NEAR EAST UNIVERSITY

Faculty of Engineering

Department of Electrical and Electronic Engineering

ADAPTIVE FILTER LMS ALGORITHEM

Graduation Project
EE- 400

Student Ramiz Al- tatary (960329)

Supervisor Prof. Dr. Fakhreddin Mamedov

Lefkoşa – 2000

CONTENTS

AKNOWLEDGEMENT

All my thanks to those who endlessly encouraged me through out my undergraduate studies and provided me with all kind of supports financially or morality. Who stands beside me at every moment of my life, My Parents. Feeling proud to dedicate this project for them together with my respectful brothers ***Ramie***, ***Mohammed*** and ***Rjaiaden***.

I would like to express my faithful thanks to my university with all its educational staff and my instructors who have been a good guide for me, specially my supervisor ***Mr. Fakhreddin Mamedov*** who provided me with valuable advises and help to achieve my graduation project beside being an example of the responsible teacher.

Thanks to all my friends and school mates who joined me and shared my work sportingly.

Special thanks to my father ***Mr. Mohammed Al- Tatary.***

ABSTRACT

The design of a Wiener filter requires *a priori* information about the statistics of the data to be processed. When this information is not known completely, however, it may not be possible to design the Wiener filter or else the design may no longer be optimum. A straightforward approach that used in such situations is the "estimate and plug" procedure. For *real-time* operation, this procedure has the disadvantage of requiring excessively elaborate and costly hardware.

A more efficient method is to use an *adaptive filter.* By such a device we mean one that is *self-designing* in that the adaptive filter relies for its operation on a *recursive algorithm,* which makes it possible for the filter to perform satisfactorily in an environment where complete knowledge of the input signal characteristics is not available.

Introducing the least-mean-square LMS algorithm. Is important because of its simplicity, ease of computation, and because it does not require off-line gradient estimations or repetitions of data.

In this thesis, design adaptive filters based on Least Mean Square Algorithm are discussed.

The first chapter represents classification of filters, approximation of the frequency response characteristics using Butterworth, Chebyshev, and Elliptic Filters. Chapter provides comparison of analog and digital filters and different frequency response Characteristic.

Chapter two is devoted to the adaptive filter that provides real time operation in unknown input signal characteristic. General properties, Open loop, closed-loop adaptation are examined.

End sections of the chapter consider application of adaptive filter in identification, noise cancellation.

Chapter Three presents analysis of LMS algorithm and their software and hardware implementation. Basic limitations related with the effect of nonstationarity of input signals, computer worldlength requirement, driftt of coefficients are considered.

Chapter Four treats Finite Precision Effect, stalling phenomenon, and parameter drifts of the precision of filtering using LMS.

CHAPTER ONE
OVERVIEW OF FILTERS

Filtering is a process by which the frequency spectrum of signal can be modified, reshaped, or manipulated according to some desired specification. It may entail amplifying or attenuating a range of frequency components, rejecting or isolating one specific or attenuating a range of frequency component, etc. The uses of filtering are manifold, e.g., to eliminate signal contamination such as noise to remove signal distortion brought about by an imperfect transmission channel or by inaccuracies in measurement, to separate two or more distinct signals which were purposely mixed in order to maximize channel utilization, to demodulate signals, to convert discrete-time signals into continuous-time signals.
The digital filter is a digital system that can be used to filter discrete-time signals. It can Be implemented by mean of software (computer programs) or by means of dedicated Hardware, and in either case it can be used to filter real-time signals or non-real-time (Recorded) signals.

Software digital filters made their appearance along with the first digital computer in the late forties, although the name digital filter did not emerge until the midsixties. Early in the history of the digital computer many of the classical numerical analysis formulas of NEWTON, STARLING, etc,. and others were used to carry out interpolation, differentiation, and integration of function (signals) represented by mean of sequences of numbers (discrete-time signals). Since interpolation, differentiation, or integration of a signal represents a manipulation of the frequency spectrum of the signal, the subroutines or programs constructed to carry out these operations were essentially digital filters. In subsequent years, many complex and highly sophisticated algorithms and programs were developed to perform a variety of filtering tasks in numerous application, e.g., data smoothing and prediction, pattern recognition, electrocardiogram processing, and spectrum analysis. In fact, as time goes on, interest in the software digital filter is becoming progressively more intense while its applications are increasing at an exponential rate.
Band-limited continuous-time signals can be transformed into discrete-time signals By means of sampling. Conversely, the discrete-time signals so generated can be used to

regenerate the original continuous-time signals by means of interpolation, by virtue of Shannon's sampling theorem. As a consequence, hardware Digital's filters can be used to perform real-time filtering tasks, which in the not too distant past were performed almost Exclusively by analog filters. The advantages to be gained are the traditional advantages Associated with digital systems in general:

1. Component tolerances are uncritical.
2. Component drift and spurious environmental signals have no influence on the system Performance.
3. Accuracy is high.
4. Physical size is small.
5. Reliability is high.

A very important additional advantage of digital filters is the ease with which filter Parameters can be changed in order to change the filters characteristics. This feature Allows one to design programmable filters which can be used to perform a multiplicity of Filtering tasks. Also one can design new types of filters such as adaptive filters. The main disadvantage of hardware digital filters at present is their relatively high cost. However, with the tremendous advancements in the domain of large-scale integration, the cost of hardware digital filters is likely to drop drastically in the not too distant future.

1.1 Classic Analog Filters

While the importance of analog filters is continuously being reduced by their digital counterparts, they remain an important study, if for no other reason than they provide a gateway to the study of digital filters. The design of a contemporary analog filter, in many cases, remains today as it was during the early days of radio. The design objective Of the radio engineers was to shape the frequent-spectrum of a received or transmitted Signal using modulators, demodulators, and frequency-selective filters. The frequency-Selective filters were defined in terms of a mathematical ideal. The ideal models represent Low-pass, high-pass, band-pass, band-stop, and all-pass filters. These are graphically Interpreted in Figure 1.1. Their shape represents the steady-state magnitude-frequency Response of a filter with a transfer function of $H(\Omega) = H(s)\,|_{s=j\Omega}$ where Ω denotes an analog frequency measured in radians per second. The mathematical specification of

each ideal filter is summarized as,

$$\text{Ideal Low-pass } |H(\Omega)| = \begin{cases} 1 & \text{if } \Omega \in [-B,B] \\ 0 & \text{otherwise} \end{cases} \tag{1.1}$$

$$\text{Ideal High-pass } |H(\Omega)| = \begin{cases} 0 & \text{if } \Omega \in [-B,B] \\ 1 & \text{otherwise} \end{cases} \tag{1.2}$$

$$\text{Ideal Band-pass } |H(\Omega)| = \begin{cases} 1 & \text{if } \Omega \in [-B_2,-B_1] \text{ or } \Omega \in [B_1,B_2] \\ 0 & \text{otherwise} \end{cases} \tag{1.3}$$

$$\text{Ideal Band-stop } |H(\Omega)| = \begin{cases} 0 & \text{if } \Omega \in [-B_2,-B_1] \text{ or } \Omega \in [B_1,B_2] \\ 1 & \text{otherwise} \end{cases} \tag{1.4}$$

$$\text{All-pass } |H(\Omega)| = 1 \text{ for all } \Omega \in [-\infty,\infty] \tag{1.5}$$

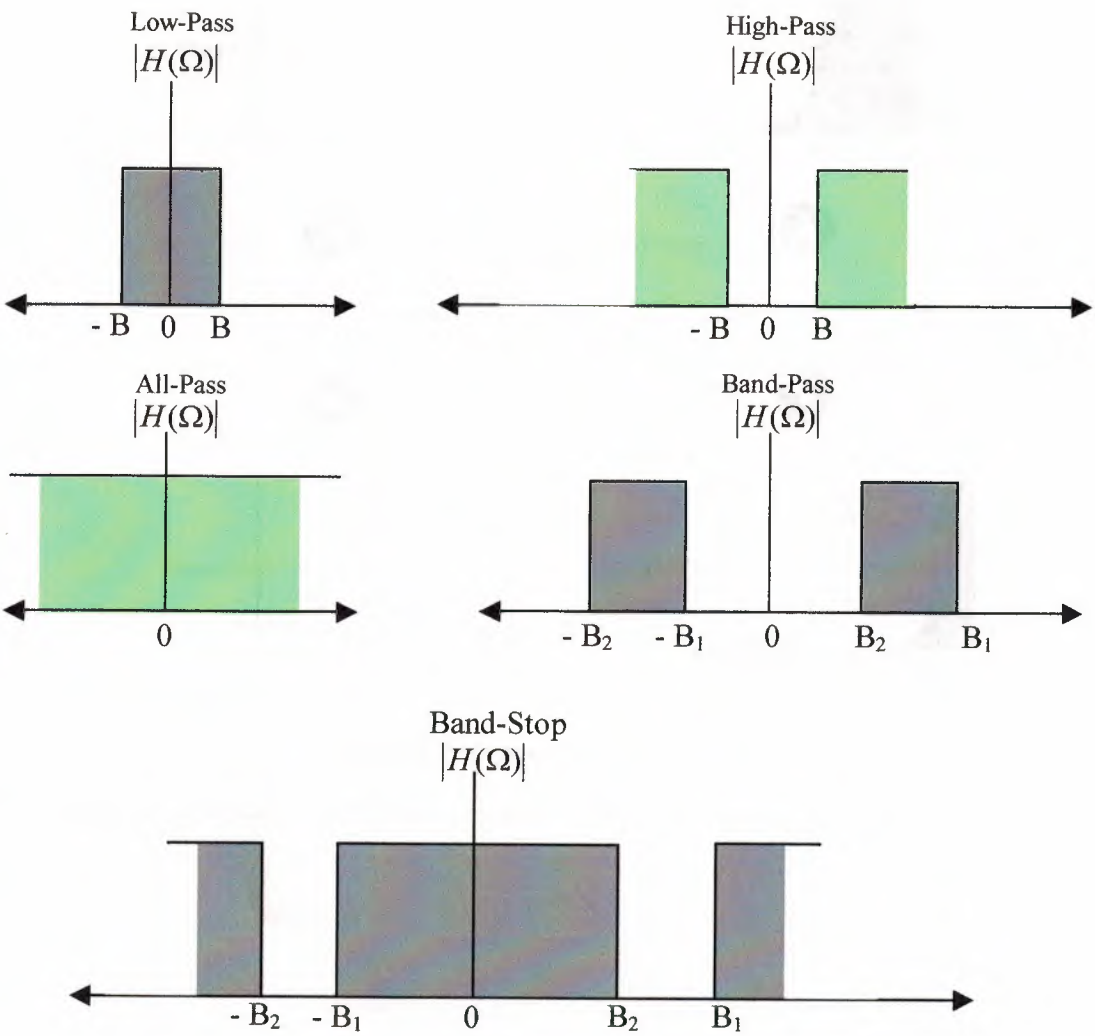


Figure 1.1 Basic Ideal Filter Types

Analog filter design is often based on the use of several well-known models called Butterworth, Chebyshev, and elliptic (Cauer) filters. To standardize the design procedure, a set of normalized analog prototype filter models was agreed upon and reduced to tables, charts, and graphs. These models, called prototypes, were all developed as low-pass systems having a known gain (typically -1 dB or -3 dB pass-band attenuation) at a known critical cut-off frequency (typically 1 radian/second). The transfer function of an analog prototype filter, denoted Hp(s), would be encapsulated in a standard table as a function of filter type and order. The prototype filter $H_p(s)$ would then be mapped into a final filter H(s) having critical frequencies specified by the designer. The mapping rules,

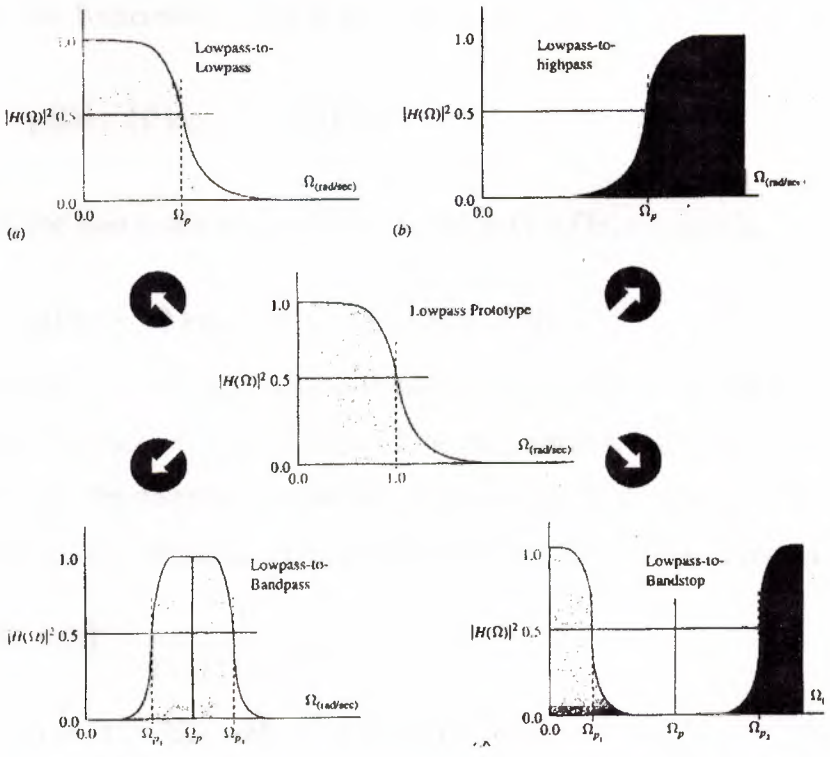


Figure 1.2 Frequency Transform

called frequency - frequency transforms, as shown in figure above.

1.2 Polynomial Approximations of the Frequency

1.2.1 Butterworth Approximation

The magnitude-squared response of an analog low-pass Butterworth filter $H_a(s)$ of Nth order is given by [13],

$$|H_a(j\Omega)|^2 = \frac{1}{1+(\Omega/\Omega_c)^{2N}} \tag{1.1}$$

It can be easily shown that the first 2N-1 derivatives of $|H_a(j\Omega)|^2$ at $\Omega = 0$ are equal to zero, and as a result, the Butterworth filter is said to have a maximally-flat magnitude at $\Omega = 0$. The gain of the Butterworth filter in dB is given by,

$$g(\Omega) = 10 \log_{10} |H_a(j\Omega)|^2 \text{ dB.}$$

A dc i.e., at $\Omega = 0$, the gain in dB is equal to zero, and at $\Omega = \Omega c$, the gain is,

$$g(\Omega c) = 10 \log_{10} (1/2) = -3.0103 \cong -3 \text{ dB}$$

and therefore, He is often called the 3-dB cutoff frequency. Since the derivative of the squared-magnitude response, or equivalently, of the magnitude response is always negative for positive values of Ω, the magnitude response, is monotonically decreasing with increasing Ω. For $\Omega >> \Omega_c$, the squared-magnitude function can be approximated by,

$$|H_a(j\Omega)|^2 = \frac{1}{1+(\Omega/\Omega_c)^{2N}}$$

The gain $g(\Omega_2)$ in dB at $\Omega_2 = 2\Omega_1$ with $\Omega_1 >> \Omega c$ is given by,

$$g(\Omega_c) = -20 \log_{10}\left(\frac{\Omega_2}{\Omega_c}\right)^{2N} = g(\Omega_1) - 6N \text{ dB,}$$

where $g(\Omega_1)$ is the gain in dB at Ω_1. As a result, the gain roll-off per octave in the stop-band decreases by 6 dB, or equivalently, by 20 dB per decade for an increase of the filter order by one. In other words, the pass-band and the stop-band behaviors of the magnitude response improve with a corresponding decrease in the transition band as the

fitter order N increases. A plot of the magnitude response of the normalized Butterworth low-pass filter with $\Omega_c = 1$ for some typical values of N is shown in figure.

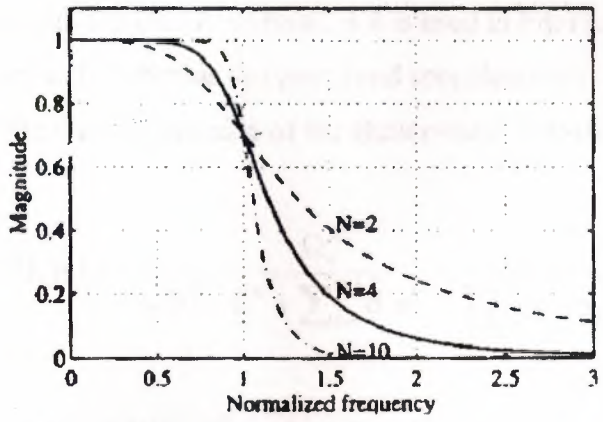


Figure 1.1 Typical Butterworth low-pass filter response.

.

The two parameters completely characterizing a Butterworth filter are therefore the 3-dB cutoff frequency Ω_c and the order N. These are determined from the specified pass-band edge Ω_p, the minimum pass-band magnitude $1/\sqrt{1+\varepsilon^2}$, the stop-band edge Ω_s, and the maximum stop-band ripple 1/A. From Eq. (1.1) we get,

$$\left|H_a(j\Omega_p)\right|^2 = \frac{1}{1+(\Omega_p/\Omega_c)^{2N}} = \frac{1}{1+\varepsilon^2} \tag{1.2a}$$

$$\left|H_a(j\Omega_s)\right|^2 = \frac{1}{1+(\Omega_s/\Omega_c)^{2N}} = \frac{1}{A^2} \tag{1.2b}$$

By solving the above we get the expression for the order N as,

$$N = \frac{1}{2}\frac{\log_{10}\left[(A^2-1)/\varepsilon^2\right]}{\log_{10}(\Omega_s/\Omega_p)} = \frac{\log_{10}(1/k_1)}{\log_{10}(1/k)} \tag{1.3}$$

Since the order N of the filter must be an integer, the value of N computed using the

above expression is rounded up to the next higher integer. This value of N can be used next in either Eq. (1.2a) or (1.2b) to solve for the 3-dB cutoff frequency Ω_c. If it is used in Eq. (1.2a), the pass-band specification is met exactly, whereas the stop-band specification is exceeded. On the Other hand, if it is used in Eq. (1.2b), the stop-band specification is met exactly, whereas the pass-band specification is exceeded.
The expression for the transfer function of the Butterworth low-pass filter is given by,

$$H_a(s) = \frac{C}{D_N(s)} = \frac{\Omega_c^N}{s^N + \sum_{\ell=0}^{N-1} d_\ell s^\ell} = \frac{\Omega_c^N}{\prod_{\ell=1}^{N}(s - p_\ell)} \tag{1.4}$$

Where,

$$p\ell = \Omega_c e^{j[\pi(N+2\ell-1)/2N]}, \quad \ell = 1,2,...,N \tag{1.5}$$

The denominator $D_N(s)$ of Eq. (1.4) is known as the Butterworth polynomial of order N and is easy to compute.

I.2.2 Chebyshev Approximation

In this case, the approximation error, defined as the difference between the ideal brick wall characteristic and the actual response, is minimized over a prescribed band of frequencies. In fact, the magnitude error is equiripple in the band. There are two types of Chebyshev transfer functions [2]. In the Type 1 approximation, the magnitude characteristic is equiripple in the pass-band and monotonic in the stop-band, whereas in the Type 2 approximation, the magnitude response is monotonic in the pass-band and equiripple in the stop-band.

1.2.3 Type 1 Chebyshev Approximation

The type 1 Chebyshev transfer function $H_a(s)$ has a magnitude response given by,

$$|H_a(j\Omega)|^2 = \frac{1}{1 + \varepsilon^2 T_N^2(\Omega/\Omega_p)}, \tag{1.7}$$

Where $T_N(\Omega)$ is the Chebyshev polynomial of order N:

$$T_N(\Omega) = \begin{cases} \cos\left(N \cos^{-1}\Omega\right), & |\Omega| \le 1, \\ \cosh\left(N \cosh^{-1}\Omega\right), & |\Omega| > 1, \end{cases} \tag{1.8}$$

The above polynomial can also be derived by recurrence relation given by,

$$T_r(\Omega) = 2\Omega\, T_{r-1}(\Omega) - T_{r-2}(\Omega), \qquad r \ge 2, \tag{1.9}$$

with $T_0(\Omega) = 1$ and $T_1(\Omega) = \Omega$

Typical plots of the magnitude responses of the Type 1 Chebyshev low-pass filter are shown in Figure 1.2 for three different values of filter order N with the same pass-band ripple ε. From these plots it is seen that the square-magnitude response is equiripple between $\Omega = 0$ and $\Omega = 1$, and it decreases monotonically for all $\Omega > 1$.

The zeros are on the jΩ-axis and are given by,

$$z_\ell = j\frac{\Omega_s}{\cos\left[\dfrac{(2\ell - 1)\pi}{2N}\right]}, \qquad \ell = 1,2,.....,N. \tag{1.16}$$

If N is odd, then for $\ell = (N + 1)/2$, the zero is at $s = \infty$. The poles are located at,

$$p_\ell = \sigma_\ell + j\Omega_\ell, \qquad \ell = 1,2,.....,N, \tag{1.17}$$

Where,

$$\sigma_\ell = \frac{\Omega_s \alpha_\ell}{\alpha_\ell^2 + \beta_\ell^2}, \qquad \Omega_\ell = \frac{\Omega_s \beta_\ell}{\alpha_\ell^2 + \beta_\ell^2},$$

$$\alpha_\ell = -\Omega_p \varsigma \sin\left[\frac{(2\ell - 1)\pi}{2N}\right], \qquad \beta_\ell = \Omega_p \xi \cos\left[\frac{(2\ell - 1)\pi}{2N}\right], \tag{1.18}$$

$$\varsigma = \frac{\gamma^2 - 1}{2\gamma}, \qquad \xi = \frac{\gamma^2 + 1}{2\gamma}, \qquad \gamma = \left(A + \sqrt{A^2 - 1}\right)^{1/N}.$$

The order N of the Type 2 Chebyshev low-pass filter is determined from given ε, Ω_s, and A using Eq. (1.11).

1.2.4 Elliptic Approximation:

An elliptic filter [8], also known as a Cauer filter, has an equiripple pass-band and an equiripple stop-band magnitude response, as indicated in Figure 1.3 for typical elliptic low-pass filters. The transfer function of an elliptic filter meets a given set of filter specifications, pass-band edge frequency Ω_p, stop-band edge frequency ^, pass-band ripple Ω_s, and minimum stop-band attenuation A, with the lowest filter order N. The theory of elliptic filter approximation is mathematically quite involved. The square-magnitude response of an elliptic low-pass filter is given by,

$$|H_a(j\Omega)|^2 = \frac{1}{1 + \varepsilon^2 R_N^2(\Omega/\Omega_p)} \tag{1.20}$$

where $R_N(\Omega)$ is a rational function of order N satisfying the property $R_N(1/\Omega) = 1/R_N(\Omega)$, with the roots of its numerator lying within the interval $0 < \Omega < 1$ and the roots of its denominator lying in the interval $1 < \Omega < \infty$. For most applications, the filter order meeting a given set of specifications of pass-band edge frequency Ω_p, pass-band ripple ε, stop-band edge frequency Ω_s, and the minimum stop-band ripple A can be, estimated by using the approximate formula,

$$N \cong \frac{2\log_{10}(4/k_1)}{\log_{10}(1/p)} \tag{1.21}$$

where k_1 is the discrimination parameter and ρ is computed as follows :

$$k' = \sqrt{1 - k^2}$$
$$\rho_0 = \frac{1 - \sqrt{k'}}{2\left(1 + \sqrt{k'}\right)} \tag{1.22}$$
$$\rho = \rho_0 + 2(\rho_0)^5 + 15(\rho_0)^9 + 150(\rho_0)^{13}.$$

in Eq.(1.22a), k is the selective parameter.

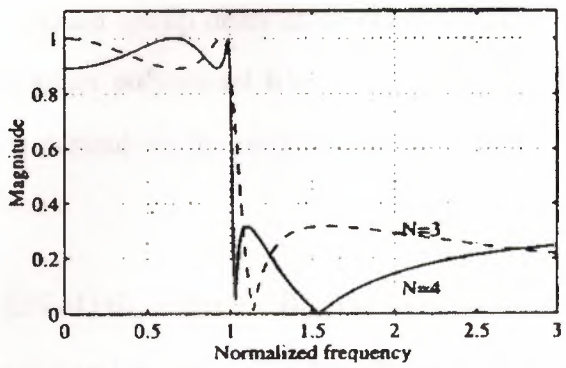


Figure 1.3 Typical elliptic low-pass filter responses with 1 dB pass-band ripple and 10 dB minimum stop-band attenuation.

1.2.5 Linear-Phase Approximation

The previous three approximation techniques are for developing analog low-pass transfer functions meeting specified magnitude or gain response specifications without any concern for their phase responses. In a number of applications it is desirable that the analog low-pass, filter being designed have a linear-phase characteristic in the pass-band, in addition to approximating the magnitude specifications. One way to achieve this goal is to cascade an analog all-pass filter with the filter designed to meet the magnitude specifications, so that the phase response of the overall cascade realization approximates linear-phase response in the pass-band. This approach increases the overall hardware complexity of the analog filter and may not be desirable for designing an analog anti-aliasing filter in some A/D conversion or designing an analog reconstruction filter in D/A

conversion applications. It is possible to design a low-pass filter that approximates a linear-phase characteristic in the pass-band but with a poorer magnitude response than that can be achieved by the previous three techniques. Such a filter has an all-pole transfer function of the form,

$$H(s) = \frac{d_0}{B_N(s)} = \frac{d_0}{d_0 + d_1 s + ... + d_{N-1} s + s^N} \tag{1.24}$$

and provides a maximally flat approximation to the linear-phase characteristic at $\Omega = 0$, i.e., has a maximally flat constant group delay at dc ($\Omega = 0$). For a normalized group delay of unity at dc, the denominator polynomial fi $B_N(s)$ of the transfer function, called the Bessel polynomial[2,8], can be derived via the recursion relation,

$$B_N(s) = (2N-1)\, B_{N-1}(s) + s^2 B_{N-2}(s) \tag{1.25}$$

starting with $B_1(s) = s + 1$ and $B_2(s) = s^2 + 3s + 3$. Alternatively, the coefficients of the Bessel polynomial $B_N(s)$ can be found from,

$$d_\ell = \frac{(2N-\ell)!}{2^{N-\ell}\,\ell!(n-\ell)!}, \qquad \ell = 0,1,....,N-1 \tag{1.26}$$

These filters are often referred to as Bessel filters.

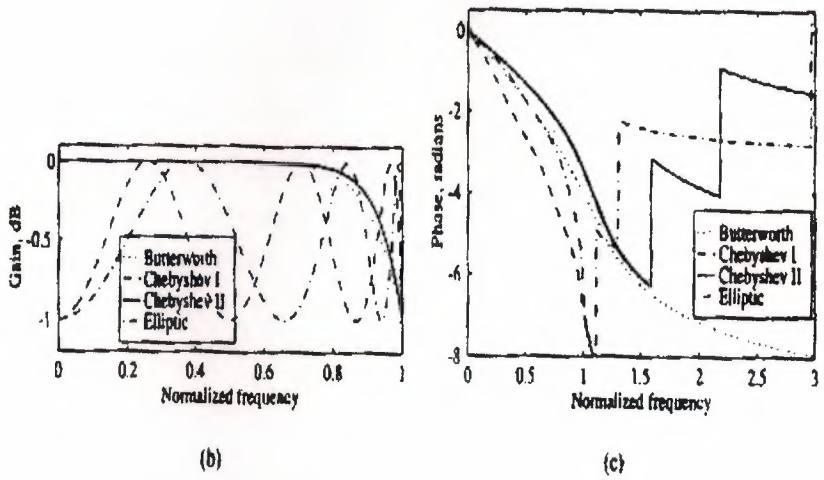
1.3 A Comparison of the Filter Types

In the previous sections we have discussed four types of analog low-pass fitter approximations, three of which have been developed primarily to meet the magnitude response specifications while the fourth has been developed primarily to provide a linear-phase approximation. In order to determine which filter type to choose to meet a given magnitude response specification, we need to compare the performances of the four types of approximations. To this end, we compare here the frequency responses of the normalized Butterworth, Chebyshev, and elliptic analog low-pass filters of same order. The pass-band

ripple of the Type 1 Chebyshev and the equiripple filters are assumed to be the same, while the minimum stop-band attenuation of the Type 2 Chebyshev and the equiripple filters are assumed to be the same. The filter specifications used for comparison are as follows: filter order of 6, pass-band edge at $\Omega = 1$, maximum pass-band deviation of 1 dB, and minimum stop-band attenuation of 40 dB. The frequency responses computed using MATLAB are plotted in Figure 1.5.

As can be seen from Figure 1.5, the Butterworth filter has the widest transition band, with a monotonically decreasing gain response. Both types of Chebyshev filters have a transition band of equal width that is smaller than that of the Butterworth filter but greater

Figure 1.5 A comparison of the frequency response of the four types of analog low-pass.



than that of the elliptic filter. The Type 1 Chebyshev filter provides a slightly faster roll-Off in the transition band than the Type 2 Chebyshev filter. The magnitude response of The Type 2 Chebyshev filter in the pass-band is nearly identical to that of the Butterworth Filter. The elliptic filter has the narrowest transition band, with an equiripple pass-band And an equiripple stop band response.

The Butterworth and Chebyshev fillers have a nearly linear-phase response over about Three-fourths of the pass-band, whereas the elliptic filter has a nearly linear-phase Response over about one-half of the pass-band. One the other hand, the Bessel filter may Be more attractive if the linearity of the phase response over a larger portion of the pass-

Band is desired at the expense of a poorer gain response. Figure 1.6 shows the gain and Phase responses of a sixth order Bessel filter frequency scaled to have a pass-band edge At $\Omega = 1$ with a maximum pass-band deviation of 1 dB. However, the Bessel filter Provides a minimum of 40 dB attenuation at approximately $\Omega = 9.4$ and as a result, has The largest transition band compared to the other three types.

CHAPTER TWO

ADAPTIVE FILTERS

2.1 Overview

The design of a Wiener filter requires *a priori* information about the statistics of the data to be processed. The filter is optimum only when the statistical characteristics of the input data match the *priori* information on which the design of the filter is based. When this information is not known completely, however, it may not be possible to design the Wiener filter or else the design may no longer be optimum.

A straightforward approach that we may use in such situations is the "estimate and plug" procedure. This is a two-stage process whereby the filter first "estimates" the statistical parameters of the relevant signals and then "plugs" the results so obtained into a *non-recursive* formula for computing the filter parameters. For *real-time* operation, this procedure has the disadvantage of requiring excessively elaborate and costly hardware. A more efficient method is to use an *adaptive filter.* By such a device we mean one that is *self-designing* in that the adaptive filter relies for its operation on a *recursive algorithm,* which makes it possible for the filter to perform satisfactorily in an environment where complete knowledge of the relevant signal characteristics is not available. The algorithm starts from some predetermined set of *initial conditions,* representing whatever we know about the environment. Yet, in a stationary environment, we find that after successive iterations of the algorithm it *converges* to the optimum Wiener solution in some statistical sense.

In a non-stationary environment, the algorithm offers a *tracking* capability , in that it can track time variations in the statistics of the input data, provided that the variations are sufficiently slow .

As a direct consequence of the application of a recursive algorithm whereby the parameters of an adaptive filter are updated from one iteration to the next, the parameters become *data dependent.* This, therefore, means that an adaptive filter is in reality a *nonlinear device, in the sense that it does not obey the principle of superposition.* The adaptive filters are commonly classified as linear or nonlinear. An adaptive filter is said to be *linear* if the estimate of a quantity of interest is computed adaptively (at the output of the filter) as a *linear combination of the available set of*

observations applied to the filter input. Otherwise, the adaptive filter is said to be *nonlinear.*

A wide variety of recursive algorithms have been developed in the literature for the operation of linear adaptive filters [13]. In the final analysis, the choice of one algorithm over another is determined by one or more of the following factors:

- *Rate of convergence.* This is defined as the number of iterations required for the algorithm, in response to stationary inputs, to converge "close enough" to the optimum Wiener solution in the mean-square sense. A fast rate of convergence allows the algorithm to adapt rapidly to a stationary environment of unknown statistics.
- *Misadjustment.* For an algorithm of interest, this parameter provides a quantitative measure of the amount by which the final value of the mean-squared error, aver- aged over an ensemble of adaptive filters, deviates from the minimum mean- squared error that is produced by the Wiener filter.
- *Tracking.* When an adaptive filtering algorithm operates in a non-stationary environment, the algorithm is required to *track* statistical variations in the environment. The tracking performance of the algorithm, however, is influenced by two contradictory features: (1) rate of convergence, and (b) steady-state fluctuation due to algorithm noise.
- *Robustness.* For an adaptive filter to be *robust,* small disturbances (i.e., disturbances with small energy) can only result in small estimation errors. The disturbances may arise from a variety of factors, internal or external to the filter.
- *Computational requirements.* Here the issues of concern include (a) the number of operations (i.e., multiplications, divisions, and additions/subtractions) required to make one complete iteration of the algorithm, (b) the size of memory locations required to store the data and the program, and (c) the investment required to program the algorithm on a computer.
- *Structure.* This refers to the structure of information flow in the algorithm, determining the manner in which it is implemented in hardware form. For example, an algorithm whose structure exhibits high modularity, parallelism, or concurrency is well suited for implementation using very large-scale integration (VLSI).I
- *Numerical properties.* When an algorithm is implemented numerically, inaccuracies are produced due to *quantization errors.* The quantization errors are due to analog-to-digital conversion of the input data and digital representation of internal calculations.

Ordinarily, it is the latter source of quantization errors that poses a serious design problem. In particular, there are two basic issues of concern: numerical stability and numerical accuracy. Numerical stability is an inherent characteristic of an adaptive filtering algorithm. *Numerical accuracy,* on the other hand, is determined by the number of *bits* (i.e., binary digits) used in the numerical representation of data samples and filter coefficients. An adaptive filtering algorithm is said to be numerically robust when it is insensitive to variations in the word-length used in its digital implementation.

These factors, in their own ways, also enter into the design of nonlinear adaptive filters, except for the fact that we now no longer have a well-defined frame of reference in the form of a Wiener filter. Rather, we speak of a nonlinear filtering algorithm that may converge to a local minimum or, hopefully, a global minimum on the error-performance surface.

In recent years, growing field of research in "adaptive systems" has resulted in a variety of adaptive automatos whose characterestics in limited ways resemble certain characterestics of living systems and biological adaptive processes.

Some meanings of "adaptation" can be applied in industrial, biological, social and etc.

An adaptive automation is asystem whose structure is alterable or adjustable in such a way that its behavior or performance (according to some desired criterion) improves throug contact with its environment. A simple example of an automaton or automatic adaptive system is the automatic gain control (AGC) used in radio and television receivers. The function of this circuit is to adjust the sensitivity of the receiver inversely as the average incoming signal strength. The receiver is thus able to adapt a wide range of input levels and to produce a much narrower range of output signals.

The purpose of this work is to present certain basic principles of adaptation; to explain the design, operating characteristics, and applications of the simpler forms of adaptive systems; and to describe means for their physical realization. The types of systems discussed include those designed primarily for the purposes of adaptive control and adaptive signal processing. Such systems usually have some or all of the following characteristics:

1.They can automatically adapt (self-optimize) in the face of changing (nonstationary) environments and changing system requirements.

2.They can be trained to perform specific filtering and decision-making tasks. Synthesis of systems having these capabilities can be accoplished automatically through

training. In a sense, adaptive systems can be "programmed" by atrain process.

1. Because of the above, adaptive systems do not require the elaborate synthesis procedures usually needed for nonadaptive systems. Instead, they tend to be "self-designing."
2. They can be extrapolate a model of behavior to deal with new situations after having been trained on a finite and often small number of training signals or patterns.
3. To alimited extent, they can repair themselves; that is, they can adapt around certain kinds of internal defects.
4. They can usually be described as nonlinear systems with time-varying parameters.
5. Usually, they are more complex and difficult to analyze than nonadaptive systems, but they offer the possibility of substantially increased system performance when input signal characteristics are unknown or time varying.

2.2 General Properties

The essential and principal property of the adaptive system is its time-varying, self-adjusting performance. The need for such performance may readily be seen by realizing that if a designer develops a system of fixed design which he or she considers optimal, the implications are that the designer has foreseen all possible input conditions, at least statistically, and knows what he or she would like the system to do under each of these conditions. The designer has then chosen a specific criterion whereby performance is to be judged, such as the amount of error between the output of the actual system and that of some selected model or "ideal"system.

Finally, the designer has chosen the system that appears best according to the performance criterion selected, generally choosing this system from an a priorirestricted class of designs (such as linear systems).

In many instances, however, the complete range of input conditions may not be known exactly, or even statistically; or the conditions may change from time to time. In such circumstances,an adaptive system that continually seeks the optimum within an allowed class of possibilities, using an ordinarly search process, would give superior performance compared with a system of fixed design.

By their very nature, adaptive systems must be time varying and nonlinear. Their characteristics depend, among other things, on their input signals. If an input signals x_1 is applied, an adaptive system will adapt to it and produce an output y_1. If

another input signal, x_2, is applied, the system will adapt to this second signal and will again produce an output y_2.

Generally, the form or the structure or the adjustments of the adaptive system will be different for the two different inputs. If the sum of the two inputs is applied to the adaptive system, the latter will adapt to this new input-but it will produce an output that will generally not be the same as y_1+y_2, the sum of the outputs that would have corresponded to inputs x_1 and x_2. In such a case, as illustrated in Figure 1.1, the principle of superposition does not work as it does with linear systems. If a signal is applied to the input of an adaptive system to test its response characteristics, the systems adapts to this specific input and thereby changes its own form. Thus the adaptive system is inherently difficult to characterize in conventional terms.

Whithin the realm of nonlinear systems, adaptive systems cannot be distinguished as belonging to an absolutely clear subset. However, they have two features that generally distinguish them from other forms of nonlinear systems.

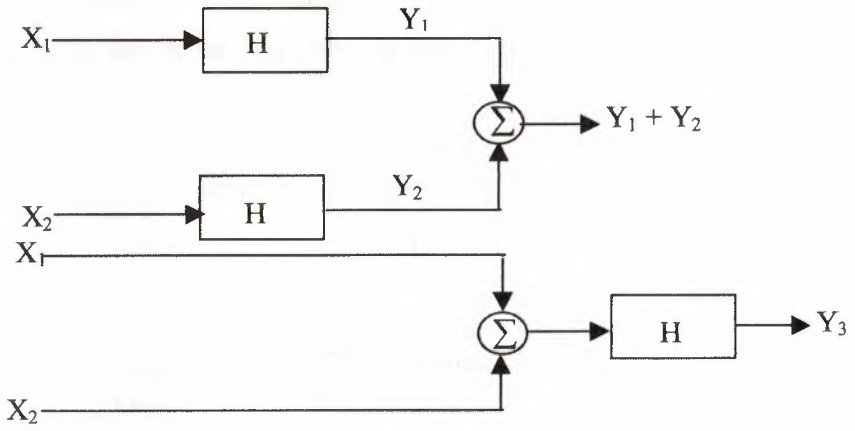


Figure 2.1 The lower output Y3 if H is a linear system, if H is adaptive Y3 is generally different from Y1+Y2

First, adaptive systems are adjustable, and their adjustments usually depend on finite-term average signal characteristics rather than on instantaneous values of signals or instantaneous values of the internal system state. Second, the adjustments of the adaptive systems are changed purposefully in order to optimize specified performance measures.

Certain forms of adaptive systems become linear systems when their adjustments are held constant after adaptation. These may be called "linear adaptive systems." They are

very useful; they tend to be mathematically tractable; and they are generally easier to design than other forms of adaptive systems.

2.3 Open-And Closed-Loop Adaptation

Several ways to classify adaptive schemes have been proposed in the literature [4]. It is most convenient here to begin by thinking in terms of open-loop and closed-loop adaptation. The open-loop adaptive process involves making measurements of input or environmental characteristics, applying this information to a formula or to a computational algorithm, and using the results to set the adjustments of the adaptive system.

Closed-loop adaptation, on the other hand, involves automatic experimentation with these adjustments and knowledge of their outcome in order to optimize a measured system performance. The latter process called adaptation by "performance feedback."

The principles of open- and closed-loop adaptation are illustrated in figures 1.2 and 1.3. The "other data" in these figures may be data about the environment of the adaptive system, or in the closed-loop case, it may be a desired version of the output signal.

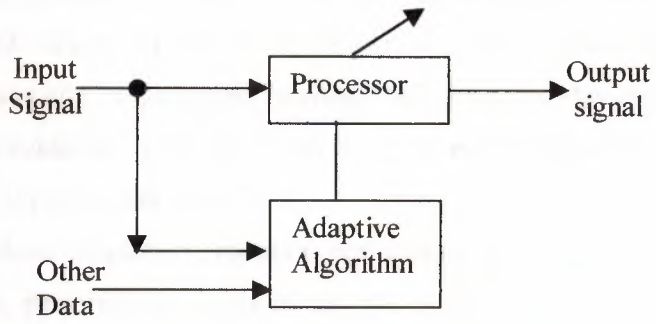


Figure 2.2 Open loop adaptations

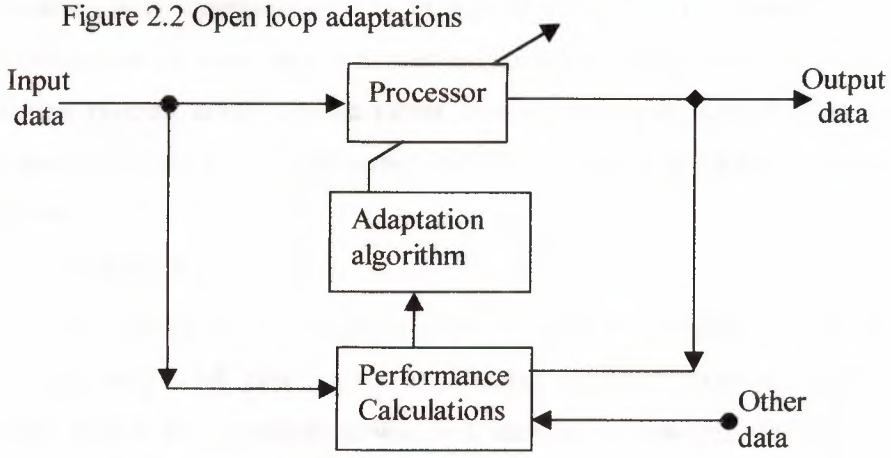


Figure 2.3 Closed loop adaptation

When designing an adaptive process, many factors determine the chpice of clsed-loop versus open-loop adaptation. The availability of input signals and performance-indicating signals is a major consideration. Also, the amount of computing capacity and the type of computer required to implement the open-loop and closed-loop adaptation algorithms will generally differ. Certain algorithms require the use of a general-purpose digital computer, whereas other algorithms could be implemented far more economically with special-purpose chips or other apparatus.

It is difficult to develop general principles to guide all choices, but several advantages and a few disadvatntages of closed-loop adaptation, which is the main subject can be pointed out here.

Closed-loop adaptation has the advantages of being workable in many applications where no analytic synthesis procedure either exists or is known, for example, where error criteria other than mean-square are used, where systems are nonlinear or time variable, where signals are nonsattionary, and so on.

Closed-loop can also be used effectively in situations where physical system component values are variable or inaccurately known. Closed-loop adaptation will find the best choice of component values. In the event of partial system failure, an adaptation mechanism that continually monitors performance will optimize this performance by adjusting and reoptimizing the intact parts. As a result, system reliability can often be improved by the use of performance feedback.

The closed-loop adaptation process is not always free of difficulties, however. In certain situations, performance functions do not have unique optima. Automatic optimization is an uncertain process in such situations. In othersituations, the closed-loop adaptation process, like a closed-loop control system, could be unstable. The adaptation process could diverge rather than converge. In spiteof these possibilities, performance feedback is a powerful, widely applicable technique for implementing adaptation.

2.4 Applications

The ability of an adaptive filter to operate satisfactorily in an unknown environment and track time variations of input statistics make the adaptive filter a powerful device for signal-processing and control applications. Indeed, [4] adaptive filters have been successfully applied in such diverse fields as communications, radar,

sonar, seismology, and biomedical engineering. Although these applications are indeed quite different in nature, nevertheless, they have one basic common feature: an input vector and a desired response are used to compute an estimation error, which is in turn used to control the values of a set of adjustable filter coefficients. The adjustable coefficients may take the form of tap weights, reflection coefficients, rotation parameters, or synaptic weights, depending on the filter structure employed. However, the essential difference between the various applications of adaptive filtering arises in the manner in which the desired response is extracted. In this context, we may distinguish four basic classes of adaptive filtering applications, as depicted in Fig. 2.4. For convenience of presentation, the following notations are used in this figure:

u = input applied to the adaptive filter

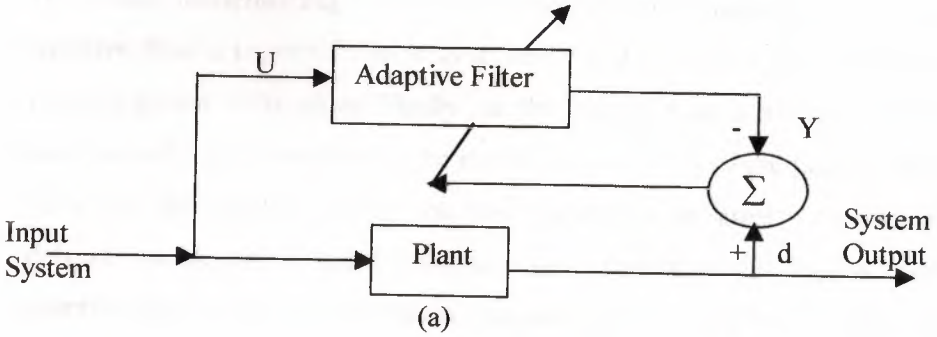
Y = output of the adaptive filter

d = desired response

$e = d - y$ = estimation error .

The functions of the four basic classes of adaptive filtering applications depicted herein are as follows:

I. *Identification* Fig. 2.4(a). The notion of a *mathematical model* is fundamental to sciences and engineering. In the class of applications dealing with identification, an adaptive filter is used to provide a linear model that represents the best fit (in some sense) to an *unknown plant.* The plant and the adaptive filter are driven by the same input. The plant output supplies the desired response of the adaptive filter. If the plant is dynamic in nature, the model will be time varying.



(a)

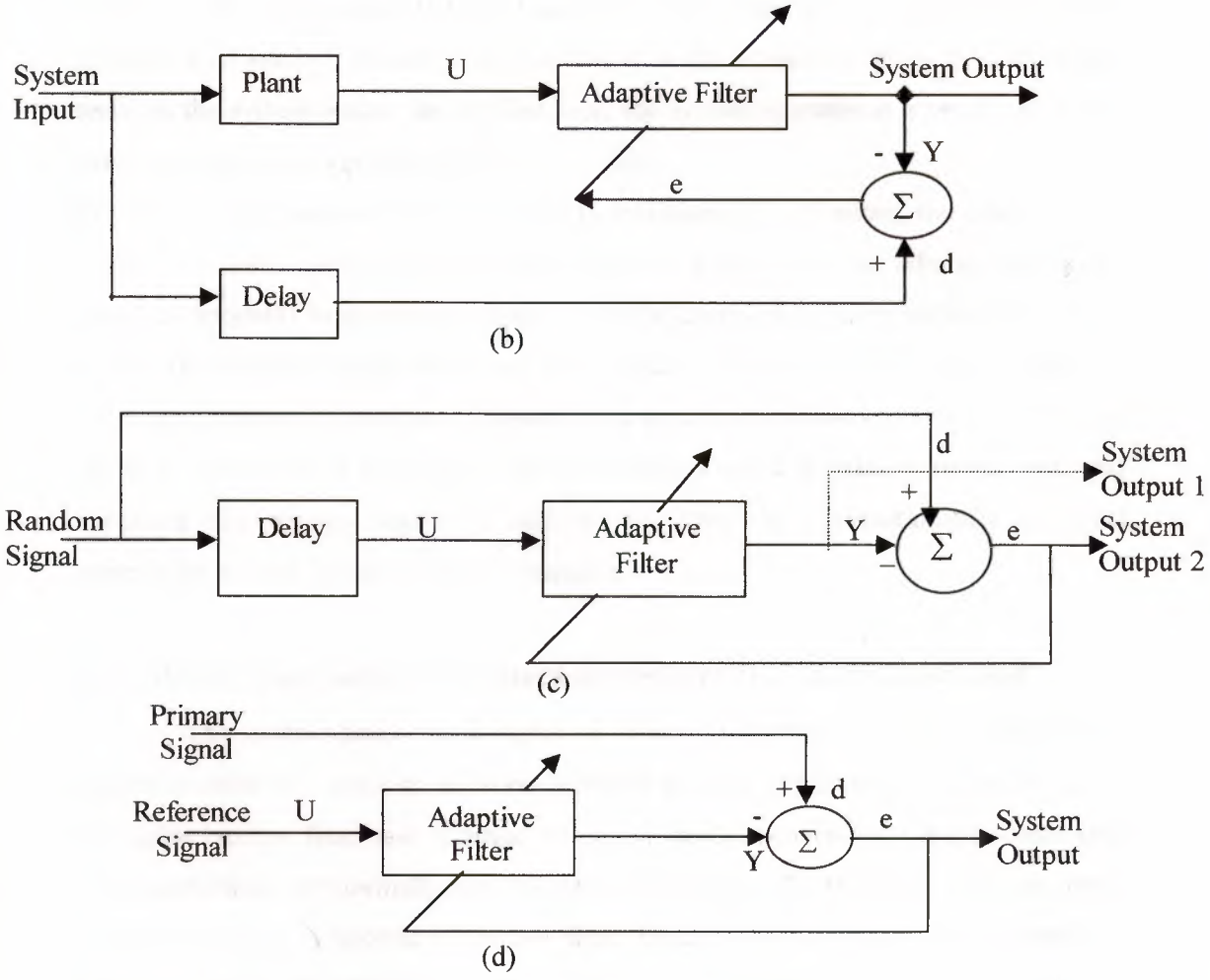


Fig. 2.4 Four Basic Classes of Adaptive Filtering Applications

(a) Identification (b) Inverse Modeling (c) Prediction (d) Interference Canceling

II. *Inverse modeling* Fig.2.4(b). In this second class of applications, the function of the adaptive filter is to provide an *inverse model* that represents the best fit (in some sense) to an *unknown noisy plant.* Ideally , in the case of a linear sys- tem, the inverse model has a transfer function equal to the *reciprocal (inverse)* of the plant' s transfer function, such that the combination of the two constitutes an ideal transmission medium. A delayed version of the plant (system) input constitutes the desired response for the adaptive filter. In some applications, the plant input is used without delay as the desired response.

III. *Prediction* Fig.2.4(c). Here the function of the adaptive filter is to provide the best *prediction* (in some sense) of the present value of a random signal. The present value of the signal thus serves the purpose of a desired response for the adaptive filter. Past

values of the signal supply the input applied to the adaptive filter. Depending on the application of interest, the adaptive filter output or the estimation (prediction) error may serve as the system output. In the first case, the system operates as a *predictor;* in the latter case, it operates as a *prediction- error filter.*

IV. *Noise Cancellation* Fig.2.4(d). In this final class of applications, the adaptive filter is used to cancel *unknown interference* contained (alongside an information-bearing signal component) in a *primary signal,* with the cancellation being optimized in some sense. The primary signal serves as the desired response for the adaptive filter. A *reference (auxiliary) signal* is employed as the input to the adaptive filter. The reference signal is derived from a sensor or set of sensors located in relation to the sensor(s) supplying the primary signal in such a way that the information-bearing signal component is weak or essentially undetectable.

2.5 When to use adaptive Filters and where they have been used

The contamination of a signal of interest by other unwanted, often larger, signals or noise is a problem often encountered in many applications. Where the signal and noise occupy fixed and separate frequency bands, conventional linear filters with fixed coefficients are normally used to extract the signal. [4]. However, there are many instances when it is necessary for the filter characteristics to be variable, adapted to changing signal characteristics, or to be altered intelligently. In such cases, the coefficients of the filter must vary and cannot be specified in advance. Such is the case where there is a spectral overlap between the signal and noise, see Figure 1.5. or if the band occupied by the noise is unknown or varies with time.

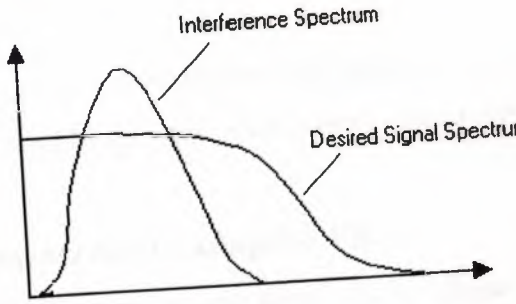


Figure 2.5 An illustration of spectral overlap between a signal and a strong interference

Typical applications where fixed coefficient filters are inappropriate are the following.

- Electroencephalography (EEG), where artefacts or signal contamination produced by eye movements or blinks is much larger than the genuine electrical activity of the brain and shares the same frequency band with signals of clinical interest. It is not possible to use conventional linear filters to remove the artefacts while preserving the signals of clinical interest.
- Digital communication using a spread spectrum, where a large jamming signal, possibly intended to disrupt communication, could interfere with the desired signal. The interference often occupies a narrow but unknown band within the wideband spectrum, and can only be effectively dealt with adaptively.
- In digital data communication over the telephone channel at a high rate. Signal distortions caused by the poor amplitude and phase response characteristics of the channel lead to pulses representing different digital codes to interfere with each other (intersymbol interference), making it difficult to detect the codes reliably at the receiving end. To compensate for the channel distortions which may be varying with time or of unknown characteristics at the receiving end, adaptive equalization is used.

An adaptive filter has the property that its frequency response is adjustable or modifiable automatically to improve its performance in accordance with some criterion, *allowing the filter to adapt to changes in the input signal characteristics. Because of their self-adjusting performance and in-built flexibility, adaptive filters have found use* in many diverse applications such as telephone echo canceling, radar signal processing, navigational systems, equalization of communication channels, and biomedical signal enhancement.

In summary we use adaptive filters

- When it is necessary for the filter characteristics to be variable, adapted to changing conditions,
- When there is spectral overlap between the signal and noise, or
- If the band occupied by the noise is unknown or varies with time.

2.6 Main Components of the Adaptive Filter

In most adaptive systems, the digital filter in figure 2.6. Is realized using a transversal or finite impulse response (FIR) structure figure 2.6.1. Other forms are sometimes used, for example the infinite impulse response (IIR) or the lattice structures,

but the FIR structure is the most widely used because of its simplicity and guaranteed stability. For the N-point filter depicted in figure 2.6.1, the output is given by

$$\hat{n}_k = \sum_{i=0}^{N-1} w_k(i) x_{k-i} \qquad (2.1)$$

where $w_k(i)$, i=0,1,..., are the adjustable filter coefficients (or weights) and $x_k(i)$ and $\hat{x}_k$ are the input and output of the filter. Figure 2.6.1. Illustrates the single-input, single-output system. In a multiple-input single-output system, the x_k may be simultaneous inputs from N different signal sources.

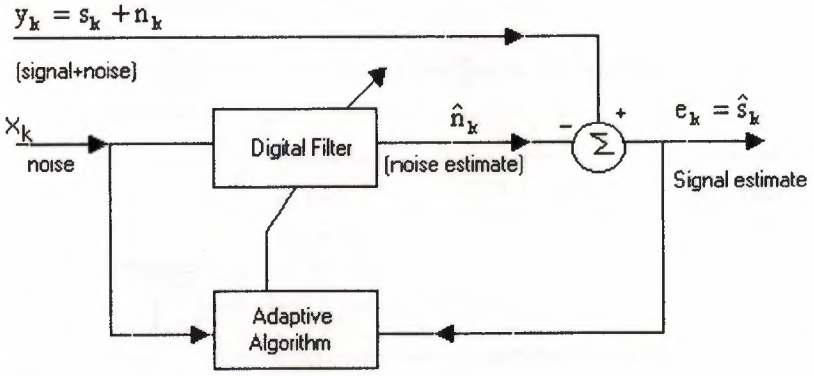


Figure 2.6. Block Diagram of an Adaptive Filter as a Noise Canceller

2.7 Other Applications

2.7.1 Loud speaking telephones

- The hybrid network is used to separate the transmit and receive paths (that is, the loudspeaker from the microphone), but there is a significant acoustic coupling between the loudspeaker and the microphone because of their proximity as well as a leakage across the imperfectly matched hybrid network (South et al., 1979).
- The difficulty then is how to provide adequate gain for the receive and transmit directions without causing instability.
- The conventional solution to the problems is to use a voice-activated switch to select the transmit and receive paths, but this is not satisfactory because it does not allow full duplex communication.
- A better solution is to use adaptive filtering techniques to estimate and control the acoustic and hybrid echoes Figure 2.7(b). The number of filter coefficients here can be quite large, for example 512, making the use of a fast algorithm attractive.
- In teleconferencing networks (or public address systems) acoustic feedback leads to problems similar to those described above. Adaptive filters used for these may require

large numbers of coefficients (250 to 1000), especially in rooms with long reverberation times, and must converge rapidly.

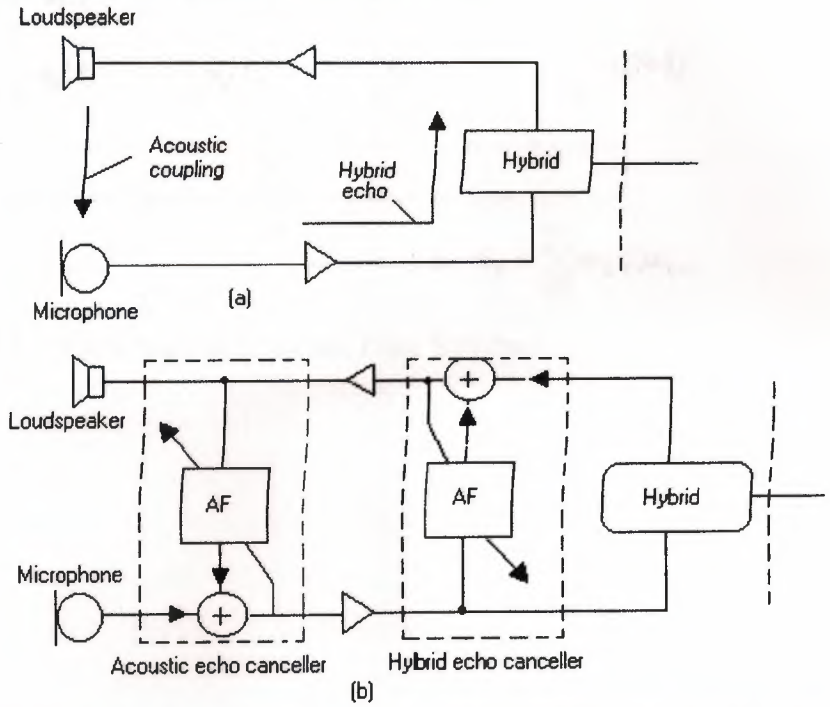


Figure 2.7. (a) Loud Speaking Telephone (b) Acoustic and Hybrid Echo Cancellation in Loud speaking Telephone

2.7.2 Radar Signal Processing

Adaptive signal processing techniques are widely used to solve a number of problems associated with radar. For example, adaptive filters are used in monostatic radar systems to remove or cancel clutter components from the desired target signals. In HF ground wave radar, adaptive filters are used to reduce co-channel interference, which is a major problem in the HF band.

2.7.3 Separation of Speech signals from background noise

Acoustic background noise is a serious problem in speech processing. An adaptive filter may be used to enhance the performance of speech systems in noisy environments (for example in fighter aircrafts, tanks, cars) to improve both

intelligibility and recognition of speech.

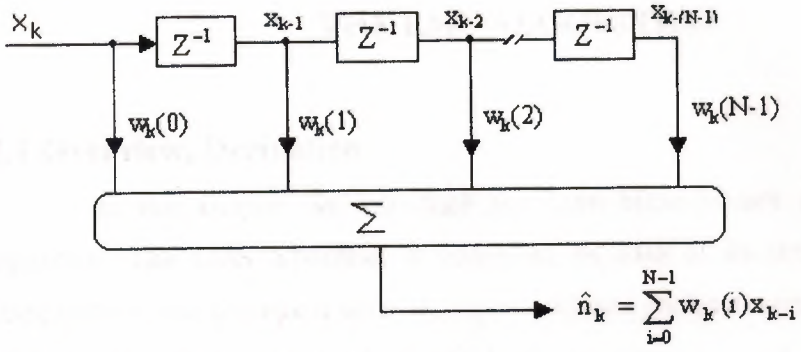


Figure 2.7.3. Finite Impulse Response Filter Structure

CHAPTER THREE
THE LMS ALGORITHM

3.1 Overview, Derivation

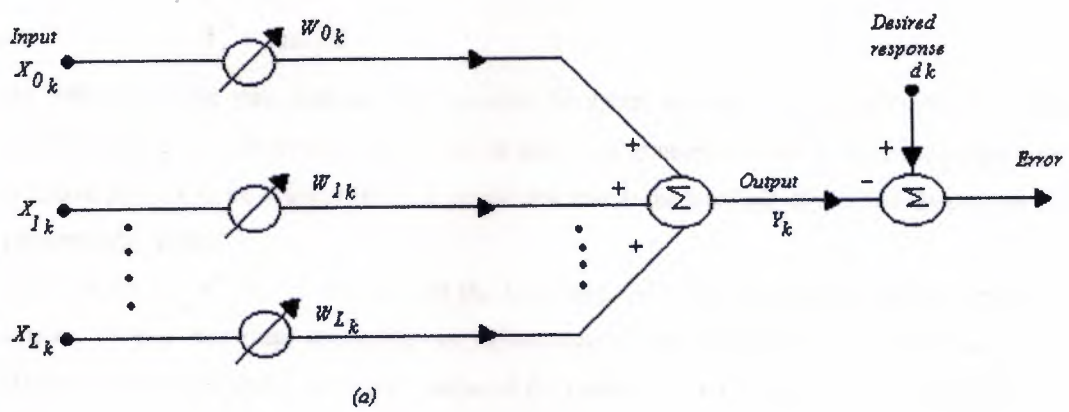
In this chapter we introduce the Least-Mean-Square Algorithm, or LMS algorithm. The LMS algorithm is important because of its simplicity and ease of computation, and because it does not require off-line gradient estimations or repetitions of data. If the adaptive system is an adaptive linear combiner, and if the input vextor X_k and the desired response d_k are available at each iteration, the LMS algorithm is generally the best choice for many different applications of adaptive signal procesing.

We recall that the adaptive linear combiner was applied in two basic ways, depending on whether the input is available in parallel (multiple inputs) or series (single input) form. These two ways are shown in figure 3.1.

In both cases we have the combiner output, y_k, as a linear combination of the input samples. We have

$$\varepsilon_k = d_k - X_k^T W_k \tag{3.1}$$

Where X_k is the vector of the input samples in either of the two configurations in figure 3.1.



(a)

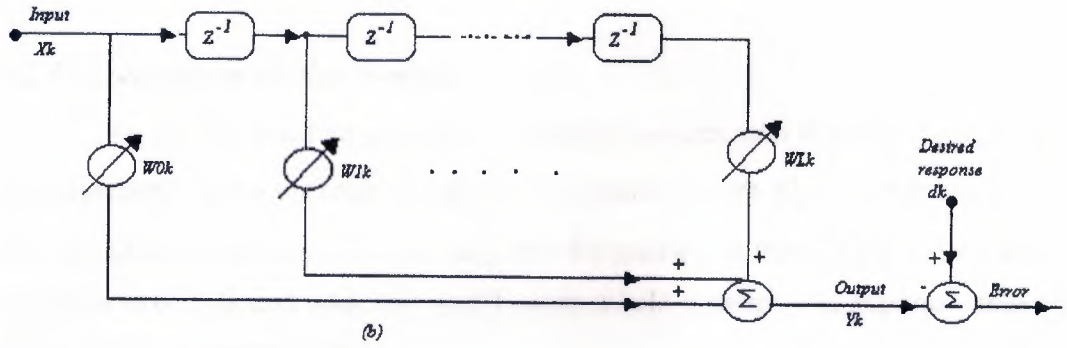


Figure 3.1 the adaptive Linear Combiner: (a) in general form; (b) as a transversal.

To develop an adaptive algorithm using the previous methods, we would estimate the gradient of $\xi = E\left[\varepsilon_k^2\right]$ by taking differencies between short-term averages of ε_k^2. Instead, to develop the LMS algorithm, we take ε_k^2. itself as an estimate of ξ_k.

Then, at each iteration in the adaptive process, we have a gradient estimate of the form

$$\hat{\nabla}_k = \begin{bmatrix} \frac{\partial \varepsilon_k^2}{\partial W_0} \\ \vdots \\ \frac{\partial \varepsilon_k^2}{\partial W_L} \end{bmatrix} = 2\varepsilon_k \begin{bmatrix} \frac{\partial \varepsilon_k}{\partial W_0} \\ \vdots \\ \frac{\partial \varepsilon_k}{\partial W_L} \end{bmatrix} = -2\varepsilon_k X_k \tag{3.2}$$

The derivatives of ξ_k. with respect to the weights follow derictly from (3.1).

With simple estimate of the gradient, we can now specify a steepest-descent type of adaptive algorithm. We have

$$\begin{aligned} W_{k+1} &= W_k - \mu\hat{\nabla}_k \\ &= W_k + 2\mu\varepsilon_k X_k \end{aligned} \tag{3.3}$$

As before, μ is the gain constant that regulates the speed and stability of adaptation. Since the weight changes at each iteration are based on imperfect gradient estimates, we would expect the adaptive process to be noisy, that is, it would not follow the true line of steepest descent on the performance surface, [6].

From its form in (3.3), we can see that the LMS algorithm can be implemented in a practical system without squaring, averaging, or differentiation and is elegant in its simplicity and efficiency. As noted above, each component of the gradient vector is obtained from a single data sample without pertubing the weight vector.

Without averaging, the gradient components do contain a large component of noise, but the noise is attenuated with time by the adaptive process, which acts as a low-pass filter in this respect.

3.2 Convergence of the Weight Vector

As with all adaptive algorithms, a primary concern with the LMS algorithm is its convergence to the optimum weight vector solution, where $E[\varepsilon_k^2]$ is minimized. To examine LMS convergence, we first note that the gradient estimate in (3.2) can readily be shown to be unbiased when the weight vector is held constant. The expected value of (3.2) with W_k held equal to W is

$$\begin{aligned} E[\hat{\nabla}_k] &= -2E[\varepsilon_k X_k] \\ &= -2E[d_k X_k - X_k X_k^T W] \\ &= 2(RW - P) = \nabla \end{aligned} \tag{3.4}$$

The second line of (3.4) follows from (3.1) plus the fact that ε_k is a scalar and can thus be commuted. Since the mean value of $\hat{\nabla}_k$ is equal to the true gradient $\nabla, \hat{\nabla}_k$ must be an unbiased estimate.

Seeing that the gradient estimate is unbiased, we could make the LMS algorithm into a true steepest-descent algorithm, at least in limiting case, by estimating ∇ at each step as in (3.2) but not adapting the weights until many steps have occurred, in this way $\hat{\nabla}_k$ could be made to approach ∇_k. With the weight vector changing at each iteration, we need to examine the weight vector convergence in a different manner, as follows.

From (3.3) we can see that the weight vector W_k is a function only of the past input vectors $X_{k-1}, X_{k-2}, \ldots\ldots X_0$. If we assume that successive input vectors are independent over time W_k is independent of X_k. For stationary input processes meeting this condition, the expected value of the weight vector E [W_k] after a sufficient number of iterations can be shown as follow to converge to the Wiener optimal solution , that is, to $W^* = R^{-1}P$

Taking the expected value of both sides of (3.3) yields the difference equation

$$\begin{aligned} E[W_{k+1}] &= E[W_k] + 2\mu E[\varepsilon_k X_k] \\ &= E[W_k] + 2\mu(E[d_k X_k] - E[X_k X_k^T W_k]) \end{aligned} \tag{3.5}$$

Using the foregoing assumption that X_k and W_k are independent, we have the expected products as in (3.5). Also, we have the optimum weight vector given as $W^* = R^{-1}P$. Thus (3.5) becomes

$$E[W_{k+1}] = E[W_k] + 2\mu(P - RE[W_k])$$

$$=(\text{I-}2\mu\text{R})\text{E}[W_k]+2\mu\text{RW*} \quad (3.6)$$

Using expected values, the solution is

$$E[V_k'] = (I - 2\mu R\Lambda)^k V'_0 \quad (3.7)$$

Where V' is the weight vector, W, in the principal-axis system A is the diagonal eigeanvalue matrix of R, and V_0' the initial weight vector in the principal-axis system.

Thus, as k increases without bound, we see that expected weight vector in (3.7) reaches the optimum solution (i.e., zero in the principal-axis system) only if the right side of the optimum convergence to zero.

$$\frac{1}{\lambda_{\max}} > \mu > 0 \quad (3.8)$$

Where $\lambda_{\max}$ is the largest eigenvalue, that is, the largest diagonal element in Λ. So in (3.8), we have bounds on μ for convergence of the weight vector mean to the optimum weight vector. Within these bounds, the speed of adaptation and also the noise in the weight vector solution are determined by the size of μ We also note that $\lambda_{\max}$ cannot be greater than the trace of R, which is the sum of the diagonal elements of R, that is,

$$\lambda_{\max} \le tr[\Lambda] = \sum(diagonal.elements.of\Lambda)$$

$$= \sum(diagonal.elements.of.R) = tr[R] \quad (3.9)$$

Furthermore, with a transversal adaptive filter gives tr[R] as just $(L+1)E[x_k^2]$ or L+1 times the input signal power. Thus convergence of the weight vector mean is assured by:

In general: $0<\mu<1/tr[R]$

Transversalfilter$0<\mu<1/(l+1)$ (3.10)

But is much easier to apply, because the elements R and the signal power can generally be estimated more easily eigenvalues of R.

The assumption of deceleration and stationary of input vector used to drive the result in this section are not necessary condition for convergence of the LMS algorithm but have been adopted in this chapter for analytic convergence.

Convergence with certain correlated and non-stationary inputs is demonstrated in the literature on the LMS algorithm [4].

Under these conditions the analysis becomes much more complex. We know of no unconditional proof of convergence of the LMS algorithm.

3.3 Noise In The Weight-Vector Solution

With the LMS algorithm, the gradient estimate as given by (3.2) is not based on weight perturbation, so we must reexamine its variance.

Let us define N_k as a vector of noise in the gradient estimate at the kth iteration. Thus

$$\hat{\nabla}_k = \nabla_k + N_k \tag{3.11}$$

If we assume that the LMS process, using a small value of the adaptive gain constant μ, has converged to a steady-state weight vector solution near W^*, then ∇_k in (3.11) will be close to zero. Then, in accordance with (3.2), the gradient noise is close to

$$N_k = \hat{\nabla}_k = -2\varepsilon_k X_k \tag{3.12}$$

The covariance of the noise is thus given by

$$\text{cov}[N_k] = E[N_k N_k^T] = 4E[\varepsilon_k^2 X_k X_k^T] \tag{3.13}$$

If we assume that the weight vector, W_k, remains near its optimum, W^*, we conclude that ε_k^2 is approximately uncorrelated with the signal vector, so that (3.14) becomes

$$\begin{aligned} \text{cov}[N_k] &\approx 4E[\varepsilon_k^2]E[X_k X_k^T] \\ &\approx 4\xi_{\min} R \end{aligned} \tag{3.14}$$

We need to transform (3.14) into the principal-axis coordinate system, as follows:

$$\begin{aligned} \text{cov}[N_k'] &= \text{cov}[Q^{-1} N_k] \\ &= E[Q^{-1} N_k (Q^{-1} N_k)^T] \\ &= Q^{-1} E[N_k N_k^T] Q \\ &= Q^{-1} \text{cov}[N_k] Q \approx 4\xi_{\min} \Lambda \end{aligned} \tag{3.15}$$

The weight vector covariance in the principal-axis coordinate system. The result is

$$\begin{aligned} \text{cov}[V_k'] &= \frac{\mu}{4} (\Lambda - \mu\Lambda^2)^{-1} \text{cov}[N_k'] \\ &\approx \mu\xi_{\min} (\Lambda - \mu\Lambda^2)^{-1} \Lambda \end{aligned} \tag{3.16}$$

In practical situations the elements of $\mu\Lambda$ tend to be considerably less than 1, so we simplify the expression in (3.16) by neglecting the term $\mu\Lambda^2$ to obtain

$$\begin{aligned} \text{cov}[V_k'] &\approx \mu\xi_{\min}\Lambda^{-1}\Lambda \\ &\approx \mu\xi_{\min} I \end{aligned} \tag{3.17}$$

Thus, transforming back to unprimed coordinates, we have the steady-state noise in the weight vector solution given approximately by

$$\begin{aligned} \text{cov}[V_k] &= Q\,\text{cov}[V_k']Q^{-1} \\ &\approx \mu\xi_{\min} QIQ^{-1} \\ &\approx \mu\xi_{\min} I \end{aligned} \tag{3.18}$$

A further development of cov[V_k] under less restrictive (non-stationary) conditions than those imposed to obtain (3.18) may be found in [6].

3.4 The basic LMS adaptive algorithm

One of the most successful adaptive algorithms is the LMS algorithm developed by Windrows and his coworkers (Windrow et al., 1975a). Instead of computing W_{OPT} in one go as suggested by equation 4.18, in the LMS the coefficients are adjusted from sample to sample in such a way as to minimize the MSE. This amounts to descending along the surface of Figure 3.6 towards its bottom.

The LMS is based on the steepest descent algorithm where the weight vector is updated from sample to sample as follows:

$$W_{k+1} = W_k - \mu\nabla_k \tag{3.19}$$

Where W_k and ∇_k are the weight and the true gradient vectors, respectively, at the kth sampling instant. μ Controls the stability and rate of convergence.

The steepest descent algorithm in Equation 3.18 still requires knowledge of R and P, since ∇_k is obtained by evaluating Equation 3.16. The LMS algorithm is a practical method of obtaining estimates of the filter weights W_k in real time without the matrix inversion in Equation 3.17 or the direct computation of the auto correlation and cross-correlation. The Widrow-Hopf LMS algorithm for updating the weights from sample to is given by

$$W_{k+1} = W_k + 2\mu e_k X_k \tag{3.20a}$$

Where:

$$e_k = y_k - W_k^T X_k \tag{3.20b}$$

Clearly, the LMS algorithm above does not require prior knowledge of the signal statistics (that is the correlation's R and P), but instead uses their instantaneous estimates. The weights obtained by the LMS algorithm are only estimates, but these estimates improve gradually with time as the weights are adjusted and the filter learns the characteristics of the signals. Eventually, the weights converge. The condition for convergence is

$$0\langle\mu\rangle\frac{1}{\lambda\max} \tag{3.21}$$

Where W_k is the maximum eigenvalue of the input data covariance matrix. In practice, W_k never reaches the theoretical optimum (the Wiener solution), but fluctuates about it see figure (3.7).

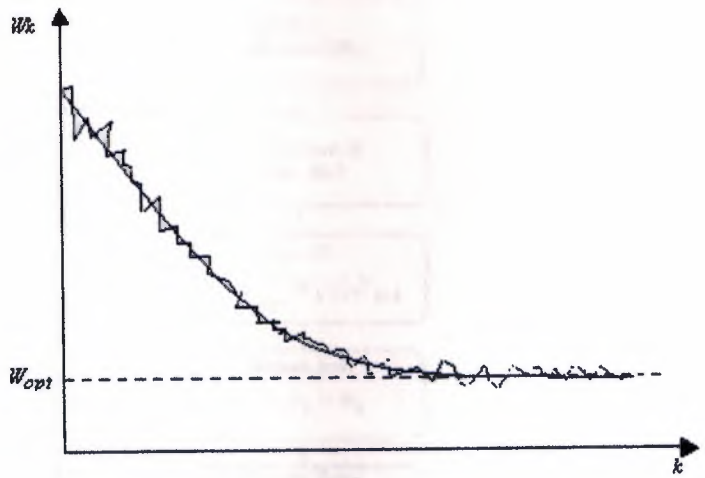


Figure 3.7 An illustration of the variations in the filter weights.

3.5 Implementation of the basic LMS algorithm

The computational procedure for the LMS algorithm is summarized below.

(1) Initially, set each weight Wk (*i*) I= 0,1,……,N-1, to an arbitrary fixed value, such as 0.

For each subsequent sampling instants, k=1,2,3.., carry out steps (2) to (4) below:

(2) Compute filter output.

$$\hat{n}_k = \sum_{i=0}^{N-1} w_k(i) x_{k-i}$$

(3) compute the error estimate

$$e_k = y_k - \hat{n}_k$$

(4) update the next filter weights

$$w_{k+1}(i) = w_k(i) + 2\mu e_k x_{k-i}$$

The simplicity of the LMS algorithm and ease of implementation, evident from above, make it the algorithm of first choice in many real-time systems. The LMS algorithm requires approximately 2N+1 multiplications and 2N+1 additions for each new set of input and output samples. Most signals processors are suited to the mainly multiply-accumulate arithmetic operations involved, making a direct implementation of the LMS algorithm attractive.

The flowchart for the LMS algorithm is given in Figure 3.8 figure 3.9

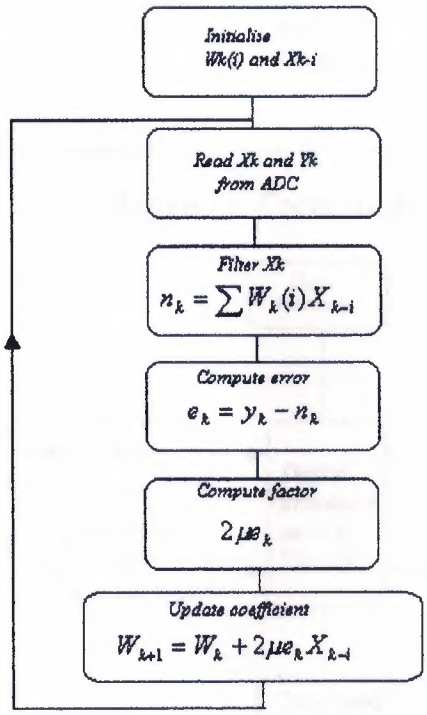
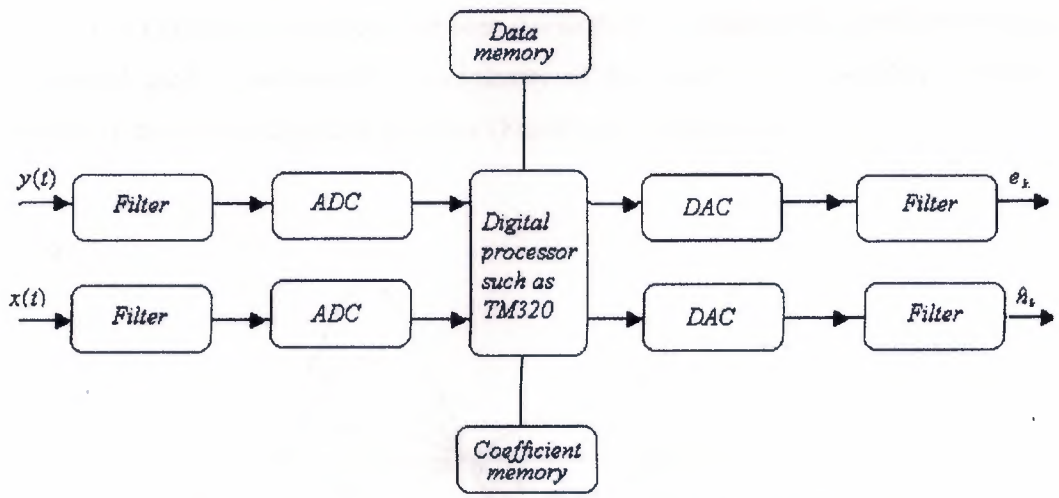


Figure 3.8 Flowchart for the LMS adaptive filter.

```
Inputs:          xk(i)   vector of the latest input samples

                 yk      current contaminated signal sample
                 wk(i)   vector of filter coefficients

Outputs:         ek      current desired output (or error) sample
                 Wk(i)   vector of updated filter coefficients

        /* compute the current error estimate   */

                 ek = yk
                 for i =1 to N do
                     ek = ek – xk(i)*wk(i)
                 end

        /* update filter coefficients           */

                 gk = 2u*ek
                 for i =1 to N do
                     wk(i) = wk(i) + xk(i)*gk
                 end
                  return
```

Figure 3.9 Coding of the LMS adaptive filter.



Figur 3.10 Hardware implementation for real-time LMS adaptive filtering.

And 3.10, respectively, show a pseudo-code for the software and hardware implementations.

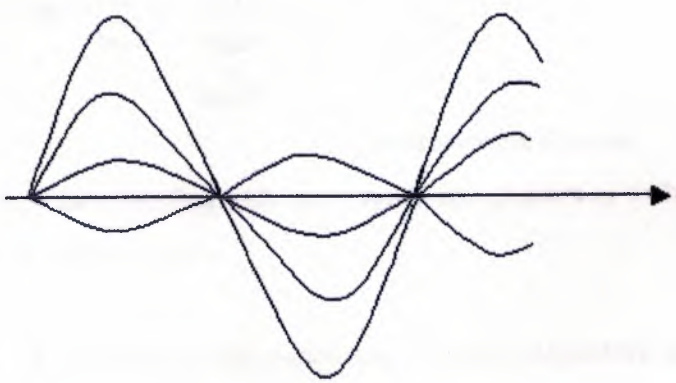
3.6 Practical limitations of the basic LMS algorithm

In practice, several practical problems are encountered when using the basic LMS algorithm, leading to a lowering of performance. Some of the more important problems are discussed here.

3.6.1 Effect of non-stationarity

In a stationary environment, the error performance surface of the filter has a constant shape and orientation, and the adaptive filter merely converges to and operates at or near the optimum point. If the signal statistics change after the weights have converged, the filter responds to the change by re-adjusting its Weights to anew set of optimal values, provided that the change in signal statistics is sufficiently slow for the filter to converge between change. In a nonstationary environment, however, the bottom or minimum point continually moves, and its orientation and curvature may also be changing (see Figure 3.11) thus the algorithm in this case has the task not only of seeking the minimum point of the surface but also of tracking the changing position, leading to significant lowering of performance. (Such as mean, variance, autocorrelation) change with time. Such change can result from, for example, sudden changes due to sporadic interference of short duration (Figure 3.12)
or bad data, and often upset the filter weights).

A number of schemes have been developed to overcome this problem but these in general tend to increase the complexity of the basic LMS algorithm. One such scheme is the time- sequenced adaptive (Ferrari and Windrow, 1981).



(a)

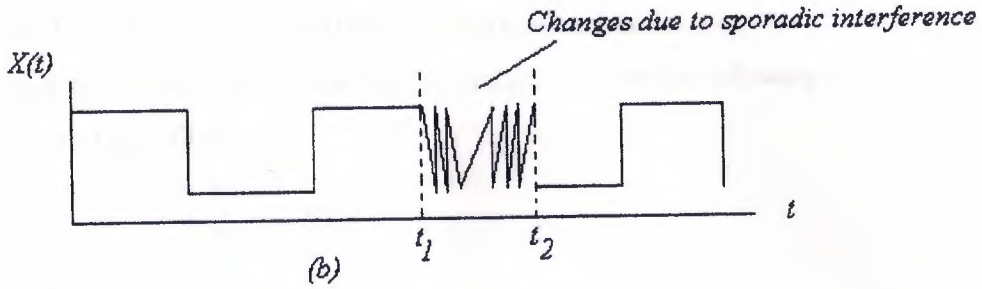


Figure 3.12 an illustration of nonstationary processes
(a) modulated waveform; (b) sporadic interference.

3.6.2 Effects of signals component on the interference input channel

The performance of the algorithm relies on the measured interference signal, $X_k(i)$ being highly correlated with the actual interference, but weakly correlated (theoretically zero) with the desired signal. In most cases, this condition is not met. In some applications, the contaminating input may contain both the undesired interference as well as low-level signal components. Such a situation is illustrated in Figure (3.13). It is shown in Windrow et al. (1975a) that the adaptive noise canceling process still leads to a significant improvement in the desired signal-to-noise ratio in these cases but only at the expense of a small signal

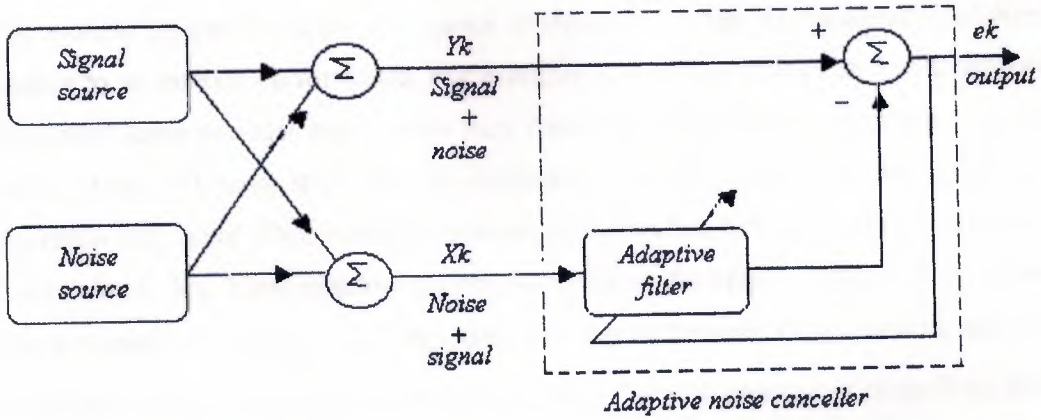


Figure 3.13 Adaptive noise canceling with some signal components in both the desired signal and interference input channels.

Distortion. However, if x_k contains only signals and no noise component what so ever, the desired signal in Y_k may be completely obliterated. Our work in biomedical signal processing confirms their results (Ifeachor et al.1986).

3.6.3 Computer worldlength requirements

The LMS-based FIR adaptive filter is characterized by the following equations:

For the digital filter,

$$n = \sum_{i=0}^{N-1} w_k(i) x_{k-1}$$

(3.22a)

For the adaptive algorithm,

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

(3.22b)

When adaptive filters are implemented in the real world, the filter weights, W_k, and the input variable, X_k and Y_k, are of necessity represented by a finite number of bits. Similarly, the numerical operations involved are carried out using a finite precision arithmetic. The recessive nature of the LMS algorithm means that the word-length will grow without limit and so some of the bits must be discarded before each updated weight is stored. Thus the Y_k, e_k and W_k (i) may differ significantly from their true values. The use of filter weights and results of arithmetic operations with limited accuracy may include (i) possible non-convergence of the adaptive filter whose effects may include (i) possible non-convergence of the adaptive filter to the optimal solution, leading to an inferior performance. For example, if the filter is used as an interference chancellor some residual interference may remain, (ii) the filter outputs may contain noise, which will cause it to fluctuate randomly, and (iii) aperture termination of the algorithm may occur. Thus sufficient number of bits should be used to keep these errors at tolerable levels. Most adaptive system described in the open literature represent the digital signals, x_{k-1} *and* y_k, as fixed point numbers of between 8 and 16 bits, with the coefficients quantized to between 16 and 24 bits. The multipliers used range from 8x8 to 24*16bits, and accumulators of between 16 and 40 bits are used. It appears that for low order filters (up to about 100 coefficient) it is sufficient to store the coefficient to no more than 16-bit accuracy and to use a 16*16 bit multiplier with an accumulator of length 32 bits.

3.6.4 Coefficient drift

In the presence of certain types of inputs (for example narrowband signals), the flitter coefficient may drift from the optimum values and grow slowly, eventually exceeding the permissible wordlength. This is an inherent problem in the LMS algorithm and leads to a long-term degradation in performance. In practice, introducing a leakage factor, which gently nudges the coefficients towards zero, counteracts coefficient drift. Two such schemes are given in Equations 3.27:

$$w_{k+1}(i) = \delta w_k + 2\mu e_k x_{k-i}$$

(3.27a)

$$w_{k+1}(i) = w_k + 2\mu e_k x_{k-i} \pm \delta$$

(3.27b)

small δ, the leakage factor, ensures that drift is contained, but introduce bias in the error term, e_k.

The usefulness of the basic LMS algorithm has been extended by more sophisticated LMS-based algorithm as mentioned before. These include

(1) The complex LMS algorithm which allows the handling of complex data,

(2) The block LMS algorithm which offers substantial computational advantages in some cases faster convergence, and

(3) Time-sequenced LMS algorithm to deal with particular types of non-stationary.

3.7 Fast LMS algorithm

A number of blocks LMS algorithms have been proposed which offer substantial computational saving especially when the number of filter coefficients is large. The computational is saving result from processing the data in blocks instead of one sample at a time. Frequency domain implementations of the block LMS exploit the computational advantage of the fast Fourier transforms (FFT) in performing convolutions (Mansour and Gray, 1982). An efficient frequency domain filter is depicted in Figure 3.14.

3.8 Recursive least squares algorithm

The RLS algorithm is based on the well known least square method (Figure 3.15). An output signal, Y_k, is measured at the discrete time, k, in response to a set of

input signals, $X_k(i)$, i= 1,2,3,……n. The input and output signals are related by the simple regression model

$$y_k = \sum_{i=0}^{n-1} w(i)x_k(i) + e_k$$

(3.29)

Where e_k represents measurements errors or other effects that cannot be accounted for, and $w(i)$ represents the proportion of the it input that is contained in the primary signal, Y_k. The problem in the LS method is, given the $X_k(i)$ and Y_k above, to obtain estimates of

$w(0)$ to $w(n-1)$.

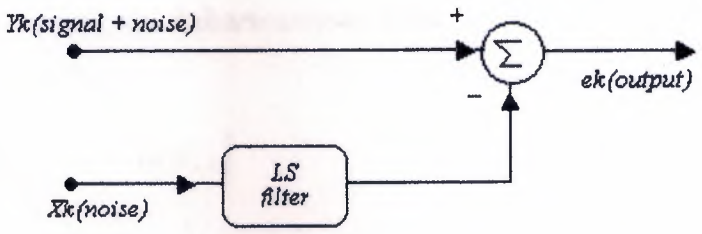


Figure 3.15 An illustration of the basic idea of the least-squares method.

Optimum estimates (in the least square sense) of the filter weights, w(i),are given by

$$W_k = \left[X_m^T X_m\right]^{-1} X_m^T Y_m$$

(3.30)

Where $Y_m, W_m and X_m$ are given by

$$Y_m = \begin{bmatrix} y0 \\ y1 \\ y2 \\ . \\ . \\ y_{m-1} \end{bmatrix} \quad X_m = \begin{bmatrix} x^T(0) \\ x^T(1) \\ x^T(2) \\ . \\ . \\ x^T{}_{(m-1)} \end{bmatrix} \quad W_m = \begin{bmatrix} w(0) \\ w(1) \\ w(2) \\ . \\ . \\ w(n-1) \end{bmatrix}$$

$$x^T(k) = \left[x_k(0)x_k(1).....x_k(n-1)\right], \qquad k=0,1,2,3,\ldots\ldots\ldots\ldots m-1$$

The suffix m indicates that each matrix above is obtained using all m data points and T indicates transposition. Equation 3.30 gives the OLS estimates of Wm which can be

obtained using any suitable matrix inversion technique. The filter output is then obtained as

$$n_k = \sum_{i=0}^{n-1} w(i) k_{k-1,} \qquad \text{k=1,2,3,.....,m} \tag{3.31}$$

The computation of Wm. in Equation 3.30 requires the time-consuming computation of the inverse matrix. Clearly, the LS method above is not suitable for real-time or on- line filtering. In practice, when continuous data is being acquired and we wish to improve our estimate of Wm. using the new data, recursive methods are preferred. With the recursive least squares algorithm the estimates of Wm and to allow the tracking of slowly varying signal characteristics. Thus

$$W_k = W_{k-1} + G_k e_k \tag{3.32a}$$

$$P_k = \frac{1}{\gamma}\left[P_{k-1} - G_k X^T(k) P_{k-1}\right] \tag{3.32b}$$

where

$$G_k = \frac{P_{k-1} x(k)}{\alpha_k}$$

$$e_k = y_k - X^T(k) W_{k-1}$$

$$\alpha_k = \gamma + X^T(k) P_{k-1} X(k)$$

P_k is essentially a recursive way of computing the inverse matrix $\left[X_k^T X_k\right]^{-1}$.

The argument *k* emphasizes the fact that the quantities are obtained at each sample point. γ is referred to as the forgetting factor . This weighting scheme reduces to that of the LS when $\gamma = 1$. Typically, γ is between 0.98 and 1. Smaller values assign too much weight to the more recent data, which leads to wildly fluctuating estimates. The number of previous samples that significantly contribute to the value of W_k at each sample point is called the asymptotic sample length (ASL) given by

$$\sum_{k=1}^{\infty} \gamma^k = \frac{1}{1-\gamma} \tag{3.33}$$

This effectively defines the memory of the RLS filter. When γ=1, that is when it corresponds to the LS, the filter has an infinite memory.

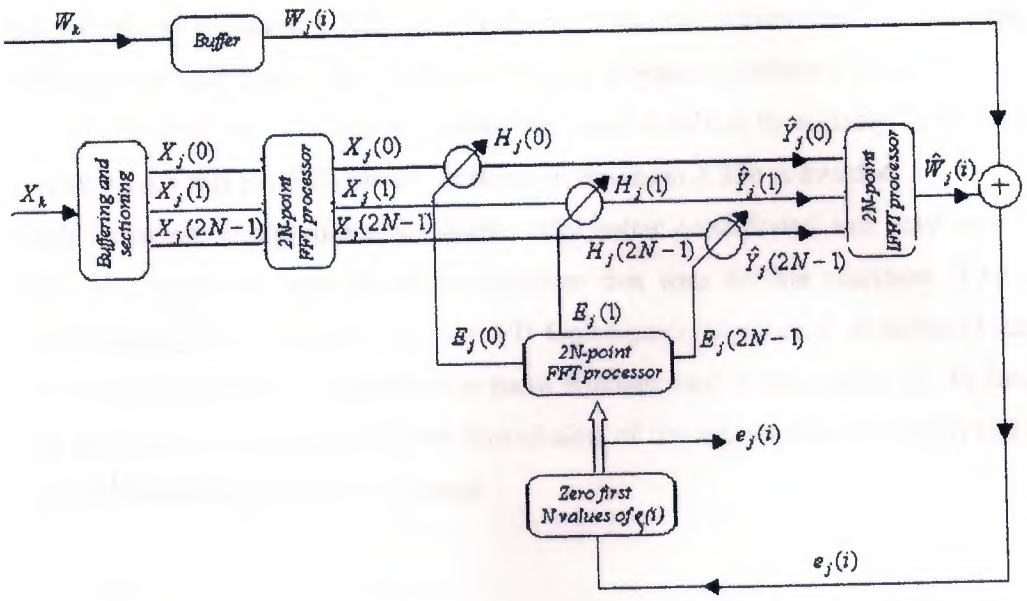


Figure 3.14 Simplified block diagram of a frequency domain LMS filter

3.8.1 Limitations of the recursive least squares algorithm

The RLS method is very efficient and involves exactly the same number of arithmetic operations between samples as W_k and P_k in Equations 3.32 a and b; have fixed dimensions. This is an important requirement for efficient real-time filtering. There are, however, two main problems that may be encountered when the RLS algorithm is implemented directly. The first, referred to as blow-up', results if the signal X_k (i) is zero for a long time, when the matrix P_k will grow exponentially as a result of division by γ (which is less than unity) at each sample point:

$$\lim_{k\to\infty} P_k = \lim_{k\to\infty}\left(\frac{P_{k-1}}{\gamma_{k-1}}\right) \tag{3.34}$$

The second problem with the RLS is its sensitivity to computer round off errors, which results in a negative definite P matrix and eventually to instability. For successful estimation of W, it is necessary that the P be positive semi definite which is equivalent to requiring in the LS method that the matrix $X^T X$ be inevitable, but, because of differencing of terms in Equation 4.32b, positive definiteness of P cannot be guaranteed. This problem can be worse in multi parameter models, especially if the variables are linearly dependent and when the algorithm is implemented on a small system with a finite word length, when the algorithm has iterated for a long time the two terms in the

parentheses in Equation 4.32b are very nearly equal and subtraction of such terms in a finite word length system may lead to errors and a negative definite P_k matrix.

The problem of numerical instability may be solved by suitably factorizing the matrix P such that the differencing of terms in Equation 3.32b is avoided.
Such factorization algorithms are numerically better conditioned and have accuracies that are comparable with the RLS algorithm that uses double precision. Two such algorithms are the square root and the UD factorization algorithms. In terms of storage and computation the UD algorithm is more efficient, and is thus preferred. In fact, the UD algorithm is a square-root-free formulation of the square root algorithm and thus shares the same properties as the latter.

CHAPTER FOUR
FINITE-PRECISION EFFECTS

4.1 Overview

In order to simplify the discussion of finite-precision effects on the performance of the LMS algorithm. we will depart from the practice followed in previous chapters, and assume that the input data and therefore the filter coefficients are all *real valued.* This

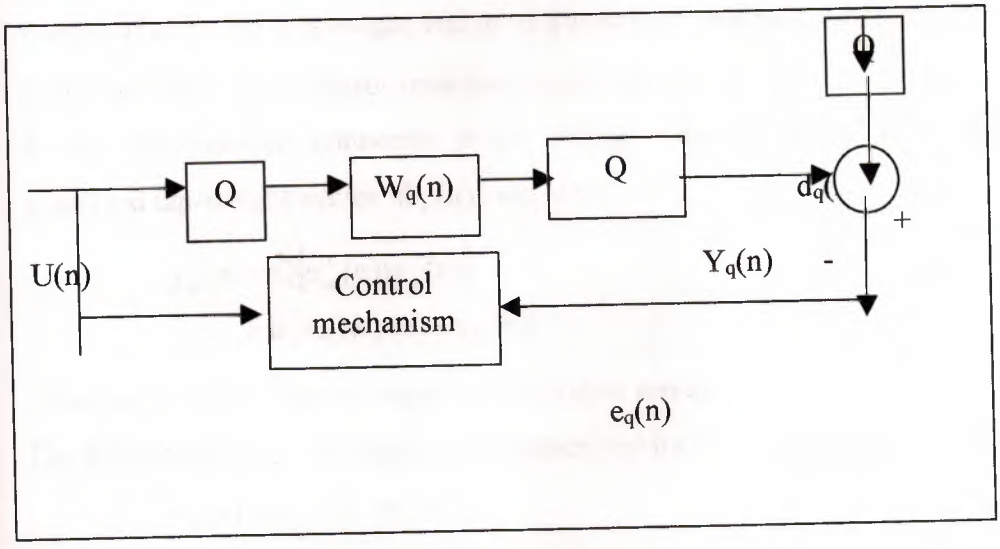


Figure 4.1 Block diagram representation of the finite-precision form of LMS algorithm

Assumption, made merely for convenience of presentation, will in no way affect the validity of the findings presented in this section.

A block diagram of the finite-*precision least-'meant-square (LMS) algorithm* depicted in Fig 4.1. Each of the blocks (operators) labeled Q represents a quentizer. Each one introduces a quantization, or round-off error of its own. Specifically, we ma the input-output relations of the quantizers operating in Fig. 4.1 as follows:

1. For the input quantizer connected to u(n) we have

$$\begin{aligned} U_q(n) &= Q[U(n)] \\ &= U(n) + \eta_u(n) \end{aligned} \tag{4.1}$$

Where $\eta_u(n)$ is the input quentization error vector.

2. For the quantizer connected to the desired response d(n), we have

$$\begin{aligned} d_q(n) &= Q[d(n)] \\ &= d(n) + \eta_d(n) \end{aligned} \tag{4.2}$$

Where $\eta_d(n)$ is the desired response quentization error.

3. For the quantized tap-weight vector $\hat{w}_q(n)$, we write

$$\begin{aligned} \hat{w}_q(n) &= Q[\hat{w}(n)] \\ &= \hat{w}(n) + \Delta\hat{w}(n) \end{aligned} \tag{4.3}$$

where $\hat{w}(n)$ is the tap-weight vector in the infinite-precision LMS algorithm ,and $\Delta\hat{w}(n)$ is the tap-weight error vector resulting from quantization.

4. For the quantizer connected to the output of the transversal filter represented by the quantized tap-weight vector $\hat{w}_q(n)$, we write

$$\begin{aligned} y_q(n) &= Q[u_q^T(n)\hat{w}_q(n)] \\ &= u_q^T(n)\hat{w}_q(n) + \eta_y(n) \end{aligned} \tag{4.4}$$

Where $\eta_y(n)$ is the *filtered output quantizotion error.*

The finite-precision LMS algorithm is described by the following pair of relations:

$$e_q(n) = d_q(n) - y_q(n) \tag{4.5}$$

$$\hat{w}_q(n+1) = \hat{w}_q(n) + Q[\mu e_q(n)u_q(n)] \tag{4.6}$$

where $y_q(n)$ is itself defined in Equation (4.4). The quantizing operation indicated on the right-hand side of Equation (4.6) is not shown explicitly in Fig. 4.1; nevertheless. It is basic to the operation of the finite-precision LMS algorithm. The use of Equation (4.6) has the following practical implication. The product $\mu e_q(n)u_q(n)$, representing a scaled version of the gradient vector estimate, is quantized *before* addition to the contents of the *tap-weight accumulator*. Because of hardware constraints, this form of digital implementation is preferred to the alternative method of operating the tap-weight accumulator in double precision and then quantizing the tap weight to single precision at the accumulator output.

In a statistical analysis of the finite-precision LMS algorithm, it is customary to make the

following assumptions:

1. The input data are properly scaled so as to *prevent overflow* of the elements of the quantized tap-weight vector $\hat{w}_q(n)$ and the quantizcd output $y_q(n)$ during the filtering operation.

2. Each data sample is represented by B_D bits plus sign, and each tap weight is represented by B_W bits plus sign. Thus, the quantization error associated with a B_D-plus-sign bit number (i.e.. data sample) has the variance

$$\sigma_D^2 = \frac{2^{-2B_D}}{12} \tag{4.7}$$

Similarly, the quantization error associated with a B_w plus-sign hit number (i.e.. tap weight) has the variance

$$\sigma_w^2 = \frac{2^{-2Bw}}{12} \tag{4.8}$$

3. The elements of the input quantization error vector $\eta_u(n)$ and the desired response quantization error $\eta_d(n)$ are *white-noise* sequences. independent of the signals and from each other. Moreover, they have zero mean and variance σ_D^2.

4. The output quantization error $\eta_y(n)$ is a white-noise sequence, independent of the input signals and other quantization errors. It has a mean of zero and a variance equal to $c\sigma_D^2$, where c is a constant that depends on the way in which the inner product $u_q^T(n)\hat{w}_q(n)$ is computed. If the individual scalar products in $u_q^T(n)\hat{w}_q(n)$ are all computed without quantization, then summed, and the final result is quentized in B_D bits plus sign, the constant c is unity and the variance of $\eta_y(n)$ is σ_D^2 as defined in Eq. (4.7). If, on the other hand, the individual scalar products in $u^T(n)\hat{w}_q(n)$ are quantized and then summed, the constant c is M and the variance of $\eta_y(n)$ is $M\sigma_D^2$ where M is the number of taps in the transversal filter implementation of the LMS algorithm.

5. The independence theory dealing with the infinite- precision LMS algorithm, is invoked.

4.2 Total Output Mean-squared Error

The filtered output $y_q(n)$, produced by the finite-precision LMS algorithm, presents a quentized *estimate* of the desired response. The *total Output error* is therefore equal to the difference *d(n)* - $y_q(n)$. Using Equation (4.4), we may therefore express this error as

$$\begin{aligned} e_{total}(n) &= d(n) - y_q(n) \\ &= d(n) - u_q^T(n)\hat{w}_q(n) - \eta_y(n) \end{aligned} \tag{4.9}$$

Substituting Equation (4.1) and (4.3) in Equation (4.9), and ignoring all quantization error terms higher than first order, we get

$$e_{total}(n) = \left[d(n) - u^T(n)\hat{w}(n)\right] - \left[\Delta\hat{w}^T(n)u(n) + \eta_u^T(n)\hat{w}(n) + \eta_y(n)\right] \tag{4.10}$$

The term inside the first set of square brackets on the right-hand side of Equation (4.10) is the estimation error *e(n)* in the infinite-precision LMS algorithm. The term inside the second set of square brackets is entirely due to quantization errors in the finite-precision algorithm. Because of assumptions 3 and 4 (i.e., the quantization errors η_u and η_y are independent of the input signals and of each other), the quantization error-related terms $\Delta\hat{w}^T(n)u(n)$,and η_y are uncorrelated with each other. Basically, for the same reason, the infinite-precision estimation error *e(n)* is uncorrelated with both $\eta_u^T(n)\hat{w}(n)$ and $\eta_y(n)$.

$$E\left[e(n)\Delta\hat{w}^T(n)u(n)\right] = E\left[\Delta\hat{w}^T(n)\right]E\left[e(n)u(n)\right]$$

Moreover, by invoking this same independence assumption. We may show that the expectation $E[\Delta\hat{w}(n)]$is zero. Hence. e(n) and $\Delta\hat{w}^T(n)u(n)$ are also uncorrelated.

In other words, the infinite-precision estimation error e(n) is uncorrelated with all quantization-error-related terms $\Delta\hat{w}^T(n)u(n), \eta_u^T(n)\hat{w}(n)$, and $\eta_y(n)$ in Equation (4.10).

Using these observations, and assuming that the step-size parameter μ is small, shown in Caraiscos and Liu (l984) that the total output *mean-squared error* produced the finite-precision algorithm has the following *steady-state* structure:

$$E\left[e_{total}^2(n)\right] = J_{min}(1+M) + \zeta_1(\sigma_w^2, \mu) + \zeta_2(\sigma_D^2) \quad (4.11)$$

The first term $J_{min}(1 + M)$ on the right-hand side of Equation (4.11) is the mean-squared error of the infinite-precision LMS algorithm. In particular, J_{min} is the minimum mean squared

Error of the optimum Wiener filters, and M is the misadjustment of the infinite-precision LMS algorithm. The second term $\zeta_1(\sigma_w^2, \mu)$ arises because of the error $\Delta\hat{w}(n)$ in the quantized tap-weight vector $\hat{w}_q(n)$. This contribution to the total output mean-squared error is inversely proportional to the step-size parameter μ. The third term $\zeta_2(\sigma_D^2)$ arises because of two quantization errors: the error $\eta_u(n)$ in the quantized input vector $u_q(n)$ and the error $\eta_y(n)$ in the quantized filter output $y_q(n)$. However, unlike $\zeta_1(\sigma_w^2, \mu)$, this final contribution to the total output mean-squared error is, to a first order of approximation, independent of the step-size parameter μ.

We know that decreasing μ reduces the misadjustment M and thus leads to an improved performance of the algorithm. In contrast, the inverse dependence of the contribution $\zeta_1(\sigma_w^2, \mu)$ on μ in Equation (4.11) indicates that decreasing μ has the effect of increasing the deviation from infinite-precision performance. In practice, therefore. The step-size parameter μ may only be decreased to a level at which the degrading effects of quantization errors in the tap weights of the finite-precision LMS algorithm become significant.

Since the misadjustment M decreases with μ and the contribution $\zeta_1(\sigma_w^2, \mu)$

Increases with reduced μ we may (in theory) find an optimum value of μ for which the total output mean-squared error in Equation (4.11) is minimized. However, it turns out that this minimization results in an optimum value μ_0 for the step-size parameter μ that is too small to be of practical value. In other words, it does not permit the LMS algorithm to converge completely. Indeed. Equation (4.11) for calculating the total output mean-squared error is valid only for a μ that is well in excess of μ_0. Such a choice of μ is necessary so as to prevent the occurrence of a phenomenon known as stalling, described later in the section.

4.3 Leaky LMS Algorithm

To further stabilize the digital implementation of the LMS algorithm, we may use a technique known as leakage. Basically, leakage prevents the occurrence of overflow in a limited-precision environment by providing a compromise between minimizing the mean squared error and containing the energy in the impulse response of the adaptive filter. However, the prevention of overflow is attained at the expense of an increase in hardware cost and at the expense of degradation in performance compared to the infinite- precision form of the conventional LMS algorithm.

In the leaky LMS algorithm, the cost function

$$J(n) = e^2(n) + \alpha\|\hat{w}(n)\|^2 \tag{4.12}$$

Is minimized with respect to the tap-weight vector w(n), where α is a positive control parameter. The first term on the right-hand side of Equation (4.12) is the squared estimation error, and the second term is the energy in the tap-weight vector $\hat{w}(n)$. The minimization described herein (for real data) yields the following time update for the tap-weight vector

$$\hat{w}(n+1) = (1-\mu\alpha)\hat{w}(n) + \mu e(n)u(n) \tag{4.13}$$

Where α is a constant that satisfies the condition

$$0 \le \alpha < \frac{1}{\mu}$$

Except for the leakage factor (1 - μα) associated with the first term on the right-hand side of Equation (4.13), the algorithm is of the same mathematical form as the conventional LMS algorithm.

Note that the inclusion of the leakage factor (1 - μα) in Equation. (4.13) has the equivalent effect of adding a white-noise sequence of zero mean and variance α to the input process

u(n). This suggests another method for stabilizing a digital implementation of the LMS algorithm. Specifically. A relatively weak white-noise sequence (of variance α). Known as *dither,* is added to the input process *u*(n). And samples of the combination are then used as tap inputs (Werner. 1983).

4.4 Stalling Phenomenon

There is another phenomenon. Known as the stalling or lock-up phenomenon, not evident from Equation (4.11), which may arise in a digital implementation of the LMS algorithm. This phenomenon occurs when the gradient estimate is not sufficiently noisy. To be specific. A digital implementation of the LMS algorithm stops adapting or stalls, whenever the correction term $\mu e_q(n)u_q(n-i)$ for the ith tap weight in the update equation is smaller in magnitude than the least significant bit (LSB) of the tap weight. As shown by (Gitlin et al.. 1973)

$$\left|\mu e_q(n_0)u_q(n_0 - i)\right| \le \text{LSB} \tag{4.14}$$

Here, n_0 is the time at which the ith tap weights stops adapting. Suppose that the condition of Equation (4.14) is first satisfied for the ith tap eight. To a first order of approximation. We may replace $u_q(n_0 - i)$ by its root -mean-square (rms) value. A_{rms}. Accordingly, using this value in Equation (4.14), we get the following relation for the rms value of the quantized estimation error when adaptation in the digitally implemented LMS algorithm stops:

$$\left|e_q(n)\right| \le \frac{\text{LSB}}{\mu A_{rms}} = e_D(\mu) \tag{4.15}$$

The quantity $e_D(\mu)$, defined on the right-hand side of (4.15). is called the digital residual error.

To prevent the algorithm-stalling phenomenon due to digital effects, the digital residual error $e_D(\mu)$ must be made as small as possible. According to the definition of Equation (4.15), this requirement may be satisfied in one of two ways:

1. The least significant bit (LSB)is reduced by picking a sufficiently large number of bits for the digital representation of each tap weight reduces
2. The step-size parameter μ is made as large as possible, while still guaranteeing convergence of the algorithm.

Another method of preventing the stalling phenomenon is to insert *dither* at the input of the quantizer that feeds the tap-weight accumulator (Sherwood and Bershad, 1987). Dither is a

random sequence that essentially "linearizes" the quantizer. In order word, the addition of dither guarantees that the quantizer input is *noisy* enough for the gradient quentization error vector η_w, to be again modeled as white noise (i.e.. the element of η_w are uncorrelated in time and with each other, and have a common variance σ_w^2). When dither is used in the manner described here, it is desirable to minimize its effect on the overall operation of the LMS algorithm. This is commonly achieved by shaping the power spectrum of the dither so that the algorithm at its output effectively rejects it.

4.5 Parameter Drift

In addition to the numerical problems associated with the LMS algorithm. There is one other rather subtle problem that is encountered in practical applications of the algorithm Specifically. Certain classes of input excitation can lead to parameter drift, that is parameter estimates or tap weights in the LMS algorithm attain arbitrarily large values despite bounded inputs. Bounded disturbances. And hounded estimation errors (sethares et al.,1986). Although such an unbounded behavior may he unexpected, it is possible for the parameter estimates to drift to infinity while all the signals observable in the algorithm converge to zero. Parameter drift in the LMS algorithm may be viewed as a hidden form of instability, since the tap weights represent internal" variables of the algorithm it may result in new numerical problems, increased sensitivity to unmodeled disturbances, and degraded long-term performance.

In order to appreciate the subtleties of the parameter drift problem. We need to introduce some new concepts relating to the parameter space. We therefore digress briefly from the issue at hand to do so.

A sequence of information-bearing tap-input vectors u(n) for varying time n may be used-to partition the real M-dimensional parameter space R^M into orthogonal subspace where M is the number of tap weights (i.e.. the available number of degrees of freedom). The aim of this partitioning is to convert the stability analysis of an adaptive filtering algorithm.

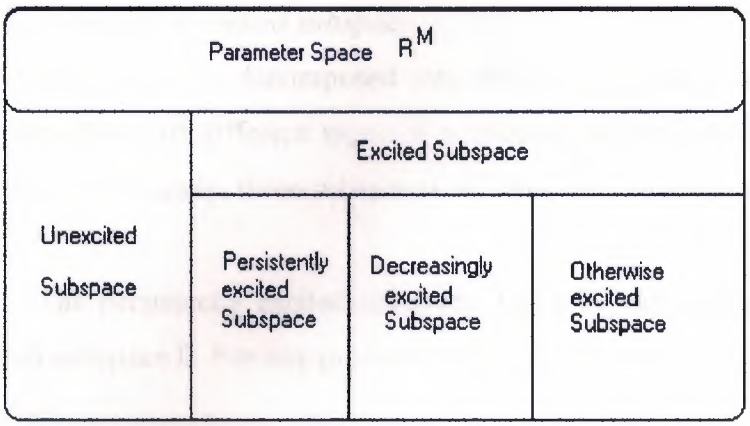


Figure 4.2 Decomposition of parameter space R^M, based on excitation.

(e.g.. the LMS algorithm) into simpler subsystems and thereby provide a closer linkage between the transient behavior of the parameter estimates and the filter excitations. The partitioning we have in mind is depicted in Fig. 4.2. In particular, we may identify the following subspaces of R^M

1. on-excited subspace. Let the *M-by-* 1 vector z be any element of the parameter space R^M, which satisfies two conditions:

- The Euclidean norm of the vector z is 1; that is,

$$\|z\| = 1$$

- The vector z is orthogonal to the tap-input vector u(n) for all but a finite number of n; that is,

$$z^T u(n) \neq 0, \quad \text{only finitely often} \qquad (4.16)$$

Let $\mathfrak{f}_u$ denote the subspace of R^M that is spanned by the set of all such vectors z. The subspace $\mathfrak{f}_u$ is called the unexcited subspace in the sense that it spans those directions in the parameter space R^M that are excited only *finitely often.*

2. The excited subspace. Let $\mathfrak{f}_e$ denote the orthogonal complement *of* the unexcited Subspace $\mathfrak{f}_u$. Clearly, $\mathfrak{f}_e$ is also a subspace of the parameter space R^M. It contains those directions in the parameter space R^M that are excited *infinity often.* Thus, except for the null vector, every element z *belonging to the subspace* $\mathfrak{f}_e$ satisfies the condition

$$z^T u(n) \neq 0, \qquad \text{Infinitely often} \qquad (4.17)$$

The subspace $\mathcal{E}_e$, is called the *excited subspace.*

The subspace $\mathcal{E}_e$ may itself be decomposed into three orthogonal subspaces of its own, depending on the effects of different types of excitation on the behavior of the adaptive filtering algorithm. Specifically, three subspaces of $\mathcal{E}_e$ may be identified as follows (Sethares et al.. 1986):

- The persistently excited subspace. Let z be any vector of unit norm that lies in the excited subspace $\mathcal{E}_e$. For any positive integer m and any $\alpha > 0$, choose the vector z such that we have

$$z^T u(i) > \alpha \quad \text{for } n \le i \le n + m \text{ and for all but a finite number of n} \tag{4.18}$$

Given the integer m and the constant α, let $\mathcal{E}_p(m,\alpha)$ be the subspace spanned by all such vectors z that satisfy the condition of (4.18). There exist a finite m_0 and a positive α_0 for which the subspace $\mathcal{E}_e(m_0,\alpha_0)$ is maximal. In other words $\mathcal{E}_p(m_0,\alpha_0)$ contains $\mathcal{E}_p(m,\alpha)$ for all $m > 0$ and for all $\alpha > 0$. The subspace $\mathcal{E}_p \equiv \mathcal{E}_p(m_0,\alpha_0)$ is called the persistently excited subspace; and m_0 is called the interval of excitation. For every direction z that lies in the persistently excite subspace $\mathcal{E}_p$ there is an excitation of level α_0 at least once in all but a finite number of intervals of length m_0. In the persistently excited subspace, we are therefore able to find a tap-input vector u(n) rich enough to excite all the internal modes that govern the transient behavior of the adaptive filtering algorithm being probed (Narendra and Annaswamy, 1989).

- The subspace of decreasing excitation. Consider a sequence u(i) for which we have

$$\left(\sum_{i=1}^{\infty} |u(i)|^p \right)^{1/p} < \infty \tag{4.19}$$

Such a sequence is said to be an element of the normed linear space l^p for $1 < p < \infty$. The norm of this new space is defined by

$$\|u\|_p = \left(\sum_{i=1}^{\infty} |u(i)|^p \right)^{1/p} \tag{4.20}$$

Note that if the sequence u(i) is an element of the normed linear space l^p for $1 < p \le \infty$, then

$$\lim_{n \longrightarrow \infty} (n) = 0 \tag{4.21}$$

Let z be any unit-norm vector z that lies in the excited subspace $\int_e$ such that for $1 < p \leq \infty$ the sequence $z^T u(n)$ lies in the normed linear space l^P . Let $\int_d$ be the subspace that is spanned by all such vectors z. The subspace $\int_d$ is called the subspace of decreasing excitation in the sense that each direction of $\int_d$ is decreasingly excited, for any vector $z \neq 0$, the two conditions

$$\left|z^T u(n)\right| = \alpha > 0. \qquad \text{infinitely often}$$

and

$$\lim_{n \to \infty} z^T u(n) = 0$$

Cannot be satisfied simultaneously. In actual fact, we find that the subspace of decreasing excitation $\int_d$ is orthogonal to the subspace of persistent excitation $\int_p$

The otherwise excited subspace. Let $\int_p \cup \int_d$ denote the union of the persistently excited subspace $\int_p$ and the subspace of decreasing excitation $\int_d$. Let $\int_o$ denote the orthogonal complement of $\int_p \cup \int_d$ that lies in the excited subspace $\int_e$. The subspace $\int_o$ is called the *otherwise excited subspace.* Any vector that lies in the subspace $\int_o$ is not unexciting, not persistently exciting, and not in the normal linear space L^P for any finite p. An example of such a signal is the sequence

$$z^T u(n) = \frac{1}{\ln(1+n)} \qquad n = 1,2,3,..... \tag{4.22}$$

Returning to our discussion of the parameter drift problem in the LMS algorithm. We find that for bounded excitations and bounded disturbances. In the case of unexcited and persistently exciting subspaces the parameter estimates resulting from the application of the LMS algorithm are indeed bounded. It however, in the decreasing and otherwise excited cases, parameter drift may occur (Sethares et al., 1986). A common method of counteracting the parameter drift problem in the LMS algorithm is to introduce leakage into the tap-weight update equation of the algorithm. Here is another reason for using the leaky LMS algorithm that was described previously.

4.6 Recursive Least-Squares Algorithm

The recursive least-squares (RLS) algorithm offers an alternative to the LMS algorithm as a tool for the solution of adaptive filtering problems. We know that the RLS algorithm is characterized by a fast rate of convergence that is relatively insensitive to the eigenvalue spread of the underlying correlation matrix of the input data, and a negligible misadjusiment (zero for a stationary environment without disturbances). Moreover, although it is computationally demanding (in the sense that its computational complexity is on the order of M^2. where M is the dimension of the tap-weight vector), the mathematical formulation and therefore implementation of the RLS algorithm is relatively simple. However, there is a numerical instability problem to be considered when the RLS algorithm is implemented in finite-precision arithmetic.

Basically, numerical instability or explosive divergence of the RLS algorithm is of a similar nature to that experienced in Kalman filtering, of which the RLS algorithm is a special case. Indeed, the problem may be traced to the fact that the time-updated matrix P(n) in the Riccati equation is computed as the difference between two nonnegative definite matrices. Accordingly, explosive divergence of the algorithm occurs when the matrix P(n) loses the property of positive definiteness or Hermitian symmetry.

Table 4.1. Summary Of A Computationally Efficient Symmetry-Preserving Version Of The RLS Algorithm

Initialize the algorithm by setting

$$P(0) = \delta^{-1}I. \qquad \delta = \text{small positive constant}$$

$$\hat{w}(0) = 0$$

For each instant of time, n = 1,2,....., compute

$$\pi(n) = P(n-1)u(n)$$

$$r(n) = \frac{1}{\lambda + u^H(n)\pi(n)}$$

$$K(n) = r(n)\pi(n)$$

$$\xi(n) = d(n) - \hat{w}^H(n-1)u(n)$$

$$\hat{w}(n) = \hat{w}(n-1) + K(n)\xi(n)$$

$$P(n) = \mathrm{Tri}\left[\lambda^{-1}\left[P(n-1) - K(n)\pi^H(n)\right]\right]$$

This is precisely what happens in the usual formulation of the RLS algorithm Described in Table 4.1 (Verhaegen. 1989).

How then can the RLS algorithm be formulated so that the Hermitian symmetry of the matrix P(*n*) is preserved despite the presence of numerical errors? For obvious practical reasons, it would also be satisfying if the solution to this fundamental question can be attained in a computationally efficient manner. With these issues in mind, we present in Table 4.1 a particular version of the RLS algorithm from Yang (1994), which describes a computationally efficient procedure for preserving the Hermitian symmetry of P(*n*) by design. The improved computational efficiency of this algorithm is achieved because it computes simply the upper / lower triangular part of the matrix P(n), as signified by the operator Tri{ }, and then fills in the rest of the matrix to preserve Hermitian symmetry Moreover. Division by λ is replaced by multiplication with the precompuled value of λ^{-1}

4.7Error Propagation Model

According to the algorithm of Table 4.1. The recursions involved in the computation of

P(n) proceed as follows:

$$\pi(n) = P(n-1)u(n) \tag{4.23}$$

$$r(n) = \frac{1}{\lambda + u^{H}(n)\pi(n)} \tag{4.24}$$

$$k(n) = r(n)\pi(n) \tag{4.25}$$

$$P(n) = \mathrm{Tri}\left[\lambda^{-1}\left[P(n-1) - k(n)\pi^{H}(n)\right]\right] \tag{4.26}$$

Where λ is the exponential weighting factor. Consider the propagation of a *single* quantization error at time *n-1* to subsequent recursions. Under the assumption that no other quantization errors are made. In particular, let

$$P_q(n-1) = Pn-1) + \eta_p(n-1) \tag{4.27}$$

Where the error matrix η_p(n-1) arises from the quantization of P(n - 1). The corresponding quantized value of $\pi(n)$ is

$$\pi_q(n) = \pi(n) + \eta_p(n-1)u(n) \tag{4.28}$$

Let $r_q(n)$ denote the quantized value of r(n) Using the defining equation (4.28), we may write

$$\begin{aligned} r_q(n) &= \frac{1}{\lambda + u^H(n)\pi_q(n)} \\ &= \frac{1}{\lambda + u^H(n)\pi(n) + u^H(n)\eta_p(n-1)u(n)} \\ &= \frac{1}{\lambda + u^H(n)\pi(n)}\left(1 + \frac{u^H(n)\eta_p(n-1)u(n)}{\lambda + u^H(n)\pi(n)}\right)^{-1} \\ &= \frac{1}{\lambda + u^H(n)\pi(n)} - \frac{u^H(n)\eta_p(n-1)u(n)}{\left(\lambda + u^H(n)\pi(n)\right)^2} + O(\eta_p{}^2) \\ &= r(n) - \frac{u^H(n)\eta_p(n-1)u(n)}{\left(\lambda + u^H(n)\pi(n)\right)^2} + O(\eta_p{}^2) \end{aligned} \tag{4.29}$$

Where $O(\eta_p^2)$ denotes the order of magnitude $\|\eta_p\|^2$.

In an ideal situation. The infinite-precision scalar quantity r(n) is nonnegative, taking on values between zero and $1/\tilde{\lambda}$. On the other hand. if $u^H(n)\pi(n)$ is small compared to λ and λ itself is small enough compared to 1. Then according to Equation (4.29), in a finite-precision environment it is possible for the quantized quality $r_q(n)$ to take on a negative value large in magnitude than $1/\lambda$. When this happens. The RLS algorithm exhibits explosive divergence (Bottomley and Alexander. I 989).

The quantized value of the gain vector k(n) is written as

$$\begin{aligned} k_q(n) &= r_q(n)\pi_q(n) \\ &= k(n) + \eta_k(n) \end{aligned} \tag{4.30}$$

Where $\eta_k(n)$is the gain vector quentization error, defined by

$$\eta_k(n) = r(n)(I - k(n)u^H(n))\eta_p(n-1)u(n) + O(\eta_p{}^2) \tag{4.31}$$

Finally, using Equation (4.26), we find that the quantization error incurred in computing in updated inverse-correlation matrix $P(n)$ is

$$\eta_p(n) = \lambda^{-1}(I - k(n)u^H(n))\eta_p(n-1)(I - k(n)u^H(n))^H \tag{4.32}$$

where the term $O(\eta_p^2)$ has been ignored.

On the basis of Equation (4.32), it would be tempting to conclude that $\eta_p^H(n) = \eta_p(n)$ and therefore the RLS algorithm of Table 4.1 is *Hermitian*-symmetry preserving, if we can assume that the condition $\eta_p^H(n-1) = \eta_p(n-1)$ holds at the previous iteration. We are justified in making this assertion by virtue of the fact there is no blow-up in this formulation of the RLS algorithm, as demonstrated in what follows (it is also assumed that there is no stalling).

Equation (4.32) defines the error propagation mechanism for the RLS algorithm summarized in Table 4.1 on the basis of a single quantization error in P(*n*-1).

The matrix I- *k(n) U^H(n)* plays a crucial role in the way in which the single quantization error $\eta_p(n\text{-}1)$ propagates through the algorithm.

$$k(n) = \Phi^{-1}(n)u(n) \tag{4.33}$$

We may write

$$I - k(n)u^H(n) = I - \Phi^{-1}(n)u(n)u^H(n) \tag{4.34}$$

Next, we have

$$\Phi(n) = \lambda\Phi(n-1) + u(n)u^H(n) \tag{4.35}$$

Multiplying both sides of Equation (4.35) by the inverse matrix $\Phi^{-1}(n)$ and rearranging terms, we get

$$I - \Phi^{-1}(n)u(n)u^H(n) = \lambda\Phi^{-1}(n)\Phi(n-1) \tag{4.36}$$

Comparing Equations (4.34) and (4.36), we readily deduce that

$$I - k(n)u^H(n) = \lambda\Phi^{-1}(n)\Phi(n-1) \tag{4.37}$$

Suppose now we consider the effect of the quantization error $\eta_p(n_0)$ induced at time $n_0 \leq n$. When the RLS algorithm of Table 4.1 is used and the matrix P(*n*) remains Hermitian, then according to the error propagation model of Equation (4.32). The quantization error $\eta_p(n_0)$becomes modified at time *n* as follows:

$$\eta_p(n) = \lambda^{-(n-n_0)}\varphi(n,n_0)\eta_p(n_0)\varphi^H(n,n_0), \quad n \geq n_0 \tag{4.38}$$

Where $\varphi(n, no)$ is a transition matrix defined by

$$\varphi(n,n_0) = (I - k(n)u^H(n))\cdots\left(I - k(n_0+1)u^H(n_0+1)\right) \tag{4.39}$$

The repeated use of Equation (4.37) in (4.39) leads us to express the transition matrix in the equivalent form

$$\varphi(n,n_0) = \lambda^{n-n_0}\Phi^{-1}(n)\Phi(n_0) \tag{4.40}$$

The correlation matrix $\Phi(n)$ is defined by

$$\Phi(n) = \sum_{i=1}^{n}\lambda^{n-i}u(i)u^H(i) \tag{4.41}$$

On the basis of this definition, the tap-input vector u(n) is said to be uniformly persistently exciting for sufficiently large n if there exist some a> 0 and it $n > 0$ such that the following condition is satisfied (Ljung and Ljung. 1985):

$$\Phi(n) \geq aI \qquad \text{for } n \geq N \tag{4.42}$$

The notation used in Equation (4.42) is shorthand for saying that the matrix $\Phi(n)$ is positive definite. The condition for persistent excitation not only guarantees the positive definiteness of $\Phi(n)$, but also guarantees its matrix norm to be uniformly bounded for $n \geq N$, as shown by

$$\left\|\Phi^{-1}(n)\right\| \leq \frac{1}{a} \qquad \text{for } n \geq N \tag{4.43}$$

Returning to the transition matrix $\varphi(n,no)$ of Equation (4.40) and invoking the mutual consistency property of a matrix norm. We may write

$$\left\|\varphi(n,n_0)\right\| \leq \lambda^{n-n_0}\left\|\Phi^{-1}(n)\right\|\left\|\Phi(n_0)\right\| \tag{4.44}$$

Next. Invoking the inequality of (4.43). We may rewrite that of Equation (4.44) as

$$\left\|\varphi(n,n_0)\right\| \leq \frac{\lambda^{n-n_0}}{a}\left\|\Phi(n_0)\right\| \tag{4.45}$$

finally. We may use the error propagation equation (4.38) to express the vector norm of $\eta_p(n)$ as

$$\left\|\eta_p(n)\right\| \leq \lambda^{-(n-n_0)}\left\|\varphi(n,n_0)\right\|\left\|\eta_p(n-1)\right\|\left\|\varphi^H(n,n_0)\right\|$$

Which, in light of (4.48), may be rewritten as

$$\left\|\eta_p(n)\right\| \le \lambda^{n-n_0} M \qquad n \ge n_0 \tag{4.46}$$

Where M is a positive number defined by

$$M = \frac{1}{a^2}\left\|\Phi(n_0)\right\|^2 \left\|\eta_p(n-1)\right\| \tag{4.47}$$

Equation (4.47) states that the RLS algorithm of Table 4.1 is exponentially *stable* in the sense that a single quantization error $\eta_p(n_0)$ occurring in the inverse correlation matrix $P(n_0)$ at time n_0 decays exponentially provided that $\lambda<1$ (i.e., the algorithm has finite memory). In other words, the propagation of a single error through this formulation of the standard RLS algorithm with finite memory is contractive. Computer simulations validating this result are presented in Verhaegen (1989).

However, the single-error propagation for the case of growing memory,(i.e..$\lambda=1$) is *not* contractive. The reason for saying so is that when $\lambda = 1$, neither $\varphi(n, n_0) \le I$ nor $\left\|\varphi(n, n_0)\right\| \le 1$ holds, even if the input vector u(n) is persistently exciting. Consequently, the accumulation of numerical errors may cause the algorithm to be divergent (Yang 1994). In an independent study, Slock and Kai lath (1991) also point out that the error propagation mechanism in the RLS algorithm with $\lambda = 1$ is unstable and of a random walk type. Moreover, there is experimental evidence for this numerical divergence. Which reported in (Ardalan and Alexander, 1987).

4.8 Stalling Phenomenon

As with the LMS algorithm, a second form of divergence. Referred to as the stalling phenomenon, occurs when the tap weights in the RLS algorithm stop adapting. In particular this phenomenon occurs when the quantized elements of the matrix P(n) become very small, such that multiplication by P(n) is equivalent to multiplication by a zero matrix (Bottomley and Alexander. 1989). Clearly. The stalling phenomenon may arise no matte how the RLS algorithm is implemented.

The stalling phenomenon is directly linked to the exponential weighting factor λ and the variance σ_u^2 of the input data $u(n)$. Assuming that is λ close to unity, we find from the

definition of the correlation matrix $\Phi(n)$ that the expectation of $\Phi(n)$ is given by .

$$E[\Phi(n)] \approx \frac{R}{1-\lambda} \qquad \text{Large n} \tag{4.48}$$

For λ close to unity, we have

$$E[P(n)] = E[\Phi^{-1}(n)] \approx (E[\Phi(n)])^{-1} \tag{4.49}$$

Hence, using Equation (4.48) in Equation (4.49), we get

$$E[P(n)] \approx (1-\lambda)R^{-1} \qquad \text{Large n} \tag{4.50}$$

Where R^{-1} is the inverse of matrix R. Assuming that the tap-input vector u(n) is drawn from a wide-sense stationary process with zero mean, we may write

$$\Re = \frac{1}{\sigma_u^2} R \tag{4.51}$$

Where $\Re$ is a *normalized correlation matrix* with diagonal elements equal to 1 and off-diagonal elements less than or equal to 1 in magnitude, and σ_u^2 is the variance of an input data sample u(n). We may therefore rewrite Equation (4.50) as

$$E[P(n)] \approx \left(\frac{1-\lambda}{\sigma_u^2}\right)\Re^{-1} \qquad \text{For large n} \tag{4.52}$$

Equation (4.52) reveals that the RLS algorithm may stall if the exponential weighting factor λ is close to 1 and/or the input data variance σ_u^2 is large. Accordingly, we may prevent stalling of the standard RLS algorithm by using a sufficiently large number of accumulator bits in the computation of the inverse correlation matrix P(n).

CONCLUSION

The digital filter is a digital system that can be used to filter discrete-time signals. It can Be implemented by mean of software (computer programs) or by means of dedicated Hardware, and in either case it can be used to filter real-time signals or non-real-time (Recorded) signals.

By adaptive filter we mean one that is *self-designing* in that the adaptive filter relies for its operation on a *recursive algorithm,* which makes it possible for the filter to perform satisfactorily in an environment where complete knowledge of the relevant signal characteristics is not available.

The LMS algorithm is important because of its simplicity and ease of computation, and because it does not require off-line gradient estimations or repetitions of data. If the adaptive system is an adaptive linear combiner, and if the input vextor X_k and the desired response d_k are available at each iteration, the LMS algorithm is generally the best choice for many different applications of adaptive signal procesing.

References

[1] Cioffi and J. J. M. A. C. Bingham, A datadriven multitone echo canceller, *IEEE Trans. on Signal Processing*, 42(10):2853–2869, Oct. 1994.

[2] Chao, J. S. Kawabe, and S. Tsujii, A new IIR adaptive echo canceler: GIVE, *IEEE Trans. on Comm.*, 42(12):3090–3094, Dec. 1994.

[3] Eriksson, A. G. Eriksson, J. Karlsen, A. Roxstrom, and T. V. Hulth, Ericsson echo cancellers, *Ericsson Review*, (1):25–33, 1996.

[4] Haykin, S. *Adaptive Filter Theory*. Prentice-Hall, Upper Saddle River, NJ, 1996.

[5] Liu, Z. *Qr methods of on complexity in adaptive parameter estimation*, *IEEE Trans. on Signal Processing*, 43(3):720–729, March 1995.

[6] Mayyas and T. K. Aboulnasr, Leaky LMS algorithm: Mse analysis for gaussian data, *IEEE Trans. on Signal Processing*, 45(4):927–934, April 1997.

[7] Murano, K. S. Unagami, and F. Amano, Echo cancellation and applications, *IEEE Comm. Magazine*, 28(1):49–55, Jan. 1990.

[8] Petillon, T. A. Gilloire, and S. Theodoridis, The fast newton transversal filter: An efficient scheme for acoustic echo cancellation in mobile radio, *IEEE Trans. on Signal Processing*, 42(3):509– 518, March 1994.

[9] Sayed and T. A. H. Kailath, A state-space approach to adaptive RLS filtering, *IEEE Signal Processing Magazine*, 11(3):18–60, July 1994.

[10] Slock and T D. T. M.. Kailath, Numerically stable fast transversal filters for recursive least squares adaptive filtering, *IEEE Trans. on Signal Processing*, 39(1):92–114, Jan. 1991.

[11] Sondhi and W. M. M. Kellermann. *Advances in Speech Signal Processing*, chapter Adaptive echoes cancellation for speech signals, pp. 327–356. Marcel-Decker, 1991.

[12] Vaseghi, S. *Advanced Signal Processing and Digital Noise Reduction*. John Wiley & Sons, Chich-ester, NY, 1996.

[13] Vinay K. Ingle, *Digital Signal Processing*. The PWS BookWare Companion Series, International Thomson Publishing, ITP, 1997.

[14] Yasukawa, H. I. Furukawa, and Y. Ishiyama, Acoustic echo control for high quality audio teleconferencing, In *IEEE Int. Conf. on Acoustics, Speech and Signal Processing*, pp. 2041–2044, 1989.

[15] S.M. Kuo and D.M. Morgan, *Active Noise Control Systems: Algorithms and DSP Implementations* (New York: Wiley-Interscience, 1996).

[16] E.A. Lee ``Programmable DSP architectures: Part I," *IEEE Signal Processing Magazine*, vol. 5, no. 4, pp. 4-19, October 1988.

[17] E.A. Lee ``Programmable DSP architectures: Part II," *IEEE Signal Processing Magazine*, vol. 6, no. 1, pp. 4-14, January 1989.

[18] M.R. Smith, ``How RISCy is DSP?" *IEEE Micro*, vol. 12, no. 6, pp. 10-23, December 1992.

[19] P. Lapsley and G. Blalock, ``How to estimate DSP processor performance," *IEEE Spectrum*, vol. 33, no. 7, pp. 74-78, July 1996.

[20] *TMS320C4X User's Guide* (Dallas, TX: Texas Instruments, Inc., 1991).

[21] *PowerPC 604 RISC Microprocessor User's Manual* (Phoenix, AZ: Motorola, Inc., 1994).

[22] *Pentium Processor Family Developer's Manual,* vols. 1 and 2, (Mt. Prospect, IL: Intel Corporation, 1995).