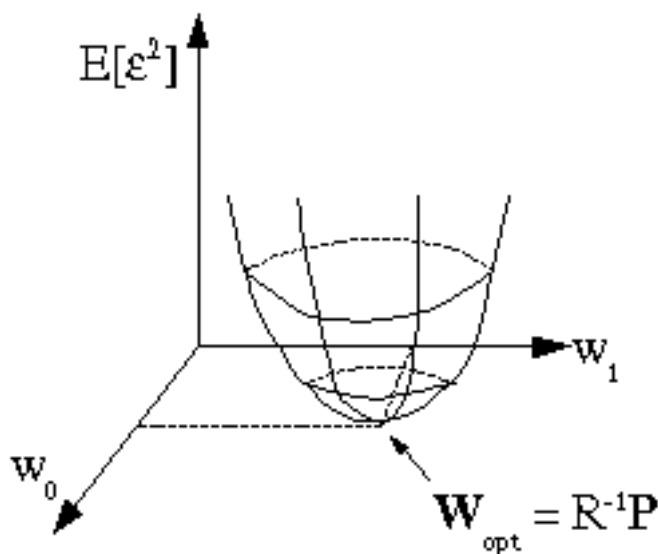


Figure 1.4

The problem is to find the bottom of the bowl. In an adaptive filter context, the shape and bottom of the bowl may drift slowly with time; hopefully slow enough that the adaptive algorithm can track it.

For a quadratic error surface, the bottom of the bowl can be found in one step by computing $\mathbf{R}^{-1}\mathbf{P}$. Most modern nonlinear optimization methods (which are used, for example, to solve the L^P optimal IIR filter design problem!) locally approximate a nonlinear function with a second-order (quadratic) Taylor series approximation and step to the bottom of this quadratic approximation on each iteration. However, an older and simpler approach to nonlinear optimization exists, based on **gradient descent**.

³This content is available online at <http://cnx.org/content/m11826/1.2/>.

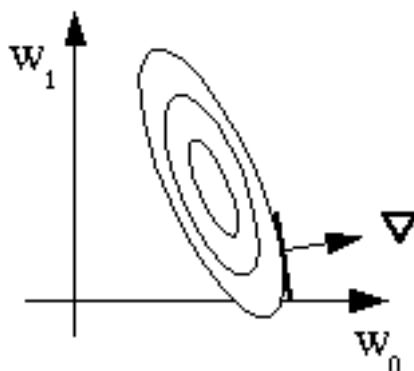
Contour plot of ε -squared

Figure 1.5

The idea is to iteratively find the minimizer by computing the gradient of the error function: $E\nabla = \frac{\partial}{\partial w_i} (E[\epsilon^2])$. The gradient is a vector in \mathbb{R}^M pointing in the steepest uphill direction on the error surface at a given point \mathbf{W}^i , with ∇ having a magnitude proportional to the slope of the error surface in this steepest direction.

By updating the coefficient vector by taking a step **opposite** the gradient direction: $\mathbf{W}^{i+1} = \mathbf{W}^i - \mu\nabla^i$, we go (locally) "downhill" in the steepest direction, which seems to be a sensible way to iteratively solve a nonlinear optimization problem. The performance obviously depends on μ ; if μ is too large, the iterations could bounce back and forth up out of the bowl. However, if μ is too small, it could take many iterations to approach the bottom. We will determine criteria for choosing μ later.

In summary, the gradient descent algorithm for solving the Wiener filter problem is:

Guess W^0

do $i = 1, \infty$

$$\nabla^i = -(2P) + 2RW^i$$

$$W^{i+1} = W^i - \mu\nabla^i$$

repeat

$$W_{\text{opt}} = W^\infty$$

The gradient descent idea is used in the LMS adaptive filter algorithm. As presented, this algorithm costs $O(M^2)$ computations per iteration and doesn't appear very attractive, but LMS only requires $O(M)$ computations and is stable, so it is very attractive when computation is an issue, even though it converges more slowly than the RLS algorithms we have discussed so far.

Solutions to Exercises in Chapter 1

Solution to Exercise 1.2 (p. 6)

Estimate the statistics

$$\widehat{r_{xx}}(l) \approx \frac{1}{N} \sum_{k=0}^{N-1} (x_k x_{k+l})$$

$$\widehat{r_{xd}}(l) \approx \frac{1}{N} \sum_{k=0}^{N-1} (d_k x_{k-l})$$

then solve $\hat{W}_{\text{opt}} = \widehat{R}^{-1} = \hat{P}$

Solution to Exercise 1.3 (p. 6)

Use short-time windowed estimates of the correlation functions.

NOTE:

$$\left(\widehat{r_{xx}}(l)\right)^k = \frac{1}{N} \sum_{m=0}^{N-1} (x_{k-m} x_{k-m-l})$$

$$\left(\widehat{r_{dx}}(l)\right)^k = \frac{1}{N} \sum_{m=0}^{N-1} (x_{k-m-l} d_{k-m})$$

and $W_{\text{opt}}^k \approx \left(\hat{R}_k\right)^{-1} \hat{P}_k$

Solution to Exercise 1.4 (p. 6)

Recursively!

$$r_{xx}^k(l) = r_{xx}^{k-1}(l) + x_k x_{k-l} - x_{k-N} x_{k-N-l}$$

This is critically stable, so people usually do

$$(1 - \alpha) r_{xx}^k(l) = \alpha r_{xx}^{k-1}(l) + x_k x_{k-l}$$

Chapter 2

The LMS Adaptive Filter Algorithm

2.1 The LMS Adaptive Filter Algorithm¹

Recall the Weiner filter problem

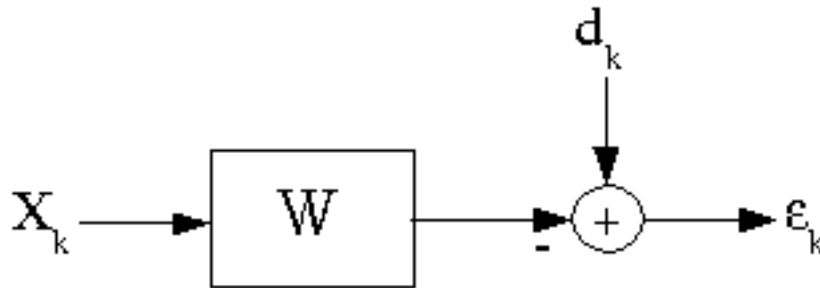


Figure 2.1

$\{x_k\}, \{d_k\}$ jointly wide sense stationary
Find W minimizing $E[\epsilon_k^2]$

$$\epsilon_k = d_k - y_k = d_k - \sum_{i=0}^{M-1} (w_i x_{k-i}) = d_k - X^k T W^k$$

$$X^k = \begin{pmatrix} x_k \\ x_{k-1} \\ \vdots \\ x_{k-M+1} \end{pmatrix}$$

¹This content is available online at <<http://cnx.org/content/m11829/1.1/>>.

$$W^k = \begin{pmatrix} w_0^k \\ w_1^k \\ \vdots \\ w_{M-1}^k \end{pmatrix}$$

The superscript denotes absolute time, and the subscript denotes time or a vector index. the solution can be found by setting the gradient = 0

$$\begin{aligned} \nabla^k &= \frac{\partial}{\partial W} (E [\epsilon_k^2]) \\ &= E [2\epsilon_k (-X^k)] \\ &= E [-2 (d_k - X^{kT} W_k) X^k] \\ &= -(2E [d_k X^k]) + (E [X^{kT}]) W \\ &= -2P + 2RW \\ &\Rightarrow (W_{\text{opt}} = R^{-1}P) \end{aligned} \tag{2.1}$$

Alternatively, W_{opt} can be found iteratively using a gradient descent technique

$$W^{k+1} = W^k - \mu \nabla^k$$

In practice, we don't know R and P exactly, and in an adaptive context they may be slowly varying with time.

To find the (approximate) Wiener filter, some approximations are necessary. As always, the key is to make the **right** approximations!

NOTE: Approximate R and P : \Rightarrow RLS methods, as discussed last time.

NOTE: Approximate the gradient!

$$\nabla^k = \frac{\partial}{\partial W} (E [\epsilon_k^2])$$

Note that ϵ_k^2 itself is a very noisy approximation to $E [\epsilon_k^2]$. We can get a noisy approximation to the gradient by finding the gradient of ϵ_k^2 ! Widrow and Hoff first published the LMS algorithm, based on this clever idea, in 1960.

$$\widehat{\nabla}^k = \frac{\partial}{\partial W} (\epsilon_k^2) = 2\epsilon_k \frac{\partial}{\partial W} (d_k - W^{kT} X^k) = 2\epsilon_k (-X^k) = -(2\epsilon_k X^k)$$

This yields the LMS adaptive filter algorithm

Example 2.1: The LMS Adaptive Filter Algorithm

1. $y_k = W^{kT} X^k = \sum_{i=0}^{M-1} (w_i^k x_{k-i})$
2. $\epsilon_k = d_k - y_k$
3. $W^{k+1} = W^k - \mu \widehat{\nabla}^k = W^k - \mu (-2\epsilon_k X^k) = W^k + 2\mu \epsilon_k X^k$ ($w_i^{k+1} = w_i^k + 2\mu \epsilon_k x_{k-i}$)

The LMS algorithm is often called a **stochastic gradient** algorithm, since $\widehat{\nabla}^k$ is a noisy gradient. This is **by far** the most commonly used adaptive filtering algorithm, because

1. it was the first
2. it is very simple
3. in practice it works well (except that sometimes it converges slowly)
4. it requires relatively little computation
5. it updates the tap weights every sample, so it continually adapts the filter
6. it tracks slow changes in the signal statistics well

2.1.1 Computational Cost of LMS

To Compute \Rightarrow	y_k	ϵ_k	W^{k+1}	= Total
multiplies	M	0	$M + 1$	$2M + 1$
adds	$M - 1$	1	M	$2M$

Table 2.1

So the LMS algorithm is $O(M)$ per sample. In fact, it is nicely balanced in that the filter computation and the adaptation require the same amount of computation.

Note that the parameter μ plays a very important role in the LMS algorithm. It can also be varied with time, but usually a constant μ ("convergence weight factor") is used, chosen after experimentation for a given application.

2.1.1.1 Tradeoffs

large μ : fast convergence, fast adaptivity

small μ : accurate $W \rightarrow$ less misadjustment error, stability

2.2 First Order Convergence Analysis of the LMS Algorithm²

2.2.1 Analysis of the LMS algorithm

It is important to analyze the LMS algorithm to determine under what conditions it is stable, whether or not it converges to the Wiener solution, to determine how quickly it converges, how much degradation is suffered due to the noisy gradient, etc. In particular, we need to know how to choose the parameter μ .

2.2.1.1 Mean of W

does W^k , $k \rightarrow \infty$ approach the Wiener solution? (since W^k is always somewhat random in the approximate gradient-based LMS algorithm, we ask whether the expected value of the filter coefficients converge to the Wiener solution)

$$\begin{aligned}
 E[W^{k+1}] &= \overline{W^{k+1}} \\
 &= E[W^k + 2\mu\epsilon_k X^k] \\
 &= \overline{W^k} + 2\mu E[d_k X^k] + 2\mu E\left[-\left((W^{kT} X^k) X^k\right)\right] \\
 &= \overline{W^k} + 2\mu P + \left(-\left(2\mu E\left[(W^{kT} X^k) X^k\right]\right)\right)
 \end{aligned} \tag{2.2}$$

2.2.1.1.1 Patently False Assumption

X^k and X^{k-i} , X^k and d^{k-i} , and d_k and d_{k-i} are statistically independent, $i \neq 0$. This assumption is obviously false, since X^{k-1} is the same as X^k except for shifting down the vector elements one place and adding one new sample. We make this assumption because otherwise it becomes extremely difficult to analyze the LMS algorithm. (First good analysis not making this assumption: *Macchi and Eweda*[6]) Many simulations and much practical experience has shown that the results one obtains with analyses based on the patently false assumption above are quite accurate in most situations

²This content is available online at <<http://cnx.org/content/m11830/1.1/>>.

With the independence assumption, W^k (which depends only on previous X^{k-i} , d^{k-i}) is statistically independent of X^k , and we can simplify $E \left[\left(W^{kT} X^k \right) X^k \right]$

Now $\left(W^{kT} X^k \right) X^k$ is a vector, and

$$\begin{aligned}
 E \left[\left(W^{kT} X^k \right) X^k \right] &= E \left[\begin{pmatrix} \vdots \\ \left(\sum_{i=0}^{M-1} (w_i^k x_{k-i}) \right) x_{k-j} \\ \vdots \end{pmatrix} \right] \\
 &= \begin{pmatrix} \vdots \\ \sum_{i=0}^{M-1} (E [w_i^k x_{k-i} x_{k-j}]) \\ \vdots \end{pmatrix} \\
 &= \begin{pmatrix} \vdots \\ \sum_{i=0}^{M-1} ((w_i^k) E [x_{k-i} x_{k-j}]) \\ \vdots \end{pmatrix} \\
 &= \begin{pmatrix} \vdots \\ \sum_{i=0}^{M-1} (\overline{w_i^k} r_{xx}(i-j)) \\ \vdots \end{pmatrix} \\
 &= \overline{RW^k}
 \end{aligned} \tag{2.3}$$

where $R = E [X^k X^{kT}]$ is the data correlation matrix.

Putting this back into our equation

$$\begin{aligned}
 \overline{W^{k+1}} &= \overline{W^k} + 2\mu P + \left(- \left(2\mu R \overline{W^k} \right) \right) \\
 &= (I - 2\mu R) \overline{W^k} + 2\mu P
 \end{aligned} \tag{2.4}$$

Now if $\overline{W^{k \rightarrow \infty}}$ converges to a vector of finite magnitude ("convergence in the mean"), what does it converge to?

If $\overline{W^k}$ converges, then as $k \rightarrow \infty$, $\overline{W^{k+1}} \approx \overline{W^k}$, and

$$\overline{W^\infty} = (I - 2\mu R) \overline{W^\infty} + 2\mu P$$

$$2\mu R \overline{W^\infty} = 2\mu P$$

$$R \overline{W^\infty} = P$$

or

$$\overline{W_{\text{opt}}} = R^{-1} P$$

the Wiener solution!

So the LMS algorithm, if it converges, gives filter coefficients which on average are the Wiener coefficients! This is, of course, a desirable result.

2.2.1.2 First-order stability

But does \overline{W}^k converge, or under what conditions?

Let's rewrite the analysis in term of \overline{V}^k , the "mean coefficient error vector" $\overline{V}^k = \overline{W}^k - W_{\text{opt}}$, where W_{opt} is the Wiener filter

$$\overline{W}^{k+1} = \overline{W}^k - 2\mu R \overline{W}^k + 2\mu P$$

$$\overline{W}^{k+1} - W_{\text{opt}} = \overline{W}^k - W_{\text{opt}} + \left(- \left(2\mu R \overline{W}^k \right) \right) + 2\mu R W_{\text{opt}} - 2\mu R W_{\text{opt}} + 2\mu P$$

$$\overline{V}^{k+1} = \overline{V}^k - 2\mu R \overline{V}^k + (-2\mu R W_{\text{opt}}) + 2\mu P$$

Now $W_{\text{opt}} = R^{-1}$, so

$$\overline{V}^{k+1} = \overline{V}^k - 2\mu R \overline{V}^k + (-2\mu R R^{-1} P) + 2\mu P = (I - 2\mu R) \overline{V}^k$$

We wish to know under what conditions $\overline{V}^{k \rightarrow \infty} \rightarrow \overline{0}$?

2.2.1.2.1 Linear Algebra Fact

Since R is positive definite, real, and symmetric, all the eigenvalues are real and positive. Also, we can write R as $(Q^{-1} \Lambda Q)$, where Λ is a diagonal matrix with diagonal entries λ_i equal to the eigenvalues of R , and Q is a unitary matrix with rows equal to the eigenvectors corresponding to the eigenvalues of R .

Using this fact,

$$V^{k+1} = (I - 2\mu (Q^{-1} \Lambda Q)) V^k$$

multiplying both sides through on the left by Q : we get

$$Q \overline{V}^{k+1} = (Q - 2\mu \Lambda Q) \overline{V}^k = (1 - 2\mu \Lambda) Q \overline{V}^k$$

Let $V' = QV$:

$$V'^{k+1} = (1 - 2\mu \Lambda) V'^k$$

Note that V' is simply V in a rotated coordinate set in \mathbb{R}^m , so convergence of V' implies convergence of V .

Since $1 - 2\mu \Lambda$ is diagonal, all elements of V' evolve independently of each other. Convergence (stability) boils down to whether all M of these scalar, first-order difference equations are stable, and thus $\rightarrow 0$.

$$\forall i, i = [1, 2, \dots, M] : (V_i'^{k+1} = (1 - 2\mu \lambda_i) V_i'^k)$$

These equations converge to zero if $|1 - 2\mu \lambda_i| < 1$, or $\forall i : (|\mu \lambda_i| < 1)$ μ and λ_i are positive, so we require $\forall i : \left(\mu < \frac{1}{\lambda_i} \right)$ so for convergence in the mean of the LMS adaptive filter, we require

$$\mu < \frac{1}{\lambda_{\max}} \tag{2.5}$$

This is an elegant theoretical result, but in practice, we may not know λ_{\max} , it may be time-varying, and we certainly won't want to compute it. However, another useful mathematical fact comes to the rescue...

$$\text{tr}(R) = \sum_{i=1}^M r_{ii} = \sum_{i=1}^M \lambda_i \geq \lambda_{\max}$$

Since the eigenvalues are all positive and real.

For a correlation matrix, $\forall i, i \in \{1, M\} : (r_{ii} = r(0))$. So $\text{tr}(R) = M r(0) = M E[x_k x_k]$. We can easily estimate $r(0)$ with $O(1)$ computations/sample, so in practice we might require

$$\mu < \frac{1}{M r(0)}$$

as a conservative bound, and perhaps adapt μ accordingly with time.

2.2.1.3 Rate of convergence

Each of the modes decays as

$$(1 - 2\mu\lambda_i)^k$$

NOTE: The **initial** rate of convergence is dominated by the fastest mode $1 - 2\mu\lambda_{\max}$. This is not surprising, since a gradient descent method goes "downhill" in the steepest direction

NOTE: The **final** rate of convergence is dominated by the slowest mode $1 - 2\mu\lambda_{\min}$. For small λ_{\min} , it can take a long time for LMS to converge.

Note that the convergence behavior depends on the data (via R). LMS converges relatively quickly for roughly equal eigenvalues. Unequal eigenvalues slow LMS down a lot.

2.3 Second-order Convergence Analysis of the LMS Algorithm and Misadjustment Error³

Convergence of the mean (first-order analysis) is insufficient to guarantee desirable behavior of the LMS algorithm; the variance could still be infinite. It is important to show that the variance of the filter coefficients is finite, and to determine how close the average squared error is to the minimum possible error using an exact Wiener filter.

$$\begin{aligned} E[\epsilon_k^2] &= E\left[\left(d_k - W^{kT}X^k\right)^2\right] \\ &= E\left[d_k^2 - 2d_kX^{kT}W^k - W^{kT}X^kX^{kT}W^k\right] \\ &= r_{\text{dd}}(0) - 2W^{kT}P + W^{kT}RW^k \end{aligned} \quad (2.6)$$

The minimum error is obtained using the Wiener filter

$$W_{\text{opt}} = R^{-1}P$$

$$\begin{aligned} \epsilon_{\min}^2 &= E[\epsilon^2] \\ &= (r_{\text{dd}}(0) - 2P^TR^{-1}P + P^TR^{-1}RR^{-1}P) \\ &= r_{\text{dd}}(0) - P^TR^{-1}P \end{aligned} \quad (2.7)$$

To analyze the average error in LMS, write (2.6) in terms of $V^k = Q[W - W_{\text{opt}}]$, where $Q^{-1}\Lambda Q = R$

$$\begin{aligned} E[\epsilon_k^2] &= r_{\text{dd}}(0) - 2W^{kT}P + W^{kT}RW^k + -\left(W^{kT}RW_{\text{opt}}\right) - W_{\text{opt}}^TRW^k + \\ &W_{\text{opt}}^TRW_{\text{opt}} + W^{kT}RW_{\text{opt}} + W_{\text{opt}}^TRW^k - W_{\text{opt}}^TRW_{\text{opt}} = r_{\text{dd}}(0) + V^{kT}RV^k - \\ &P^TR^{-1}P = \epsilon_{\min}^2 + V^{kT}RV^k = \epsilon_{\min}^2 + V^{kT}Q^{-1}QRQ^{-1}QV^k = \epsilon_{\min}^2 + V^{kT}\Lambda V^k \end{aligned} \quad (2.8)$$

$$E[\epsilon_k^2] = \epsilon_{\min}^2 + \sum_{j=0}^{N-1} \left(\lambda_j E[v_j^{k2}]\right)$$

So we need to know $E[v_j^{k2}]$, which are the diagonal elements of the covariance matrix of V^k , or $E[V^kV^{kT}]$.

From the LMS update equation

$$W^{k+1} = W^k + 2\mu\epsilon_kX^k$$

³This content is available online at <<http://cnx.org/content/m11831/1.2/>>.

we get

$$V^{k+1} = W^k + 2\mu\epsilon_k QX^k$$

$$\begin{aligned} \mathcal{Y}^{k+1} &= E \left[V^{k+1} V^{k+1T} \right] \\ &= E \left[4\mu^2 \epsilon_k^2 QX^k X^{kT} Q^T \right] \\ &= \mathcal{Y}^k + 2\mu \left(\epsilon_k QX^k V^{kT} \right) + 2\mu \left(\epsilon_k V^k X^{kT} Q^T \right) + 4\mu^2 E \left[\epsilon_k^2 QX^k X^{kT} Q^T \right] \end{aligned} \quad (2.9)$$

Note that

$$\epsilon_k = d_k - W^{kT} X^k = d_k - W_{\text{opt}}^T - V^{kT} QX^k$$

so

$$\begin{aligned} E \left[\epsilon_k QX^k V^{kT} \right] &= E \left[d_k QX^k V^{kT} - W_{\text{opt}}^T X^k QX^k V^{kT} - V^{kT} QX^k V^{kT} \right] \\ &= 0 + 0 - \left(QX^k X^{kT} Q^T V^{kT} \right) \\ &= - \left(QE \left[X^k X^{kT} \right] Q^T E \left[V^k V^{kT} \right] \right) \\ &= - (\Lambda \mathcal{Y}^k) \end{aligned} \quad (2.10)$$

Note that the Patently False independence Assumption was invoked here.

To analyze $E \left[\epsilon_k^2 QX^k X^{kT} Q^T \right]$, we make yet another obviously false assumption that ϵ_k^2 and X^k are statistically independent. This is obviously false, since $\epsilon_k = d_k - W^{kT} X^k$. Otherwise, we get 4th-order terms in X in the product. These can be dealt with, at the expense of a more complicated analysis, if a particular type of distribution (such as Gaussian) is assumed. See, for example *Gardner*[2]. A questionable justification for this assumption is that as $W^k \approx W_{\text{opt}}$, W^k becomes uncorrelated with X^k (if we invoke the original independence assumption), which tends to randomize the error signal relative to X^k . With this assumption,

$$E \left[\epsilon_k^2 QX^k X^{kT} Q^T \right] = E \left[\epsilon_k^2 \right] E \left[QX^k X^{kT} Q^T \right] = E \left[\epsilon_k^2 \right] \Lambda$$

Now

$$\epsilon_k^2 = \epsilon_{\min}^2 + V^{kT} \Lambda V^k$$

so

$$\begin{aligned} E \left[\epsilon_k^2 \right] &= \epsilon_{\min}^2 + E \left[\sum \left(\lambda_j V_j^{k2} \right) \right] \\ &= \epsilon_{\min}^2 + \sum \left(\lambda_j \mathcal{Y}_{jj}^k \right) \end{aligned} \quad (2.11)$$

Thus, (2.9) becomes

$$\mathcal{Y}^{k+1} = (I - 4\mu\Lambda) \mathcal{Y}^k + 4\mu^2 \sum \left(\lambda_j \mathcal{Y}_{jj}^k \Lambda \right) + 4\mu^2 \epsilon_{\min}^2 \Lambda \quad (2.12)$$

Now **if** this system is stable and converges, it converges to $\mathcal{Y}^\infty = \mathcal{Y}^{\infty+1}$

$$\Rightarrow \left(4\mu\Lambda \mathcal{Y}^\infty = 4\mu^2 \left(\sum \left(\lambda_j \mathcal{Y}_{jj} \right) + \epsilon_{\min}^2 \right) \Lambda \right)$$

$$\Rightarrow \left(\mathcal{Y}^\infty = \mu \left(\sum \left(\lambda_j \mathcal{Y}_{jj} \right) + \epsilon_{\min}^2 \right) I \right)$$

So it is a diagonal matrix with all elements on the diagonal equal:

Then

$$\mathcal{Y}_{ii}^\infty = \mu \left(\mathcal{Y}_{ii}^\infty \sum \lambda_j + \epsilon_{\min}^2 \right)$$

$$\mathcal{Y}_{ii}^\infty \left(1 - \mu \sum \lambda_j \right) = \mu \epsilon_{\min}^2$$

$$\mathcal{V}_{ii}^{\infty} = \frac{\mu \epsilon_{\min}^2}{1 - \mu \sum \lambda_j}$$

Thus the error in the LMS adaptive filter after convergence is

$$\begin{aligned} E[\epsilon_{\infty}^2] &= \epsilon_{\min}^2 + E[V'^{\infty} \lambda V'^{\infty}] \\ &= \epsilon_{\min}^2 + \frac{\mu \epsilon_{\min}^2 \sum (\lambda_j)}{1 - \mu \sum \lambda_j} \\ &= \epsilon_{\min}^2 \frac{1}{1 - \mu \sum \lambda_j} \\ &= \epsilon_{\min}^2 \frac{1}{1 - \mu \text{tr}(R)} \\ &= \epsilon_{\min}^2 \frac{1}{1 - \mu r_{xx}(0)N} \end{aligned} \tag{2.13}$$

$$E[\epsilon_{\infty}^2] = \epsilon_{\min}^2 \frac{1}{1 - \mu N \sigma_x^2} \tag{2.14}$$

$1 - \mu N \sigma_x^2$ is called the **misadjustment factor**. Often, one chooses μ to select a desired misadjustment factor, such as an error 10% higher than the Wiener filter error.

2.3.1 2nd-Order Convergence (Stability)

To determine the range for μ for which (2.12) converges, we must determine the μ for which the matrix difference equation converges.

$$\mathcal{V}^{k+1} = (I - 4\mu\Lambda) \mathcal{V}^k + 4\mu^2 \sum (\lambda_j \mathcal{V}_{jj}^k \Lambda) + 4\mu^2 \epsilon_{\min}^2 \Lambda$$

The off-diagonal elements each evolve independently according to $\mathcal{V}_{ij}^{k+1} = 1 - 4\mu\lambda_i \mathcal{V}_{ij}^k$. These terms will decay to zero if $\forall i : (4\mu\lambda_i < 2)$, or $\mu < \frac{1}{2\lambda_{\max}}$

The diagonal terms evolve according to

$$\mathcal{V}_{ii}^{k+1} = (1 - 4\mu\lambda_i) \mathcal{V}_{ii}^k + 4\mu^2 \lambda_i \left(\sum \lambda_j \right) \mathcal{V}_{jj}^k + 4\mu^2 \epsilon_{\min}^2 \lambda_i$$

For the homogeneous equation

$$\mathcal{V}_{ii}^{k+1} = (1 - 4\mu\lambda_i) \mathcal{V}_{ii}^k + 4\mu^2 \lambda_i \left(\sum \lambda_j \right) \mathcal{V}_{jj}^k$$

for $1 - 4\mu\lambda_i$ positive,

$$\mathcal{V}_{ii}^{k+1} \leq (1 - 4\mu\lambda_i) \mathcal{V}_{i\max}^k + 4\mu^2 \lambda_i \left(\sum \lambda_j \right) \mathcal{V}_{j\max}^k = \left(1 - 4\mu\lambda_i + 4\mu^2 \lambda_i \sum \lambda_j \right) \mathcal{V}_{j\max}^k \tag{2.15}$$

\mathcal{V}_{ii}^{k+1} will be strictly less than $\mathcal{V}_{j\max}^k$ for

$$1 - 4\mu\lambda_i + 4\mu^2 \lambda_i \sum \lambda_j < 1$$

or

$$4\mu^2 \lambda_i \sum \lambda_j < 4\mu\lambda_i$$

or

$$\begin{aligned} \mu < \frac{1}{\sum \lambda_j} &= \frac{1}{\text{tr}(R)} \\ &= \frac{1}{N r_{xx}(0)} \\ &= \frac{1}{N \sigma_x^2} \end{aligned} \tag{2.16}$$

This is a more rigorous bound than the first-order bounds. Often engineers choose μ a few times smaller than this, since more rigorous analyses yield a slightly smaller bound. $\mu = \frac{\mu}{3N\sigma_x^2}$ is derived in some analyses assuming Gaussian x_k, d_k .

Chapter 3

Applications of Adaptive Filters

3.1 Applications of Adaptive Filters¹

Adaptive filters can be configured in a surprisingly large variety of ways for application in a large number of applications. The same LMS algorithm can be used to adapt the filter coefficients in most of these configurations; only the "plumbing" changes.

3.2 Adaptive System Identification²

NOTE: To approximate an unknown system (or the behavior of that system) as closely as possible

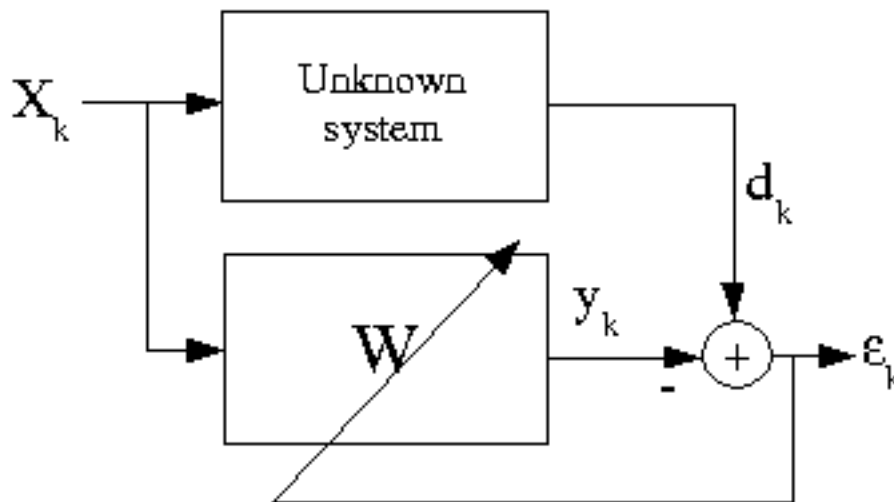


Figure 3.1

¹This content is available online at <<http://cnx.org/content/m11536/1.1/>>.

²This content is available online at <<http://cnx.org/content/m11906/1.1/>>.

The optimal solution is $R^{-1}P = W$

Suppose the unknown system is a causal, linear time-invariant filter:

$$d_k = x_k * h_k = \sum_{i=0}^{\infty} (x_{k-i}h_i)$$

Now

$$\begin{aligned}
 P &= \left(E[d_k x_{k-j}] \right) = \left(E[(\sum_{i=0}^{\infty} (x_{k-i}h_i)) x_{k-j}] \right) = \quad (3.1) \\
 &\left(\sum_{i=0}^{\infty} (h_i E[x_{k-i}x_{k-j}]) \right) = \left(\sum_{i=0}^{\infty} (r_{xx}(j-i)) \right) = \\
 &\left(\begin{array}{cccc|cc} r_{xx}(0) & r(1) & \dots & \dots & r(M-1) & r(M) & r(M+1) & \dots \\ r(1) & r(0) & \ddots & \ddots & \vdots & \vdots & \vdots & \dots \\ r(2) & r(1) & \ddots & \ddots & \vdots & \vdots & \vdots & \dots \\ \vdots & \vdots & \dots & r(0) & r(1) & r(2) & r(3) & \dots \\ r(M-1) & r(M-2) & \dots & r(1) & r(0) & r(1) & r(2) & \dots \end{array} \right) \left(\begin{array}{c} h(0) \\ h(1) \\ h(2) \\ \vdots \end{array} \right)
 \end{aligned}$$

If the adaptive filter H is a length- M FIR filter ($h(m) = h(m+1) = \dots = 0$), this reduces to

$$P = Rh^{-1}$$

and

$$W_{\text{opt}} = R^{-1}P = R^{-1}(Rh) = \mathbf{h}$$

FIR adaptive system identification thus converges in the mean to the corresponding M samples of the impulse response of the unknown system.

3.3 Adaptive Equalization³

NOTE: Design an approximate inverse filter to cancel out as much distortion as possible.

³This content is available online at <<http://cnx.org/content/m11907/1.1/>>.

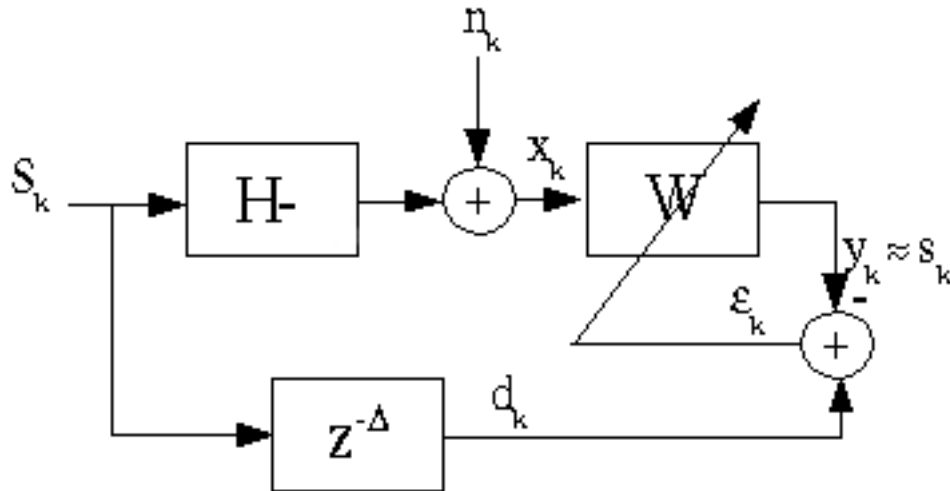


Figure 3.2

In principle, $WH \approx z^{-\Delta}$, or $W \approx \frac{z^{-\Delta}}{H}$, so that the overall response of the top path is approximately $\delta(n - \Delta)$. However, limitations on the form of W (FIR) and the presence of noise cause the equalization to be imperfect.

3.3.1 Important Application

Channel equalization in a digital communication system.



Figure 3.3

If the channel distorts the pulse shape, the matched filter will no longer be matched, intersymbol interference may increase, and the system performance will degrade.

An adaptive filter is often inserted in front of the matched filter to compensate for the channel.

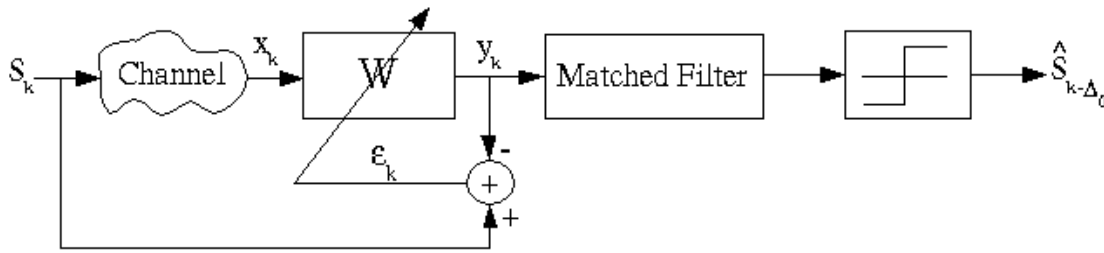


Figure 3.4

This is, of course, unrealizable, since we do not have access to the original transmitted signal, s_k . There are two common solutions to this problem:

1. Periodically broadcast a known **training signal**. The adaptation is switched on only when the training signal is being broadcast and thus s_k is known.
2. Decision-directed feedback: If the overall system is working well, then the output $\hat{s}_{k-\Delta_0}$ should almost always equal $s_{k-\Delta_0}$. We can thus use our received digital communication signal as the desired signal, since it has been cleaned of noise (we hope) by the nonlinear threshold device!

Decision-directed equalizer

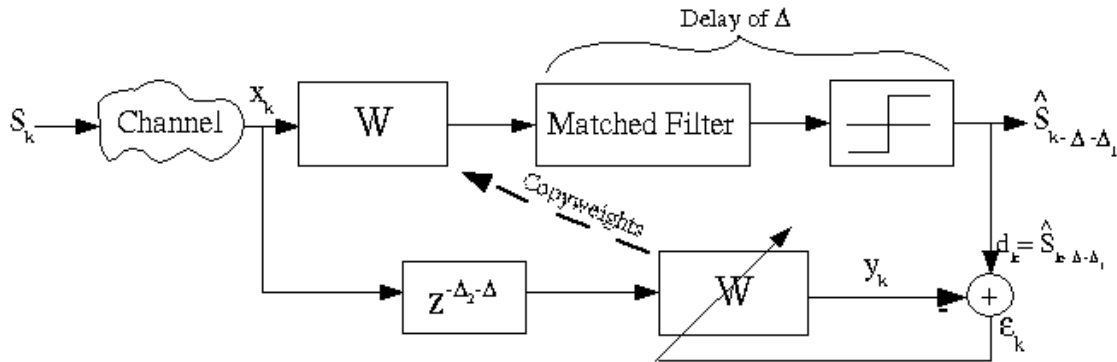


Figure 3.5

As long as the error rate in \hat{s}_k is not too high (say $< 75\%$), this method works. Otherwise, d_k is so inaccurate that the adaptive filter can never find the Wiener solution. This method is widely used in the telephone system and other digital communication networks.

3.4 Adaptive Interference (Noise) Cancellation⁴

NOTE: Automatically eliminate unwanted interference in a signal.

⁴This content is available online at <http://cnx.org/content/m11835/1.1/>.

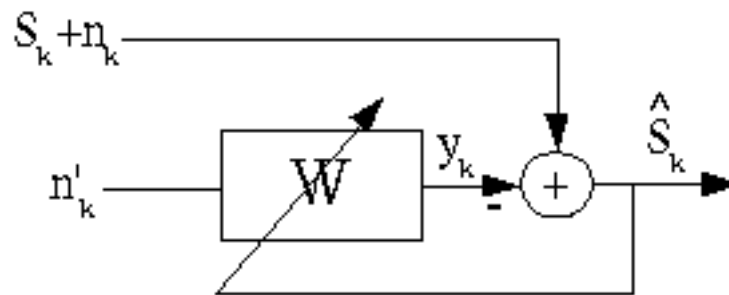


Figure 3.6

The object is to subtract out as much of the noise as possible.

Example 3.1: Engine noise cancellation in automobiles

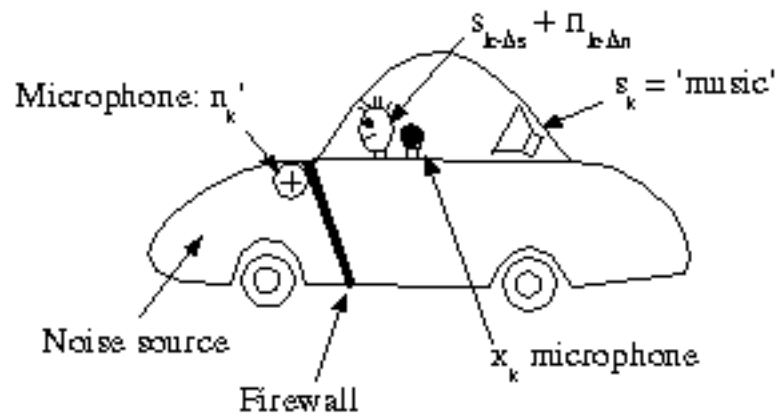


Figure 3.7

The firewall attenuates and filters the noise reaching the listener's ear, so it is not the same as n'_k . There is also a delay due to acoustic propagation in the air. For maximal cancellation, an adaptive filter is thus needed to make n'_k as similar as possible to the delayed n_k .

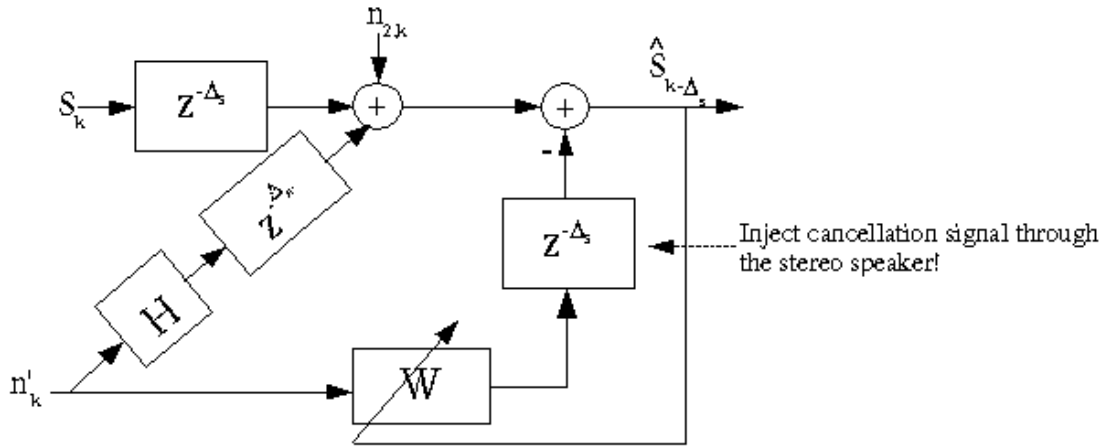


Figure 3.8

Exercise 3.1

What conditions must we impose upon the microphone locations for this to work? (Think causality and physics!)

3.4.1 Analysis of the interference cancellor

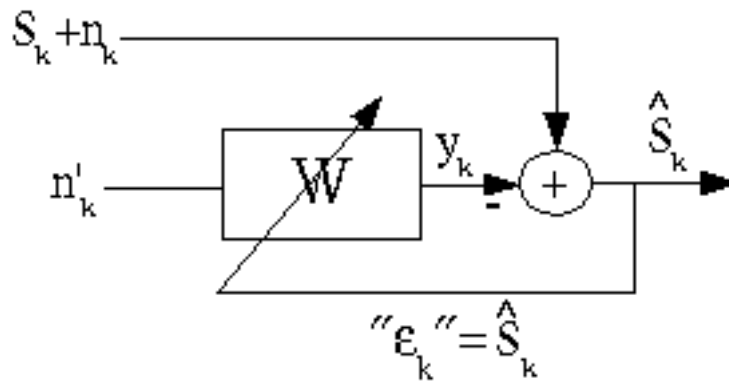


Figure 3.9

$$E[\epsilon_k^2] = E[(s_k + n_k - y_k)^2] = E[s_k^2] + 2E[s_k(n_k - y_k)] + E[(n_k - y_k)^2]$$

We assume s_k , n_k , and n'_k are zero-mean signals, and that s_k is independent of n_k and n'_k . Then

$$E[s_k(n_k - y_k)] = E[s_k]E[n_k - y_k] = 0$$

Echo canceller

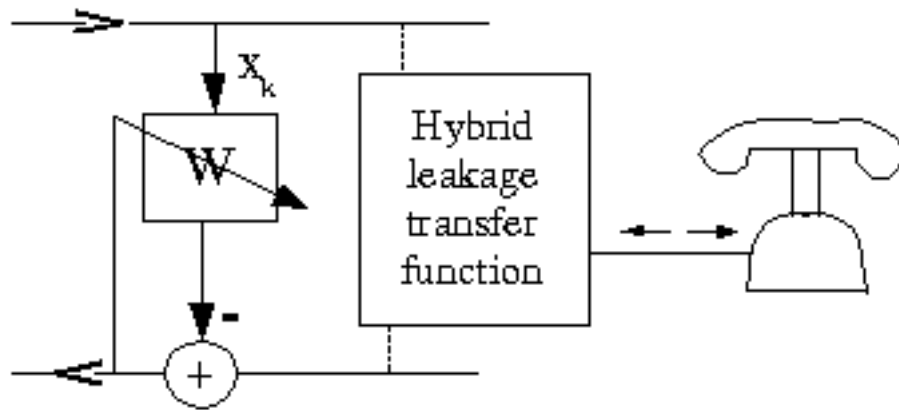


Figure 3.11

Because the input to the adaptive echo canceller contains only the signal from the far end that will echo off of the hybrid, it cancels the echo while passing the near-end signal as desired.

3.5.1 Narrowband interference canceller

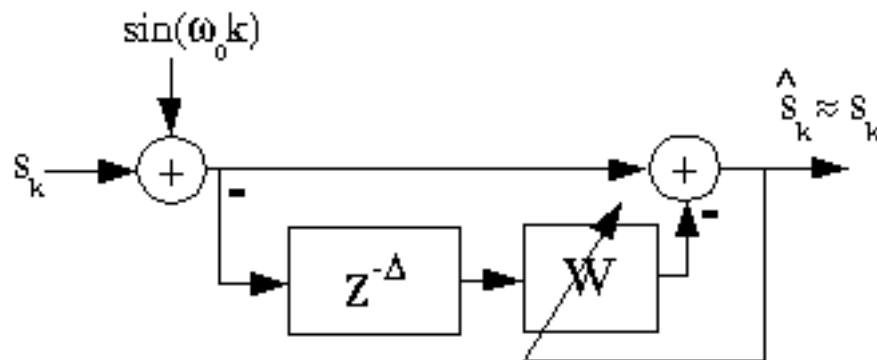


Figure 3.12

A sinusoid is predictable Δ samples ahead, whereas s_k may not be, so the sinusoid can be cancelled using the adaptive system in the Figure. This is another special case of the adaptive interference canceller in which the noise reference input is a delayed version of the primary (signal plus noise) input. Note that Δ must be large enough so that s_k and $s_{k-\Delta}$ are uncorrelated, or some of the signal will be cancelled as well!

Exercise 3.2

How would you construct an "adaptive line enhancer" that preserves the sinusoids but cancels the uncorrelated noise?

Other Applications

- Adaptive array processing
- Adaptive control
- etc...

Chapter 4

Other Adaptive Filter Algorithms

4.1 Beyond LMS: an overview of other adaptive filter algorithms¹

4.1.1 RLS algorithms

FIR adaptive filter algorithms with faster convergence. Since the Wiener solution can be obtained on one step by computing $W_{\text{opt}} = R^{-1}P$, most RLS algorithms attempt to estimate R^{-1} and P and compute W_{opt} from these.

There are a number of $O(N^2)$ algorithms which are stable and converge quickly. A number of $O(N)$ algorithms have been proposed, but these are all unstable except for the lattice filter method. This is described to some extent in the text. The adaptive lattice filter converges quickly and is stable, but reportedly has a very high noise floor.

Many of these approaches can be thought of as attempting to "orthogonalize" R , or to rotate the data or filter coefficients to a domain where R is diagonal, then doing LMS in each dimension **separately**, so that a fast-converging step size can be chosen in all directions.

4.1.2 Frequency-domain methods

Frequency-domain methods implicitly attempt to do this:

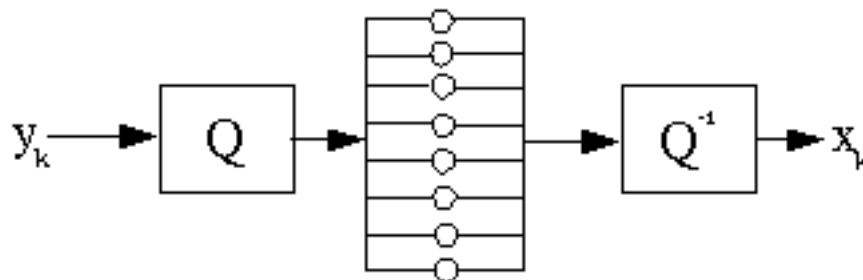


Figure 4.1

If QRQ^{-1} is a diagonal matrix, this yields a fast algorithm. If Q is chosen as an FFT matrix, each

¹This content is available online at <http://cnx.org/content/m11911/1.1/>.

channel becomes a different frequency bin. Since R is Toeplitz and not a circulant, the FFT matrix will not exactly diagonalize R , but in many cases it comes very close and frequency domain methods converge very quickly. However, for some R they perform no better than LMS. By using an FFT, the transformation Q becomes inexpensive $O(N \log N)$. If one only updates on a block-by-block basis (once per N samples), the frequency domain methods only cost $O(\log N)$ computations per sample. which can be important for some applications with large N . (Say 16,000,000)

4.2 Adaptive IIR filters²

Adaptive IIR filters are attractive for the same reasons that IIR filters are attractive: many fewer coefficients may be needed to achieve the desired performance in some applications. However, it is more difficult to develop stable IIR algorithms, they can converge **very** slowly, and they are susceptible to local minima. Nonetheless, adaptive IIR algorithms are used in some applications (such as low frequency noise cancellation) in which the need for IIR-type responses is great. In some cases, the exact algorithm used by a company is a tightly guarded trade secret.

Most adaptive IIR algorithms minimize the **prediction** error, to linearize the estimation problem, as in deterministic or block linear prediction.

$$y_k = \sum_{n=1}^L (v_n^k y_{k-n}) + \sum_{n=0}^L (w_n^k x_{k-n})$$

Thus the coefficient vector is

$$W_k = \begin{pmatrix} v_1^k \\ v_2^k \\ \vdots \\ v_L^k \\ w_0^k \\ w_1^k \\ \vdots \\ w_L^k \end{pmatrix}$$

and the "signal" vector is

$$U_k = \begin{pmatrix} y_{k-1} \\ y_{k-2} \\ \vdots \\ y_{k-L} \\ x_k \\ x_{k-1} \\ \vdots \\ x_{k-L} \end{pmatrix}$$

The error is

$$\epsilon_k = d_k - y_k = d_k - W_k^T U_k$$

²This content is available online at <<http://cnx.org/content/m11912/1.1/>>.

An LMS algorithm can be derived using the approximation $E[\epsilon_k^2] = \epsilon_k^2$ or

$$\hat{\nabla}_k = \frac{\partial}{\partial W_k} (\epsilon_k^2) = 2\epsilon_k \frac{\partial}{\partial W_k} (\epsilon_k) = 2\epsilon_k \begin{pmatrix} \frac{\partial}{\partial v_1^k} (\epsilon_k) \\ \vdots \\ \frac{\partial}{\partial \epsilon_k} (w_1^k) \\ \vdots \end{pmatrix} = -2\epsilon_k \begin{pmatrix} \frac{\partial}{\partial v_1^k} (y_k) \\ \vdots \\ \frac{\partial}{\partial v_L^k} (y_k) \\ \frac{\partial}{\partial w_0^k} (y_k) \\ \vdots \\ \frac{\partial}{\partial w_L^k} (y_k) \end{pmatrix}$$

Now

$$\begin{aligned} \frac{\partial}{\partial v_i^k} (y_k) &= \frac{\partial}{\partial v_i^k} \left(\sum_{n=1}^L (v_n^k y_{k-n}) + \sum_{n=0}^L (w_n^k x_{k-n}) \right) = y_{k-n} + \sum_{n=1}^L \left(v_n^k \frac{\partial}{\partial v_i^k} (y_{k-n}) \right) + 0 \\ \frac{\partial}{\partial w_i^k} (y_k) &= \frac{\partial}{\partial w_i^k} \left(\sum_{n=1}^L (v_n^k y_{k-n}) + \sum_{n=0}^L (w_n^k x_{k-n}) \right) = \sum_{n=1}^L \left(v_n^k \frac{\partial}{\partial w_i^k} (y_{k-n}) \right) + x_{k-n} \end{aligned}$$

Note that these are difference equations in $\frac{\partial}{\partial v_i^k} (y_k)$, $\frac{\partial}{\partial w_i^k} (y_k)$: call them $\alpha_i^k = \frac{\partial}{\partial v_i^k} (y_k)$, $\beta_i^k = \frac{\partial}{\partial w_i^k} (y_k)$, then

$\hat{\nabla}_k = \left(\beta_1^k \quad \beta_2^k \quad \dots \quad \beta_L^k \quad \alpha_0^k \quad \dots \quad \alpha_L^k \right)^T$, and the IIR LMS algorithm becomes

$$y_k = W_k^T U_k$$

$$\alpha_i^k = x_{k-i} + \sum_{j=1}^L \left(v_j^k \alpha_i^{k-j} \right)$$

$$\beta_i^k = y_{k-i} + \sum_{j=1}^L \left(v_j^k \beta_i^{k-j} \right)$$

$$\hat{\nabla}_k = -2\epsilon_k \left(\beta_1^k \quad \beta_2^k \quad \dots \quad \alpha_0^k \quad \alpha_1^k \quad \dots \quad \alpha_L^k \right)^T$$

and finally

$$W_{k+1} = W_k - U \hat{\nabla}_k$$

where the μ may be **different** for the different IIR coefficients. Stability and convergence rate depends on these choices, of course. There are a number of variations on this algorithm.

Due to the slow convergence and the difficulties in tweaking the algorithm parameters to ensure stability, IIR algorithms are used only if there is an overriding need for an IIR-type filter.

4.3 The Constant-Modulus Algorithm and the Property-Restoral Principle³

The adaptive filter configurations that we have examined so far require a "desired signal" d_k . There are many clever ways to obtain such a signal, but in some potential applications a desired signal is simply not available. However, a "property-restoral algorithm" can sometimes circumvent this problem.

³This content is available online at <<http://cnx.org/content/m11913/1.1/>>.

If the uncorrupted signal has special properties that are characteristic of the signal and not of the distortion or interference, an algorithm can be constructed which attempts to cause the output of the adaptive filter to exhibit that property. Hopefully, the adapting filter will restore that property by removing the distortion or interference!

Example 4.1: the Constant-Modulus Algorithm (CMA)

Certain communication modulation schemes, such as PSK and FSK, transmit a sinusoid of a constant analytic magnitude. Only the frequency or phase change with time. The constant modulus algorithm tries to drive the output signal to one having a constant amplitude:

$$\epsilon_k = (|y_k|)^2 - A^2$$

One can derive an LMS (or other) algorithm that seeks a Wiener filter minimizing this error. In practice, this works very well for equalization of PSK and FSK communication channels.

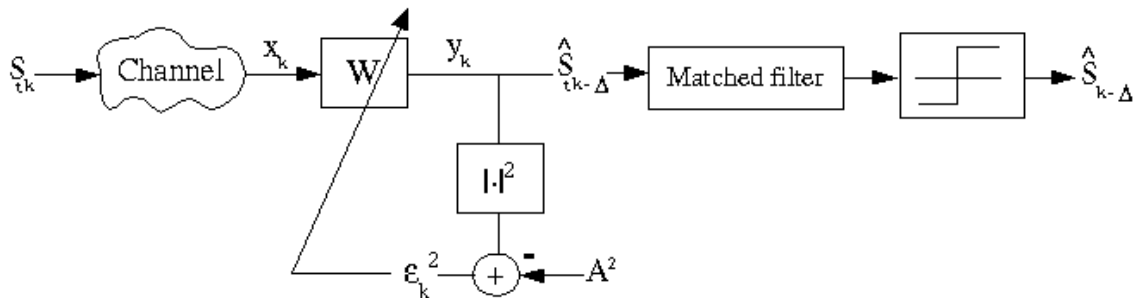


Figure 4.2

CMA is simpler than decision-directed feedback, and can work for high initial error rates!

This property-restoral idea can be used in any context in which a property-related error can be defined.

4.4 Complex LMS⁴

LMS for complex data and coefficients (such as quadrature communication systems) takes the form

$$y_k = W_k^H X_k$$

$$e_k = d_k - y_k$$

$$W_{k+1} = W_k + 2\mu e_k^* X_k$$

It is derived in exactly the same way as LMS, using the following complex vector differentiation formulas

$$\frac{d}{d\mathbf{W}} (P^H \mathbf{W}) = 0$$

⁴This content is available online at <<http://cnx.org/content/m11914/1.3/>>.

$$\frac{d}{d\mathbf{W}} (W^H \mathbf{P}) = 2\mathbf{P}$$

$$\frac{d}{d\mathbf{W}} (W^H R \mathbf{W}) = 2R\mathbf{W}$$

or by differentiating with respect to the real and imaginary parts separately and recombining the results.

4.5 Normalized LMS⁵

In "normalized" LMS, the gradient step factor μ is normalized by the energy of the data vector:

$$\mu_{\text{NLMS}} = \frac{\alpha}{X_k^H X_k + \sigma}$$

where α is usually $\frac{1}{2}$ and σ is a very small number introduced to prevent division by zero if $X_k^H X_k$ is very small.

$$W_{k+1} = W_k + \frac{1}{X_k^H X_k} e_k X_k$$

The normalization has several interpretations

1. corresponds to the 2nd-order convergence bound
2. makes the algorithm independent of signal scalings
3. adjusts W_{k+1} to give zero error with current input: $W_{k+1} X_k = d_k$
4. minimizes mean effort at time $k + 1$

NLMS usually converges **much** more quickly than LMS at very little extra cost; NLMS is **very** commonly used. In some applications, normalization is so universal that "we use the LMS algorithm" implies normalization as well.

⁵This content is available online at <<http://cnx.org/content/m11915/1.2/>>.

Chapter 5

Summary of Adaptive Filtering Methods¹

1. **LMS** - remains the simplest and best algorithm when slow convergence is not a serious issue (typically used) $O(N)$
2. **NLMS** - simple extension of the LMS with much faster convergence in many cases (very commonly used) $O(N)$
3. **Frequency-domain methods** - offer computational savings ($O(\log N)$) for long filters and usually offer faster convergence, too (sometimes used; very commonly used when there are already FFTs in the system)
4. **Lattice methods** - are stable and converge quickly, but cost substantially more than LMS and have higher residual EMSE than many methods (very occasionally used) $O(N)$
5. **RLS** - algorithms that converge quickly and are stable exist. However, they are considerably more expensive than LMS. (almost never used) $O(N)$
6. **Block RLS** - (least squares) methods exist and can be pretty efficient in some cases. (occasionally used) $O(\log N)$, $O(N)$, $O(N^2)$
7. **IIR** - methods are difficult to implement successfully and pose certain difficulties, but are sometimes used in some applications, for example noise cancellation of low frequency noise (very occasionally used)
8. **CMA** - very useful when applicable (blind equalization); CMA is **the** method for blind equalizer initialization (commonly used in a few specific equalization applications) $O(N)$

NOTE: In general, getting adaptive filters to work well in an application is much more challenging than, say, FFTs or IIR filters; they generally require lots of tweaking!

¹This content is available online at <<http://cnx.org/content/m11916/1.1/>>.

Bibliography

- [1] C.F.N. Cowan and P.M. Grant. *Adaptive Filters*. Prentice-Hall, 1985. Good overview of lots of topics.
- [2] W.A. Gardner. Learning characteristics of stochastic-gradient-descent algorithms: A general study, analysis, and critique. *Signal Processing*, 6:113–133, 1984.
- [3] S. Haykin. *Adaptive Filters Theory*. Prentice-Hall, 1986. Nice coverage of adaptive filter theory; Good reference.
- [4] M.L. Honig and D.G. Messerschmidt. *Adaptive Filters: Structures, Algorithms, and Applications*. Kluwer, 1984. Good coverage of lattice algorithms.
- [5] C.R. Johnson J.R. Treichler and M.G. Larimore. *Theory and Design of Adaptive Filters*. Wiley-Interscience, 1987. Good introduction to adaptive filtering, CMA; nice coverage of hardware.
- [6] O. Macchi and E. Eweda. Second-order convergence analysis of stochastic adaptive linear filtering. *IEEE Trans. on Automatic Controls*, AC-28 #1:76–85, Jan 1983.
- [7] B. Widrow and S.D. Stearns. *Adaptive Signal Processing*. Prentice-Hall, 1985. Good on applications, LMS.

Index of Keywords and Terms

Keywords are listed by the section with that keyword (page numbers are in parentheses). Keywords do not necessarily appear in the text of the page. They are merely associated with that section. *Ex.* apples, § 1.1 (1) **Terms** are referenced by the page they appear on. *Ex.* apples, 1

- | | | | |
|----------|---|----------|----------------------------|
| A | adaptive, 4, 6 | M | misadjustment factor, 18 |
| | adaptive interference cancellation, § 3.4(22) | P | prediction, 30 |
| | adaptive interference canceller, § 3.4(22) | R | Recursive least Squares, 6 |
| | adaptive noise cancellation, § 3.4(22) | S | stochastic gradient, 12 |
| | adaptive noise canceller, § 3.4(22) | T | training signal, 22 |
| G | gradient descent, 7 | | |

Attributions

Collection: *Adaptive Filters*

Edited by: Douglas L. Jones

URL: <http://cnx.org/content/col10280/1.1/>

License: <http://creativecommons.org/licenses/by/1.0>

Module: "Introduction to Adaptive Filters"

By: Douglas L. Jones

URL: <http://cnx.org/content/m11535/1.3/>

Page: 1

Copyright: Douglas L. Jones

License: <http://creativecommons.org/licenses/by/1.0>

Module: "Discrete-Time, Causal Wiener Filter"

By: Douglas L. Jones

URL: <http://cnx.org/content/m11825/1.1/>

Pages: 3-5

Copyright: Douglas L. Jones

License: <http://creativecommons.org/licenses/by/1.0>

Module: "Practical Issues in Wiener Filter Implementation"

By: Douglas L. Jones

URL: <http://cnx.org/content/m11824/1.1/>

Page: 6

Copyright: Douglas L. Jones

License: <http://creativecommons.org/licenses/by/1.0>

Module: "Quadratic Minimization and Gradient Descent"

By: Douglas L. Jones

URL: <http://cnx.org/content/m11826/1.2/>

Pages: 7-8

Copyright: Douglas L. Jones

License: <http://creativecommons.org/licenses/by/1.0>

Module: "The LMS Adaptive Filter Algorithm"

By: Douglas L. Jones

URL: <http://cnx.org/content/m11829/1.1/>

Pages: 11-13

Copyright: Douglas L. Jones

License: <http://creativecommons.org/licenses/by/1.0>

Module: "First Order Convergence Analysis of the LMS Algorithm"

By: Douglas L. Jones

URL: <http://cnx.org/content/m11830/1.1/>

Pages: 13-16

Copyright: Douglas L. Jones

License: <http://creativecommons.org/licenses/by/1.0>

Module: "Second-order Convergence Analysis of the LMS Algorithm and Misadjustment Error"

By: Douglas L. Jones

URL: <http://cnx.org/content/m11831/1.2/>

Pages: 16-18

Copyright: Douglas L. Jones

License: <http://creativecommons.org/licenses/by/1.0>

Module: "Applications of Adaptive Filters"

By: Douglas L. Jones

URL: <http://cnx.org/content/m11536/1.1/>

Page: 19

Copyright: Douglas L. Jones

License: <http://creativecommons.org/licenses/by/1.0>

Module: "Adaptive System Identification"

By: Douglas L. Jones

URL: <http://cnx.org/content/m11906/1.1/>

Pages: 19-20

Copyright: Douglas L. Jones

License: <http://creativecommons.org/licenses/by/1.0>

Module: "Adaptive Equalization"

By: Douglas L. Jones

URL: <http://cnx.org/content/m11907/1.1/>

Pages: 20-22

Copyright: Douglas L. Jones

License: <http://creativecommons.org/licenses/by/1.0>

Module: "Adaptive Interference (Noise) Cancellation"

By: Douglas L. Jones

URL: <http://cnx.org/content/m11835/1.1/>

Pages: 22-25

Copyright: Douglas L. Jones

License: <http://creativecommons.org/licenses/by/1.0>

Module: "Adaptive Echo Cancellation"

By: Douglas L. Jones

URL: <http://cnx.org/content/m11909/1.1/>

Pages: 25-27

Copyright: Douglas L. Jones

License: <http://creativecommons.org/licenses/by/1.0>

Module: "Beyond LMS: an overview of other adaptive filter algorithms"

By: Douglas L. Jones

URL: <http://cnx.org/content/m11911/1.1/>

Pages: 29-30

Copyright: Douglas L. Jones

License: <http://creativecommons.org/licenses/by/1.0>

Module: "Adaptive IIR filters"

By: Douglas L. Jones

URL: <http://cnx.org/content/m11912/1.1/>

Pages: 30-31

Copyright: Douglas L. Jones

License: <http://creativecommons.org/licenses/by/1.0>

Module: "The Constant-Modulus Algorithm and the Property-Restoral Principle"

By: Douglas L. Jones

URL: <http://cnx.org/content/m11913/1.1/>

Pages: 31-32

Copyright: Douglas L. Jones

License: <http://creativecommons.org/licenses/by/1.0>

Module: "Complex LMS"

By: Douglas L. Jones

URL: <http://cnx.org/content/m11914/1.3/>

Pages: 32-33

Copyright: Douglas L. Jones

License: <http://creativecommons.org/licenses/by/1.0>

Module: "Normalized LMS"

By: Douglas L. Jones

URL: <http://cnx.org/content/m11915/1.2/>

Page: 33

Copyright: Douglas L. Jones

License: <http://creativecommons.org/licenses/by/1.0>

Module: "Summary of Adaptive Filtering Methods"

By: Douglas L. Jones

URL: <http://cnx.org/content/m11916/1.1/>

Page: 35

Copyright: Douglas L. Jones

License: <http://creativecommons.org/licenses/by/1.0>

Adaptive Filters

Covers different types of adaptive filters and their implementations. In depth analysis is done of the Wiener Filter algorithm and the LMS algorithm.

About Connexions

Since 1999, Connexions has been pioneering a global system where anyone can create course materials and make them fully accessible and easily reusable free of charge. We are a Web-based authoring, teaching and learning environment open to anyone interested in education, including students, teachers, professors and lifelong learners. We connect ideas and facilitate educational communities.

Connexions's modular, interactive courses are in use worldwide by universities, community colleges, K-12 schools, distance learners, and lifelong learners. Connexions materials are in many languages, including English, Spanish, Chinese, Japanese, Italian, Vietnamese, French, Portuguese, and Thai. Connexions is part of an exciting new information distribution system that allows for **Print on Demand Books**. Connexions has partnered with innovative on-demand publisher QOOP to accelerate the delivery of printed course materials and textbooks into classrooms worldwide at lower prices than traditional academic publishers.