Chapter 1. Background and WLAN Overview

IN THIS CHAPTER

- Review of Stochastic Processes and Random Variables
- Review of Discrete-Time Signal Processing
- Components of a Digital Communication System
- OFDM WLAN Overview
- Single Carrier Versus OFDM Comparison
- Bibliography

Before delving into the details of orthogonal frequency division multiplexing (OFDM), relevant background material must be presented first. The purpose of this chapter is to provide the necessary building blocks for the development of OFDM principles. Included in this chapter are reviews of stochastic and random process, discrete-time signals and systems, and the Discrete Fourier Transform (DFT). Tooled with the necessary mathematical foundation, we proceed with an overview of digital communication systems and OFDM communication systems. We conclude the chapter with summaries of the OFDM wireless LAN standards currently in existence and a high-level comparison of single carrier systems versus OFDM.

The main objective of a communication system is to convey information over a channel. The subject of digital communications involves the transmission of information in digital form from one location to another. The attractiveness of digital communications is the ease with which digital signals are recovered as compared to their analog counterparts. Analog signals are continuous-time waveforms and any amount of noise introduced into the signal bandwidth can not be removed by amplification or filtering. In contrast, digital signals are generated from a finite set of discrete values; even when noise is present with the signal, it is possible to reliably recover the information bit stream exactly. In the sections to follow, brief reviews of stochastic random processes and discrete-time signal processing are given to facilitate presentation of concepts introduced later.

Review of Stochastic Processes and Random Variables

The necessity for reviewing the subject of random processes in this text is that many digital communication signals [20, 21, 22, 25] can be characterized by a *random or stochastic process*. In general, a signal can be broadly classified as either *deterministic* or *random*. Deterministic signals or waveforms can be known precisely at instant of time, usually expressed as a mathematical function of time. In constrast, random signals or waveforms always possess a measure of uncertainty in their values at any instant in time since random variables are rules for assigning a real number for every outcome ξ of a probabilistic event. In other words, deterministic signals can be reproduced exactly with repeated measurements and random signals cannot.

A stochastic or random process is a rule of correspondence for assigning to every outcome ξ to a function $X(t,\xi)$ where t denotes time. In other words, a stochastic process is a family of time-functions that depends on the parameter ξ. When random variables are observed over very long periods, certain regularities in their behavior are exhibited. These behaviors are generally described in terms of probabilities and statistical averages such as the mean, variance, and correlation. Properties of the averages, such as the notion of stationarity and ergodicity, are briefly introduced in this section.

Random Variables

A random variable is a mapping between a discrete or continuous random event and a real number. The *distribution function,* $F_X(\alpha)$, of the random variable, *X,* is given by

Equation 1.1

$$F_X(\alpha) = \Pr(X \leq \alpha)$$

where $\Pr(X \leq \alpha)$ is the probability that the value taken on by the random variable *X* is less than or equal to a real number α. The distribution function $F_X(\alpha)$ has the following properties:

- $0 \leq F_X(\alpha) \leq 1$
- $F_X(\alpha) \leq F_X(\beta)$ if $\alpha \leq \beta$
- $F_X(-\infty) = 0$
- $F_X(+\infty) = 1$

Another useful statistical characterization of a random variable is the *probability density function* (pdf), $f_X(\alpha)$, defined as

Equation 1.2

$$f_X(\alpha) = \frac{\partial}{\partial \alpha} F_X(\alpha)$$

Based on properties of $F_X(\alpha)$ and noting the relationship in Equation 1.2, the following properties of the pdf easily deducted:

- $f_X(\alpha) \geq 0$

- $$\int_{-\infty}^{\infty} f_X(\alpha)\partial\alpha = F_X(+\infty) - F_X(-\infty) = 1$$

Thus, the pdf is always a nonnegative function with unit area.

Ensemble Averages

In practice, complete statistical characterization of a random variable is rarely available. In many applications, however, the average or *expected value* behavior of a random variable is sufficient. In latter chapters of this book, emphasis is placed on the expected value of a random variable or function of a random variable. The mean or expected value of a continuous random variable is defined as

Equation 1.3

$$m_X = E\{X\} = \int_{-\infty}^{\infty} \alpha f_X(\alpha)\partial\alpha$$

and a discrete random variable as

Equation 1.4

$$m_X = E\{X\} = \sum_k \alpha_k \Pr(X = \alpha_k)$$

where $E\{\cdot\}$ is called the expected value operator. A very important quantity in communication systems is the *mean squared value* of a random variable, *X,* which is defined as

Equation 1.5

$$E\left\{X^2\right\} = \int_{-\infty}^{\infty} \alpha^2 f_X(\alpha)\partial\alpha$$

for continuous random variables and

Equation 1.6

$$E\left\{X^2\right\} = \sum_k \alpha_k^2 \Pr(X = \alpha_k)$$

for discrete random variables. The mean squared value of a random variables is a measure of the average power of a random variable. The *variance* of X is the mean of the second central moment and defined as

Equation 1.7

$$\sigma_X^2 = E\left\{(X - m_X)^2\right\} = \int_{-\infty}^{\infty} (\alpha - m_X)^2 f_X(\alpha)\partial\alpha$$

Note a similar definition holds for the variance of discrete random variables by replacing the integral with a summation. The variance is a measure of the "random" spread of the random variable *X*. Another well-cited characteristic of a random *X* is its *standard deviation* σ_X, which is defined as the square root of its variance. One point worth noting is the relationship between the variance and mean square value of a random variable, i.e.,

Equation 1.8

$$\begin{aligned}\sigma_X^2 &= E\left\{X^2 - 2Xm_X + m_X^2\right\} \\ &= E\left\{X^2\right\} - 2E\{X\}m_X + m_X^2 \\ &= E\left\{X^2\right\} - m_X^2\end{aligned}$$

In view of Equation 1.8, the variance is simply the difference between the mean square value and the square of the mean.

Two additional ensemble averages importantance in the study of random variables are the correlation and covariance. Both quantities are expressions of the interdependence of two or more random variables to each other. The correlation between complex random variables *X* and *Y,* r_{XY}, is defined as

Equation 1.9

$$r_{XY} = E\left\{XY^*\right\}$$

where * denotes the complex conjugate of the complex random variable. A closely related quantity to the correlation of between random variables is their covariance c_{XY}, which it is defined as

Equation 1.10

$$c_{XY} = E\left\{[X - m_X][Y - m_Y]^*\right\} = E\left\{XY^*\right\} - m_X m_Y^*$$

Clearly, if either X or Y has zero mean, the covariance is equal to the correlation. The random variables X and Y need not stem from separate probabilistic events; in fact, X and Y can be samples of the same event A observed at two different time instants t_1 and t_2. For this situation, the correlation r_{XY} and covariance c_{XY} become the autocorrelation $R_X(t_1,t_2)$ and autocovariance $C_X(t_1,t_2)$, respectively, which are defined as

Equation 1.11

$$R_X(t_1,t_2) = E\left\{X(t_1)X^*(t_2)\right\}$$
$$C_X(t_1,t_2) = E\left\{[X(t_1) - m_X(t_1)][X(t_2) - m_X(t_2)]^*\right\}$$

Thus, the autocorrelation and autocovariance are measures of the degree to which two time samples of the same random process are related.

There are many examples of random variables that arise in which one random variable does not depend on the value of another. Such random variables are said to be statistically independent. A more precise expression of the meaning of statistical independence is given in the following definition.

Definition 1 *Two random variables X and Y are said to be statistically independent if the joint probability density function is equal to the product of the individual pdfs, i.e.,*

Equation 1.12

$$f_{XY}(\alpha,\beta) = f_X(\alpha)f_Y(\beta)$$

A weaker form of independence occurs when the correlation r_{XY} between two random variable is equal to the product of their means, i.e.,

Equation 1.13

$$r_{XY} = E\left\{XY^*\right\} = m_X m_Y^*$$

Two random variables that satisfy Equation 1.13 are said to be *uncorrelated*. Note that since

Equation 1.14

$$c_{XY} = r_{XY} - m_X m_Y^*$$

then two random variables X and Y will be uncorrelated if their covariance is zero. Note, statistically independent random variables are always uncorrelated. The converse, however, is usually not true in general.

Up to this point, most of the discussions have focused on random variables. In this section, we focus on random processess. Previously, we stated that a random process is a rule of correspondence for assigning to every outcome ξ of a probabilistic event to a function X(t, ξ). A collection of X(t, ξ) resulting from many outcomes defines an ensemble for X(t, ξ). Another, more useful, definition for a random process is an indexed sequence of random variables. A random processes is said to be *stationary* in the *strict-sense* if none of its statistics are affected by a shift in time origin. In other words, the statistics depend on the length of time it is observed and not when it is started. Furthermore, a random process is said to be *wide-sense stationary* (WSS) if its mean and variance do not vary with a shift in the time origin, i.e.,

$$m_X = E\{X(k)\} = a\ constant, \quad \forall k$$

and

$$R_X = (\tau + k, k) = R_X(\tau)$$

Strict-sense stationarity implies wide-sense stationary, but not vice versa. Most random processes in communication theory are assumed WSS. From a practical view, it is not necessary for a random process to be stationary for all time, but only for some observation interval of interest. Note that the autocorrelation function for a WSS process depends only on time difference τ. For zero mean WSS processes, $R_X(\tau)$ indicates the time over which samples of the random process X are correlated. The autocorrelation of WSS processes has the following properties:

- $R_X(\tau) = R_X(-\tau)$
- $R_X(\tau) \leq R_X(0)$ for all τ
- $R_X(0) = E\{X^2(t)\}$

Unfortunately, computing m_X and $R_X(\tau)$ by ensemble averaging requires knowledge of a collection realizations of the random process, which is not normally available. Therefore, time averages from a single realization are generally used. Random processes whose time averages equal their ensemble averages are known as *ergodic processes*.

Review of Discrete-Time Signal Processing

In this brief overview of discrete-time signal processing, emphasis is placed on the specification and characterization of discrete-time signals and discrete-time systems. The review of stochastic processes and random variables was useful to model most digital communication signals. A review of linear discrete-time signal processing, on the other hand, is needed to model the effects of the channel on digital communication signals. Digital-time signal processing is a vast and well-documented area of engineering. For interested readers, there are several excellent texts [7, 10, 18] that give a more rigorous treatment of discrete-time signal processing to supplement the material given in this section.

Discrete-Time Signals

A discrete-time signal is simply an indexed sequence of real or complex numbers. Hence, a random process is also a discrete-time signal. Many discrete-time signals arise from sampling a continuous-time signal, such as video or speech, with an analog-to-digital (A/D) converter. We refer interested readers to "Discrete-Time Signals" for further details on A/D converters and sampled continuous-time signals. Other discrete-time signals are considered to occur naturally such as time of arrival of employees to work, the number of cars on a freeway at an instant of time, and population statistics. For most information-bearing signals of practical interest, three simple yet important discrete-time signals are used frequently to described them. These are the unit sample, unit step, and the complex exponential. The *unit sample,* denoted by δ(n), is defined as

Equation 1.15

$$\delta(n) = \begin{cases} 1, & n = 0 \\ 0, & n \neq 0 \end{cases}$$

The unit sample may be used to represent an arbitrary signal as a sum of weighted sample as follows

Equation 1.16

$$x(n) = \sum_{k=-\infty}^{\infty} x(k)\delta(n-k)$$

This decomposition is the discrete version of the *sifting property* for continuous-time signals. The *unit step,* denoted by $u(n)$, defined as

Equation 1.17

$$u(n) = \begin{cases} 1, & n \geq 0 \\ 0, & n < 0 \end{cases}$$

and is related to the unit sample by

Equation 1.18

$$u(n) = \sum_{k=-\infty}^{\infty} \delta(k)$$

Finally, the *complex exponential* is defined as

Equation 1.19

$$e^{jn\omega_0} = \cos(n\omega_0) + j\sin(n\omega_0)$$

where ω_0 is some real constant measured in radians. Later in the book, we will see that complex exponentials are extremely useful for analyzing linear systems and performing Fourier decompositions.

Discrete-Time Systems

A discrete-time system is a rule of correspondence that transforms an input signal into the output signal. The notation T[·] will be used to represent a general transformation. Our discussions shall be limited to a special class of discrete-time systems called *linear shift-invariant* (LSI) systems. As a notation aside, discrete-time systems are usually classified in terms of properties they possess. The most common properties include linearity, shift-invariance, causality, and stability, which are described below.

Linearity and Shift-Invariance

Two of the most desirable properties of discrete-time system for ease of analysis and design are *linearity* and *shift-invariance*. A system is said to be *linear* if the response to the superposition of weighted input signals is the superposition of the corresponding individual outputs weighted in accordance to the input signals, i.e.,

Equation 1.20

$$y = T[\alpha x_1 + \beta x_2] = \alpha T[x_1] + \beta T[x_2]$$

A system is said to be shift-invariant if a shift in the input by n_0 results in a shift in the output by n_0. In other words, shift-invariance means that the properties of the system do not change with time.

Causality

A very important property for real-time applications is *causality*. A system is said to be *causal* if the response of the system at time n_0 does not depend of future input values, i.e.,

Equation 1.21

$$y(n_0) = T[x(n-\tau)], \qquad 0 \le \tau \le \infty$$

Thus, for a causal system, it is not possible for changes in the output to precede changes in the input.

Stability

In many applications, it is important for a system to have a response that is bounded in amplitude whenever the input is bounded. In other words, if the unit sample response of LSI system is absolutely summable, i.e.,

Equation 1.22

$$\sum_{n=-\infty}^{\infty} |h(n)| < \infty$$

then for any bounded input |x(n)| ≤A < ∞the output is bounded |y(n)| ≤B < ∞This system is said to be *stable* in the Bounded-Input Bounded-Output (BIBO) sense. There are many other definitions for stability for a system, which can found in [7, 18]; however, BIBO is one of the most frequently used.

Thus far, we have discussed only the properties of LSI systems without providing an example of it. Consider an LSI system whose *q* coefficients are contained in the vector *h*. The response of the system to an input sequence x(n) is given by the following relationship

Equation 1.23

$$y(n) = \sum_{k=0}^{q} x(k)h(n-k)$$

is referred to as a *Finite length Impulse Response* (FIR) system. Notice that the output depends only on the input values in the system. It is possible, however, for the output of the system to depend on past outputs of the system as well as the current inputs, i.e.,

Equation 1.24

$$y(n) = \sum_{k=0}^{q} x(k)h(n-k) - \sum_{k=1}^{p} y(k)g(n-k)$$

This type of system is referred to as an *Infinite length Impulse Response* (IIR) system.

Filtering Random Processes

Earlier we have mentioned that all digital communication signals can be viewed as random processes. Thus, it is important to qualify the effects filtering has on the statistics of a random process. Of particular importance are LSI filters because of their frequent use in signal representation, detection, and estimation. In this section, we examine the effects of LSI filtering on the mean and autocorrelation of an input random process.

Let x(n) be a WSS random process with mean m_x, and autocorrelation $R_x(n)$. If x(n) is filtered by a stable LSI filter having a unit sample response h(n), the output y(n) is a random process that is related to input random process x(n) via the convolution sum

Equation 1.25

$$y(n) = x(n) * h(n) = \sum_{k=-\infty}^{\infty} h(k)x(n-k)$$

The mean of y(n), m_y, is found by taking the expected value of Equation 1.25 as follows,

$$E\{y(n)\} = E\left\{\sum_{k=-\infty}^{\infty} h(k)x(n-k)\right\} = \sum_{k=-\infty}^{\infty} h(k)E\{x(n-k)\}$$

$$m_y = E\{y(n)] = m_x \sum_{k=-\infty}^{\infty} h(k) = m_x H\left(e^{j0}\right)$$

where $H(e^{j0})$ is the zero-frequency response, or non-time varying response for the filter. Hence, the mean of y(n) is a constant equal to the mean x(n) scaled by the sample average of the unit sample response.

The autocorrelation of y(n), is derived and understood best by first proceeding with the cross-correlation r_{yx} between x(n) and y(n), which is given by

Equation 1.26

$$r_{yx}(k) = E\left\{y(n+k)x^*(n)\right\} = E\left\{\sum_{k=-\infty}^{\infty} h(l)x(n+k-l)x^*(n)\right\}$$

$$= \sum_{k=-\infty}^{\infty} h(l)E\left\{x(n+k-l)x^*(n)\right\}$$

$$= \sum_{k=-\infty}^{\infty} h(l)r_x(k-l) = r_x(k) * h(k)$$

Recall, under the assumption of WSS, the cross-correlation r_{xy} depends only on the difference between sampling instants. It is interesting to note that the cross-correlation r_{xy} as defined in Equation 1.26 is just the convolution of the input autocorrelation $R_x(n)$ with the channel impulse response h(n). We will use this result, shortly, to relate the output autocorrelation $R_y(n)$ to input autocorrelation $R_x(n)$.

The output autocorrelation is defined as

Equation 1.27

$$R_y(n) = E\{y(n+k)y^*(k)\} = E\left\{y(n+k)\sum_{l=-\infty}^{\infty} x^*(l)h^*(k-l)\right\}$$
$$= \sum_{l=-\infty}^{\infty} h^*(k-l)E\{y(n+k)x^*(l)\}$$
$$= \sum_{l=-\infty}^{\infty} h^*(k-l)r_{yx}(n+k-l)$$

Inspection of Equation 1.27 reveals that the autocorrelation of y(n), is really a convolution sum. Changing the index of summation by setting m = l – k, we obtain

Equation 1.28

$$R_y(n) = \sum_{l=-\infty}^{\infty} h^*(-m)r_{yx}(n-m) = r_{yx}(n) * h^*(-n)$$

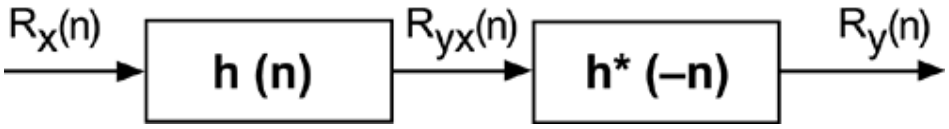
Combining Equations 1.26 and 1.28 we have

Equation 1.29

$$R_y(n) = R_x(n) * h(n) * h^*(-n)$$

Equation 1.29 is a key result exploited often in the reception of wireless communication signals. Figure 1.1 illustrates the concepts expressed in Equations 1.26 and 1.28.

Figure 1.1. Input-output autocorrelation for filtered random processes.



Discrete Fourier Transform (DFT)

The Discrete Fourier Transform (DFT) is, arguably, the most widely used design and analysis tool in electrical engineering. For many situations, frequency-domain analysis of discrete-time signals and systems provide insights into their characteristics that are not easily ascertainable in the time-domain. The DFT is a discrete version of the discrete-time Fourier transforms (DTFT); that is, the DTFT is a function of a continuous frequency variable, whereas the DFT is a function of a discrete frequency variable. The DFT is useful because it is more amenable to digital implementations. The *N-point* DFT of a finite length sequence x(n) is defined as

Equation 1.30

$$X(k) = \sum_{n=0}^{N-1} x(n) e^{-j2\pi kn/N}$$

Clearly, it can be seen from Equation 1.30 that the DFT is a sample version of the DTFT, i.e.,

Equation 1.31

$$X(k) = X(\omega)\Big|_{\omega=2\pi k/N}$$

where

Equation 1.32

$$X(\omega) = \sum_{n=-\infty}^{\infty} x(n) e^{-jn\omega}$$

The DFT has an inverse transformation called the *inverse DFT* (IDFT). The IDFT provides a means of recovering the finite length sequence x(n) through the following relationship,

Equation 1.33

$$x(n) = \frac{1}{N} \sum_{k=0}^{N-1} X(k) e^{j2\pi kn/N}$$

Let's now consider the DFTs of the discrete-time signals given in "Discrete-Time Signals," since they form the basic building blocks to generate more complex signals. Using the definition in Equation 1.15, the DFT of the unit sample $\delta(n)$ becomes

Equation 1.34

$$X(k) = \sum_{n=0}^{N-1} \delta(n) e^{-j2\pi kn/N} = 1$$

Using the definition of the unit step given in Equation 1.17, its DFT is given by

Equation 1.35

$$X(k) = \sum_{n=0}^{N-1} e^{-j2\pi kn/N} = N\delta(k)$$

The result in Equation 1.35 follows directly since, with the exception of the n = 0 term, each of the complex exponentials sums to zero over the sample period of length N. Finally, the DFT of the complex exponential is given by

Equation 1.36

$$X(k) = \sum_{n=0}^{N-1} e^{jn\omega_0} e^{-j2\pi kn/N} = N\delta(k - \frac{N\omega_0}{2\pi})$$

Besides the DFTs given above, some useful properties of the discrete Fourier transforms that facilitate communication system analysis are summarized in Table 1.1. where $\circledast$ denotes circular convolution, $(\cdot)_N$ denotes modulo-N operation, and * denotes the complex conjugate as previously defined. This concludes the review of the mathematical background material.

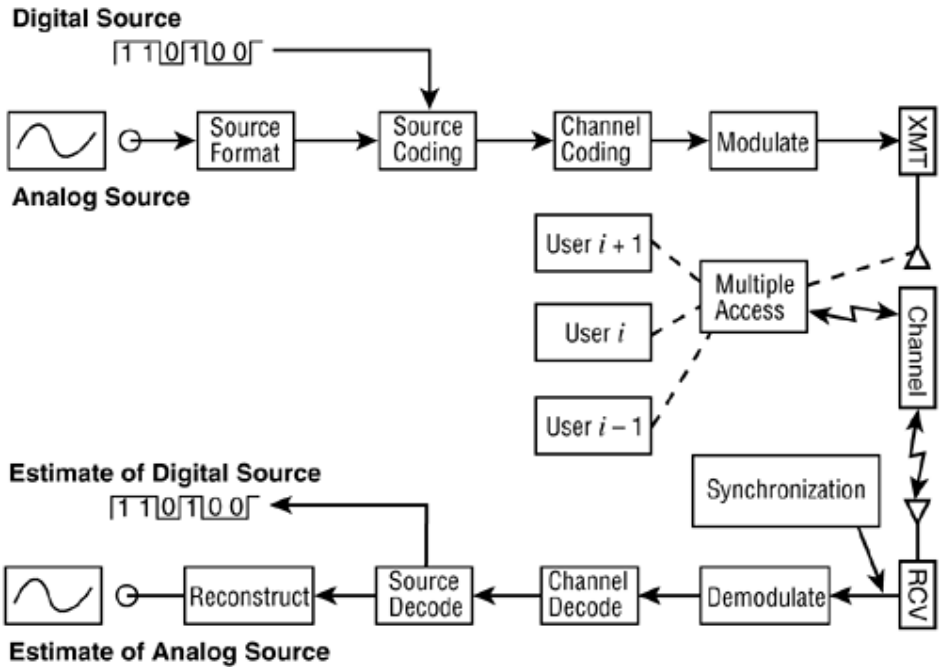
Table 1.1. Some Useful Properties of the DFT

Discrete-Time Signal		***DFT***
$x(n)$	$\leftrightarrow$	$X(k)$
$x(n-n_0)$	$\leftrightarrow$	$X(k)e^{-j2\pi kn_0/N}$
$\alpha x_1(n) + \beta x_2(n)$	$\leftrightarrow$	$\alpha X_1(k) + \beta X_2(k)$
$x^*(n)$	$\leftrightarrow$	$X((-k))_N$
$y(n) \circledast x(n)$	$\leftrightarrow$	$Y(k)X(k)$
$x^*((-n))_N$	$\leftrightarrow$	$X^*(k)$
$e^{-j2\pi mn/N}x(n)$	$\leftrightarrow$	$X((k-m))_N$

Components of a Digital Communication System

In this section, the basic elements of a digital communication system are reviewed. The fundamental principle that governs digital communications is the "divide and conquer" strategy. More specifically, the information source is divided into its smallest intrinsic content, referred to as a *bit*. Then each bit of information is transmitted reliably across the channel. In general, the information source may be either analog or digital. Analog sources are considered first since they require an additional processing step before transmission. Examples of analog sources used in our everyday lives are radios, cameras, and camcorders. Each of these devices is capable of generating analog signals, such as voice and music in the case of radios and video images in the case of camcorders. Unfortunately, an analog signal can not be transmitted directly by means of digital communications; it must be first converted into a suitable format. With reference to the top chain of Figure 1.2, each basic element and its corresponding receiver function will be reviewed in the order they appear left to right.

Figure 1.2. Basic elements of a digital communication system.



Source Formatting

Source formatting is the process by which an analog signal or continuous-time signal is converted digital signal or discrete-time signal. The device used to achieve this conversion is referred to as *analog-to-digital (A/D)* converter. The basic elements of an A/D converter shown in Figure 1.3 consists of a sampler, quantizer, and encoder. The first component, the sampler, extracts sample values of the input signal at the sampling times. The output of the sampler is a discrete-time signal but with a continuous-valued amplitude. These signals are often referred to as *sampled data signals;* refer to Figure 1.4 for an illustration. Digital signals, by definition, are not permitted to have continuous-valued amplitudes; thus, the second component, the quantizer, is needed to quantize the continuous range of sample values into a finite number of sample values. Finally, the encoder maps each quantized sample value onto a digital word.

Figure 1.3. Basic elements of A/D converter.

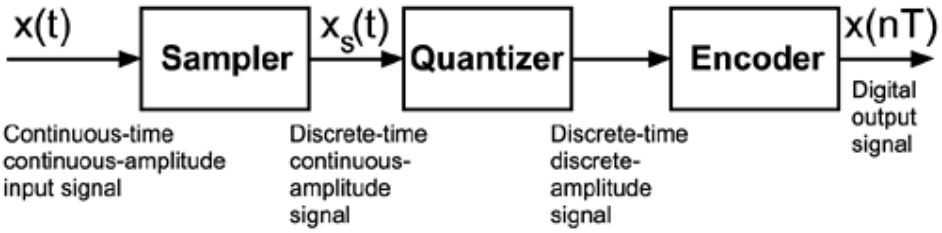
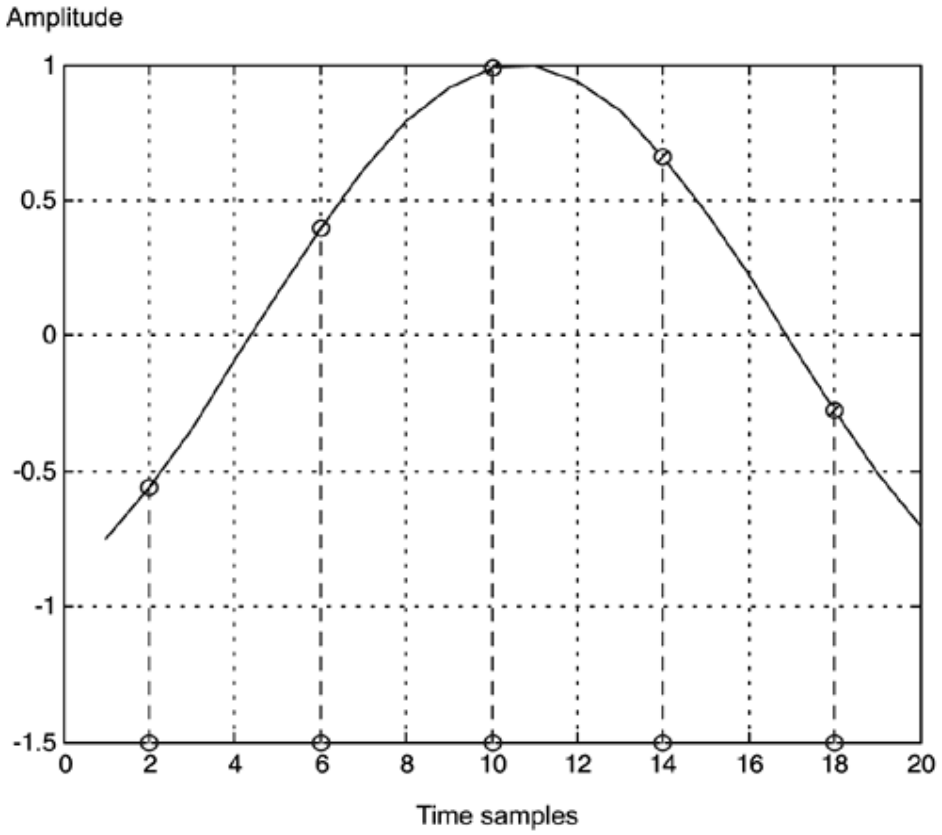


Figure 1.4. An example of a discrete-time continuous amplitude signal $X_s(t)$.



Sampling and Reconstruction

To model the sampling operation of a continuous-time signal, we make use of a variant of the unit sample function $\delta(n)$ defined in "Discrete-Time Signals." An ideal sampler is a mathematical abstraction but is useful for analysis and given by

Equation 1.37

$$p(t) = \sum_{n=-\infty}^{\infty} \delta(t - nT_s)$$

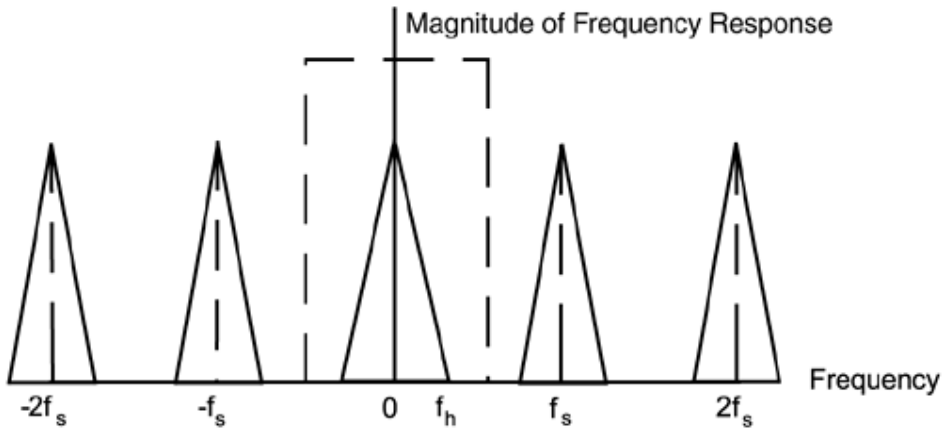
where T_s is the sampling interval. Now, the sampled data signal $x_s(t)$ can be expressed as

Equation 1.38

$$x_s(t) = x(t)p(t)$$

where x(t) is the continuous-time continuous-amplitude input signal as seen in Figure 1.4. Obviously, the more samples of the x(t) we have, the easier it will be to reconstruct the signal. Each sample will be eventually transmitted over the channel. In order to save bandwidth, we would like to send the bare minimum number of samples needed to reconstruct the signal. The sampling rate that produces this is called the *Nyquist rate*. Nyquist sampling theorem states that samples taken at a uniform rate $2f_h$ where f_h is the highest frequency component of a band-limited signal x(t), is sufficient to completely recover the signal. The Nyquist Sampling Theorem is conceptually illustrated in Figure 1.5. Notice first that sampling introduces periodic spectral copies of the original spectrum center at the origin. Second, the copies are sufficiently spaced apart such that they do not overlap. This simple principle is the essence of the Nyquist Sampling Theorem. If the spectral copies of the spectrum were permitted to overlap, aliasing of the spectral copies would occur. It would then be impossible to recover the original spectrum by passing it through a low-pass filter whose ideal frequency response is represented by the dashed box.

Figure 1.5. Spectrum of a sampled waveform.

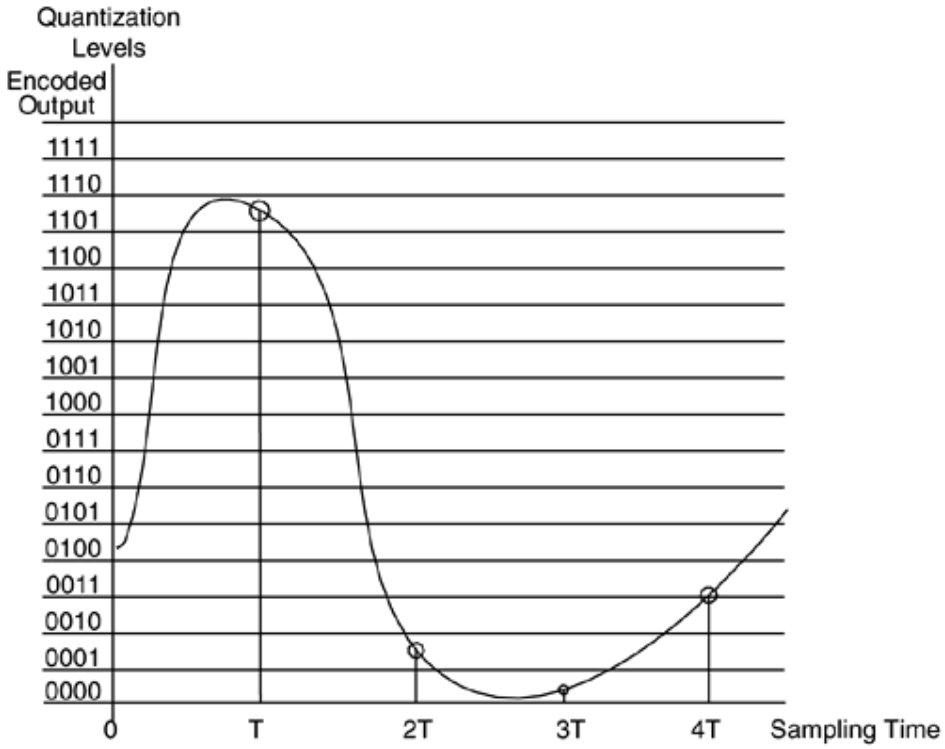


Quantization and Encoding

After the signal has been sampled, the amplitude values must be quantized in a discrete set of values. The process of quantization entails rounding off the sample value to the nearest of a finite set of permissible values. Encoding is accomplished by assigning a digital word of length *k,* which represents the binary equivalent for the numerical value of the quantization level, as depicted in Figure 1.6. For example, at sampling time T, the amplitude of the signal lies within quantization level 12, which has a binary representation of 1101. Furthermore, the number of quantization levels *q* and the digital word length *k* are related by

$$q = 2^k$$

Figure 1.6. Quantization and digital encoding.



Another key observation that should be taken from Figure 1.6 is that the quantization process introduces an error source usually referred to as *quantization noise*. When decoding, it is usually assumed the amplitude falls at the center of the quantization level. Hence, the maximum error which can occur is $\pm\frac{1}{2}\Delta L$, where

Equation 1.39

$$\Delta L = \frac{A}{2^k}$$

and A is the difference between the maximum and minimum values of the continuous-time signal. If we assume a large number of quantization levels, error function is nearly linear within the quantization level, i.e.,

Equation 1.40

$$e(t) = \frac{\Delta L}{2\tau} t$$

where 2τ is the time the continuous-time signal remaining within the quantization level. The mean-square error P_n is thereby given by

Equation 1.41

$$P_n = \frac{1}{2\tau}\int_{-\tau}^{\tau} e^2(\lambda)\partial\lambda = \frac{\Delta L^2}{4\tau^3}\int_0^{\tau} \lambda^2 \partial\lambda$$

Equation 1.42

$$P_n = \frac{\Delta L^2}{12}$$

The quantity of most interest, however, is the *signal-to-noise ratio* (SNR) at the output of the A/D converter, which is defined as

Equation 1.43

$$SNR = \frac{P_s}{P_n}$$

where P_s and P_n are the power in the signal and noise, respectively. The signal power for an input sinusoidal signal is defined as

Equation 1.44

$$P_s = \frac{(A/2)^2}{2} = \Delta L^2\left(2^{2k-3}\right)$$

which leads to a SNR of

Equation 1.45

$$SNR = \frac{12\Delta L^2\left(2^{2k-1}\right)}{\Delta L^2} = \frac{3}{2}\left(2^{2k}\right)$$

or, in decibels,

$$SNR(dB) = 10\log_{10}(SNR) = 1.76 + 6.02k$$

Thus, the SNR at the output of the A/D converter increases by approximately 6 dB for each bit added to the word length, assuming the output signal is equal to the input signal to the A/D converter.

Source Coding

Source coding entails the efficient representation of information sources. For both discrete and continuous sources, the correlations between samples are exploited to produce an efficient representation of the information. The aim of source coding is either to improve the SNR for a given bit rate or to reduce the bit rate for a given SNR. An obvious benefit of the latter is a reduction in the necessary system resource of bandwidth and/or energy per bit to transmit the information source.

A discussion on source coding requires the definition of a quantity, which measures the average *self-information* in each discrete alphabet, termed *source entropy* [14]. The self-information $I(x_{j})$ for discrete symbol or alphabet x_j is defined as

Equation 1.46

$$I(x_j) = -\log_2(p_j)$$

where p_j is the probability of occurrence for x_j. The source entropy $H(x_j)$ is found by taking the statistical expectation of the self-information, i.e.,

Equation 1.47

$$H(x_j) = E\{I(x_j)\} = -\sum_{j=1}^{M} p_j \log_2(p_j)$$

where M is the cardinality of the discrete alphabet set and the units of measure for $H(x_j)$ are bits/symbol. The source entropy can also be thought of as the average amount of uncertainty contained in the alphabet x_j. Therefore, the source entropy is the average amount of information that must be communicated across the channel per symbol. It can be easily shown that $H(x_j)$ is bounded by the following expression

Equation 1.48

$$0 \le H(x_j) \le \log_2 M$$

In other words, the source entropy is bounded below by zero if there is no uncertainty, and above by $\log_2 M$ if there is maximum uncertainty. As an example, consider a binary source x_j that generates independent symbols 0 and 1 with respective probabilities of p_0 and p_1. The source entropy is given by

Equation 1.49

$$H(x_j) = -\left[p_1 \log_2(p_1) + p_0 \log_2(p_0)\right]$$

Table 1.2 lists the source entropy as the probabilities of p_0 and p_1 are varied between 0 and 1.

Table 1.2. Source Entropy

p_0	p_1	$H(x_j)$
0.5	0.5	1.00
0.1	0.9	0.47
0.2	0.8	0.72

Table 1.2 illustrates that as the relative probabilities vary, the number of bits/symbol needed to transmit the information vary as well. One might wonder what information source would have such probabilistic distribution. English text, for instance, is known to have unequal probability for its alphabet; in other words, some letters appear in English text more frequently than others. Another key consideration when determining the source entropy is whether the source is *memoryless*. A discrete source is said to be memoryless if the sequence of symbols is statistically independent. The readers should refer to "Review of Stochastic Processes and Random Variables" for a definition of statistical independence. In contrast, if a discrete source is said to have memory, the sequence of symbols is not independent. Again, consider English text as an example. Given that the letter "q" has been transmitted, the next letter will probably be a u. Transmission of the letter "u" as the next symbol resolves very little uncertainty from a communication perspective. We can thus conclude that the entropy of an *M*-tuple from a source with memory is always less than the entropy of a source with the same alphabet and symbol probability without memory, i.e.,

Equation 1.50

$$H_M(x_j)_{with\ memory} < H_M(x_j)_{without\ memory}$$

Another way to interpret the relationship in Equation 1.50 is that the average entropy per symbol of an *M*-tuple from a source with memory decreases as the length *M* increases. Hence it is more efficient to encode symbols from a source with memory three at a time than it is to encode them two at a time or one at a time. Encoder complexity, memory constraints, and delay considerations require that practical source encoding be performed on finite-length sequences. Interested readers are encouraged to examine References [8, 13, 15, 25] dealing with source coding.

Channel Coding

Channel coding refers to the class of signal transformations designed to improve communications performance by enabling the transmitted signal to better withstand the effects of various channel impairments, such as noise, fading, and jamming. The goal of channel coding is to improve the bit error rate (BER) performance of power-limited and/or bandwidth limited channels by adding structured redundancy to the transmitted data. The two major categories of channel coding are block coding and

convolutional coding. In the following sections, we describe the fundamental principles governing each of these two types of codes.

Linear Block Codes

Linear block codes are a class of parity check codes that can be characterized by an integer pair (*n, k*). As such, the parity bits or symbols are formed by a linear sum of the information bits. In general, the encoder transforms any block of *k* message symbols into a longer block of *n* symbols called a *codeword*. A special class of linear block codes called systematic codes append the parity symbols to the end of the *k* symbol message to form the coded sequence. These codes are of particular interest since they result in encoders of reduced complexity.

For binary inputs, the *k*-bit messages, referred to as *k-tuples,* form 2^k distinct message sequences. The *n*-bit blocks, referred to as *n*-tuples, form 2^k distinct codewords or message sequences out of 2^n possible *n*-tuples in the finite dimensional vector space.

Hence, one can view linear block codes as a finite dimensional vector subspace defined over the extended Galois fields GF(2^p). The transformation from a *k*-dimensional to an *n*-dimensional vector space is performed according to a linear mapping specified in the generator matrix ***G***. In other words, an (*n, k*) linear block code ***C*** is formed by partitioning the information sequence into message blocks of length *k*. Each message block u_i is encoded or mapped to a unique codeword or vector v_i in accordance with ***G***

Equation 1.51

$$\mathbf{v}_i = \mathbf{u}_i \mathbf{G}$$

Another useful matrix, often defined when describing linear block codes, is the parity-check matrix ***H***. The parity check matrix has the property that it generates code words that lie in the null space of ***G***. Mathematically, we describe this relationship as

Equation 1.52

$$\mathbf{G}\mathbf{H}^T = \mathbf{0}_{n-k}$$

where 0_{n-k} is the zero vector of length n-k. As we will see in the following paragraphs, ***H*** plays an integral part in the detection process for linear block codes.

Consider a code vector v_i transmitted over a noisy channel. Denote the received vector from the channel as r_i. The equation relating r_i to v_i is

Equation 1.53

$$\mathbf{r}_i = \mathbf{v}_i + \mathbf{e}$$

where ***e*** is an error pattern or vector from the noisy channel. At the receiver, the decoder tries to determine v_i given r_i. This decision is accomplished using a two-step process: one, compute the *syndrome;* and two, add the *coset leader* corresponding to the syndrome to the received vector. We have just introduced two new terms that need defining. The syndrome ***s*** is defined as the projection of the received vector onto the subspace generated by ***H***

Equation 1.54

$$\mathbf{s}_i = \mathbf{r}_i \mathbf{H}^T$$

Using Equations 1.51–1.53, we see that Equation 1.54 simplifies to

Equation 1.55

$$\mathbf{s}_i = \mathbf{e}\mathbf{H}^T$$

The foregoing development is evidence that the syndrome test, whether performed on a corrupted codeword or on the error pattern that caused it, yields the same syndrome. An important property of linear block codes, fundamental in the decoding process, is the one-to-one mapping between correctable error patterns and the associated syndrome. This equivalence leads us to our discussion on coset leaders.

The coset leaders consist of all the correctable error patterns. A correctable error pattern is one whose *Hamming weight* is less than or equal to $\lfloor (d_{min}-1)/2 \rfloor$, where $\lfloor x \rfloor$ means the largest integer not to exceed *x,* and d_{min} is defined as the minimum *Hamming distance* between any two code words for an (*n, k*) linear block code. For binary vectors, the Hamming weight is defined as the number of non-zero elements in a vector, and the Hamming distance between two code vectors is defined as the number of elements in which they differ. It is interesting to note that d_{min} for a linear block code is equal to the non-zero codeword with minimum Hamming weight. These concepts are easily generalized to the extended Galois field.

Note that there are exactly 2^{n-k} coset leaders for the binary case. Now, let's form a matrix as follows: first, list all the coset leaders in the first column; second, list all the possible code words in the top row; and, last, form the (*i, j*) element of the matrix from the sum of *i*th element of the first column with the *j*th element of the top row. The resulting matrix represents all possible received vectors and is often referred to as the *standard array*.

Now, we will relate how all the above fits into the decoding process. As stated earlier, the task of the decoder is to estimate the transmitted code vector, v_i from the received vector r_i. The maximum likelihood solution $\hat{\mathbf{v}}_{ML}$ found by maximizing the conditional probability density function for the received vector for all possible code vectors v_j; namely,

Equation 1.56

$$\hat{\mathbf{v}}_{ML} = \arg \max_{\mathbf{v}_j} \Pr\left(\mathbf{r}_i \middle| \mathbf{v}_j\right)$$

The optimization criterion for Equation 1.56 over a binary symmetric channel (BSC) is to decide in favor of the code word that is the minimum Hamming distance from the received vector

Equation 1.57

$$d(\mathbf{r}_i, \mathbf{v}_i) = \min d(\mathbf{r}_i, \mathbf{v}_j) \quad \forall \mathbf{v}_j$$

A systematic procedure for the decoding process proceeds as follows:

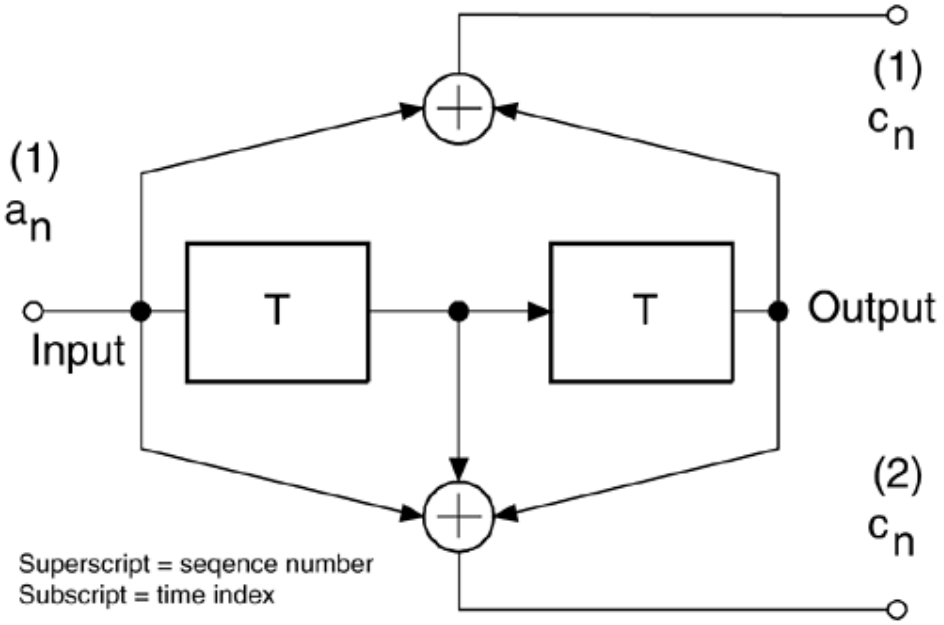
1. Calculate the syndrome of ***r*** using $s = rH^T$.
2. Locate the coset leader whose syndrome equals rH^T from the standard array table.
3. This error pattern is added to the received vector to compute the estimate of ***v*** or read directly from the corresponding column in the standard array table.

Convolutional Codes

Another major category of channel coding is convolutional coding. An important characteristic of convolutional codes, different from block codes, is that the encoder has memory. That is, the *n*-tuple generated by the convolutional encoder is a function of not only the input *k*-tuple but also the previous K-1 input *k*-tuples. The integer *K* is a parameter known as the *constraint length,* which represents the number of memory elements in the encoder. A shorthand convention typically employed in the description of a convolutional encoder is (*n, k, K*) for the integers defined above.

To understand the fundamental principles governing the convolutional codes, we direct our attention to the (2,1,2) convolutional encoder depicted in Figure 1.7. This encoder, at any given sampling time instant, accepts *k* input bits, makes a transition from its current state to one of the 2^K possible successor states, and outputs *n* bits.

Figure 1.7. Four state convolutional encode.



The two noteworthy characteristics of a convolutional encoder are the tap connections and the contents of the memory elements. For the tap connections of the encoder, the generator polynomial representation is commonly used. Yet, for the state information as well as the output codewords, a state diagram or trellis diagram is typically employed.

With the generator polynomial representation, the taps for each output of the encoder are specified by a polynomial $g_i(D)$ where the coefficients of $g_i(D)$ are taken from $GF(2^p)$. Note, *p* equal to one corresponds to the binary field. For this case, a 1 coefficient denotes a connection and a 0 coefficient denotes no connection. The argument of the polynomial, *D,* denotes a unit time delay. In other words, we represent two unit time delays as D^2. The coded bits are formed by computing the sum over $GF(2^p)$ dictated by the tap coefficients. The two components of the sum are the current state of memory elements and the *k* input bits. To clarify these concepts, again consider the encoder shown in Figure 1.7. The generator polynomials for the outputs of this encoder are

Equation 1.58

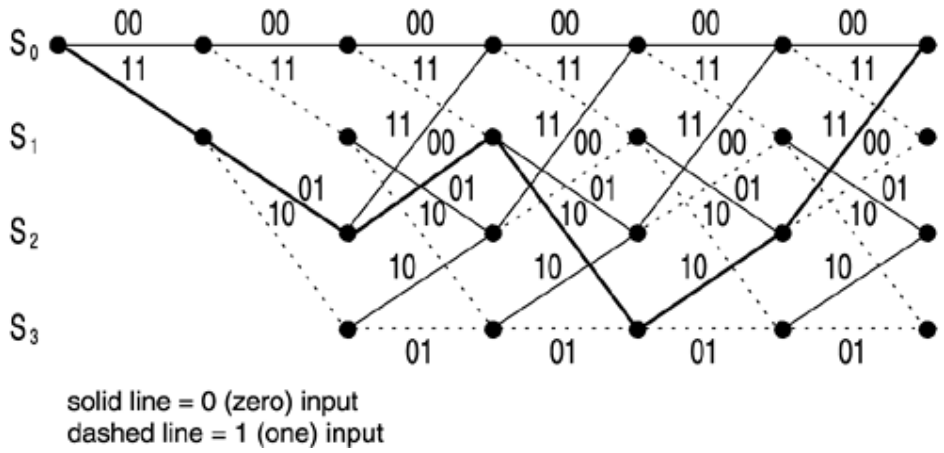
$$g_1(D) = 1 + D^2$$

Equation 1.59

$$g_2(D) = 1 + D + D^2$$

We now briefly describe a method for viewing the state information as well as the output bits: the trellis diagram. We noted earlier that the convolutional encoder is a finite state machine. Thus, a natural way to describe its behavior is to view its state transition at each time instant. The state of the encoder is considered to be the contents of its memory elements. For a convolutional code with *K* memory elements, there are 2^K states. Each state is assigned a number from 0 to 2^K-1, which is obtained from the binary representation of its memory elements. The trellis diagram shows the state transitions and associated outputs for all possible inputs as seen in Figure 1.8. For binary inputs, solid and dashed lines are used, respectively, to represent a 0 or 1 input into the encoder. We also notice in the figure that there are two branches entering and leaving each state at every time instant. In general, there are 2^k branches emanating from each state and 2^k branches merging to each state. Each new sequence of *k* input bits causes a transition from an initial state to 2^k states at the next node in the trellis or time instant. Note, the encoder output is always uniquely determined by the initial state and the current input. Additionally, the trellis diagram shows the time evolution of the the states. Later, we will see that the trellis diagram is also very useful for evaluating the performance of these codes.

Figure 1.8. Trellis diagram for the four state convolutional encoder.



Note that after three stages into the trellis diagram, each node at subsequent stages has two branches entering and leaving it. Furthermore, it can be shown that the branches originate from a common node three stages back into the trellis. In general, for any constraint length (*n, k, K*) convolutional code, every *K* stages into the trellis diagram mark the point where 2^k branches merge and diverge at each node. Furthermore, the merging branches in a node can be traced back *K* stages into the past to the same originating node. This observation led to the development of a systematic approach for the optimum decoding of convolutional codes.

Decoder

As was the case for linear block codes, the decoder task is to estimate the transmitted code word from a received vector. It was shown that the optimum decoder strategy for linear block codes is to select the codeword with the minimum distance metric, the Hamming distance. The same is true for convolutional codes as well. Recall that an encoder for a convolutional code has memory. Thus, it is a reasonable approach to use all codewords associated with a particular symbol in the decision determination for it. Hence, for convolutional codes a sequence of codewords, or paths through the trellis, are compared to determine the transmitted codeword. In 1967, Viterbi [25] developed an algorithm that performs the maximum likelihood decoding with reduced computational load by taking advantage of the special structure of the code trellis.

The basis of *Viterbi decoding* is the following observation. If any two paths in the trellis merge to a single state, one of them can always be eliminated in the search for the optimum path. A summary of the Viterbi decoding algorithm proceeds as follows:

- Measure the similarity or distance between the received signals at each sampling instant t_i and all the paths entering each state or node at time t_i.
- The Viterbi algorithm eliminates from consideration the paths from the trellis whose distance metrics are not the minimum for a particular node. The distance metric can be either the Hamming distance or Euclidean distance. In other words, when two paths enter the same state, the one having the best distance metric is chosen. This path is called the *surviving path*. The selection of the surviving paths is performed for all the states.
- The decoder continues in this way, advancing deeper in the trellis and making decisions by elimination of least likely paths. In the process, the cumulative distance metric for each surviving path is recorded and used later to determine the maximum likelihood path.

The early rejection of the unlikely paths reduces the decoding complexity. In 1969, Omura [25] demonstrated that the Viterbi algorithm is, in fact, the maximum likelihood estimator. For a discussion of

more advance of coding techniques, the reader is referred to Chapter 3, "Modulation and Coding," of this book.

Modulation

Modulation is the process by which information signals, analog or digital, are transformed into waveforms suitable for transmission across channel. Hence, digital modulation is the process by digital information are transform into digital waveforms. For baseband modulation, the waveforms are pulses; but for band-pass modulation, the information signals are transformed into radio frequency (RF) carriers, which have embedded the digital information. Since RF carriers are sinusoids, the three salient features are its amplitude, phase, and frequency. Therefore, digital band-pass modulation can be defined as the process whereby the amplitude, phase, or frequency of an RF carrier, or any combination of them, is varied in accordance with the digital information to be transmitted. The general form of a complex RF carrier is given by

Equation 1.60

$$s(t) = A(t)\exp(\omega_c t + \theta(t))$$

where ω_c is the *radian frequency* of the carrier and $\theta(t)$ is the time-varying phase. The radian frequency of the RF carrier is related to its frequency in Hertz by

Equation 1.61

$$\omega_c = 2\pi f_c$$

At the receiver, the transmitted information embedded in the RF carrier must be recovered. When the receiver uses knowledge of the phase of the carrier to detect the signal, the process is called *coherent detection;* otherwise, the process is known as *non-coherent detection*. The advantage of non-coherent detection over coherent detection is reduced complexity at the price of increased probability of symbol error (P_E).

Whether the receiver uses coherent or non-coherent detection, it must decide which of the possible digital waveforms most closely resembles the received signal, taking into account the effects of the channel. A more rigorous treatment of common digital modulation formats and demodulation techniques is given in Chapter 3 of this book.

Multiple Access Techniques

Multiple access refers to the remote sharing of a fixed communication resource (CR), such as a wireless channel, by a group of users. For wireless communications, the CR can be thought of as a hyperplane in frequency and time. The goal of multiple access is to allow users to share the CR without creating unmanageable interferences with each other. In this section, we review the three most basic multiple access techniques for wireless communications: frequency division multiple access (FDMA), time division multiple access (TDMA), and code division multiple access (CDMA). Other more sophisticated techniques are combinations or variants of these three.