UNIVERSITY

NEAR EAST UNIVERSITY

Faculty of Engineering

Department Of Electrical And Electronic Engineering

ADAPTIVE FILTER
(ECHO SUPPRESSOR AND ECHO CANCELLERS)

Graduation Project
EE- 400

Student: Karam Abdalbari (970968)

Supervisor: Mr. Jamal Fathi

Nicosia - 2001

48 13 67 09 74
NEU

NEAR EAST UNIVERSITY
1988 - LEFKOSA

ACKNOWLEDGEMENTS

I would like to thank my supervisor Mr.Jamal FATHI for his valuable advice and help to achieve my graduation project.

I would like to express my faithful thanks to my university with its entire educational staff.

I thank my family for their constant encouragement and support during the preparation of this project.

I would like also to thank all my friends for their advice and support.

ABSTRACT

An adaptive filter having self organizing structure based on recursive algorithm make it possible to perform satisfactory filtering in an environment where complete knowlege of the relevant signal characteristics are not available.

In block processing (or block implementation), a block of samples of the filter input and desired output are collected and then processed together to obtain a block of output samples.

There are many other applications in which adaptive filters may be usefully employed. At this time it can be confidently stated that these filters have matured to the point where their signal processing capabilities are well understood and documented in the technical literature.

The transversal adaptive filters form a large, diverse, and versatile family, which can satisfy the requirements of applications in many technical fields. Their complexity can be tailored to the resources of the users, and their performance assessed accordingly.

CONTENTS

INTRODUCUTION

The subject of adaptive filters has matured to the point where it new constitutes an important part of statistical signal processing. Whenever there is a requiretnent to process signals that result from operation in an environment of unknown statistics, the use of an adaptive filter offers an attractive solution to the problem as it usually provides a significant improvement in performance over the use of a fixed filter designed by cdnventional methods. Furthermore, the use of adaptive filters provides new signal processiqg capabilities that would not be possible otherwise. We thus find that adaptive filters are successfully applied in such diverse fields as communications, control, radar, sonar, seismology, and biomedical engineering.

In chapter one we discuss in general terms linear filter, adaptive filter and structure of adaptive filter. And we represent also adaptation approaches and real and complex form of adaptive filter. We end this chapter with some applications of adaptive filter.

Chapter two is representing the block LMS algorithm and the mathematic background of block implementation of adaptive filter. And we end this chapter with the FBLMS algorithm.

Chapter three is about the other applications of adaptive filter like modeling, spectral estimation, and adaptive array processing.

Chapter four represents the covariance algorithm, sliding window algorithm, the case of complex signals, multidimensional input signal, M-D algorithm based on all prediction errors and filters of nonuniform length.

CHAPTER 1
OVERVIEW OF ADAPTIVE FILTER

1.1 Linear Filters

The term 'filter' is commonly used to refer to any device or system that takes a mixture of particles/elements from its input and process them according to some specific rules to generate a corresponding set of particles/elements at its output. In the context of signals and systems, particles/elements are the frequency components of the underlying signals and, traditionally, filters are used to retain all the frequency components that belong to a particular band of frequencies, while rejecting the rest of them, as much as possible. In a more general sense, the term filter may be used to refer to a system that reshapes the frequency components of the input to generate an output signal with some desirable features, and this is how we view the concept of filtering throughout the chapters, which follow.

Filters (or systems, in general) may be either linear or non-linear. In this book, we consider only linear filters and our emphasis will also be on discrete-time signals and systems. Thus, all the signals will be represented by sequences, such as x (n). The most Basic feature of linear systems is that their behaviour is governed by the principle of superposition. This means that if the responses of a linear discrete-time system to input sequences x_1 (n) and x_2 (n) are Yi (n) andy_2 (n), respectively, then the response of the same system to the input sequence x (n) = ax_1 (n) + bx_2 (n), where a and b are arbitrary constants, will be y (n) = ay_1 (n) ± by_2 (n). This property leads to many interesting results in 'linear system theory'. In particular, a linear system is completely characterized by its impulse response or the Fourier transform of its impulse response, known as the transfer function. The transfer function of a system at any frequency is equal to its gain at that frequency. In other words, in the context of our discussion above, we may say that the transfer function of a system determines how the various frequency components of its input are reshaped by the system. In particular, the filter is used to reshape certain input signals in such a way that its output is a good estimate of

the given desired signal. The process of selecting the filter parameters (coefficients) so as to achieve the best match between the desired signal and the filter output is often done by optimizing an appropriately defined performance function. The performance function can be defined in a statistical or deterministic framework. In the statistical approach, the most commonly used performance function is the mean-square value of the error signal, i.e. the difference between the desired signal and the filter output. For stationary input and desired signals, minimizing the mean-square error results in the well-known Wiener filter, which is said to be optimum in the mean-square sense. In the deterministic approach, the usual choice of performance function is a weighted sum of the squared error signal. Minimizing this function results in a filter, which is optimum for the given set of data. However, under some assumptions on certain statistical properties of the data, the deterministic solution will approach the statistical solution, i.e. the Wiener filter, for large data lengths.

1.2 Adaptive Filters

As mentioned in the previous section, the filter required for estimating the given desired signal can be designed using either the stochastic or deterministic formulations. In the deterministic formulation, the filter design requires the computation of certain average quantities using the given set of data that the filter should process. On the other hand, the design of Wiener filter (i.e. in the stochastic approach) requires a priori knowledge of the statistics of the underlying signals. Strictly speaking, a large number of realizations of the underlying signal sequences are required for reliably estimating these statistics. This procedure is not feasible in practice since we usually have only one realization for each of the signal sequences. To resolve this problem, it is assumed that the underlying signal sequences are ergodic, which means that they are stationary and their statistical and time averages are identical. Thus, by using time averages, Wiener filters can be designed, even though there is only one realization for each of the signal sequences.

Although direct measurement of the signal averages to obtain the necessary information for the design of Wiener or other optimum filters is possible, in most of the applications the signal averages (statistics) are used in an indirect manner. The reasons for solving the problem of adaptive filtering in an iterative manner are:

1. Direct computation of the necessary averages and their application for computing the filter coefficients requires the accumulation of a large amount of signal\
samples. Iterative solutions, on the other hand, do not require accumulation of signal samples, thereby resulting in a significant amount of saving in memory.
2. The accumulation of signal samples and their post processing to generate the filter output, as required in non-iterative solutions, introduces a large delay in the filter output. This is unacceptable in many applications. Iterative solutions on the contrary, do not introduce any significant delay in the filter output.
3. The use of iterations results in adaptive Solutions with some tracking capability. That is, if the signal statistics are changing with time, then the solution provided by an iterative adjustment of the filter coefficients will be able to adapt to the new statistics.
4. Iterative Solutions, in general, are much simpler to code in software or to implement in hardware than their non-iterative counterparts.

1.3 Adaptive Filter Structures

The most commonly used structure in the implementation of adaptive filters is the transversal structure, Here, the adaptive filter has a single input, x (n), and an output, y (n). The sequence d (n) is the desired signal. The output, y (n), is, generated as a linear combination of the delayed samples of the input sequence, x (n), according to the equation

$$y(n) = \sum_{i=0}^{N-1} w_i(n)\, x(n-1), \qquad (1.1)$$

Where the w_1 (n) s is the filter tap weights (coefficients) and N is the filter length. We refer to the input samples, x (n - i), for i = 0,1... N- l, as the filter tap inputs. The tap weights, the wi (n) s, which may vary in time, are controlled by the adaptation algorithm.
Through various adaptive algorithms. Because of these points, the non-recursive filters are the sole candidates in most of the applications of adaptive filters.
The FIR and IIR structures shown in Figures 1.2 and 1.4 are obtained by direct realization of the respective difference equations (1.1) and (1.3). These filters may

alternatively be implemented using the lattice structures. The lattice structures, in general, are more complicated than the direct implementations. However, in certain applications they have some advantages, which make them better candidates than the direct forms. For instance, in the application of linear prediction for speech processing where we need to realize all pole (IIR) filters, the lattice structure can be more easily controlled to prevent possible instability of the filter.

The FIR and IIR filters that were discussed above are classified as linear filters since their outputs are obtained as linear combinations of the present and past samples of input and, in the case of the hR filter, the past samples of the output also. Although most applications are restricted to the use of linear filters, non-linear adaptive filters become necessary in some applications where the underlying physical phenomena to be modelled are far from being linear. A typical example is magnetic recording where the recording channel becomes non-linear at high densities as a result of the interaction between the magnetization transitions written on the medium. The Volterra series representation of systems is usually used in such applications. The output, y(n), of a Volterra system is related to its input, x(n). According to the equation

$$\begin{aligned} y(n) &= w_{0,0}(n) + \sum_{i} w_{1,i}(n)x(n-1) + \\ &\sum_{i,j} w_{2,i,j}(n)x(n-i)x(n-j) + \\ &\sum_{i,j,k} w_{2,i,j,k}(n)x(n-i)x(n-j)x(n-k) + \ldots\ldots, \end{aligned} \tag{1.2}$$

Where $w_{0,0}$ (n), the $w_{1,I}$(n)s, the $w_{2,1J}$ (n)s, the $w_{11,J,K}$ (n)s, ... are filter coefficients. . However, we note that all the summations in (1.2) may be put together and the Volterra filter may be thought of as a linear combiner whose inputs are determined by the delayed samples of x (n) and their cross-multiplications. Noting this, we find that the extension of most of the adaptive filtering algorithms to the Volterra filters is straightforward.

1.4 Adaptation Approaches

As introduced in Sections 1.1 and 1.2, there are two distinct approaches that have been widely used in the development of various adaptive algorithms; namely, stochastic and deterministic. Both approaches have many variations in their implementations leading to a rich variety of algorithms, each of which offers desirable features of its own. In this section we present a review of these two approaches and highlight the main features of the related algorithms.

1.4.1 Approach Based On Wiener Filter Theory

According to the Wiener filter theory, which comes from the stochastic framework, the optimum coefficients of a linear filter are obtained by minimization of its mean-square error (MSE). As already noted, strictly speaking, the minimization of MSE requires certain statistics obtained through ensemble averaging, which may not be possible in practical applications. The problem is resolved using ergodicity so as to use time averages instead of ensemble averages. Furthermore, to come up with simple recursive algorithms, very rough estimates of the required statistics are used. In fact, the celebrated least-mean-square (LMS) algorithm, which is the most basic and widely used algorithm in various adaptive filtering applications, uses the instantaneous value of the square of the error signal as an estimate of the MSE. It turns out that this very rough estimate of the MSE, when used with a small step-size parameter in searching for the optimum coefficients of the Wiener filter, leads to a very simple and yet reliable adaptive algorithm.

The main disadvantage of the LMS algorithm is that its convergence behaviour is highly dependent on the power spectral density of the filter input. When the filter input is white, i.e. its power spectrum is fiat across the whole range of frequencies, the LMS algorithm converges very fast. However, when certain frequency bands are not well excited (i.e. the signal energy in those bands is relatively low), some slow modes of convergence appear, resulting in very slow convergence compared with the case of white input. In other words, to converge fast, the LMS algorithm requires equal excitation over the whole range of frequencies. Noting this, over the years researchers

have developed many algorithms, which effectively divide the frequency band of the input signal into a number of subbands and achieve some degree of signal whitening by using some power normalization mechanism, prior to applying the adaptive algorithm.

In some applications, we need to use adaptive filters whose length exceeds a few hundreds or even a few thousands of taps. Clearly, such filters are computationally expensive to implement. An effective way of implementing such filters at a much lower computational complexity is to use the fast Fourier transform (FFT) algorithm to implement time domain convolutions in the frequency domain, as is commonly done in the implementation of long digital filters.

1.4.2 Method Of Least Squares

The adaptive filtering algorithms whose derivations are based on the Wiener filter theory have their origin in a statistical formulation of the problem. In contrast to this, the method of least squares approaches the problem of filter optimization from a deterministic point of view. As already mentioned, in the Wiener filter theory the desired filter is obtained by minimizing the mean-square error (MSE), i.e. a statistical quantity. In the method of least

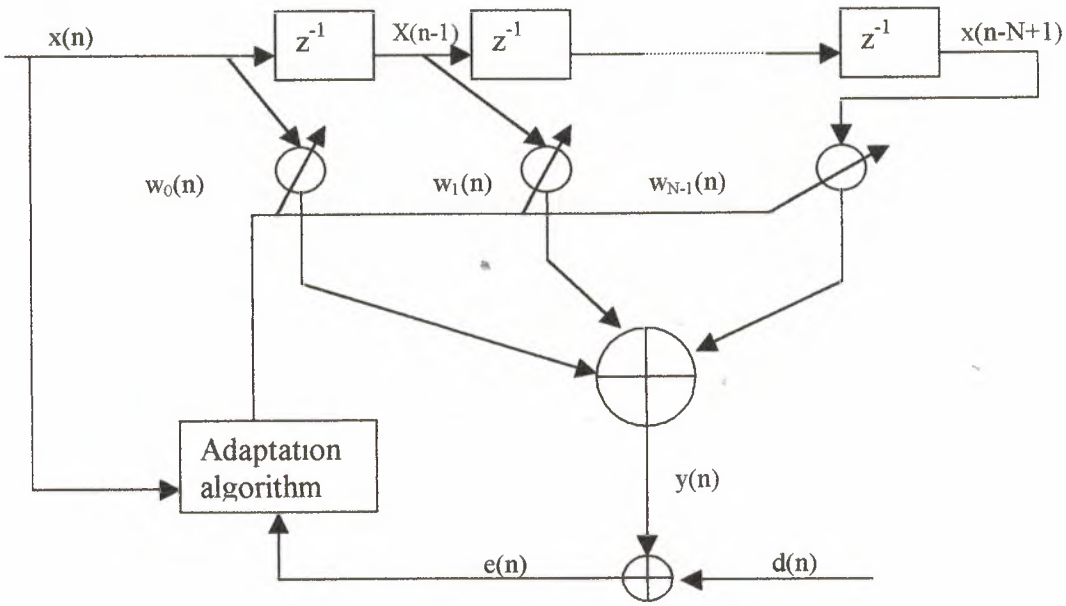


Figure 1.2 Adaptive transversal filter

In some applications, such as beam forming, the filter tap inputs are not the delayed samples of a single input. In such cases the structure of the adaptive filter assumes the form shown in Figure 1 .3. This is called a linear combiner, since its output is a linear combination of the different signals received at its tap inputs:

$$y(n) = \sum_{i=0}^{N-1} w_i(n) x_i(n). \tag{1.3}$$

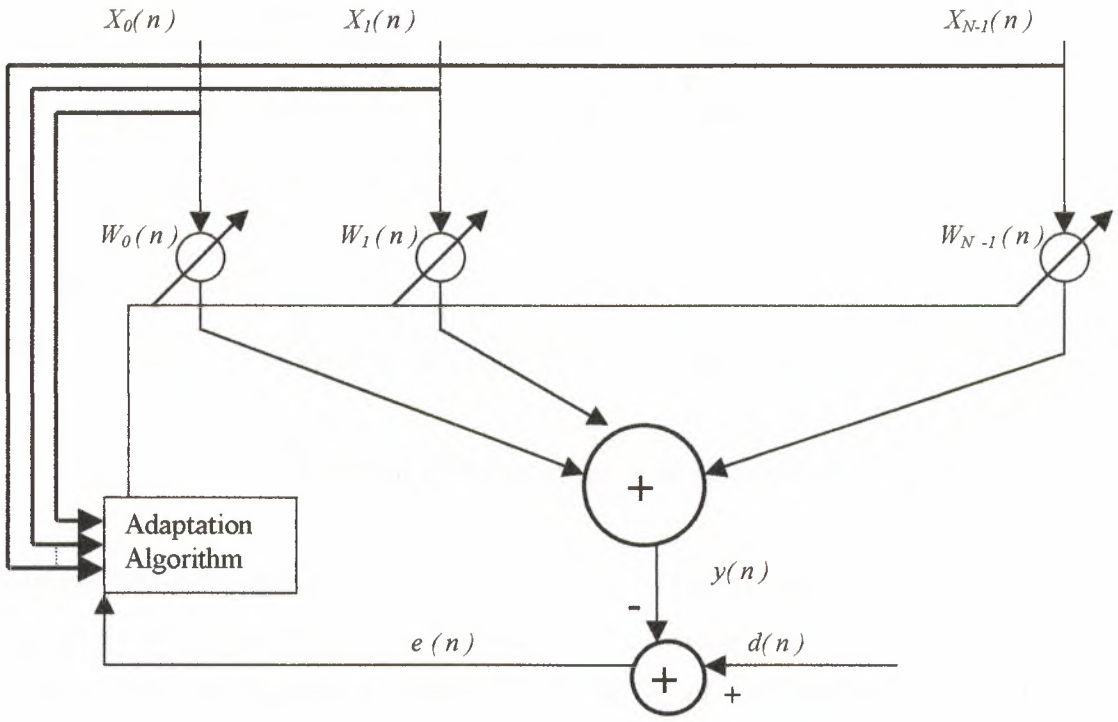


Figure 1.3 Adaptive linear combiner

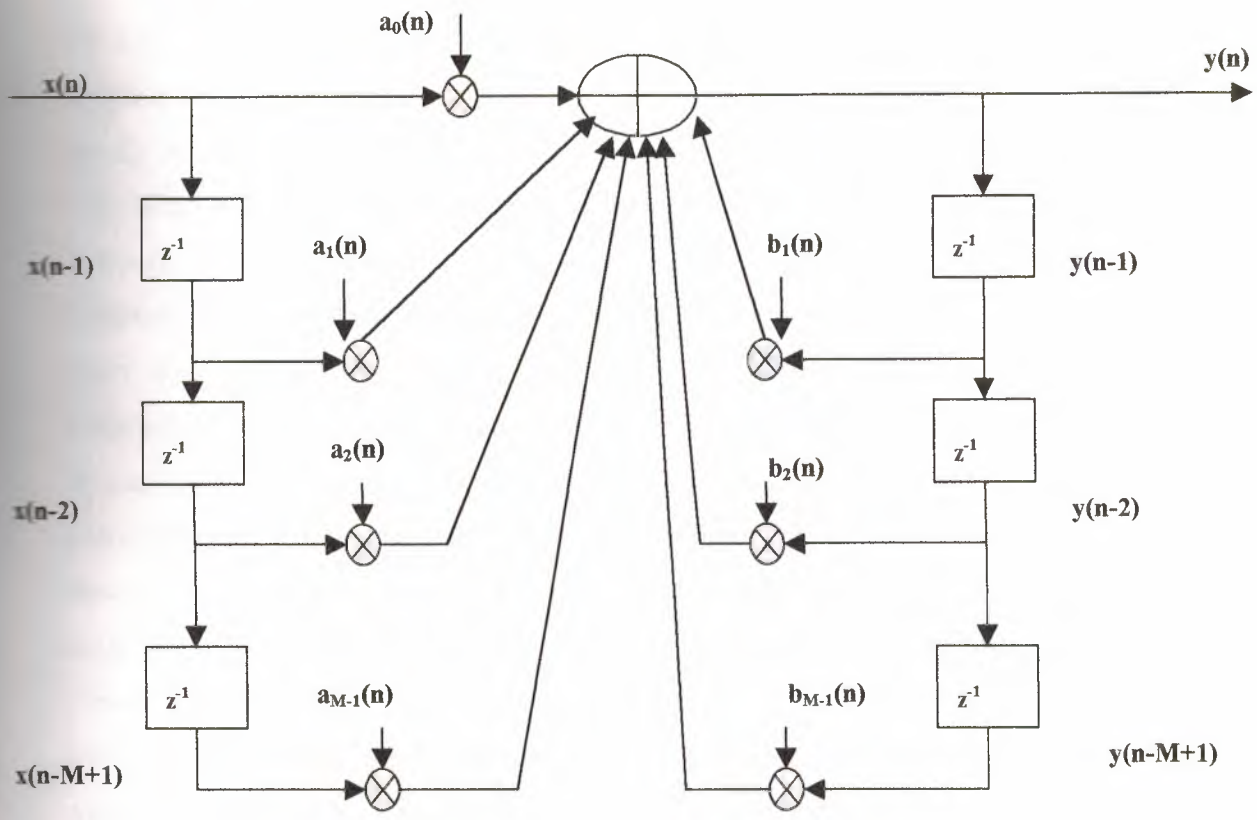


Figure 1.4 the structure of an IIR filter

Note that the linear combiner structure is more general than the transversal. The latter, as a special case of the former, can be obtained by choosing $x_1(n) = x(n - i)$.

The structures of Figures 1.2 and 1.3 are those of the non-recursive filters, i.e. computation of filter output does not involve any feedback mechanism. We also refer to Figure 1.2 as a finite-impulse response (FIR) filter, since its impulse response is of finite duration in time. An infinite-impulse response (IIR) filter is governed by recursive equations such as (see Figure 1.4)

$$y(n) = \sum_{i=0}^{N-1} a_i(n)x(n-i) + \sum_{i-1}^{M-1} b_i(n)y(n-i), \qquad (1.4)$$

Where $a_i(n)$ and $b_i(n)$ are the forward and feedback tap weights, respectively. IIR filters have been used in many applications. However, as we shall see in the later chapters, because of the many difficulties involved in the adaptation of hR filters, their application in the area of adaptive filters is rather limited. In particular, they can easily

become unstable since their poles may get shifted out of the unit circle (i.e. | z | = 1, in the z-plane (see next chapter)) by the adaptation process. Moreover, the performance function (e.g. mean-square error as a function of filter coefficients) of an hR filter usually has many local minima points. This may result in convergence of the filter to one of the local minima and not to the desired global minimum point of the performance function. On the contrary, the mean-square error functions of the FIR filter and linear combiner are well-behaved quadratic functions with a single minimum point which can easily be found squares, on the other hand, the performance index is the sum of weighted error squares for the given data, i.e. a deterministic quantity. A consequence of this deterministic approach is that the least-squares-based algorithms, in general, converge much faster than the LMS-based algorithms. They are also insensitive to the power spectral density of the input signal. The price that is paid for achieving this improved convergence performance is higher computational complexity and poorer numerical stability.

Direct formulation of the least-squares problem results in a matrix formulation of its solution, which can be applied on a block-by-block basis to the incoming signals. This, which is referred to as the block estimation of the least-squares method, has some useful applications in areas such as linear predictive coding of speech signals. However, in the context of adaptive filters, recursive formulations of the least-squares method that update the filter coefficients after the arrival of every sample of input are preferred, for reasons that were given in Section 1.2. There are three major classes of recursive least-squares (RLS) adaptive filtering algorithms:

- The standard RLS algorithm
- The QR-decomposition-based RLS (QRD-RLS) algorithm
- Fast RLS algorithms

1.4.3 The Standard RLS Algorithm

The derivation of this algorithm involves the use of a well-known result from linear algebra known as the matrix inversion lemma. Consequently, the implementation of the standard RLS algorithm involves matrix manipulations that result in a computational complexity proportional to the square of the filter length.

1.4.4 The QR-Decomposition-Based RLS (QRD-RLS) Algorithm

This formulation of RLS algorithm also involves matrix manipulations, which lead to a computational complexity that grows with the square of the filter length. However, the operations involved here are such that they can be put into some regular structures known as systolic arrays. Another important feature of the QRD-RLS algorithm is its robustness to numerical errors as compared with other types of RLS algorithms.

1.4.5 Fast RLS Algorithms

In the case of transversal filters, the tap inputs are successive samples of the input signal, x(n) (see Figure 1.2). The fast RLS algorithms use this property of the filter input and solve the problem of least squares with a computational complexity which is proportional to the length of the filter, thus the name fast RLS. Two types of fast RLS algorithms may be recognized:

1. RLS lattice algorithms. These lattice algorithms involve the use of order-update as well as the time-update equations. A consequence of this feature is that it results in modular structures which are suitable for hardware implementations using the Pipelining technique. Another desirable feature of these algorithms is that certain variants of them are very robust against numerical errors arising from the use of finite word lengths in computations.
2. Fast transversal RLS algorithm: In terms of number of operations per iteration, the fast transversal RLS algorithm is less complex than the lattice RLS algorithms. However, it suffers from numerical instability problems, which require careful attention to prevent undesirable behaviour in practice.

1.5 Real And Complex Forms Of Adaptive Filters

There are some practical applications in which the filter input and its desired signal are complex-valued. A good example of this situation appears in digital data transmission, where the most widely used signalling techniques are phase shift keying (P5K) and quadrature amplitude modulation (QAM). In this application, the baseband signal consists of two separate components, which are the real and imaginary parts of a complex-valued signal.

1.6 Applications

Adaptive filters, by their very nature, are self-designing systems, which can adjust themselves to different environments. As a result, adaptive filters find applications in such diverse fields as control, communications, radar and sonar signal processing, interference cancellation, active noise control, biomedical engineering, etc. The common feature of these applications that brings them under the same basic formulation of adaptive filtering is that they all involve a process of filtering some input signal to match a desired response. The filter parameters are updated by making a set of measurements of the underlying signals and applying that set to the adaptive filtering algorithm such that the difference between the filter output and the desired response is minimized in either a statistical or a deterministic sense. In this context, four basic classes of adaptive filtering applications are recognized. Namely, modelling, inverse modelling, linear prediction, and interference cancellation. In the rest of this chapter, we present an overview of these applications.

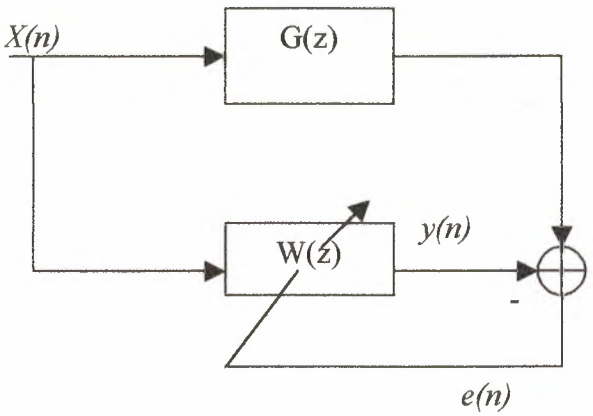


Figure 1.5 Adaptive system modeling

1.6.1 Modeling

Figure 1.5 depicts the problem of modelling in the context of adaptive filters. The aim is to estimate the parameters of the model. W (z), of a plant, G (z). On the basis of some a priori knowledge of the plant, G (z), a transfer function, W (z), with certain number of adjustable parameters is selected first. The parameters of W (z) are then chosen through an adaptive filtering algorithm such that the difference between the plant output, d(n), and the adaptive filter output, v(n), is minimized.

An application of modelling. Which may be readily thought of, is system identification. In most modern control systems the plant under control is identified on-line and the result is used in a self-tuning regulator (STR) loop, as depicted in Figure 1.6 .

Another application of modelling is echo cancellation. In this application an adaptive filter is used to identify the impulse response of the path between the source from which the echo originates and the point where the echo appears. The output of the adaptive filter, which is an estimate of the echo signal, can then be used to cancel the undesirable echo. The subject of echo cancellation is discussed further below in Section 1.6.4.

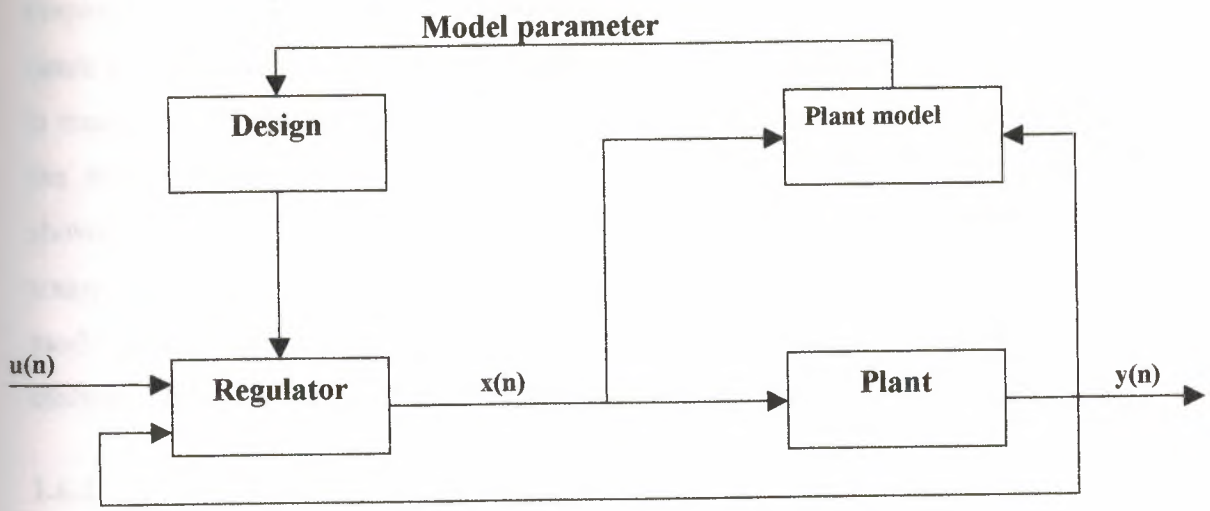


Figure 1.6 Block diagram of a self-tuning regulator

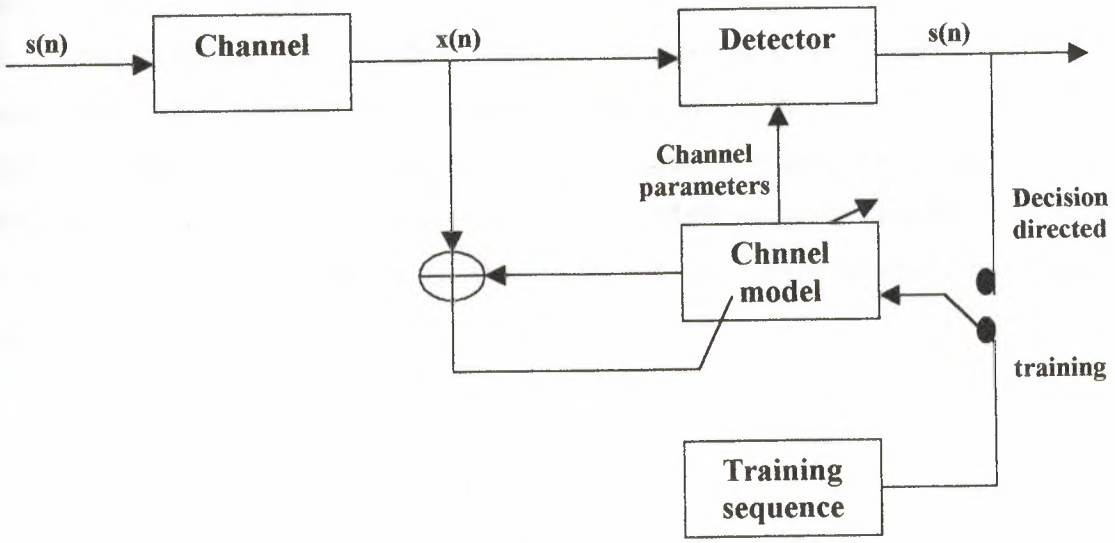


Figure 1.7 An adaptive data receiver using channel identification

Non-ideal characteristics of communication channels often result in some distortion in the received signals. To mitigate such distortion, channel equalizers are usually used. This technique, which is equivalent to implementing the inverse of the channel response, is discussed below in Section 1.6.2. Direct modelling of the channel, however, has also been found useful in some implementations of data receivers. For instance, data receivers equipped with maximum likelihood detectors require an estimate of the channel response. Furthermore, computation of equalizer coefficients from channel

response has been proposed by some researchers since this technique has been found to result in better tracking of time-varying channels. In such applications, a training pattern is transmitted in the beginning of every connection. The received signal, which acts as the desired signal to an adaptive filter, is used in a set-up to identify the channel, as shown in Figure 1.7. Once the channel is identified and the normal mode of transmission begins, the detected data symbols, (n), are used as input to the channel model and the adaptation process continues for tracking possible variations of the channel. This is known as the decision directed mode and is also shown in Figure 1.7.

1.6.2 Inverse Modeling

Inverse modelling, also known as disconsolation, is another application of adaptive filters, which has found extensive use in various engineering disciplines. The most widely used application of inverse modelling is in communications where an inverse model (also called an equalizer) is used to mitigate the channel distortion. The concept of inverse modelling has also been applied to adaptive control systems where a controller is to be designed and cascaded with a plant so that the overall response of this cascade matches a desired (target) response. The process of prediction, which will be explained later, may also be viewed as an inverse modelling scheme (see Section 1.6.3). In this section we concentrate on the application of inverse modeling in channel equalization.

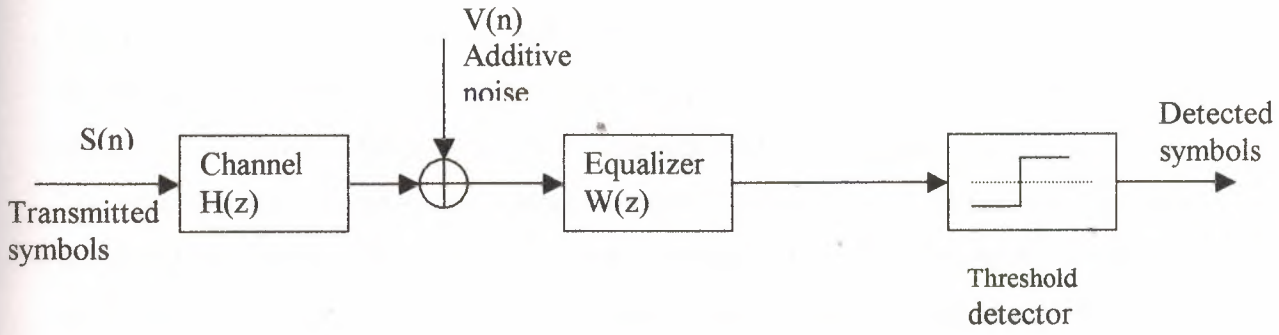


Figure 1.8 A baseband data transmission system channel equalizer

1.6.3 Channel Equalization

Depicts the block diagram of a base band transmission system equipped with a channel equalizer. Here, the channel represents the combined response of the transmitter filter, the actual channel, and the receiver front-end filter. The additive noise sequence, v(n), arises from thermal noise in the electronic circuits and possible cross-talks from neighboring channels. The transmitted data symbols, s (n), that appear in the form of amplitude/phase-modulated pulses, are distorted by the channel. The most significant among the different distortions is the pulse-spreading effect, which results because the channel impulse response is not equal to an ideal impulse function, but rather a response that is non-zero over many symbol periods. This distortion results in interference of the neighbouring data symbols with one another, thereby making the detection process through a simple threshold detector unreliable. The phenomenon of interference between neighbouring data symbols is known as intersymbol interference (ISI). The presence of the additive noise samples, v (n), further deteriorates the performance of data receivers. The role of the equalizer, as a filter, is to resolve the distortion introduced by the channel (i.e. rejection or minimization of ISI) while minimizing the effect of additive noise at the threshold detector input (equalizer output) as much as possible. If the additive noise could be ignored, then the task of equalizer would be rather straightforward. For a channel H (z), an equalizer with transfer function W (z) = l/H (z) could do the job perfectly, as this results in an overall channel-equalizer transfer function H (z) W (z) = 1, which implies that the transmitted data sequence, s (n), will appear at the detector input without any distortion. Unfortunately, this is an ideal situation, which cannot be used, in most of the practical applications.

We note that the inverse of the channel transfer function, i.e. 1 /H (z), may be non-causal if H (z) happens to have a zero outside the unit circle, thus making it unrealizable in practice. This problem is solved by selecting the equalizer so that H (z) W (z) $z\Delta$, where Δ is an appropriate integer delay. This is equivalent to saying that a delayed replica of the transmitted symbols appears at the equalizer output.

We also note that the choice of W (z) = 1/H (z) (or W (z) $z \approx -\Delta$ /H (z)) may lead to a significant enhancement of the additive noise, v (n), in those frequency bands where the magnitude of H (z) is small (i.e. 1 /H (z) is large). Hence, in choosing an equalizer,

We should keep a balance between residual hSh and noise enhancement at the equalizer output.

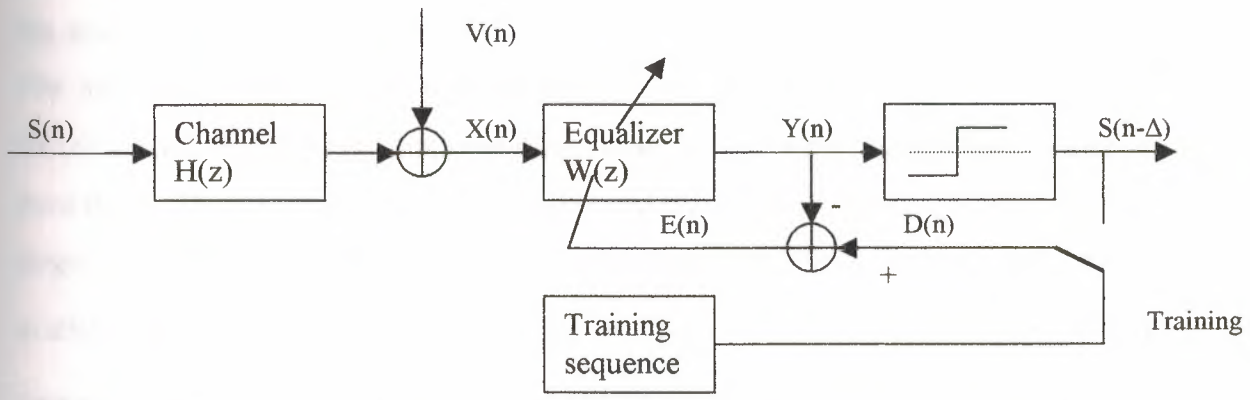


Figure 1.9 Details of baseband data transmission system equipped with an adaptive channel equalizer.

A Wiener filter is a solution with such a balance.

Figure 1.9 presents the details of a base band transmission system, equipped with an adaptive equalizer. The equalizer is usually implemented in the form of a transversal filter. Initial training of the equalizer requires knowledge of the transmitted data symbols (or, to be more accurate, a delayed replica of them) since they should be used as the desired signal samples for adaptation of the equalizer tap weights. This follows from the fact that the equalizer output should ideally be the same as the transmitted data symbols. We thus require an initialization period during which the transmitter sends a sequence of training symbols that are known to the receiver. This is called the training mode. Training symbols are usually specified as part of the standards and the manufacturers of data modems should comply with these so that the modems of different manufacturers can communicate with one another. (The term modem, which is an abbreviation for modulator and demodulator', is commonly used to refer to data transceivers (transmitter and receiver).)

At the end of the training mode the tap weights of the equalizer would have converged close to their optimal values. The detected symbols would then be similar to the transmitted symbols with probability close to one. Hence, from then onwards, the detected symbols can be treated as the desired signal for further adaptation of the equalizer so that possible variations in the channel can be tracked. This mode of operation of the equalizer is called the decision directed mode. The decision directed mode successfully

works as long as the channel variation is slow enough so that the adaptation algorithm is able to follow the channel variations satisfactorily. This is necessary for the purpose of ensuring low symbol error rates in detection so that these symbols can still be used as the desired signal.

The inverse modelling discussed above defines the equalizer as an approximation of z /H (z), i.e. the target/desired response of the cascade of channel and equalizer is z -Δ, a pure delay. This can be generalized by replacing the target response z -Δ by a general target response, say Γ(z). In fact, to achieve higher efficiency in the usage of the available bandwidth, some special choices of $\Gamma(z) \neq z^{-\Delta}$ are usually considered in communication systems. Systems that incorporate such non-trivial target responses are referred to as partial-response signalling systems. The detector in such systems is no more the simple threshold detector, but one which can exploit the information that the overall channel is

now Γ(z), instead of the trivial memoryless channel z^{Δ}. The Viterbi detector (Proakis, 1995) is an example of such a detector. The target response, Γ(z), is selected so that its magnitude response approximately matches the channel response, i.e. $\Gamma(e^{iw})$

over the range of frequencies of interest. The impact of this choice is that the equalizer, which is now $W(z) \approx \Gamma(z)/H(z)$, has a magnitude response that is approximately equal to one, thereby minimizing the noise enhancement. To clarify this further and also to mention another application of inverse modeling.

1.6.4 Magnetic Recording

The process of writing data bits on a magnetic medium (tape or disk) and reading them back later is similar to sending data bits over a communication channel from one end of a transmission line and receiving them at the other end of the line. The data bits, which are converted to signal pulses prior to recording, undergo some distortion due to the non-perfect behaviour of the head and medium, as happens in communication channels because of the non-ideal response of the channel. Additive thermal noise and interference from neighbouring recording tracks (just like neighbouring channels in communications) are also present in the magnetic recording channels.

Magnetic recording channels are usually characterized by their response to an isolated pulse of width one bit interval, T. This is known as the dibit response, and in the case of hard-disk channels it is usually modelled by the superposition of positive and negative Lorentzian pulses, separated by one bit interval, T. In other words, the Lorentzian pulse models the step response of the channel. The Lorentzian pulse is defined as

$$g_a(t) = \frac{1}{1 + \left(\frac{2t}{t_{50}}\right)^2} \tag{1.5}$$

Where t_{50} is the pulse width measured at 50% of its maximum amplitude. The subscript 'a' in ga (t) and other functions that appear in the rest of this subsection is to emphasize that they are analog (non-sampled) signals. The ratio D = t_{50}/T is known as the recording density. Typical values of D are in the range 1 to 3. A higher density means that more bits are contained in one t_{50} interval, i.e. more 151. We may also note that t_{50} is a temporal measure of the recording density. When measured spatially, we obtain another parameter, pit'$_{50}$ = tso/v, where v is the velocity of the medium with respect to the head. Accordingly, for a given speed, v, the value of D specifies the actual number of bits written on a length pw_{50} along the track on the magnetic medium.

Using (1.5), the dibit response of a hard-disk channel is obtained as the response of the channel to a sequence s (n) of data bits is then given by the convolution sum

$$H_a(t) = g_a(t) - g_a(t - T) \tag{1.6}$$

Thus, the dibit response, g_a (t), is nothing but the impulse response of the recording channel.

Figures 1.10(a) and (b) show the dibit (time domain) and magnitude (frequency domain) responses, respectively, of the magnetic channels (based on the Lorentzian model) for densities D = 1, 2 and 3. From Figure 1.10(b) we note that most of the energy in the read-back signals is concentrated in a midband range between zero and an upper-limit around 1 /2T. Clearly, the bandwidth increases with increase in density. In the light of our previous discussions, we may thus choose the target response, Γ(z), of the equalizer so that it resembles a bandpass filter whose bandwidth and magnitude response are close to that of the Lorentzian dibit responses.

In magnetic recording, the most commonly used partial responses (i.e. target responses) are given by the class IV response

$$\Gamma(z) = z^{-\Delta}(1+z^{-1})^{K}(1-z^{-1}), \qquad (1.7)$$

Where Δ, as before, is an integer delay and K is an integer greater than or equal to one. As the recording density increases, higher values of K will be required to match the channel characteristics. But, as K increases, the channel length also increases, implying higher complexity in the detector.

1.6.5 Linear Prediction

Prediction is a spectral estimation technique that is used for modelling correlated random processes for the purpose of finding a parametric representation of these processes. In general, different parametric representations could be used to model the processes. In the context of linear prediction, the model used is shown in Figure 1.11. Here, the random process, X(n), is assumed to be generated by exciting the filter G (z) with the input u (n). Since G (z) is an all-pole filter, this is known as autoregressive (AR) modelling. The choice/type of the excitation signal, u (n),is application dependent and may vary depending on the nature of the process being modelled. However, it is usually chosen to be a white process.

Other models used for parametric representation are moving average (MA) models, where G (z) is an all-zero (transversal) filter, and autoregressive moving average (ARMA) models, where G (z) has both poles and zeros. However, the use of AR model is more popular than the other two.

The rationale behind the use of AR modelling may be explained as follows. Since the samples of any given non-white random signal, x (n), are correlated with one another, these correlations could be used to make a prediction of the present sample of the process, x (n), in terms of its past samples, x (n — 1), x (n —2)..., x (n — N), as in Figure 1.12. Intuitively, such prediction improves as the predictor length increases.

However, the improvement obtained may become negligible once the predictor length, N. exceeds a certain value, which depends upon the extent of the correlation in the given process. The prediction error, e (n), will then be approximately white. We now note that the transfer function between the input process, x (n), and the prediction error, e (n), is

$$H\ (z)\ = 1 - \sum_{i=1}^{N} a_i z^{-i}\ , \tag{1.8}$$

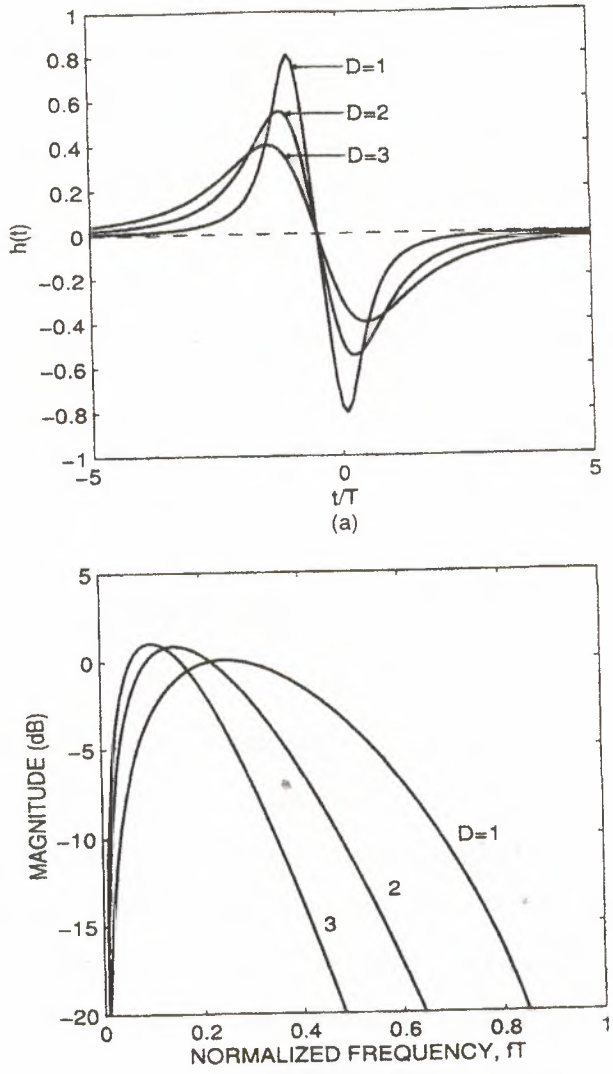


Figure 1.10 Time and frequency domain responses of magnetic recording channels for i =1 densities D = 1, 2 and 3, modeled using the Lorentzian pulse. (a) Dibit response. (b) Magnitude response of dibit response.

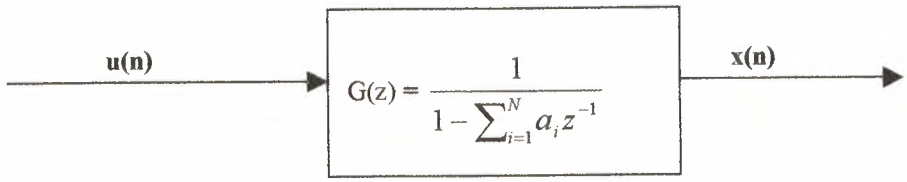


Figure 1.11 Autoregressive modeling of a random process

Where the a_1s are the predictor coefficients. Now, if a white process, u(n), with similar statistics as e(n) is passed through an all-pole filter with the transfer function,

$$G(Z)\frac{1}{1-\sum_{i=1}^{N} a_i z^{-1}}, \qquad (1.9)$$

As in Figure 1.11, then the generated output. x (n), will clearly be a process with the same statistics as x(n).

With the background developed above, we are now ready to discuss a few applications of adaptive prediction.

1.6.6 Autoregressive Spectral Analysis

In certain applications we need to estimate the power spectrum of a random process. A trivial way of obtaining such an estimate is to take the Fourier transform (discrete Fourier transform (DFT) in the case of discrete-time processes) and use some averaging (smoothing) technique to improve the estimate. This comes under the class of non-parametric spectral estimation techniques. When the number of samples of the input is limited, the estimates provided by non-parametric spectral estimation techniques will become unreliable. In such cases the parametric spectral estimation, as explained above, may give more reliable estimates.

As mentioned already, parametric spectral estimation could be done by using AR, MA or ARMA models. In the case of AR modelling we proceed as follows. We first choose

a proper order, N, for the model. The observed sequence, x (n), is then applied to a predictor structure similar to Figure 1.12 whose coefficients, the a_is, are optimized by minimizing the prediction error, e (n). Once the predictor coefficients have converged.

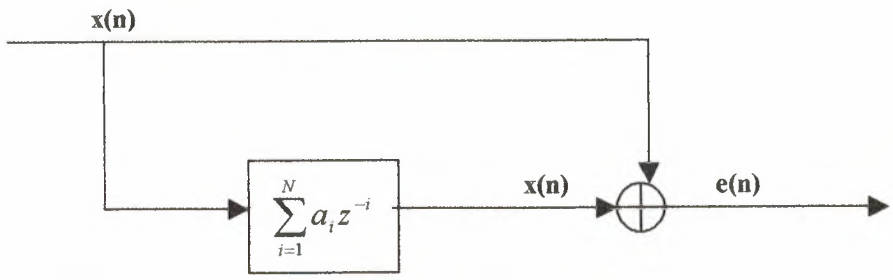


Figure 1.12 Linear predictor

Where N_0 is an estimate of the power of the prediction error, e (n). This follows from the model of Figure 1.11 and the fact that after convergence of the predictor, e (n) is approximately white.

1.6.7 Adaptive Line Enhancement

Adaptive line enhancement refers to the situation where a narrow-band signal embedded in a wide-band signal (usually white) needs to be extracted. Depending on the application, the extracted signal may be the signal of interest, or an unwanted interference that should be removed. Examples of the latter case are a spread spectrum signal that has been corrupted by a narrow-band signal and biomedical measurement signals that have been corrupted by the 50/60 Hz power-line interference.

The idea of using prediction to extract a narrow-band signal when mixed with a wide-band signal follows from the following fundamental result of signal analysis: successive samples of a narrow-band signal are highly correlated with one another, whereas there is almost no correlation between successive samples of a wide-band process. Because of this, if a process x (n) consisting of the sum of narrow-band and wide-band processes is applied to a predictor, then the predictor output, .x (n), will be a good estimate of the narrow-band portion of x (n). In other words, the predictor will act as a narrow-band filter which rejects most of the wide-band portion of x (n) and keeps (enhances) the narrow-band portion, thus the name line enhancer.

We also note that in the applications where the narrow-band portion of x (n) has to be rejected the difference between x (n) and 4n), i.e. the estimation error, e (n), is taken as the system output. In this case the transfer function between the input, x (n), and the output, e (n), will be that of a notch filter.

1.6.8 Speech Coding

Since the advent of digital signal processing, speech processing has always been one of the focused research areas. Among various processing techniques that have been applied to speech signals, linear prediction has been found to be the most promising technique, leading to many useful algorithms. In fact, most of the theory of prediction was developed in the context of speech processing.

There are two major speech-coding techniques that involve linear prediction. Both techniques aim at reducing the number of bits used for every second of speech to achieve saving in storage and/or transmission bandwidth. The first technique, which is categorized under the class of source coders, strives to produce digitized voice data at low bit rates in the range 2-10 kb/s. The synthesized speech, however, is not of a high quality. It sounds more synthetic, lacking naturalism. Hence, it becomes difficult to recognize the speaker. The second technique, which comes under the class of waveform coders, gives much better quality at the cost of a much higher bit rate (typically. 32 kb/s).

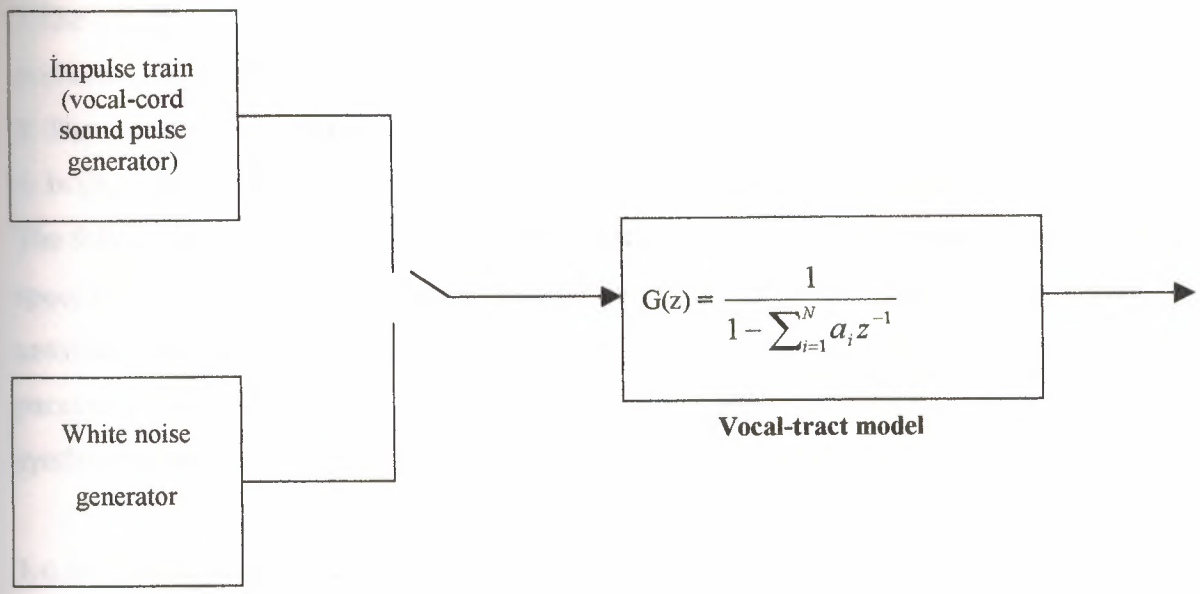


Figure 1.13 Speech-production model

The main reason for linear prediction being widely used in speech coding is that speech signals can be accurately modelled, as in Figure 1.13. Here, the all-pole filter is the vocal tract model. The excitation to this model, u (n), is either a white noise in the case of unvoiced sounds (fricatives such as /s/ and /f/), or an impulse train in the case of voiced sounds (vowels such as /i/). The period of the impulse train, known as the pitch period, and the power of the white noise, known as the excitation level, are parameters of the speech model, which are to be identified in the coding process.

1.6.9 Linear Predictive Coding (LPC)

Speech signal is a highly non-stationary process. The vocal-tract shape undergoes variations to generate different sounds in uttering each word. Accordingly, in LPC, to code a speech signal, it is first partitioned into segments of 10-30 ms long. These segments are short enough for the vocal-tract shape to be nearly stationary, so that the parameters of the speech-production model of Figure 1.13 could be assumed fixed. Then, the following steps are used to obtain the parameters of each segment:

1. Using the predictor structure shown in Figure 1.12, the predictor coefficients. the a_is. are obtained by minimizing the prediction error e (n) in the least-squares sense. for the given segment.

2.The energy of the prediction error e (n) is measured. This specifies the level of excitation required for synthesizing this segment.

3. The segment is classified as voiced or unvoiced.

4. In the case of voiced speech, the pitch period of the segment is measured.

The following parameters are then stored or transmitted for every segment, as the coded speech: (i) the predictor coefficients, (ii) the energy of the excitation signal, (iii) voiced! unvoiced classification, and (iv) the pitch period in the case of voiced speech. These parameters can then (when necessary) be used in a model similar to Figure 1.13 to synthesize the speech signal.

1.6.10 Waveform Coding.

The most direct way of waveform coding is the standard pulse code modulation (PCM) technique, where the speech signal samples are directly digitized

a prescribed number of bits to generate the information bits associated with the coded speech. Direct quantization of speech samples requires relatively a large number of bits (usually 8 bits per sample) in order to be able to reconstruct the original speech with an acceptable quality.

A modification of the standard PCM, known as differential pulse code modulation (DPCM), employs a linear predictor such as Figure 1.12 and uses the bits associated with the quantized samples of the prediction error, e (n), as the coded speech. The rationale here is that the prediction error, e (n), has a much smaller variance than the input, x (n). Thus, for a given quantization level, e (n) may be quantized with fewer bits, as compared with x (n). Moreover, since the number of information bits per every second of the coded speech is directly proportional to the number of bits used per sample, the bit rate of the DPCM will be less compared with the standard PCM.

The prediction filter used in DPCM can be fixed or be made adaptive. A DPCM system with an adaptive predictor is called an adaptive DPCM (ADPCM). In the case of speech signals, use of the ADPCM results in superior performance as compared with the case where a non-adaptive DPCM is used. In fact, the ADPCM has been standardized and widely used in practice (International Telecommunication Unit OITU) Recommendation G.726.

Figure 1.14 depicts a simplified diagram of the ADPCM system, as proposed in ITU Recommendation G.726. Here, the predictor is a six-zero, two-pole adaptive IIR filter. The coefficients of this filter are adjusted adaptively so that the quantized error, é (n), is minimized in the mean-square sense. The predictor input, x (n), is the same as the original input, x (n), except for the quantization error in ë (n). To understand the joint operation of the encoder and decoder in Figure 1.14, note that the same signal, é (n), is used as inputs to the predictor structures at the encoder and decoder. Hence, if the stability of the loop consisting of the predictor and adaptation algorithm could be guaranteed, then the steady state value of the reconstructed speech at the decoder, i.e. x'(n), will be equal to that at the encoder. i.e. 4n), since non-equal initial conditions of the encoder and decoder loops will die away after their transient phase.

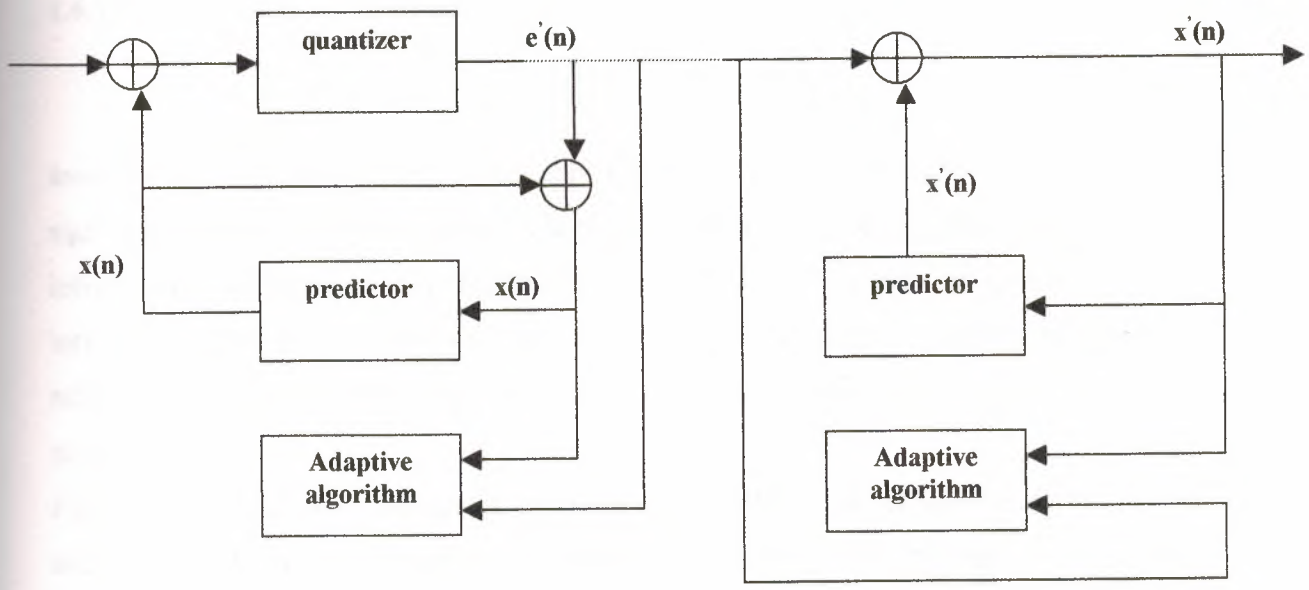


Figure 1.14 ADPCM encoder-decoder

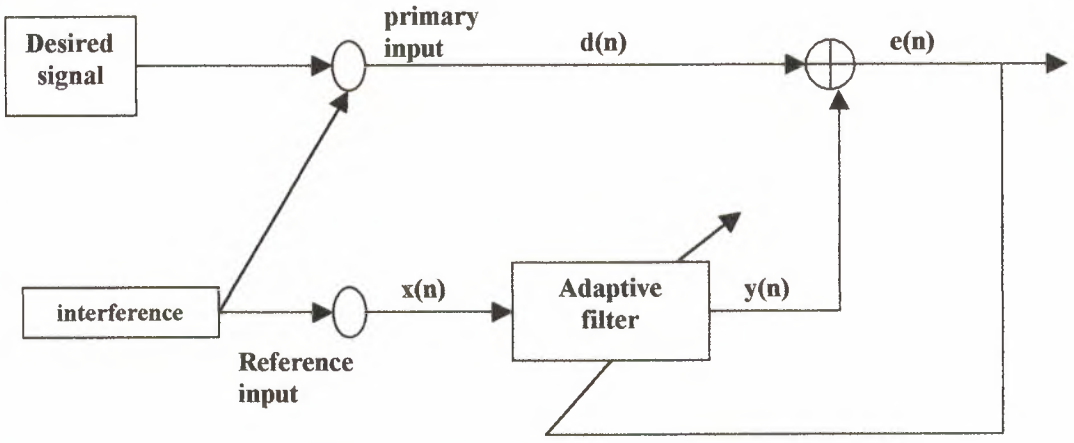


Figure 1.15 Interference cancellation

1.6.11 Interference Cancellation

Interference cancellation refers to situations where it is required to cancel an interfering signal/noise from the given signal, which is a mixture of the desired signal and the interference. The principle of interference cancellation is to obtain an estimate of the interfering signal and subtract that from the corrupted signal. The feasibility of this idea relies on the availability of a reference source from which the interfering signal originates.

Figure 1.15 depicts the concept of interference cancellation, in its simplest form. There are two inputs to the canceller: primary and reference. The primary input is the corrupted signal, i.e. the desired signal plus interference. The reference input, on the other hand, originates from the interference source only. The adaptive filter is adjusted so that a replica of the interference signal that is present in the primary signal appears at its output, y (n). Subtracting this from the primary input results in an output that is cleared from interference, thus the name interference cancellation.

We note that the interference cancellation configuration of Figure 1.15 is different from the previous cases of adaptive filters, in the sense that the residual error (which was discarded in other cases) is the cleaned-up signal here. The desired signal in the previous cases has been replaced here by a noisy (corrupted) version of the actual desired signal. Moreover, the use of the term 'reference' to refer to the adaptive filter input is clearly related to the role of this input in the canceller.

In the rest of this section we present some specific applications of interference cancelling.

1.6.12 Echo Cancellation In Telephone Lines

Echoes in telephone lines mostly occur at points where hybrid circuits are used to convert four-wire networks to two-wire networks. Figure 1.16 presents a simplified diagram of a telephone connection network, highlighting the points where echoes occur. The two-wires at the ends are subscriber loops connecting customers' telephones to central offices. It may also include portions of the local network. The four-wires, on the

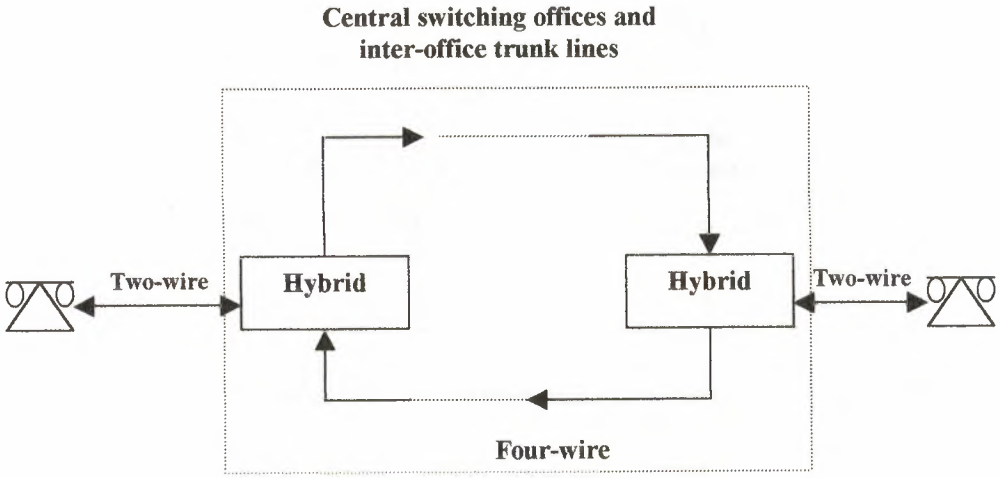


Figure 1.16 Simplified diagram of a telephone network

Other hand, are carrier systems (trunk lines) for medium- to long-haul transmission. The distinction is that the two-wire segments carry signals in both directions on the same lines, while in the four-wire segments signals in the two directions are transmitted on two separate lines. Accordingly, the role of the hybrid circuit is to separate the signals in the two directions. Perfect operation of the hybrid circuit requires that the in-coming signal from the trunk lines should be directed to the subscriber line and that there be no leakage (echo) of that to the return line. In practice, however, such ideal behaviour cannot be expected from hybrid circuits. There would always be some echo on the return path. In the case of voice communications (i.e. ordinary conversation on telephone lines), the effect of the echoes becomes more obvious (and annoying to the speaker) in long-distance calls where the delay with which the echo returns to the

speaker may be in the range of a few hundred milliseconds. In digital data transmission, both short- and long-delay echoes are serious.

As noted earlier, and also can clearly be seen from Figure 1.17, the problem of echo cancellation may be viewed as one of system modelling. An adaptive filter is put between the in-coming and out-going lines of the hybrid. By adapting the filter to realize an approximation of the echo path, a replica of the echo is obtained at its output. This is then subtracted from the out-going signal to clear that from the undesirable echo. Echo cancellers are usually implemented in transversal form. The time spread of echoes in a typical hybrid circuit is in the range 20—30 ms. If we assume a sampling rate of 8 kHz for the operation of the echo canceller, then an echo spread of 30 ms requires an

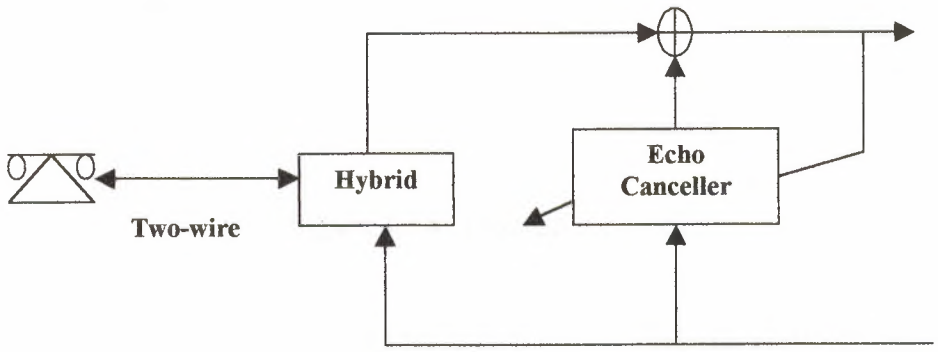


Figure 1.17 Adaptive echo canceller

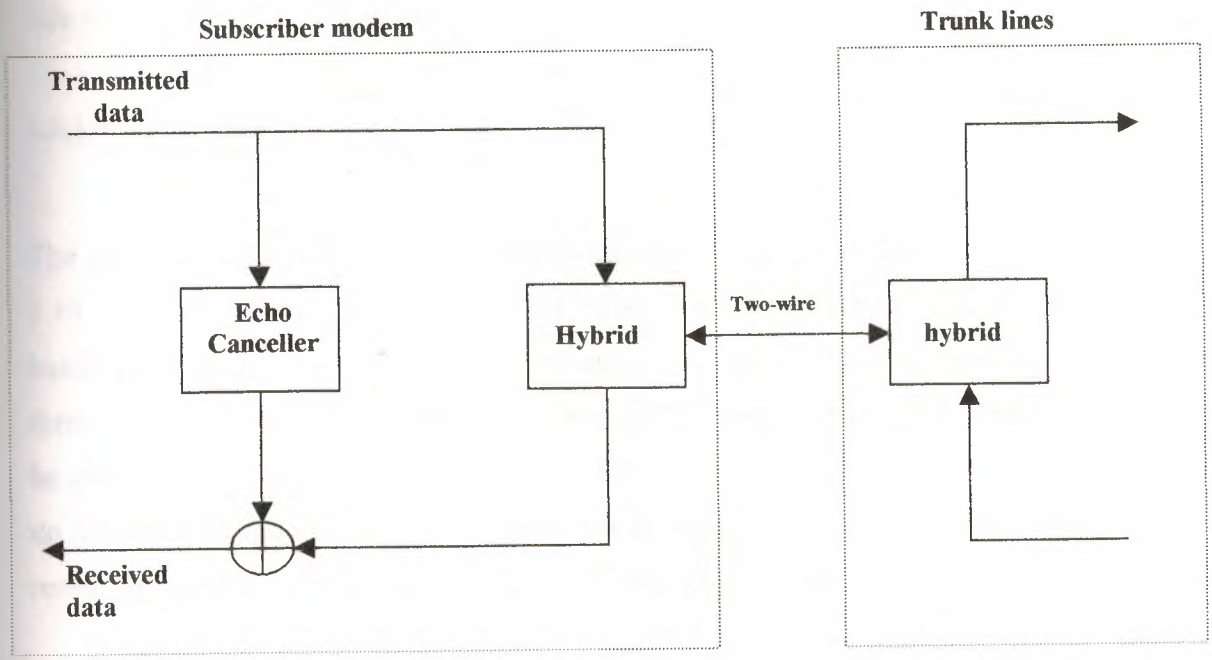


Figure 1.18 Data echo canceller

Adaptive filter with at least 240 taps (30 ms x 8 kHz). This is a relatively long filter, requiring a high-speed digital signal processor for its realization. Frequency domain processing is often used to reduce the high computational complexity of long filters.

The echo cancellers described above are applicable to both voice and data transmission. However, more stringent conditions need to be satisfied in the case of data transmission. To maximize the usage of the available bandwidth, full-duplex data transmission is often used. This requires the use of a hybrid circuit for connecting the data modem to the two-wire subscriber loop, as shown in Figure 1.18. The leakage of the transmitted data back to the receiver input is thus inevitable and an echo canceller has to be added, as indicated in Figure 1.18. However, we note that the data echo cancellers are different from the voice echo cancellers used in central switching offices in many ways. For instance, since the input to the data echo canceller is data symbols, it can operate at the data symbol rate, which is in the range of 2.4-3 kHz (about three times smaller than the 8 kHz sampling frequency used in voice echo cancellers). For a given echo spread, a lower sampling frequency implies fewer taps for the echo canceller. Clearly, this greatly simplifies the implementation of the echo canceller. On the other hand, the data echo cancellers require a much higher level of echo cancellation to ensure the reliable

transmission of data at higher bit rates. In addition, the echoes returned from the other side of the trunk lines should also be taken care of.

1.6.13 Acoustic Echo Cancellation

The problem of acoustic echo cancellation can be best explained by referring to Figure 1.19, which depicts the scenario that arises in teleconferencing applications. A loudspeaker in a room broadcasts the speech signal from a far-end speaker, received through a communication channel, and a microphone picks up its echo. This echo must be cancelled to prevent its feedback to the far-end speaker. The microphone also picks up the near-end speaker's speech and possible background noise, which may exist in the room. An adaptive transversal filter with sufficient length is used to model the

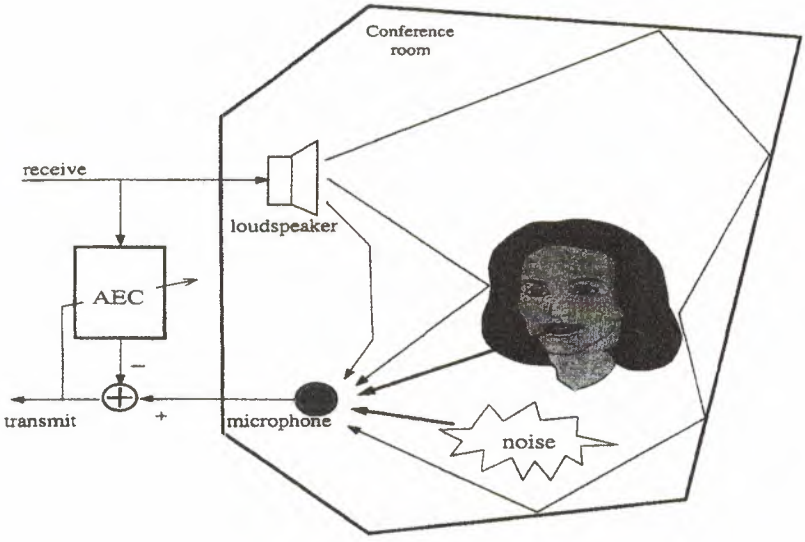


Figure 1.19 Acoustic echo cancellation. Reprinted from farhang- boroujent

Acoustics of the room. A replica of the loudspeaker echo is then obtained and subtracted from the microphone signal prior to transmission.

Clearly, the problem of acoustic echo cancellation can also be posed as one of system modelling. The main challenge here is that the echo paths spread over a relatively long length in time. For typical office rooms, echoes in the range 100—250 ms spread is quite common. For a sampling rate of S kHz, this would mean 800—2000 taps! Thus, the main problem of acoustic echo cancellation 15 that of realizing very long adaptive filters. In addition. since speech is a low-pass signal, it becomes necessary to use special algorithms to ensure fast adaptation of the echo canceller.

1.6.14 Active Noise Control

Active noise control (ANC) refers to situations where acoustic antinoise waves are generated from electronic circuits. The ANC can be best explained by the following example.

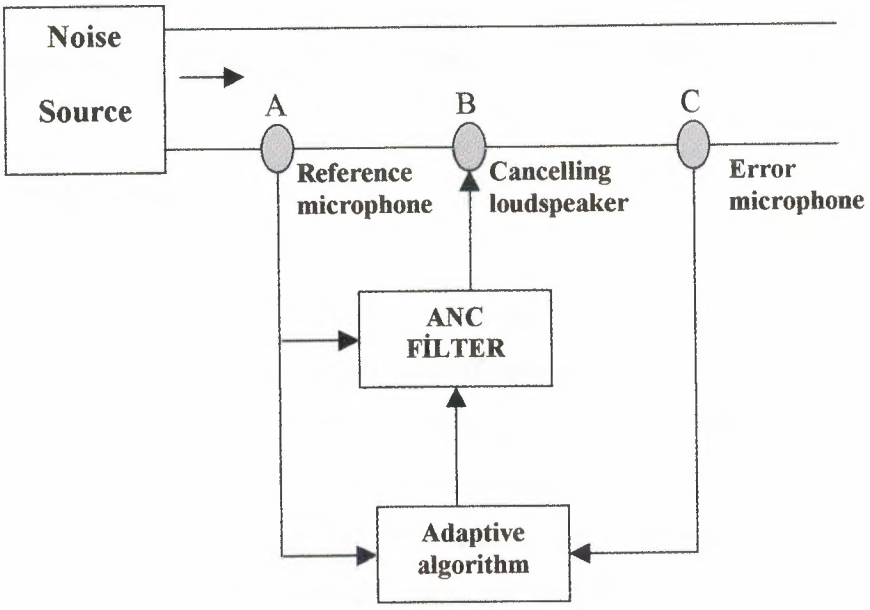


Figure 1.20 Active noise cancellation in a narrow duct

A well-exain1ned application of ANC is cancellation of noise in narrow ducts, such as exhaust pipes and ventilation systems, as illustrated in Figure 1.20. A microphone at position a picks up the acoustic noise traveling along the duct. This is used as reference Input to an ANC filter whose parameters are adapted so that its output, after conversion to an acoustic wave (through the cancelling loudspeaker), is equal to the negative value of the duct noise at position B, thereby cancelling that. The residual noise, picked up by the error microphone at position C. is the error signal used for adaptation of the ANC filter.

Comparing this ANC set-up with the interference cancellation set-up given in Figure 1.15, we may note the following. The source of interference here is the duct noise, the reference input is the noise picked up by the reference microphone, the desired output (i.e. what we wish to see after cancelling the duct noise) is zero. and the primary input is

the duct noise reaching position B. Accordingly, the role of the ANC filter is to model the response of the duct from position A to B.

The above description of ANC assumes that the duct is narrow and the acoustic noise waves are travelling along the duct, which is like a one-dimensional model. The acoustic models of wider ducts and large enclosures, such as cars and aircraft, are usually more complicated. Multiple microphones/loudspeakers are needed for successful implementation of ANC5 in such enclosures. The adaptive filtering problem is then that of a multipl-input-multiple-output system. Nevertheless, the basic principle remains the same, i.e. the generation of antinoise to cancel the actual noise.

1.6.15 Beamforming

In the applications that have been discussed so far the filters/predictors are used to combine samples of the input signal(s) at different time instants to generate the output. Hence, these are classified as temporal filtering. Beamforming, however, is different from these in the sense that the inputs to a beamformer are samples of incoming signals at different positions in space. This is called spatial filtering. Beamforming finds applications in communications, radar and sonar and also imaging in radar and medical engineering.

In spatial filtering, a number of independent sensors are placed at different points in space to pick up signals coming from various sources (see Figure 1.21). In radar and

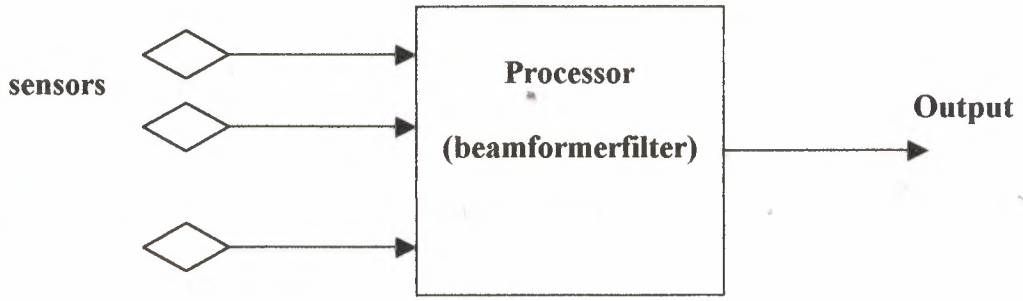


Figure 1.21 Spatial filtering (beam forming)

Communications, the signals are usually electromagnetic waves and the sensors are thus antenna elements. Accordingly. The term antenna array is often used to refer to these

applications of beamformers. In sonar applications, the sensors are hydrophones designed to respond to acoustic waves.

In a beamformer, the sample of signals picked up by the sensors at a particular instant of time constitutes a snapshot. The samples of snapshot (spatial samples) play the same role as the successive (temporal) samples of input in a transversal filter. The beamformer filter linearly combines the sensors' signals so that signals arriving from some particular directions are amplified, while signals from other directions are attenuated. Thus, in analogy with the frequency response of temporal filters, spatial filters have responses that vary according to the direction of arrival of the in-coming signal(s). This is given in the form of a polar plot (gain vs. angle) and is referred to as the beam pattern.

In many applications of beamforiners. The signals picked up by sensors are narrow-band having the same carrier (centre) frequency. These signals differ in their directions-of-arrival, which are related to the location of their sources. The operation of beamformers in such applications can be best explained by the following example.

Consider an antenna array consisting of two omni-directional elements A and B, as presented in Figure 1.22. The tone (as an approximation to narrow-band) signals s (n) = $\alpha \cos w_0 n$ and v (n) = $\beta \cos w_0 n$ arriving at angles 0 and θ_0 (with respect to the line perpendicular to the line connecting A and B), respectively, are the inputs to the array (beamformer) filter which consists of a phase-shifter and a subtracter. The signal s (n)

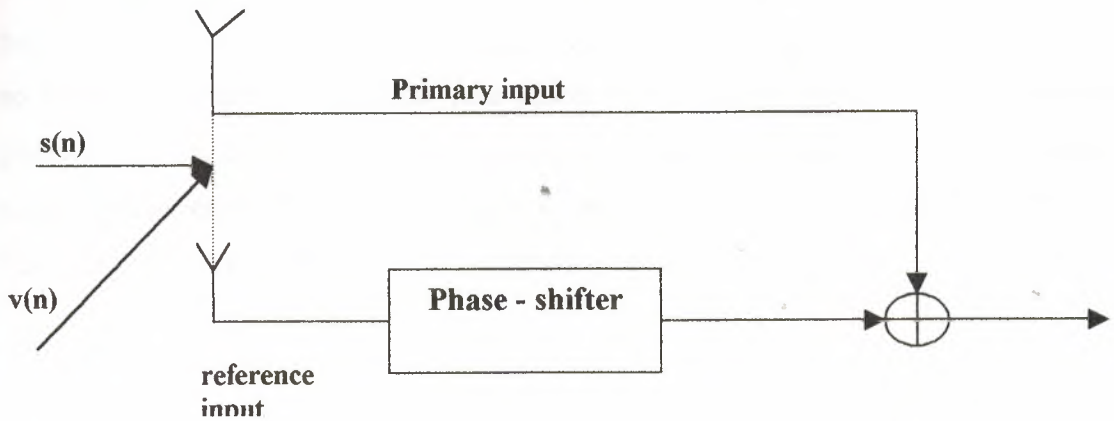


Figure 1.22 A two-element beamformer

Arrives at elements A and B at the same time, whereas the arrival times of signal v (n) at A and B are different. We may thus write

Where the subscripts A and B are used to denote the signals picked up by elements A and B, respectively, and p is the phase-shift arising from the time delay of arrival of v(n) at element A with respect to its arrival at element B.

Now, if we assume that s (n) is the desired signal and v (n) is an interference, then, by inspection, we can see that if the phase-shifter phase is chosen equal to p. then the interference, v (n), will be completely cancelled by the beamforiner. The desired signal, on the other hand, reaches the beamformer output as α (cos ω_0n - cos (ω_0n - φ)). Which is non-zero (and still holding the information contained in its envelope, ü) when $\varphi \neq 0$, i.e. when the interference direction is different from the direction of the desired signal. This shows that we can tune a beamformer to allow the desired signal arriving from a direction to pass through it, while rejecting the unwanted signals (interferences) arriving from other directions.

The idea of using a phase-shifter to adjust the beam pattern of two sensors. is easily extendible to the general case of more than two sensors. In general, by introducing appropriate phase shifts and also gains at the output of the various sensors and summing up these outputs, we can realize any arbitrary' beam pattern. This is similar to the selection of tap weights for a transversal filter so that the filter frequency response becomes a good approximation to the desired response. Clearly, by increasing the number of elements in the array, better approximations to the desired beam pattern can be achieved.

The final point that we wish to add here is that in cases where the input signals to the beamformer are not narrow-band, a combination of spatial and temporal filtering needs to be used. In such cases, spatial information is obtained by having sensors at different positions in space, as was discussed above. The temporal information is obtained by using a transversal filter at the output of each sensor. The output of the broadband beam-former is the summation of the outputs of these transversal filters.

CHAPTER 2
BLOCK IMPLEMENTATION OFADAPTIVE FILTERS

2.1 Introduction

There are certain applications of signal processing that require adaptive filters the lengths of which exceed a few hundred or even a few thousand taps. For instance, to prevent the return of speaker echo to the far end of the telephone line, in the application of hand-free telephony, the use of an acoustic echo canceller the length of which exceeds a few thousand taps is not uncommon. Other applications, such as active noise control and the equalization of some communication channels, may also require adaptive filters with exceedingly long lengths. In such applications we find that even the conventional LMS algorithm, which is known for its simplicity, is computationally expensive to implement.

In this chapter we show how bldck processing of the data samples can significantly reduce the computational complexity of adaptive filters. In block processing (or block implementation), a block of samples of the filter input and desired output are collected and then processed together to obtain a block of output samples. Thus, the process involves serial to parallel conversion of the input data, parallel processing of the collected data, and parallel to serial conversion of the generated output data. This is illustrated in Figure 2.1. The computational complexity of the adaptive ifiter can then be reduced significantly through elegant parallel processing of the data samples. We note that the parallel processing involved in Figure 2.1 is repeated only after the collection of every block of data samples. Thus, a good measure of the computational complexity in a block processing system is given by the number of operations required to process one block of data divided by the block length. We may then note that the sharing of the processing time among the samples in each block is the key to achieving high computational efficiency.

In this chapter we discuss an efficient technique for block processing of data samples in the adaptive filtering context. This involves a special implementation of the LMS

algorithm which is called the block LMS (BLMS). We introduce a computationally efficient implementation of the BLMS algorithm in the frequency domain. This is called the fast BLMS (FBLMS) algorithm. The high computational efficiency of the FBLMS algorithm is achieved by employing the following result from the theory of digital signal processing. Linear convolution of time domain sequences can be efficiently implemented using frequency domain processing. In particular, the linear convolution of an indefinite

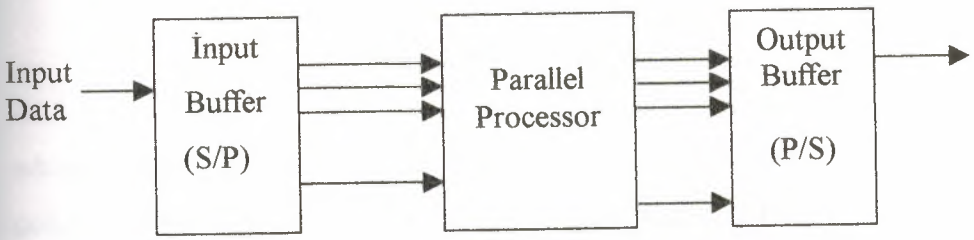


Figure 2.1 Schematic of a block processing System

length sequence, x(n), with a finite length sequence, h(n) (which may be that of the impulse response of a FIR filter), is obtained by partitioning x(n) into a set of overlapping finite duration blocks, finding the circular convolution of h(n) (appended with some extra zeros) with these blocks, and then choosing the portions of the circular convolutions that match the desired linear convolution samples. The circular convolutions can be very efficiently performed in the frequency domain, using the properties of the discrete Fourier transform (DFT).

Throughout this chapter we adopt the following notations. As in the previous chapters, bold lower-case letters represent vectors, bold upper-case letters denote matrices, and non-bold lower-case letters represent scalars. As before, we use n as the time (sample) index. The letter k is reserved for block index. The subscript ~ is used to refer to frequency domain signals, e.g. the DFT of the time domain vector xis denoted as x~. In the derivations that follow we frequently need to extend the dimensions of vectors and matrices to some certain dimensions by appending zeros. We use 0 (in bold) to refer to zero vectors and zero matrices and the dimensions of these zero vectors and/or matrices will be clear from the context,

Our discussion in this chapter is limited to the case where the filter input, x(n), and the desired output, d(n), are real-valued processes. Ilowever, we note that the frequency

domain equivalent of these processes is complex-valued and hence the LMS recursion that is used is the complex LMS algorithm.

2.2 Block LMS Algorithm

The conventional LMS algorithm that was introduced in Chapter 6 uses the following recursion to adjust the tap weights of an adaptive filter:

$$W(n+1) = w(n) + 2\mu e(n)x(n) \tag{2.1}$$

where $x(n) = [x(n)\ x(n-1)...x(n-N+1)]^T$ and $w(n)=[w_0(n)\ w_1(n)\ ...w_{n-1}(n)]^T$ are the column vectors consisting of the filter tap inputs and tap weights respectively, $e(n) = d(n) - y(n)$ is the output error, $d(n)$ and $y(n) = w^T(n)x(n)$ are the desired and actual output of the filter, respectively, and μ is the step-size parameter We also recall that the conventional LMS algorithm is a stochastic implementation of the steepest-descent method using the instantaneous gradient vector

$$\nabla_w e^2(n) = -2e(n)x(n). \tag{2.2}$$

The block LMS (BLMS) algonthm works on the basis of the following strategy. The filter tap weights are updated once after the collection of every block of data samples. The gradient vector used to update the filter tap weights is an average of the instantaneous gradient vectors of the form (8.2) which are calculated during the current block. Using k to denote the block index, the BLMS recursion is obtained as

$$W(k+1)=w(k) +2\mu_B \frac{\sum_{I=0}^{L-1} e(kL+\imath)x(kL+\imath)}{L} \tag{2.3}$$

where L is the block length and μB is the algorithm step-size parameter. We also note that for the computation of the output error samples $e(kL + i) = d(kL + i) - y(kL+i)$; for $i = 0.1... L - 1$, the output samples $y(kL + i) = w^T(k)x(kL + i)$ are calculated using the update of the filter tap-weight vector, w(k), from the previous block.

The derivations presented in the following sections make use of, to a large extent, the vector formulation of the BLMS algorithm. Hence, we now present this formulation. Define the matrix

$$X(k) = [x(kL)\ x(kL+1)\ \ldots\ x(kL+L-1)]^T, \tag{2.4}$$

and the column vectors

$$d(k) = [d(kL)\ d(kL+1)\ \ldots\ d(kL+L-1)]^T \tag{2.5}$$

$$y(k) = [y(kL)\ y(kL+1)\ \ldots\ y(kL+L-1)]^T \tag{2.6}$$

$$e(k) = [e(kL)\ e(kL+1)\ \ldots\ e(kL+L-1)]^T \tag{2.7}$$

and note that

$$y(k) = X(k)w(k) \tag{2.8}$$

and

$$e(k) = d(k) - y(k). \tag{2.9}$$

We also note that

$$\sum_{i=0}^{L-1} e(kL+i)x(kL+i) = X^T(k)e(k). \tag{2.10}$$

Substituting (2.10) in (2.3) we obtain

$$w(k+1) = w(k) + \frac{2\mu B}{L} X^T(k)e(k). \tag{2.11}$$

Equations (2.8) (2.9) and (2.11), which correspond to filtering, error estimation and tap-weight vector updating, respectively, define one iteration of the BLMS algorithm.

On the basis of our background from the method of steepest-descent and, also, the conventional LMS algorithm, the following comments may be made intuitively:

1. The convergence behaviour of the BLMS algorithm is governed by the eigenvalues of the correlation matrix $R = E[x(n)x^T(n)]$. This follows from the fact that similar to the conventional LMS algorithm, the BLMS algorithm is also a stochastic implementation of the steepest-descent method.

2.The BLMS algorithm has N modes of convergence which are characterized by the time constants

$$\tau_{B,i} = \frac{1}{4\mu_B \lambda_i}, \quad \text{for } i = 0,1,\ldots,N-1, \tag{2.12}$$

where the A_1s are the eigenvalues of the correlation matrix R. These time constants are in the unit of iteration (block) interval.

3. Averaging the instantaneous (stochastic) gradient vectors as was done in the BLMS algorithm results in gradient vectors with a lower variance, as compared with that in the conventional LMS algorithm. This allows the use of a larger step-size parameter for the BLMS algorithm compared to the conventional LMS algorithm. For block lengths. L, comparable or less than the filter length. N, and small misadjustments, in the range of 10% or less, misad)ustment, MB, of the BLMS algorithm can be approximated by the following expression:

$$M_B \approx \frac{\mu B}{L} \text{tr}[R] \tag{2.13}$$

This result is derived in Appendix 8A.

And

$$\mu_B = L\mu \tag{2.14}$$

where μ is the step-size parameter of the conventional LMS algorithm. Substitutint (2.14) in (2.12), we get

$$\tau_{B,i} = \frac{1}{4L\mu\lambda_i} \quad \text{block interval} \tag{2.15}$$

$$\tau_{B,i} = \frac{1}{4\mu\lambda_i} \quad \text{sample interval} \tag{2.16}$$

We conclude that the convergence behaviour of BLMS and conventional LMS algorithms are the same.

2.3 Mathematical Background

The mathematical and signal processing tools required for the rest of this chapter are briefly reviewed in this section. In particular, we discuss how time domain linear convolutions can be efficiently performed using discrete Fourier transform. We also introduce circular matrices and review some of their properties that are relevant to our studv of BLMS alttonthms.

2.3.1 Linear Convolution Using The Discrete Fourier Transtorm

We consider the filtering of a sequence x(n) through a FIR filter with coefficients w_0, w_1, ... w_{N-1} . This involves computation of the linear convolution

$$y(n) = \sum_{i=0}^{N-1} w_i x(n-i). \tag{2.17}$$

This process requires N multiplications and N - 1 additions for computing every sarr of the output, y(n). When N is large, the samples of y(n) can be obtained with a redu number of multiplications and additions, as discussed below.

Let us define the column vector x(k) of length $N' = N + L - 1$ as

$$x(k)=[x(kL-N+l)\ \ x(kL-N+2)\ \ldots\ x(kL+L-l)]^T \tag{2.18}$$

and w(k) of length N' as

$$w(k) = \begin{bmatrix} w(k) \\ 0 \end{bmatrix} \tag{2.19}$$

where $w(k) = [w_0(k)\ w_2(k)\ \ldots.\ \ w_{N-1}(k)]^T$ is the filter tap-weight vector, and 0 rele a column vector consisting of L - 1 zeros. In order to maintain uniformity in derivations of the subsequent sections. the block index k has been added to the filtel weights. indicating that the weights vary only from block to block, as happens ill implementation of the ELMS algorithm.

From the properties of the DFT we know that the circular convolution of w(k) and x(k) can be obtained by transforming both vectors to their respective frequency dor equivalents (using the DFT), performing an element-by-element multiplication or transformed samples, and transforming the result back to the time domain (usint inverse DFT (ID FT)). This process can be efficiently implemented by using the EFT inverse FET (IFFT) algorithms. Examining the circular convolution of w(k) and reveals that only the last L elements of the result coincide with the correspon elements of the linear convolution (2.17).

$$\begin{bmatrix} * \\ * \\ .. \\ .. \\ * \\ y(kL) \\ y(kL+1) \\ .. \\ .. \\ y(kL+L-1) \end{bmatrix} = \begin{bmatrix} x(kL-N+1) & x(kL+L-1) & x(kL+L-2) & .... & x(kL-N+2) \\ x(kL-N+2) & x(kL-N+1) & x(kL+L-1) & .... & x(kL-N+3) \\ .. & .. & .. & .. & .. \\ .. & .. & .. & .. & .. \\ x(kL-1) & x(kL-2) & x(kL-3) & .... & x(kL) \\ x(kL) & x(kL-1) & x(kL-2) & .... & x(kL+1) \\ x(kL+1) & x(kL) & x(kL-1) & .... & x(kL+2) \\ .. & .. & .. & .. & .. \\ .. & .. & .. & .. & .. \\ x(kL+L-1) & x(kL+L-2) & x(kL+L-3) & .... & x(kL-N+1) \end{bmatrix} = \begin{bmatrix} w_0(k) \\ w_1(k) \\ .. \\ .. \\ w_{N-2}(k) \\ w_{N-1}(k) \\ 0 \\ .. \\ .. \\ 0 \end{bmatrix}$$

(2.20)

In (2.20) the elements represented by asterisks correspond to circular convoll results which do not coincide with linear convolution samples, as required by (2.17)

Careful examination of the summations related to these elements reveals that the samples experience some discontinuity in their order. The procedure explained by (2.20) is commonly known as the overlap-save method. This name reflects the fact that in each block of input, x(k) consists of L new samples and N - 1 overlapped samples from the previous bock(s). Another equally efficient method for the computation of linear convolutions using DFT is the overlap – add method. However, the overlap-add method has been found to be computationally efficient than the overlap-save method when applied to the implementation of BLMS algorithm.

2.3.2 Circular Matrices

Circular matrices are used extensively in the derivation and analysis of the fast BLMS. (FBLMS) algorithm. Hence, it is verv useful as well as necessary to have a go understanding of the properties of these matrices before we start our discussion of the FBLMS algorithm.

Consider the M x M circular matrix

$$A_C = \begin{bmatrix} a_0 & a_{M-1} & a_{M-2} & \cdots & a_0 \\ a_1 & a_0 & a_{M-1} & \cdots & a_2 \\ .. & .. & .. & .. & .. \\ .. & .. & .. & .. & .. \\ a_{M-2} & a_{M-3} & a_{M-4} & \cdots & a_{M-1} \\ a_{M-1} & a_{M-2} & a_{M-3} & \cdots & a_0 \end{bmatrix} \tag{2.21}$$

Clearly, the name circular' refers to the fact that each row (column) of A~ is obtaine by circularly shifting the previous row (column) by one element. A special propert of circular matrices that is extensivelv used in the following sections is that sue matrices are diagonalized by DFT matrices. That is, if Y is the M x M DFT matri defined as

$$F = \begin{bmatrix} 1 & 1 & 1 & \cdots & 1 \\ 1 & e^{-j2\pi/M} & e^{-j4\pi/M} & \cdots & e^{-j2\pi(M-1)/M} \\ 1 & e^{-j4\pi/M} & e^{-j8\pi/M} & \cdots & e^{-j4\pi(M-1)/M} \\ .. & .. & .. & .. & .. \\ .. & .. & .. & .. & .. \\ 1 & e^{-j2\pi(M-1)/M} & e^{-j4\pi(M-1)/M} & \cdots & e^{-j2\pi(M-1)^2/M} \end{bmatrix} \tag{2.22}$$

then

$$A_F = FA_CF^{-1} \tag{2.23}$$

is a diagonal matrix. Furthermore, the diagonal elements of A_F correspond to the DFT of the first column of A_C. In matrix notation, this may be written as

$$A_F = \text{diag}[a_F] \tag{2.24}$$

where $a_F = Fa$, $a = [a_0 \ a_1 \ \dots \ a_{M-1}]^T$ is the first column of A_C, and diag[a_F] denotes the diagonal matrix consisting of the elements of a_F. This can be proved as follows.
Since F is a DFT matrix, recall that

$$F^{-1} = \frac{1}{M} F \tag{2.25}$$

where M is the length of DFT and an asterisk denotes complex conjugation. In other words, the /th column of F^{-1} can be given as

$$g_l = \frac{1}{M} f_l \tag{2.26}$$

where f_1,is the ith column of the DFT matrix F given by

$$f_l = (1 \ \ e^{-J2\pi l/M} \ \ e^{-J4\pi l/M} \ \ \dots e^{-J2(M-1)\pi l/M})^T \tag{2.27}$$

Next, by direct insertion.

$$A_c g_l = a_{F,l} g_1 \ , \text{ for } l = 0,1,\dots,M-1 \tag{2.28}$$

Where $a_{F.l} = \sum_{i=0}^{M-1} a_i e^{-j2\pi li/M}$ is the l element of a_F. Using (8.24). theM equations in (8.29) may be put together to obtain

$$A_C F^{-1} = F^{-1} A_F \tag{2.29}$$

Premultiplying (8.29) on both sides by F gives (8.23).

Another important result of the circular matrices which will be useful for our later application is derived next. Applying Hermitian transposition on both sides of (8.23), we obtain

$$A_F^H = F^{-H} A_C^H F^H \tag{8.30}$$

where F^{-H} is the short-hand notation for $(F^{-1})^H$ Since A_F is diagonal, $A_F^H = A_F$.Furthermore, from (2.22) and (2.25), $F^{-H} = (1/M) F$ and $F^H = MF^{-1}$ since $FT = F$ Using these in (2.30) we get

$$A_F = F A_c^H F^{-1} \tag{2.31}$$

When elements of A_C are real-valued, $A_C^H = A_C^T$ and thus (2.30) may be written as

$$A_F = F A_C^T F^{-1} \tag{2.32}$$

2.3.3 Window Matrices And Matrix Formulation Of The Overlap-Save Method

Let us define the N' x N' circular matrix, for N' = L + N - I, as

$$X_C(k) = \begin{bmatrix} x(kL-N+1) & x(kL+L-1) & x(kL+L-2) & \ldots & x(kL-N+2) \\ x(kL-N+2) & x(kL-N+1) & x(kL+L-1) & \ldots & x(kL-N+3) \\ .. & .. & .. & .. & .. \\ .. & .. & .. & .. & .. \\ .. & .. & .. & .. & .. \\ x(kL+L-1) & x(kL+L-2) & x(kL+L-3) & \ldots & x(kL-N+1) \end{bmatrix} \tag{2.33}$$

We note that this is nothing but the data matrix on the right-hand side of (2.20).

We also define the length N' column vector

$$y(k) = \begin{bmatrix} 0 \\ y(k) \end{bmatrix}, \tag{2.34}$$

where y(k), as defined in (2.6) is the column vector consisting of the output samples the kth block, and 0 is the length N - I zero vector. Let us denote by $y_c(k)$ the colunt vector that appears on the left-hand side of (2.20) and note that y(k) can be obtained from $y_c(k)$ by substituting all the elements in the latter with zeros.

This substitution ca be written in the form of a matrix-vector product as

$$y(k) = P_{0.L}y_c(k) \tag{2.35}$$

where $P_{0.L}$ is the N' x N' windowing matrix defined as

$$P_{0..L} = \begin{bmatrix} 0 & 0 \\ 0 & I_L \end{bmatrix} \tag{2.36}$$

with I_L being the L x L identity matrix and the 0s are zero matrices with appropriat dimensions. Using (2.20), (2.19) and the above definitions, we obtain

$$y(k) = P_{O,L}X_C(k)w(k). \tag{2.37}$$

Implementation of (2.37) in the frequency domain can now be obtained simply by noting that (2.38) may be written as

$$y(k) = P_{O,L}\, F^{-1}\, FX_C(K)\, F^{-1}\, Fw(k), \tag{2.38}$$

where F is the N' x N' DFT matrix. Next, define

$$w_F(k) = Fw(k) \tag{2.39}$$

and

$$X_F(k) = FX_C(k)F^{-1} \tag{2.40}$$

and note that $X_F(k)$ is the diagonal matrix consisting of the elements of the DFT of the first column of $X_C(k)$, since the latter is a circular matrix. We also note that the first column of $X_C(k)$ is the input vector x(k), as defined in (2.18). Using (2.38) and (2.39) in (2.39) we obtain

$$y(k) = P_{0,L}F^{-1}X_F(k)\,w_F(k) \tag{2.41}$$

This equation has the following interpretation. Since $X_F(k)$ is diagonal. $X_F(k)$ $w_F(k)$ is nothing but the element-by-element multiplication of the filter input and its coefficients in the frequency domain. This gives the output samples of the filter in the frequency domain. Premultiplication of this result by F^{-1} converts the frequency domain samples of the output to the time domain. Furthermore. premultiplying the result by the windowing matrix $P_{0.L}$ results in selecting only those samples that coincide with the required linear convolution samples.

2.4 The FBLMS Algorithm

The FBLMS algorithm, as mentioned in the introduction. is nothing but a fast (numerically efficient) implementation of the BLMS algorithm in the frequency domain.

Equation (2.42) corresponds to the filtering part of the FBLMS algorithm. Element-by~element multiplication of the frequency domain samples of the input and filter coefficients is followed by an IDET and a proper windowing of the result to obtain the output vector y(k), in extended form, as defined by (2.34). The vector of desired outputs, in extended form. is defined as

$$d(k) = \begin{bmatrix} 0 \\ d(k) \end{bmatrix} \tag{2.42}$$

where d(k) is defined by (2.5) and 0 is the N - 1 element zero column vector. We also define the extended error vector

$$e(k) = d(k) - y(k) \tag{2.43}$$

To obtain the frequency domain equivalent of the recursion (2.11). we replace w(k) and e(k) by their extended versions and note that (2.11) may also be written as

$$w(k + 1) = w(k) + 2\mu P_{N.0}\, X_C^T(k)\, e(k) \tag{2.44}$$

where $X_C(k)$ is the circular matrix of samples of the filter input as defined by (2.33), $\mu = \mu_B/L$, and

$$P_{N,0} = \begin{bmatrix} \dot{I}_N & 0 \\ 0 & 0 \end{bmatrix} \tag{2.45}$$

is an N' x N' windowing matrix which ensures that the last L - I elements of the updated weight vector w(k + 1) remain equal to zero after each iteration of (2.44). The fact that (2.1l) and (2.44) are equivalent can easily be shown by substituting for the vectors and matrices in (2.44) and expanding the result

Conversion of the recursion (2.44) to its frequency domain equivalent can be done by premultiplying it on both sides by the DFT matrix F and using the identity $F^{-1} F = \dot{I}$ to obtain

$$w_F(k + 1) = w_F(k) + 2\mu F P_{N,0}\, F^{-1}\, F X_C^T(k)\, F^{-1}\, Fe(k) \tag{2.46}$$

Using (2.40)and the identity (2.33) (2.46)can be written as

$$w_F(k + 1) = w_F(k) + 2\mu P_{N,0}\, X_F(k) e_F(k) \tag{2.47}$$

where $e_F(k) = Fe(k)$ and

$$P_{N,0} = F P_{N,0}\, F^{-1} \tag{2.48}$$

Equations (2.41), (2.43) and (2.47) are the three steps required to complete each iteration of the FBLMS algorithm: namely, Iiltenng, error estimation and tap-weight adaptation respectively. Figure 2.3 depicts a block diagram of the FBLMS algorithm,

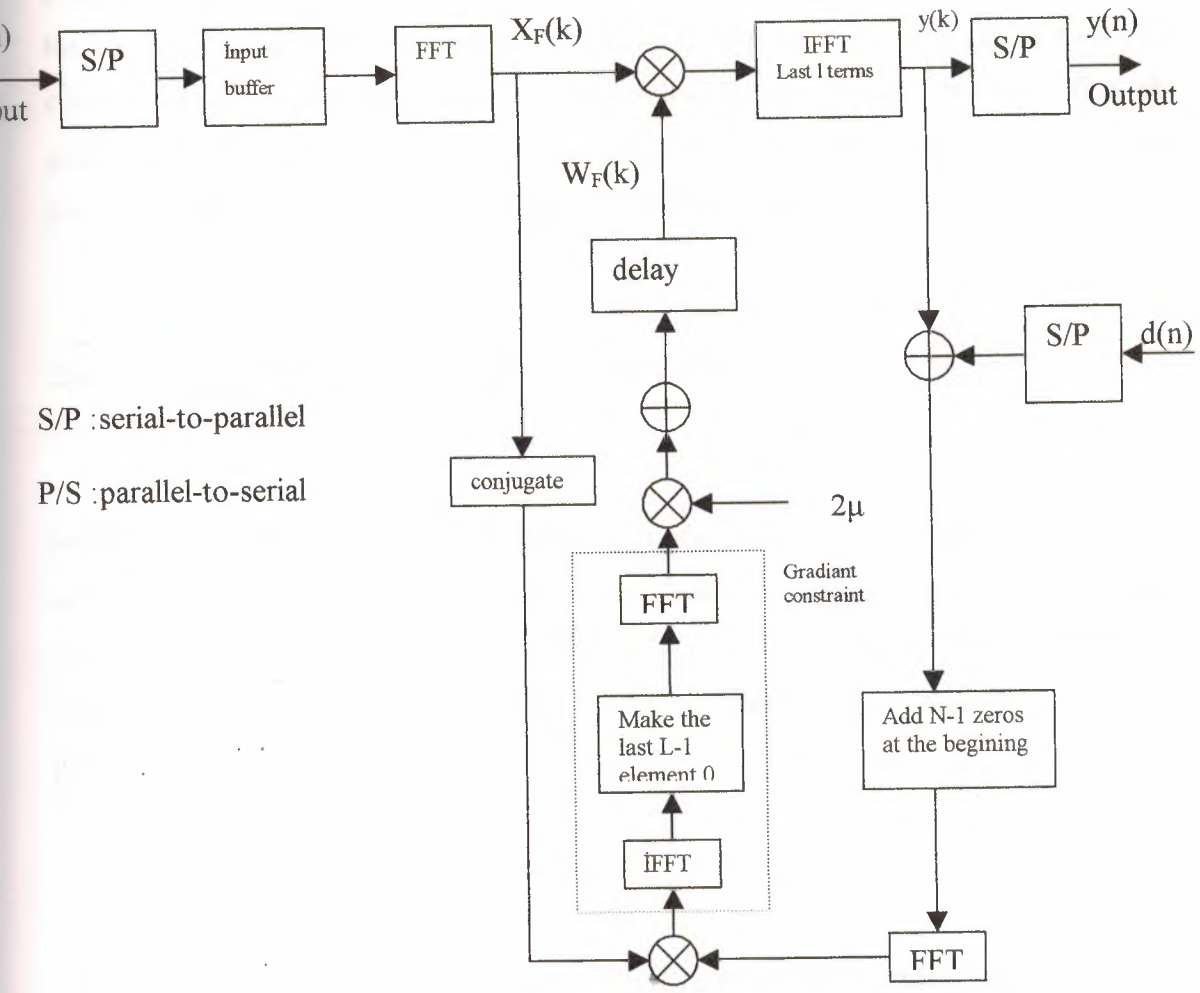


Figure 2.3 Implementation of the FBLMS algorithm

which shows how these steps are realized efficiently. The input samples are collected in an input buffer whose output is the vector x(k), consisting of L new samples and N - 1 samples from the previous block(s). The vector x(k) is converted to the frequency domain and multiplied by the associated tap-weight vector, $w_F(k)$, on an element-by-element basis. This gives the samples of the filter output in the frequency domain which are subsequently converted to the time domain using an IFFT. The last L samples of this

result correspond to the output samples of the current block and are sent to the output buffer as well as the error estimation section. The error vector, e(n), which consists of L elements is extended to the length of N + L - 1 by appending N - I zeros at its beginning and converted to the frequency domain using an FFT algorithm. An element-by-element multiplication of the error and conjugate of the input samples is perforraed in the frequency domain and the result is used to update the filter tap weights. Premultiplication of the gradient vector $X_F(k)e_F(k)$ by $P_{N,0}$ is necessarv to ensure that the last L - 1 elements of the time domain equivalent of the tap-weieht vector $w_F(k)$ are constrained to zero. This constraining operation is implemented by converting the gradient vector $X_F(k)e_F(k)$ to the time domain, making the last L - 1 elements zero, and converting back to the frequency domain. as shown in Figure 2.3.

2.4.1 Constrained And Unconstrained FBLMS Algorithms

Mansour and Gray (1982) have shown that under fairly mild conditions the FBLMS algorithm can work well even when the tap-weight constraining matrix $P_{N,0}$ dropped from (2.47). They have shown that when the filter length, N, is chosen sufficiently large. and the input process, x(n). does not satisfy some specific (unlikely to happen in practice) conditions, the update equation (2.47) and the recursion

$$w_F(k + 1) = w_F(k) + 2\mu X_F(k)\, e_F(k) \tag{2.49}$$

converge to the same set of tap weights. To differentiate between the two cases. (2.49) is called the unconstrained FBLMS recursion, while (2.47) is referred to as the co,istrai'ied FBLMS recursion.

The block diagram given in Figure 2.3 is that of the constrained FBLMS algorithm. However, it is easily converted to the unconstrained FBLMS algorithm if the gradient constraining operation, enclosed by the dotted-line box, is dropped. We may thus note that the unconstrained FBLMS algorithm is much simpler to implement. since two of the five FFTs and IFFTs are deleted from Figure 2.3. As we show in the next section. this simplification is at the cost of a higher misadjustment.

2.4.2 Convergence Behaviour Of The FBLMS Algorithm

In this section we present a convergence analysis of the FBLMS algorithm. We start with the unconstrained recursion (2.49). Substituting (2.41) in (2.43) we get

$$e(k) = d(k) - P_{0,L}\, F^{-1}\, X_F(k)\, w_F(k) \tag{2.50}$$

The fact that the first N - 1 elements of d(k) are aU zero implies that $d(k) = P_{0,L}\, d(k)$. Using this in (2.50) we obtain

$$\begin{aligned} e_F(k) &= P_{0,L}\,(d_F(k) - F^{-1}\, X_F(k)\, w_F(k) \\ &= P_{0,L}\; F^{-1}\,(F\, d(k) - X_F(k)\, w_F(k) \end{aligned} \tag{2.51}$$

Premultiplying (2.51) on both sides by F, we get

$$e_F(k) = P_{0,L}\,(d_F(k) - X_F(k)\, w_F(k)) \tag{2.52}$$

where $d_F(k) = Fd(k)$ and

$$P_{0,L} = F\, P_{0,L}\, F^{-1} \tag{2.53}$$

Substituting (2.52) in (2.49), we obtain

$$w_F(k+1) = w_F(k) + 2\mu X_F(k)\,[P_{0,L}\,(d_F(k) - X_F(k)\, w_F(k))] \tag{2.54}$$

Next. we define the tap-weight error vector

$$v_F(k) = w_F(k) - w_{0.F} \tag{2.55}$$

where $w_{0.F}$ is the optimum value of the filter tap-weight vector in the frequency domain. Using (2.55) in (2.54) we obtain, after some simple manipulation,

$$v_F(k+1) = (1 - 2\mu X_F(k)\, P_{0,L}\, X_F(k))\, v_F(k) + 2\mu X_F(k)\, P_{0,L}\, e_{0.F}(k) \tag{2.56}$$

where $e_{0.F}(k)$ is the optimum error vector obtained when $w_F(k)$ is replaced by $e_{0.F}$

Now. if we use the indepeendence assumption and follow the same procedure, we will find that the convergence of the unconstrained FBLMS algorithm is controlled by the eigenvalues of the matrix

$$R^{u}_{xx} = E\,[X_F(k)\, P_{0,L}\, X_F(k)] \tag{2.57}$$

The matrix R^{u}_{xx} may be evaluated as follows. Substituting (2.40) and (2.53) in (2.57), we obtain

$$R^{u}_{xx} = F\, R^{u}_{xx}\, F^{-1} \tag{2.58}$$

Where

$$R^{u}_{xx} = E[X^{T}_{C}(k)\, P_{0,L}\, X_C(k)] \tag{2.59}$$

A careful examination of Rxuy reveals that when L and N are large and the autocorrela tion function of the input process, x(n).

2.4.3 Step-Normalization

The convergence performance of the FBLMS algorithm can be greatly improved by using individually normalized step-size parameters for each element of the tap-weight vector $w_F(k)$, rather than a common step-size parameter. It is implemented by replacing the scalar step-size parameter μ by the diagonal matrix

$$\mu(k) = \mathrm{diag}[\mu_0(k)\,,\ \mu_1(j)\,,\ \ldots\,,\ \mu_{N'-1}(k)] \tag{2.60}$$

where $\mu_1(k)$ is the normalized step-size parameters for the ith tap.

These are obtained according to the equations

$$\mu_i(k) = \frac{\mu_0}{\sigma^2_{x_{F,i}}(k)}, \qquad \mu_0 \quad \text{for } i = 0,1,\ldots,N'-1, \tag{2.61}$$

where μ_0 is a common unnormalized step-size parameter and the $\sigma^2_{x_{F,i}}(k)$s are the power estimates of the samples of the filter input in the frequency domain, the$x_{F,i}(k)$s. These estimates may be obtained using the following recursion: $\sigma^2_{x_{F,i}}(k) = \beta\sigma^2_{x_{F,i}}(k-1) + (1-\beta)\left|x_{F,i}(k)\right|^2$ for i =0,1,..., N' - 1, where β is a constant close to, but smaller than, one.

2.4.4 FBLMS Misadjustment Equations

Derivation of the misadjustment equations for the various implementations of the FBLMS algorithms is quite tedious and long. This is done in Appendix 8B. The derivations presented in Appendix 8B result in the following misadjustment equations;

$$M^{C}_{FBLMS} \approx \mu N \vartheta_{xx}(0) \tag{2.62}$$

$$M^{U}_{FBLMS} \approx \mu N' \vartheta_{xx}(0) \tag{2.63}$$

$$M^{cN}_{FBLMS} \approx \mu_0 N/N' \tag{2.64}$$

$$M^{uN}_{FBLMS} \approx \mu_0 \tag{2.65}$$

In these equations the superscripts c and u refer to the constrained and unconstrained versions of the FBLMS algonthm, respectively, and N indicates that the step-normalization has been applied.

It can be immediately concluded that the constrained FBLMS algorithm outperforms its unconstrained counterpart. in the sense that the former results in a lower misadjustment for a given step-size parameter. Equivalently, for a given misadjustment the constrained FBLMS algorithm converges faster than its unconstrained counterpart. The difference between the two algorithms is determined by the ratio N/N' (= N/(N + L - 1)), which, in turn, is determined by the ratio L/N. Clearly, when L<< N, N/N' ≈ 1 and thus the difference between the constrained FBLMS algorithm and its unconstrained counterpart becomes insignificant. On the other hand, when L and N are comparable, the difference between the two algorithms will be significant.

2.4.5 Selection Of The Block Length

Block processing of signals, in general, results in a certain time delay at the system output. In many applications this processing delay may be intolerable and hence it has to be minimized. It arises because a block of samples of input signal has to be collected before the processing of the data can begin. Consequently, the processing delay increases with block length. On the other hand, the per sample computational complexity of a block processing system varies with the block length, L. For values of L smaller than the filter length, N, per sample computational complexity of the FBLMS algorithm decreases as L increases. It reaches close to its minimum when L ≈ N. Thus. in applications where the processing delay is not an issue, L is usually chosen close to N. The exact value of L depends on N. For a given N, one should choose L so that N' = N + L - 1 is an appropriate composite number so that efficient FFT and IFFT algorithms can be used in the realization of the FBLMS algorithm. On the other hand. in applications where it is important to keep the processing delay small, one may need to strike a compromise between system complexity and processing delay. In such applications an alternative implementation of the FBLMS algorithm, which is introduced in the next section, is found to be more efficient.

NEAR EAST UNIVERSITY 1988 - LEFKOŞA

CHAPTER 3

OTHER ADAPTIVE FILTER APPLICATIONS

3.1 Introduction

Seismic survey data are normally held on magnetic tape and the information is preprocessed to enhance the required information in a computer program that models the desired adaptive processor before applying image reconstruction algorithms. Here the preprocessor complexity influences the length of the program and hence the speed of computation, but in general, speed is of less importance in off-line processing. However the analysis of multisensor data often requires sophisticated distributed array processors to achieve realistic processing rates for the sophisticated algorithms that are adopted.

One of the key advances over recent years has occurred in real-time adaptive filter applications, where rapid developments in technology are reducing the cost and increasing the sophistication of adaptive filters. Several designs of analog adaptive filters, digital adaptive transversal filters and adaptive lattice filters now exist, which can process signals at sample rates exceeding 100 kHz.

In addition, general-purpose signal processing circuits, such as programmable signal processors and fast, high-accuracy (32-bit) microprocessors also make it possible to realize adaptive filters for the sample rates used in speech processing and data modems for local area networks. These developments are forcing systems designers to consider seriously increasing the use of adaptive processors to improve the performance of next-generation systems.

This chapter first covers other applications of adaptive filters in adaptive estimation, over and above those reported, then reviews their use in spectral estimation before briefly concluding with other application areas such as spatial nulling and bearing estimation in adaptive arrays.

3.2 Adaptive Estimation

Adaptive estimation covers the use of an adaptive processor or filter to measure and identify the key parameters that define a signal or are present in an unknown system.

3.2.1 Inverse System Modeling

The adaptive equalization examples for speech-band data modems and local network digital transmission provide two examples of the use of inverse system modeling adaptive estimation techniques. Similar processing techniques are also being applied to multipath compensation in high-frequency (HF), troposcatter, and digital microwave radio communication systems, in addition to spread-spectrum transmissions in urban digital radio, where severe multi-path arises from reflections off buildings.

When analog radio systems are subject to multipath, the transmission bandwidth is restricted to less than the reciprocal of the multipath delay by the dispersion in the propagation medium. In a frequency modulation system the dispersion also introduces degrading intermodulation products. Hence it was common to apply some signal, code, or diversity reception to alleviate the problem. However, with digital transmission, adaptive processing can be used to measure a multipath, which is slowly fading with respect to the data rate and use it as a form of implicit diversity to improve the overall system performance. Thus the capacity of digital troposcatter and other systems are not restricted to the same extent by the multipath returns. In many systems two- or fourfold path diversity reception techniques are incorporated and the receiver combines the separate processed returns from each channel before making the decision as to the polarity of the received data. Combining can be implemented at RF, IF, or baseband.

Typical values for multipath decorrelation spreading factors in time and frequency for three systems are as follows. In the 2- to 30-MHz long-range (>100-mile) HF radio systems, the fading rate or Doppler spread is of the order of 0.1 Hz and the multipath delay spread is approximately 1 ms. In the 0.4- to 5-GHz troposcatter links these factors become 1 Hz and 100 ns, respectively, while in the shorter-range (<50-mile) microwave

line-of-sight radio systems they are typically 0.01 Hz and 10 ns. The multipath delay spreads of these three systems are related to the length of the propagation path.

One of the earliest processors proposed for overcoming multipath degradations in low-intersymbol-interference systems was the RAKE filter, which bears a very strong resemblance to the FIR adaptive filter. It operates by raking together all the separate multipath components and adjusting their amplitude and phase in weighting multipliers to achieve a coherent summation. An alternative structure is the correlation filter . *Which uses a single-stage hR filter (Figure 3.1) where the previous information bits are* used as a coherent reference to implement a matched filter to each data bit. Maximum-likelihood-sequence estimation techniques have also been proposed for this application, but their complexity is much higher than that of simpler adaptive FIR filters such as the RAKE.

In line-of-sight microwave radio links the fades are slow with respect to the transmitted bit rates, with it taking from a few seconds to almost a minute for a multipath notch (Figure 3.2) to move across the 30- to 75-MHz-wide bandwidth of these equipments. The primary requirement in the new digital microwave radio systems is that they must accommodate the same number of 3.3-kHz bandwidth telephone channels as the earlier analog systems. This is providing the thrust behind the development of bandwidth-efficient modulation techniques such as 16-or 64-state quadrature amplitude modulation (QAM) and other approaches as well as the use of dual (orthogonal) antenna polarization techniques for frequency reuse, which all give rises to increased intersymbol interference.

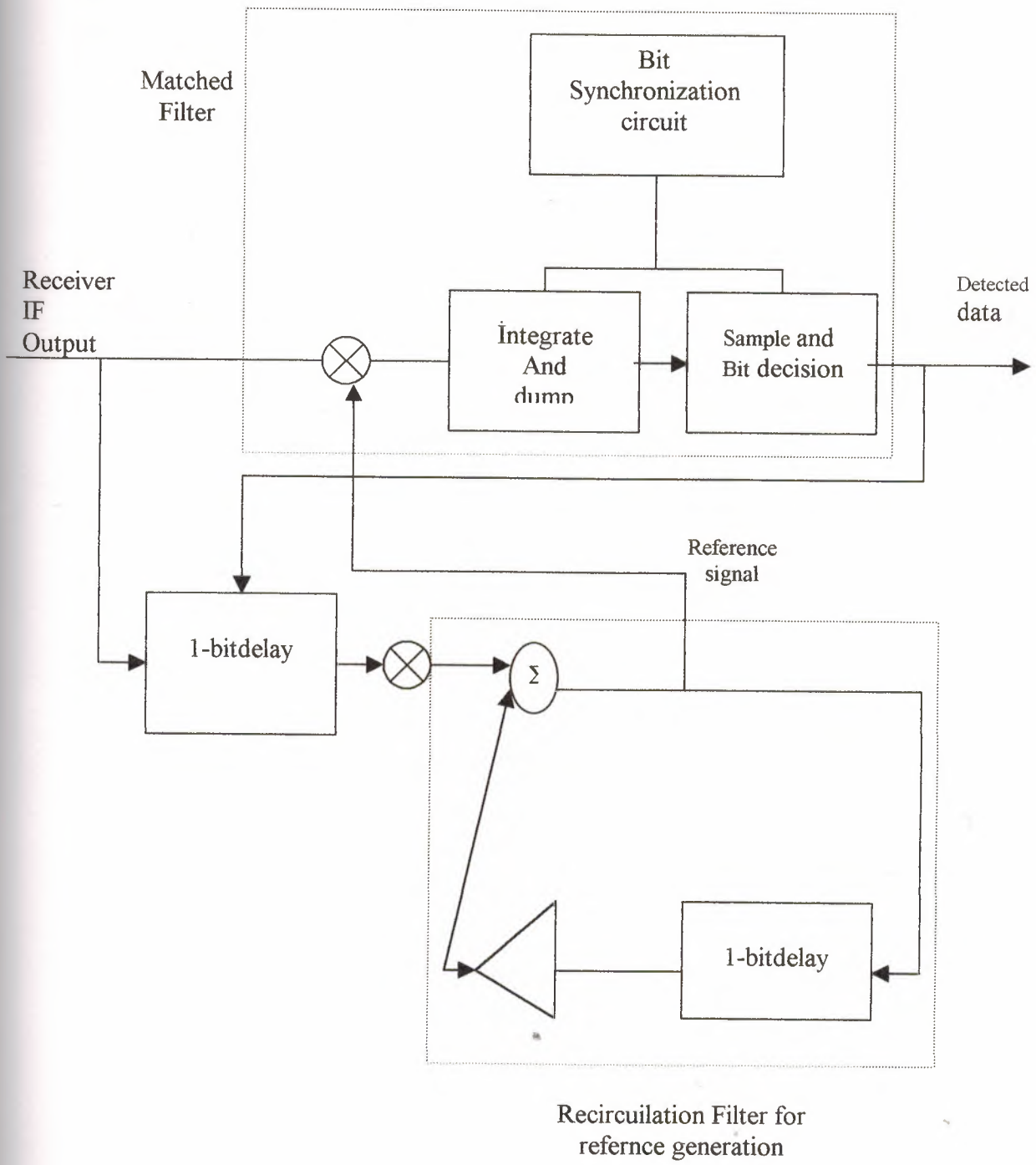


Figure 3.1 Application of correlation techniques to implementing a matched filter receiver phase-shift -keyed signal

When this is combined with the 20- to 30-dB deep fades experienced in these equipments, effective equalization schemes are required before low-bit-error-rate communications can be established. Figure 3.2

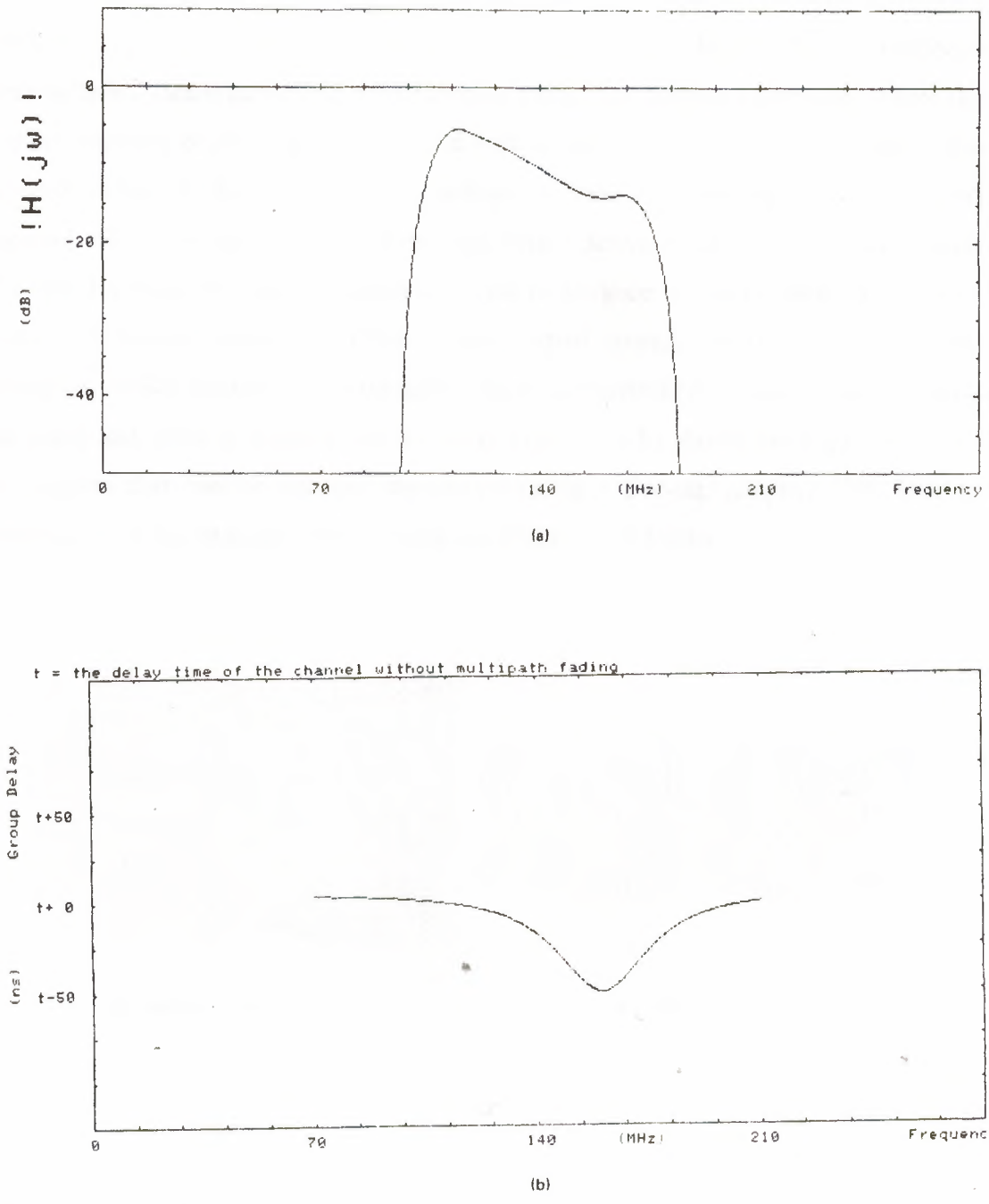


Figure 3.2 Shows the amplitude and group-delay responses for a 70-MHz-wide multipath fade from a secondary ray of relative amplitude 0.8 and delays 2 ns. This introduces a 14-dB deep notch at a frequency 20 MHz above the band center

shows the typical frequency and group delay response for a three-ray multipath fade with a notch at 160 MHz in a system with a 140-MHz intermediate frequency. Figure 3.3(a) shows a simulation of the degraded output eye diagrams for the 0.3 raised cosine channel when subject to the multipath fade shown in Figure 3.2.

Current approaches for handling multipath interference in microwave radio have used diversity reception techniques, but this only compensates for flat rather than frequency selective fades. Equalization has been based on frequency-domain approaches where the spectral response of the receiver is altered with adjustable filters to compensate for the frequency-selective distortion by attempting to level the energy in the received spectrum. This approach can be implemented with a derivative of the simple correlation filter, but the receiver must be manually preset in advance to handle either a minimum or a nonminimum phase fad. There is thus thrust toward the FIR adaptive filter techniques, which provide programmable phase compensation to accommodate both, fade types and achieve superior performance. Figure 3.3(b) shows the improvement in eye diagram that can be obtained by incorporating a five-tap adaptive FIR filter to compensate for the multipath fade causing the distortion of Figure 3.2.

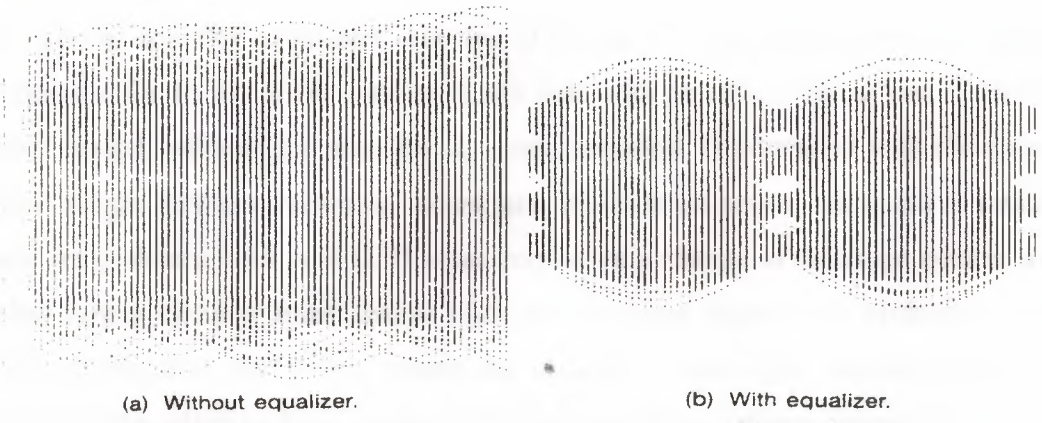


Figure 3.3 Simulations of receiver eye diagrams with and without equalization for the multipath fade of Figure 3.2. Equalization was simulated with a five-tap complex adaptive finite impulse response filter.

Decision feedback adaptive equalizers are also being incorporated in reduced-bandwidth quaternary-phase-shift-keyed 6- and 11-GHz digital system. These equalizers, which operate at the European 140-megabit/s standard rate, have clearly demonstrated that they can compensate for the intersymbol interference, which is introduced by the narrow-bandwidth (60 MHz) transmitter filters. These are incorporated to provide the reduced bandwidth capability of 0.8 times symbol rate and to achieve the consequent high transmission rate of 4 to 5 bits/s per hertz.

The use of a pure decision feedback technique without the linear equalizer of Figure 8.10 greatly simplifies the overall design of the equalizer, as the feedback data comprise a regenerated bit stream which requires only bipolar shift registers, compared to the multilevel registers in the adaptive FIR filters. The tap-weight adjustment can be obtained with a variable current source, which is summed with the incoming data in a common load at the input of the quantizer. Such a design can be realized with 6-to 8-ns settling delay, which permits operation at 74 megabaud, corresponding to the 140-megabit/s transmission rate (Figure 3.4). These equalizers also permit the use of orthogonal polarization in the antenna, which doubles the traffic to 280 megabits/s in the same bandwidth allocation and provides equivalent number of subscribers to the previous analog FDM/FM transmission systems.

The two-tap complex equalizers reported in Figure 3.4 have also shown that they can compensate for 15 dB of two-ray multipath fade with an echo delay of one tenth of the symbol period, without the need for diversity reception. If combined with the normally applied height diversity reception techniques, much more severe multipath is tolerable. There are problems with pure DFB equalizers when a change of fade type occurs, as the symbol timing circuitry must always track the strongest signal. This problem is not so severe in adaptive FIR filters, where the change in fade type requires only a time reversal of the adaptive filter weights. Five to nine-tap equalizers appear to adequately compensate for expected fades.

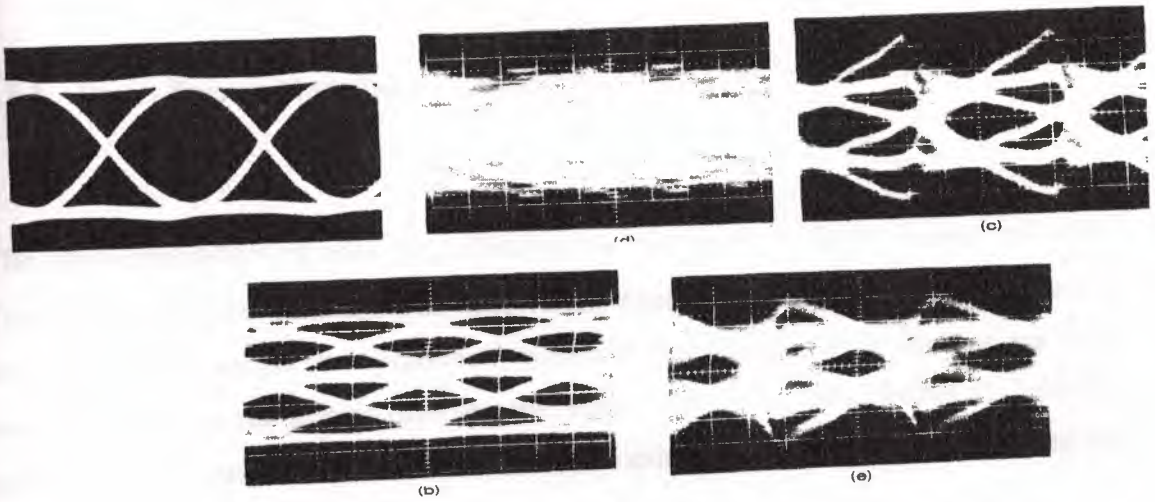


Figure 3.4 Eye diagrams show operation of practical decision feedback equalizer on microwave digital communications link. (a) Without intersymbol interference; (b) intersymbol interference introduced by the reduced bandwidth phase-shift keying; (c) as in (b) but with adaptive equalization in the receiver; (d) system-performance with multipath interference as well as reduced bandwidth transmission, without equalization; (e) as in (d) but with adaptive equalizer.

The problems of multipath cancellation become slightly more difficult in troposcatter links as the data rate reduces to 2 megabits/s due to the increase in fading rate in these longer-transmission-path systems. These problems become even more severe in HF links. First, the lower 3- to 30-MHz carrier frequency reduces the available channel bandwidth to the 3-kHz separation of analog radio systems. Thus although the fade rate reduces in absolute terms compared to a troposcatter link, in this narrowband channel the relative fade rate, with respect to the channel baud rate, is increased by a factor of approximately 100. As a result the received signal can no longer be considered as stationary, and adaptive processor designs, other than the gradient search algorithms, must be applied to provide the faster tracking rates.

Short-duration block processing is normally used to overcome these difficulties, and this provides another thrust behind the development of fast-tracking adaptive filters such as the Kalman and lattice. Alternatively, the adaptive filter weights can be calculated by open-loop matrix inversion techniques, which are computationally demanding but will be realizable with developments in VLSI.

3.2.2 Direct System Modeling

The other application of adaptive estimation techniques is in direct system modeling. Two areas where this technique has been widely applied are in noise cancellation and in echo cancellation across the telephone line hybrid transformer. Other applications have been reported in a number of diverse uses, such as for echo cancellation in Teletext receivers where integrated adaptive filters now exist, in electrocardiography, and in canceling mains interference.

A further application for system modeling or identification techniques occurs in adaptive control systems. These have very close similarities to the adaptive filters described in this text. Adaptive control systems normally comprise an adjustable feedback loop, which is employed to stabilize an overall system response in the presence of external disturbances and changes in the system parameters. Adaptive control systems are usually based on one of several algorithms, which include the model reference approach and nonlinear and impulsive response techniques. Adaptive control is applied extensively in chemical process and power systems modeling, in the control of industrial machines and hydraulic drives, and in the prediction of flight paths of spacecraft and in their altitude control. Widrow and Stearns provide a brief review of the application of adaptive signal processing in control systems.

3.3 Spectral Estimation

3.3.1 Introduction

Spectral analysis or estimation is another potential application area for adaptive filters. Spectral estimation can be divided into two categories: those approaches, which are based on parametric modeling techniques and nonparametric approaches. Fourier analysis, which is widely applied in the characterization of broadband signals, is an example of the nonparametric approach.

In parametric spectral estimation it is assumed that the response to be analyzed comprised wideband noise which has been filtered in a system or passed through a transmission path. An adaptive filter is then used in the spectral estimation processor to model the inverse of the system or transmission response, and after convergence, further processing of the poles or zeros of the modeling filter provides information on the spectral properties of the input signal via the system model. These parametric modeling techniques, which are similar to the Kalman filter modeling techniques covered in Figure 2.3, further subdivide dependent on the different types of filter that are used to perform the spectral estimation. FIR all-zero filters provide inverses for autoregressive (AR) models, while fully recursive (all-pole) filters give inverses for moving-average (MA) models. Autoregressive moving-average (ARMA) models are obtained from pole-zero IIR filters. To minimize the computation and ensure stable convergence properties, it is important to know in advance the type of system model so that the optimum filter model can be employed. However, as a FIR filter can produce a similar response to a recursive filter, provided that a sufficient number of stages are employed, AR models may generally be used throughout, although this may result in less efficient use of hard-ware or computer processing time.

Figure 3.5 shows the effect of applying parametric spectral estimation to a signal comprising five separate sinusoids. The estimation is performed with a set of cascaded FIR prediction error filters. These progressively whiten the input signal and if, after convergence, the filter coefficients are Fourier transformed, they yield a power spectral density analyzer response. Figure 3.5 shows the transformed output from prediction error filters whose order increased progressively from 1 to 10. These simulations show that each second-order stage can control the positions of a zero pair to model the generating pole pair corresponding to one sinusoid. Thus a filter of order 10 is required for this five-sinusoid-input signal. Extending the order beyond 10 does not provide significant improvements. If the filter order is insufficiently large, it groups the sinusoids as pairs and converges to pole values corresponding to the three discrete frequencies that it can identify. As this test signal is a set of noise-free sinusoids, their respective frequencies are accurately identified in Figure 3.5, but accurate amplitude information is not provided.

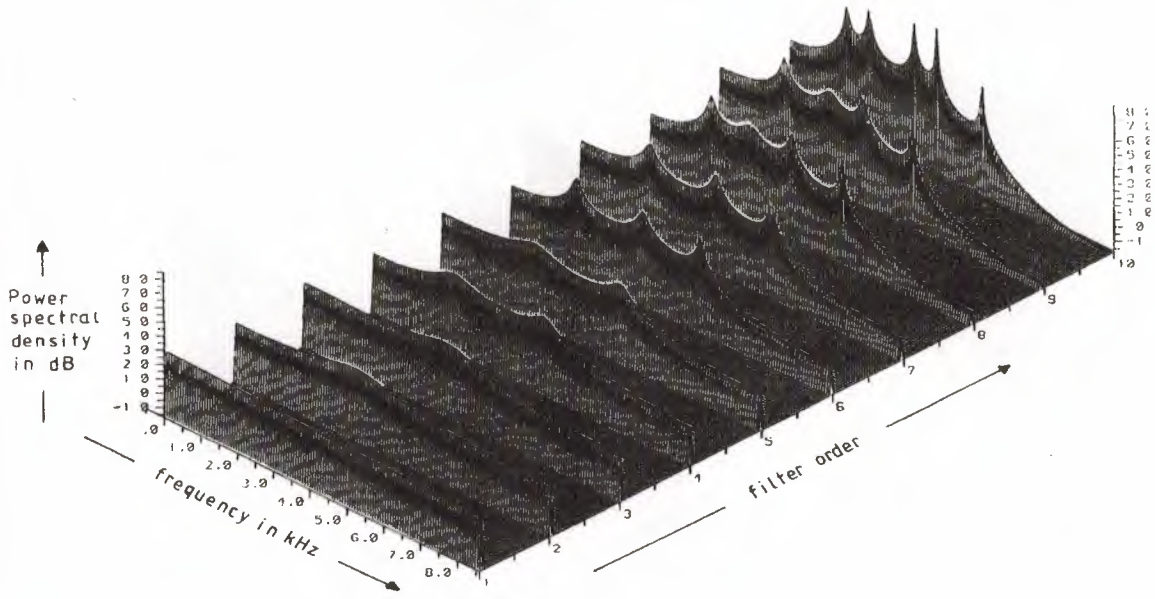


Figure 3.5 Application of prediction error filters to parametric spectral analysis. Figure shows how the transformed filter coefficients yield the power spectral density response for an input signal comprising five separate sinusoids. Simulations show how analysis accuracy improves with increasing filter order or complexity.

The operation of the prediction error filter cascade is illustrated further in Figure 3.6, which shows how the output spectrum is whitened as one moves progressively down the cascade. In this example the filter is input with white noise convolved with the synthetic channel impulse response $0.28z^{1} + z^{-2} + 0.28z^{3}$ to provide the filtered input power spectral density shown at zero order in Figure 3.6.

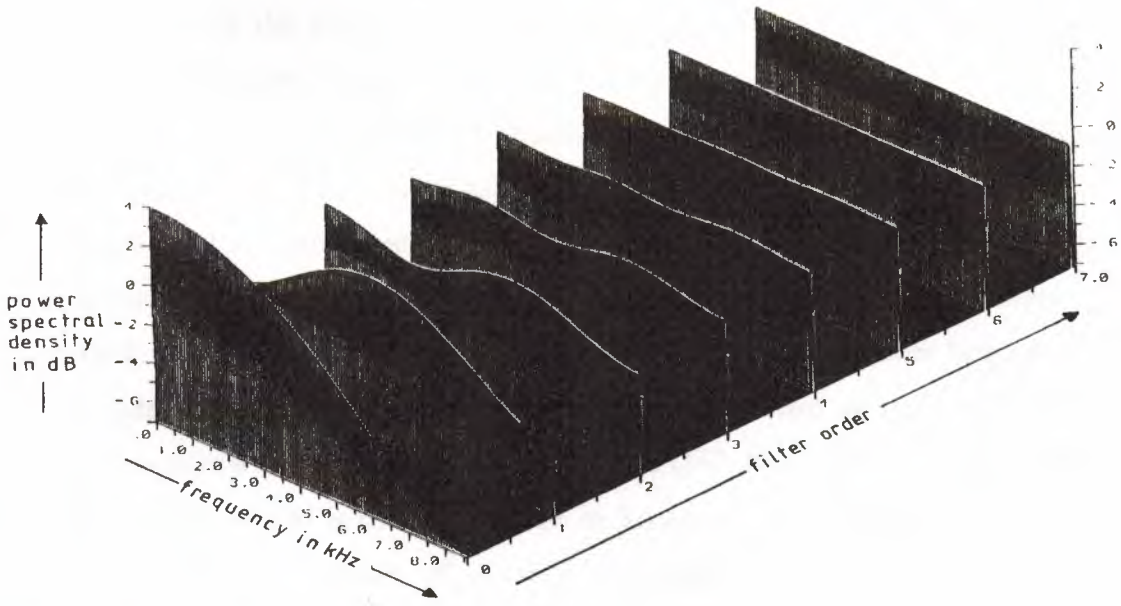


Figure 3.6 Shows how the output spectrum from a prediction error filter cascade becomes progressively whiter as the filter order increases. Input test signal is white noise convolved with the impulse response $0.28z^{-1} + z^{-2} + 0.28z^{3}$.

The prediction error filter now fits appropriate zeros to model the input. This results in a progressive whitening of the spectrum as it emerges from each filter stage. Compared to Fourier analysis techniques, the adaptive parametric spectral estimation approach optimally identifies the locations of the input sinusoids, but it provides no detail in the regions in between. For a low signal-to-noise ratio at the input, the noise components in between the sinusoidal tones drive the filter from convergence, making this approach not as useful as Fourier techniques for determining the input spectral response. Thus it is generally accepted that nonparametric Fourier analysis techniques are superior for broadband analysis with low input signal-to-noise ratios (SNR). However, the resolution of this technique, which is proportional to the length of the observation window, is inferior. For time varying spectra or short time series, such as pulsed radar returns, the observation window has to be restricted, reducing the resolution of the Fourier approach. In addition, leakage in the discrete Fourier transform (DFT) results in smearing of the spectral components. These deficiencies have spurred interest in the newer parametric modeling techniques, which obtain increased resolution by

extrapolating values for the autocorrelation beyond known lags. In the DFT processor these are assumed to be zero, introducing the spectral leakage.

The maximum-entropy method (MEM) of spectrum analysis, which is based on a FIR autoregressive model, has been shown to be a more general approach rather than a subset of AR spectral estimation techniques. The maximum-likelihood method (MLM), which is another AR technique, measures the power out of a set of narrowband filters. Unlike the DFT these filters can each have a different band shape and center frequency and they are adaptively set onto the frequencies of the signals that are present at the input. It thus produces a resolution that is superior to the DFT, but it is not quite as good as other AR methods. MLM is used extensively in frequency wave-number analysis in seismic arrays. The key feature of these linear algorithm-based autoregressive techniques is that they perform well when only a few sinusoids are present, as they provide a data reduction capability, which is put to great use in applications such as speech analysis and synthesis as well as seismic processing.

3.3.2 Spectral Line Enhancement

One application of spectral estimation is in the identification of the presence of signals using enhancement techniques. An example of this is the recovery of a narrowband signal from wideband noise by adaptive line enhancement (ALE). The ALE is implemented by connecting the received signal to a delay module before it enters the signal input of the adaptive filter, while the desired or training input is connected directly to the received signal. The delay is selected such that the narrowband signals, which we wish to enhance, are correlated between the signal and desired inputs while the wideband noise components are not correlated.

For ALE applications the adaptive filter can be either a FIR or an IIR design, depending on whether an autoregressive or ARMA processor is preferred. The ALE is a simplification of the MEM spectral analysis discussed previously, as MEM typically uses sophisticated processing algorithms or fast Kalman techniques, which are computationally demanding. However, they rapidly provide values for the prediction error filter weights. In the simple ALE approach the prediction error filter is a simplification of this where the prediction error filter weights are derived from slower

"adaptive" algorithms, such as the stochastic gradient search technique, which provide computational efficiency at the expense of a slower response.

Figure 3.7 shows simulated input and output waveforms for ALE, which is based on the frequency-domain adaptive processing techniques.

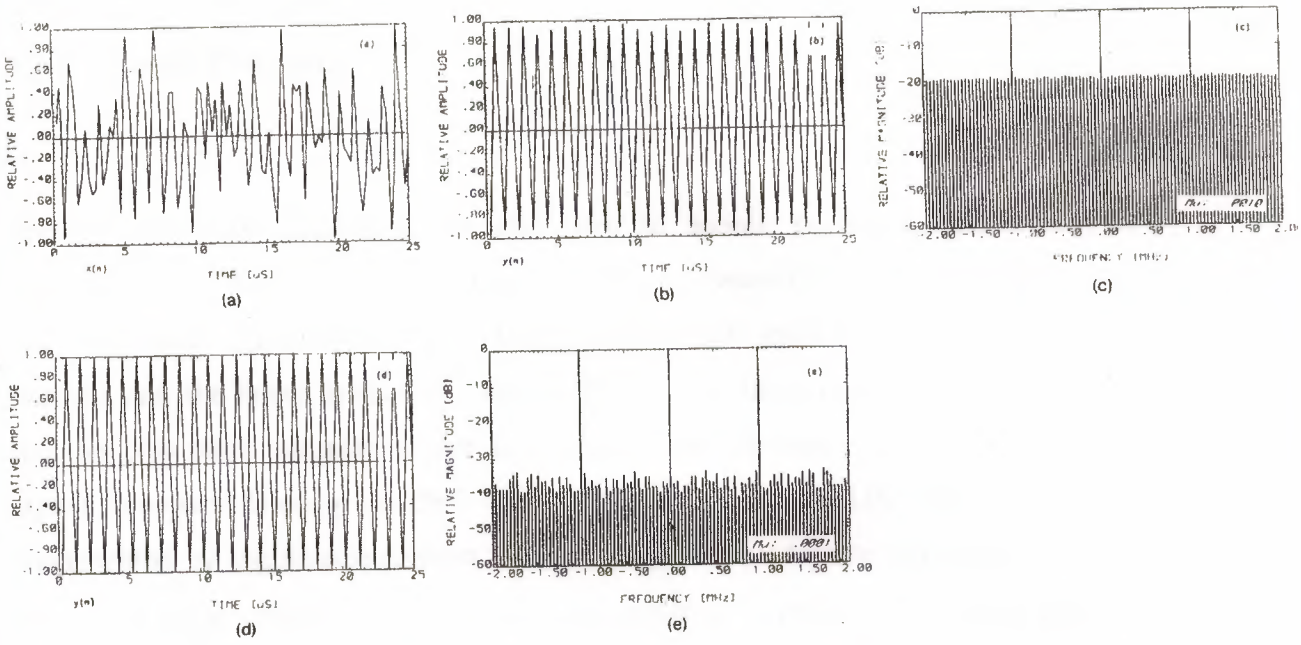


Figure: 3.7 operation of adaptive filter as a spectral line enhancer with an unknown sinusoid and wideband noise at filter input. (a) input comprising 1-MHz sinusoid and wideband noise at 0 dB SNR; (b) output signal; (c) corresponding spectrum with an adaptive filter convergence coefficient of 0.001; (d) output signal; (e) corresponding output spectrum with convergence coefficient of 0.0001

This uses an input signal comprising a 1-MHz sinusoid plus wideband noise at 0 dB SNR [Figure 3.7(a)]. The outputs [Figure 3.7(b) and (d)] clearly illustrate how the sinusoid is enhanced and show that the level of noise suppression [Figure 3.7(c) and (e)] is also dependent on the selected convergence coefficient μ. Examples of the application of IIR line enhancers based on optimal least-squares estimation techniques have also been reported.

For signals contaminated by white noise the FIR-based ALE performance is broadly equivalent in terms of resolution and SNR improvement to a DFT processor if the number of transform points equals the order of the FIR filter. However, as the ALE

processes signals continuously whereas the DFT is a block processor, the DFT output must be averaged over several frames to integrate and obtain the same SNR improvement. The performance of the ALE is superior to the DFT approach when the input comprises either colored noise or a mix of strong and weak sinusoids. Under these conditions the weak signals are enhanced and the colored noise is suppressed by the ALE. A further possible advantage is the reduced computational load of the ALE.

3.3.3 Speech Processing

Spectral estimation techniques are also used in speech processing, particularly in vocoders, which exploit the redundancy in the speech waveform to achieve low-bit-rate (<2.4 kilobaud) transmission rates. Two major designs exist at present, the channel vocoder and the linear predictive coder (LPC). The channel vocoder transmits coarse spectral plus pitch information that is normally obtained with conventional analog or digital bandpass filtering or DFT techniques. Although the LPC design has been implemented by the autocorrelation method, which uses adaptive transversal filters, a cascade of linear prediction error filters is the preferred approach, as it is less sensitive to coefficient inaccuracies. The prediction error filters remove from the signal the components that can be predicted from the previous history by modeling the vocal tract as an all-pole (AR) filter.

In the LPC vocoder the analyzer and encoder normally process the signal in 30-ms frames and subsequently transmit the coarse spectral information via the filter coefficients. The residual error (noise output from the prediction error whitening filter) is not transmitted; instead, it is used to provide an estimate of the input power level which is sent along with the pitch information, and an indication as to whether the input is voiced or unvoiced (Figure 3.8). The latter can be ascertained by examining whether the first lag of the autocorrelation function of the signal lies above or below a certain threshold. If above the threshold, the value of the autocorrelation term provides the pitch information. The decoder and synthesizer apply the received filter coefficients to an AR synthesizing filter, which is excited with impulses at the pitch frequency if voiced, or white noise if unvoiced. The excitation amplitude is controlled by the input power estimate information.

The adaptive lattice filter can be effectively applied to implement

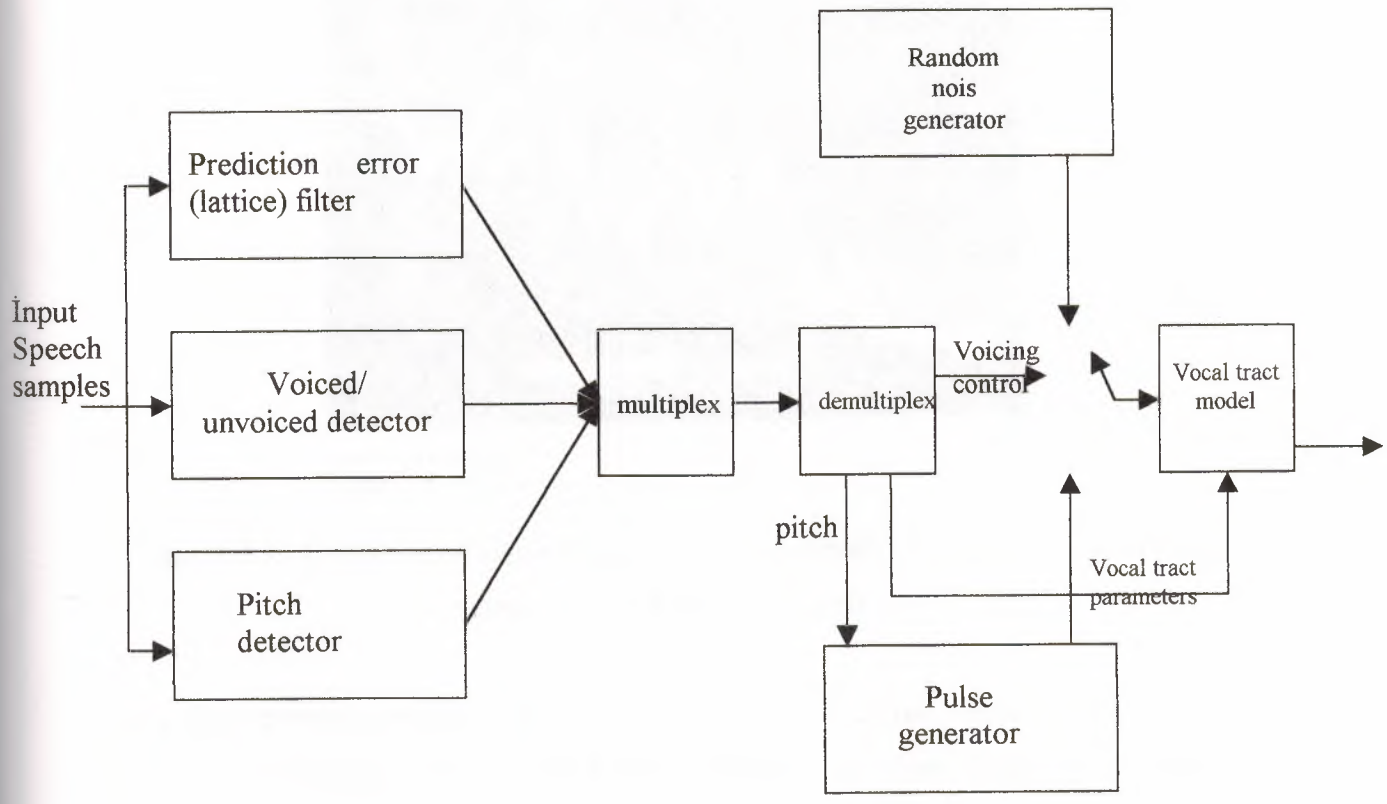
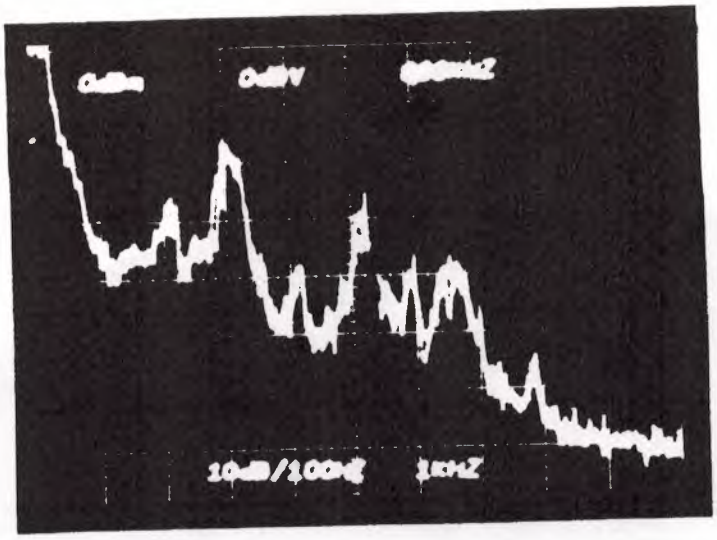


Figure 3.8 Linear predictive coder for the vocoding of speech signal.

an all-pole filter representation of the physical lossless acoustic tube model of the vocal tract . This approach, which outputs a continuous representation of the filter prediction error coefficients, has received most emphasis to date because it is a regular structure that is amenable to implementation with digital LSI circuits.

Further to the results shown in Figures 3.9 and 3.10 compare the performance of a conventional swept-frequency (nonparametric) spectrum analyzer with the autoregressive (parametric) estimator on a sample of male human speech. The vowel "ee" as in the word "feed" was sung into a tape recorder to maintain as far as possible a constant fundamental frequency. This was then played back into a commercially available spectrum analyzer. The parametric estimator simulation used to obtain the results of Figure 3.5 was applied separately for comparison purposes. The outputs are shown in Figures 3.9 and 3.10, respectively, where the individual parametric estimates, for up to 16 orders, have been interpolated to provide a fully filled display.



vert: 10 dB/div
horiz: 1 kHz/div

Figure 3.9 Conventional swept-frequency spectrum analyzer display of male voice saying "ee" as in the word "feed."

On a visual comparison one is impressed by the similarity of the results between the 12- to 16-order parametric estimator and the conventional analyzer display. Both show a similar overall shape exhibiting the 6-dB/octave reduction with frequency which is caused by the vocal tract response. However, the parametric estimator does not possess the fine detail of the swept-frequency response, due to the much shorter analysis time. Further investigation shows close agreement between approaches when estimating the overall spectral density, but

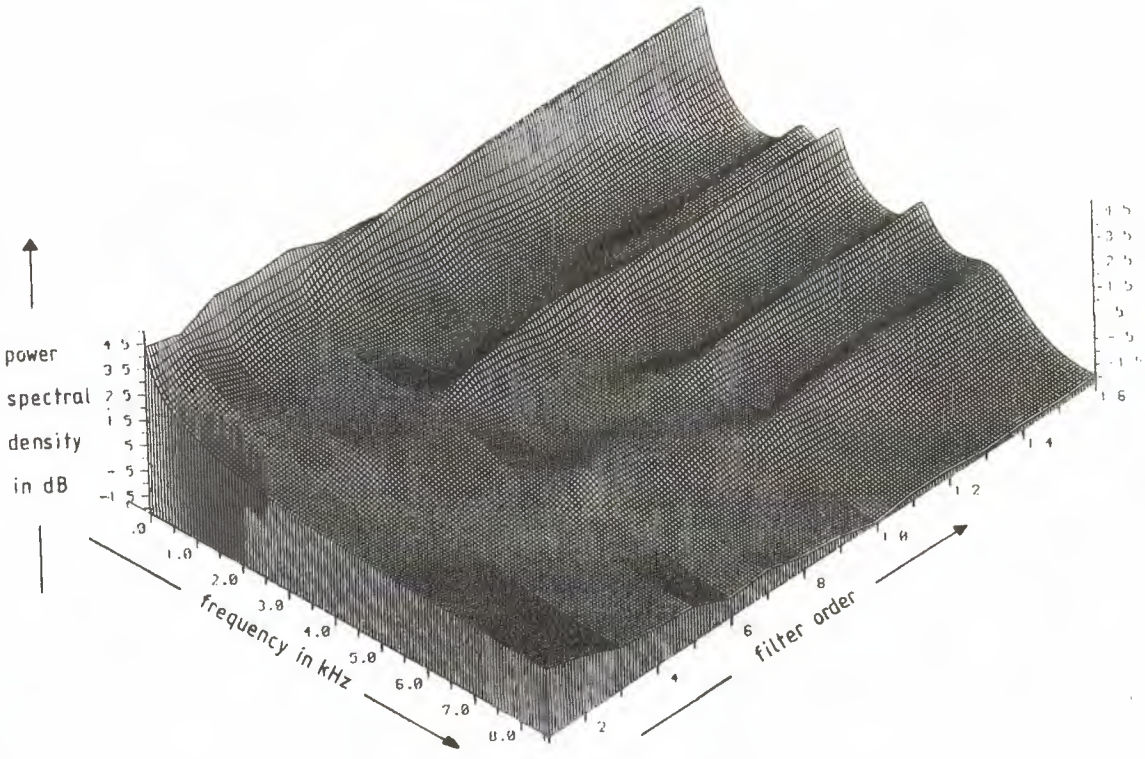


Figure 3.10 Autoregressive spectral estimate for filter order up to 16 for the same input signal as used in figure 3.9

the parametric analyzer is less accurate in estimating the absolute frequencies of the peaks, and the minor peak at 4.2 kHz has not been detected. Accurately placed zeros close to the unit circle modeled the test signal of Figure 3.5, which had a well-defined spectrum with sharp peaks. In contrast, the speech waveform used here had a less sharply defined spectrum, which resulted in less accurately, placed zeros. Figures 3.9 and 3.10 give a broad comparison of the two approaches and show how the adaptive zero fitting in the simpler autoregressive estimator provides a sufficiently accurate spectral representation for synthetic and possibly also communications-quality speech transmission. Toll-quality transmission requires more accurate sampling techniques, such as pulse-code modulation.

Lattice adaptive prediction error filters offer a performance in terms of complexity and speed of operation, which lies in between the sophisticated MEM and relatively simple ALE spectral estimation techniques. Key attractions of the lattice approach are again based on independent optimization of successive components plus the fact that there is a trade-off between convergence factor and residual error with filter length. With delays in the vocal tract of about 1 ms and typical speech sample rates of 8 to 10 kHz the

number of lattice stages is normally in the range 8 to 12, with 10 being the number adopted in the integrated LPC vocoder standard, which transmits at a 2.4-kilobaud rate. Multichip microprocessor-based vocoders are also available. In addition to these vocoder applications, digital lattice filters are used in several commercial speech synthesis systems.

3.4 Adaptive Array Processing

Adaptive antennas, which are applied to receiver designs to maximize a desired signal in the presence of interference, use processing techniques that are very similar to those of adaptive filters. They use the spatial separation between the antenna elements to provide a parallel set of signal samples rather than using the time-delayed or partly processed versions of a one-dimensional input signal. Early work on adaptive antennas was concerned primarily with self-phasing systems, which reradiated energy in the same direction as a received signal. These systems are less widely applied at present and the term "adaptive antenna" now refers almost exclusively to spatial nulling techniques, which reduce the effect of unintentional cochannel interference or deliberate jamming.

These processors, which can automatically respond to an unknown interference environment in real time, have thus found widespread use in military radar, sonar, navigation, and communication equipment, where a high speed of adaption for fast nulling is an important property. Civilian applications such as VHF/UHF and satellite communications and broadcast systems are less demanding, as the convergence rate is usually of secondary consideration and hence less sophisticated adaptive algorithms can be employed. The interference bearing is normally fixed and the achievement of satisfactory cancellation at low cost is more important. Bearing estimation is another function that can be performed with antennas, either by directly analyzing the received signals or by transforming the weight values from a converged adaptive array to determine the bearing of the source.

Adaptive antenna nulling uses a very similar processor configuration to the adaptive filter. In its simplest form it has two inputs from the main and auxiliary antennas. If these are microwave signals, they are 'usually down-converted to IF and the auxiliary channel is multiplied in a complex weighting network before summation with main

channel (Figure 3.11). For interference cancellation the combined output is fed back and cross-correlated with the signal in the auxiliary channel to derive the adaptive weights to minimize the signal, which is present in both channels.

This approach is commonly used with high-gain 5- and X-band microwave reflector-based antennas to cancel out jamming which enters through the main antenna sidelobes. In this coherent sidelobe canceller (CSLC) (Figure 3.8) the difference in gain between the main beam of the directional antenna and the wider coverage or omnidirectional auxiliary antenna result in desired signals (such as low-level radar return echos) being received only in the main channel. In pulsed radar, where these echo returns are present only for short periods in the overall scan interval, their average energy is low and hence they do not need to be subtracted out, as they introduce only minor degradations in the processor. Jamming through the main antenna sidelobes enters both channels at approximately the same signal level and is automatically nulled out in this self-steering processor. Additional auxiliary channels must be added to the scheme of Figure 3.11 to handle multiple jammer scenarios. Lubell and Rebhun provide an example of the application of CSLC techniques in a satellite communications receiver.

Alternative adaptive antennas, based on fully phased arrays with an adaptive control loop on each of the individual elements, are employed in communications, radar, and navigation applications, but these normally require a separate sample of the desired signal to be subtracted from the combined output before it is fed back as the error signal. In theory such an N-element adaptive array can handle up to N - 1 interfering sources.

3.4.1 Bearing Estimation

Antenna arrays, often with only two elements, can be used for bearing discrimination. Bearing information is translated in the antenna into a difference in timing of the received signals due to the path differences to the different elements. Thus processing of the received signals to extract the time difference of arrival (TDOA) provides information relating to the target bearing. This technique has been used for flow measurement of dangerous liquids in pipes by sensing the disturbances due to flow with

a pair of transducers attached to the pipes. Cross-correlation of the two signals [Beck] yields the delay information, but this requires

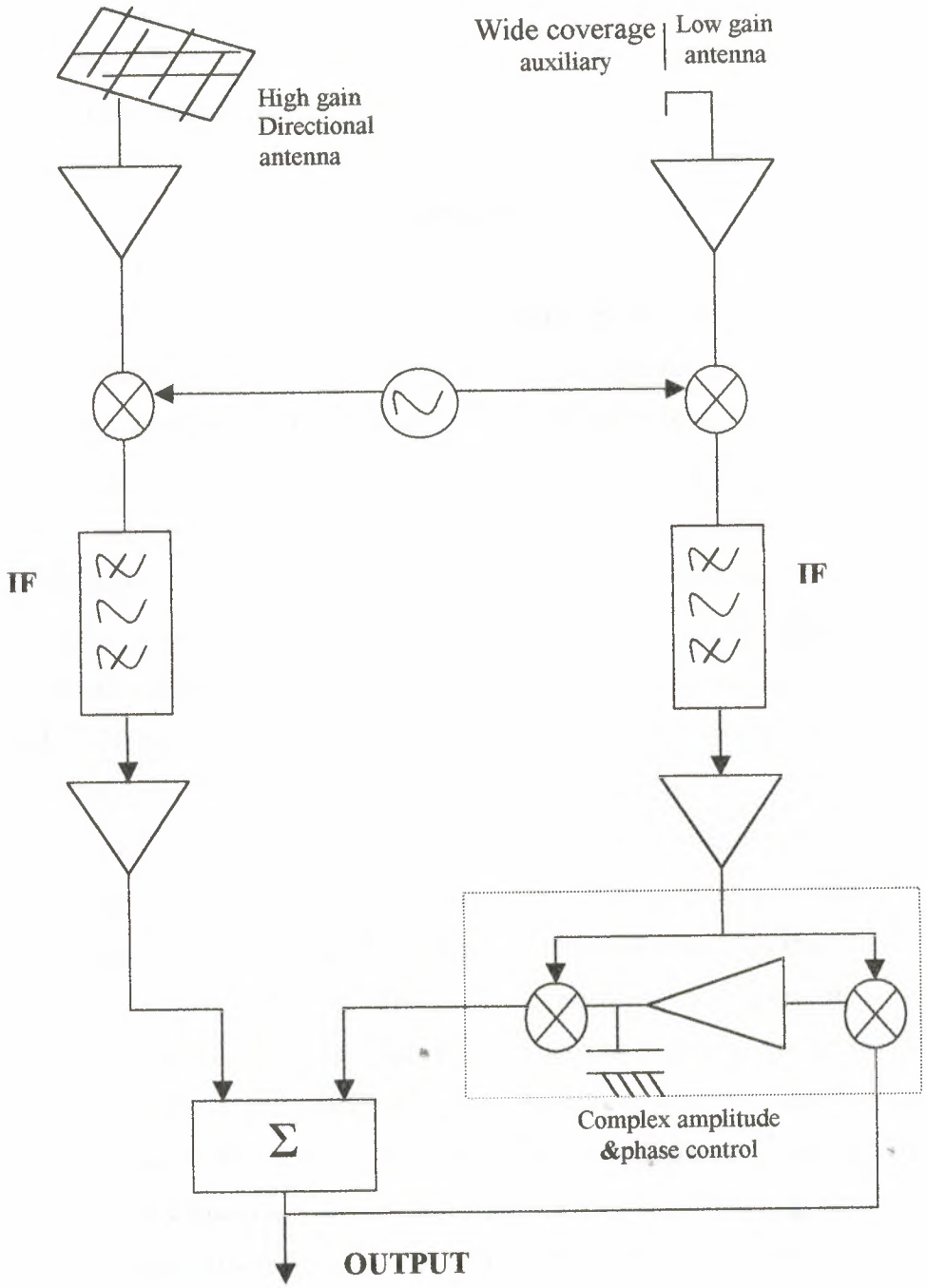


Figure 3.11 Coherent sidelobe canceler configuration of adaptive antenna

Some knowledge of the statistics of the signals and the transducer separation must be such that there is not a significant loss in correlation between the two received signals.

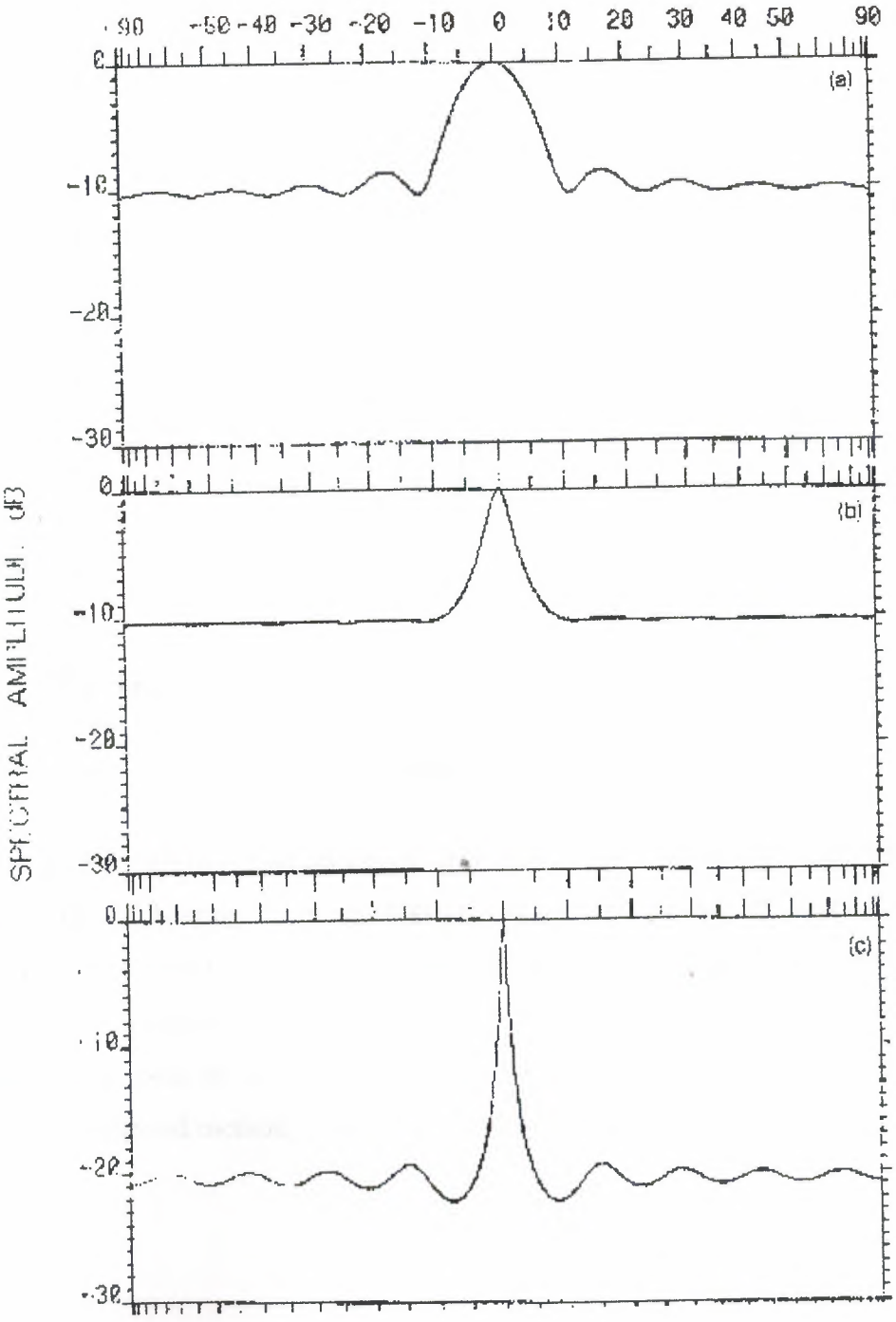
Other applications of this correlation-based transit-time measurement systems are in gas chromatography, biomedical engineering and in sonar.

Adaptive time-difference-of-arrival estimation techniques, which can be applied to enhance the output of these passive-sensing systems, have the advantage that no prior signal information is required. One approach is based on an extension of the earlier correlation method where adaptive spectral whitening techniques are added prior to the cross-correlation processor. Such preprocessing of narrowband input signals prior to cross-correlation of the residuals reduces the incidence of multiple peaks at the output. Most of the adaptive spectral whitening techniques reported in this text is applicable to this processor.

These concepts have also been extended to multielement arrays for accurate bearing estimation of received signals. Simple spatial Fourier analysis of the radiation field by sampling the received signals from an array gives a bearing resolution capability, which is inversely proportional to the aperture of the array. Johnston and De Graaf have shown that superior resolution can be obtained (Figure 3.12) by applying the spectral estimation techniques reported earlier to find a solution to this constrained optimization problem. Adaptive spectral estimation techniques employing MLM, MEM, AR, and ARMA parameter modeling as well as the eigenvector decomposition approach have all been applied to the passive sonar problem. The outputs from the filter model provide the time-delay information, which is converted into bearing estimates.

The inclusion of an adaptive processor into the array permits the beam-former to utilize the null rather than the mainlobe information for bearing measurement. As the nulls are generally much sharper, this provides a consequent resolution improvement or super-resolution capability from the deployment of adaptive techniques. Widrow and Stearns provide an analysis of the improvements gained from the use of maximum-likelihood adaptive processors shows how the resolution of a nonadaptive Fourier beamformer in (a) and (d) compares with the maximum-likelihood processor (b) and (e), for both a single source on boresight and a pair of sources at ±50 angular separation from boresight. For this 10-element array at a 10-dB signal-to-noise ratio, the resolution improvement due to MLM processing is calculated as 18 times. Figure 3.12 also shows that if a linear predictive autoregressive adaptive processor is deployed, there is further resolution improvement, as shown in (c) and (1).The resolution improvement from these adaptive spectral estimation techniques is dependent on a satisfactory received signal-to-noise ratio, and it typically requires positive values to achieve the required

improvement. The future extension of this work to other application areas, such as radar and communications, can now be anticipated to produce further potential applications for adaptive spectral estimation techniques.



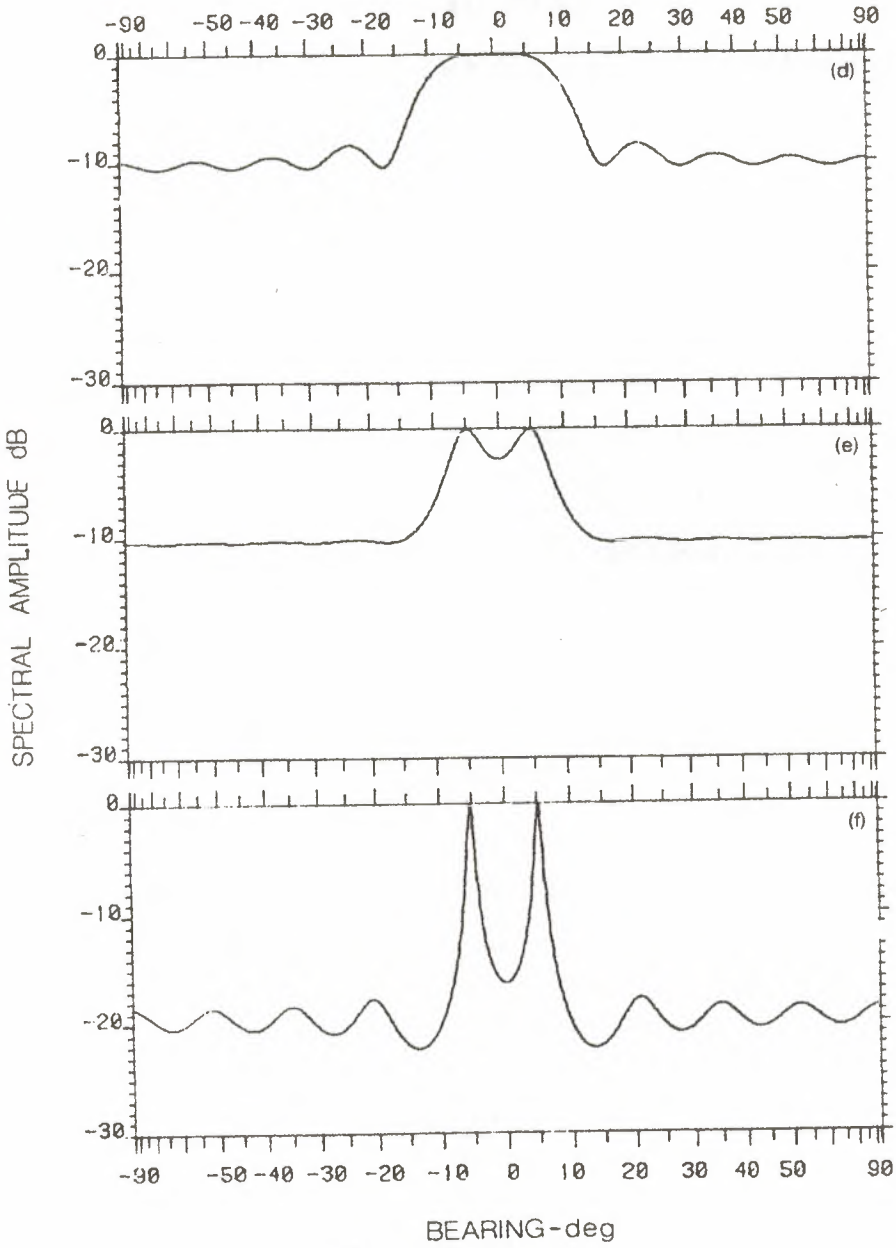


Figure 3.12 Shows application of various spectral estimation algorithms to the measurement of bearing from (a) through (c) a single source on boresight, and (d) through (I) two equal-strength sources at ±5° relative to boresight. Measurements are made with an array of 10 sensors equally spaced at λ/ 2apart with a received sensor signal-to-noise ratio of 10 dB. (a) and (d) Fourier transform techniques; (b) and (e) maximum-likelihood method; (c) and (1) linear predictive estimates.

CHAPTER 4

OTHER ADAPTIVE FILTER ALGORITHMS

4.1 Covariance Algorithms

The essential link in the derivation of the fast algorithms is provided by the (N + 1) x (N + 1) matrix $R_{N+1}(n + 1)$, which relates the adaptation gains G (n + 1) and G (n) at two consecutive instants. Here, a slightly different definition of that matrix has to be taken, Because the first (N + 1)-element data vector which is available is X_1 (2): $[X_1(2)]^t$ = [x (2), X^t (1)]

Thus

$$R_{N+1}(n)=\sum W^{n-p} X_1(p) X^t_1(p) \tag{4.1}$$

The LS procedure for the prediction filters, because of the definitions, can only start at time n = 2, and the correlation vectors are

$$r^a_N(n)=\sum_{p=2}^{n} w^{n-p} X_1(p) X^t_1(p)$$

$$r^a_N(n)=\sum_{p=2}^{n} w^{n-p} X(p-N) X(p) \tag{4.2}$$

The matrix $R_{N+1}(n + 1)$ can be partitioned in two ways:

$$R_{N+1}(n+1)=\begin{bmatrix} \sum_{p+2}^{n+1} W^{n+1-px^2}(p)\,[r^a_N(n+1)]^t \\ r^a_N(n+1) \quad R_N(n) \end{bmatrix}$$

$$R_{N+1}(n+1)=\begin{bmatrix} R_N(n+1) - w^n x^t(1) \quad r^b_N(n+1) \\ [r^b_N(n+1)]^t \quad \sum_{p+2}^{n+1} W^{n+1-px^2}(p-N) \end{bmatrix} \tag{4.3}$$

Several modifications have to be made because of the initial term $W\ ^{n}X\ (1)\ Xt\ (1)$ in (4.4).

The (N + 1)-element adaptation gain vector $G_1\ (n + 1)$ can be calculated by , which yields M (n + 1) and m (n + 1). Equation (4.4) leads to

$$[R_N\ (n + 1) - W\ ^{n}X\ (1)\ Xt\ (1)]\ M\ (n + 1) + m\ (n + 1)\ r^{b}\ (n + 1) = X\ (n + 1) \tag{4.5}$$

Similarly the backward prediction matrix equation combined with partitioning (4.4) leads to

$$[R_N\ (n + 1) - W\ ^{n}X\ (1)\ Xt\ (1)]\ B\ (n + 1) = r^{b}\ (n + 1) \tag{4.6}$$

Now the definition of G (n + 1) yields

$$G\ (n + 1) = R_N^{-1}\ (n + 1)\ X\ (n + 1) = [I_N - W\ R_N^{-1}\ (n + 1)\ X\ (1)\ Xt\ (1)] \times [M\ (n + 1) + m\ (n + 1)\ B\ (n + 1)] \tag{4.7}$$

Let us consider the vector

$$D\ (n + 1) = W\ R_N^{-1}\ (n + 1)\ X\ (1) \tag{4.8}$$

A recursion is readily obtained by

$$R_N\ (n + 1)\ D\ (n + 1) = W\ ^{n}X \tag{4.9}$$

Which at time n corresponds to $R_N\ (n)\ D\ (n) = W\ ^{n}X\ (1)$.Taking into account relationship between $R_N\ (n)$ and $R_N\ (n + 1)$, one gets

$$D\ (n + 1) = [I_N - R_N^{-1}\ (n + 1)\ X\ (n + 1)\ X^{t}\ (n + 1)]\ D\ (n) \tag{4.10}$$

which with (4.7) and some algebraic manipulations yields

$$D(n+1)=\frac{1}{1-x^{t}\ (1)F(n+1)\,x^{t}(n+1)D(n)}\ \left[I_{x}-F(n+1)x^{t}(n+1)\right]D(N) \qquad (4.11)$$

Where

$$F\ (n) = M\ (n) + m\ (n)\ B\ (n) \qquad (4.12)$$

The adaptation gain is obtained by rewriting (4.7) as

$$G\ (n + 1) = [I_N — D\ (n + 1)\ Xt\ (1)]\ F\ (n + 1) \qquad (4.13)$$

Finally, the covariance version of the fast algorithm is obtained by incorporating equations (4.11) and (4.13) in the sequence of operations. The additional cost in computational complexity amounts to 4N multiplications and one division.

Some care has to be exercised in the initialization. If the prediction coefficients are zero, A (1) = B (1) = 0, since the initial data vector is nonzero, an initially constrained LS procedure has to be used, which, as mentioned corresponds to the following cost function for the filter.

$$J_{e}(n)=\sum_{p+1}^{n} w^{n-p}\left[y(p)-X^{t}(p)H(n)\right]^{2}+E_{0}H^{t}(n)W(n)H(n) \qquad (4.14)$$

Where

$$W\ (n)=\text{diag}\ (W^{n},\ W^{n-1}\ \ldots\ldots\ W^{n+1N}) \qquad (4.15)$$

And E_0 is the initial prediction error energy.

In these conditions, the actual AC matrix estimate is

$$R_{N}(n)=\sum_{p+1}^{n} w^{n-p}\left[X(p)X^{t}(p)\right]+E_{0}W(n) \qquad (4.16)$$

The value $R^{-1}_N(1)$ is needed because

$$D(1) = R^{-1}_N(1)\, X(1) = G(1) \tag{4.17}$$

It can be calculated with the help of the matrix inversion lemma. Finally,

$$D(1) = G(1) = \frac{1}{E_0 + X^t(1)W^{-1}(1)X(1)} W^{-1}(1)X(1) \tag{4.18}$$

And for the prediction error energy $E_0(1) = WE_0$.

The weighting factor W introduces an exponential time observation window on the signal. Instead, it can be advantageous in some applications- for example, when the signal statistics can change abruptly-to use a constant time-limited window. The FLS algorithms can cope with that situation.

4.2 Sliding Window Algorithm

The sliding window algorithms are characterized by the fact that the cost function $J_{sw}(n)$ to be minimized bears on the N_0 most recent output error samples:

$$J_{SW}(n) = \sum_{p=n+1-N_0}^{n} \left[y(p) - X^t(p)H(n) \right]^2 \tag{4.19}$$

Where N_0 is a fixed number representing the length of the observation time window, which slides on the time axis. In general, no weighting factor is used in that case, W = 1. Clearly, the AC matrix and cross-correlation vector estimations are again the matrix $RN+_1(h + 1)$ can be partitioned as

$$R_N(n) = \sum_{p=n+2-N_0}^{n+1} \begin{bmatrix} x(p) \\ X(p-1) \end{bmatrix} \left[x(p), X^t(p-1) \right]$$

$$= \begin{bmatrix} \sum_{p=n+2-N_0}^{n+1} x^2(n+1) & \left[r_N^a(n+1) \right] \\ r_N^a(n+1) & R_N(n) \end{bmatrix} \quad (4.20)$$

and

$$R_{N+1}(n+1) = \sum_{p=n+2-N_0}^{n+1} \begin{bmatrix} x(p) \\ X(p-N) \end{bmatrix} \left[X^t(p), X^t(p-N) \right]$$

$$= \begin{bmatrix} R_N(n+1) & r_N^a(n+1) \\ \left[r_N^a(n+1) \right] & \sum_{p=n+2-N_0}^{n+1} x^2(n+1) \end{bmatrix} \quad (4.21)$$

However, the recurrence relations become more complicated. For the AC matrix estimate, one has

$$R_N(n+1) = R_N(n) + X(n+1)\,Xt(n+1) - X(n+1-N_0)\,Xt(n+1-N_0) \quad (4.22)$$

For the cross-correlation vector,

$$r_{xy}(n+1) = r_{xy}(n) + y(n+1)X(n+1) - y(n+1-N_0)\,X(n+1-N_0) \quad (4.23)$$

The coefficient updating equation is obtained, as before, from

$$R_N(n+1)\,H(n+1) = r_{xy}(n+1) \quad (4.24)$$

By substituting (4.23) and then, replacing RN(n) by its equivalent given by (4.22):

$$H(n+1) = H(n) + R_N^{-1}(n+1)\,X(n+1)[y(n+1) - X^t(n+1)\,H(n)]$$

$$- R_N^{-1}(n+1)\, X(n+1-N_0)$$
$$x\,[y(n+1-N_0) - X^t(n+1-N_0)\, H(n)] \qquad (4.25)$$

Backward variables are showing up: the backward innovation error is

$$e_0(n+1) = v(n+1-N_0) - X^t(n+1-N_0)H(n) \qquad (4.26)$$

And the backward adaptation gain is

$$G_0(n+1) = R_N^{-1}(n+1)\, X(n+1-N_0) \qquad (4.27)$$

In concise form, equation (4.25) is rewritten as

$$H(n+1) = H(n) + G(n+1)\, e(n+1) — G_0(n+1)\, e_0(n+1) \qquad (4.28)$$

These variables have to be computed and updated in the sliding window algorithms. Partitioning (4.20) yields

$$R_{N+1}(n+1)\begin{bmatrix} 0 \\ G_0(n) \end{bmatrix} = \begin{bmatrix} x(n+1-N_0) \\ X(n-N_0) \end{bmatrix} - \begin{bmatrix} \varepsilon_0(n+1) \\ 0 \end{bmatrix} \qquad (4.29)$$

with

$$\varepsilon_{0a}(n+1) = x(n+1-N_0) - A^t(n+1)\, X(n-N_0) \qquad (4.30)$$

where the forward prediction coefficient vector is

$$A(n+1) = R_N^{-1}(n) \sum_{p=n+2-N_0}^{n+1} X(p)X(p-1) \qquad (4.31)$$

Similarly, the second partitioning (4.22) yields

$$R_{N+1}(n+1)\begin{bmatrix} G_0(n+1) \\ 0 \end{bmatrix} = X_1(n+1-N_0) - \begin{bmatrix} 0 \\ \varepsilon_{0b}(n+1) \end{bmatrix} \tag{4.32}$$

With

$$\varepsilon_{Ob}\,(n+1) = x\,(n+1-N_0-N) - B^t\,(n+1)\,X\,(n+1-N_0) \tag{4.33}$$

and

$$B\,(n+1) = R_N^{-1}\,(n+1)\,r^b\,(n+1) \tag{4.34}$$

Now, combining the above equations with matrix prediction equations, leads to

$$G_{01}(n+1) = \begin{bmatrix} 0 \\ G(n) \end{bmatrix} - \frac{\varepsilon_{0b}(n+1)}{E_b(n+1)}\begin{bmatrix} 1 \\ -A(n+1) \end{bmatrix} = \begin{bmatrix} M_0(n+1) \\ m_0(n+1) \end{bmatrix} \tag{4.35}$$

And

$$G_0(n+1) = M_0(n+1) + \frac{\varepsilon_{0b}(n+1)}{E_b(n+1)}\,B(n+1) \tag{4.36}$$

Clearly, the updating technique is the same for both adaptation gains G (n) and G_0 (n). The adequate prediction errors have to be employed.

The method used to derive the coefficient recursion (4.25) applies to linear prediction as well; hence

$$\begin{aligned} A\,(n+1) = A\,(n) &+ R_N^{-1}\,(n)\,X\,(n)[x\,(n+1) - X^t\,(n)\,A\,(n)] \\ &- R_N^{-1}\,(n)\,X\,(n-N_0)[x\,(n+1-N_0) - X\,(n-N_0)\,A\,(n)] \end{aligned} \tag{4.37}$$

or, in more concise form,

$$A\,(n+1) = A\,(n) + G\,(n)\,e_a\,(n+1) - G_o\,(n)\,e_{oa}\,(n+1) \tag{4.38}$$

Substituting (4.36) and the recursion for $r_i(n+1)$ into the above expression, lead to

$$E_0(n+1) = E_0(n) + e\varepsilon(n+1)\varepsilon_0(n+1) - e_{oa}(n+1)\varepsilon_{0a}(n+1) \tag{4.39}$$

The variables needed to perform the calculations in (4.34), and in the same equation for $G_1(n+1)$, are available and the results can be used to get the updated gains.

The backward prediction coefficient vector is updated by

$$B(n+1) = B(n) + G(n+1)\, e_b(n+1) - G_0(n+1)\varepsilon_{ob}(n+1) \tag{4.40}$$

Which leads to the set of equations:

$$\begin{aligned} &G(n+1)[1 - m(n+1)\, e_b(n+1)] \\ &= M(n+1) + m(n+1)\, B(n) - G_0(n+1)\, e_{ob}(n+1)\, m(n+1)\, G_0(n+1)[1 + m_0(n+1) \\ &e_{Ob}(n+1)] \\ &= M_0(n+1) + m_0(n+1)\, B(n) + G(n+1)\, e_b(n+1)\, m_0(n+1) \end{aligned} \tag{4.41}$$

Finally, letting $k = \dfrac{m(n+1)}{1+m_0(n+1)e_{0b}(n+1)}$, $k = \dfrac{m_0(n+1)}{1-m(n+1)e_b(n+1)}$ we obtain the adaptation gains

$$\begin{aligned} G(n+1) &= \frac{1}{1-ke_b(n+1)}\left[M(n+1) + ke_{0b}(n+1)M_0(n+1)\right] \\ G_0(n+1) &= \frac{1}{1-k_0e_{0b}(n+1)}\left[M_0(n+1) + k_0B(n) + k_0e_{0b}(n+1)M(n+1)\right] \end{aligned} \tag{4.42}$$

The algorithm is then completed by the backward coefficient updating equation (4.41).

The initial conditions are those of the algorithm, the extra operations being carried out only when the time index n exceeds the window length N_0.

Overall the sliding window algorithm based on a priori errors has a computational organization, which closely follows that of the exponential window algorithm, but it performs the operations twice to update and use its two adaptation gains.

More efficient sliding window algorithms, but with a less regular structure, can be worked out by decomposing in two different steps the sequence of operations for each new input signal sample.

4.3 The Case Of Complex Signals

Complex signals take the form of sequences of complex numbers and are encountered in many applications, particularly in communications. Adaptive filtering techniques can be applied to complex signals in a straightforward manner, the main peculiarity being that the cost functions used in the optimization process must remain real and therefore moduli are involved.

For reasons of compatibility with the subsequent study of the multidimensional case, the cost function is taken as

$$J_{eX}(n) = \sum_{p=1}^{n} W^{n-p} \left| y(p) - \hat{H}^{*}(n) X(p) \right|^2 \tag{4.43}$$

or

$$J_{eX}(n) = \sum_{p=1}^{n} W^{n-p} e(n) \bar{e}(n) \tag{4.44}$$

Where ë(n) denotes the complex conjugate of e(n), and the weighting factor W is assumed real.

Based on that cost function. FLS algorithms can be derived through the procedures presented previously.

The minimization of the cost function leads to

$$H(n) = R^{-1}{}_{N}(n)\, r_{xy}(n) \tag{4.45}$$

The connecting matrix RN+ $_1$(n + 1) can be partitioned as

$$R_{N+1}(n+1)=\sum_{p=1}^{n+1}W^{n+1-p}\begin{bmatrix}x(p)\\X(p-1)\end{bmatrix}\begin{bmatrix}\bar{x}(p),\bar{X}^t(p-1)\end{bmatrix}$$

$$=\begin{bmatrix}\sum_{p=1}^{n+1}W^{n+1-p}|x(p)|^2 & \left[\bar{r}_N^a(n+1)\right]\\ r_N^a(n+1) & R_N(n)\end{bmatrix} \qquad (4.46)$$

and

$$R_{N+1}(n+1)=\sum_{p=1}^{n+1}W^{n+1-p}\begin{bmatrix}x(p)\\X(p-N)\end{bmatrix}\begin{bmatrix}\bar{X}^t(p),\bar{x}(p-N)\end{bmatrix}$$

$$=\begin{bmatrix}R_N(n+1) & r_N^a(n+1)\\ \left[\bar{r}_N^a(n+1)\right] & \sum_{p=1}^{n+1}W^{n+1-p}|x(p-N)|^2\end{bmatrix} \qquad (4.47)$$

Following the definitions (4.45) and (4.46), the forward prediction coefficient vector is updated by

$$A\,(n+1)=R_N^{-1}\,(n)\,r_N\,(n+1)=A\,(n)+R_N^{-1}\,(n)\,X\,(n)[x\,(n+1)-X^t\,(n)\,A\,(n)] \qquad (4.48)$$

Or

$$A\,(n+1)=A\,(n)+G\,(n)\,e_a\,(n+1) \qquad (4.49)$$

Where the adaptation gain has the conventional definition and $e_a\,(n+1)=x\,(n+1)-A^t\,(n)\,X\,(n)$. Now, using the partitioning (4.46) as before, one gets

$$R_{N+1}(n+1)\begin{bmatrix}0\\G(n)\end{bmatrix}=X_1(n+1)-\begin{bmatrix}\varepsilon_a(n+1)\\0\end{bmatrix} \qquad (4.50)$$

Which, taking into account the prediction matrix equations,

leads to the same equations as for real signals:

$$G_1(n+1)=\begin{bmatrix}0\\ G(n)\end{bmatrix}+\frac{\varepsilon_a(n+1)}{E_a(n+1)}\begin{bmatrix}1\\ -A(n+1)\end{bmatrix}=\begin{bmatrix}M(n+1)\\ \mathrm{m(n+1)}\end{bmatrix} \tag{4.51}$$

The prediction error energy E_a (n + 1) can be updated by the following recursion, for R_N (n) Hermitian:

$$E_a\,(n+1) = W\,E_a\,(n) + e_a\,(n+1)\varepsilon_a\ \bar{}(n+1) \tag{4.52}$$

The end of the procedure uses the partitioning of R_{N+1} (n + 1) given in equation (4.49) to express the order N + 1 adaptation gain in terms of backward prediction variables. It can be verified that the conjugate of the backward prediction error

$$e_b\,(n+1) = x\,(n+1-N) - B^t(n)\,X\,(n+1) \tag{4.53}$$

Appears in the updated gain

$$G(n+1)=\frac{1}{1-\bar{e}_b(n+1)m(n+1)}\left[M(n+1)+B(n)m(n+1)\right] \tag{4.54}$$

The backward prediction coefficients are updated by

$$B\,(n+1) = B\,(n) + G\,(n+1)\,\ddot{e}_b(n+1) \tag{4.55}$$

Finally the FLS algorithm for complex signals based on a priori errors is for real data. There is an identity between the complex signals and the two-dimensional signals, which are considered in the next section. Algorithms for complex signals are directly obtained from those given in Figure 4.2 and Figure 4.3 by adding complex conjugation to transposition.

The prediction error ratio

$$\varphi(n) = \frac{\varepsilon_a(n+1)}{e_a(n+1)} = 1 - \overline{X}^t(n) R_N^{-1}(N) X(n) \tag{4.56}$$

is a real number, due to the Hermitian property of the AC matrix estimation RN(n). It is still limited to the interval [0, 1] and can be used as a reliable checking variable.

4.4 Multidimensional Input Signals

The input and reference signals in adaptive filters can be vectors. To begin with, The case of an input signal consisting of K elements $x_1(n)(1 \leq i \leq k)$ and a scalar reference is considered. It is illustrated in Figure 4.1. The programmable filter, who's output V (n) is a scalar like the reference y (n), consists

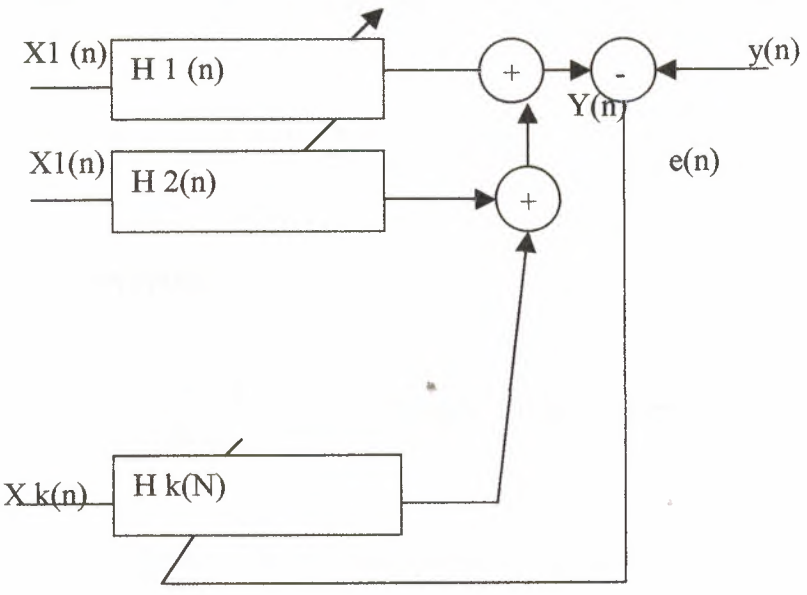


Figure 4.1 Adaptive filter with multidimensional input and scalar reference.

Of a set of k different filters with coefficient vectors H_i (n) ($1 \leq i \leq k$). These coefficients can be calculated to minimize a cost function in real time, through FLS algorithms.
Let x (n) denote the k-element input vector

$$X^t = [x_1(n), x_2(n)\ldots\ldots, \quad x_k(n)] \tag{4.57}$$

Assuming that each filter coefficient vector H_1 (n) has N elements, let X (n) denote the following input vector with KN elements:

$$X^t(n) = [x^t(n), x^t(n-1),\ldots, x^t(n+1-N)] \tag{4.58}$$

And let H (n) denote the KN element coefficient vector. H^t (n) = [h_{11} (n).. h_{KI} (n),h_{12} (n),....h_{K2} (n),. . . , h_{1N} (n) h_{kN} (n)]. The output error signal e (n) is

$$e(n) = y(n) - H^t(n)X(n) \tag{4.59}$$

The minimization of the cost function J (n) associated with an exponential time window,

$$J(n) = \sum_{p=1}^{n} W^{n-p} e^2(p) \tag{4.60}$$

Leads to the set of equations

$$\frac{\delta J(n)}{\delta h_{ij}(n)} = 2\sum_{p=1}^{n} W^{n-p}\left[y(p) - H^t(n)X(p)\right]x_i(p-j) = 0 \tag{4.61}$$

With $1 \leq I \leq K$, $0 \leq j \leq N-1$. Hence the optimum coefficient vector at time n is

$$H(n) = R_{Kn}(n)\, r_{KN}(n) \tag{4.62}$$

With

$$R_{KN}(n) = \sum_{p=1}^{n} W^{n-p} X(p) X^{t}(p)$$
$$r_{KN}(n) = \sum_{p=1}^{n} W^{n-p} y(p) X(p) \tag{4.63}$$

The matrix R_{KN} (n) is cross-correlation matrix estimation. The updating recursion for the coefficient vector takes the form

$$H(n+1) = H(n) + R_{KN}(n+1)\, X(n+1)\, e(n+1) \tag{4.64}$$

and the adaptation gain

$$G_K(n) = R_{KN}(n)\, X(n) \tag{4.65}$$

Is a KN-element vector,

The connecting matrix R_{KN1} (n + 1) is defined by

$$R_{KN1}(n+1) = \sum_{p=1}^{n} W^{n+1-p} \begin{bmatrix} X(p) \\ X(p-1) \end{bmatrix} \begin{bmatrix} X^{t}(p) X^{t}(p-1) \end{bmatrix} \tag{4.66}$$

and can be partitioned as

$$R_{KN1}(n+1) = \begin{bmatrix} \sum_{p=1}^{n+1} W^{n+1-p} x(p) X^{t}(p) & \left[r^{a}_{KN}(n+1) \right]^{t} \\ r^{a}_{KN}(n+1) & R_{KN}(n) \end{bmatrix} \tag{4.67}$$

Where r_{KN} (n + 1) is the KN x K cross-correlation matrix

$$r^{a}_{KN}(n+1) = \sum_{p=1}^{n+1} W^{n+1-p} x(p-1) X^{t}(p) \tag{4.68}$$

From the alternative definition

$$R_{KN1}(n+1)=\sum_{p=1}^{n+1}W^{n+1-p}\begin{bmatrix}X(p)\\X(p-N)\end{bmatrix}\begin{bmatrix}X^t(p)X^t(p-N)\end{bmatrix} \tag{4.69}$$

A second partitioning is obtained:

$$R_{KN1}(n+1)=\begin{bmatrix}R_{KN}(n+1) & r^b_{KN}(n+1)\\ \left[r^b_{KN}(n+1)\right]^t & \sum_{p=1}^{n+1}W^{n+1-p}x(n+1-N)X^t(n+1-N)\end{bmatrix} \tag{4.70}$$

Where $r_{KN}(n+1)$ is the KN x K matrix

$$r^b_{KN}(n+1)=\sum_{p=1}^{n+1}W^{n+1-p}X(p)X^t(p-N) \tag{4.71}$$

The fast algorithms use the prediction equations. The forward prediction error takes the form of a K-element vector

$$e_{Ka}(n+1)=x(n+1)-A_K,(n)\,X(n) \tag{4.72}$$

Where the prediction coefficients form a KN x K matrix, which is computed to minimize the prediction error energy, defined by

$$E_a(n)=\sum_{p=1}^{n}W^{n-p}e^t{}_{ka}(p)e_{ka}(p)=trace\left[E_{ka}(n)\right] \tag{4.73}$$

with the quadratic error energy matrix defined by

$$E_{Ka}(n)=\sum_{p=1}^{n}W^{n-p}e_{ka}(p)e^t{}_{ka}(p) \tag{4.74}$$

The minimization process yields

$$A_K (n + 1) = R_{KN} (n)\, r_{KN} (n + 1) \tag{4.74}$$

the forward prediction coefficients, updated by

$$A_K (n + 1) = A_K (n) + G_K (n)\, e_{ka} (n + 1) \tag{4.75}$$

are used to derive the a posteriori prediction error $\varepsilon_{Ka}(n + 1)$, also a K-element vector, by

$$\varepsilon_{ka} (n + 1) = X (n + 1) - A_K (n + 1)\, X (n) \tag{4.76}$$

The quadratic error energy matrix can also be expressed by

$$E_{Ka}(n+1) = \sum_{p=1}^{n+1} W^{n+1-p} X(p) X^t(p) - A_K^t(n+1) r_{KN}^a (n+1) \tag{4.77}$$

Which, by the same approach as, yields the updating recursion

$$E_{Ka} (n + 1) = W E_{Ka} (n) + e_{ka} (n + 1)\, \varepsilon_{ka} (n + 1) \tag{4.78}$$

The a priori adaptation gain G_K (n) can be updated by reproducing the developments given in and using the two partitioning equations (4.61) and (4.66) for R_{KN1} (n + 1). The fast algorithm based on a priori errors is given in Figure 4.2.

If the predictor order N is sufficient, the prediction error elements, in the steady-state phase, approach white noise signals and the matrix E_{Ka} (n) approaches a diagonal matrix. Its initial value can be taken as a diagonal matrix

$$E_{Ka} (0) = E_O\, I_K \tag{4.79}$$

Where F_0 is a positive scalar; all other initial values can be zero.

A stabilization constant, as, can be introduced by modifying recursion (4.74) as follows:

$$E_{Ka}(n+1) = WE_{Ka}(n) + e_{Ka}(n+1)\,\varepsilon_{ka}(n+1) + CI_K \qquad (4.80a)$$

Where C is a positive scalar.

The matrix inversion in Figure 4.2 is carried out, by updating the inverse quadratic error matrix:

$$E^{-1}{}_{Ka}(n+1) = W^{-1}\left[E_{ka}^{-1}(n) - \frac{E_{ka}^{-1}(n)e_{ka}(n+1)\varepsilon_{ka}^{t}(n+1)E_{ka}^{-1}(n)}{W + \varepsilon_{ka}^{t}(n+1)E_{ka}^{-1}(n)e_{ka}(n+1)}\right] \qquad (4.81)$$

The computational complexity of that expression amounts to $3K^2 + 2K$ multiplications and one division or inverse calculation.

ALGORITHM F.L.S. 1-K

AVAILABLE AT TIME n

COEFFICIENTS OF ADAPTIVE FILTER: H (n)

FORWARD PREDICTION MATRIX: AK (fl)

BACKWARD PREDICTION MATRIX: BK (fl)

DATA VECTOR: X (n)

ADAPTATION GAIN: GK (n)

QUADRATIC ERROR MATRIX: EKa (n)

WEIGHTING FACTOR: W

NEW DATA AT TIME n:

Input signal: X (n+l) Reference: y (n+l)

ADAPTATION GAIN UPDATING•

$$e_{Ka}(n+l) = X(n+1) - A_{Kt}(n)\,X(n)$$

$$A_K(n+1) = A_K(n) + G_K(n)\,e_{Ka}(n+l)$$

$$\epsilon_{Ka}(n+1) = X(n+1)\,A_K(n+l)\,X(n)$$

$$E_{Ka}(n+l) = W_{Ea}(n) + e_{Ka}(n+l)\,E_{ka}(n+l)$$

$$G_K(n+1) = \begin{bmatrix} 0 \\ G_K(n) \end{bmatrix} + \begin{bmatrix} I_K \\ -A_K(n+1) \end{bmatrix} E_{ka}^{-1}(n+1)\varepsilon_{ka}(n+1) = \begin{bmatrix} M_K(N+1) \\ m_k\ (n+1) \end{bmatrix}$$

$$e_{KB}(n+l) = X(n+1 - N) - B_K^{t}(n)\,X(n+1)$$

$$B_K(n+l) = B_K(n) + G_K(n+l)\,e_{Kb}(n+l)$$

ADAPTIVE FILTER

$$E(n+1) = y(n+l) - H^{t}(n)\,X(n+l)$$

$$H(n+l) = H(n) + G_K(n+l)\,e(n+1)$$

Figure 4.2 FLS algorithm for multidimensional input signals.

Note that if N = 0, which means that there is no convolution on the input data, then E_{ka} (n) is just the inverse cross-correlation matrix R_{ka} (n), and it is updated directly from the input signal data as in conventional RLS techniques.

For the operations related to the filter order N, the algorithm presented in Figure 4.2 requires $7K^2N + KN$ multiplications for the adaptation gain and 2KN multiplications for the filter section. The FORTRAN program is given in Annex 4.1.

The ratio φ (n) of a posteriori to a priori prediction errors is still a scalar, because

$$\varepsilon_{aK} (n + 1) = e_{ak} (n + 1)[1 - G_k (n) X (n)] \tag{4.82}$$

Therefore it can still serve to check the correct operation of the multidimensional algorithms. Moreover, it allows us to extend to multidimensional input signals the algorithms based on all prediction errors.

4.5 M-D Algorithm Based On All Prediction Errors

An alternative adaptation gain vector, which leads to exploiting a priori and a posteriori prediction errors, is defined by

$$G_k (n + 1) = R_{Kn} (n) X (n + 1) = G_K (n + 1) W/ \varphi (n+1) \quad (4.83)$$

The updating procedure uses the ratio of a posteriori to a priori prediction errors, under the form of the scalar cc (n) defined by

$$\alpha (n) = W + X^t (n) R_{KN} (n - 1) X (n) = W/\varphi (n) \quad (4.84)$$

The computational organization of the corresponding algorithm is shown in Figure 4.3. Indeed, but vectors and matrices when appropriate have replaced scalars and vectors.
The operations related to the filter order N correspond to $6K^2N$ multiplications for the gain and 2KN multiplications for the filter section.
In the above procedure, the backward a priori prediction error vector $e_{Kb} (n + 1)$ can also be calculated directly by

$$e_{Kb} (n + 1) = E_{Kb} (n) m_K (n + 1) \quad (4.85)$$

Again that provides means to control the roundoff error accumulation, through updating the backward prediction coefficients, of

ALGORITHM F.L.S.2-K

AVAILABLE AT TIME n

COEFFICIENTS OF ADAPTIVE FILTER: $H(n)$

FORWARD PREDICTION MATRIX: $A_K(fl)$

BACKWARD PREDICTION MATRIX: $B_K(n)$

DATA VECTOR: $X(n)$

ADAPTATION GAIN: $G_k(n)$

OUADRATIC ERROR MATRICES: $E_{Ka}(n)$, $E_{Kb}(n)$

PREDICTION ERROR RATIO: $\alpha(n)$

$$e_{Kb}(n+l) = X(n+l-N) - B_{Kt}(n+1)\,X(n+l)$$

$$G_k(n+1) = M_K(n+l) + B_k(n)\,m_K(n+l)$$

WEIGHTING FACTOR: W

NEW DATA AT TIME fl:

Input signal: $X(n+1)$ Reference: $y(n+1)$

ADAPTATION GAIN UPDATING

$$e_{Ka}(n+l) = X(n+1) - A_{Kt}(n)\,X(n)$$

$$A_K(n+l) = A_K(n) + G_K((n)\,e_{Ka}^{t}(n+l) / x(n)$$

$$E_{Ka}(n+l) = (E_{Ka}(n) + e_{Ka}(n+l)\,e_{Kat}(n+l)/\alpha(n))\,W$$

$$G_{K1}(n+1) = \begin{bmatrix} 0 \\ G_K(n) \end{bmatrix} + \begin{bmatrix} I_K \\ -A_K(N+1) \end{bmatrix} E_{ka}^{-1}(n+1)e_{ka}(n+1) = \begin{bmatrix} M_K(N+1) \\ m_k\ (n+1) \end{bmatrix}$$

$$Cx_1(n+1) = cx_1(n+1) - e_{Kb}(n+l)\,m_K(n+l)$$

$$E_{Kb}(n+l) = (E_{Kb}(n) + e_{Kb}(n+l)\,e_{Kb}^{t}(n+1)/\alpha(n+1))\,W$$

$$B_K(n+l) = B_K(n+l) + G_k((n)\,e_{Kb}(n+l)/\alpha(n+1)$$

ADAPTIVE FILTER

$$e(n+1) = y(n+1)\ H^{t}(n)\,X(n+1)$$

$$H(n+1) = H(n) + G_K(n+1)\,e(n+l)/\alpha(n+l)$$

Figure 4.3 Algorithm based on all prediction errors for M-D input signals.

$$B_K(n+1) = B_K(n) + G_k(n+1)$$
$$X\ [e_{Kb}(n+1) + C_{Kb}(n+1) - E_{Kb}(n)\, m_k(n+1)]/\alpha(n+I) \quad (4.86)$$

Up to now, the reference signal has been assumed to be a scalar sequence. The adaptation gain calculations, which have been carried out, only depend on the input signals, and they are valid for multidimensional reference signals as well. The case of K-dimensional (K-D) input and L-dimensional (L-D) reference signals is depicted in Figure 4.4. The only modifications with respect to the previous algorithms concern the filter section. The L-element reference vector $Y_L(n)$ is used to derive the output error vector $e_L(n)$ from the input and the KN x L coefficient matrix $H_L(n)$ as follows:

$$e_L(n+1) = Y_L(n+1) - H_L(n)\, X(n+1) \quad (4.87)$$

The coefficient matrix is updated by

$$H_L(n+1 = H_L(n) + G_k(n+1)\, e_L(n+1)/\alpha(n+1) \quad (4.88)$$

The associated complexity amounts to 2NKL + L multiplications.

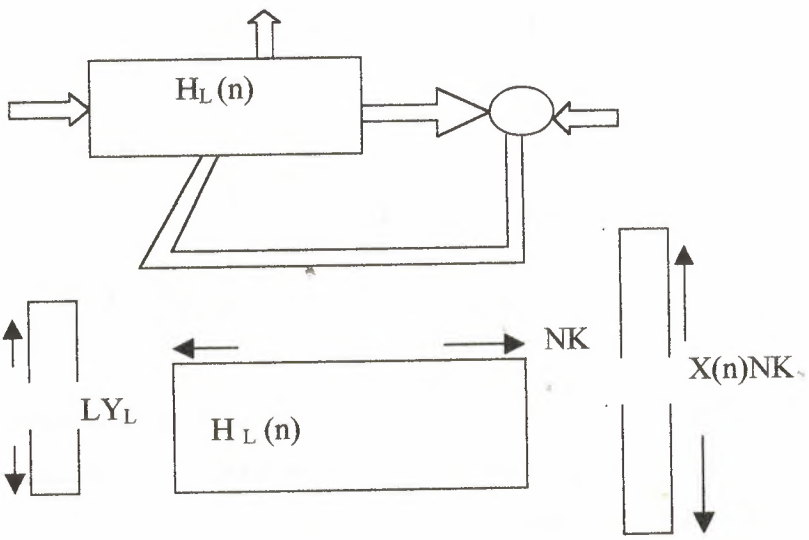


Figure 4.4 Adaptive filter with M-D input and reference signals.

The developments and the preceding sections have illustrated the flexibility of the procedures used to derive fast algorithms. Another example is provided by filters of nonuniform length.

4.6 Filters Of Nonuniform Length

In practice it is desirable to tailor algorithms to meet the specific needs of applications. The input sequences may be fed to filters with different lengths, and adjusting the fast algorithms accordingly can provide substantial savings.

Assume that the K filters in Figure 4.1 have lengths N_I ($1 \leq I \leq K$). The data vector X (n) can be rearranged as follows:

$$X^t(n) = [Xi\ (n),\ Xi\ (n)\ X_k\ (n)] \tag{4.91}$$

Where $X^t(n) = [x_i(n), x_i(n-1),\ldots,x_i(n+1-N_i)]$. The number of elements EN is

$$\Sigma N = \sum Ni \tag{4.92}$$

The connecting ($\Sigma N + K$) ($\Sigma N + K$) matrix $R_{\Sigma N1}$ (n + 1). Defined by

$$R_{\varepsilon N1}(n+1) = \begin{bmatrix} x_1(n+1) \\ X_1(n) \\ X_K(n+1) \\ X_K(n) \end{bmatrix} \left[x_1(n+1), X_1^t(n),\ldots,\ x_K(n+1), X_K^t(n)\right] \tag{4.93}$$

can again be partitioned in two different manners and provide the gain updating operations. The algorithms obtained are those shown in Figures 4.2 and 4.3. The only difference is that the prediction coefficient EN x K matrices are organized differently to accommodate the rearrangement of the data vector X(n).

A typical case where filter dimensions can be different is pole-zero modeling.

4.7 FLS Pole-Zero Modeling

Pole-zero modeling techniques are used in control for parametric system identification .
An adaptive filter with zeros and poles can be viewed as a filter with 2-D
Input data and 1 -D reference signal. The filter defined by

$$Y(n+1) = A^t(n)\, X(n+1) + B^t(n)\, Y(n) \tag{4.94}$$

is equivalent to a filter as in Figure 4.1 with input signal vector

$$x(n+1) = \begin{bmatrix} x(n+1) \\ \tilde{y}(n) \end{bmatrix} \tag{4.95}$$

For example, let us consider the pole-zero modeling of a system with output y (n) when fed with x(n). An approach, which ensures stability, A 2-D FLS algorithm can be used to compute the model coefficients with input signal vector

$$x(n+1) = \begin{bmatrix} x(n+1) \\ y(n) \end{bmatrix} \tag{4.96}$$

However, as pointed out, that series-parallel type of approach is biased when noise is added to the reference signal. It is preferable to use the parallel approach. But stability can only be guaranteed if the smoothing filter with: -transfer function C (z) satisfying strictly positive real (SPR) is introduced on the error signal.
An efficient approach to pole-zero modeling is obtained by incorporating the smoothing filter in the LS process [6]. A 3-D FLS algorithm is employed, and the corresponding diagram is shown in Figure 4.5. The output error signal f (n) used in the adaptation process is

$$F(n) = y(n) - [u_1(n) + u_2(n) + u_3(n)] \tag{4.97}$$

Where u_1 (n), u_2 (n), and u_3 (n) are the outputs of the three filters fed by y (n), x (n), and e (n) = y (n) - y' (n), respectively.

The cost function is

$$J_3(n) = \sum_{p=1}^{n} W^{n-p} f^2(p) \tag{4.98}$$

Let the unknown system output be

$$y(n) = \sum_{i=0}^{N} a_i x(n-i) + \sum_{i=1}^{N} b_i y(n-i) \tag{4.99}$$

or

$$y(n) = \sum_{i=0}^{N} a_i x(n-i) + \sum_{i=1}^{N} b_i \tilde{y}(n-i) + \sum_{i=1}^{N} b_i e(n-i) \tag{4.100}$$

From (7.86), the error signal is zero in the steady state if

$a_i(\infty) = a_i$, $b_i(\infty) = b_i$, $c_i(\infty) = b_i$, $\quad 1 \le i \le N$

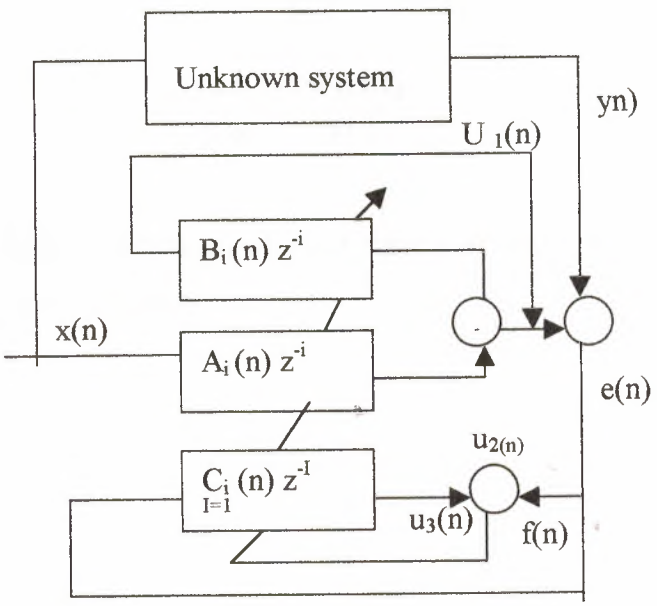


Figure 4.5 adaptive pole-zero modeling with a 3-D FLS algorithm.

Now, assume that a white noise sequence x (n) with power a is added to the system output. The cost function to be minimized becomes

$$J_{3n}(n)=\sum_{p=1}^{n}W^{n-p}\left[f(p)+n(p)-\sum_{i=1}^{N}n(p-i)\right]^2 \qquad (4.101)$$

Which, for sufficiently large n can be approximated by

$$J_{3n}(n)=\sum_{p=1}^{n}W^{n-p}\left[f^2(p)+\sigma_\eta^2\left[1+\sum_{i=1}^{N}c_i^2(p)\right]\right] \qquad (4.102)$$

The steady-state solution is

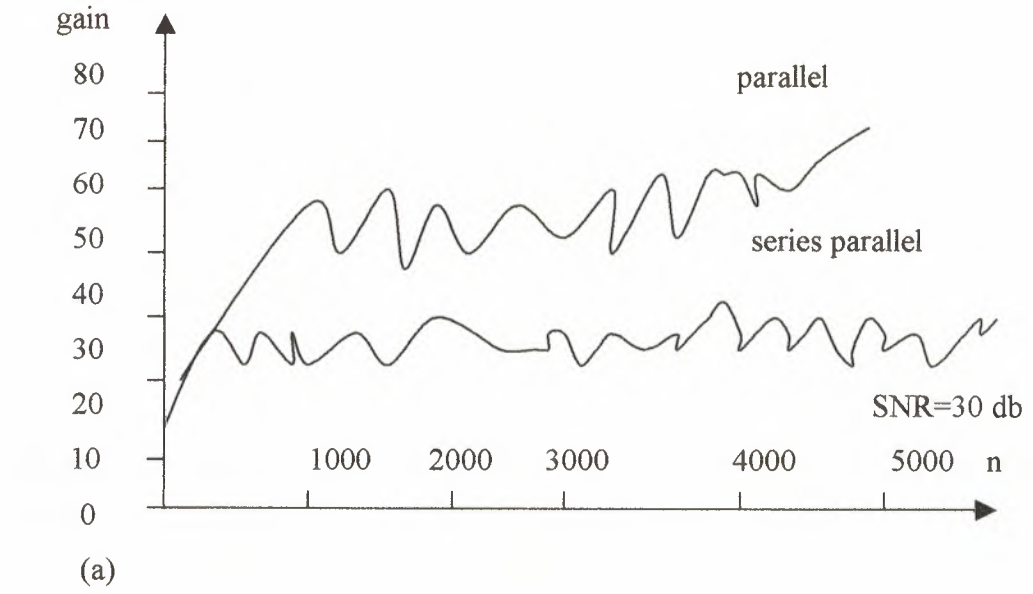
$a_i(\infty) = a_i,\ b_i(\infty) = b_i,\ c_i(\infty) = 0, \qquad 1 \le I \le N$

Finally, the correct system identification is achieved, in the presence of noise or not. The smoothing filter coefficients vanish on the long run when additive noise is present. An illustration is provided by the following example.

Let the transfer function of the unknown system be

$$H(z)=\frac{0.05+0.1z^{-1}+0.07z^{-2}}{1-0.96z^{-1}+0.94z^{-2}} \qquad (4.103)$$

And let the input be the first-order AR signalx (n) = e_0 (n) + 0.8x (n - 1) Where e_0 (n) is a white Gaussian sequence.



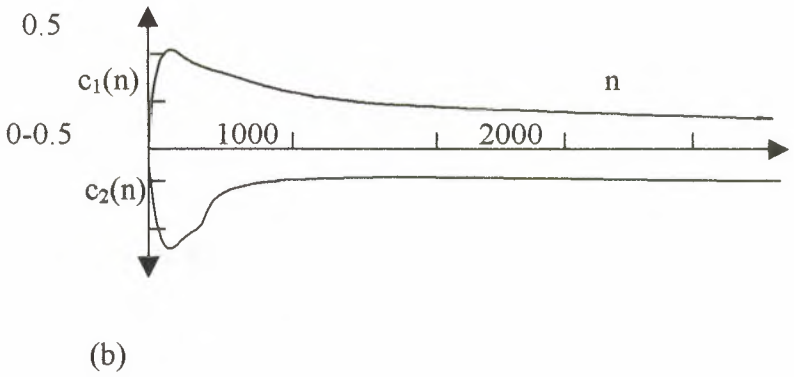


Figure 4.6 Pole-zero modeling of an unknown system: (a) System gain in FLS identification. (b) Smoothing filter coefficients.

The system gain G_s defined by

$$G_s = 10 \log \frac{E\left[y^2(n)\right]}{E\left[e^2(n)\right]} \tag{4.104}$$

is shown in Figure 4.6(a) versus time. The ratio of the system output signal to additive noise power is 30 dB. For comparison the gain obtained with the series-parallel approach is also given. In accordance with expression, it is bounded by the SNR. The

smoothing filter coefficients are shown in Figure 4.6(b). They first reach the b_1 values (I = 1, 2) and decay to zero after.

The 3-D parallel approach requires approximately twice the number of multiplications of the 2-D series-parallel approach.

4.8 Multirate Adaptive Filters

The sampling frequencies of input and reference signals can be different. In the sample rate reduction case, depicted in Figure 4.7, the input and reference sampling frequencies are f_s and $f_{s/k}$ respectively. The input signal sequence is used to form K sequences with sample rate $f_{s/K}$ which are fed to K filters with coefficient vectors H_1 (n)($0 \leq I \leq K - 1$). The cost function to be minimized in the adaptive filter, J_{sRR} (Kn), is

$$J_{sRR}(Kn) = \sum_{p=1}^{n} W^{n-p}\left[y(kp) - H^t(Kp)X(Kp)\right]^2 \tag{4.105}$$

The data vector X Kn) is the vector of the NK most recent input values. The input may be considered as consisting of K different signals, and the algorithms presented in the preceding sections can be applied. The corresponding calculations are carried out at the frequency

The sample rate increase case is shown in Figure 4.8. It corresponds to 1-D input and multidimensional reference signals.

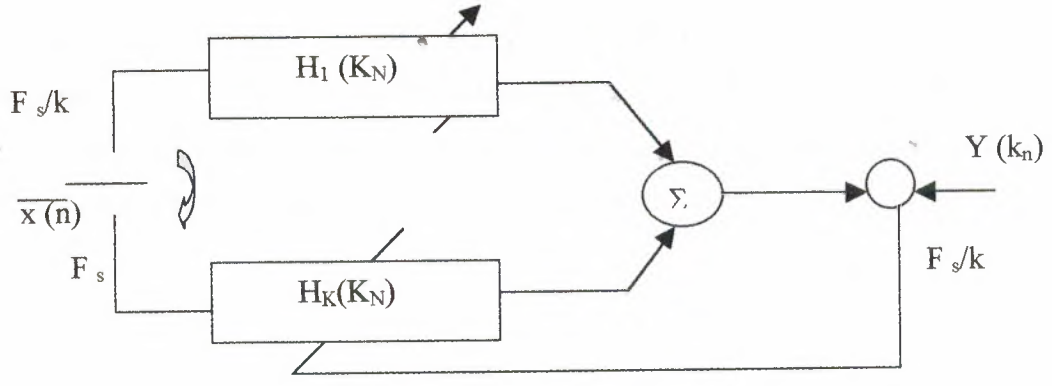


Figure 4.7 Sample rate reduction adaptive filter.

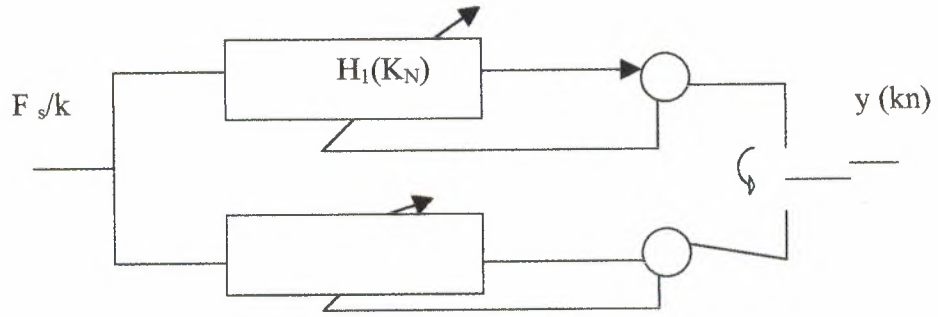


Figure 4.8 Sample rate increase adaptive filter.

It is much more economical in terms of computational complexity than the sample rate reduction, because the adaptation gain is computed once for the K interpolating filters. All the calculations are again carried out at frequency $f_{s/N}$ the reference sequence being split into K sequences at that frequency. The system boils down to K different adaptive filters with the same input.

In signal processing, multirate aspects are often linked with DFT applications and filter banks, which correspond to frequency domain conversions.

4.9 Frequency Domain Adaptive Filters

The power conservation principle states that the power of a signal in the time domain equals the sum of the powers of its frequency components. Thus, the LS techniques and adaptive methods worked out for time data can be transposed in the frequency domain.

The principle of a frequency domain adaptive filter (FDAF) is depicted in Figure 4.9. The N-point DFTs of the input and reference signals are computed. The complex input data obtained are multiplied by complex coefficients and subtracted from the reference to produce the output error used to adjust the coefficients.

At first glance, the approach may look complicated and farfetched. However, there are two motivations.

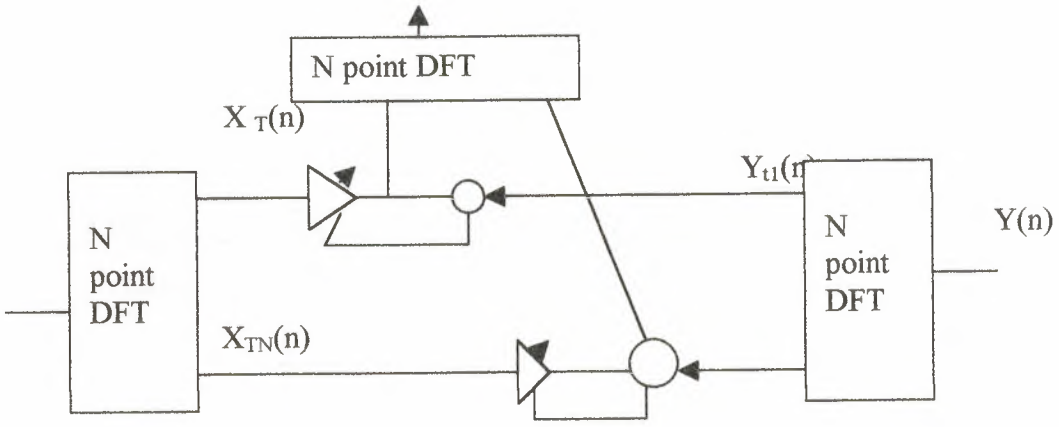


Figure 4.9 FDAF Structure.

View, the DFT computer is actually a filter bank which performs some orthogonalization of the data thus, an order N adaptive filter becomes a set of N separate order 1 filters. Second, from a practical standpoint, the efficient FFT algorithms to compute the DFT of blocks of N data, particularly for large N, can potentially produce substantial savings in computation speed, because the DFT output sampling frequency can be reduced by the factor N.

Assuming N separate complex filters and combining the results we obtain the LS solution for the coefficients

$$h_i(n) = \frac{\sum_{p=1}^{n} W^{n-p} \tilde{y}_{Ti}(p) x_{Ti}(p)}{\sum_{p=1}^{n} W^{n-p} x_{Ti}(p) \tilde{x}_{Ti}(p)} \qquad (4.106)$$

Where $X_{TI}(n)$ and $y_{Ti}(n)$ are the transformed sequences.

For sufficiently large n, the denominator of that equation is an estimate of the input power spectrum, and the numerator is an estimate of the cross-power spectrum between input and reference signals. Overall the FDAF is an approximation of the optimal Wiener filter, itself the frequency domain counterpart of the time domain filter

associated with the normal equations. Note that the optimal method along these lines, in case of stationary signals, would be to use.

The updating equations associated with (4.104) are

$$H_i(n+1) = h_i(n) + r_i^{-1}(n+1)\, x_{TI}(n+1) \times [Y_{T1}(n+1) - h_i(n)\, x_{T1}(n+1)] \qquad (4.106)$$

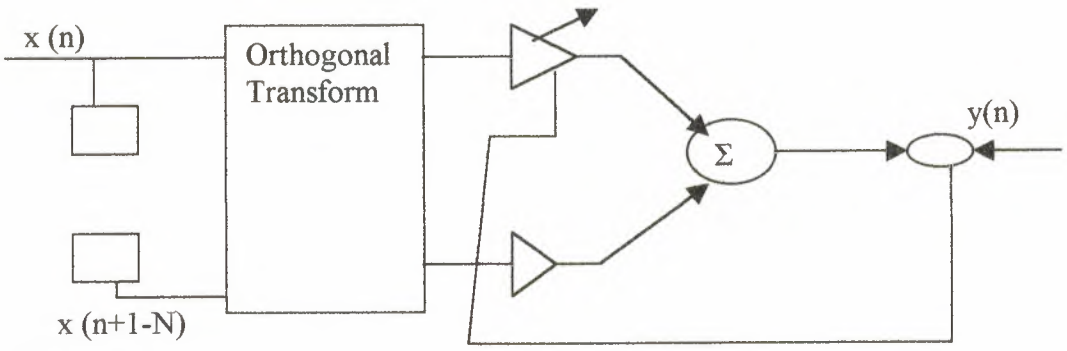


Figure 4.10 FDAF with a single orthogonal transform

and

$$r_i(n+1) = Wr_i(n) + x_{Ti}(n+1)\, 8_{T1}(n+1) \qquad (4.107)$$

The FFT algorithms need about $(N/2)\log_2(N/2)$ complex multiplications each, which have to be added to the N order 1 adaptive filter operations. Altogether savings can be significant for large N, with respect to FLS algorithms.

The LMS algorithm can also be used to update the coefficients, and the results given in can serve to assess complexity and performance.

It must be pointed out that the sample rate reduction by N at the DFT output can alter the adaptive filter operation, due to the circular convolution effects [8]. A scheme without sample rate reduction is shown in Figure 4.10, where a single orthogonal transform is used. If the first row of the transform matrix consists of 1's only, the inverse transformed data are obtained by just summing the transformed data [10]. Note also that complex operations are avoided if a real transform, such as the DCT is used.

A general observation about the performance of frequency domain adaptive filters is that they can yield poor results in the presence of nonstationary signals, because the sub band decomposition they include can enhance the nonstationary character of the signals.

CONCLOUSION

Adaptive filtering has emerged as an important part of signal processing, rich in both theory and application. We say that a filter, be it in hardware or software form, is *adaptive* if it satisfies two requirements:

1. The filter has a built-in mechanism for the automatic adjustment of its coefficients in response to statistical variations of the environment in which the filter operates.

2. The coefficient adjustments are made for the purpose of progressively moving the filter (in an iteration-by-iteration or block-by-block manner) toward an optimum performance; here, optimality is defined in some' statistical sense.

Adaptive filter theory applies to the processing of a *time series as* well as a *space series*. In the temporal case, the filter input may consist of a "vector" of uniformly spaced samples taken from a long data stream. In the spatial case, the filterInput may be consist of a "snapshot" of elemental outputs derived from an array of uniformly spaced sensors at a particular instant of time. In some practical situaation,the umformity of data samples is not adhered to.

Adaptive filters have heen successfully applied in diverse fields, inc adaptive equalizers, echo cancellers, adaptive heamformers for sonar and speech encoders, time-varying spectrum estimators, and system identification

REFERENCES

[1]. Ulryer. T. J., And R. W. Clayton (1976). "Time series modelling and maximum entropy," Phys. Earth Planet. Inter., vol. 12, pp.188-200.

[2]. Bracewell. R. N., The Fourier Transform and Its Applications. McGraw-Hill, New York, 1987.

[3]. Bershad, N.J. (1986). "Analysis of the normalized LMS algorithm with Gaussian inputs," IEEE Trans. Acoust. Speech Signal Process., vol. ASSP-34, pp. 793-806

[4]. Ungerboeck, 0. (1972). "Theory on the speed of convergence in adaptive equalizers for digital communication," IBM J. Res. Dev., vol.16, pp.546-555.

[5]. Van Den Bos, A. (1971). "Alternative interpretation of maximum entropy spectral analysis," IEEE Trans. Inf. Theory, vol. IT-17, pp.493-494.

[6]. Van Huefel, S., J. VANDEWALLE, and A. HAEGEMANS (1987). "An efficient and reliable algorithm for computing the singular subspace of a matrix, associated with its smallest singular values," J. Computational and Applied Mathematics, vol.19, pp.313-330.

[7]. Bienvenue, G., And L. KOPP (1980). "Adaptivity to background noise spatial coherence for high resolution passive methods," Proc. ICASSP, Denver, Cob., pp.307-310.

[8]. Van Loan, C. (1989). "Matrix Computations in Signal Processing," Chapter 4 in the book entitled Selected Topics in Signal Processing, edited by S. Haykin, Prentice-Hall.

[9]. Van Trees, H. L. (1968). Detection, Estimation and Modulation Theory, part I, Wiley, New York.

[10]. Botto, J. L., And G. V. Moustakides (1989). "Stabilizing the fast Kalman algorithms," IEEE Trans. Acoust., Speech Signal Process., vol. ASSP-37, pp. 1342-1348.

[11]. Verdu, 5. (1984). "On the selection of memoryless adaptive laws for blind equalization in binary communications, Proc. 6th Intern. Conference on Analysis and Optimization of Systems, Nice, France, pp.239-249.

[12]. Verhaegen, M. H. (1989). "Round-off error propagation in four generally-applicable, recursive, least-squares estimation schemes," Automatica, vol.25, pp.437-444.

[13]. Box, G. E. P., And G. M. JENKINS (1976). Time Series Analysis: Forecasting and Control, Holden-Day, San Francisco.

[14]. Volder, J. E. (1959). "The CORDIC trigonometric computing technique, IEEE Thans. Electron. Comput., vol. EC-8, pp.330-334.

[15]. Wakıta, H. (1973). "Direct estimation of the vocal tract shape by inverse filtering of acoustic speech waveforms," IEEE Trans. Audio Electroacoust., vol. AU-21, pp.417-427.

[16]. Walach, E., And B. Widrow (1984). "The least mean fourth (LMF) adaptive algorithm and its family," iEEE Trans. Inf. Theory, vol. IT-30, Special Issue on Linear Adaptive Filtering, pp.275-283.

[17]. Walzman, T., And M. Schwam7 (1973). "Automatic equalization using the discrete frequency domain," IEEE Trans. Inf. Theory, vol. TT-19, pp.59-68.

[18]. Ward, C. R., Er Al. (1984). "Application of a systolic array to adaptive beamforming," Proc. IEE (London), vol.131, pt. F, pp.638-645.

[19]. WAX, M. (1985). "Detection and estimation of superimposed signals," Ph. D. dissertation, Stanford University, Stanford, Calif.

[20]. Bienvenue, G., And H. Mermoz (1985). "Principles of high-resolution array processing," in VLSI and Modern Signal Processing, ed. S. Y. Kung, H. J. Whitehouse, and T. Kailath, Prentice-Hall, Englewood Cliffs, N.J., pp.83.59

[21]. Bierman, G. J. (1977). Factorization Methods for Discrete Sequential Estimation. Academic Press, New York.

[22]. Bitmead, R. R., And B. D. 0. Anderson (1980). "Lyapunov Techniques for the Exponential Stability of Linear Difference Equations with Random Coefficients." IEEE Trans. Autom. Control, vol. AC.25, pp.782-787.

[23]. Bj6rck, A. (1967). "Solving linear least squares problems by Gram-Schmidt orthogonalization," BIT, vol.7, pp. 1-21.

[24]. Bode, H. W., And C. E. Shannon (1950). "A simplified derivation of linear least square smoothing and prediction theory," Proc. IRE, vol.38, pp. 417-425.

[25]. Bojanczyk, A. W., And F. T. Luk (1987). "A novel MVDR beamforming algorithm," Proc. SPIE, vol.826, Advanced Algorithms and Architectures for Signal Processing, pp.12-16.

[26]. Bottomley, G.E., And S. T. Alexander (1989). "A theoretical basis for the divergence of conventional recursive least squares filters," Proc. IEEE International Conf. on Acoustics, Speech, Signal Processing, ICASSP'89, Glasgow, Scotland, pp.908-911.

[27]. Bracewell, R. N. (1978). The Fourier Transform and Its Applications, 2nd ed., McGraw-Hill, New York.