1

Introduction

As we begin our study of "adaptive filters," it may be worth trying to understand the meaning of the terms *adaptive* and *filters* in a very general sense. The adjective "adaptive" can be understood by considering a system that is trying to adjust itself so as to respond to some phenomenon that is taking place in its surroundings. In other words, the system tries to adjust its parameters with the aim of meeting some well-defined goal or target that depends on the state of the system as well as its surrounding. This is what "adaptation" means. Moreover, there is a need to have a set of steps or certain procedure by which this process of "adaptation" is carried out. And finally, the "system" that carries out and undergoes the process of "adaptation" is called by the more technical, yet general enough, name "filter" – a term that is very familiar to and a favorite of any engineer. Clearly, depending on the time required to meet the final target of the adaptation process, which we call convergence time, and the complexity/resources that are available to carry out the adaptation, we can have a variety of adaptation algorithms and filter structures. From this point of view, we may summarize the contents/contribution of this book as "the study of some selected adaptive algorithms and their implementations along with the associated filter structures from the points of view of their convergence and complexity performance."

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 processes 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 *nonlinear*. 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 behavior is governed by *the principle of superposition*. This means that if the responses of a linear discrete-time system to input sequences

Adaptive Filters: Theory and Applications, Second Edition. Behrouz Farhang-Boroujeny.

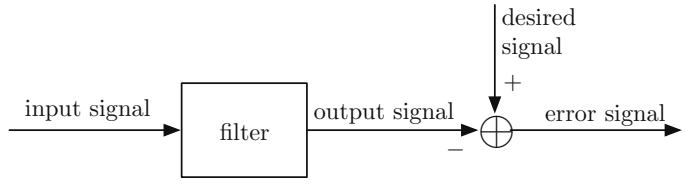


Figure 1.1 Schematic diagram of a filter emphasizing its role of reshaping the input signal to match the desired signal.

$x_1(n)$ and $x_2(n)$ are $y_1(n)$ and $y_2(n)$, respectively, 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 *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.

Figure 1.1 depicts a general schematic diagram of a filter emphasizing the purpose for which it is used in different problems addressed/discussed in this book. In particular, the filter is used to reshape a certain *input signal* 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-squared value* of the *error signal*, that is, difference between the desired signal and the filter output. For stationary input and desired signals, minimizing the mean squared error (MSE) results in the well-known *Wiener filter*, which is said to be *optimum in the mean-square sense*. The subject of Wiener filters is extensively covered in Chapter 3. Most of the adaptive algorithms that are studied in this book are practical solutions to Wiener filters. 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 that 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, that is, the Wiener filter, for large data lengths. Chapters 12 and 13 deal with the deterministic approach in detail. We refer the reader to Section 1.4 for a brief overview of the adaptive formulations under the stochastic (i.e., statistical) and deterministic frameworks.

1.2 Adaptive Filters

As we mentioned in the previous section, the filter required for estimating the given desired signal can be designed using either the stochastic or the 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 practically not feasible because 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, using the 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. All the algorithms that are covered in this book take the output error of the filter, correlate that with the samples of filter input in some way, and use the result in a recursive equation to adjust the filter coefficients iteratively. The reasons for solving the problem of adaptive filtering in an iterative manner are as follows:

1. Direct computation of the necessary averages and their application for computing the filter coefficients requires 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. Accumulation of signal samples and their postprocessing to generate the filter output, as required in noniterative 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, 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 implement in hardware than their noniterative counterparts.

1.3 Adaptive Filter Structures

The most commonly used structure in the implementation of adaptive filters is the *transversal structure*, depicted in Figure 1.2. 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 Equation (1.1)

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

where $w_i(n)$'s are the filter *tap weights* (coefficients) and N is the filter length. We refer to the input samples, $x(n-i)$, for $i = 0, 1, \ldots, N-1$, as the filter *tap inputs*. The tap weights, $w_i(n)$'s, which may vary with time, are controlled by the adaptation algorithm.

In some applications, such as beamforming (Section 1.6.4), the filter tap inputs are not the delayed samples of a single input. In such cases, the structure of the adaptive filter

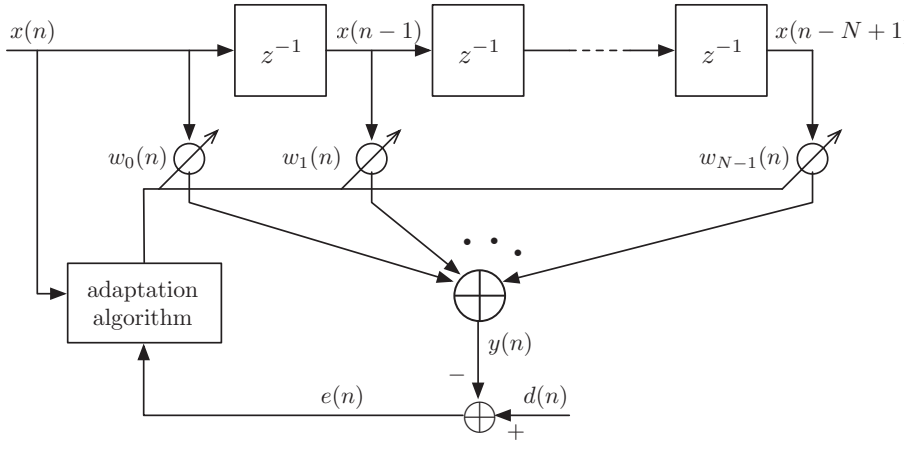


Figure 1.2 Adaptive transversal filter.

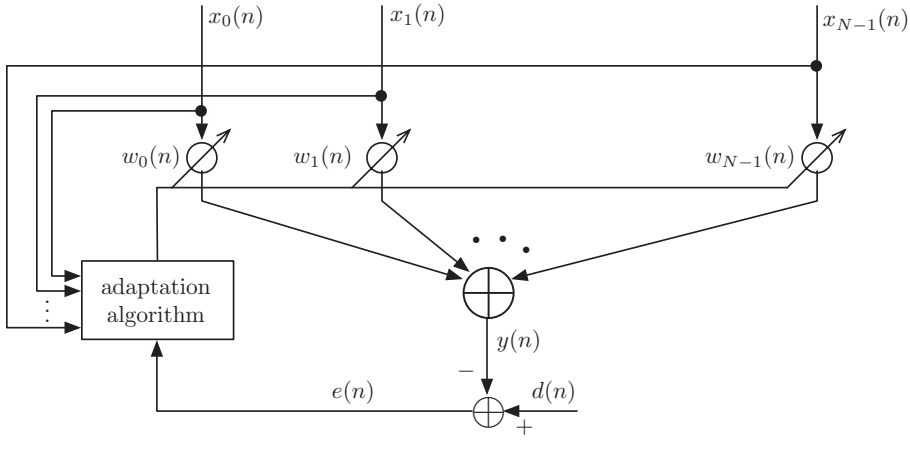


Figure 1.3 Adaptive linear combiner.

assumes the form shown in Figure 1.3. This is called *a linear combiner* as its output is a linear combination of the different signals received at its tap inputs:

$$y(n) = \sum_{i=0}^{N-1} w_i(n) x_i(n) \tag{1.2}$$

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 $x_i(n) = x(n - i)$.

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

$$y(n) = \sum_{i=0}^{N-1} a_i(n)x(n-i) + \sum_{i=1}^{M-1} b_i(n)y(n-i) \tag{1.3}$$

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 IIR filters, their application in the area of adaptive filters is rather limited. In particular, they can easily become unstable because their poles may get shifted out of the unit circle (i.e., $|z| = 1$, in the z-plane, Chapter 2) by the adaptation process. Moreover, the performance function (e.g., MSE as a function of filter coefficients) of an IIR 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 MSE functions of FIR filter and linear combiner are well-behaved quadratic functions with a single minimum point, which can easily be found through various adaptive algorithms. Because of these points, the nonrecursive filters are the sole candidates in most of the applications of adaptive filters. Hence, most of our discussions in the subsequent chapters are limited to the nonrecursive filters. The IIR-adaptive filters with two specific examples of their applications are discussed in Chapter 10.

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. Derivation of lattice structures for both FIR and IIR filters are presented in Chapter 11. Also, in the implementation of the method of least-squares (Section 1.4.2), the use of lattice structure leads to a computationally efficient algorithm known as *recursive least-squares (RLS) lattice*. A derivation of this algorithm is presented in Chapter 13.

The FIR and IIR filters which were discussed above are classified as linear filters because their outputs are obtained as linear combinations of the present and past samples of input and, in the case of IIR filter, the past samples of the output also. Although most applications are restricted to the use of linear filters, nonlinear adaptive filters become necessary in some applications where the underlying physical phenomena to be modeled are far from being linear. A typical example is magnetic recording where the recording channel becomes nonlinear at high densities because of the interaction among 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

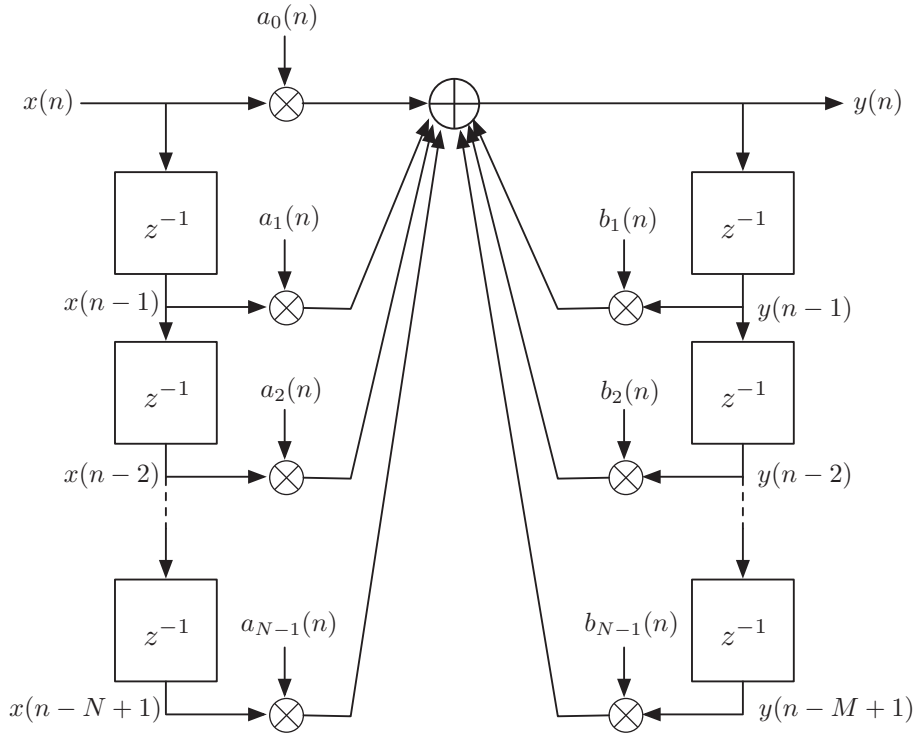


Figure 1.4 The structure of an IIR filter.

its input, $x(n)$, according to the equation

$$\begin{aligned} y(n) = w_{0,0}(n) &+ \sum_i w_{1,i}(n)x(n-i) \\ &+ \sum_{i,j} w_{2,i,j}(n)x(n-i)x(n-j) \\ &+ \sum_{i,j,k} w_{3,i,j,k}(n)x(n-i)x(n-j)x(n-k) + \ldots \end{aligned} \tag{1.4}$$

where $w_{0,0}(n)$, $w_{1,i}(n)$'s, $w_{2,i,j}(n)$'s, $w_{1,i,j,k}(n)$'s, ... are filter coefficients. In this book, we do not discuss the Volterra filters any further. However, we note that all the summations in Eq. (1.4) 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, viz. stochastic and deterministic. Both these 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 is obtained by minimization of its MSE. As was noted before, 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 behavior is highly dependent on the power spectral density of the filter input. When the filter input is white, that is, its power spectrum is flat across the whole range of frequencies, the LMS algorithm converges very fast. However, when certain bands of frequencies are not well excited (i.e., the signal energy in those bands is relatively low), some slow modes of convergence appear, thus resulting in very slow convergence compared to 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 that effectively divide the frequency band of the input signal into a number of subbands and achieve some degree of signal whitening using some power normalization mechanism before applying the adaptive algorithm. These algorithms which appear in different forms are presented in Chapters 7, 9, and 11.

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 the time domain convolutions in the frequency domain, as is commonly done in the implementation of long digital filters (Oppenheim and Schafer, 1975, 1989). Adaptive algorithms which use FFT for reducing computational complexity are presented in Chapter 8.

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 mentioned before, in the Wiener filter theory, the desired filter is obtained by minimizing the MSE, that is, a statistical quantity. In the method of least-squares, on the other hand, the performance index is the *sum of weighted error squares* for the given data, that is, a deterministic quantity. A consequence of this deterministic approach (that will become clear as we go through its derivation in Chapter 12) 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 block-by-block basis to the incoming signals. This, which is referred to as *block estimation of the least-squares method*, has some useful applications in areas such as linear predictive coding (LPC) 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 because of the reasons that were given in Section 1.2. There are three major classes of RLS adaptive filtering algorithms and are as follows:

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.

QR-decomposition-based RLS (QRD-RLS) algorithm. This formulation of RLS algorithm also involves matrix manipulations, which leads 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 compared to other types of RLS algorithms (Haykin, 1991, 1996)

Fast RLS algorithms. In the case of transversal filters, the tap inputs are successive samples of input signal, $x(n)$ (Figure 1.1). 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:

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.

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 that require careful attention to prevent undesirable behavior in practice.

In this book, we present a complete treatment of the various LMS-based algorithms in seven chapters. However, our discussion on RLS algorithms is rather limited. We present a comprehensive treatment of the properties of the method of least-squares and a derivation of the standard RLS algorithm in Chapter 12. The basic results related to the development of fast RLS algorithms and some examples of such algorithms are presented in Chapter 13. A study of the *tracking behavior* of selected adaptive filtering algorithms is presented in Chapter 14 of this book. The use of these algorithms to various applications are discussed in Chapters 15 through 20.

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 signaling techniques are phase shift keying (PSK) 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. Moreover, in the case of frequency domain implementation of adaptive filters (Chapter 8) and subband adaptive filters (Chapter 9), we will be dealing with complex-valued signals, even though the original signals may be real-valued. Thus, we find cases where the formulation of the adaptive filtering algorithms must be given in terms of complex-valued variables.

In this book, to keep our presentation as simple as possible, most of the derivations are given for real-valued signals. However, wherever we find it necessary, the extensions to complex forms will also be followed.

1.6 Applications

Adaptive filters by their very nature are self-designing systems that 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 (ANC), biomedical engineering, and so on. 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 to the adaptive filtering algorithm such that the difference between the filter output and the desired response is minimized in either statistical or deterministic sense. In this context, four basic classes of adaptive filtering applications are recognized. Namely, modeling, inverse modeling, linear prediction, and interference cancellation. In the rest of this chapter, we present an overview of these applications.

1.6.1 Modeling

Figure 1.5 depicts the problem of modeling 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

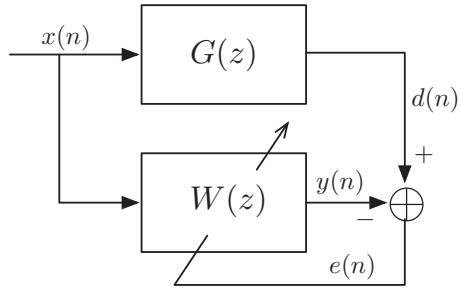


Figure 1.5 Adaptive system modeling.

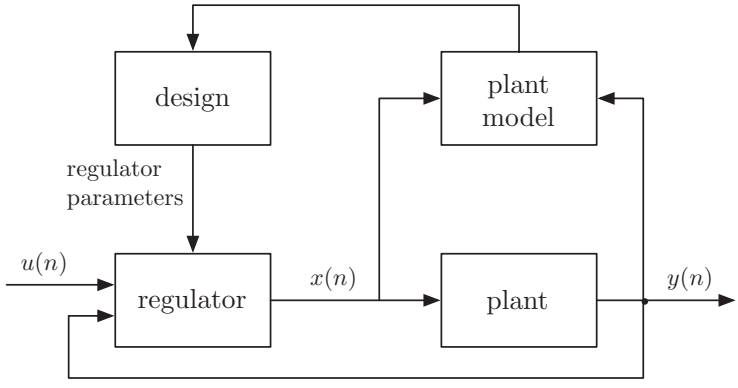


Figure 1.6 Block diagram of a self-tuning regulator.

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 by an adaptive filtering algorithm such that the difference between the plant output, $d(n)$, and the adaptive filter output, $y(n)$, is minimized.

An application of modeling, 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 (see e.g., Astrom and Wittenmark (1980)).

Another application of modeling 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 under the topic of interference cancellation.

Nonideal 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,

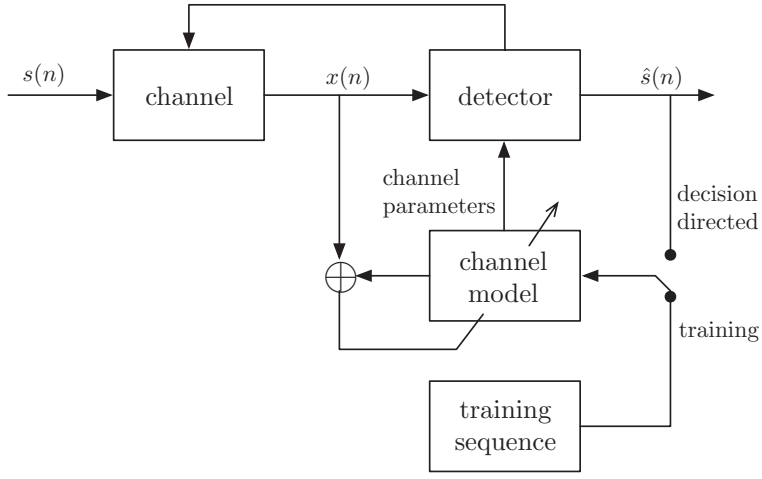


Figure 1.7 An adaptive data receiver using channel identification.

is discussed in the following under the topic of inverse modeling. Direct modeling 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 (Proakis, 1995). Furthermore, computation of equalizer coefficients from channel response has been proposed by some researchers because this technique has been found to result in better tracking of time-varying channels (Fechtel and Meyr (1991) and Farhang-Boroujeny and Wang (1995)). 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 setup, as shown in Figure 1.7 to identify the channel. Once the channel is identified and the normal mode of transmission begins, the detected data symbols, $\hat{s}(n)$, are used as input to the channel model and the adaptation process continues for tracking possible variations of the channel. This is known as *decision-directed mode* and is also shown in Figure 1.7.

1.6.2 Inverse Modeling

Inverse modeling, also known as *deconvolution*, is another application of adaptive filters that has found extensive use in various engineering disciplines. The most widely used application of inverse modeling is in communications where an inverse model (also called *equalizer*) is used to mitigate the channel distortion. The concept of inverse modeling 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 (Widrow and Stearns, 1985). The process of prediction, which is explained later, may also be viewed as an inverse modeling scheme (Section 1.6.3). In this section, we concentrate on the application of inverse modeling in channel equalization. The full treatment of the subject of channel equalization is presented in Chapter 17.

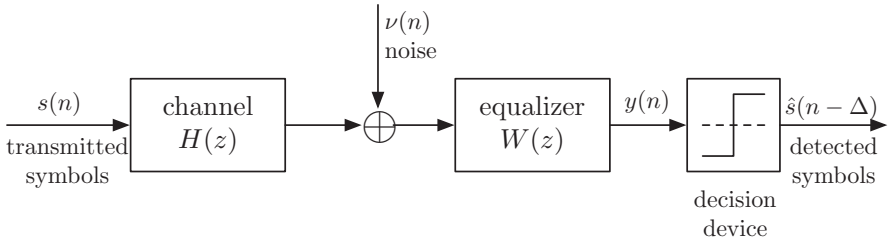


Figure 1.8 A baseband data transmission system with channel equalizer.

Channel Equalization

Figure 1.8 depicts the block diagram of a baseband 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, $\nu(n)$, arises from thermal noise in the electronic circuits and possible crosstalks from neighboring channels. The transmitted data symbols, $s(n)$, which 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, and instead a response which is nonzero over many symbol periods. This distortion results in interference of neighboring data symbols with one another, thereby making the detection process through a simple threshold detector unreliable. The phenomenon of interference among neighboring data symbols is known as *intersymbol interference* (ISI). The presence of the additive noise samples, $\nu(n)$, 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, the task of equalizer would be rather straightforward. For a channel $H(z)$, an equalizer with transfer function $W(z) = 1/H(z)$ 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, that is, $1/H(z)$, may be noncausal 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 $H(z)W(z) \approx z^{-\Delta}$, 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. Example 3.4 of Chapter 3 clarifies the concept of noncausality of $1/H(z)$ and also the way the problem is (approximately) solved by introducing a delay, Δ. Greater details appear in Chapter 17.

We also note that the choice of $W(z) = 1/H(z)$ (or $W(z) \approx z^{-\Delta}/H(z)$) may lead to a significant enhancement of the additive noise, $\nu(n)$, in those frequency bands where the magnitude of $H(z)$ is small (i.e., $1/H(z)$ is large). Hence, in choosing an equalizer, $W(z)$, one should keep a balance between residual ISI and noise enhancement at the equalizer output. Wiener filter is a solution with such a balance (Chapter 3, Section 3.6.4).

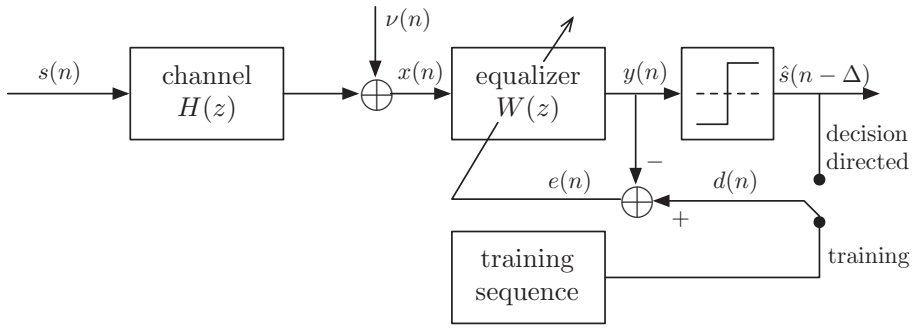


Figure 1.9 Details of a baseband data transmission system equipped with an adaptive channel equalizer.

Figure 1.9 presents the details of a baseband 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 as they (to be more accurate, a delayed replica of them) 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[1] should comply with these so that the modems of different manufacturers can communicate with one another.

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 a probability close to 1. Hence, then onward, the detected symbols can be treated as the desired signal for further adaptation of the equalizer so that possible variations of 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 modeling discussed previously defines the equalizer as an approximation of $z^{-\Delta}/H(z)$, that is, the target/desired response of the cascade of channel and equalizer is $z^{-\Delta}$, a pure delay. This can be generalized by replacing the target response $z^{-\Delta}$ by a general target response, say $\Gamma(z)$. 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 which incorporate such nontrivial target responses are referred to as *partial-response signaling 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 $\Gamma(z)$, instead of the trivial memoryless channel $z^{-\Delta}$. The Viterbi detector

[1] The term *modem* which is the abbreviation for "modulator and demodulator" is commonly used to refer data transceivers (transmitter and receiver).

(Proakis, 1995) is an example for such a detector. The target response, $\Gamma(z)$, is selected so that its magnitude response approximately matches the channel response, that is, $|\Gamma(e^{j\omega})| \approx |H(e^{j\omega})|$, 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 1, thereby minimizing the noise enhancement. To clarify further on this and also to mention another application of inverse modeling, we discuss the problem of magnetic recording next.

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 side of the line. The data bits, which are converted to signal pulses before recording, undergo some distortion because of nonperfect behavior of the head and medium, as it happens in communication channels because of the nonideal response of the channel. Additive thermal noise and interference from neighboring recording tracks (just like neighboring channels in communications) are also present in the magnetic recording channels (Bergmans, 1996).

Magnetic recording channels are usually characterized by their response to an isolated pulse of width 1-bit interval, T. This is known as *dibit response* and in the case of hard-disk channels, it is usually modeled by the superposition of a positive and negative *Lorentzian* pulses, separated by 1-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 $g_a(t)$ and other functions that appear in the rest of this subsection are to emphasize that they are analog (nonsampled) signals. The ratio $D = t_{50}/T$ is known as the *recording density*. Typical values of D are in the range of 1 to 3. A higher density means more bits are contained in one t_{50} interval, that is, more ISI. We may also note that t_{50} is a temporal measure of the recording density. When measured spatially, we obtain another parameter $pw_{50} = t_{50}/v$, where v is the velocity of the medium with respect to 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 Eq. (1.5), the dibit response of a hard-disk channel is obtained as

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

The response of the channel to a sequence $s(n)$ of data bits is then given by the convolution sum

$$u_a(t) = \sum_n s(n) h_a(t - nT) \tag{1.7}$$

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

Figure 1.10a and b shows 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 1.10b, 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, $\Gamma(z)$, of the equalizer so that it resembles a bandpass filter whose bandwidth and magnitude response are close to those 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

$$\Gamma(z) = z^{-\Delta}(1 + z^{-1})^K(1 - z^{-1}) \tag{1.8}$$

where Δ, as before, is an integer delay and K is an integer greater than or equal to 1. 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. In Chapter 10, we elaborate on these aspects of partial-response systems.

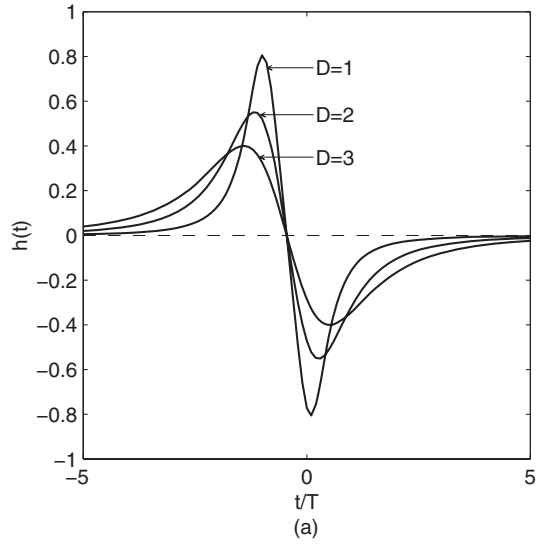
1.6.3 Linear Prediction

Prediction is a spectral estimation technique that is used for modeling 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, $\tilde{x}(n)$, is assumed to be generated by exciting the filter $G(z)$ with the input $u(n)$. As $G(z)$ is an all-pole filter, this is known as *autoregressive* (AR) modeling. The choice/type of the excitation signal, $u(n)$, is application dependent and may vary depending on the nature of the process being modeled. 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 other two.

The rationale behind the use of AR modeling may be explained as follows. As the samples of any given nonwhite 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 shown 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 certain value, which depends on the extent of 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

$$H(z) = 1 - \sum_{i=1}^{N} a_i z^{-i} \tag{1.9}$$



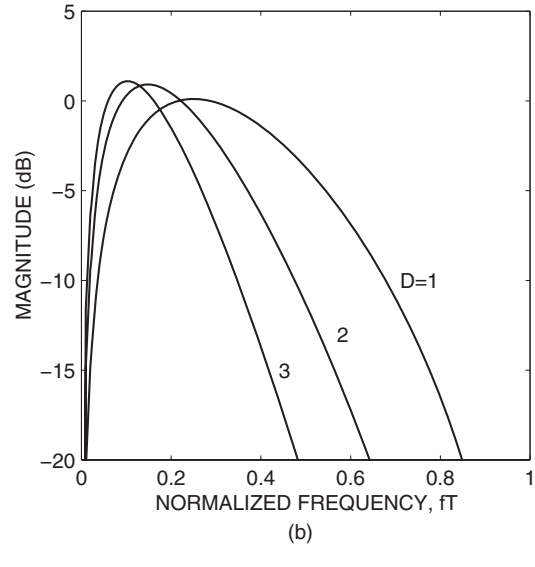


Figure 1.10 Time and frequency domain responses of magnetic recording channels for densities $D = 1$, 2, and 3 modeled using the Lorentzian pulse: (a) dibit response; (b) magnitude response of dibit response.

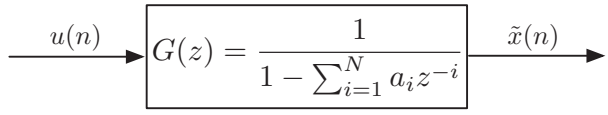


Figure 1.11 Autoregressive modeling of a random process.

where a_i's 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

$$G(z) = \frac{1}{1 - \sum_{i=1}^{N} a_i z^{-i}} \tag{1.10}$$

as shown in Figure 1.11, the generated output, $\tilde{x}(n)$, 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.

Autoregressive Spectral Analysis

In certain applications, we need to estimate the power spectrum of a random process. A trivial way of obtaining such 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 *nonparametric spectral estimation techniques* (Kay, 1988). When the number of samples of the input is limited, the estimates provided by nonparametric 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 using either AR, MA, or ARMA models (Kay, 1988). In the case of AR modeling, 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, a_i's, are optimized by minimizing the prediction error, $e(n)$. Once the predictor coefficients have

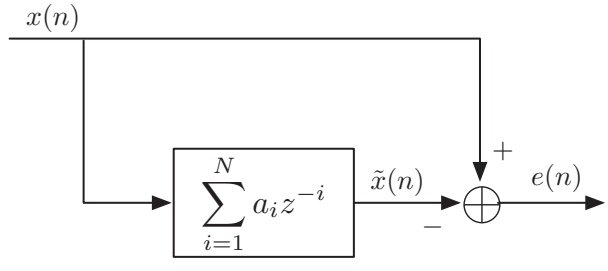


Figure 1.12 Linear predictor.

converged, an estimate of the power spectral density of $x(n)$ is obtained according to the following equation:

$$\Phi_{xx}(e^{j\omega}) = N_o \left| \frac{1}{1 - \sum_{i=1}^{N} a_i e^{-j\omega i}} \right|^2 \tag{1.11}$$

where N_o 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 the convergence of the predictor, $e(n)$ is approximately white. For further explanation on the derivation of Eq. (1.11) from the signal model of Figure 1.11, refer to Chapter 2 (Section 2.4.4).

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 a narrow-band and wide-band processes is applied to a predictor, the predictor output, $\hat{x}(n)$, 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*. Examples of line enhancers can be found in Chapters 6 and 10. In particular, in Chapter 10, we find that line enhancers can be best implemented using IIR filters.

We also note that in the applications where the narrow-band portion of $x(n)$ has to be rejected (such as the examples mentioned above), the difference between $x(n)$ and $\hat{x}(n)$, that is, 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.

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 (Jayant and Noll, 1984). Both these 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

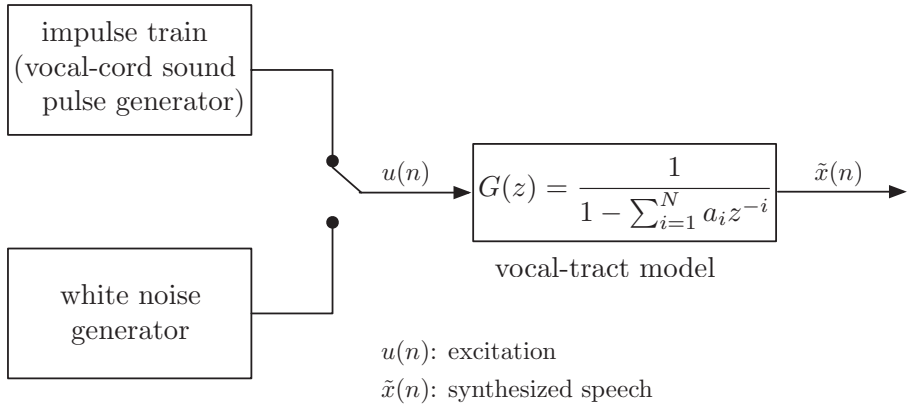


Figure 1.13 Speech-production model.

technique, which is categorized under the class of *source coders*, strives to produce digitized voice data at low bit rates in the range of 2 to 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).

The main reason for linear prediction being widely used in speech coding is that speech signals can be accurately modeled as shown 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 *pitch period*, and the power of the white noise, known as *excitation level*, are parameters of the speech model which are to be identified in the coding process.

Linear Predictive Coding (LPC)

Speech signal is a highly nonstationary 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, a_i's, 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) predictor coefficients, (ii) energy of excitation signal, (iii) voiced/unvoiced classification, and (iv) 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.

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 into a prescribed number of bits to generate the information bits associated with the coded speech. Direct quantization of speech samples requires relatively 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 much smaller variance than the input, $x(n)$. Thus, for a given quantization level, $e(n)$ may be quantized with less number of bits compared to $x(n)$. Moreover, as the number of information bits per every second of the coded speech is directly proportional to the number of bits used per sample, bit rate of the DPCM will be less compared to 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 *adaptive DPCM* (ADPCM). In the case of speech signals, use of ADPCM results in superior performance compared to the case where a nonadaptive DPCM is used. In fact, the ADPCM has been standardized and widely used in practice (ITU Recommendation G.726).

Figure 1.14 depicts a simplified diagram of the ADPCM system, as proposed in ITU Recommendation G.726.[2]. 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 $\tilde{e}(n)$ is minimized in mean-square sense. The predictor input $\tilde{x}(n)$ is same as the original input $x(n)$ except for the quantization error in $\tilde{e}(n)$. To understand the joint operation of the encoder and decoder shown in Figure 1.14, note that the same signal, $\tilde{e}(n)$, 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, that is, $\tilde{x}'(n)$, will be equal to that at the encoder, that is, $\tilde{x}(n)$, as nonequal initial conditions of the encoder and decoder loops will die away after their transient phase.

1.6.4 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 interfering

[2] ITU stands for International Telecommunication Union

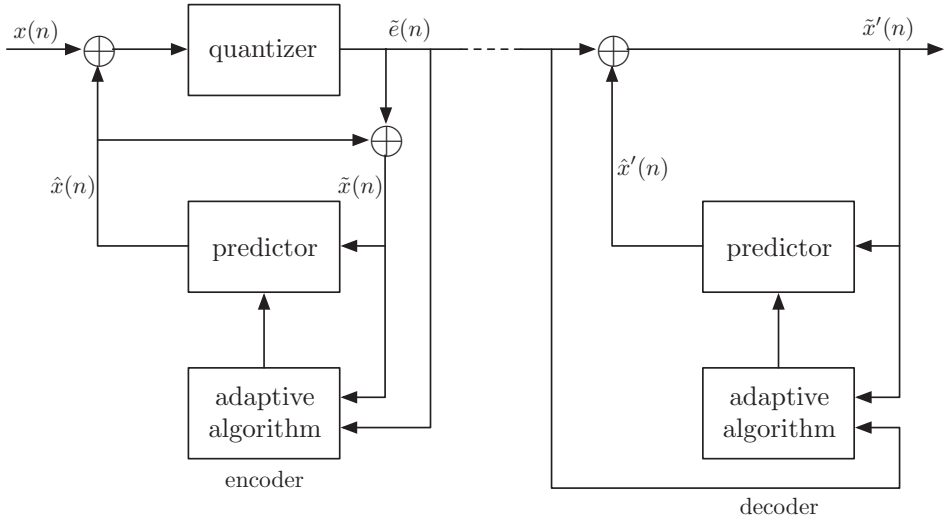


Figure 1.14 ADPCM encoder–decoder.

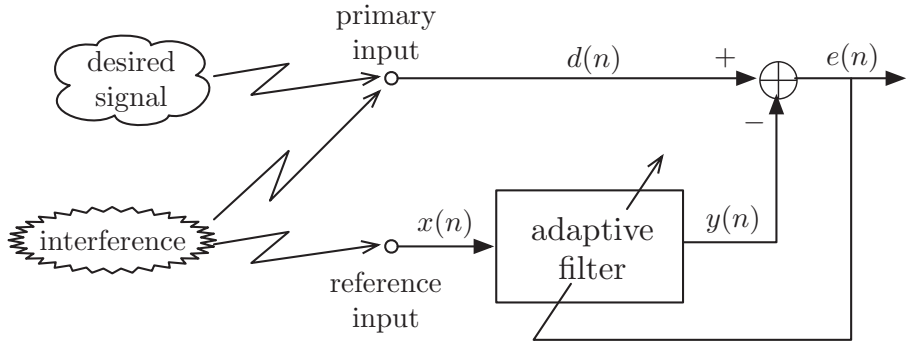


Figure 1.15 Interference cancellation.

signal and subtract that from the corrupted signal. 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 canceler: *primary* and *reference*. The primary input is the corrupted signal, that is, the desired signal plus interference. The reference input, on the other hand, originates from the interference source only.[3] 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 which is cleared from interference, thus the name interference cancellation.

[3] In some applications of interference cancellation, there might also be some leakage of the desired signal to the reference input. Here, we have ignored this situation for simplicity.

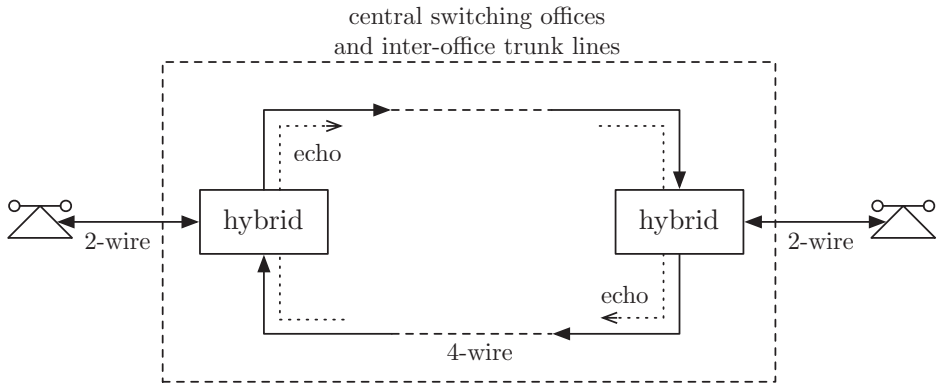


Figure 1.16 Simplified diagram of a telephone network.

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 the adaptive filter input is clearly related to the role of this input in the canceler.

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

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 ones. 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 some portions of the local network. The four wires, on the 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 segment signals in the two directions are transmitted on two separate lines. Accordingly, the role of hybrid circuit is to separate the signals in the two directions. Perfect operation of the hybrid circuit requires that the incoming 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 behavior 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), 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 was noted before and also can clearly be seen from Figure 1.17, the problem of echo cancellation may be viewed as one of system modeling. An adaptive filter is put

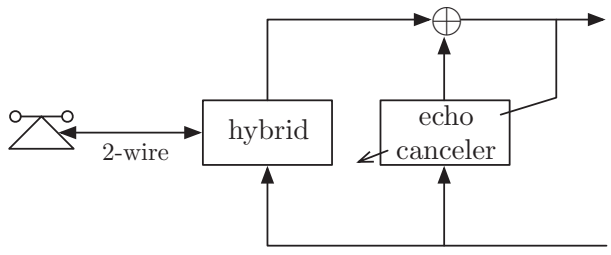


Figure 1.17 Adaptive echo canceler.

between the incoming and outgoing 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 outgoing signal to clear that from the undesirable echo.

Echo cancelers are usually implemented in transversal form. The time spread of echoes in a typical hybrid circuit is in the range of 20–30 ms. If we assume a sampling rate of 8 kHz for the operation of the echo canceler, an echo spread of 30 ms requires an adaptive filter with at least 240 taps (30 ms$\times$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 subject of frequency domain adaptive filters is covered in Chapter 8.

The echo cancelers described previously 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 canceler has to be added, as indicated in Figure 1.18. However, we note that the data echo cancelers are different from the voice echo cancelers used in central switching offices in many ways. For instance, because the input to the data echo canceler are data symbols, it can operate at the data symbol rate that is in the range of 2.4–3 kHz (about three times smaller than the 8 kHz sampling frequency used in voice echo cancelers). For a given echo spread, a lower sampling frequency implies less number of taps for the echo canceler. Clearly, this simplifies the implementation of the echo canceler, greatly. On the other hand, the data echo cancelers require to achieve a much higher level of echo cancellation to ensure 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. Detailed discussions on these issues can be found in Lee and Messerschmitt (1994) and Gitlin, Hayes, and Weinstein (1992).

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. The speech signal from a far-end speaker, received through a communication channel, is broadcast by a loudspeaker in a room and its echo is picked up by a microphone.

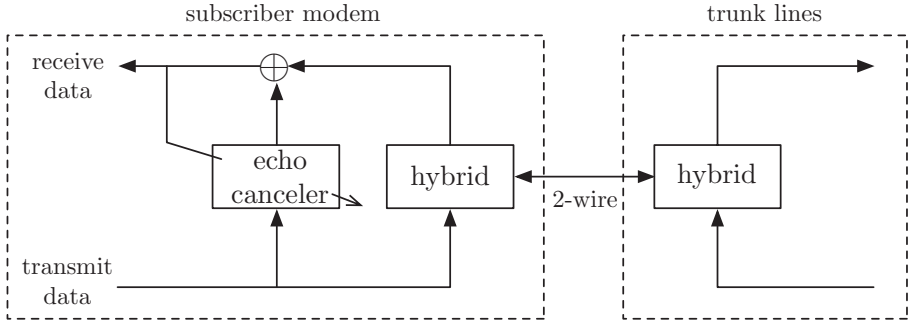


Figure 1.18 Data echo canceler.

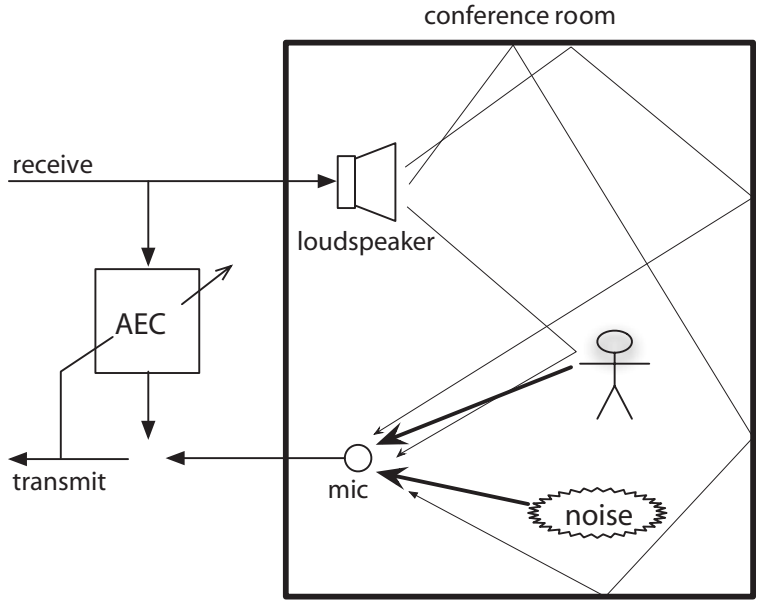


Figure 1.19 Acoustic echo cancellation.

This echo must be canceled 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 acoustics of the room. A replica of the loudspeaker echo is then obtained and subtracted from the microphone signal before the transmission.

Clearly, the problem of acoustic echo cancellation can also be posed as one of system modeling. 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 of 100–250 ms spread is quite common. For a sampling rate of 8 kHz, this would mean 800–2000 taps! Thus, the main problem of acoustic echo cancellation is that of realizing very long adaptive filters. In addition, as speech is a lowpass signal, it becomes necessary to use special algorithms

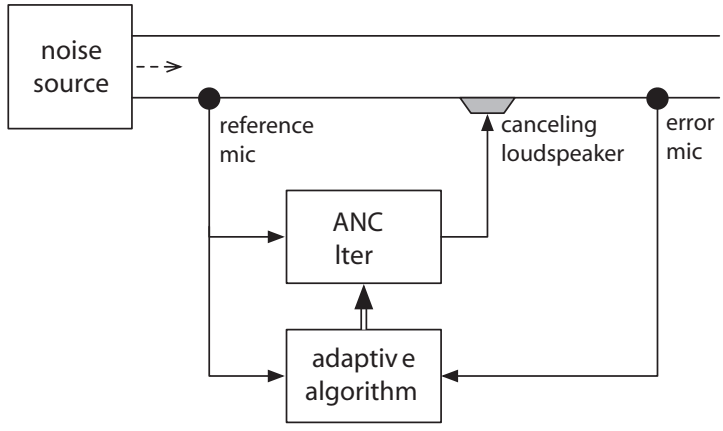


Figure 1.20 Active noise cancellation in a narrow duct.

to ensure fast adaptation of the echo canceler. The algorithms discussed in Chapters 8 and 9 have been widely used to overcome these difficulties in the implementation of acoustic echo cancelers. The topic of echo cancelers, with particular emphasis on acoustic echo cancelers, is covered in Chapter 15.

Active Noise Control

ANC refers to situations where acoustic *antinoise* waves are generated from electronic circuits (Kuo and Morgan, 1996). The ANC can be best explained by the following example.

A well-examined application of ANC is cancellation of noise in narrow ducts, such as exhaust pipes and ventilation systems, as illustrated in Figure 1.20. The acoustic noise traveling along the duct is picked up by a microphone at position A. 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 canceling loudspeaker), is equal to the negative value of the duct noise at position B, thereby canceling 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 setup with the interference cancellation setup shown in Figure 1.15, we may note the following. The source of interference here is the duct noise, reference input is the noise picked up by the reference microphone, desired output (i.e., what we wish to see after canceling the duct noise) is zero, and primary input is the duct noise reaching position B. Accordingly, the role of 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 traveling along the duct, which is like a one-dimensional model. The acoustical models of wider ducts and large enclosures, such as cars and aircrafts, are usually more complicated. Multiple microphones/loudspeakers are needed for successful implementation of ANCs in such enclosures. The adaptive filtering problem is then that of a multiple-input multiple-output system (Kuo and Morgan, 1996). Nevertheless, the basic principle remains the same, that is, generation of antinoise to cancel the actual noise. The subject of active noise control is covered in detail in Chapter 16.

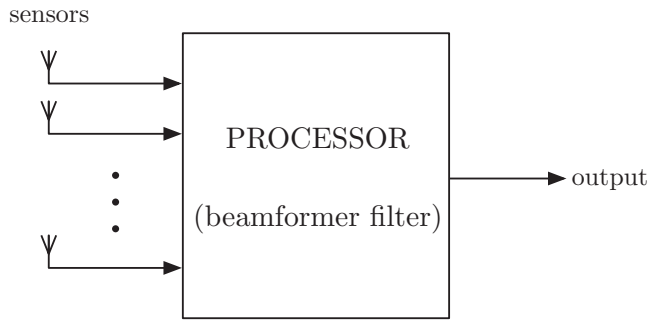


Figure 1.21 Spatial filtering (beamforming).

Beamforming

In the applications that have been discussed so far, the filters/predictors are used to combine together 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 (Johnson and Dudgeon, 1993), and also imaging in radar and medical engineering (Soumekh, 1994).

In spatial filtering, a number of independent sensors are placed at different points in space to pick up signals coming from various sources (Figure 1.21). In radar and communications, the signals are usually electromagnetic waves and the sensors are thus antenna elements. Accordingly, the term *antenna arrays* 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 samples of the 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 sensor signals so that signals arriving from some particular direction 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 incoming signal(s). This is given in the form of a polar plot (gain versus angle) and is referred to as *beam pattern*.

In many applications of beamformers, the signals picked up by sensors are narrow bands having the same carrier (center) frequency. These signals differ in their direction-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 omnidirectional elements A and B, as presented in Figure 1.22. The tone (as approximation to narrow-band) signals $s(n) = \alpha \cos \omega_o n$ and $v(n) = \beta \cos \omega_o n$ arriving at angles 0 and θ_o (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 $s(n)$ arrives at elements A and B at the same time, whereas the arrival times of signal $v(n)$ at

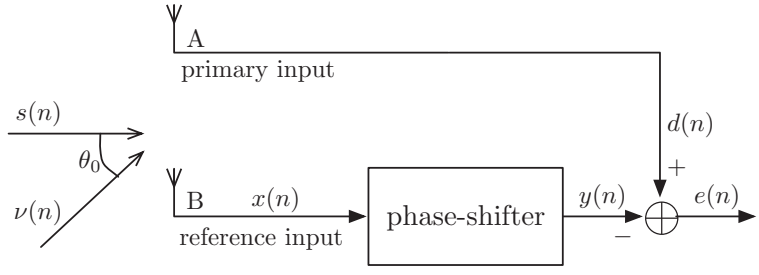


Figure 1.22 A two-element beamformer.

A and B are different. We may thus write

$$s_A(n) = s_B(n) = \alpha \cos \omega_o n$$

$$\nu_B(n) = \beta \cos \omega_o n$$

and

$$\nu_A(n) = \beta \cos(\omega_o n - \varphi)$$

where subscripts A and B are used to denote the signals picked up by elements A and B, respectively, φ is the phase shift arising from the time delay of arrival of $\nu(n)$ at element A with respect to its arrival at element B.

Now, if we assume that $s(n)$ is the desired signal and $\nu(n)$ is an interference, by inspection, one can see that if the phase-shifter phase is chosen equal to φ, then the interference, $\nu(n)$, will be completely canceled by the beamformer. The desired signal, on the other hand, reaches the beamformer output as $\alpha(\cos \omega_o n - \cos(\omega_o n - \varphi))$, which is nonzero (and still holding the information contained in its envelope, α) when $\varphi \neq 0$, that is, when the interference direction is different from the direction of the desired signal. This shows that one can tune a beamformer so as 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, one can realize any arbitrary beam pattern. This is similar to the selection of tap weights of 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 last 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 previously. The temporal information is obtained using a transversal filter at the output of each sensor. The output of the broadband beamformer is the summation of the outputs of these transversal filters. Detailed discussions on these points and the relevant mathematical backgrounds are presented in Chapter 18.