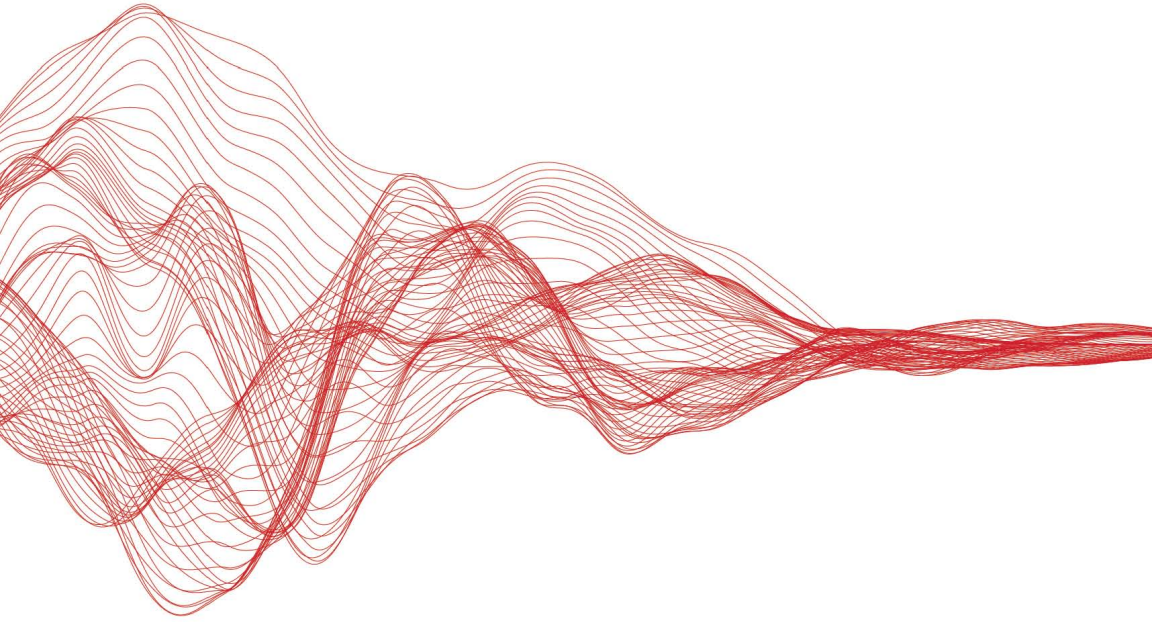
Introduction to OFDM Receiver Design and Simulation

Y. J. Liu



Introduction to OFDM Receiver Design and Simulation

For a listing of recent titles in the
Artech House Mobile Communications Library,
turn to the back of this book.

Introduction to OFDM Receiver Design and Simulation

Y. J. Liu

ARTECH HOUSE

BOSTON | LONDON

artechhouse.com

Library of Congress Cataloging-in-Publication Data
A catalog record for this book is available from the U.S. Library of Congress

British Library Cataloguing in Publication Data
A catalog record for this book is available from the British Library.

ISBN-13: 978-1-63081-738-1

Cover design by John Gomes

685 Canton St.
Norwood, MA

10 9 8 7 6 5 4 3 2 1

To my family and friends for their love and support

Contents

Preface

Orthogonal frequency-division multiplexing (OFDM) has been widely used for Wi-Fi communications. It also has application in digital audio broadcasting (DAB), digital video broadcasting (DVB), and the 4G Long Term Evolution (LTE) cellular system. Its major characteristic is to use many subcarriers for data modulation. In comparison with a single-carrier scheme, the major advantage is its ease in handling a dispersive fading channel. Currently, Wi-Fi is provided in many places including homes, hotels, airports, and shops. A variety of terminal devices including laptops, personal computers, cameras, televisions, and media players have a Wi-Fi interface for internet access.

Wi-Fi digital communications are based upon several IEEE 802.11 standards. The 802.11a specifies a maximum data rate of 54 Mbps. The 802.11g has a similar OFDM modulation to 802.11a. The 802.11n uses a combination of OFDM and multiple-input multiple-output (MIMO) technology to transmit up to four spatial streams to achieve a peak data rate of 600 Mbps. The 802.11ac is a further upgrade of 802.11n aiming at a data rate in the gigabit range.

The goal of this book is to provide a fundamental understanding of the receiver design applying OFDM technology. One distinct feature of this book is the extensive simulation study of an algorithm. The simulation program was written in C running on an Ubuntu Linux operating system. Because of the simulation needs, a transmission waveform must be selected. In spite of several IEEE standards mentioned above, the OFDM principle remains the

same. The transmission waveform specified in 802.11a was chosen. Once the reader has an understanding of the design concepts based upon 802.11a, a similar principle can be easily extended to other OFDM waveforms.

For example, the next generation cellular wireless digital communication is 5G (fifth generation), which is emerging. The 5G technology provides much higher speed and is capable of connecting a massive number of devices. The Third Generation Partnership Project (3GPP), the standards body behind cellular 5G, has again adopted OFDM. Therefore, an understanding of basic OFDM will help with stepping into the 5G era.

From Chapters 8 to 12, many simulation examples are provided. Besides the simulation study, detailed derivations leading to the final formula for any algorithm are given. In doing it this way, the reader can clearly understand the approximations and conditions behind the formulas and apply them in the correct way. If some important intermediate derivations are omitted, it is very difficult to understand any algorithm. The emphasis is not on mathematics, but includes enough to grasp the concepts behind the formula.

For any design topic used in the receiver, there can be many algorithms in the literature. Only algorithms that were required for illustration and comparison were provided to not confuse the reader. In most cases, only one algorithm that is most promising from the simulation study was given. In addition to the simulations, examples were provided in many cases to help to understand the design subject. Therefore, this book is ideal for undergraduate seniors, graduate students, engineers, and even professors who would like to start understanding the OFDM technology.

This book is organized into 12 chapters. Chapter 1 provides the basics of digital signal processing. We start with discrete time signals generated by sampling an analog waveform. Next, Fourier series representation is given. From its spectrum, the requirement for perfect analog signal reconstruction is derived and that leads to the sampling theorem. The discrete Fourier transform, inverse discrete Fourier transform, and an interpolation formula that can reconstruct an analog signal based upon the discrete samples are then subsequently derived.

The discrete Fourier transform has an important time and frequency relationship and is presented. In addition, some operations including linearity, time shift, frequency shift, and circular convolution are discussed. The real-time signal has interesting symmetry properties that are illustrated. Lastly, the z-transform and its usage to specify a time invariant linear system are provided.

Chapter 2 reviews the single-carrier modulation techniques. The analog modulated signal is first shown to be a bandpass signal. Its information

carrying a baseband signal is further represented in terms of the in-phase and quadrature-phase components. Based upon how the phase, frequency, and amplitude are modulated, different modulation techniques are discussed. They include phase shift keying (PSK), quadrature phase shift keying (QPSK), 16-QAM (quadrature amplitude modulation), 64-QAM, and frequency shift keying (FSK). Pulse-shaping waveforms are also discussed. They include sinc and raised cosine waveforms.

Chapter 3 introduces the multicarrier OFDM modulation. Through inverse Fourier transform, the serial bit stream is converted into a multicarrier modulated waveform. Several OFDM waveform characteristics such as orthogonality among subcarriers, cyclic prefix, fast Fourier transform (FFT), classical spectrum, and peak-to-average power ratio (PAPR) are explained in detail through derived equations. The modified rectangular window used in IEEE 802.11a is presented. To show its spectral properties, the discrete Fourier spectrum is derived. Lastly, the single-carrier modulation technique in Chapter 2 is applied to modulate each subcarrier in the frequency domain.

Chapter 4 presents the OFDM transmitter signal format, which can be different depending upon the applications and the standard. To give the reader some basic concepts, the format specified by the IEEE 802.11a is followed. First, a general transmitter and receiver architecture is discussed. Next, a preamble signal that aids in signal detection and synchronization is given. For the IEEE 802.11a, it consists of both a short sequence and a long sequence that are periodic in nature. After the preamble is a header, which informs the receiver of the selected modulation method, and finally, is the source data transmission.

Chapter 5 centers on the shift register sequence and the scrambler. The source data from the transmitter can be scrambled for security protection. Since the IEEE 802.11a specifies a pseudo-noise sequence for data scrambling, its generation technique is given in detail. First, both a binary field having only two elements and a Galois field having 2^m elements are reviewed. The binary field performs an operation of modulo 2 while a Galois field performs a modulo operation over a primitive polynomial. Next, a sequence generator through polynomial division to generate a periodic sequence with a desired period is presented. The period obtained through an inverse polynomial is illustrated. The maximum length sequence that generates a maximum period using a primitive polynomial is also discussed. Having the periodic sequence generated, the scrambler is just the exclusive-OR operation between the source data bit sequence and the periodic sequence. The descrambler in the receiver has just a similar operation.

Chapter 6 discusses the radio-wave propagation loss through the media and the receiver front-end noise. Both a large-scale model and a small-scale model for the propagation loss are discussed. The large-scale model includes a free-space model and a two-ray mode. An empirical model that can fit many indoor and outdoor propagation losses is also presented. The small-scale model, including both time dispersion and frequency dispersion, is given. The time dispersion further discusses the flat channel and the frequency-selective channel, while the frequency dispersion discusses both the slow channel and the fast channel. A Clark fading model with Rayleigh distribution for the received envelope is also given.

After the propagation loss, the receiver noise is discussed to compute the signal-to-noise ratio. The amplifier noise model, cable loss model, and thermal noise from the receiver front end are all presented. The noise factor that can be used to generate an equivalent temperature for these devices is included. An equivalent temperature referred to the antenna through a cascade of amplifiers, cables, and receivers is derived. Based upon the transmitter power, the received power, and the loss prediction, the range estimate can be obtained.

Chapter 7 concentrates on error correction codes that are necessary to correct the transmission errors from the channel. Two classes of codes are covered: the block code and the convolutional code. For the block code, the first is the linear block code, which covers the generator matrix, parity check matrix, syndrome matrix, and error correction. The second is the cyclic code, which operates on polynomials. The topics covered are the generator polynomial, syndrome polynomial, and error correction. The Hamming code, which is a special case of linear block codes, is also discussed.

For the convolutional code, both an encoder and a decoder using a Viterbi algorithm are discussed. The convolutional code used in the IEEE802.11a is also analyzed. Another special case is the punctured convolutional code, which is derived from the rate 1/2 convolutional code. Both rate 2/3 and rate 3/4 punctured codes are illustrated in detail.

The last topic covered is the interleaver, which is also widely used to decrease burst errors. The special interleaver and deinterleaver used in the IEEE 802.11a are also illustrated.

Chapter 8 covers the OFDM signal detection using the preamble waveform. After direct IQ conversion, three detection metrics including cross-correlation, minimum mean square error (MMSE), and normalized cross-correlation are analyzed. A simplified algorithm from the maximum likelihood detection is derived and justified through simulation. All three metrics are compared using this algorithm.

The detection includes both coarse detection and fine detection. The coarse detection is done in two steps. The first step is the segment detection for fast detecting the periodic segment in the preamble waveform. The second step is the sample detection to narrow down the detection error in samples, followed by the fine detection to pinpoint the sample onset.

Chapter 9 deals with synchronization issues in the receiver. The topics covered include direct current (DC) offset, carrier frequency offset, frame timing offset, sampling clock offset, and in phase and quadratic phase (IQ) imbalance. For the sampling clock offset, the formulas in the acquisition mode using the preamble waveform and the tracking mode from using the pilot carriers in the data symbols are demonstrated. All the issues were analyzed and derived in detail without omitting any intermediate steps. The possible approximations and conditions are clearly illustrated. Simulation examples are also given to help to understand each topic.

Chapter 10 concentrates on channel estimation. Two types of pilot patterns are analyzed. They are the block type and the comb type, which are mainly used for slow- and fast-varying channels, respectively. The channel estimate for the block type is though discrete Fourier transform (DFT). For the comb type, it is though interpolation of a few pilot carriers inside each data symbol. Using the Lagrange interpolation formula, linear interpolation, parabolic interpolation, and cubic interpolation are discussed. The channel can also be adaptively tracked through the use of the least mean square (LMS) algorithm, which is derived in detail using the method of steep descent. The condition for convergence is also derived.

Chapter 11 talks about the decoding process. After the channel effect is removed, the data demodulation follows. Both hard decision decoding and soft decision decoding are discussed. For the hard decision decoding, either a brute-force approach or a simplified approach can be applied. The brute-force approach has to compute the Euclidean distances for all the constellation points. However, the simplified approach can determine each in-phase and quadrature-phase bits individually by using a simple logic that is derived for QPSK, 16-QAM, and 64-QAM. After the demodulation is complete, deinterleaving, error correction decoding, and descrambling discussed in the previous chapters follow. For the soft decision decoding, the received in-phase and quadrature samples are combined with Viterbi decoding to generate corrected output sequence. Both the Euclidean distance and the log likelihood ratio (LLR) used for the distance metric are discussed.

Chapter 12 performs the simulation study of the OFDM performance in a multipath channel. First, the time-varying multipath channel is characterized.

Then the transmitter and receiver architecture used for simulation are reviewed. The simulation consists of two parts. The first part is the OFDM performance in an additive white Gaussian noise, (AWGN) channel. The performance of binary phase shift keying (BPSK), QPSK, 16-QAM, and 64-QAM are compared using a 1/2 convolutional code. The study using punctured convolutional code is also conducted. The second part is the simulation study of OFDM in a multipath channel with the use of a 1/2 convolutional code. The channel also has fading and a simulation model is provided. The block pilot pattern together with LMS algorithm-channel tracking are of particular interest. Various fading strength and block pilot periods are used for performance comparisons.

1

Discrete Time Signals and Discrete Fourier Transform

1.1 Introduction

In digital communication, the input signal is digitized, processed, and modulated before going through a digital-to-analog converter in order for transmission. For example, the input speech from a microphone is digitized first before encoding and modulation. This analog signal can be perfectly reconstructed if an appropriate sampling rate is specified.

Orthogonal frequency division multiplexing (OFDM) is a frequency-domain modulation technique and will be discussed in later chapters in detail. The major operation of OFDM is fast Fourier transform (FFT), which is a fast computation method of discrete Fourier transform (DFT). In preparation for understanding OFDM, all the steps from discrete time signals to DFT and sampling theorem are reviewed. The major operations of DFT are also discussed.

The z-transform, which is useful in specifying the time-invariant system response, is also briefly reviewed.

1.2 Discrete Time Signals

An analog signal is a time continuous waveform. If this signal is sampled with a time period T, then a sequence of samples is obtained. The sampling frequency f_s is just the inverse of the T or $1/T$. Figure 1.1 shows the analog waveform $x(t)$ and the sampled values at nT where n is an integer.

The sampling pulse is assumed to have a negligible width and is represented by the impulse delta function, $p(t)$. The Dirac delta function $\delta(t)$ given below is 1 at $t = 0$ and 0 elsewhere:

$$p(t) = \sum \delta(t - nT) \tag{1.1}$$

Figure 1.2 shows such an impulse delta function. Note that $p(t)$is zero except at the sampling instant nT. The discrete time signal $x(t)$ is then represented by the following formula [1, 2]:

$$x(t) = x_a(t)p(t) \tag{1.2}$$

$$= \sum x_a(nT)\delta(t - nT) \tag{1.3}$$

where $x_a(t)$ is a corresponding analog signal. The discrete signal $x(t)$ only has the sequence of values $x(T)$, $x(2T)$, $x(3T)$, and so forth, assuming that the sampling time is greater than zero.

1.3 Fourier Series Representation

Since $p(t)$ is a periodic function with period T, it can be represented by a Fourier series given below [2]:

$$p(t) = \sum a_n e^{jn\Omega_s t} \tag{1.4}$$

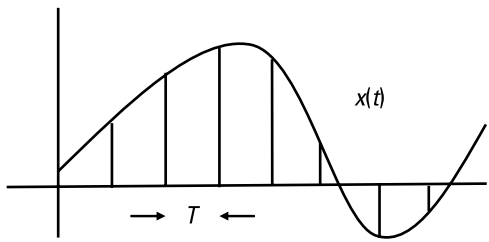


Figure 1.1 An analog signal $x(t)$ sampled with period T.

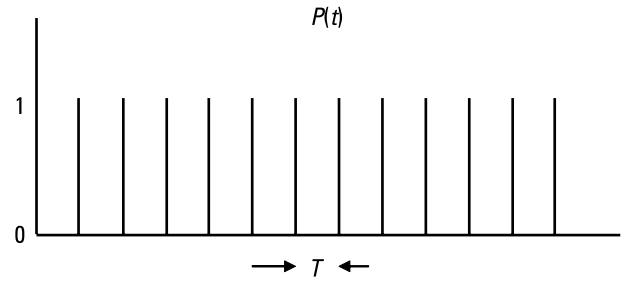


Figure 1.2 Impulse delta function.

where Ω_S is the radian sampling frequency defined as

$$\Omega_s = 2\pi f_s = \frac{2\pi}{T} \tag{1.5}$$

The coefficient a_n is in general complex. From (1.4), it is clear that $p(t) = p(t + kT)$ where k is an integer. Multiplying both sides of (1.4) by $e^{-jm\Omega_s t}$ and integrating from $-T/2$ to $T/2$, we have

$$\int_{-T/2}^{T/2} p(t)e^{-jm\Omega_s t}\,dt = \int_{-T/2}^{T/2} a_m\,dt + \int_{-T/2}^{T/2}\left(\sum_{n\neq m} a_m e^{-j(m-n)\Omega_s t}\right)dt \tag{1.6}$$

The second term on the right side of (1.6) integrates to 0 for $m \neq n$. The coefficient a_n is then given by

$$a_n = \frac{1}{T}\int_{-T/2}^{T/2} p(t)e^{-jn\Omega_s t}\,dt \tag{1.7}$$

Substituting (1.1) into (1.7), we have $a_n = 1/T$. The impulse function in (1.4) then becomes

$$p(t) = \sum\left(\frac{1}{T}\right)e^{jn\Omega_s t} \tag{1.8}$$

Substituting (1.8) to (1.2), the discrete time function $x(t)$ is then given by

$$x(t) = \left(\frac{1}{T}\right)\sum x_a(t)e^{jn\Omega_s t} \tag{1.9}$$

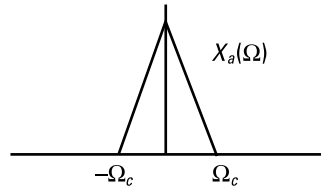


Figure 1.3 Band-limited spectrum of the analog signal $x_a(t)$.

Equation (1.9) is the Fourier series representation of the discrete time signal $x(t)$. Let f represent the frequency and then the analog radian frequency $\Omega = 2\pi f$. The Fourier transform of $y(t)e^{jat}$ is just $Y(\Omega - a)$. Assuming the Fourier transform of $x_a(t)$ to be $X_a(\Omega)$, then the Fourier transform of $x_a(t)e^{jn\Omega_s t}$ becomes $X_a(\Omega - n\Omega_s)$. By taking the Fourier transform on both sides of (1.9), the frequency-domain relationship given by (1.10) is obtained:

$$X(\Omega) = \left(\frac{1}{T}\right)\sum X_a\left(\Omega - n\Omega_s\right) \tag{1.10}$$

Equation (1.10) shows that the sampled spectrum is a summation of the shifted spectrum of the analog signal by $n\Omega_s$. In other words, the sampled spectrum is periodic in the frequency domain with period equal to the sampling frequency Ω_s. Figure 1.3 shows an analog spectrum with cutoff frequency Ω_c.

Figure 1.4 shows the sampled spectrum under the assumption that $\Omega_s > \Omega_c$. The spectrum is periodic with a period of Ω_s. The analog spectrum can be completely recovered without aliasing in the baseband. However, there will be aliasing and spectral distortion if $\Omega_s < \Omega_c$. According to Figure 1.4, the minimum sampling rate such that no spectral distortion occurs is to meet the following condition:

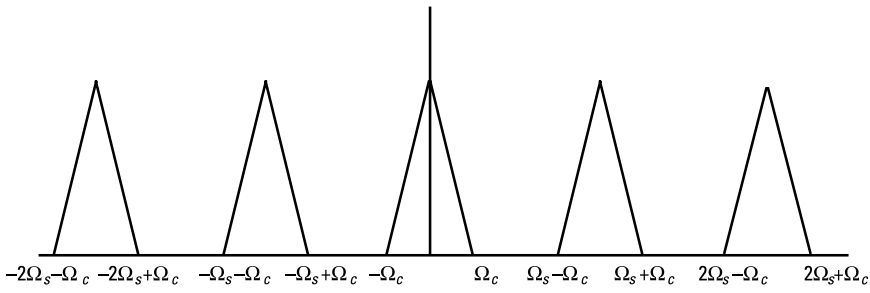


Figure 1.4 Sampled spectrum of the analog signal $x_a(t)$.

$$n\Omega_s - \Omega_c = (n-1)\Omega_s + \Omega_c \tag{1.11}$$

After some simple algebra from (1.11), the minimum sampling rate is given here:

$$\text{Minimum of } \Omega_s = 2\Omega_c \tag{1.12}$$

Equation (1.12) is the well-known result and states that the minimum sampling rate must be twice the cutoff frequency of the analog signal to completely reconstruct the original signal without any spectral distortion. The sampling rate given in (1.12) is also the Nyquist rate.

1.4 DFT

The DFT [1, 2] is derived by first taking the Fourier transform on both sides of (1.3) to have the following:

$$\int_{-\infty}^{\infty} x(t)e^{-j\Omega t}\,dt = \int_{-\infty}^{\infty}\sum_{-\infty}^{\infty} x_a(nT)\delta(t-nT)e^{-j\Omega t}\,dt \tag{1.13}$$

Expressing the left term by $X(\Omega)$, we have

$$\begin{aligned} X(\Omega) &= \sum x_a(nT)\int \delta(t-nT)e^{-j\Omega t}\,dt \\ &= \sum x_a(nT)e^{-j\Omega nT} \end{aligned} \tag{1.14}$$

Defining a radian frequency ω which is related to the analog frequency Ω through the following relationship:

$$\omega = \Omega T \tag{1.15}$$

Using (1.15), (1.14) then becomes

$$\begin{aligned} X(\omega) &= \sum x_a(nT)e^{-j\omega n} \\ &= \sum x_n e^{-j\omega n} \end{aligned} \tag{1.16}$$

where we have represented $x_a(nT)$ by x_n which is the nth sample of the analog signal $x_a(t)$. Equation (1.16) is the Fourier transform of the discrete samples x_n.

The Fourier transform $X(\omega)$ is periodic with period of 2π. The ω can be divided into N spectral components. Since ω ranges from 0 to 2π, ω can be written as

$$\omega = \frac{2\pi k}{N} \qquad k = 0,1,\ldots,N-1 \tag{1.17}$$

where k is the kth radian frequency component and ranges from 0 to $N-1$. Substituting (1.17) into (1.16), we have

$$X_k = \sum_{n=0}^{N-1} x_n e^{-j2\pi nk/N} \qquad k = 0,\ldots,N-1 \tag{1.18}$$

Equation (1.18) defines the DFT of the discrete time sequence x_n. The sequence X_k is periodic with a period of N as evidenced by the fact that $X_k = X_{k+N}$. Therefore, there are only N unique values of X_k.

To obtain the coefficients $x(n)$ from X_k, we multiply both sides of (1.18) by the complex exponential $e^{j2\pi mk/N}$ and summing k from 0 to $N-1$ and obtain the following:

$$\sum_{k=0}^{N-1} X_k e^{j2\pi mk/N} = \sum_{k=0}^{N-1}\sum_{n=0}^{N-1} x_n e^{-j2\pi nk/N} e^{j2\pi mk/N} \tag{1.19}$$

Interchanging the order of summation on the right side of (1.19), we have

$$\sum_{k=0}^{N-1} X_k e^{j2\pi mk/N} = \sum_{n=0}^{N-1} x_n \left(\sum_{k=0}^{N-1} e^{-j2\pi(n-m)k/N} \right) \tag{1.20}$$

Equation (1.20) can be simplified using the following relationship:

$$\sum_{k=0}^{N-1} e^{-j2\pi rk/N} = \begin{cases} N & \text{for } r = jN, j \text{ an integer} \\ 0 & \text{otherwise} \end{cases} \tag{1.21}$$

Equation (1.20) then becomes

$$x_m = \frac{1}{N} \sum_{k=0}^{N-1} X_k e^{j2\pi mk/N} \qquad m = 0,\ldots,N-1 \tag{1.22}$$

Replacing the index m by n, the following equation results:

$$x_n = \frac{1}{N}\sum_{k=0}^{N-1} X_k e^{j2\pi nk/N} \qquad n = 0,\ldots,N-1 \tag{1.23}$$

The sequence x_n is also periodic with a period N and $x_n = x_{n+N}$. Therefore, there are only N unique values of x_n. Equations (1.18) and (1.23) constitute the DFT pairs.

1.5 Sampling Theorem and Signal Interpolation

To derive the sampling theorem [1–3], we start with the following continuous Fourier transform:

$$x_a(t) = \left(\frac{1}{2\pi}\right)\int X_a(\Omega)e^{j\Omega t}\, d\Omega \tag{1.24}$$

From (1.12), the minimum sampling rate is $\Omega_s = 2\Omega_c$. This gives $\Omega_c = \Omega_s/2 = \pi/T$. In the baseband, the range of Ω is then $-\pi/T < \Omega < \pi/T$ from Figure 1.3. From (1.10), $X_a(\Omega) = TX(\Omega)$ inside this baseband and the following equation results:

$$x_a(t) = \left(\frac{T}{2\pi}\right)\int_{-\pi/T}^{\pi/T} X(\Omega)e^{j\Omega t}\, d\Omega \tag{1.25}$$

Substituting (1.14) into (1.25), we have

$$x_a(t) = \left(\frac{T}{2\pi}\right)\int_{-\pi/T}^{\pi/T} \sum x_n e^{-j\Omega nT} e^{j\Omega t} d\Omega \tag{1.26}$$

Interchanging the order of integration and summation in (1.26), the following results:

$$\begin{aligned} x_a(t) &= \left(\frac{T}{2\pi}\right)\int_{-\pi/T}^{\pi/T} \sum x_n e^{-j\Omega nT} e^{j\Omega t} d\Omega \\ &= \left(\frac{T}{2\pi}\right)\sum x_n \left(\int_{-\pi/T}^{\pi/T} e^{j\Omega(t-nT)}\, d\Omega\right) \end{aligned} \tag{1.27}$$

The integral inside the parentheses of (1.27) can be evaluated to give

$$x_a(t) = \sum x(nT) \frac{\sin\left[\left(\frac{\pi}{T}\right)(t - nT)\right]}{\left(\frac{\pi}{T}\right)(t - nT)} \tag{1.28}$$

Equation (1.28) is a well-known sampling theorem. It is an interpolation formula that shows how an analog signal can be reconstructed from the discrete samples $x(nT)$. The Nyquist rate is used in order to get this interpolation formula. To prevent aliasing, the input signal must be band-limited between $-\pi/T$ and π/T.

The interpolation filter in (1.28) is a sinc function that has an infinite duration. In reality, this function is truncated in order for computation. The spectrum of the sinc function has a rectangular shape that will be discussed in detail in Chapter 2.

1.6 Properties of the DFT

The DFT given by (1.18) and (1.23) are given again here:

$$X_k = \sum_{n=0}^{N-1} x_n e^{-j2\pi nk/N} \qquad k = 0,1,\ldots,N-1 \tag{1.29}$$

$$x_n = \frac{1}{N} \sum_{k=0}^{N-1} X_k e^{j2\pi nk/N} \qquad n = 0,1,\ldots,N-1 \tag{1.30}$$

Assume that the input has a real sequence x_n. The real and imaginary parts of X_k are then given by

$$\mathrm{Re}\left(X_k\right) = \sum_{n=0}^{N-1} x_n \cos\left(\frac{2\pi nk}{N}\right) \tag{1.31a}$$

$$\mathrm{Im}\left(X_k\right) = -\sum_{n=0}^{N-1} x_n \sin\left(\frac{2\pi nk}{N}\right) \tag{1.31b}$$

The $\mathrm{Re}(X_k)$ and $\mathrm{Im}(X_k)$ satisfy the following relationship:

$$\mathrm{Re}\left(X_k\right) = \mathrm{Re}\left(X_{-k}\right) \tag{1.32}$$

$$\mathrm{Im}\left(X_k\right) = -\mathrm{Im}\left(X_{-k}\right) \tag{1.33}$$

If the input sequence is real, the real part of X_k is then an even function of the frequency while the imaginary part of X_k is an odd function of the frequency.

The power spectrum of the kth frequency component is the sum of the square of the real and imaginary parts of X_k and is given here:

$$P_k = \left[\mathrm{Re}\left(X_k\right)\right]^2 + \left[\mathrm{Im}\left(X_k\right)\right]^2 \tag{1.34}$$

Using (1.32) and (1.33), P_k satisfies the following relationship:

$$P_k = P_{-k} \tag{1.35}$$

Therefore, the power spectrum of a real sequence is an even function of the frequency.

In the frequency domain, the DFT X_k is also periodic with period N. Using (1.29), the following relationship results:

$$\begin{aligned} X_{N-k} &= \sum_{n=0}^{N-1} x_n e^{-j2\pi n(N-k)/N} \\ &= \sum_{n=0}^{N-1} x_n e^{j2\pi nk/N} \\ &= X_{-k} \end{aligned} \tag{1.36}$$

Changing k by $-k$ in (1.36) we have the familiar relationship of $X_k = X_{N+k}$ to confirm the periodicity of X_k.

Using (1.32) and (1.33), we can write X_{-k}as follows:

$$\begin{aligned} X_{-k} &= \mathrm{Re}\left(X_{-k}\right) + j\mathrm{Im}\left(X_{-k}\right) \\ &= \mathrm{Re}\left(X_k\right) - j\mathrm{Im}\left(X_k\right) \\ &= \left(X_k\right)^* \end{aligned} \tag{1.37}$$

Using both (1.36) and (1.37), we have for a real sequence

$$X_{N-k} = \left(X_k\right)^* \tag{1.38}$$

The following relationship is true for both N odd and N even:

$$X_{(N-1)/2} = \left(X_{(N+1)/2}\right)^* \qquad k = \frac{(N+1)}{2} \text{ and } N \text{ odd} \tag{1.39a}$$

$$X_{N/2} = \left(X_{N/2}\right)^* \qquad k = \frac{N}{2} \text{ and } N \text{ even} \tag{1.39b}$$

Therefore, for N even, the spectrum at the middle frequency is real. Using (1.34) and (1.36), it can be seen that $P_{N-k} = P_{-k}$. Using (1.35) for a real sequence, then $P_k = P_{N-k}$. An example is shown in Figure 1.5 for $N = 8$ and we have $P_7 = P_1$, $P_6 = P_2$, and $P_5 = P_3$.

If the input sequence $x_n$is complex, then the DFT X_k is, in general, complex. However, if the X_k sequence is complex, then the x_n sequence is also complex.

1.7 Time and Frequency Relationship of the DFT

The DFT has an interesting but important time and frequency relationship [2]. Figure 1.6 is a typical plot of the input sequence in the time domain. Assuming that the sampling period is T and there are N samples, the time duration L_t is then given by

$$L_t = NT \tag{1.40}$$

$$d_t = T \tag{1.41}$$

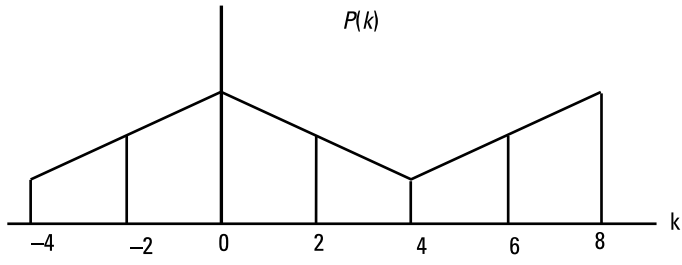


Figure 1.5 Power spectrum $P(k)$ for k from –4 to 8.

where d_t is the time duration between two consecutive samples in the time domain and is also equal to the sampling time T. Similarly, Figure 1.7 is a typical plot of the DFT in the frequency domain. The bandwidth is then given by

$$L_f = \frac{1}{T} \tag{1.42}$$

$$d_f = \frac{1}{NT} \tag{1.43}$$

where d_f is the frequency width between two consecutive samples in the frequency domain. The bandwidth is also equal to the sampling frequency $1/T$.

Equations (1.41) to (1.44) also illustrates the following relationship:

$$L_t = \frac{1}{d_f} \tag{1.44}$$

$$d_t = \frac{1}{L_f} \tag{1.45}$$

From (1.44), the time duration L_t is the inverse of the frequency increment d_f. From (1.45), the time increment d_t is the inverse of the bandwidth L_f.

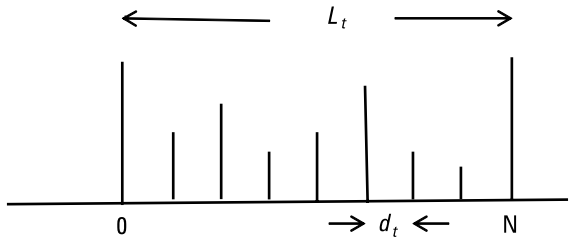


Figure 1.6 Discrete input sequence in the time domain.

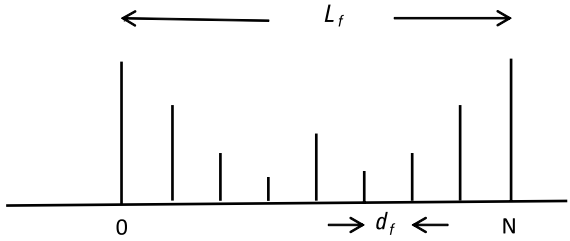


Figure 1.7 DFT in the frequency domain.

Therefore, as the frequency increment decreases, the time duration increases. However, as the bandwidth decreases, the time increment increases. These two relationships are useful to define the characteristics of the time sequence when the bandwidth is known.

1.8 Operations of the DFT

The DFT has a few important operations [1, 2] that are useful. Four operations, linearity, time shift, frequency shift, and convolution, are considered here. For ease of presentation, the factor W_N is defined here:

$$W_N = e^{j2\pi/N} \tag{1.46}$$

1.8.1 Linearity

Consider two periodic sequences, x_n and y_n. The DFT of x_n is X_k and the DFT of y_n is Y_k. The two periodic sequences can be linearly combined to generate a third periodic sequence $q_n = ax_n + by_n$ where a and b are constants. Then clearly, the DFT of $ax_n + by_n$ is $aX_k + bY_k$.

1.8.2 Time Shift

Assume that a periodic sequence x_n is shifted by m to generate $y_n = x_{n-m}$. Then its DFT Y_k is given here:

$$Y_k = X_k W_N^{-mk} \tag{1.47}$$

This can be shown by finding the inverse DFT of Y_k:

$$\begin{aligned} y_n &= \left(\frac{1}{N}\right)\sum_{k=0}^{N-1} Y_k W_N^{nk} = \left(\frac{1}{N}\right)\sum_{k=0}^{N-1} X_k W_N^{nk} W_N^{-mk} \\ &= \left(\frac{1}{N}\right)\sum_{k=0}^{N-1} X_k W_N^{(n-m)k} \\ &= x_{n-m} \end{aligned} \tag{1.48}$$

Equation (1.47) implies that a time shift in the time domain is equivalent to multiplication by a phase factor in the frequency domain.

1.8.3 Frequency Shift

Assume that a periodic sequence x_n is multiplied by a phase factor to generate $y_n = x_n W_N^{nr}$. Then its DFT is given by

$$Y_k = X_{k-r} \tag{1.49}$$

This can be shown by finding the DFT of y_n:

$$\begin{aligned} Y_k &= \sum_{n=0}^{N-1} y_n W_N^{-nk} \\ &= \sum_{n=0}^{N-1} x_n W_N^{-nk} W_N^{nr} \\ &= \sum_{n=0}^{N-1} x_n W_N^{-n(k-r)} \\ &= X_{k-r} \end{aligned} \tag{1.50}$$

Equation (1.50) implies that a multiplication in the time domain by a phase factor is equivalent to a frequency shift in the frequency domain. This frequency shift is equivalent to a Doppler shift. Numerically, it is equal to $r/(NT)$.

1.8.4 Circular Convolution

Consider two periodic sequences x_n and y_n of period N. Then its circular convolution $q_n = \sum_{m=0}^{N-1} x_m y_{n-m}$ has the DFT of $X_k Y_k$. This can be derived by finding the inverse DFT of $X_k Y_k$:

$$\begin{aligned} q_n &= \left(\frac{1}{N}\right) \sum_{k=0}^{N-1} X_k Y_k W_N^{kn} \\ &= \left(\frac{1}{N}\right) \sum_{k=0}^{N-1} \left(\sum_{m=0}^{N-1} x_m W_N^{-km} \right) \left(\sum_{r=0}^{N-1} y_r W_N^{-kr} \right) W_N^{kn} \\ &= \left(\frac{1}{N}\right) \sum_{m=0}^{N-1} x_m \sum_{r=0}^{N-1} y_r \left(\sum_{k=0}^{N-1} W_N^{(n-m-r)k} \right) \end{aligned} \tag{1.51}$$

Using (1.21), the term in the parentheses is 0 unless $n - m - r = jN$. Therefore, $r = n - m$ and (1.51) becomes

$$q_n = \sum_{m=0}^{N-1} x_m y_{n-m} \tag{1.52}$$

The sequence q_n is then the circular convolution of the two periodic sequences x_n and y_n. Its DFT is then the product of X_k and Y_k. Instead of computing the sum given in (1.52), another way is to compute the product $X_k Y_k$ first. Its inverse DFT generates the convolution.

It is frequently necessary to find the linear convolution of the sequences x_n and y_n. For example, an input signal such as speech has to be lowpass-filtered to prevent aliasing. In this case, the two sequences of finite duration N are not periodic. However, its linear convolution can still be implemented using circular convolution. Since q_n in (1.52) has duration $2N - 1$, the circular convolution can be applied by appending $N - 1$ zeros to the end of both x_n and y_n.

More details on both linear and circular convolution can be found in [1].

1.9 *Z*-Transform

The z-transform [1, 2] is defined by the following equation:

$$X(z) = \sum_{n=-\infty}^{n=\infty} x_n z^{-n} \tag{1.53}$$

where z is a complex variable. If $z = e^{j\omega}$, then the z-transform is the same as the DFT given in (1.16):

$$X(w) = \sum x_n e^{-j\omega n} \tag{1.54}$$

The absolute value of z is 1. As ω varies from $-\pi$ to π, the z traverses along the unit circle from –1 to 1. The region of convergence is the range of z such that $\sum_{n=-\infty}^{n=\infty} \left| x_n z^{-n} \right| < \infty$.

Example 1.1

Consider the positive sequence given here:

$$x_n = \begin{cases} b^n & n \geq 0 \\ 0 & n < 0 \end{cases} \tag{1.55}$$

The z-transform is then given by

$$X(z) = \sum_{n=0}^{n=\infty} b^n z^{-n} = \sum_{n=0}^{n=\infty} \left(\frac{b}{z}\right)^n = \frac{z}{z-b} \tag{1.56}$$

The region of convergence is then $(b/z) < 1$ or $z > b$.

The two most important operations of the z-transform are time shift and convolution. Assume that a time sequence $y_n = x_{n-m}$ or y_n is a time shift of x_n by m samples. Its z-transform is then given by

$$\begin{aligned} Y(z) &= \sum_{n=-\infty}^{n=\infty} x_{n-m} z^{-n} = \sum_{k=-\infty}^{k=\infty} x_k z^{-(k+m)} \\ &= z^{-m} \sum_{k=-\infty}^{k=\infty} x_k z^{-k} = z^{-m} X(z) \end{aligned} \tag{1.57}$$

where the variable $k = n - m$ is defined. Therefore, a time shift by m in the time domain is equivalent to a multiplication of z^{-m} in the z domain.

Assume that x_n and h_n are two sequences. Their convolution is the sequence $y_n = \sum_{m=-\infty}^{m=\infty} x_m h_{n-m}$. The z-transform of y_n is then given by

$$Y(z) = \sum_{n=-\infty}^{n=\infty} \sum_{m=-\infty}^{m=\infty} x_m h_{n-m} z^{-n} \tag{1.58}$$

By defining a new variable $k = n - m$, (1.58) becomes

$$\begin{aligned} Y(z) &= \sum_{k=-\infty}^{k=\infty} \sum_{m=-\infty}^{m=\infty} x_m h_k z^{-m-k} \\ &= \left(\sum_{k=-\infty}^{k=\infty} h_k z^{-k} \right) \left(\sum_{m=-\infty}^{m=\infty} x_m z^{-m} \right) \\ &= X(z) H(z) \end{aligned} \tag{1.59}$$

Assume a linear system $H(z)$ whose time sequence is h_n. Passing through the input sequence x_n to $H(z)$ generates an output sequence y_n, which is the convolution of x_n and h_n. The z-transform of the output is simply the product of the input z-transform $X(z)$ and the system z-transform $H(z)$. This is also similar to the case for the DFT.

The system function described in (1.59) has the following general form:

$$H(z) = \frac{N(z)}{D(z)} \tag{1.60}$$

Both $N(z)$ and $D(z)$ are polynomials of z^{-1} and they may have different degrees. Assuming that $N(z)$ has the degree u and $D(z)$ has the degree v, they can be written as

$$N(z) = b_0 + b_1 z^{-1} + b_2 z^{-2} + \cdots + b_u z^{-u} \tag{1.61}$$

$$D(z) = 1 + a_1 z^{-1} + a_2 z^{-2} + \cdots + a_v z^{-v} \tag{1.62}$$

Substituting (1.60), (1.61), and (1.62) into (1.59), we have

$$\frac{Y(z)}{X(z)} = \frac{b_0 + b_1 z^{-1} + b_2 z^{-2} + \cdots + b_m z^{-u}}{1 + a_1 z^{-1} + a_2 z^{-2} + \cdots + a_n z^{-v}} \tag{1.63}$$

Equation (1.63) can be rewritten in the following summation form

$$Y(z) = \sum_{k=0}^{k=u} X(z) b_k z^{-k} - \sum_{k=1}^{k=v} Y(z) a_k z^{-k} \tag{1.64}$$

Using the time shift operations given in (1.57), (1.64) can be converted into the following difference equations

$$y_n = \sum_{k=0}^{k=u} b_k x_{n-k} - \sum_{k=1}^{k=v} a_k y_{n-k} \tag{1.65}$$

Equation (1.65) is realizable using tapped delay lines. More details on the implementation can be found in [2].

1.10 Summary

A discrete time signal is obtained by sampling an analog signal at a periodic rate. The DFT of these discrete samples is then derived. It was found that the sampled spectrum is a summation of the shifted spectrum of the analog signal. To reconstruct the original analog signal, the discrete samples must be taken at least at the Nyquist rate. For a band-limited analog signal with a cutoff

frequency f_c, the Nyquist rate is at least $2f_c$. Based upon the Nyquist rate, an interpolation formula was derived to reconstruct the analog signal.

The DFT has a relationship between time and frequency. The bandwidth in the frequency domain is the inverse of the sampling period in the time domain. The signal duration in the time domain is the inverse of the frequency increment in the frequency domain.

Three important operations of the DFT were illustrated: time shift, frequency shift, and convolution. A time shift in the time domain is equivalent to the multiplication by a phase factor in the frequency domain. A multiplication in the time domain by a phase factor is equivalent to a frequency shift in the frequency domain. This frequency shift is equivalent to a Doppler shift. The circular convolution of two discrete time sequences was defined. Its DFT is the product of the DFT of the two individual sequences.

The z-transform was also defined. It is useful in specifying the time-invariant system response. The output generated by passing the input through such a system can be implemented using a tapped delay line.

In the next chapter, single-carrier modulation is reviewed.

References

[1] Oppenheim, A. V., and R. W. Schafer, *Digital Signal Processing*, Upper Saddle River, NJ: Prentice Hall, 1975.

[2] Stanley, W. D., *Digital Signal Processing*, AU: Provide city of publication: Reston Publishing Company, 1975.

[3] Proakis, J. G., *Digital Communications*, New York: McGraw-Hill, 1983.

2

Single-Carrier Modulation

2.1 Introduction

OFDM is a multicarrier modulation technique. First, let us explain single-carrier modulation. From the name itself, it is clear there is only one carrier. Depending upon the type of modulation, the amplitude, phase, or frequency of this single carrier is modulated.

In digital communication, a sequence of digital bits is first grouped into bauds. Each baud is then mapped into a unique phase, amplitude, or frequency to modulate the carrier. The four most commonly applied modulation techniques are discussed. They are pulse amplitude modulation (PAM), phase shift keying (PSK), quadrature amplitude modulation (QAM), and frequency shift keying (FSK).

The analog signal can be reconstructed through interpolation and was discussed in Chapter 1. Two interpolation filters, rectangular and raised cosine, are given.

2.2 Data Transmission Rate

The transmission rate depends upon the number of bits k sent in a baud interval. A baud is defined to have k bits. The number k is an integer and is different

for different modulation schemes. As will be shown in later sections, $k = 1$, 2, 4, and 6 for binary phase shift keying (BPSK), quadrature phase shift keying (QPSK), 16-QAM, and 64-QAM, respectively.

The simplest case is to have $k = 1$ and there are only two possible levels. During each baud interval, there are only two possible amplitudes, phases, or frequencies according to PAM, PSK, or FSK. The resulting modulation is called binary modulation.

In general, there are $M = 2^k$ possible levels where k is an integer. In other words, k binary digits are mapped to M possible transmitted waveforms. These M possible levels are converted into complex numbers representing constellation points of the digital modulation. If the input bit rate is R bits/sec, then the baud rate is R/k. Assuming that the baud interval is T, then the resulting bit transmission rate is k/T.

2.3 Bandpass Signals

The modulated digital signal can in general be represented in the following form:

$$x(t) = a(t)\cos\left(2\pi f_c t + \theta(t)\right) \tag{2.1}$$

where $a(t)$ is the amplitude, $\theta(t)$ is the phase, and f_c is the carrier frequency of the modulated signal. Equation (2.1) can be represented in complex form as given here:

$$x(t) = \mathrm{Re}\left(a(t)e^{j\theta(t)}e^{j2\pi f_c t}\right) \tag{2.2a}$$

$$= \mathrm{Re}\left(s(t)e^{j2\pi f_c t}\right) \tag{2.2b}$$

where $s(t)$ is the lowpass signal bearing the information of the input digital bit stream

$$s(t) = a(t)e^{j\theta(t)} \tag{2.3}$$

The spectrum of $x(t)$ can be obtained by taking the Fourier transform on both sides of (2.2b) to obtain

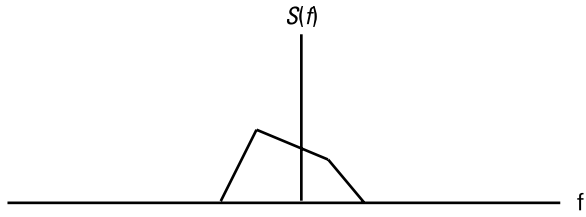


Figure 2.1 Spectrum of the lowpass signal $s(t)$.

$$\begin{aligned} X(f) &= \frac{1}{2}\int \left(s(t)e^{j2\pi f_c t} + s^*(t)e^{-j2\pi f_c t} \right) e^{-j2\pi f t}\, dt \\ &= \frac{1}{2}\left(S\left(f - f_c\right) + S^*\left(f + f_c\right) \right) \end{aligned} \tag{2.4}$$

where $S(f)$ is the spectrum of $s(t)$. Figure 2.1 is a plot of the lowpass signal $s(t)$ and Figure 2.2 is a plot of the bandpass signal $x(t)$. Clearly, $X(f)$ is a frequency shifted version of the baseband signal $x(t)$.

2.4 Digitally Modulated Signals

In each baud interval, $a(t)$, $\theta(t)$, or both can only take a set of discrete levels. Therefore, $s(t)$ can also be represented in the following form [1]:

$$s(t) = \sum_{n=-\infty}^{\infty} s_n q(t - nT) \tag{2.5}$$

where T is the baud interval, q is the pulse-shaping waveform, and s_n is the nth sample of the information bearing signal $s(t)$. The term s_n is, in general,

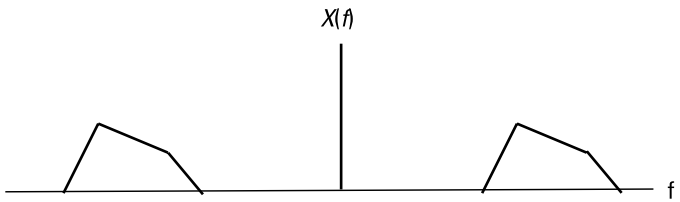


Figure 2.2 Spectrum of the bandpass signal $X(f)$.

complex and can be represented as

$$s_n = a_n e^{j\theta_n} \tag{2.6}$$

where a_n is the nth sample of $a(t)$ and θ_n is the nth sample of $\theta(t)$. The in-phase component I_n and the quadrature-phase component Q_n can then be written as

$$I_n = a_n \cos(\theta_n) \tag{2.7a}$$

$$Q_n = a_n \sin(\theta_n) \tag{2.7b}$$

Using (2.7a) and (2.7b), s_n defined in (2.6) can be written as

$$s_n = I_n + jQ_n \tag{2.8}$$

To make sure the average transmitted power is the same independent of the types of modulation, the amplitude a_n can be adjusted properly.

The $s_n$can only take one of possible M levels for each of the $k = \log_2 M$ input bits. Depending on how a_n and θ_n are changed in each baud interval T, different types of modulated digital signals are generated.

2.4.1 PAM

For PAM, only the amplitude a_n is changed while the phase θ_n remains constant. The simplest case is the on-off keying where $k = 1$ and $M = 2$. If the input bit is 1, there is a transmission. However, if the input bit is 0, there is no transmission. In general, there are M different amplitude levels for every k bits.

2.4.2 PSK

For PSK, the amplitude a_n remains constant while only the phase θ_n is changed. There are two popular PSK modulations. One is BPSK and the other is QPSK.

For BPSK, $k = 1$ and $M = 2$. Figure 2.3 shows the signal constellation. The phase angle is either 180° or 0° and s_n is either 1 or −1. Table 2.1 lists the complex amplitude s_n.

For QPSK, $k = 2$ and $M = 4$, and there are four possible phases. The phase angles can be encoded as 45°, 135°, 225°, and 315°. Figure 2.4 shows the signal constellation and Table 2.2 lists the complex amplitude $\sqrt{2}s_n$ [2]. The factor $\sqrt{2}$ is a constant to normalize the average energy.

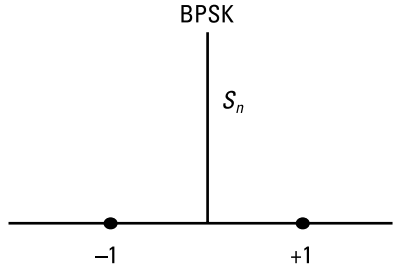


Figure 2.3 Signal constellation of BPSK.

Table 2.1
BPSK Encoding Table

Input Bits	s_n
0	–1
1	1

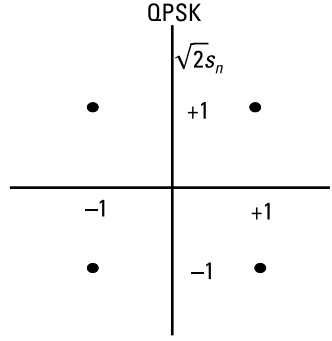


Figure 2.4 Signal constellation of QPSK.

Table 2.2
QPSK Encoding Table

Input bits	$\sqrt{2}s_n$
00	$-1-j$
01	$-1+j$
11	$1+j$
10	$1-j$

2.4.3 QAM

For QAM, both the amplitude a_n and the phase θ_n are changed. In the IEEE 802.11a standard [2], there are two types of QAM: 16-QAM and 64-QAM. Depending on the phase angle, the amplitude can be different to make sure that the average transmitted power remains the same.

In 16-QAM, $k = 4$ and $M = 16$. In other words, four input bits are encoded into 16 different discrete levels. Figure 2.5 shows the signal constellation while Table 2.3 lists the complex amplitude s_n/C_1 where C_1 is a normalization constant. The encoding of the constellation point is based upon the

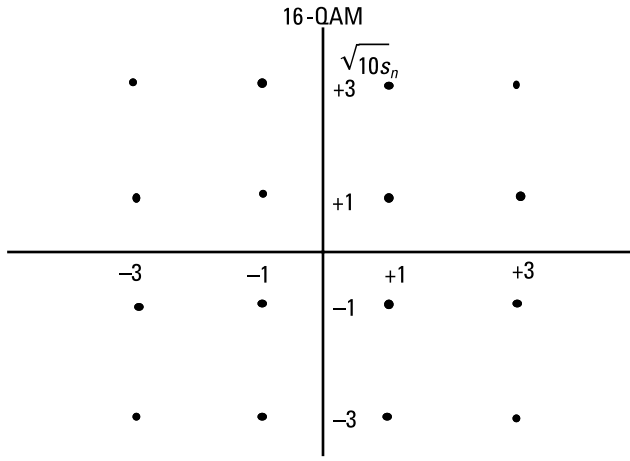


Figure 2.5 Signal constellation of 16-QAM.

Table 2.3
16-QAM Encoding Table

Input Bits	s_n/C_1	Input Bits	s_n/C_1
0000	$-3-3j$	1000	$3-3j$
0001	$-3-j$	1001	$3-j$
0010	$-3+3j$	1010	$3+3j$
0011	$-3+j$	1011	$3+j$
0100	$-1-3j$	1100	$1-3j$
0101	$-1-j$	1101	$1-j$
0110	$-1+3j$	1110	$1+3j$
0111	$-1+j$	1111	$1+j$

concept of Gray coding. In other words, any two adjacent constellation points differ by only 1 bit. For example, $3 + j$ encoded as 1011 and $3 - j$ encoded as 1001 are adjacent but differ only in the third bit.

In 64-QAM, $k = 6$ and $M = 64$. Therefore, six input bits are encoded into 64 discrete levels. Figure 2.6 shows the signal constellation while Table 2.4 lists the complex amplitude s_n/C_2 where C_2 is a normalization constant. The same Gray coding is applied.

The constant C_1 in Table 2.3 and the constant C_2 in Table 2.4 can be determined by normalizing the average energy to be a constant 1. To determine C_1 in Table 2.3 to normalize the average energy, the following is done:

$$\left(\frac{1}{16}\right)\sum_{i=0}^{15}\left|s_i\right|^2 = 1 \tag{2.9}$$

Table 2.4
64-QAM Encoding Table

Input Bits	s_n/C_2	Input Bits	s_n/C_2	Input Bits	s_n/C_2	Input Bits	s_n/C_2
000000	$-7-7j$	010000	$-1-7j$	100000	$7-7j$	110000	$1-7j$
000001	$-7-5j$	010001	$-1-5j$	100001	$7-5j$	110001	$1-5j$
000010	$-7-j$	010010	$-1-j$	100010	$7-j$	110010	$1-j$
000011	$-7-3j$	010011	$-1-3j$	100011	$7-3j$	110011	$1-3j$
000100	$-7+7j$	010100	$-1+7j$	100100	$7+7j$	110100	$1+7j$
000101	$-7+5j$	010101	$-1+5j$	100101	$7+5j$	110101	$1+5j$
000110	$-7+j$	010110	$-1+j$	100110	$7+j$	110110	$1+j$
000111	$-7+3j$	010111	$-1+3j$	100111	$7+3j$	110111	$1+3j$
001000	$-5-7j$	011000	$-3-7j$	101000	$5-7j$	111000	$3-7j$
001001	$-5-5j$	011001	$-3-5j$	101001	$5-5j$	111001	$3-5j$
001010	$-5-j$	011010	$-3-j$	101010	$5-j$	111010	$3-j$
001011	$-5-3j$	011011	$-3-3j$	101011	$5-3j$	111011	$3-3j$
001100	$-5+7j$	011100	$-3+7j$	101100	$5+7j$	111100	$3+7j$
001101	$-5+5j$	011101	$-3+5j$	101101	$5+5j$	111101	$3+5j$
001110	$-5+j$	011110	$-3+j$	101110	$5+j$	111110	$3+j$
001111	$-5+3j$	011111	$-3+3j$	101111	$5+3j$	111111	$3+3j$

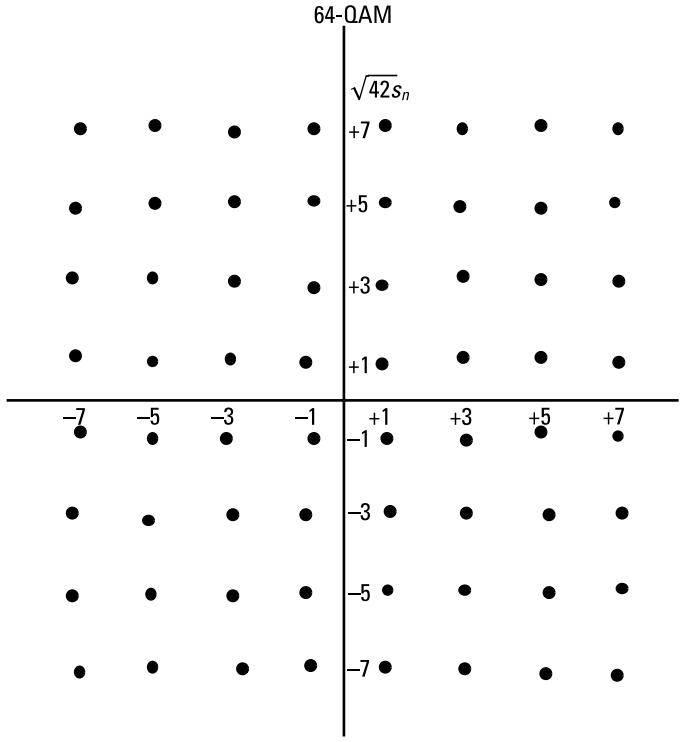


Figure 2.6 Signal constellation of 64-QAM.

The constant C_1 is then found to be $1/\sqrt{10}$. Similarly, the constant C_2 is found to be $1/\sqrt{42}$.

2.4.4 FSK

For FSK, the information s_n is used to change the frequency of the signal waveform $s(t)$. Assuming the frequency deviation is given by Δfs_n, the $s(t)$ is then given by [1]

$$s(t) = \sum_{n=-\infty}^{\infty} e^{j2\pi(\Delta f)ts_n} q(t - nT) \tag{2.10}$$

Since $s_n$can take M different levels, there are M possible frequencies from which to choose. The end result is the transmitted carrier frequency is shifted based upon the input information s_n.

2.5 Pulse Shaping

The analog signal $x_a(t)$ can be reconstructed from the discrete samples $x(nT)$ according to the sampling theorem given by (1.28). That equation is rewritten here:

$$x_a(t) = \sum x(nT) \frac{\sin\left[\left(\frac{\pi}{T}\right)(t - nT)\right]}{\left(\frac{\pi}{T}\right)(t - nT)} \tag{2.11}$$

Assume that the signal cutoff frequency is $f = f_c$ as shown in Figure 1.3. The minimum sampling rate is the Nyquist rate or $2f_c$. By setting the baud period $T = 1/2f_c$, (2.11) becomes

$$x_a(t) = \sum x\left(\frac{n}{2f_c}\right) \frac{\sin\left[\left(2\pi f_c\right)\left(t - n/2f_c\right)\right]}{\left(2\pi f_c\right)\left(t - n/2f_c\right)} \tag{2.12}$$

Equivalently, each discrete sample is multiplied by a pulse-shaping function given here:

$$q(t) = \frac{\sin\left(\frac{\pi t}{T}\right)}{\frac{\pi t}{T}} \tag{2.13}$$

The pulse $q(t)$, which is defined in (2.13), is 1 at $t = 0$ and 0 at $t = nT$ where n is an integer. As can be seen from (2.11), $x_a(kT) = x(kT)$. Therefore, to reconstruct an analog signal $x_a(t)$ without intersymbol interference (ISI), the following condition must be satisfied:

$$q(kT) = \begin{cases} 1 & k = 0 \\ 0 & k \neq 0 \end{cases} \tag{2.14}$$

Equation (2.11) is just the convolution of the discrete signal $x(t)$ with the pulse $q(t)$. Therefore, the analog spectrum is the product of the pulse spectrum and the signal spectrum. In order to have no spectral distortion, the ideal frequency spectrum of the function $q(t)$ is the rectangular function, $Q(\Omega)$ given here:

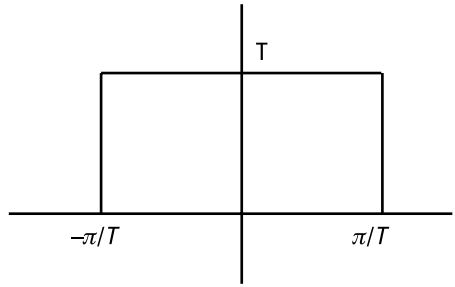


Figure 2.7 The frequency spectrum of the sinc function versus $\Omega = 2\pi f$.

$$Q(\Omega) = \begin{cases} T & -\dfrac{\pi}{T} < \Omega < \dfrac{\pi}{T} \\ 0, & \text{otherwise} \end{cases} \tag{2.15}$$

The function $Q(\Omega)$ is also plotted in Figure 2.7. As shown next, the inverse Fourier transform of (2.15) generates $q(t)$:

$$\begin{aligned} q(t) &= \left(\frac{T}{2\pi}\right) \int_{-\pi/T}^{\pi/T} Q(\Omega) e^{j\Omega t}\, d\Omega \\ &= \left(\frac{T}{2\pi}\right) \int_{-\pi/T}^{\pi/T} e^{j\Omega t}\, d\Omega \\ &= \frac{\sin\left(\dfrac{\pi t}{T}\right)}{\dfrac{\pi t}{T}} \end{aligned} \tag{2.16}$$

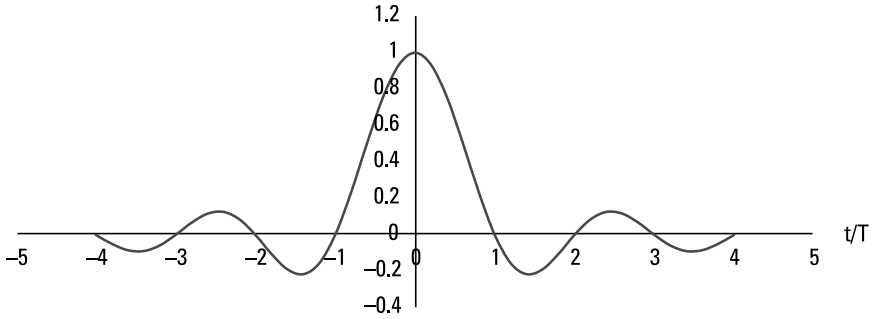


Figure 2.8 Pulse having a sinc function.

Figure 2.8 is a plot of sinc function given in (2.16). As can be seen, $q(t)$ is 1 at $t = 0$ and 0 when t/T has integral values.

Even though the sinc function in the time domain generates no ISI, it has infinite duration and cannot be realized. In practice, the sinc function is truncated to approximate the ideal function. If enough side lobes are kept, negligible ISI can result.

The sinc function drops at the rate of $1/t$. One method frequently used in digital communication is to use the raised cosine function defined next [1]:

$$q(t) = \frac{\sin\left(\frac{\pi t}{T}\right)}{\frac{\pi t}{T}} \frac{\cos\left(\frac{\beta \pi t}{T}\right)}{\frac{1 - 4\beta^2 t^2}{T^2}} \tag{2.17}$$

Its spectrum $Q(f)/T$ has the following closed form [1]:

$$\frac{Q(f)}{T} = \begin{cases} 1 & 0 \le |f| \le \frac{(1-\beta)}{2T} \\ 0.5\left(1 - \sin\left(\frac{\pi T\left(f - \frac{1}{2T}\right)}{\beta}\right)\right) & \frac{(1-\beta)}{2T} \le |f| \le \frac{(1+\beta)}{2T} \end{cases} \tag{2.18}$$

Figure 2.9 is a plot of the raised function against t/T for $\beta = 0$, 0.5, and 1. For $\beta = 0$, the raised cosine function is the same as the sinc function. As β increases, the side lobe also decreases. For $\beta = 1$, the side lobe is negligibly

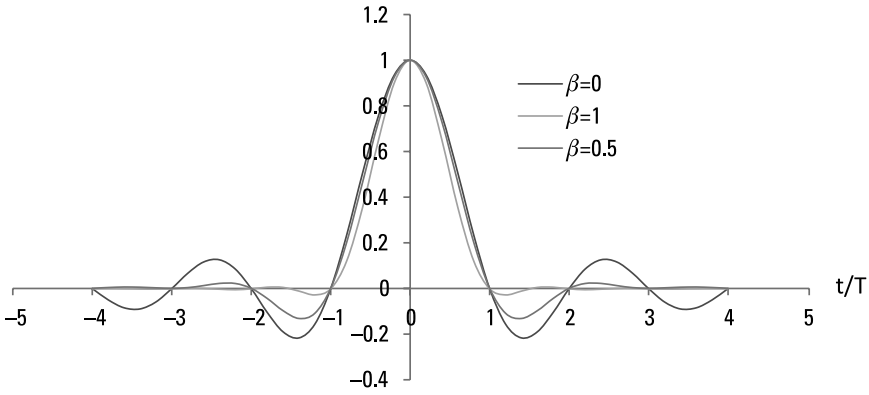


Figure 2.9 Pulse having a raised cosine function.

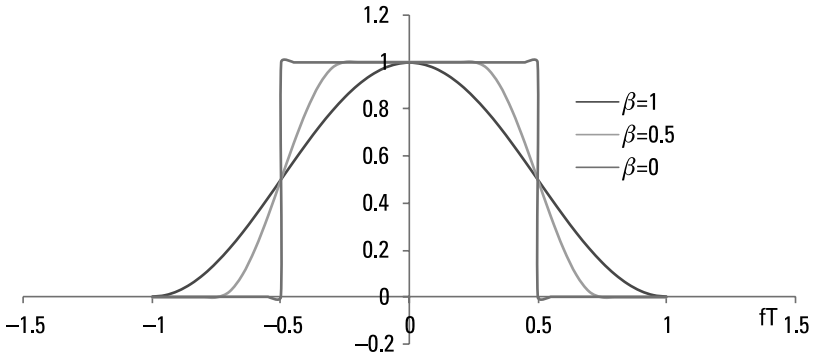


Figure 2.10 Frequency spectrum of the raised cosine function versus *fT*.

small after the main lobe. Figure 2.10 is a plot $Q(f)/T$ against fT. When $\beta = 0$, the spectrum converges to the ideal rectangle with bandwidth $fT = 0.5$ or $\Omega = 2\pi f = \pi/T$.

The raised cosine function also satisfies the conditions for reconstruction without ISI as defined in (2.14).

2.6 Summary

In digital communication, the traditional way is to have only one carrier to carry the data information. A sequence of k input bits is first grouped to generate a baud that has a total of $M = 2^k$ levels. Each level is mapped onto the phase, amplitude, or frequency of this single carrier for transmission.

Four modulation techniques, PAM, PSK, QAM, and FSK were presented. In PAM, only the amplitude is modulated. In PSK, only the phase is modulated. If $M = 2$ or $k = 1$, it is BPSK. If $M = 4$ or $k = 2$, it is QPSK. In QAM, both amplitude and phase are modulated. If $M = 16$ or $k = 4$, it is 16-QAM. If M increases to 64 or $k = 6$, it is then 64-QAM. In FSK, M different frequencies corresponding to k bits can be selected to change the carrier frequency.

From the discrete digital samples, the analog signal can be reconstructed by applying an interpolation filter. Two pulse-shaping waveforms were given. One has a rectangular spectrum implying a sinc function in the time domain. This sinc function has an infinite duration requiring truncation for implementation. To reduce the side lobes for fast convergence, a raised cosine function can be applied. A parameter β was defined in this function to control the rate

of convergence. For $\beta = 0$, it is the same as the sinc function. As β increases, the side lobe also decreases.

In the next chapter, multicarrier modulation is introduced.

References

[1] Proakis, J. G., *Digital Communications*, New York: McGraw-Hill, 1983.

[2] Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications, IEEE Standard 802.11a, 2007.

3

Multicarrier Modulation

3.1 Introduction

OFDM is a modulation technique that applies multiple carriers to be modulated by an input data stream. In a single-carrier modulation, the input data is carried by only one carrier and the modulation is carried out in the time domain. In a multicarrier modulation, the input data is split into bauds and carried by multiple subcarriers and the modulation is carried out in the frequency domain. Any two subcarriers are orthogonal in a symbol interval. This is also why OFDM is so named.

A transmission through a multipath channel can cause ISI and signal distortion. One characteristic of OFDM is its insertion of guard interval between adjacent symbols to eliminate this degradation. This guard interval is also called cyclic prefix to maintain the orthogonality of the subcarriers. As will become clear in later chapters, the multipath degradation and fading can be removed much more easily in the frequency domain using OFDM.

Even though it is easier to remove the multipath interference using OFDM, there are also some disadvantages. One is the peak-to-average power ratio (PAPR). This is because the peak power can be much higher than the average power. When this happens, the OFDM signal is clipped to cause distortion. In other words, the efficiency of the power amplifier is reduced.

Two other major characteristics of the OFDM are the FFT and the classical OFDM spectrum. A DFT is an essential tool used in this multicarrier modulation. However, the FFT is a fast computation technique of DFT. Except for the computation speed, there is no difference between the DFT and the FFT. The classical OFDM spectrum is a summation of the sinc function from each subcarrier. It is a sinc function because of the finite window length in the time domain.

To understand the OFDM waveform, we therefore start with the DFT. Subsequently, cyclic prefix, FFT, classical OFDM spectrum, PAPR, and orthogonality are all analyzed. After discussing the OFDM characteristics, time and frequency relationship, window function, and digital modulation are given in sequence in the subsequent sections.

3.2 OFDM Waveform

In a multicarrier waveform such as OFDM, each carrier is modulated by k bits having one of $M = 2^k$ possible levels. This waveform is defined by the inverse discrete Fourier transform given in (1.30) and is rewritten here:

$$x_n = \frac{1}{N}\sum_{k=0}^{N-1} X_k e^{j2\pi nk/N} \qquad n = 0,1,\ldots, N-1 \tag{3.1}$$

In (3.1), k is the frequency index, n is the time index, and X_k is the kth frequency component. During the modulation process, the amplitude, phase, or frequency of X_k is modified. Therefore, the whole process is performed in the frequency domain in contrast to the time-domain modulation of the single-carrier waveform.

To get more insight into the multicarrier waveform, define the kth radian frequency component given here:

$$w_k = \frac{2\pi k}{N} \tag{3.2}$$

Substituting (3.2) into (3.1), we have

$$x_n = \frac{1}{N}\sum_{k=0}^{N-1} X_k e^{jnw_k} \qquad n = 0,1,\ldots,N-1 \tag{3.3}$$

In general, X_k is complex and can be written as

$$X_k = A_k e^{j\Psi_k} \tag{3.4}$$

Substituting (3.4) into (3.3), we have

$$x_n = \left(\frac{1}{N}\right)\sum_{k=0}^{N-1} A_k e^{j\left(nw_k+\Psi_k\right)} \tag{3.5}$$

It is clear from (3.5) that there are N carriers. These carriers are also called subcarriers. Each subcarrier has an amplitude A_k, a phase Ψ_k, and a radian frequency w_k.

The discrete samples x_n can be reconstructed to generate $x(t)$ as long as the sampling rate is at least twice the baseband cutoff frequency. Assuming that the radio frequency (RF) carrier frequency is f_c, then the transmitted analog waveform for the jth OFDM symbol is given by

$$y(t) = \mathrm{Re}\left(x(t)e^{j2\pi f_c t}\right) \qquad jNT < t < (j+1)NT \tag{3.6}$$

In the same way as the single-carrier case, the input bit stream is grouped into bauds. Each baud has k bits where $k = \log_2 M$ and M is the total number of levels. Each baud of k bits is then mapped onto one of M possible phases, frequencies, or amplitudes.

3.3 OFDM Characteristics

As mentioned in the introduction, the most important characteristics of OFDM are orthogonality, sinc spectrum, PAPR, and FFT. Each of these subjects is discussed next.

3.3.1 Orthogonality

In OFDM, any two subcarriers are orthogonal. To show that, assume any two subcarriers, y_i and y_k, given below:

$$y_k = X_k e^{j2\pi kn/N} \tag{3.7}$$

$$y_i = X_i e^{j2\pi in/N} \tag{3.8}$$

Then the following relationship must be true:

$$\left(\frac{1}{N}\right)\sum_{n=0}^{N-1} y_k y_i^* = \begin{cases} 0 & \text{if } k \neq i \\ |X_k|^2 & \text{if } k = i \end{cases} \tag{3.9}$$

Substituting (3.7) and (3.8) into (3.9), we have

$$\begin{aligned}\left(\frac{1}{N}\right)\sum_{n=0}^{N-1} y_k y_i^* &= \left(\frac{1}{N}\right)\sum_{n=0}^{N-1} X_k X_i^* e^{j2\pi n(k-i)/N} \\ &= \left(\frac{1}{N}\right) X_k X_i^* \left(\sum_{n=0}^{N-1} e^{j2\pi n(k-i)/N}\right)\end{aligned} \tag{3.10}$$

However, the term inside the brackets is 0 unless $k = i$ as given in (1.21); we then have

$$\left(\frac{1}{N}\right)\sum_{n=0}^{N-1} y_k y_i^* = \begin{cases} 0 & \text{if } k \neq i \quad (3.11) \\ |X_k|^2 & \text{if } k = i \quad (3.12) \end{cases}$$

Therefore, any two subcarriers are orthogonal in OFDM.

3.3.2 OFDM Spectrum

OFDM transmits a large number of subcarriers and each subcarrier has the same spacing of $2\pi/N$ in radians or f_c/N in frequency. Assume each subcarrier is not modulated, then X_k is 1. In the digital domain, each subcarrier then has the form $(e^{j2\pi nk/N})/N$. Converting it to the analog domain, it has the equivalent form $(e^{j2\pi kf_s t/N})/N$ where f_s is the sampling frequency. Since the DFT interval is NT, the classical spectrum $X(f)$ is then given here:

$$X(f) = \left(\frac{1}{N}\right)\sum_{k=0}^{N-1} \int_{-NT/2}^{NT/2} e^{j2\pi kf_s t/N} e^{-j2\pi ft}\,\mathrm{d}t \tag{3.13}$$

The integral inside the summation of (3.13) can be easily evaluated to give the following:

$$\frac{X(f)}{T} = \sum_{k=0}^{N-1} \frac{\sin(\pi fNT - \pi k)}{\pi fNT - \pi k} \tag{3.14}$$

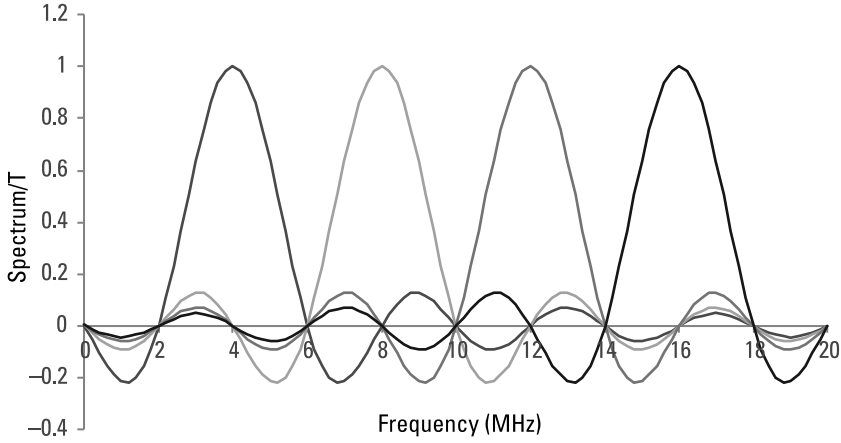


Figure 3.1 OFDFM spectrum for $N = 5$ and $f_s = 20$ MHz.

Equation (3.14) shows that the OFDM spectrum is a summation of sinc function from different subcarriers.

Example 3.1

Assuming that $f_s = 20$ MHz, then $T = 0.05\ \mu$s. Assume that also $N = 5$ and k ranges from 1 to 4. The subcarrier at $k = 0$ is assumed to be unused. The spectrum is plotted in Figure 3.1. The spectrum is maximum at frequency 4 MHz, 8 MHz, 12 MHz, and 16 MHz corresponding to $k = 1$, 2, 3, and 4. It can be seen also that the maximum at a given subcarrier is the zero-crossing for all the rest of subcarriers. This also means there is no intercarrier interference at the peak frequencies.

3.3.3 PAPR

The PAPR is defined to be the ratio of maximum power to the average power in an OFDM symbol interval:

$$\text{PAPR} = \frac{P_{\max}}{P_{\text{avg}}} \tag{3.15}$$

$$P_{\max} = \max\left(\sum_{n=0}^{N-1} x_n x_n^*\right) \tag{3.16}$$

$$P_{\text{avg}} = \text{E}\left(\sum_{n=0}^{N-1} x_n x_n^* \right) \tag{3.17}$$

where $P_{\max}$ represents the maximum power, P_{avg} represents the average power, and E is the expected value. As was mentioned in the introduction, PAPR can be large to cause detrimental effects to the OFDM signal.

To get some idea about the magnitude of PAPR, both $P_{\max}$ and P_{avg} can be evaluated by substituting (3.1) into (3.16) and (3.17). To simplify the analysis, each subcarrier is assumed to be unmodulated and $X_k = 1$. Under this assumption, (3.16) becomes

$$\begin{aligned} P_{\max} &= \frac{1}{N^2} \sum_{n=0}^{N-1} \left(\sum_{k=0}^{N-1} \max\left(e^{j2\pi nk/N} \right) \right) \left(\sum_{k=0}^{N-1} \max\left(e^{-j2\pi nk/N} \right) \right) \\ &= \frac{1}{N^2} \sum_{n=0}^{N-1} \left(N^2 \right) = N \end{aligned} \tag{3.18}$$

Similarly, (3.17) becomes

$$P_{\text{avg}} = \frac{1}{N^2} \sum_{k=0}^{N-1} \sum_{j=0}^{N-1} \left(\sum_{n=0}^{N-1} e^{j2\pi n(k-j)/N} \right) \tag{3.19}$$

Using the orthogonality relationship given in (1.21), the integral inside the parentheses is 0 unless $k = j$; (3.19) can then be further simplified to give

$$P_{\text{avg}} = \frac{N^2}{N^2} = 1 \tag{3.20}$$

Substituting (3.18) and (3.20) into (3.15), we then have the following:

$$\text{PAPR} = N \tag{3.21}$$

In (3.21), N is defined to be the number of modulated subcarriers and can be quite large. This proves PAPR in OFDM can be quite significant.

To minimize the impact of PAPR, many approaches are available. They can be classified either as signal scrambling techniques or as signal distortion techniques [1]. On signal distortion techniques, one is to perform clipping, filtering, and peak windowing [2]. On signal scrambling techniques, one is to perform block coding [3]. There are many others [1].

Example 3.2

For IEEE 802.11a, there are 52 modulated subcarriers as will be given later, and then $N = 52$. The PAPR in decibels is then $10\log 52 = 10 * 1.71 = 17.1$ dB. This means that the power amplifier must have an output back-off of 17 dB due to the power surge.

3.3.4 Cyclic Prefix

When an OFDM signal travels through the communication media, the received signal can be degraded due to the multipath interference. For example, the receiver may receive multiple delayed copies of the same transmitted waveform. If the channel impulse response has a length of N_c samples, then the received signal can also be prolonged by $N_c$samples.

An OFDM symbol is defined to have N samples. If two adjacent OFDM symbols are transmitted without any gap in between, then the tail end of one symbol may interfere with the leading edge of the subsequent symbol. To remove this type of ISI, the delay spread of the channel must first be determined. A gap of $N_g$samples must be inserted between two adjacent OFDM symbols such that $N_g \geq N_c$. This time gap is also called the guard interval.

Figure 3.2 is a plot of the OFDM signal without any guard interval between three adjacent symbols $j - 1$, j, and $j + 1$. For simplicity of notation, each symbol here refers to one FFT interval without including the guard interval. Each OFDM symbol has N samples. Figure 3.3 is a plot of the OFDM signal with a guard interval of N_g samples inserted between any two adjacent symbols.

The guard interval has a size of N_g samples. Note that an insertion of guard interval is a waste of transmission resources. However, the price paid is a reward of reduced ISI.

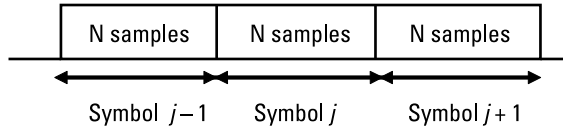


Figure 3.2 OFDM signal without time gap between adjacent symbols.

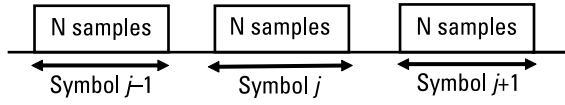


Figure 3.3 OFDM with time gap between adjacent symbols.

Transmissions don't have to happen inside the guard interval. If there is no transmission, the advantage is to save transmission power. However, with channel delay spread, different delayed subcarriers are added together and the orthogonality is lost. To solve this problem, a cyclic prefix is transmitted during the guard interval. From (3.1), it is seen that each OFDM symbol transmits N samples and N is the period. To make sure OFDM signal is periodic to maintain the orthogonality, the last N_g samples from the current symbol are inserted into the gap preceding the beginning of the current symbol.

Figure 3.4 is a plot of how the cyclic prefix is generated. In the graph, there are two adjacent symbols $j-1$ and j with a guard interval of N_g samples in between. Assume that the symbol j has samples from $n=0$ to $n=N-1$. The dark region in Figure 3.4 has the last N_g samples from $n=N-N_g$ to $n=N-1$ of symbol j. These N_g samples are copied to the dark region from $n=-N_g$ to $n=-1$ between symbols $j-1$ and j. With this insertion of cyclic prefix, there is no ISI if the channel delay spread is shorter than N_g samples.

3.3.5 FFT

So far, the DFT has been used to describe OFDM characteristics. However, it is time-consuming in using DFT for direct computation. Equation (1.18) defined the DFT and is rewritten here for easy reference:

$$X_k = \sum_{n=0}^{N-1} x_n e^{-j2\pi nk/N} \qquad k = 0,\ldots,N-1 \tag{3.22}$$

From (3.22), there are N complex multiplications and additions for each point of X_k. Since there are N points, the total number of complex multiplications and additions becomes N^2.

To reduce the number of computations, the FFT is designed. The principle is easily described when N is even using the decimation-in-time technique.

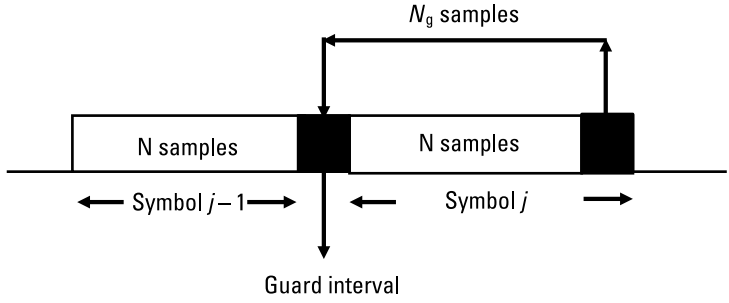


Figure 3.4 OFDM transmission of cyclic prefix.

Initially, the N points are split into two groups. One group has $N/2$ even points and the other group has $N/2$ odd points. In other words, a N-point DFT is split into two $N/2$-point DFTs. These $N/2$ even points and odd points can be further split into $N/4$ even points and odd points. This process continues until a total of $\log_2 N$ stages are reached. For each stage, the number of complex multiplications and additions is N [4]. Therefore, the total number of complex multiplications and additions becomes $N\log_2 N$. A detailed quantitative analysis can be found in [4].

The above qualitative description shows there are significant computation savings. The computation reduction ratio from the DFT to the FFT becomes $N/\log_2 N$. This ratio is 4 for $N = 16$ and 32 for $N = 256$. As the size of FFT increases, the computation time savings are even more obvious.

3.4 Time and Frequency Parameters

There are some important time and frequency parameters defined in OFDM. They are T, T_{fft}, T_{gi}, Δ_f, and T_{sym} specifying the sampling interval, FFT interval, guard interval, subcarrier spacing, and OFDM symbol interval, respectively. Equivalently, T, T_{fft}, and Δ_f correspond to d_t, L_t, and d_f defined in (1.41), (1.40), and (1.43).

Table 3.1 lists these parameter values for channel spacing of 20 MHz, 10 MHz, and 5 MHz. The parameter N is the number of samples for performing FFT. The sampling interval is just the inverse of channel spacing. The Fourier transform interval is then NT. The subcarrier spacing Δ_f is channel spacing divided by N. The OFDM symbol interval T_{sym} is the sum of the FFT interval and the guard interval or $T_{sym} = T_{fft} + T_{gi}$. For IEEE 802.11a, the guard interval is specified as 1/4 of the FFT interval. It can be seen from

Table 3.1
OFDM Time and Frequency Parameters from IEEE 802.11a

Parameter	20-MHz Channel Spacing	10-MHz Channel Spacing	5-MHz Channel Spacing
N	64	64	64
T	0.05 μs	0.1 μs	0.2 μs
T_{fft}	3.2 μs	6.4 μs	12.8 μs
Δ_f	0.3125 MHz	0.15625 MHz	0.078125 MHz
T_{gi}	0.8 μs	1.6 μs	3.2 μs
T_{sym}	4 μs	8 μs	16 μs

this table the symbol interval increases and the subcarrier spacing decreases as the channel spacing decreases.

Unless specified, a symbol interval refers to the data portion only without including the guard interval in the future chapters.

3.5 Window Function

The OFDM symbol is normally multiplied by a window function in the time domain. The purpose of the window is to smooth the transition at the symbol boundary so as not to create higher spectral side lobes of the transmitted waveform. Another is to meet the spectral mask requirements. For operations using a 20-MHz bandwidth, the IEEE 802.11a specifies a spectrum mask to be 28 dB down at a 20-MHz frequency offset from the carrier.

The IEEE 802.11a specifies a rectangular window. However, it has sharp transitions at the symbol boundary. To smooth the transition, the following modified rectangular window is defined [5]:

$$h(t) = \begin{cases} \sin^2\left(\dfrac{\pi}{4} + \dfrac{t\pi}{2\beta}\right) & -\dfrac{\beta}{2} < t < \dfrac{\beta}{2} \\ 1 & \dfrac{\beta}{2} \le t < P - \dfrac{\beta}{2} \\ \sin^2\left(\dfrac{\pi}{4} - \dfrac{\pi(t-p)}{2\beta}\right) & P - \dfrac{\beta}{2} \le t < P + \dfrac{\beta}{2} \end{cases} \tag{3.23}$$

where β is defined to extend the original rectangular window of time duration P. The sine function is defined to smooth the transition during the transition period of length β on both edges of the rectangular window. Note that as β becomes 0, the window function $h(t)$ converges to the rectangular window of length P given here:

$$h(t) = 1 \qquad 0 \le t < P \tag{3.24}$$

The spectrum can be found by taking the Fourier transform. For the rectangular window given in (3.24), the spectrum is given here:

$$\begin{aligned} H(f) &= \int_0^P e^{-j\Omega t}\,\mathrm{d}t \\ &= P\left(e^{-j\pi fP}\right)\frac{\sin(\pi fP)}{\pi fP} \end{aligned} \tag{3.25}$$

where $\Omega = 2\pi f$ and f is the frequency. For the modified window given from (3.23), the spectrum is derived here:

$$H(f) = \int_{-\beta/2}^{\beta/2} \sin^2\left(\frac{\pi}{4} + \frac{t\pi}{2\beta}\right) e^{-j\Omega t}\, dt \int_{\beta/2}^{P-\beta/2} e^{-j\Omega t}\, dt + \int_{P-\beta/2}^{P+\beta/2} \sin^2\left(\frac{\pi}{4} - \frac{\pi(t-P)}{2\beta}\right) e^{-j\Omega t}\, dt \tag{3.26}$$

After some lengthy algebra, (3.26) becomes

$$H(f) = e^{-j\pi fP}\left[\frac{\sin\left(\pi f(P-\beta)\right)}{\pi f} + \frac{\sin(\pi f\beta)\cos(\pi fP)}{\pi f} - \frac{4\pi f\cos(\pi f\beta)\sin(\pi fP)}{\left(4\pi^2 f^2 - \frac{\pi^2}{\beta^2}\right)}\right] \tag{3.27}$$

If $\beta = 0$, (3.27) converges to (3.25). However, if $f = 0$, both spectra have a magnitude of P.

The original rectangular window is from 0 to P. Figure 3.5 is a plot of the modified window function defined in (3.23). This window overlaps with the adjacent window by β. The effective window length is shortened due to the β transition period on both ends. From (3.27), the normalized power spectrum is computed as $10\log (H(f)/P)(H(f)/P)^*$. Figure 3.6 is a plot of this power spectrum against frequency in megahertz for $\beta = 0$, 0.6 μs, and 0.9 μs. The channel spacing is 20 MHz and the extended symbol width P is 4 μs. As the figure shows, the side lobe decreases as β increases, even though the main lobe does not change much.

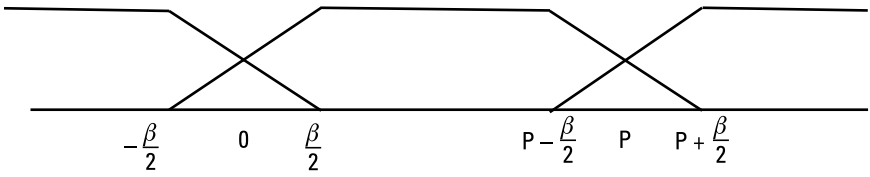


Figure 3.5 Modified rectangular window function.

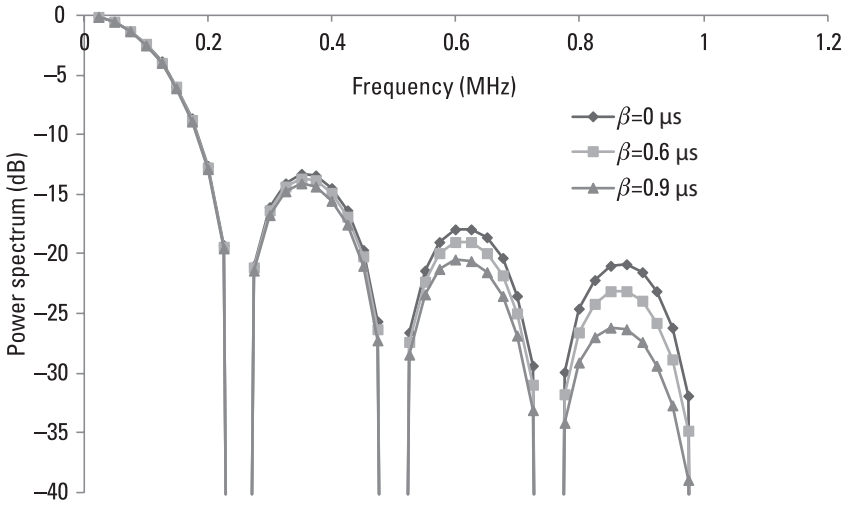


Figure 3.6 Power spectrum of the modified rectangular window at a 20-MHz channel spacing.

3.6 Digital Modulation

The complex amplitude of the kth carrier given in (3.4) is rewritten here:

$$\begin{aligned} X_k &= A_k e^{j\Psi_k} \\ &= A_k \cos\left(\Psi_k\right) + jA_k \sin\left(\Psi_k\right) \\ &= I_k + jQ_k \end{aligned} \tag{3.28}$$

where A_k is the spectral amplitude, Ψ_k is the spectral phase, I_k is the in-phase spectral component, and Q_k is the quadrature-phase spectral component. Depending on how A_k and Ψ_k are utilized, different modulated waveforms are generated.

The IEEE 802.11a utilizes three different modulation techniques: PAM, PSK, and QAM. For PAM, only the amplitude A_k is changed. For PSK, only the phase Ψ_k is changed. The two most commonly used are BPSK and QPSK. For BPSK, there are only two-phase levels and, for QPSK, there are four phase levels. For QAM, both amplitude $A_k$and phase Ψ_k are changed. For 16-QAM, there are 16 different X_k levels. As Table 2.3 and Figure 2.5 show, there are four different I_k levels and four different Q_k levels. Therefore, the total number of X_k levels is 4*4 = 16. For 64-QAM, there are 64 different X_k levels. As

Table 2.4 and Figure 2.6 show, there are 8 different I_k levels and 8 different Q_k levels. Therefore, the total number of X_k levels is 8*8 = 64.

Not all of the N subcarriers in an OFDM symbol are used for modulation. The number of subcarriers actually used is M and $M < N$. The subcarrier at DC is normally set null to minimize the impact of a DC component generated from the receiver front end. Other subcarriers not used are normally around the band edges. This can reduce the interference from side lobes of the OFDM transmission. For IEEE 802.11a, $N = 64$ and $M = 52$, there are 12 unused subcarriers including the DC. Figure 3.7 shows the region of unused subcarriers in a dark color. The subcarriers numbered from −26 to −1 and from 1 to 26 are used.

To modulate the subcarriers, the input bit stream is first grouped into bauds. Each baud has k bits. The total number of levels is $M = 2^k$ or $k = \log_2 M$. Based upon the baud, each available subcarrier is modulated using one of the M levels according to the desirable modulation technique.

Example 3.3

Assume that the input bit stream is 001011011000 using QPSK modulation. For QPSK, $k = 2$ and $M = 4$. Grouping every two bits together, the baud stream becomes 023120. Using Table 2.2 of the QPSK encoding table, the X_k sequence becomes $1\sqrt{2}\,(-1 - j)$, $1\sqrt{2}\,(1 - j)$, $1\sqrt{2}\,(1 + j)$, $1\sqrt{2}\,(-1 + j)$, $1\sqrt{2}\,(1 - j)$, and $1\sqrt{2}\,(-1 - j)$.

3.7 OFDM Waveform Properties

All the important operations in OFDM such as linearity, time shift, frequency shift, and convolution are the same as described in Section 1.8 and are not repeated here.

The sequence $X_n$is normally complex. As given in (1.32) and (1.33), X_n can be real if its spectrum has some symmetry properties. In other words,

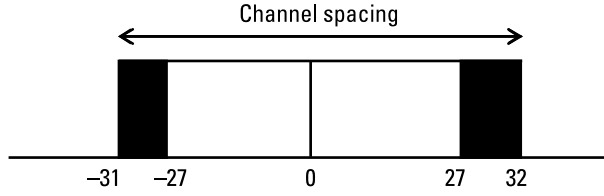


Figure 3.7 Unused subcarriers indices in IEEE 802.11a.

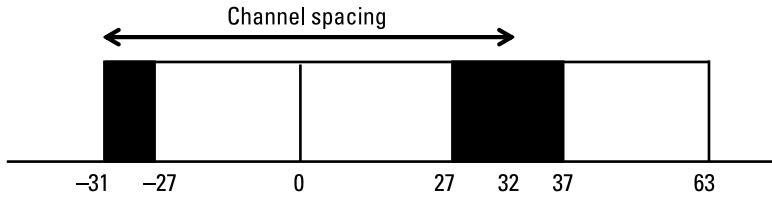


Figure 3.8 Unused subcarriers indices in IEEE 802.11a.

the real part of the spectrum has to be even and the imaginary part of the spectrum has to be odd.

The transmission of the OFDM signal through the media normally suffers from multipath delay and the Doppler shift. The multipath delay is the operation of a time shift. The impact is the multiplication by a phase factor in the frequency domain as given in (1.47). The Doppler shift is equivalent to the multiplication by a phase factor in the time domain. The impact is the frequency shift in the frequency domain as given in (1.49).

Example 3.4

The null subcarrier region can be plotted in the positive frequency domain using the periodic property $X_{-k} = X_{N-k}$. Since $N = 64$, null subcarriers from a frequency index −31 to −27 are the same as those from a frequency index 33 to 37. Figure 3.7 is plotted again in Figure 3.8.

3.8 Summary

OFDM utilizes the multicarrier modulation technique through an inverse DFT, which has a summation of N terms. Each term represents one subcarrier and there are a total of N subcarriers. Several important characteristics were discussed: orthogonality, OFDM spectrum, PAPR, cyclic prefix, and FFT.

The orthogonality characteristics showed that any two subcarriers are orthogonal in an OFDM symbol duration. The OFDM spectrum is a summation of sinc function from each subcarrier. At the peak for any subcarrier, the rest of subcarriers exhibit zero crossing. One disadvantage of OFDM is PAPR because the peak power in a single OFDM symbol can be N times larger than the average power. This effect can saturate the power amplifier and cause signal distortion. To reduce the number of computations, the DFT is performed using the FFT. The number of complex multiplications and additions for DFT against FFT is $N/\log_2 N$.

A cyclic prefix was inserted between two adjacent symbols to remove the ISI. Assume that this guard interval has a duration of N_g samples and the channel has a multipath spread of N_c samples. To remove this ISI, a necessary requirement is for $N_g > N_c$.

Each OFDM symbol was multiplied by a window function in the time domain for the purpose of reducing the higher spectral side lobes of the transmitted waveform. This window function used in IEEE 802.11a is a modified rectangular function. Its frequency spectrum was derived. A parameter was defined in the window function to control the drop rate of these side lobes.

On digital modulation, the input bit stream was applied to change the amplitude, phase, or frequency of the complex spectral amplitude. Following IEEE 802.11a, PSK, QPSK, 16-QAM, and 64-QAM were described. The signal constellation is very similar to that of the single-carrier modulation. If the DFT has a dimension N, then there is a maximum of N subcarriers. In reality, the number of subcarriers used is normally smaller than N to remove interference of side lobes from the band edges.

In the next chapter, the transmission waveform specified by the IEEE 802.11a is presented.

References

[1] Grangwar, A., and M. Bhardwaj, "An Overview: Peak to Average Power Ratio in OFDM System & Its Effect," *International Journal of Communication and Computer Technologies*, Vol. 1, No. 2, 2012, pp. 22–25.

[2] O'Neill, R., and L. B. Lopes, "Envelope Variations and Spectral Splatter in Clipped Multicarrier Signals," *Proceedings of the IEEE International Symposium on Personal, Indoor, and Mobile Radio Communications*, Vol. 1, 2005, pp. 71–75.

[3] Jones, A., and T. A. Wilkinson, "Block Coding Scheme for Reduction of Peak to Mean Envelope Power Ratio on Multicarrier Transmission Schemes," *Electronic Letters*, Vol. 30, No. 25, 1994, pp. 2098–2099.

[4] Oppenheim, A. V., and R. W. Schafer, *Digital Signal Processing*, Upper Saddle River, NJ: Prentice Hall, 1975.

[5] Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications, IEEE Standard 802.11a, 2007.

4

OFDM Transmission

4.1 Introduction

Before data is transmitted through OFDM, the preamble and header fields are normally sent first. A preamble is usually in a periodic sequence with a known data structure. Its purpose is to help the receiver in signal detection, channel estimation, timing synchronization, and frequency offset estimation. A header field normally contains a rate field to inform the receiver of the transmission data rate.

To help understand the preamble, header, and data structure, the frame format specified in IEEE 802.11a is used for illustration. However, any other useful format can be used to meet the specific requirements.

Before getting into the signal transmission format, a general transmitter block diagram is presented first to have an idea of the overall transmitter architecture. Subsequently, the frame format for preamble, header, and data is discussed in detail for IEEE 802.11a. Lastly, a typical receiver architecture is also given in preparation for future discussion of all the major components.

4.2 OFDM Transmitter Architecture

Figure 4.1 is the block diagram of a typical OFDM transmitter. The input bit sequence can be scrambled first for transmission security. The military

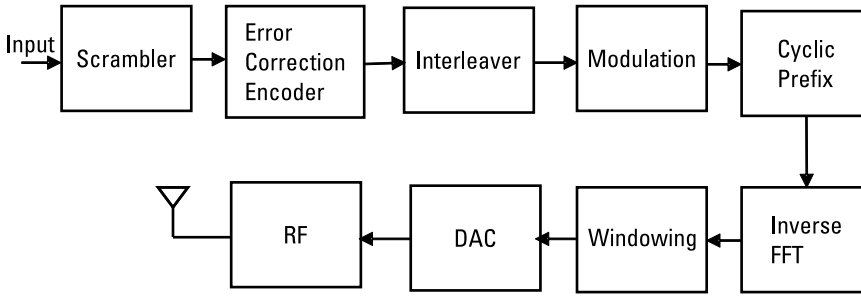


Figure 4.1 Typical OFDM transmitter block diagram.

communication normally has a complicated encryption algorithm in place to scramble the input data. The propagated OFDM signal through the media can suffer channel degradation such as multipath and fading that cause bit errors after reception. The scrambled bit sequence is therefore encoded with error correcting codes in order for the receiver to correct the bit errors. However, the bit errors can appear in bursts and it is difficult for the receiver to correct. The encoded bits then go through an interleaving procedure to spread the bits so as to make the error correction much easier.

The next four blocks, modulation, inverse FFT, cyclic prefix, and windowing, were discussed in Chapter 3. These are key blocks to generate OFDM discrete samples. These discrete samples are then converted to an analog signal by passing through a digital-to-analog converter (DAC). At last, this baseband signal modulates an RF carrier to be transmitted.

Before the data transmission, a preamble signal is normally transmitted first to help the receiver for signal detection, channel estimation, and timing synchronization.

4.3 Signal Transmission Format

OFDM signal transmission consists of three parts: preamble, signal header, and data. They are shown in Figure 4.2. The preamble is a periodic signal. For IEEE 802.11a, it consists of a short sequence followed by a long sequence. Its purpose is to help the receiver to perform signal detection, frequency offset estimation, time synchronization, and channel estimation.

The header field usually takes one OFDM symbol length. The processes of error correction encoding, interleaving, modulation, inverse FFT, and cyclic prefix insertion as shown in Figure 4.1 are applied. Their purpose is to send some key transmission parameters to the receiver. The data modulation format

Figure 4.2 OFDM signal transmission format.

and the total length are required for the receiver to demodulate. The tail bits can be inserted to make a full OFDM symbol. The header field is normally modulated with BPSK. However, the header bits are not necessarily scrambled.

After both the preamble and header transmission, the user data starts. The detailed frame format of preamble, header, and data is discussed in the following sections.

4.4 Preamble Signal for IEEE 802.11a

The preamble is a transmission of fixed signal patterns and is also known to the receiver. For OFDM, it is normally a periodic signal with a fixed signal pattern. For each subcarrier k, the complex amplitude $X_k = A_k e^{j\Psi_k}$ given in (3.4) is completely specified. The periodic signal plus the unique OFDM characteristics make the just mentioned receiver tasks easier to perform.

For IEEE 802.11a, the preamble consists of a short sequence and a long sequence [1]. The short sequence is designed for coarse timing estimation while the long sequence is used to make further refinements. These two-step processes can extract the necessary OFDM parameters for the demodulation of future data signals. Figure 4.3 is the preamble design of IEEE 802.11a. The short sequence consists of 10 periodic segments and each segment has the same number of 16 samples [1]. The long sequence consists of two OFDM symbols. Two guard intervals are followed by two consecutive FFT intervals.

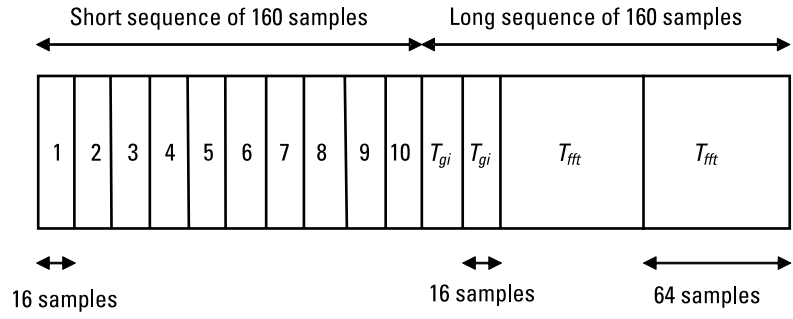


Figure 4.3 Preamble signal format of IEEE 802.11a.

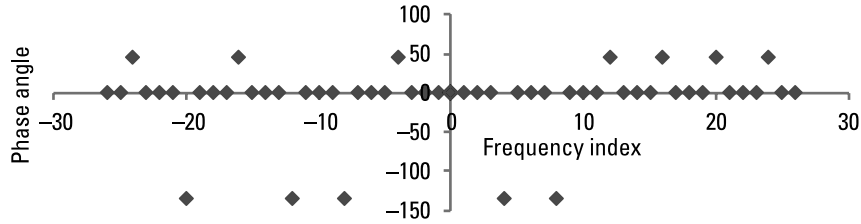


Figure 4.4 The phase of one short sequence segment in the frequency domain.

4.4.1 Short Sequence

Figure 4.4 is a plot of the phase of one fundamental short sequence segment from frequency index equal to −26 to 26. There are only 12 nonzero subcarriers. The complex amplitude X_k is either $A(1 + j)$ or $A(-1 - j)$. The constant A is selected so that the average power per subcarrier is 1. The total power of the 12 carriers is $24A^2$. The average power per subcarrier is $24A^2/52$ since the total number of subcarriers from frequency index −26 to 26 is 52. To normalize this average power, A is solved to be $\sqrt{13/6}$. The phase angle Ψ_k is either $\pi/4$ or $-3\pi/4$ and the amplitude A_k is a constant equal to $\sqrt{13/6}$. Note the phase is nonzero for every fourth frequency index. Therefore, the samples in the frequency domain can be downsampled by a factor of 4.

The short sequence specifies 52 subcarriers with the index ranging from −26 to −1 and from 1 to 26. The data from −26 to −1 can be copied to an index from 38 to 63. The data from an index from 27 to 37 including index 0 has zero value. After the downsampling by a factor of 4 starting from index 0, only 12 samples are left. Figure 4.5 is a plot of 16 frequency samples against the frequency index. From this figure, it can be seen that there are only 12 nonzero samples.

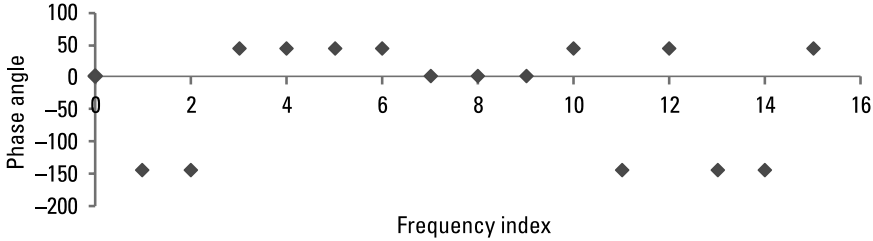


Figure 4.5 The phase of downsampled short sequence segment in the frequency domain.

Taking the inverse FFT of the 16 frequency-domain samples, the time-domain samples for each fundamental segment are obtained. The short sequence is a repetition of 10 such fundamental segments. The time duration t_{short} is given here:

$$t_{\text{short}} = 10\left(\frac{T_{fft}}{4}\right) = \frac{10NT}{4} \tag{4.1}$$

Equation (4.1) shows that the duration increases with increasing sampling period or decreasing channel spacing. The number of samples, N_{short}, is then given here:

$$N_{\text{short}} = \frac{10N}{4} \tag{4.2}$$

Therefore, the number of samples is independent of the channel spacing. Assuming a 20-MHz channel spacing, we have $T = 0.05$ μs and $N = 64$. Equation (4.1) gives $t_{\text{short}} = 8$ μs and $N_{\text{short}} = 160$. Each fundamental segment has 16 samples of duration 0.8 μs.

Before transmission, the 160 time samples must be multiplied by a time window shown in Figure 3.5. Repeating each fundamental time segment 10 times and multiplying by such a time window, the short sequence is obtained. Figure 4.6 is an amplitude plot, and Figure 4.7 is a phase angle plot of four

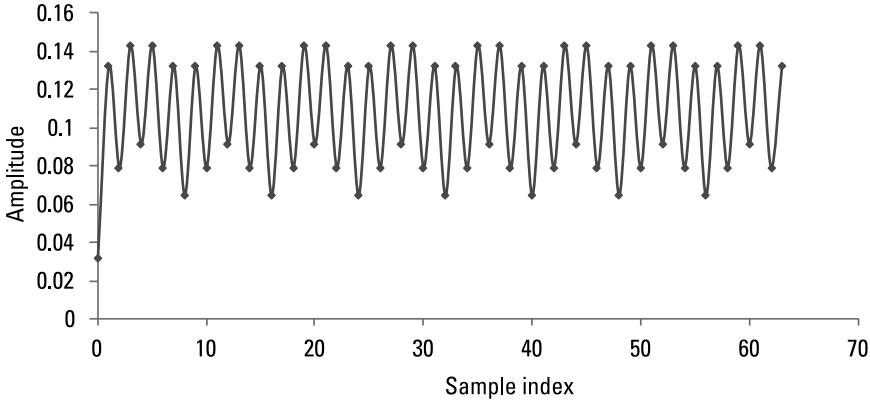


Figure 4.6 The amplitude of the first four segments of a short sequence in the time domain.

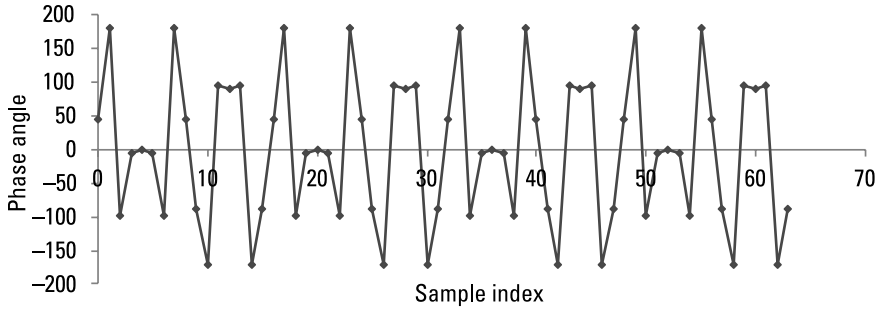


Figure 4.7 Phase angle of the first four segments of a short sequence in the time domain.

such fundamental segments in a short sequence. Note the amplitude of the first sample of the first segment is down by a half due to the multiplication by the window function. Both figures clearly show the pattern repeats every 16 samples.

4.4.2 Long Sequence

The long sequence has two OFDM symbols [1]. Each OFDM symbol consists of one inverse FFT interval and one guard interval. The duration of the long sequence is then the following:

$$t_{\text{long}} = 2\left(T_{fft} + T_{gi}\right) = 2\left(NT + \frac{NT}{4}\right) = (2.5)NT \tag{4.3}$$

The number of samples N_{long} is again independent of the bandwidth and is given here:

$$N_{\text{long}} = \frac{t_{\text{long}}}{T} = (2.5)N \tag{4.4}$$

Assume again 20-MHz channel spacing, Equation (4.3) gives $t_{\text{long}} = 8\ \mu\text{s}$ and $N_{\text{long}} = 160$. Therefore, the short sequence and the long sequence have the same number of samples. Also, for a long sequence, the guard interval has a total of $2 * N/4 = 32$ samples while the two FFT intervals have a total of $2 * N = 128$ samples.

The complex amplitude X_k is either 1 or −1. Therefore, the amplitude A_k is 1 and the phase angle Ψ_k is 0° or 180°. Figure 4.8 is a plot of the long

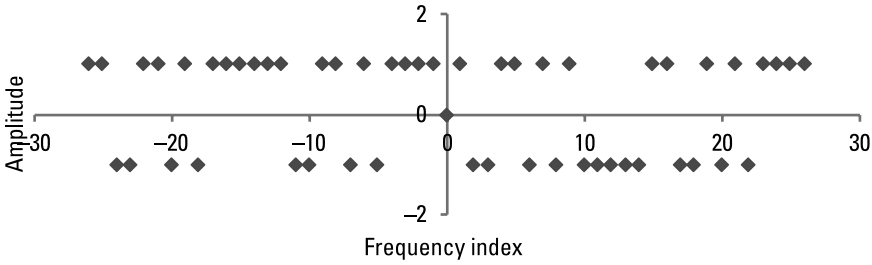


Figure 4.8 The complex amplitude of the long sequence in the frequency domain.

sequence amplitude against a frequency index from –26 to 26. Except at DC or frequency index zero, the complex amplitude is nonzero at all other frequency indexes.

Taking an inverse FFT, 64 time-domain samples can be obtained. Following the design of Figure 4.3, the long sequence of 160 samples is obtained. After multiplying by a time window, Figure 4.9 is a plot of the long sequence amplitude and Figure 4.10 is a plot of the long sequence phase in the time domain. Note that the amplitude and phase are the same between frequency index from 32 to 95 and a frequency index from 96 to 159. Note also that the amplitude and phase of the guard interval from a frequency index 0 to 31 are copied from those of a frequency index from 128 to 159. The amplitude at the time index zero is multiplied by a factor of 0.5 because of the window function.

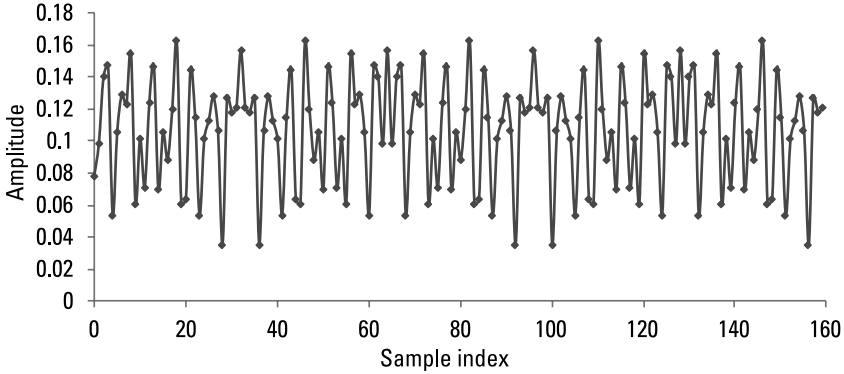


Figure 4.9 The amplitude of the long sequence in the time domain.

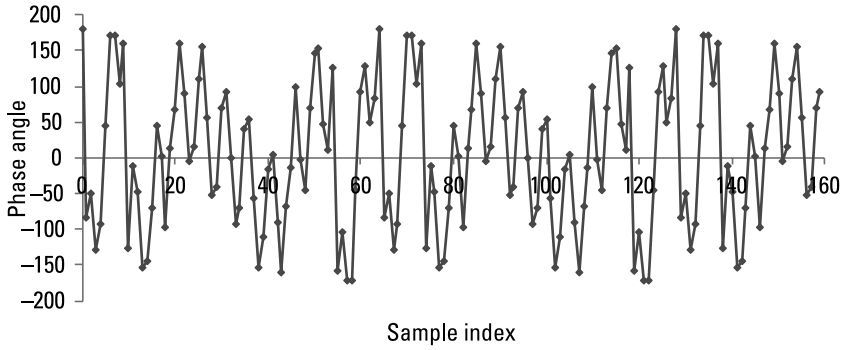


Figure 4.10 The phase angle of the long sequence in the time domain.

4.5 IEEE 802.11a Header Format

For IEEE 802.11a, the header format is shown in Figure 4.11 [1]. There are a total of 5 fields. The RATE field has 4 bits and can specify a maximum of 16 different combinations of modulation and coding rate. The LENGTH field has 12 bits and specifies the total data length in bytes. There are six TAIL bits attached at the end to make for a total of 24 bits. The modulation is BPSK using convolutional code at a rate of 1/2. Therefore, the total coded bits per symbol are 48.

Table 4.1 lists eight different data rates specified by IEEE 802.11a. The convolutional code is selected for the error correction. Based upon a convolutional coding rate R, the other parameters can be either specified or derived. There are only four different modulation techniques and they are BPSK, QPSK, 16-QAM, and 64-QAM. The coded bits per subcarrier N_{bpsc} are 1, 2, 4, and 6, respectively. The coded bits per OFDM symbol N_{cbps} and data bits per OFDM symbol N_{dbps} are then computed as here:

$$N_{\text{cbps}} = N_{\text{sd}} N_{\text{bpsc}} \tag{4.5}$$

$$N_{\text{dbps}} = R_{\text{coding}} \left(N_{\text{cbps}} \right) \tag{4.6}$$

RATE (4 bits)	RESERVED (1 bit)	LENGTH (12 bits)	PARITY (1 bit)	TAIL (6 bits)

Figure 4.11 Header format of IEEE 802.11a.

where N_{sd} is the number of data subcarriers and R_{coding} is the coding rate. For a convolutional encoder, the coding rate R_{coding} specifies the ratio of data bits against the coded bits. For IEEE 802.11a. there are 52 subcarriers with a frequency index ranging from −26 to −1 and from 1 to 26. However, subcarriers −21, −7, 7, and 21 are reserved for pilots. That means the total number of data subcarriers is $N_{sd} = 48$. The data rate, R_{data} is then computed as follows:

$$R_{data} = \frac{N_{dbps}}{T_{fft} + T_{gi}} \tag{4.7}$$

The parameters T_{fft} and T_{gi} are given in Table 3.1.

Example 4.1

Assuming that QPSK and coding rate $R = 3/4$, then $N_{bpsc} = 2$ from Table 4.1 and there are $N_{sd} = 48$ data subcarriers using IEEE 802.11a. From (4.5), $N_{cbps} = 48 * 2 = 96$. From (4.6), $N_{dbps} = 96 * 3/4 = 72$. At 20 MHz, the symbol duration is $T_{fft} + T_{gi} = 4$ μs from Table 3.1. From (4.7), the data rate is then $R_{data} = 72/4 = 18$ Mbps.

Example 4.2

For the header field, the modulation is BPSK and the coding rate is 1/2. From Table 4.1, we have $N_{cbps} = 48$ and $N_{dbps} = 48 * 1/2 = 24$. The data rate at 20 MHz is then 24/4 = 6 Mbps.

Table 4.1
IEEE 802.11a Modulation Parameters at 20-MHz Channel Spacing

Modulation	R	N_{bpsc}	N_{cbps}	N_{dbps}	Data Rate (Mbps)
BPSK	1/2	1	48	24	6
BPSK	3/4	1	48	36	9
QPSK	1/2	2	96	48	12
QPSK	3/4	2	96	72	18
16-QAM	1/2	4	192	96	24
16-QAM	3/4	4	192	144	36
64-QAM	2/3	6	288	192	48
64-QAM	3/4	6	288	216	54

SERVICE (16 bits)	MESSAGE (Variable)	TAIL (6 bits)	PAD (Variable)

Figure 4.12 Data field of IEEE 802.11a.

4.6 IEEE 802.11a Data Format

Figure 4.12 shows the data format of IEEE 802.11a [1]. There are four fields, SERVICE, MESSAGE, TAIL, and PAD. The PAD field is used so the total data length is an integral multiple of N_{dbps}.

The total number of data bits is given here:

$$N_{\text{data}} = N_{\text{dbps}} \left(\text{Ceiling} \left(\frac{\left(16 + 8N_{\text{message}} + 6\right)}{N_{\text{dbps}}} \right) \right) \tag{4.8}$$

where N_{message} is the number of message bytes specified in the LENGTH field of the header. In (4.8), 16 is the number of SERVICE bits while 6 is the number of TAIL bits. The number of PAD bits N_{pad} is then computed here:

$$N_{\text{pad}} = N_{\text{data}} - \left(16 + 8N_{\text{message}} + 6\right) \tag{4.9}$$

The number of PAD bits is a variable and depends upon the data length in bytes specified in the header field.

Example 4.3

Assume that the incoming message has 125 bytes using the QPSK and 1/2 coding rate. The data field has 16 + 6 + 8 * 125 = 1,022 bits. From Table 4.1, $N_{\text{dbps}} = 48$. From (4.8), $N_{\text{data}} = 48$ Ceiling (1,022/48) = 48 * 22 = 1,056. From (4.9), $N_{\text{pad}} = 1{,}056 - 1{,}022 = 34$. Therefore, 34 bits must be padded.

After data demodulation, a total of N_{cbps} bits are collected for each OFDM symbol. When the processes of deinterleaver, error correction decoding and descrambling are completed, the desired number of N_{dbps} data bits are then recovered.

4.7 OFDM Receiver Architecture

Before getting into the various topics related to the receiver design, a typical receiver architecture is presented first in Figure 4.13. In this diagram, the major components inside a receiver are given.

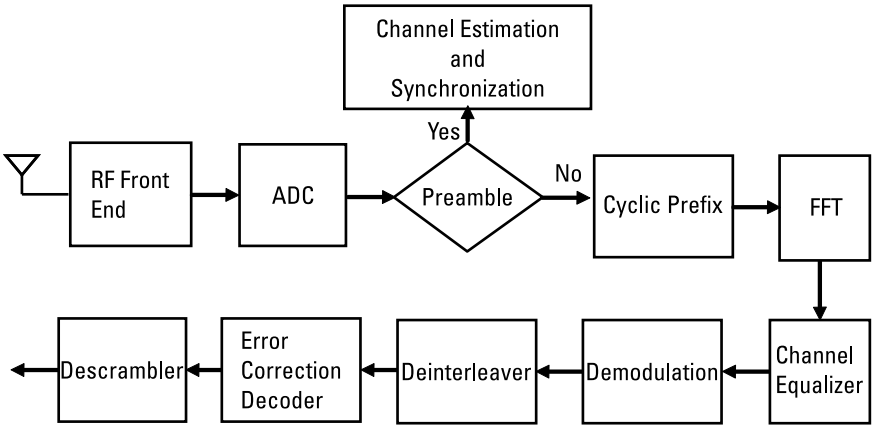


Figure 4.13 A typical OFDM receiver architecture.

After propagating through the media, the received analog signal goes through an RF front end, which performs filtering, automatic gain control (AGC), and the recovery of in-phase and quadrature-phase (IQ) signals. The IQ signal is subsequently digitized for further processing.

There are two different types of signal to be received: preamble and data. If the received signal is a preamble, the channel estimate and synchronization are performed. However, if the input is data, the cyclic prefix is first removed before performing the FFT. During the data transmission, not all the subcarriers are dedicated to data. Some are pilot subcarriers to help channel estimation and synchronization. This is true not just for 802.11a but also for 4G-LTE.

Subsequently, the channel degradation is removed based upon an earlier channel estimate. The channel estimation can also be updated concurrently using the LMS algorithm. After that, the received IQ signal is demodulated to get the bit stream. This bit stream subsequently goes through the deinterleaver, error correction decoder, and, finally, the descrambler to get the original transmitted data.

The detailed processing of every box will be discussed in subsequent chapters.

4.8 Summary

A general transmitter block diagram was first given to have a picture of the overall OFDM architecture. The transmission frame format including preamble,

header, and data was subsequently discussed. The frame format specified in IEEE 802.11a was used for illustration.

A preamble consists of a short sequence and a long sequence in IEEE 802.11a. The short sequence has 10 periodic segments and a total of 160 samples. The long sequence also has 160 samples including two guard intervals of 32 samples and two FFT intervals of 128 samples.

The header field specifies the data rate to inform the receiver for demodulation; it also has a LENGTH field specifying the total data length in bytes for transmission. The data field has tail bits to allow the convolutional encoder to return to the zero state. It also inserts some pad bits so the total duration has an integral number of data bits per OFDM symbol.

Finally, a typical receiver architecture was presented to show the major components inside a receiver. Each component in both the transmitter and receiver architecture will be discussed in subsequent chapters.

In the next chapter, the shift-register sequence, data scrambler, and descrambler are discussed.

Reference

[1] Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications, IEEE Standard 802.11a, 2007.

5

Shift Register Sequence and Data Scrambler

5.1 Introduction

In digital communication, the input data is often scrambled before the modulation and transmission. The receiver knows exactly how to generate the scrambler sequence used in the transmitter to recover the original data. For military communication, the sequence period is extremely long and is difficult for the enemy to descramble. For commercial communication, it is long enough to protect the security of the user data.

IEEE 802.11a also defines a scrambler generated from a shift register sequence. Depending upon the number of shift registers, this sequence has a unique period of maximum length. To understand the process of sequence generation from the shift registers, we start with a binary field, and a Galois field is also discussed. The details of sequence generation from the shift registers and its period determination are subsequently presented.

Finally, a data scrambler is discussed. The process of data scrambling in the transmitter and descrambling in the receiver is also discussed.

5.2 Binary Field

A Galois field [1] with p elements is normally represented as GF(p) where p is a prime. If $p = 2$, it is a binary field GF(2) having only two elements. These two elements are 0 and 1. The operation of multiplication and addition are defined in Tables 5.1 and 5.2.

The multiplication has the symbol "." while the addition has the symbol "+." Both addition and multiplication are performed under a modulo-2 operation. For example, 1 + 1 (modulo-2) = 0. The field is closed since both addition and multiplication generate an element of either 0 or 1 in the same field.

The operation of subtraction "–" and division "/" can also be defined. Assuming that a and b are two elements in a field, then $a - b = a + (-b)$. The $-b$ is defined such that $b + (-b) = 0$. The element $-b$ is also called the additive inverse element. Since the operation is modulo-2, $-b$ is then equal to $2 - b$. For $b = 1$, $-b = 1$, and for $b = 0$, $-b = 0$.

The division of a by b is defined as $a/b = a.b^{-1}$. The element b^{-1} is also called a multiplicative inverse element. It is defined such that $b.b^{-1} = 1$. For $b = 1$, $b^{-1} = 1$. The element 0 is excluded from having a multiplicative inverse.

Table 5.1 also shows that the addition is commutative and associative. Assuming that a, b, and c are three numbers from GF(2), then clearly $a + b = b + a$ and $a + (b + c) = (a + b) + c$. Tables 5.1 and 5.2 also show that the multiplication is distributive since $a(b + c) = (a.b) + (a.c)$.

Table 5.1
Binary Addition

+	0	1
0	0	1
1	1	0

Table 5.2
Binary Multiplication

.	0	1
0	0	0
1	0	1

A polynomial whose coefficients are from GF(2) can also be defined. Assuming that $f(x)$ of degree n is such a polynomial, it is given by

$$f(x) = a_0 + a_1x + a_2x^2 + a_3x^3 + \ldots + a_nx^n \tag{5.1a}$$

where $a_n = 0$ or 1. Similarly, a polynomial $g(x)$ of degree m is defined here:

$$g(x) = b_0 + b_1x + bx^2 + bx^3 + \ldots + b_mx^m \tag{5.1b}$$

where $b_m = 0$ or 1. The addition of $f(x)$ and $g(x)$ is then given here:

$$f(x) + g(x) = \left(a_0 + b_0\right) + \left(a_1 + b_1\right)x + \left(a_2 + b_2\right)x^2 + \ldots \tag{5.2}$$

The subtraction of $g(x)$ from $f(x)$ can be similarly defined:

$$\begin{aligned} f(x) - g(x) &= \left(a_0 - b_0\right) + \left(a_1 - b_1\right)x + \left(a_2 - b_2\right)x^2 + \ldots \\ &= \left(a_0 + \left(-b_0\right)\right) + \left(a_1 + \left(-b_1\right)\right)x + \left(a_2 + \left(-b_2\right)\right)x^2 + \ldots \end{aligned} \tag{5.3}$$

Example 5.1

Assuming that $f(x) = 1 + x^2 + x^3$ and $g(x) = x + x^2 + x^3 + x^4$, then

$$\begin{aligned} f(x) + g(x) &= 1 + x + (1+1)x^2 + (1+1)x^3 + x^4 = 1 + x + x^4 \\ f(x) - g(x) &= 1 - x + (1-1)x^2 + (1-1)x^3 - x^4 = 1 + x + x^4 \end{aligned}$$

The multiplication of $f(x)$ by $g(x)$ is given here:

$$\begin{aligned} f(x)g(x) &= \left(a_0 + a_1x + a_2x^2 + a_3x^3 + \ldots + a_nx^n\right)\left(b_0 + b_1x + b_2x^2 + b_3x^3 + \ldots + b_mx^m\right) \\ &= a_0b_0 + \left(a_0b_1 + a_1b_0\right)x + \left(a_0b_2 + a_1b_1 + a_2b_0\right)x^2 + \ldots a_nb_mx^{n+m} \\ &= c_0 + c_1x + c_2x^2 + c_3x^3 + \ldots + c_{n+m}x^{n+m} \end{aligned} \tag{5.4a}$$

where c_n is in general given by

$$c_n = a_0b_n + a_1b_{n-1} + a_2b_{n-2} + \ldots + a_nb_0 \tag{5.4b}$$

Example 5.2

Assume $f(x) = 1 + x + x^2$ and $g(x) = 1 + x^2$, then the product of $f(x)$ and $g(x)$ is given by

$$f(x)g(x) = 1 + x + x^2 + x^2 + x^3 + x^4 = 1 + x + (1+1)x^2 + x^3 + x^4$$
$$= 1 + x + x^3 + x^4$$

Equations (5.4a) and (5.4b) can also be used to compute the product. In this case, $a_0 = 1$, $a_1 = 1$, $a_2 = 1$, $a_3 = 0$, $a_4 = 0$ and $b_0 = 1$, $b_1 = 0$, $b_2 = 1$, $b_3 = 0$, $b_4 = 0$. From (5.4b), we have $c_0 = a_0b_0 = 1$, $c_1 = a_0b_1 + a_1b_0 = 1.0 + 1.1 = 1$, $c_2 = a_0b_2 + a_1b_1 + a_2b_0 = 1.1 + 1.0 + 1.1 = 0$, $c_3 = a_0b_3 + a_1b_2 + a_2b_1 + a_3b_0 = 1.0 + 1.1 + 1.0 + 0.1 = 1$ and $c_4 = a_2b_2 = 1$. Therefore, $f(x)g(x) = 1 + x + x^3 + x^4$ and is the same as given earlier.

The division in GF(2) can be performed through long division. If the quotient is $q(x)$ and the remainder is $r(x)$, then $f(x) = q(x)g(x) + r(x)$. The degree of $r(x)$ is less than $q(x)$. The process of long division is shown through the following example.

Example 5.3

Assuming that $f(x) = x^4$, $g(x) = 1 + x + x^2$, then $f(x)/g(x)$ is given here:

$$\begin{array}{r} x^2 + x \\ x^2 + x + 1 \overline{)\, x^4} \\ \underline{x^4 + x^3 + x^2} \\ x^3 + x^2 \\ \underline{x^3 + x^2 + x} \\ x \end{array}$$

Therefore, the quotient $q(x)$ is $x^2 + x$ and the remainder $r(x)$ is x. To prove its correctness, we have $f(x) = q(x)g(x) + r(x) = (x^2 + x)(x^2 + x + 1) + x = x^4 + x^3 + x^2 + x^3 + x^2 + x + x = x^4$.

If $r(x) = 0$, then $f(x) = q(x)g(x)$ and $f(x)$ is divisible by $g(x)$ If x is a root of $f(x)$, then $f(x) = 0$. As an example, $x = 1$ is a root of $f(x) = x^4 + x^3 + x^2 + x$ since $f(1) = 0$. Through long division, it can be shown that $f(x) = x^4 + x^3 + x^2 + x = (x + 1)(x^3 + x)$. This shows that $f(x)$may have roots not in GF(2). It is similar to the case that a real polynomial may have complex roots.

5.3 Galois Field

In general, a Galois field has more than two elements. Of particular interest is the field GF(2^m) and $m > 1$ [1]. To construct such a field, a primitive

polynomial needs to be defined. Assume that $p(x)$ is a polynomial with degree m over GF(2). It is primitive if the smallest positive n such that $p(x)$ divides $x^n + 1$ is $n = 2^m - 1$and $p(x)$ is not divisible by any polynomial of degree less than m. For example, $p(x) = 1 + x + x^3$ is primitive since $m = 3$ is the smallest degree such that $p(x)$ divides $x^7 + 1$. Through long division, it can be shown that $x^7 + 1 = (x^4 + x^2 + x + 1)(x^3 + x + 1)$. The polynomial $p(x)$ is also not divisible by any polynomial of degree less than 3. Therefore, $p(x)$ is primitive.

In a binary field of GF(2), it is a modulo-2 operation. In the extension field of GF(2^m), it is an operation by performing modulo $p(x)$ and $p(x)$ is a primitive polynomial. Equivalently, $p(x) = 0$ and is similar to the binary field by setting 2 to zero. Therefore, no polynomial in GF(2^m) can have a degree greater than m. Because $x^{2^m-1} + 1$ is divisible by $p(x)$, $x^{2^m-1} + 1 = 0$ or $x^{2^m-1} = 1$. Assume that y is an element of GF(2^m). To construct this field, we start with the number 1 and continue multiplying by y. A new element is generated with every multiplication by y. This process continues until y^{2^m-2} is reached. Including 0, a total of 2^m elements is then generated. Using the property $p(y) = 0$ and $y^{2^m-1} = 1$, each field element can be reduced to a polynomial with a degree less than m.

Example 5.4

Assume that the primitive polynomial is $p(x) = 1 + x + x^3$. Then we have $x^3 = x + 1$ and $x^7 = 1$. The 8 field elements are then generated here:

$$0$$
$$1$$
$$y$$
$$y^2$$
$$y^3 = y + 1$$
$$y^4 = y(y + 1) = y^2 + y$$
$$y^5 = y\left(y^2 + y\right) = y^3 + y^2 = y^2 + y + 1$$
$$y^6 = y\left(y^2 + y + 1\right) = y^3 + y^2 + y = y + 1 + y^2 + y = y^2 + 1$$

Since $y^7 = 1$, the process repeats and there are at most 8 elements in GF(8). To prove $y^7 = 1$, we have $y^7 = y(y^2 + 1) = y^3 + y = y + 1 + y = 1$. Also, from the listing above, every element polynomial has degree of 2 or less.

If the primitive polynomial has degree of m, then all the field element polynomials can be represented as $a_0 + a_1 y + a_2 y^2 + \cdots + a_{m-1} y^{m-1}$. The coefficient

a_n is over GF(2) and is either 0 or 1. The total number of terms is m. Therefore, the total number of field elements is 2^m.

The multiplicative inverse element and additive inverse element also exist in GF(2^m). The additive inverse element is the element itself. Therefore, the additive inverse element of y^i is y^i. The multiplicative inverse element of y^i is $y^{-i} = y^{2^m-1}y^{-i} = y^{2^m-i-1}$. From Example 5.4, the multiplicative inverse element of y^i is y^{7-i}. For example, the multiplicative inverse element of y^3 is y^4 and so forth.

The Galois field GF(2^m) is also closed under the addition and multiplication [1]. This means that the addition or multiplication of any two field elements is also an element in GF(2^m). For example, consider two field elements y and y^5. From Example 5.4, the addition of y and y^5 is $y^2 + 1 = y^6$, which is also a field element. From Example 5.4 again, the multiplication of y and y^5 is y^6, which is clearly a field element. However, this is true for any two field elements. This illustrates that the Galois field GF(2^m) is closed under the addition and multiplication.

The Galois field GF(2^m) also satisfies the commutative, associative, and distributive laws. Every polynomial in GF(2^m) has coefficients over GF(2). As has been shown before, these polynomials satisfy these laws. Therefore, these laws are also satisfied over GF(2^m).

Another interesting property of GF(2^m) is that every field element of the form y^i where $i < 2^m - 1$ is a root of $f(x) = x^{2^m-1} + 1$ [1]. This can be easily proven as follows:

$$f(x) = f\left(y^i\right) = y^{i(2^m-1)} + 1 = y^{(2^m-1)i} + 1 = \left(y^{2^m-1}\right)^i + 1 = 1^i + 1 = 1 + 1 = 0$$

Therefore, the polynomial sequence 1, y, $y^2 + \cdots + y^{2^m-2}$ are all the roots of $x^{2^m-1} + 1 = 0$.

5.4 Sequence Generator

The shift registers can be used to generate a pseudonoise (PN) sequence with the desired period. Figure 5.1 is a logical diagram to generate such a sequence through polynomial division [1, 2]. The circle with a + sign inside represents summation and the circle with z^{-1} inside means a delay by one sample. Finally, the circle with a coefficient g_n inside represents a multiplication by g_n. On the rightmost shift register, y_n is an output and input to the shift register. On the leftmost register, there is a feedback path back to the input. All the polynomial summations and multiplications are performed under modulo-2 arithmetic.

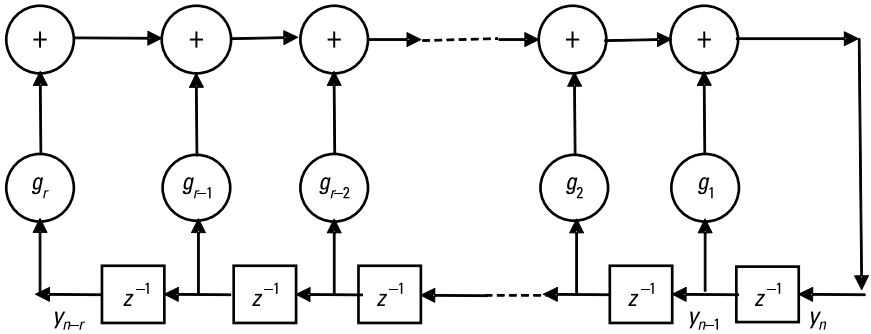


Figure 5.1 Linear feedback shift register performing polynomial division.

Based upon Figure 5.1, we have the following difference equation:

$$y_n = g_1 y_{n-1} + g_2 y_{n-2} + \ldots + g_{r-1} y_{n-r+1} + g_r y_{n-r} + A_0 \tag{5.5}$$

where A_0 is the initial condition on the shift register. Performing a z-transform using the time shift operation given by (1.57) on both sides of (5.5), we have

$$Y(z) = Y(z)\left(g_1 z^{-1} + g_2 z^{-2} + \ldots + g_{r-1} z^{-r+1} + g_r z^{-r}\right) + A(z) \tag{5.6}$$

where $A(z)$ is given by the following equation:

$$A(z) = a_0 + a_1 z^{-1} + a_2 z^{-2} + \ldots + a_{r-1} z^{-r+1} \tag{5.7}$$

The coefficients from a_0 to a_{r-1} are initial loads to the r shift registers. The coefficient a_0 is the initial load to the leftmost shift register while a_{r-1} is the initial load to the rightmost shift register. The coefficient a_n of the nth shift register is either 0 or 1. Combining terms with $Y(z)$ together, we have

$$Y(z)\left(1 - g_1 z^{-1} - g_2 z^{-2} + \ldots + g_{r-1} z^{-r+1} - g_r z^{-r}\right) = A(z) \tag{5.8}$$

Since g_n is binary, (5.8) after a modulo-2 operation becomes

$$Y(z) = \frac{A(z)}{1 + g_1 z^{-1} + g_2 z^{-2} + \ldots + g_{r-1} z^{-r+1} + g_r z^{-r}} \tag{5.9}$$

Substituting $x = z^{-1}$ into (5.9), we have

$$y(x) = \frac{a_0 + a_1 x + a_2 x^2 + \ldots + a_{r-1} x^{r-1}}{1 + g_1 x + g_2 x^2 + \ldots + g_{r-1} x^{r-1} + g_r x^r} \tag{5.10}$$

Assuming that the nominator polynomial is $A(x)$ and the denominator polynomial is $G(x)$, we have

$$\begin{aligned} A(x) &= a_0 + a_1 x + a_2 x^2 + \ldots + a_{r-1} x^{r-1} \\ G(x) &= 1 + g_1 x + g_2 x^2 + \ldots + g_{r-1} x^{r-1} + g_r x^r \\ y(x) &= \frac{A(x)}{G(x)} \end{aligned} \tag{5.11}$$

Note that a_0 is the content of the leftmost shift register while a_{r-1} is the content of the rightmost shift register [2]. Before performing the division, $A(x)$ is modified to $B(x)$ such that $B(x)$ is the polynomial of $A(x)G(x)$ after removing all terms with a degree greater than $r - 1$ [2].

Example 5.5

Assume that $G(x) = 1 + x + x^3$ and the initial load to the shift registers is 001 as illustrated in Figure 5.2. There are two methods to get the output. The first is to step through the shift register operations cycle by cycle and the second is through the polynomial divisions. Using the first method, the shift register content and output are given in Table 5.3.

During each cycle, the content of the shift register shifts one place to the left. The initial condition is that the shift register content is 001 at the first cycle. In the second cycle, the content of the third register becomes 0 while the content of the second register becomes 1. The summation result is shifted to the first register and becomes 0 + 1 = 1 and the output is 1.

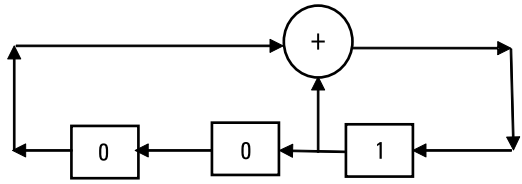


Figure 5.2 Sequence generator for Example 5.5.

Table 5.3
Register Content and Output for Example 5.5

Cycle	Register 3	Register 2	Register 1	Output
1	0	0	1	1
2	0	1	1	1
3	1	1	1	1
4	1	1	0	0
5	1	0	1	1
6	0	1	0	0
7	1	0	0	0

Therefore, the content of the shift registers becomes 011 after the first cycle. For the other cycles, the register content can be obtained in the same way. After the seventh cycle, the content of the shift register repeats so the period is 7.

Using the second method, the polynomial $B(x)$ needs to be determined first. From the initial load, $A(x) = x^2$ and $A(x)G(x) = x^2(1 + x + x^3) = x^2 + x^3 + x^5$. After removing all the terms with a degree greater than 2, we have $B(x) = x^2$. The long division is then performed here:

$$
\begin{array}{r}
x^2 + x^3 + x^4 + x^6 + x^9 \\
1 + x + x^3 \overline{)\, x^2} \\
\underline{x^2 + x^3 \qquad + x^5} \\
x^3 + \qquad + x^5 \\
\underline{x^3 + x^4 \qquad + x^6} \\
x^4 + x^5 + x^6 \\
\underline{x^4 + x^5 + \qquad x^7} \\
x^6 + x^7 \\
\underline{x^6 + x^7 + x^9} \\
x^9
\end{array}
$$

Observe that the coefficients of the quotient polynomial starting at x^9 repeats. Therefore, the long division generates a periodic output of 1110100, which is the same as that from method 1. Since there are only three registers, the maximum number of unique register contents is $2^3 - 1 = 7$. Therefore, 7 is also the maximum period.

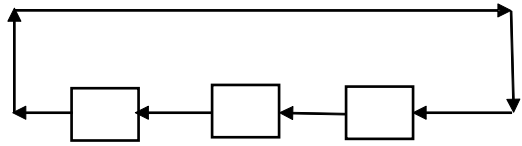


Figure 5.3 Sequence generator for Example 5.6.

Example 5.6

Consider another example with $G(x) = 1 + x^3$ illustrated in Figure 5.3. The first method is used to find the output with different initial register contents. The results are given in Table 5.4.

As can be seen from Table 5.4, different initial register content generates different output and period. For the initial register content 001, 111, 000, 110, the output sequence is 100, 111, 000, 011, respectively, and the period is either 1 or 3.

5.5 Period of Sequence Generator

Assume a sequence generator has any polynomial $G(x)$ of degree r defined by (5.11). An inverse polynomial $G^{-1}(x)$ can be defined as follows:

$$G^{-1}(x) = x^r G\left(\frac{1}{x}\right) \tag{5.12}$$

where r is the number of shift registers. It can be shown [2] the maximum period N of a shift register sequence with polynomial $G(x)$ is the smallest N such that $x^N + 1$ is divisible by $G^{-1}(x)$. This is demonstrated by the following example.

Table 5.4
Register Content for Example 5.6

Initial Register Contents/Cycles	001	111	000	110
1	010	111	000	101
2	100	111	000	011
3	001	111	000	110

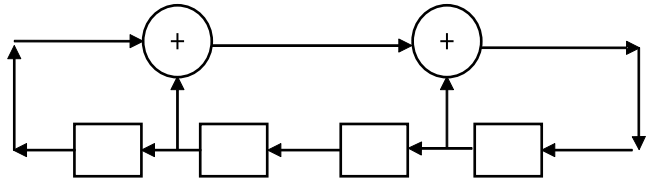


Figure 5.4 Sequence generator for Example 5.7.

Example 5.7

Let $G(x) = 1 + x + x^3 + x^4$ for the sequence generator illustrated in Figure 5.4. The inverse polynomial is then given by

$$G^{-1}(x) = x^4 G\left(\frac{1}{x}\right) = x^4\left(x^{-4} + x^{-3} + x^{-1} + 1\right) = 1 + x + x^3 + x^4$$

Through long division, it can be shown that

$$x^6 + 1 = \left(x^4 + x^3 + x + 1\right)\left(x^2 + x + 1\right)$$
$$x^{12} + 1 = \left(x^4 + x^3 + x + 1\right)\left(x^8 + x^7 + x^6 + x^2 + x + 1\right)$$

Both $x^6 + 1$and $x^{12} + 1$ are divisible by $(x^4 + x^3 + x + 1)$, but the smallest degree is 6 and the period is then 6. This can be verified by going through the state sequence as given by Table 5.5.

The maximum number of states for 4 registers is 15 and all the 15 states appear in Table 5.3 and there are no other periods. This proves that the maximum period is 6.

Table 5.5
Register Content of Example 5.7

Initial Register Contents/Cycles	**0000**	**0101**	**1111**	**1011**	**0011**	**0010**
1	0000	1010	1111	0110	0111	0100
2		0101		1101	1110	1001
3				1011	1100	0010
4					1000	
5					0001	
6					0011	

5.6 Maximum-Length Sequences

The period of a shift register sequence for any polynomial $G(x)$ has been defined in the previous section. It is the smallest N such that $x^N + 1$ is divisible by the inverse polynomial $G^{-1}(x)$. If $G(x)$ is a primitive polynomial as discussed in Section 5.3, it has been shown [2] that $G^{-1}(x)$ is also primitive. If the primitive polynomial has a degree r, the smallest degree N such that $G(x)$ divides $x^N + 1$ is $N = 2^r - 1$. Since the number of shift registers is also r, the maximum number of possible state sequences is $N = 2^r - 1$. Under this condition, the maximum period is $2^r - 1$. The shift register sequence that generates the maximum period is then called the maximum-length sequence.

Table 5.6 lists some representative primitive polynomials of degrees less than 10 [1]. A more comprehensive listing of primitive polynomials can be found in [2].

5.6.1 Properties of the Maximum-Length Sequence

In Example 5.5, $G(x) = 1 + x + x^3$ is a primitive polynomial. The output sequence is 1110100 if the initial state is 001. As indicated, the period is 7. An examination of the output sequence shows that there are 4 ones and 3 zeros. The number of ones is one more than the number of zeros. This is, in general, true for any maximum length sequence. Since the state sequence goes through all the possible states, the number of odd numbers is one more than

Table 5.6
Listing of Some Primitive Polynomials

Degree	Primitive Polynomial	Maximum Period
2	$1 + x + x^2$	3
3	$1 + x + x^3$	7
4	$1 + x + x^4$	15
5	$1 + x^2 + x^5$	31
6	$1 + x + x^6$	63
7	$1 + x^3 + x^7$	127
8	$1 + x^2 + x^3 + x^4 + x^8$	255
9	$1 + x^4 + x^9$	511
10	$1 + x^3 + x^{10}$	1023

the number of even numbers. Examining the register content of Figure 5.2, it can be seen that there are 4 odd numbers and 3 even numbers. Since the output is shifted to the rightmost register, the number of ones is one more than the number of zeros.

Including two periods, the output sequence of Example 5.5 is 11101001110100. If the initial content of the register is changed to 110, the output sequence becomes 01001110100111. Comparing the two sequences, the second sequence starts from the fourth digit of the first sequence. Therefore, the same sequence is generated except for the phase difference. That this is true is because the maximum-length sequence goes through all the possible states. Therefore, for a fixed primitive polynomial, the maximum length sequence is independent of the register content except for the phase. The period is also always the same independent of the content of the initial register state.

5.6.2 Sequence Generator from the IEEE 802.11a

In the IEEE 802.11a, a sequence generator is specified to scramble the input data. The primitive polynomial used in this sequence generator has the seventh degree. From Table 5.6, it is given here:

$$G(x) = 1 + x^3 + x^7 \tag{5.13}$$

The inverse polynomial is also primitive and is from (5.12) with $r = 7$:

$$G^{-1}(x) = x^7\left(1 + x^{-3} + x^{-7}\right) = 1 + x^4 + x^7 \tag{5.14}$$

The IEEE 802.11a uses the inverse polynomial defined in (5.14) as the generator and is shown in Figure 5.5. The period is then $2^7 - 1 = 127$. The initial state is all ones and the output repeats every 127 digits.

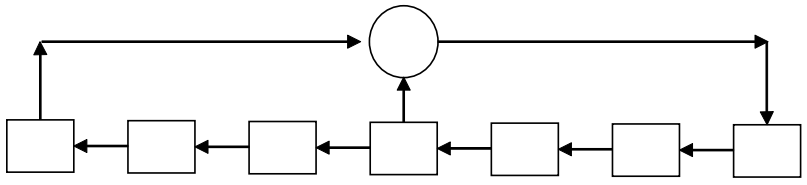


Figure 5.5 Sequence generator for the IEEE 802.11a.

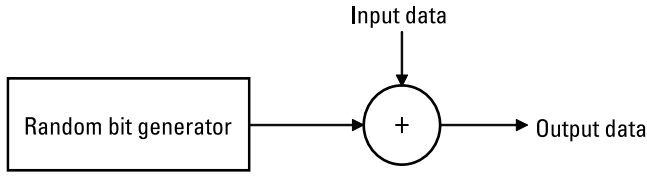


Figure 5.6 Data scrambler.

5.7 Data Scrambler

For military communications, it is required that the incoming data be encrypted before being transmitted. The encryption algorithm is sophisticated so the enemy receiver cannot decrypt it. For commercial communications, it is also necessary to protect sensitive data, which is often scrambled before transmission. Independent of the complexity of the sequence generator, the data scrambler is shown in Figure 5.6.

The data scrambler consists of two parts. The first part is to generate the random bit stream. The second part is the modulo-2 sum of the input data and the random bit stream. For military communications, the random bit generator has an extremely long period. For commercial communications, a maximum-length sequence generator is frequently used to generate the pseudo-random sequence. The IEEE 802.11a uses the polynomial defined in (5.14) to generate a maximum length sequence with a period of 127 bits.

To recover the original data, the receiver has to know the exact random bit generator used in the transmitter. The operation is the reverse process used in the transmitter. The modulo-2 sum of the received data and the output stream from the random-bit generator generates the original data. Figure 5.7 shows the process of data descrambling.

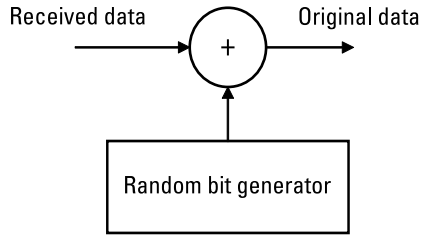


Figure 5.7 Data descrambler.

Example 5.8

Assume that the sequence generator uses $G(x) = 1 + x + x^3$ defined in Example 5.5. The pseudo-random sequence with period 7 is 1110100. Assuming that the input data is 01101100101000, then the scrambler output is 10000101011100 by performing the modulo-2 sum of 11101001110100 and 01101100101000. In the receiver, the modulo-2 sum of the received data 10000101011100 and 11101001110100 generates the original data of 01101100101000.

5.8 Summary

A binary field of GF(2) has only two elements, 0 and 1. The operation of addition, subtraction, multiplication, and division can be defined. The addition is commutative and associative while the multiplication is distributive. A polynomial whose coefficients over GF(2) was then defined. The polynomial addition, multiplication, and division were illustrated.

A Galois field of GF(2^m) has a total of 2^m elements. The operation in this field is over a primitive polynomial whose coefficients are over GF(2). Based upon this primitive polynomial, every element can be generated. This primitive polynomial has a degree m and no element polynomial can have a degree greater than m. In this field, the commutative, associative, and distributive laws are all satisfied and the operation of addition and multiplication is closed. Another interesting property is that every element in this field is also a root of the polynomial $f(x) = x^{2^m-1} + 1$ where m is the order of the primitive polynomial.

A sequence generator through shift registers was then discussed. There are two methods to get the output. The first is to step through the shift register operations cycle by cycle. The second is through polynomial divisions over a generator polynomial. The output through polynomial division was derived in detail.

The generator polynomial has an inverse polynomial. The maximum period of a shift register sequence with a given generator polynomial depends upon this inverse polynomial. Based upon this inverse polynomial, the period depends upon the initial condition of the shift register.

A maximum-length sequence was then defined. To generate this sequence, the generator polynomial is primitive. If the primitive polynomial degree is r, the maximum period is $2^r - 1$. There are two interesting properties. The first is that the number of ones is one more than the number of zeros in the output. The second is that the output is independent of the initial condition of the shift register except for the phase.

Using the maximum-length sequence, a scrambler and descrambler were then discussed. Both operations are through a simple modulo-2 operation.

The next chapter centers on wave propagation loss. Both a large-scale model and a small-scale model are discussed, and the noise model for the amplifier, cable, and receiver is presented.

References

[1] Lin, S., and D. Costello, *Error Control Coding*, Upper Saddle River, NJ: Prentice Hall, 1983.

[2] Peterson, R. L., R. Ziemer, and D. Borth, *Introduction to Spread Spectrum*, Upper Saddle River, NJ: Prentice Hall, 1995.

6

Radio-Wave Propagation Model

6.1 Introduction

In a wireless communication system, an RF carrier is modulated by the baseband signal and propagates through the air. Along the propagation path, it may suffer three major types of obstacles: reflection, diffraction, and scattering. The reflection and scattering are caused by objects with a dimension greater than and smaller than the wavelength, respectively. The reflection can occur from the ground, buildings, or mountains. If scattering occurs, the output wave may travel in different directions in a random manner. The diffraction is caused by an object with sharp edge and the output wave may even travel behind the object.

As a result of these three mechanisms, the receiver may receive signals traveling from different directions to cause multipath distortion. This phenomenon can be characterized from the point of view of large-scale path loss or small-scale fading.

The large-scale propagation loss centers on how the received power varies as a function of the distance between the transmitter and the receiver. As the propagation distance increases, the received power decreases. However, the signal may be received from different paths with varying phase and amplitude fluctuations. The small-scale fading model concentrates on how the received power fluctuates with distance around a few wavelengths.

6.2 Large-Scale Propagation Model

Two well-known models, free space and two-ray, are presented next. The empirical model can also be used through curve fitting of the experimental data. A frequently used empirical formula is also discussed.

6.2.1 Free-Space Propagation Loss

If the electromagnetic wave propagates through a free space, then the power received by a receiver at a distance d from the transmitter is related by the following formula [1],

$$P_r = \frac{P_t G_t G_r c^2}{\Omega^2 d^2} \tag{6.1}$$

where P_r is the received power, P_t is the transmitted power, G_t is the transmitter antenna gain, G_r is the receiver antenna gain, c is the velocity of light, Ω is the radian frequency, and d is the propagation distance. If the carrier frequency is f, then $\Omega = 2\pi f$. Therefore, the received power is inversely proportional to the square of both distance and frequency. Lower frequency and shorter distance can increase the power received.

Equation (6.1) can also be written using the effective isotropic radiated power (EIRP). The EIRP is the power radiated from an isotropic antenna with unit gain uniformly in all directions and is defined here:

$$\text{EIRP} = P_t G_t \tag{6.2}$$

Using (6.2), the received power then becomes

$$P_r = \frac{(\text{EIRP}) G_r c^2}{\Omega^2 d^2} \tag{6.3}$$

The free-space propagation loss in decibels is defined here:

$$\begin{aligned} \text{PL(dB)} &= 10\log\left(\frac{P_t}{P_r}\right) \\ &= 20\log(d) + 20\log(\Omega) - 10\log\left(G_t G_r c^2\right) \end{aligned} \tag{6.4}$$

Example 6.1

Assuming $G_t = G_r = 1$, Figure 6.1 is a plot of propagation loss with respect to the distance from 1 km to 4 km as the frequency increases from 400

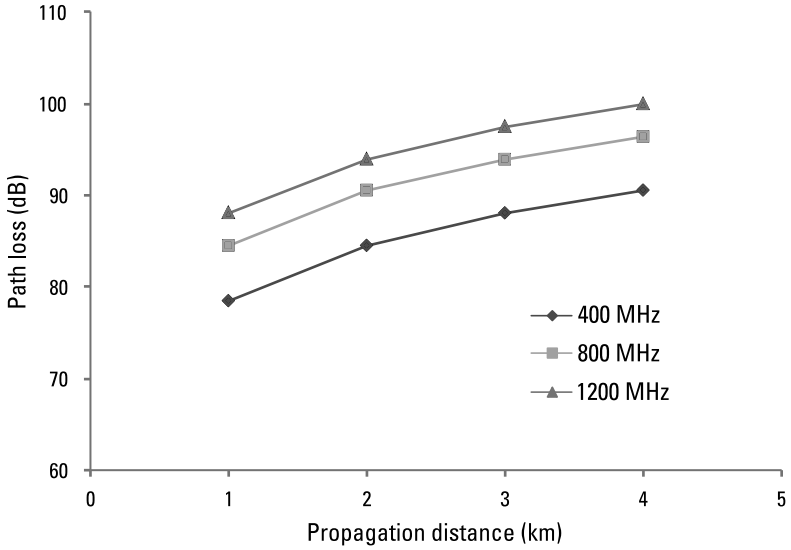


Figure 6.1 Free-space propagation loss in decibels.

MHz to 1.2 GHz. Clearly, the loss increases with the increasing distance and frequency.

Example 6.2

The received power in decibels from (6.4) is $10\log(P_r) = 10\log(P_t) - \text{PL (dB)}$. If the transmitted power is 1W or 30 dBm, the received power is then 30 dBm – PL (dB). Using the same parameters as in Figure 6.1, Figure 6.2 shows the received power in dBm.

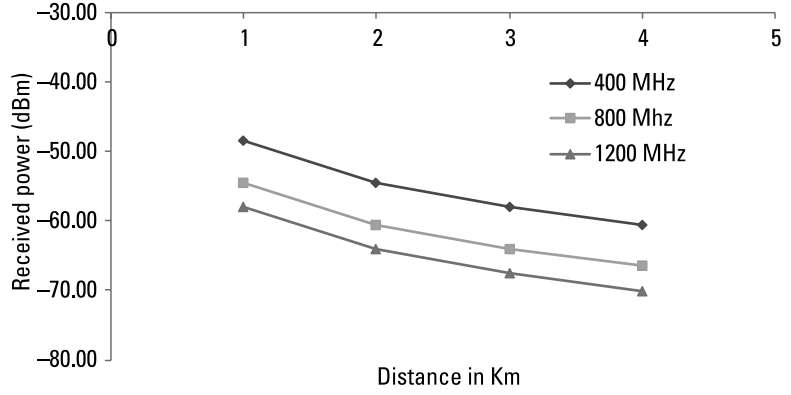


Figure 6.2 Power received using 1W of transmission power.

6.2.2 Two-Ray Model

The free space propagation model does not take into account any obstacles along the propagation path. This model is good if the line-of-sight path is high above the ground. The well-known two-ray model considers two propagation paths. The first is the direct path from the transmitter to the receiver and the second is a path reflected from the ground. The situation is illustrated in Figure 6.3.

At the receiver, the total electric field is the sum received from both paths. The power received at receiver R is given by the following formula [1]

$$P_r = \frac{P_t G_t G_r h_t^2 h_r^2}{d^4} \tag{6.5}$$

where h_t is the transmitter antenna height, h_r is the receiver antenna height, and d is the distance. Equation (6.5) shows that the received power decreases against the distance much faster than the free-space loss. In the free space, the path loss is proportional to d^2. However, in the two-ray model, it is proportional to d^4.

The path loss in decibels is $10\log(P_t/P_r)$ and can be written as

$$\text{PL (dB)} = 40\log(d) - 10\log\left(G_t G_r\right) - 20\log\left(h_t h_r\right) \tag{6.6}$$

As is evident from (6.6), the path loss increases with increasing distance d but decreases with increasing antenna gain and antenna height. One drawback of (6.6) is that there is no frequency dependence. However, the two-ray model is found to predict the path loss reasonably well at large distances.

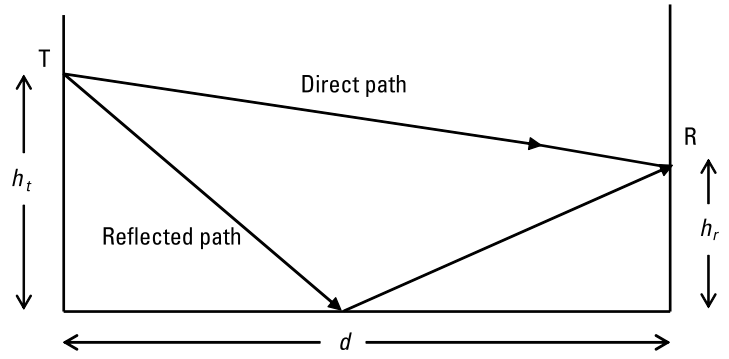


Figure 6.3 Two-ray model.

Example 6.3

Assume that both the transmitter and receiver have unit gain and the antenna height for both transmitter and receiver is 1m. The path loss is plotted against d in Figure 6.4 by setting $G_t = G_r = 1$ and $h_t = h_r = 1$m. Comparing Figures 6.1 and 6.4 at the same distance, it is evident that the two-ray model has a much larger path loss than that predicted by free space.

6.2.3 Empirical Model

The free-space and two-ray models predict the propagation loss either in free space or close to the ground. The actual propagation loss is quite complicated and depends upon many factors. Even for propagation close to the ground, the loss may depend upon ground conductivity, farmland, and desert. It may also depend upon the environment such as forests, open space, and weather. The indoor propagation or obstacles such as buildings or mountains will also impact the propagation loss. In reality, the propagation loss is measured in the field and predicted through curve fitting.

The long-distance propagation loss that fits many theoretical models can generally be given by the following formula:

$$\mathrm{PL}(d,f) = \mathrm{PL}\left(d_r, f_r\right) + 10n\log\left(\frac{d}{d_r}\right) + 10m\log\left(\frac{f}{f_r}\right) + O\ (\mathrm{dB}) \tag{6.7}$$

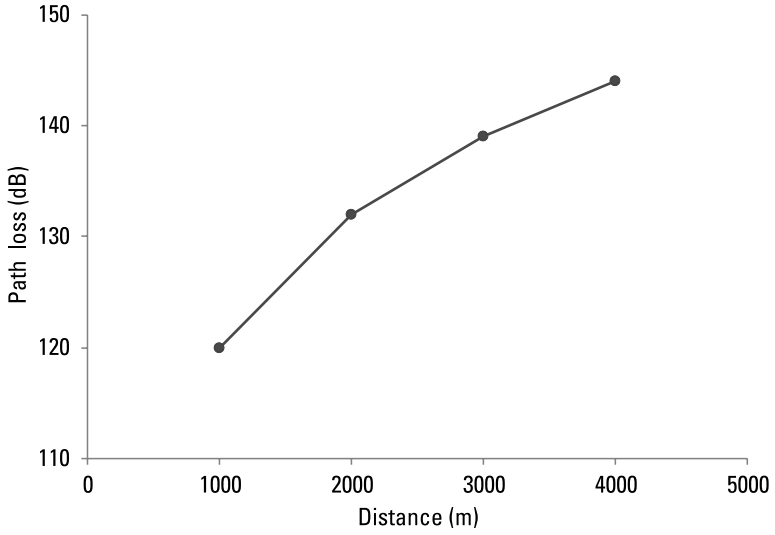


Figure 6.4 Path loss of two-ray model against the distance.

where n and m are variables. The parameter n depends upon the environment and is normally greater than 2 for urban areas and with building obstacles. Since the propagation loss increases with increasing frequency, the third term is added to emphasize frequency dependence. The last term, O, is added so as to include any additional variations against frequency, distance, and others. The parameter d_r is the reference distance and the parameter f_r is the reference frequency. If $d = d_r$ and $f = f_r$, then PL(d, f) reduces to PL(d_r, f_r). The reference distance d_r is normally set at 1 km and PL(d_r, f_r) is measured in the field using the desired reference frequency f_r. If only one frequency is of interest, then $f = f_r$ and the third term is 0. However, if the frequency is changed to a different frequency, then (6.7) can be used to predict the path loss using the new frequency f. If there is no other complicated dependence, then the last term is also 0.

To see how (6.7) can be applied to the theoretical model, consider first the free-space formula given in (6.4). By performing PL(d) − PL(d_r), we have the following

$$\mathrm{PL}(d) = \mathrm{PL}\left(d_r\right) + 20\log\left(\frac{d}{d_r}\right) + 20\log\left(\frac{f}{f_r}\right) \tag{6.8}$$

Comparing (6.7) against (6.8), we have $n = 2$, $m = 2$. Note that the O term is zero since there is no other dependence on frequency and distance.

Consider next the two-ray formula given in (6.6). By again performing PL(d) − PL(d_r), the following is obtained

$$\mathrm{PL}(d) = \mathrm{PL}\left(d_r\right) + 40\log\left(\frac{d}{d_r}\right) \tag{6.9}$$

Comparing (6.7) against (6.9), we have $n = 4$ and $m = 0$, and the O term is 0.

A more complicated path loss in the urban area using frequency from 150 MHz to 1,500 MHz is given below by Hata [1]

$$\begin{aligned}\mathrm{PL}(d, f) = {} & 69.55 + 26.16\log f - 13.82\log h_{te} - a\left(h_{re}, f\right) \\ & + \left(44.9 - 6.55\log h_{te}\right)\log d\end{aligned} \tag{6.10}$$

where f is the frequency in megahertz, h_{te} is the effective transmitter antenna height, and h_{re} is the effective receiver antenna height. Both h_{te} and h_{re} are in

units of meters. The variable $a(h_{re}, f)$ is a correction factor depending upon the city size and frequency. By setting $f = f_r$ and $d = d_r$, (6.10) then becomes

$$\begin{aligned} \mathrm{PL}(d_r, f_r) = {} & 69.55 + 26.16 \log f_r - 13.82 \log h_{te} - a\left(h_{re}, f_r\right) \\ & + \left(44.9 - 6.55 \log h_{te}\right) \log d_r \end{aligned} \tag{6.11}$$

Subtracting (6.11) from (6.10), we have the following:

$$\begin{aligned} \mathrm{PL}(d, f) = {} & \mathrm{PL}(d_r, f_r) + 26.16 \log\left(\frac{f}{f_r}\right) - \left(a\left(h_{re}, f\right) - a\left(h_{re}, f_r\right)\right) \\ & + \left(44.9 - 6.55 \log h_{te}\right) \log\left(\frac{d}{d_r}\right) \end{aligned} \tag{6.12}$$

Comparing (6.12) against (6.7), we have $n = 4.49 - 0.655 \log h_{te}$, $m = 2.616$ and, $O = a(h_{re}, f_r) - a(h_{re}, f)$.

In reality, the measured path loss could be different at different locations even if the distance and frequency are the same. To represent this phenomenon, the path loss is rewritten in the following form:

$$\mathrm{PL}(d, f) = \mathrm{PL}\left(d_r, f_r\right) + 10n \log\left(\frac{d}{d_r}\right) + 10m \log\left(\frac{f}{f_r}\right) + O\ (\mathrm{dB}) + X_\sigma \tag{6.13}$$

where X_σ is a random variable having a normal distribution with zero mean and standard deviation σ. Since X_σ has a normal distribution in the log domain, this phenomenon is also called log-normal shadowing.

Through field measurement, the parameters n and m can be determined through least square curve fitting to minimize the difference between the theoretical prediction and the measured data. The parameters n and m are not necessarily 2 as shown in (6.12). The standard deviation σ is also determined through curve fitting. Equation (6.13) emphasizes the fact that the measured path loss at a specified distance and frequency f is random following a log-normal distribution.

6.3 Small-Scale Propagation Model

The small-scale propagation model deals with the amplitude, phase, and frequency variations in a short period of time. Because of these variations,

the received power fluctuates with respect to time. When a signal propagates through the media, it may be reflected from the surrounding structures such as the ground, building, and mountain. These reflected waves may suffer different propagation delays and arrive at the receiver at different times. The resulting signal is the vector sum of all these component signals. Due to the amplitude and phase difference of each arriving signal, signal distortions can occur.

However, the relative motion between the transmitter and the receiver and the time variation of the channel can cause a Doppler shift in the received carrier frequency. Therefore, this Doppler shift can vary with time. With respect to the motion, the Doppler shift is at its highest when the receiver moves away or toward the transmitter.

These two phenomena can be grouped into time dispersion and frequency dispersion. The time dispersion accounts for a multipath channel, while the frequency dispersion accounts for the Doppler shift from the relative motion between the transmitter and the receiver. Both will be discussed in detail in the following sections. The Clark model, which has Rayleigh distribution for the received signal envelope, is also presented.

6.3.1 Time Dispersion

The time dispersion is due to the reception of the same signal from several paths having different propagation delays. Assume that there are N paths and the delay of the ith path is τ_i and let $r_i(t, \tau_i)$ represent the signal received from the ith path, the overall received signal $r(t)$ is then given by

$$r(t) = \sum_{i=0}^{i=N-1} r_i\left(t, \tau_i\right) \tag{6.14}$$

Assume that the transmitted baseband signal is $s(t)$. Since the propagation delay causes a time shift of $s(t)$, (6.14) can also be written as [1, 2]

$$r(T) = \sum_{i=0}^{i=N-1} \alpha_i(t) e^{j\theta_i(t)} s\left(t - \tau_i\right) \tag{6.15}$$

where $\alpha_i(t)$ and $\theta_i(t)$ are amplitude and phase perturbations associated with the ith path. More details will be given in Chapter 12. Because the phase $\theta_i(t)$ is random, the multipath signal may sometimes add constructively and sometimes add destructively. This amplitude variation is called signal fading.

The instantaneous power of the received multipath signal is given by

$$
\begin{aligned}
P(T) &= r(t)r^*(t) \\
&= \sum_{i=0}^{N-1} \left|a_i(t)\right|^2 \left|s\left(t-\tau_i\right)\right|^2 \\
&+ 2\sum_{i=0}^{N-1}\sum_{j>i}^{N-1} \mathrm{Re}\left\{\alpha_i(t)\alpha_j^*(t)e^{j\left(\theta_i(t)-\theta_j(t)\right)}s\left(t-\tau_i\right)s\left(t-\tau_j\right)^*\right\}
\end{aligned}
\tag{6.16}
$$

Both the channel induced amplitude perturbation $\alpha_i(t)$ and phase perturbation $\theta_i(t)$ are random variables. If each path is independent, then the amplitude and phase in one path are uncorrelated with those of the other path. Performing the time average, the second term of (6.16) then becomes 0. Under this assumption, (6.16) becomes

$$
P_{\text{avg}} = E\left(\sum_{i=0}^{N-1}\left|\alpha_i(t)\right|^2\left|s_i\left(t-\tau_i\right)\right|^2\right) \tag{6.17}
$$

where E represents the ensemble time average. If the transmitted signal is unmodulated, there is no loss of generality to assume $s(t)$ to be 1. Equation (6.17) then simplifies to

$$
P_{\text{avg}} = E\left(\sum_{i=0}^{N-1}\left|\alpha_i(t)\right|^2\right) \tag{6.18}
$$

Equation (6.18) shows the time average power is just the summation of the square of average amplitude from each path.

The received power $P(t)$ is also called the intensity profile of the channel. The width of this intensity profile with an essentially nonzero value is the delay spread, $\Delta\tau$. The Fourier transform of $P(t)$ generates the spaced frequency correlation function $\phi(\Delta f)$ defined below [2]:

$$
\phi(\Delta \mathrm{f}) = \int P(t)e^{-j2\pi\Delta ft}\,dt \tag{6.19}
$$

The width of $\phi(\Delta f)$ is called the coherence bandwidth, B_c. The delay spread is proportional to the standard deviation of the multipath delay, σ_τ. If the delay is widely distributed to generate a large standard deviation, then the delay spread $\Delta\tau$ is also large. However, the coherence bandwidth B_c is inversely proportional to the delay spread $\Delta\tau$ or the standard deviation of delay σ_τ. If the delay spread increases, then the coherence bandwidth decreases and vice versa.

The coherence bandwidth can be thought of as the frequency separation such that the channel can pass the signal with little distortion. If the coherence bandwidth is wide, it is a good channel and the delay spread is small. However, it is a bad channel for a narrow coherence bandwidth and the delay spread is large.

Associated with time dispersion, there are flat fading and frequency selective fading. This is related to the signal bandwidth and coherence bandwidth. It is called flat fading if the signal bandwidth is much smaller than the coherence bandwidth. Under this condition, the signal can pass through the channel with little amplitude or phase distortion. However, the signal bandwidth can be wider than the coherence bandwidth. When this happens, different frequency components will have different channel degradations. If the signal passes through this channel, it can suffer severe distortion. Figure 6.5 illustrates the situation. In this figure, B_{s1} is the signal 1 bandwidth, which is larger than the coherence bandwidth and the signal will pass through the channel with distortion. In the second case, B_{s2} is the signal 2 bandwidth, which is smaller than the coherence bandwidth and the signal will pass through the channel with little distortion.

6.3.2 Frequency Dispersion

The source and receiver may not be stationary when the signal is transmitted. Therefore, the relative motion between the transmitter and the receiver

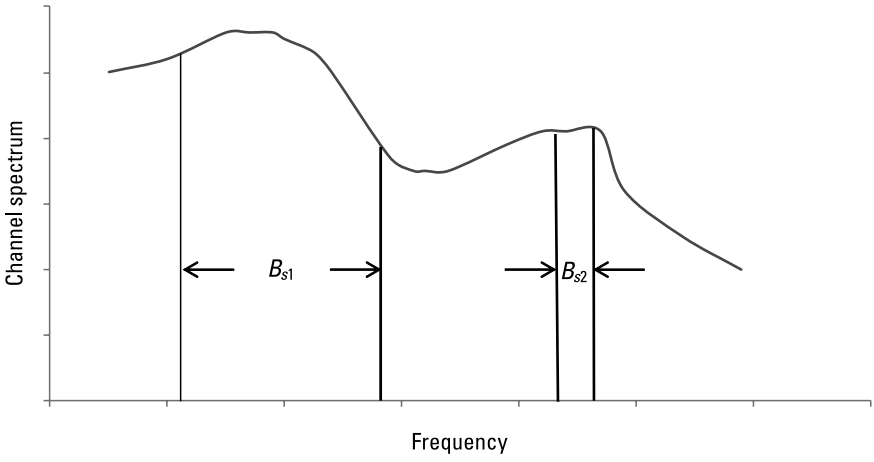


Figure 6.5 Coherence bandwidth in relation to signal bandwidth.

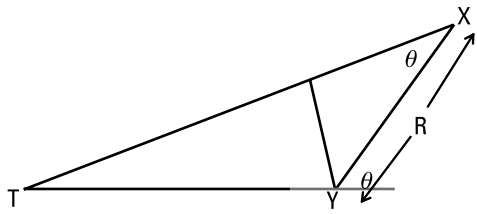


Figure 6.6 Illustration of Doppler shift.

can generate Doppler shift given as f_d. Because of this Doppler frequency, the received carrier frequency moves from f_c to $f_d \pm f_d$. The magnitude of f_d depends upon the motion velocity v and the angle between the direction of motion and the signal propagation direction θ.

Figure 6.6 illustrates the effect of Doppler shift. In this figure, the mobile moves from point X to point Y within a time period of Δt. The source T transmits from T to the mobile. The angle between direction of motion from X to Y and the wave propagation direction from T to Y or from T to X is θ. This is because source is assumed to be far away and the lines from T to X and from T to Y are, in essence, parallel. The distance from X to Y is R. The path difference is then $R\cos\theta$ which generates a phase shift φ given by

$$\varphi = \frac{2\pi R\cos\theta}{\lambda} \tag{6.20}$$

The phase shift φ corresponds to a frequency shift f_d given by

$$\begin{aligned} f_d &= \frac{\varphi}{(2\pi\Delta t)} = \frac{R\cos\theta}{(\lambda\Delta t)} = \frac{v\cos\theta}{\lambda} \\ &= \frac{vf_c\cos\theta}{c} \end{aligned} \tag{6.21}$$

where v is the velocity, c is the speed of light, and λ is the wavelength.

Equation (6.21) indicates that the Doppler shift depends both upon the velocity v and the angle θ. The Doppler shift f_d is maximum when the angle θ is either 0° or 180°. If the angle is 0°, the mobile moves toward the source and the received frequency increases by a maximum amount. If the angle is 180°, the mobile moves away from the source and the received frequency decreases by a maximum amount. If the angle is 90°, there is no Doppler shift even if motion exists.

Example 6.4

For a walkie-talkie handheld radio, the carrier frequency is around 462 MHz. Assuming that angle $\theta = 0$, Figure 6.7 is a plot of Doppler shift in hertz against the car speed from 2 miles/hour to 65 miles/hour. Also shown in the figure are two additional curves for carrier frequency 800 MHz and 1.2 GHz. Clearly, the Doppler shift increases linearly with motion speed. It also increases with carrier frequency. At 2 miles/hour or human walking speed, the Doppler shift is around 1.5 Hz at 462 MHz and is negligibly small. For a car moving at 65 miles/hour and the same carrier frequency of 462 MHz, the Doppler shift is around 49 Hz, which may need to be taken into account depending upon the signal bandwidth.

The Doppler spread, f_m, is the maximum frequency shift of the channel. In Example 6.4, the Doppler spread is 49 Hz at 462 MHz, 85 Hz at 800 MHz and 127 Hz at 1.2 GHz assuming the maximum car speed is 65 miles/hour. If the Doppler spread is much smaller than the signal bandwidth, it can then be neglected. The receiver should take Doppler spread into account if it is not small in comparison to the signal bandwidth. The coherence time, T_c, is a measure of the time variations of the channel and is proportional to the inverse of the Doppler spread. An empirical formula exists which relates the coherence time to the Doppler spread as given below [1]:

$$T_c = \frac{0.423}{f_m} \tag{6.22}$$

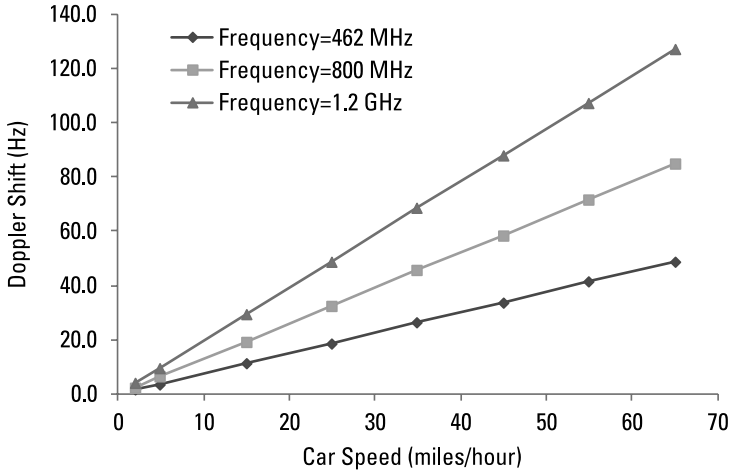


Figure 6.7 Doppler shift against car speed.

The Doppler spread changes little for a large coherence time and changes a great deal for a small coherence time. If the Doppler spread is small, the channel can be considered as time invariant and the signal can pass through with little distortion. However, the coherence time is small if the Doppler spread is large. If the coherence time is smaller than the signal duration, the signal will suffer significant distortion after passing through this channel.

For a time-varying channel, it can be classified as a slow channel or a fast channel. For a fast channel, the coherence time is smaller than the signal duration. Because the channel changes faster with time, it causes Doppler spread that is significant in comparison to the signal bandwidth. This is also called time-selective fading or frequency dispersion.

For a slow channel, the coherence time is longer than the signal duration. Because this channel is essentially flat, the Doppler spread approaches a delta function in the frequency domain. In other words, the Doppler spread is much smaller than the signal bandwidth. This time-invariant nature of the channel is considered to be a good channel. The signal can then pass through it with negligible distortion.

6.3.3 Clark's Fading Model

When a signal passes through a fading channel, it may be scattered in many directions. Clark developed a model based upon flat fading [3]. It is assumed N scattered signals are received by a mobile. Each signal has random amplitude A_n, phase ϕ_n and Doppler shift f_n. From (6.21), the Doppler shift is given here:

$$f_n = \frac{vf_c \cos\alpha_n}{c} \tag{6.23}$$

where α_n is the angle between the nth scattered wave and the direction of the moving mobile. Because of flat fading, all the scattered signals are assumed to arrive at the same time with the same delay. The received signal strength is then the sum of these N scattered signals as given here [1]:

$$r(t) = R_0 \sum_{n=0}^{N-1} A_n \mathrm{Re}\left[e^{j2\pi\left(f_c + f_n\right)t + \phi_n(t)} \right] \tag{6.24}$$

where R_0 is the average signal amplitude and is assumed to be a constant for every scattered signal. Taking the real part of (6.24), the following is obtained

$$\begin{aligned} r(t) &= R_0 \sum_{n=0}^{N-1} A_n \cos\left(j2\pi\left(f_c + f_n \right)t + \phi_n(t) \right) \\ &= U_c(t)\cos\left(2\pi f_c t\right) - U_s(t)\sin\left(2\pi\pi f_c t\right) \end{aligned} \tag{6.25}$$

where the $U_c(t)$ and $U_s(t)$ are defined here:

$$U_c(t) = R_0 \sum_{n=0}^{N-1} A_n \cos\left(2\pi f_n t + \phi_n(t)\right) \tag{6.26}$$

$$U_s(t) = R_0 \sum_{n=0}^{N-1} A_n \sin\left(2\pi f_n t + \phi_n(t)\right) \tag{6.27}$$

From (6.26) and (6.27), the envelope of the received signal is then given by

$$E(t) = \left(U_c(t)^2 + U_s(t)^2\right)^{0.5} \tag{6.28}$$

The Doppler angle f_n is assumed to be small in comparison with f_c. Therefore, $U_c(t)$ and $U_s(t)$ are assumed to be narrowband random process having Gaussian distribution with zero mean and equal variance of $\sigma^2 = R_0^2/2$ [1]. The envelope $r = E(t)$ then has the following Rayleigh distribution [1]

$$p(r) = \frac{r}{\sigma^2} e^{-r^2/\sigma^2} \tag{6.29}$$

The power spectral density $S(f)$ of the received signal was developed by Gans and is given by the following formula [1, 4]:

$$S(f) = \frac{1.5}{\pi f_m \sqrt{1-\beta^2}} \tag{6.30}$$

where $\beta = (f - f_c)/f_m$ and f_m is the maximum Doppler shift. The maximum frequency is $f = f_c + f_m$ corresponding to $\beta = 1$ while the minimum frequency is $f = f_c - f_m$ corresponding to $\beta = -1$. Figure 6.8 is a plot of $\pi f_m S(f)/1.5$ against the normalized frequency β. The spectrum becomes infinite corresponding to $f = f_c \pm f_m$. This happens when the angle of arrival is either 180° or 0°. Between these two extremes, the spectrum is almost flat.

In Clark's model, the received signal envelope has a Rayleigh distribution. This means the spectrum given by (6.30) dictates the time-domain channel

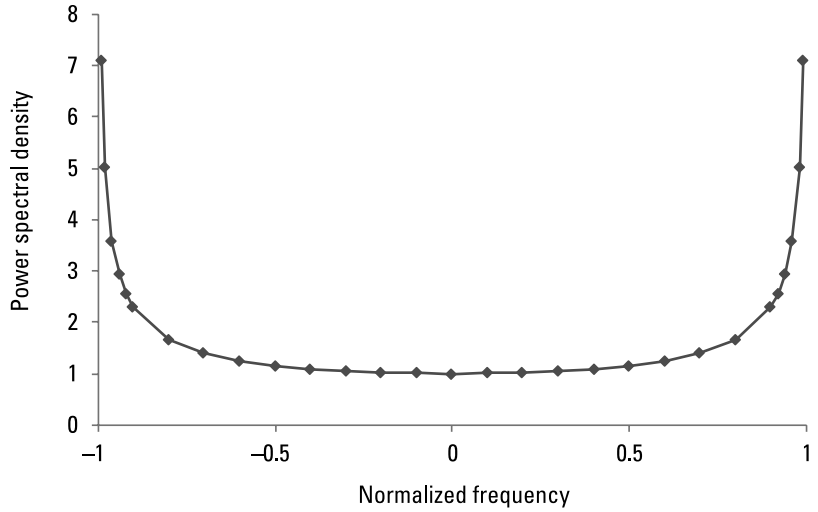


Figure 6.8 Spectrum of Clark's model.

waveform due to Rayleigh fading. In other words, the channel with Rayleigh fading must have a spectrum specified by (6.30).

6.4 Receiver Signal-to-Noise Ratio (SNR)

In the receiver, there are two types of major losses. The first is the propagation loss from the source through the media. This propagation loss can be predicted using either the theoretical or empirical model discussed earlier. The second is the thermal noise at the antenna input. However, there are also amplifier noise and cable loss, which can be referred to at the input. Based upon the received power and the accumulated receiver noise, the SNR can then be predicted.

6.4.1 Thermal Noise

The thermal noise is generated due to random electron motion inside a resistor or active devices. The thermal noise energy is given by [5]

$$N_0 = kT \tag{6.31}$$

where T is the temperature in Kelvin and k is the Boltzmann's constant equal to 1.38×10^{-23} joule/K. Therefore, the thermal noise is proportional to the temperature, and the higher the temperature, the larger the thermal noise.

Equation (6.31) defines the noise energy N_0 or the power density in watts/hertz. If the receiver has a bandwidth of B, then the noise power in watts is given by

$$N = kTB \tag{6.32}$$

6.4.2 Noise Factor

The receiver noise due to amplifier or cable can be referred to its input to generate an equivalent thermal temperature T_e. The noise power density is then given by [5]

$$N_0 = kT_e \tag{6.33}$$

An alternative way to express amplifier noise is to define a noise factor F, which is related to N_0 by the following relationship:

$$N_0 = k(F-1)T_0 \tag{6.34}$$

where T_0 is the room temperature. Equating (6.33) and (6.34) together, we have

$$T_e = (F-1)T_0 \tag{6.35}$$

where F is dimensionless. If the amplifier noise factor F is known, then its equivalent noise temperature T_e can be used to compute the noise power density given by (6.33). From (6.35), the noise factor F can also be given as

$$F = 1 + \frac{T_e}{T_0} \tag{6.36}$$

The noise factor is normally defined in decibels and is given by [5]

$$F_{\text{dB}} = 10\text{Log}\,F \tag{6.37}$$

6.4.3 Amplifier Model

Figure 6.9 shows an amplifier noise model in the receiver front end. Assume this amplifier has a power gain G and equivalent temperature T_e. The power gain G is defined to be the ratio of output power to the input power. Assuming also that the noise power density referred to its input is N_1 and the output noise power density is N_2, then we have

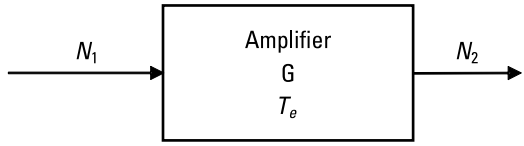


Figure 6.9 Amplifier noise model.

$$N_1 = kT_e \tag{6.38}$$

$$N_2 = GN_1 \tag{6.39}$$

If we know the amplifier output noise power density N_2, the input noise power density N_1 is then given by $N_1 = N_2/G$.

If the amplifier noise factor is F, then (6.38) becomes after using (6.35)

$$N_1 = k(F-1)T_0 \tag{6.40}$$

In other words, the amplifier noise power density referred to its input is known as long as its noise factor is available.

6.4.4 Cable Loss Model

When a signal passes through the cable, it suffers some power loss. The loss factor L is defined to be the ratio of input power to output power. Therefore, N_2 is less than N_1. Figure 6.10 shows the cable loss model and T_e is the equivalent noise temperature. It can be shown [5] that the loss factor L is equal to the noise factor F at room temperature as given here:

$$L = F \tag{6.41}$$

Based upon (6.41), the following are true for the cable loss model

$$N_1 = kT_e = k(L-1)T_0 \tag{6.42}$$

$$N_1 = LN_2 \tag{6.43}$$

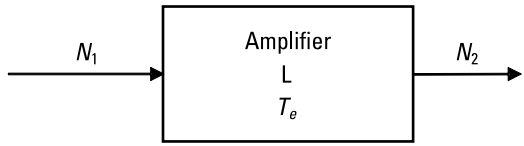


Figure 6.10 Cable loss model.

6.4.5 Equivalent Noise Temperature at Receiver Front End

Figure 6.11 is a general block diagram of the receiver front end. It consists of antenna, cable, amplifier, and receiver in cascade. Between the cable and the receiver, there are n amplifiers. For amplifier k, the gain is G_{ak}, the noise equivalent temperature is T_{ak}, and the noise factor is F_{ak}. For the cable, the loss factor is L and the noise equivalent temperature is T_c. Finally, for the receiver, the gain is G_r, the noise equivalent temperature is T_r, and the noise factor is F_r. Even though each device has its own noise equivalent temperature, what is of interest is the overall noise equivalent temperature T_e at the antenna input.

For the cable, the noise power density referred to its input is simply

$$P_c = kT_c = k(L-1)T_0 \tag{6.44}$$

For the first amplifier, the noise power density referred to its input is given by

$$P_{a1} = kT_{a1} = k\left(F_{a1}-1\right)T_0 \tag{6.45}$$

At the antenna input, the noise power density referred from the first amplifier is then

$$P_{a1,\text{ant}} = Lk\left(F_{a1}-1\right)T_0 \tag{6.46}$$

where we have used the relationship (6.43). For the second amplifier, the noise power density referred to its input is

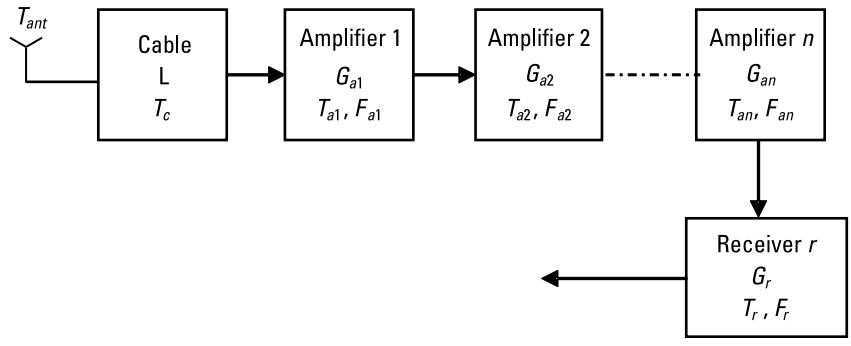


Figure 6.11 Block diagram of the receiver front end.

$$P_{a2} = kT_{a2} = k\left(F_{a2} - 1\right)T_0 \tag{6.47}$$

From the second amplifier to the antenna, there is an amplifier with a gain of G_{a1} and a cable with a loss factor of L, and the noise power density referred to the antenna input is then

$$P_{a2,\text{ant}} = \frac{kL\left(F_{a2} - 1\right)T_0}{G_1} \tag{6.48}$$

For any amplifier i, the noise power density referred to the antenna input can be derived in the same way and is given here:

$$P_{ai,\text{ant}} = \frac{kL\left(F_{ai} - 1\right)T_0}{\prod_{j=1}^{i-1} G_j} \qquad i \geq 2 \tag{6.49}$$

For the receiver, the noise power density referred to the antenna input is derived similarly and is given by

$$P_{r,\text{ant}} = \frac{kL\left(F_r - 1\right)T_0}{\prod_{j=1}^{n} G_j} \tag{6.50}$$

Using all the equivalent noise temperatures referred to the antenna input by combining (6.44) to (6.50), we have

$$\begin{aligned} kT_e = {} & kT_{\text{ant}} + k(L-1)T_0 + Lk\left(F_{a1} - 1\right)T_0 \\ & + LkT_0 \sum_{i=2}^{n} \frac{F_{ai} - 1}{\prod_{j=1}^{i-1} G_j} + \frac{Lk\left(F_r - 1\right)T_0}{\prod_{j=1}^{n} G_j} \end{aligned} \tag{6.51}$$

Removing the common term k from (6.51), we have

$$T_e = T_{\text{ant}} + (L-1)T_0 + L\left(F_{a1} - 1\right)T_0 + LT_0 \sum_{i=2}^{n} \frac{F_{ai} - 1}{\prod_{j=1}^{i-1} G_j} + \frac{L\left(F_r - 1\right)T_0}{\prod_{j=1}^{n} G_j} \tag{6.52}$$

Equation (6.52) is the equivalent noise temperature referred to the antenna input.

Example 6.5

Assume that the receiver has one amplifier with a gain of 30 dB and an equivalent noise temperature of 125K. After the amplifier is a receiver with a noise figure of 15 dB. The room temperature is 293K and the antenna noise temperature is 40K. There is no cable loss. Based upon these assumptions, $n = 1$ and $L = 1$ and (6.52) becomes

$$T_e = T_{\text{ant}} + T_0\left(F_{a1} - 1\right) + \frac{\left(F_r - 1\right)T_0}{G_1}$$
$$= T_{\text{ant}} + T_{a1} + \frac{\left(F_r - 1\right)T_0}{G_1}$$

The amplifier gain $G_1 = 10^{30/10} = 1{,}000$. The receiver noise figure $F_r = 10^{15/10} = 31.62$. Substituting these numbers into the above equation, we have

$$T_e = 40 + 125 + \frac{(31.62 - 1)293}{1000} = 40 + 125 + 8.97 = 173.97$$

Example 6.6

Assume the same configuration as in Example 6.5 except there is a cable loss of 4 dB. Since L is not 0 anymore, (6.52) becomes

$$T_e = T_{\text{ant}} + (L - 1)T_0 + LT_{a1} + \frac{L\left(F_r - 1\right)T_0}{G_1}$$

The cable loss factor $L = 10^{0.4} = 2.51$. Substituting all the numbers into the above equation, we have

$$T_e = 40 + (2.51 - 1)293 + 2.51 \times 125 + \frac{2.51(31.62 - 1)293}{1000}$$
$$= 40 + 442.43 + 313.75 + 22.51 = 818.69$$

6.5 Range Determination

For a given transmitter power, the radio range can be determined once the received power is measured. The empirical path loss formula in (6.7) is given here:

$$\mathrm{PL}(d,f) = \mathrm{PL}\left(d_r, f_r\right) + 10n\log\left(\frac{d}{d_r}\right) + 10m\log\left(\frac{f}{f_r}\right) + O \qquad (6.53)$$

Expressing PL in terms of P_t and P_r, we have

$$\mathrm{P}_t(\mathrm{dB}) - \mathrm{P}_r(\mathrm{dB}) = \mathrm{PL}\left(d_r, f_r\right) + 10n\log\left(\frac{d}{d_r}\right) + 10m\log\left(\frac{f}{f_r}\right) + O \qquad (6.54)$$

Based upon (6.54), the range d can be determined once P_r is measured.

Another receiver parameter frequently used is the receiver signal strength indicator (RSSI). It is the measurement of the minimum received power in order for signal detection. If the received power is less than RSSI, then there is no signal detection. It is normally expressed in dBm or power in milliwatt (mw).

If the receiver sensitivity is high, then RSSI is low. In other words, a good receiver normally has a low RSSI so even a weak signal can be detected.

Example 6.7

Using the two-ray model and the parameter values given in Example 6.3, we have $n = 4$, $m = 0$, $\mathrm{PL}(d_r, f_r) = 40\log(d_r)$, and $O(f, d) = 0$. Equation (6.54) then becomes

$$40\log(d) = \mathrm{P}_t(\mathrm{dB}) - \mathrm{P}_r(\mathrm{dB})$$

Assuming that the transmitted power is 1W = 30 dBm, the range is given in Table 6.1 for various received power. It is clear from Table 6.1 that the range increases as the received power drops.

Table 6.1
The Range at Various Received Power

PT (dBm)	P_r (dBm)	Range (m)
30	–80	562.3
30	–85	749.9
30	–90	1,000
30	–95	1,333.5

6.6 SNR

Based upon (6.54), the received power is given by

$$\mathrm{P}_r(\mathrm{dB}) = \mathrm{P}_t(\mathrm{dB}) - \mathrm{PL}\left(d_r, f_r\right) - 10n\log\left(\frac{d}{d_r}\right) - 10m\log\left(\frac{f}{f_r}\right) - O\ (\mathrm{dB}) \tag{6.55}$$

Assume that the noise equivalent temperature referred to the receiver antenna is T_e. Then the noise power is given by

$$N(\mathrm{dB}) = 10\log\left(kT_eB\right) \tag{6.56}$$

The SNR is obtained by subtracting (6.56) from (6.55) to have

$$\begin{aligned}\mathrm{SNR} = \mathrm{P}_t(\mathrm{dB}) - \mathrm{PL}\left(d_r, f_r\right) - 10n\log\left(\frac{d}{d_r}\right) - 10m\log\left(\frac{f}{f_r}\right) \\ - O\ (\mathrm{dB}) - 10\log\left(kT_eB\right)\end{aligned} \tag{6.57}$$

Example 6.8

Using the two-ray model as in Example 6.7, we have the SNR given by

$$\mathrm{SNR} = \mathrm{P}_t(\mathrm{dB}) - 40\log(d) - 10\log\left(kT_eB\right)$$

Using the same data from Example 6.7 and Example 6.6, we have Te = 818.69K and P_t = 30 dBm. Figure 6.12 is a plot of SNR against the bandwidth in decibels for a propagation distance of 1,000m and 2,000m. The bandwidth is from 30 dB to 60 dB or from 1,000 Hz to 1 MHz. Clearly, as the bandwidth increases, there is more receiver noise in the passband and the SNR decreases. However, as the distance increases, there is greater propagation loss and the SNR decreases as well.

6.7 Summary

For wave propagation through a communication channel, the received signal may be distorted. There are two models to account for this effect. One is the large-scale propagation model and the other is the small-scale propagation model.

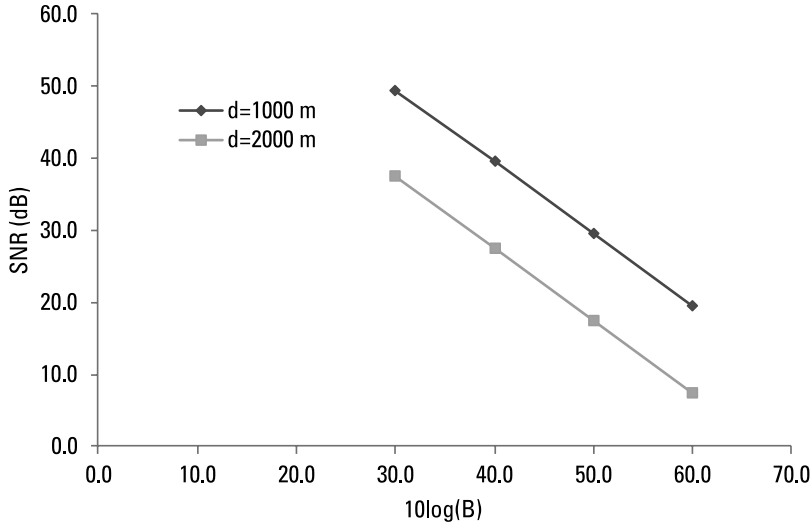


Figure 6.12 The SNR against bandwidth in decibels and propagation distance.

The large-scale model accounts for the power loss after the radio wave travels a distance through the media. Three different theoretical models were presented. The first is the free-space propagation model. The second is the two-ray model, which takes into account the radio wave reflected from the ground. The third is an empirical model developed from field measurements. A general formula that fits many theoretical propagation models was given. The frequency and distance dependence can be determined from field measurements.

The small-scale model accounts for the amplitude, phase, and frequency fluctuations as the radio wave propagates the media. There are two major phenomena. One is the time dispersion and the other is the frequency dispersion. The time dispersion is due to the receiver receiving several signals with different delays. The delay spread is related to the coherence bandwidth and there are flat fading and frequency selective fading. For the flat fading, the signal bandwidth is much smaller than the coherence bandwidth and the signal can pass through with little distortion. For the frequency-selective fading, the signal bandwidth is much wider than the coherence bandwidth and the received signal can suffer severe degradations.

The frequency dispersion is due to the time variation of the channel and the relative motion between the transmitter and the receiver. This Doppler effect is related to the coherence time and there are both fast fading and slow fading. For slow fading, the coherence time is larger compared to the signal

duration and the signal can pass through with little distortion. It this case, the channel can be considered as time invariant. For fast fading, the reverse is true and the signal will suffer significant distortion after passing through this channel.

Next, a noise model referred to the antenna input was discussed. A general receiver front end has an antenna, cable, amplifiers, and receiver. A general noise energy formula was derived based upon this front-end system. An equivalent thermal temperature referred to at the antenna input was subsequently derived. Based upon this equivalent temperature, the SNR was then derived. From the received power measurement, the propagation range can also be predicted.

In the next chapter, two classes of error correction codes are covered. They are block code and convolutional code. The Viterbi algorithm used for convolutional decoding is analyzed in detail. The punctured convolutional code, which is derived from the 1/2 convolutional code, is also discussed. At last, an interleaver used to reduce the block errors is also presented.

References

[1] Rappaport, T. S., *Wireless Communications*, New York: IEEE Press, 1996.

[2] Proakis, J. G., *Digital Communications*, New York: McGraw-Hill, 1983.

[3] Clarke, R. H., "A Statistical Theory of Mobile-Radio Reception," *Bell Systems Technical Journal*, Vol. 47, No. 6, 1968, pp. 957–1000.

[4] Gans, M. J., "A Power Spectral Theory of Propagation in the Mobile Radio Environment," *IEEE Transactions on Vehicular Technology*, Vol. 21, No. 1, 1972, pp. 27–38.

[5] Roddy, D., *Satellite Communications*, New York: McGraw-Hill, 1995.

7

Error-Correcting Codes and Interleaver

7.1 Introduction

In digital communications, a sequence of input bits can be scrambled and encoded before going through the modulation process. The scrambling is to protect the data security and was discussed in Chapter 5. The encoding is to allow the receiver to detect or correct the bit errors due to the transmission through a noisy channel.

There are many different types of error correction schemes. Two types are discussed in this chapter. The first type is the block codes. A block of k information bits is transformed to n bits and $n > k$. The coding rate is k/n. The additional $n - k$ parity bits increase the transmission overheads. However, the added advantage is to enhance the reception accuracies. Both linear block codes and cyclic codes are given. The linear block code is operated through matrix manipulations while the cyclic code is operated using algebraic polynomials. From the generator polynomial of the cyclic codes, the generator matrix of the linear block codes can be derived.

The second type is the convolutional codes. Again, k input bits are converted into n coded bits. However, the input bit sequence is infinite in nature and the output bits are obtained through shift register operations. The coding rate is still defined to be k/n. The Viterbi algorithm is used for decoding. For the convolutional code with a coding rate $R = 1/2$, the transmission redundancy

is 50%. To reduce the overhead bits, the punctured codes are discussed to reduce the overhead transmission. Based upon the 1/2 code, both $R = 2/3$ and $R = 3/4$ punctured codes can be derived. Even though the transmission data rate is increased, the price paid is a slight drop of reception accuracy. If the drop is not significant, it is still a good approach.

7.2 Linear Block Codes

There are only two elements, 0 and 1, in a binary field. In the vector space of n binary bits, there are a total of 2^n such vectors. In its subspace, there exist k independent vectors, $\boldsymbol{g_i}$, $i = 0, k-1$. Each $\boldsymbol{g_i}$ has n bits. Any vector in this k-dimensional subspace can be created by linearly combining these k independent vectors. Supposing that $\boldsymbol{c}$ is any vector in this subspace, then we have

$$\boldsymbol{c} = \sum_{i=0}^{k-1} a_i \boldsymbol{g_i} \tag{7.1}$$

where $a_i = 0$ or 1. Both $\boldsymbol{c}$ and $\boldsymbol{g_i}$ are vectors of n bits. Since the number of binary coefficients a_i is k, there are 2^k such vectors. Equation (7.1) can be thought of as transforming k information bits a_i into a code word of n bits. These 2^k code words are referred to as the (n, k) linear block code. The extra $n - k$ bits are called the parity check bits.

7.2.1 Generator Matrix

In the matrix form, (7.1) can be rewritten as

$$\boldsymbol{c} = \boldsymbol{a}\boldsymbol{G} = \left[a_0, a_1, \ldots, a_{k-1}\right] \begin{bmatrix} \boldsymbol{g_0} \\ \vdots \\ \boldsymbol{g_{k-1}} \end{bmatrix} \tag{7.2}$$

where $\boldsymbol{G}$ is called the generator matrix. Using (7.2), the k information bits are multiplied by the generator matrix $\boldsymbol{G}$ to create the code word $\boldsymbol{c}$. Any k linear independent row vectors $\boldsymbol{g_i}$, $i = 0, k-1$ can be used to create the generator matrix $\boldsymbol{G}$. However, it is more convenient to put the matrix $\boldsymbol{G}$ in the following standard form [1]:

$$\boldsymbol{G} = \begin{bmatrix} \boldsymbol{P} & \boldsymbol{I} \end{bmatrix} \tag{7.3}$$

where $\boldsymbol{P}$ is a $k \times n-k$ matrix while $\boldsymbol{I}$ is a $k \times k$ identity matrix. Specifically, $\boldsymbol{P}$ and $\boldsymbol{I}$ are in the following form:

$$\boldsymbol{P} = \begin{bmatrix} p_{0,0} & \cdots & p_{0,n-k-1} \\ \vdots & \ddots & \vdots \\ p_{k-1,0} & \cdots & p_{k-1,n-k-1} \end{bmatrix} \tag{7.4}$$

$$\boldsymbol{I} = \begin{bmatrix} 1 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & 1 \end{bmatrix} \tag{7.5}$$

The generator matrix $\boldsymbol{G}$ has a dimension of $k \times n$ and the code word $\boldsymbol{c}$ has a dimension of $1 \times n$. The advantage of the form (7.3) is that the code word $\boldsymbol{c}$ has the following form:

$$\boldsymbol{c} = [n-k \text{ parity check bits}, k \text{ information bits}] \tag{7.6}$$

The above equations can be understood more easily by considering the following example.

Example 7.1

A (15,11) linear code has the following generator matrix

$$\boldsymbol{G} = \begin{bmatrix} 1 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{7.7}$$

In the matrix $\boldsymbol{G}$, the first $n-k=4$ columns are the 11×4 matrix $\boldsymbol{P}$ while the next $k=11$ columns are the 11×11 identity matrix $\boldsymbol{I}$. The ith row is the vector $\boldsymbol{g}_i$. For example, row 3 and row 10 are $\boldsymbol{g}_3 = [1\ 1\ 0\ 1\ 0\ 0\ 0\ 1\ 0\ 0\ 0\ 0\ 0\ 0\ 0]$ and $\boldsymbol{g}_{10} = [1\ 0\ 0\ 1\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 1]$. Let $\boldsymbol{a} = [0\ 1\ 0\ 1\ 0\ 1\ 0\ 1\ 0\ 0\ 1]$ be the message to be encoded. Using (7.2) to perform the matrix multiplication, the code word $\boldsymbol{c}$ becomes

$$\begin{aligned} \boldsymbol{c} &= \begin{bmatrix} 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 1 \end{bmatrix} \boldsymbol{G} \\ &= \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 1 \end{bmatrix} \end{aligned}$$

In the code word $\boldsymbol{c}$, the last 11 bits are exactly the message [0 1 0 1 0 1 01 0 01].

7.2.2 Parity Check Matrix

Corresponding to the $k \times n$ generator matrix $\boldsymbol{G}$, there exists another $n-k \times n$ matrix $\boldsymbol{H}$ such that $\boldsymbol{G}$ and $\boldsymbol{H}$ are orthogonal. This matrix $\boldsymbol{H}$ is called the parity check matrix. The matrix $\boldsymbol{H}$ can be created from the $\boldsymbol{G}$ matrix and put in the following standard form [1]:

$$\boldsymbol{H} = \begin{bmatrix} \boldsymbol{I}_{n-k} & \boldsymbol{P}^t \end{bmatrix} \tag{7.8}$$

where $\boldsymbol{I}$ is the $n-k \times n-k$ identity matrix while $\boldsymbol{P}^T$ is a $n-k \times k$ matrix created by the transpose of the matrix $\boldsymbol{P}$. To show $\boldsymbol{G}$ and $\boldsymbol{H}$ are orthogonal to each other, we have

$$\begin{aligned} \boldsymbol{G}\boldsymbol{H}^t &= \begin{bmatrix} \boldsymbol{P} & \boldsymbol{I}_k \end{bmatrix} \begin{bmatrix} \boldsymbol{I}_{n-k} \\ \boldsymbol{P} \end{bmatrix} \\ &= \boldsymbol{P} + \boldsymbol{P} = \boldsymbol{0} \end{aligned} \tag{7.9}$$

Combining (7.2) and (7.9), we have

$$\boldsymbol{c}\,\boldsymbol{H}^t = \boldsymbol{a}\,\boldsymbol{G}\,\boldsymbol{H}^t = \boldsymbol{a}\,\boldsymbol{0} = \boldsymbol{0} \tag{7.10}$$

Therefore, any code word $\boldsymbol{c}$ is orthogonal to the parity check matrix $\boldsymbol{H}$. Since $\boldsymbol{H}$ is an $n-k \times n$ matrix, it is the generator matrix for the $(n, n-k)$ linear block code. Any code word $\boldsymbol{d}$ generated by $\boldsymbol{H}$ can be written as

$$\boldsymbol{d} = \boldsymbol{b}\,\boldsymbol{H} \tag{7.11}$$

where $\boldsymbol{d}$ is a $1 \times n$ matrix, $\boldsymbol{b}$ is a $1 \times n-k$ matrix and $\boldsymbol{H}$ is an $n-k \times n$ matrix. The row matrix $\boldsymbol{b}$ has $n-k$ message bits. Any code word $\boldsymbol{c}$ generated by $\boldsymbol{G}$ and code word $\boldsymbol{d}$ generated by $\boldsymbol{H}$ are orthogonal to each other. This is easily shown by combining (7.2), (7.9), and (7.11) as follows:

$$\boldsymbol{c}\boldsymbol{d}^t = \boldsymbol{a}\,\boldsymbol{G}\,\boldsymbol{H}^t\,\boldsymbol{b}^t = \boldsymbol{0} \tag{7.12}$$

Example 7.2

The parity check matrix $\boldsymbol{H}$ corresponding to (7.7) is the following:

$$\boldsymbol{H} = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 1 & 1 & 1 & 1 \end{bmatrix} \tag{7.13}$$

The first $n-k=4$ columns are the identity matrix. The next 11 columns are the transpose of matrix $\boldsymbol{P}$ in $\boldsymbol{G}$. By performing the matrix multiplication, it can be shown that $\boldsymbol{GH}^t = 0$. Let $\boldsymbol{b}$ = [1 0 0 1] be the message to be coded by $\boldsymbol{H}$, and then $\boldsymbol{d} = \boldsymbol{bH}$ = [1 0 0 1 1 0 1 0 1 1 1 1 0 0 0]. From Example 7.1, $\boldsymbol{c}$ = [0 0 0 0 0 1 0 1 0 1 0 1 0 0 1]. Performing the matrix multiplication again, we have $\boldsymbol{cd}^t = 0$.

7.2.3 Syndrome

Suppose $\boldsymbol{c}$ is the transmitted code word and $\boldsymbol{r}$ is the received code word. If there are no transmission errors, $\boldsymbol{r} = \boldsymbol{c}$. Since $\boldsymbol{c}$ is a code word, it must satisfy $\boldsymbol{cH}^t = 0$. If there are no transmission errors, then $\boldsymbol{s} = \boldsymbol{rH}^t = \boldsymbol{cH}^t = 0$. The product $\boldsymbol{s} = \boldsymbol{rH}^t$ is called the syndrome of $\boldsymbol{r}$ and is a $1 \times n-k$ row matrix.

If the code word $\boldsymbol{c}$ is transmitted through a noisy channel, there can be transmission errors and the received code word $\boldsymbol{r}$ is different from the transmitted code word $\boldsymbol{c}$. The difference between $\boldsymbol{r}$ and $\boldsymbol{c}$ is the error vector $\boldsymbol{e}$ and we can write

$$\boldsymbol{r} = \boldsymbol{c} + \boldsymbol{e} \tag{7.14}$$

Multiplying both sides of (7.14) by $\boldsymbol{H}^t$, the syndrome $\boldsymbol{s}$ then becomes [1]

$$\boldsymbol{s} = \boldsymbol{rH}^t = \boldsymbol{cH}^t + \boldsymbol{eH}^t = \boldsymbol{eH}^t \tag{7.15}$$

Equation (7.15) shows that the syndrome $\boldsymbol{s}$ of the received vector $\boldsymbol{r}$ is equal to the syndrome of the error vector $\boldsymbol{e}$.

If the syndrome $\boldsymbol{s} \neq 0$, then it is clear there are transmission errors. However, if $\boldsymbol{s} = 0$, there is no guarantee that there are no transmission errors. This is because $\boldsymbol{e}$ could be equal to another code word and the linear combination of two code words is also a code word. When this happens, the error is undetectable.

7.2.4 Error Correction

Equation (7.15) shows there is a one-to-one correspondence between $\boldsymbol{s}$ and $\boldsymbol{e}$. Once the received syndrome $\boldsymbol{s}$ of $\boldsymbol{r}$ is computed, the error vector $\boldsymbol{e}$ can be identified. This is because $\boldsymbol{s}$ is equal to the syndrome of the error vector $\boldsymbol{e}$. A table can be created by associating each correctable error vector with its syndrome. Through table look-up, an error vector $\boldsymbol{e}$ corresponding to the received syndrome is located. Once the error vector $\boldsymbol{e}$ is known, the transmitted code word $\boldsymbol{c}$ is then given by

$$\boldsymbol{c} = \boldsymbol{r} + \boldsymbol{e} \tag{7.16}$$

Let $\boldsymbol{C}$ represent a linear block code. The weight of any code word $\boldsymbol{c}$ in $\boldsymbol{C}$ is the number of nonzero bits in $\boldsymbol{c}$. For example, the weight of code word $\boldsymbol{c}$ = [0 0 0 0 0 1 0 1 0 1 0 1 0 0 1] is 5. The distance $d(\boldsymbol{c}, \boldsymbol{v})$ between any two code words $\boldsymbol{c}$ and $\boldsymbol{v}$ of $\boldsymbol{C}$ is the number of places that they differ or the weight of $\boldsymbol{c} + \boldsymbol{v}$. Since $\boldsymbol{c} + \boldsymbol{v}$ is also a code word, the minimum weight of a block code $\boldsymbol{C}$ is then given here:

$$d_{\min} = \min(\text{weight of } \boldsymbol{c}, \text{where } \boldsymbol{c} \text{ is any codeword of } \boldsymbol{C}) \tag{7.17}$$

Assume that t errors can be corrected in a linear block code $\boldsymbol{C}$. Then it can be shown that it is given by [1]

$$t = \frac{\left(d_{\min} - 1\right)}{2} \tag{7.18}$$

where t is the largest integer no greater than $(d_{\min} - 1)/2$.

Example 7.3

The generator matric $\boldsymbol{G}$ in Example 7.1 has k independent rows and each row of n elements is a possible code word of $\boldsymbol{C}$. From these k code words, it is seen that the minimum weight is 3. Therefore, the (15,11) code can correct 1 error.

Example 7.4

From Example 7.3, the (15,11) code can correct 1 error. Since each code word has 15 bits and each bit location is a possible error, there are 15 correctable error patterns. These 15 correctable error patterns and their corresponding syndromes are given in Table 7.1. From (7.15), the syndrome $\boldsymbol{s}$ of each error pattern $\boldsymbol{e}$ is computed from $\boldsymbol{s} = \boldsymbol{e}\boldsymbol{H}^t$. Since there is only one nonzero element in each error pattern $\boldsymbol{e}$, the syndrome $\boldsymbol{s}$ is actually equal to the column vectors of the parity check matrix $\boldsymbol{H}$ given in (7.13).

Table 7.1
Syndrome and Correctable Error Patterns of a (15,11) Code

Syndrome	Correctable Error Patterns
1 0 0 0	1 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 1 0 0	0 1 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 1 0	0 0 1 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 1	0 0 0 1 0 0 0 0 0 0 0 0 0 0 0
1 1 0 0	0 0 0 0 1 0 0 0 0 0 0 0 0 0 0
0 1 1 0	0 0 0 0 0 1 0 0 0 0 0 0 0 0 0
0 0 1 1	0 0 0 0 0 0 1 0 0 0 0 0 0 0 0
1 1 0 1	0 0 0 0 0 0 0 1 0 0 0 0 0 0 0
1 0 1 0	0 0 0 0 0 0 0 0 1 0 0 0 0 0 0
0 1 0 1	0 0 0 0 0 0 0 0 0 1 0 0 0 0 0
1 1 1 0	0 0 0 0 0 0 0 0 0 0 1 0 0 0 0
0 1 1 1	0 0 0 0 0 0 0 0 0 0 0 1 0 0 0
1 1 1 1	0 0 0 0 0 0 0 0 0 0 0 0 1 0 0
1 0 1 1	0 0 0 0 0 0 0 0 0 0 0 0 0 1 0
1 0 0 1	0 0 0 0 0 0 0 0 0 0 0 0 0 0 1

Example 7.5

From Example 7.1, assume that $\boldsymbol{c}$ = [0 0 0 0 0 1 0 1 0 1 0 1 0 0 1] is the transmitted code word. Assume that there is a single bit error in the third bit of $\boldsymbol{c}$ and the received code word $\boldsymbol{r}$ = [0 0 1 0 0 1 0 1 0 1 0 1 0 0 1]. The syndrome $\boldsymbol{s}$ of $\boldsymbol{r}$ is computed here:

$$\boldsymbol{s} = \boldsymbol{r}\boldsymbol{H}^{t} = \begin{bmatrix} 0 & 0 & 1 & 0 \end{bmatrix}$$

From Table 7.1, the error pattern is found to be $\boldsymbol{e}_2$ = [0 0 1 0 0 0 0 0 0 0 0 0 0 0 0]. Adding $\boldsymbol{e}_2$ to $\boldsymbol{r}$, the transmitted code word becomes [0 0 0 0 0 1 0 1 0 1 0 1 0 0 1], which is exactly equal to $\boldsymbol{c}$ and the single error is corrected.

7.2.5 Hamming Codes

Hamming codes [1] are special classes of an (n, k) linear block code. For any integer $m \geq 3$, $n = 2^m - 1$ and $k = 2^m - 1 - m$. The number of parity check

bits is $n - k = m$. The linear (15,11) code given in Example 7.1 is a Hamming code with $m = 4$, $n = 2^4 - 1 = 15$ and $k = 15 - 4 = 11$.

The parity check matrix $\boldsymbol{H}$ of the (15,11) Hamming code is shown in (7.13). The columns of $\boldsymbol{H}$ have all the 4-bit patterns. The sum of any two columns does not add to zero and the error correcting capability is one [1]. Actually, all the Hamming codes can correct 1 error.

7.3 Cyclic Codes

A cyclic code is also a special class of an (n, k) linear block code. If $\boldsymbol{c} = (c_0 c_1 \ldots c_{n-1})$ is a code word, then a cyclic shift to the right generates another code word $\boldsymbol{c}^1 = (c_{n-1} c_0 \ldots c_{n-2})$.

A cyclic code word is represented as a code polynomial $c(x)$ given here:

$$c(x) = c_0 + c_1 x + c_2 x^2 + \ldots + c_{n-2} x^{n-2} + c_{n-1} x^{n-1} \tag{7.19}$$

Every element of the code word $\boldsymbol{c}$ is a coefficient of a polynomial $c(x)$. The highest degree of a code polynomial is $n - 1$. Depending upon the polynomial coefficients, the degree could be less than $n - 1$. A cyclic shift to the right generates the following code polynomial:

$$c^{(1)}(x) = c_{n-1} + c_0 x + c_1 x^2 + \ldots + c_{n-3} x^{n-2} + c_{n-2} x^{n-1} \tag{7. 20}$$

7.3.1 Generator Polynomial

Assume $g(x)$ is a code polynomial of minimum degree in a cyclic code. Then $g(x)$ is a generator polynomial and is unique [1]. An ith shift to the right is equivalent to multiplying $g(x)$ by x^i and $x^i g(x)$ is also a code polynomial.

Example 7.6

Let $\boldsymbol{c}$ = (1 1 0 1 0 0 0) be a code word of a (7, 4) linear code. It is also of minimum degree and $g(x) = 1 + x + x^3$. A cyclic shift two places to the right of $\boldsymbol{c}$ generates $\boldsymbol{c}$ = (0 0 1 1 0 1 0) and $g^{(2)}(x) = x^2 + x^3 + x^5$. Multiplying $g(x)$ by x^2 gives $g^{(2)}(x) = x^2 g(x)$ and $g^{(2)}(x)$ is also a code polynomial.

The message code vector $\boldsymbol{a} = (a_0 a_1 \text{ —— } a_{k-2} a_{k-1})$ of an (n, k) linear code has k elements. The highest degree of a message polynomial is then $k - 1$. The message polynomial $a(x)$ is then given here:

$$a(x) = a_0 + a_1 x + \ldots + a_{k-2} x^{k-2} + a_{k-1} x^{k-1} \tag{7.21}$$

Assume that $g(x)$, $xg(x)$, …., and $x^{k-1}g(x)$ are all code polynomials such that the degree of $x^{k-1}g(x)$ is less than or equal to $n-1$. Since $g(x)$ is a code polynomial of a minimum degree [1], all the code polynomials $x^i g(x)$, $0 < i \leq k-1$ have a degree larger than that of $g(x)$. A linear combination of all these code polynomials must also be a code polynomial $c(x)$, we then have

$$\begin{aligned} c(x) &= \left(a_0 + a_1 x + \ldots + a_{k-2}x^{k-2} + a_{k-1}x^{k-1}\right)g(x) \\ &= a(x)g(x) \end{aligned} \tag{7.22}$$

Assuming that the highest degree of $g(x)$ is r, then $r+k-1=n-1$ and we obtain $r=n-k$. The $g(x)$ is called the generator polynomial and is given by

$$g(x) = 1 + g_1 x + g_2 x^2 + \ldots + g_{n-k-1}x^{n-k-1} + x^{n-k} \tag{7.23}$$

The coefficient of x^{n-k} must be 1; otherwise, another code polynomial of minimum degree exists. From (7.22), every code polynomial is the product of message polynomial $a(x)$ by the generator polynomial $g(x)$.

Multiplying $g(x)$ by x, $x^2 - x^{k-1}$ generates code polynomials of maximum degree $n-1$. They are given here:

$$\begin{aligned} xg(x) &= x + g_1 x^2 + g_2 x^3 + \ldots + g_{n-k-1}x^{n-k} + x^{n-k+1} = g^{(1)}(x) \\ x^2 g(x) &= x^2 + g_1 x^3 + g_2 x^4 + \ldots + g_{n-k-1}x^{n-k+1} + x^{n-k+2} = g^{(2)}(x) \\ &\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\ x^{k-1}g(x) &= x^{k-1} + g_1 x^k + g_2 x^{k+1} + \ldots + g_{n-k-1}x^{n-2} + x^{n-1} = g^{(k-1)}(x) \end{aligned}$$

However, multiplying $g(x)$ by x^k gives the following

$$\begin{aligned} x^k g(x) &= x^k + g_1 x^{k+1} + g_2 x^{k+2} + \ldots + g_{n-k-1}x^{n-1} + x^n \\ &= 1 + x^k + g_1 x^{k+1} + g_2 x^{k+2} + \ldots + g_{n-k-1}x^{n-1} + \left(x^n + 1\right) \\ &= \left(x^n + 1\right) + g^{(k)}(x) \end{aligned} \tag{7.24}$$

Since $g^{(k)}(x)$ is a code word, $g^{(k)}(x)$ can be written as $g^{(k)}(x) = b(x)g(x)$ and (7.24) becomes [1]

$$x^n + 1 = \left(x^k + b(x)\right)g(x) \tag{7.25}$$

Equation (7.25) shows that the generator polynomial $g(x)$ must be a factor of $x^n + 1$ and the degree is $n-k$.

Example 7.7

The generator polynomial of a (15,11) cyclic code can be found from the factors of $x^{15} + 1$.

$$x^{15}+1=\left(x^4+x+1\right)\left(x^{11}+x^8+x^7+x^5+x^3+x^2+x+1\right)$$

The first term has a degree $n-k=4$ and must be a generator polynomial of the (15,11) cyclic code. Therefore, $g(x) = x^4 + x + 1$.

Let $a(x)$ be the message polynomial. Dividing $x^{n-k}a(x)$ by $g(x)$ gives the following:

$$x^{n-k}a(x)=d(x)g(x)+b(x) \tag{7.26}$$

The remainder $b(x)$ has a degree less than $n - k$. The term $d(x)g(x)$ is a code polynomial and we have [1]

$$c(x)=b(x)+x^{n-k}a(x) \tag{7.27}$$

However, the term $x^{n-k}a(x)$ has a degree greater than or equal to $n - k$. Therefore, the information bits are shifted to the highest k location after encoding $a(x)$. Equation (7.27) provides a way to encode the message polynomial such that the information bits come after the parity bits.

Example 7.8

In Example 7.1, the message vector is a = [0 1 0 1 0 1 0 1 0 0 1]. The equivalent message polynomial is $a(x) = x + x^3 + x^5 + x^7 + x^{10}$. The generator polynomial from Example 7.7 is $g(x) = 1 + x + x^4$. Dividing $x^4a(x)$ by $g(x)$, we have

$$\begin{aligned} x^4a(x) &= x^5+x^7+x^9+x^{11}+x^{14} \\ &= \left(x^5+x^6+x^{10}\right)g(x)+0 \end{aligned}$$

The remainder happens to be zero. Therefore, the code polynomial is $c(x) = x^5 + x^7 + x^9 + x^{11} + x^{14}$. The equivalent code word is $\boldsymbol{c}$ = [0 0 0 0 0 1 0 1 0 1 0 1 0 0 1], which is the same as that given in Example 7.1.

Example 7.9

From Example 7.7, the generator polynomial is $g(x) = x^4 + x + 1$. From (7.26), the division of x^4, x^5, x^6, x^7, x^8, x^9, x^{10}, x^{11}, x^{12}, x^{13}, and x^{14} by $g(x)$ gives the following:

$$x^4 = g(x) + 1 + x$$
$$x^5 = xg(x) + x + x^2$$
$$x^6 = x^2 g(x) + x^2 + x^3$$
$$x^7 = \left(x^3 + 1\right)g(x) + 1 + x + x^3$$
$$x^8 = \left(x^4 + x + 1\right)g(x) + 1 + x^2$$
$$x^9 = \left(x^5 + x^2 + x\right)g(x) + x + x^3$$
$$x^{10} = \left(x^6 + x^3 + x^2 + 1\right)g(x) + 1 + x + x^2$$
$$x^{11} = \left(x^7 + x^4 + x^3 + x\right)g(x) + x + x^2 + x^3$$
$$x^{12} = \left(x^8 + x^5 + x^4 + x^2 + 1\right)g(x) + 1 + x + x^2 + x^3$$
$$x^{13} = \left(x^9 + x^6 + x^5 + x^3 + x + 1\right)g(x) + 1 + x^2 + x^3$$
$$x^{14} = \left(x^{10} + x^7 + x^6 + x^4 + x^2 + x + 1\right)g(x) + 1 + x^3$$

Moving $d(x)g(x)$ to the left, we have

$$g(x) = 1 + x + x^4$$
$$xg(x) = x + x^2 + x^5$$
$$x^2 g(x) = x^2 + x^3 + x^6$$
$$\left(x^3 + 1\right)g(x) = 1 + x + x^3 + x^7$$
$$\left(x^4 + x + 1\right)g(x) = 1 + x^2 + x^8$$
$$\left(x^5 + x^2 + x\right)g(x) = x + x^3 + x^9$$
$$\left(x^6 + x^3 + x^2 + 1\right)g(x) = 1 + x + x^2 + x^{10}$$
$$\left(x^7 + x^4 + x^3 + x\right)g(x) = x + x^2 + x^3 + x^{11}$$
$$\left(x^8 + x^5 + x^4 + x^2 + 1\right)g(x) = 1 + x + x^2 + x^3 + x^{12}$$
$$\left(x^9 + x^6 + x^5 + x^3 + x + 1\right)g(x) = 1 + x^2 + x^3 + x^{13}$$
$$\left(x^{10} + x^7 + x^6 + x^4 + x^2 + x + 1\right)g(x) = 1 + x^3 + x^{14}$$

The left sides are all code polynomials. The right sides can be written in matrix forms as follows:

$$\boldsymbol{G} = \begin{bmatrix} 1 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

The matrix $\boldsymbol{G}$ is the same as that given in (7.7). This also points out a way to generate the generator matrix.

7.3.2 Syndrome Polynomial

If the received polynomial $r(x)$ is the same as the transmitted polynomial, then

$$r(x) = c(x) = a(x)g(x) \tag{7.28}$$

In other words, $r(x)$ must be divisible by $g(x)$ and the remainder is zero. In a noisy channel, the received polynomial $r(x)$ may not be the same as the transmitted polynomial and the remainder $s(x)$ may not be zero. Under this condition, $r(x)$ can be written as

$$r(x) = b(x)g(x) + s(x) \tag{7.29}$$

The polynomial $s(x)$ is called the syndrome of $r(x)$. The syndrome $s(x)$ is the remainder after dividing $r(x)$ by $g(x)$. If the syndrome $s(x)$ is not zero, $r(x)$ is not a code polynomial and there are transmission errors. Therefore, the syndrome $s(x)$ can be used to detect whether there are any received bit errors.

Assuming that the difference between the received polynomial $r(x)$ and the transmitted polynomial $c(x)$ is $e(x)$, we can write

$$r(x) = c(x) + e(x) \tag{7.30}$$

Substituting (7.22) and (7.29) into (7.30), we have

$$e(x) = \big(a(x) + b(x)\big)g(x) + s(x) \tag{7.31}$$

Therefore, the syndrome $s(x)$ is also the remainder of dividing $e(x)$ by $g(x)$.

Based upon (7.31), a table can be created to correlate the error patterns $e(x)$ with the corresponding syndromes $s(x)$. After receiving $r(x)$, it is first divided by $g(x)$ to get the syndrome $s(x)$. From the syndrome $s(x)$, the corresponding error pattern $e(x)$ is found from the table. The error pattern $e(x)$ is then added to the received $r(x)$ to recover the transmitted code polynomial $c(x)$ or

$$c(x) = r(x) + e(x) \tag{7.32}$$

The $e(x)$ could be equal to the code polynomial and $r(x)$ is also divisible by $g(x)$. When this happens, the syndrome $s(x)$ is 0 and the error bits are not correctable. The error correction capability is the same as the linear code. Assuming that t errors can be corrected, then t is given by

$$t = \frac{\left(d_{\min} - 1\right)}{2} \tag{7.33}$$

where $d_{\min}$ is the weight of the cyclic code.

Example 7.10

A (15,11) cyclic code with a generator polynomial of $g(x) = 1 + x + x^4$ has a minimum weight of 3. Therefore, it can correct one error. There are a total of 15 error patterns with one bit error. These error patterns together with the corresponding syndromes are listed in Table 7.2.

The syndrome on the second column is the remainder of dividing $e(x)$ in the first column by $g(x)$. Comparing Tables 7.1 and 7.2, it can be seen that they are really identical.

Example 7.11

From Example 7.8, the transmitted polynomial is $c(x) = x^5 + x^7 + x^9 + x^{11} + x^{14}$. Assume that the received polynomial is $r(x) = x^2 + x^5 + x^7 + x^9 + x^{11} + x^{14}$ and an error has occurred in the second bit. The syndrome $s(x)$ of $r(x)$ is the remainder of dividing $r(x)$ by $g(x) = 1 + x + x^4$. After the division, we have $s(x) = x^2$. From Table 7.2, the error pattern is $e(x) = x^2$. Adding $e(x)$ to $r(x)$, we have $r(x) + e(x) = x^2 + x^5 + x^7 + x^9 + x^{11} + x^{14} + x^2 = x^5 + x^7 + x^9 + x^{11} + x^{14}$. Therefore, the original $c(x)$ is recovered.

7.4 Convolutional Code

Similar to the (n, k) linear block code, the convolutional code is defined to be a rate $R = k/n$. For every k-bit input message, there is an n-bit output.

Table 7.2
Error Pattern and Syndrome of a (15,11) Cyclic Code

Error Pattern	Syndrome
1	1
x	x
x^2	x^2
x^3	x^3
x^4	$1 + x$
x^5	$x + x^2$
x^6	$x^2 + x^3$
x^7	$1 + x + x^3$
x^8	$1 + x^2$
x^9	$x + x^3$
x^{10}	$1 + x + x^2$
x^{11}	$x + x^2 + x^3$
x^{12}	$1 + x + x^2 + x^3$
x^{13}	$1 + x^2 + x^3$
x^{14}	$1 + x^3$

The information bits pass through a shift register to generate the output. In general, the shift register has a total of Lk bits. The number L is called the constraint length [2].

Each output bit is generated from a binary adder with input connections to the shift register. If there are n output bits, there are n such binary adders. Similar to the sequence generator described in Chapter 5, the connections to generate each output bit can be specified by a generator polynomial.

7.4.1 Convolutional Encoder

Consider the convolutional encoder given in Figure 7.1. Each input bit ($k = 1$) generates 2 ($n = 2$) output bits. This is then called a 1/2 convolutional encoder. The shift register has a total of 2 bits. Since $k = 1$, the constraint length is equal to 2. As can be seen in Figure 7.1, each output bit is associated with a binary adder. Each adder can be described by a generator polynomial. Let $g_1(x)$ and $g_2(x)$ represent the two polynomials associated with output bits 1 and 2; we then have

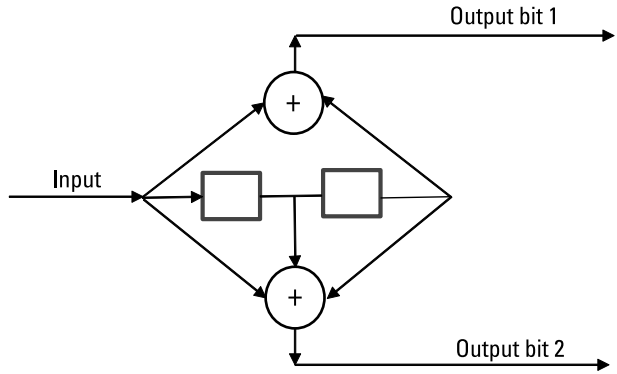


Figure 7.1 A 1/2 convolutional encoder.

$$g_1(x) = 1 + x^2 \tag{7.34}$$

$$g_2(x) = 1 + x + x^2 \tag{7.35}$$

The convolutional encoder can be described by a state-transition diagram. Figure 7.2 shows the state-transition diagram of a 1/2 convolutional encoder. The content of the shift register represents the state. Since the shift register has 2 bits, there are four possible states ranging from 0 to 3. All the numbers in Figure 7.2 are in decimal format. Therefore, the equivalent binary representations of 0, 1, 2, and 3 are 00, 01, 10, and 11. The number inside each circle represents the state. The number beside each branch is the output. The

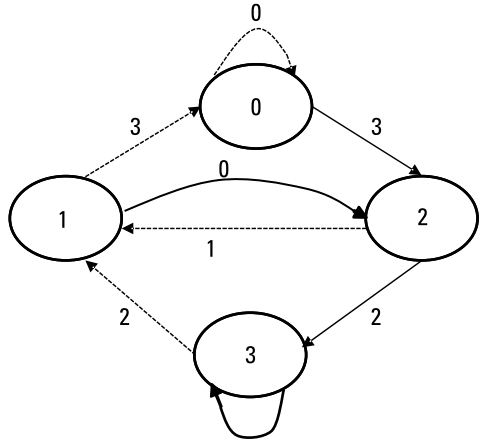


Figure 7.2 State transition diagram of a 1/2 convolutional encoder.

four possible 2-bit outputs are 00, 01, 11, and 10, which can be represented as bauds of 0, 1, 3, and 2. Assume the initial state is 0. If the input bit is 0, the next state remains at zero and the output baud is zero. If the input bit is 1, the state transitions from 0 to 2 and the output baud is 3. The dashed line represents the transition due to a bit 0 input while the solid line represents the transition due to a bit 1 input. Figure 7.2 also shows that there are two possible ways to reach any state. For example, state 1 can be reached from state 2 due to a bit 0 input and from state 3 due to a bit 0 input.

The convolutional encoder can also be represented by a trellis diagram. Figure 7.3 shows the trellis diagram of a 1/2 convolutional encoder. The leftmost column shows the four possible states of 0, 1, 2, and 3. The top two rows show the input bit sequence of either bit 0 or 1 and the corresponding output sequence of 0, 1, 2, or 3. On each branch, there is a symbol of the form a/b. The letter "a" represents the output while the letter "b" represents the single bit input. For example, the symbol 3/1 means a bit 1 input generating an output baud 3. The trellis diagram starts at state 0 with a depth of $d = 0$ and proceeds to the right. The depth d increments by 1 for each new bit input. At $d = 0$, there are two possible transitions. A bit 0 input transits from state 0 to state 0 and outputs 0. A bit one input transits from state 0 to state 2 and outputs 3. At depth 1, there are then two possible states. Since each state has two possible transitions, there are four possible states at depth $d = 2$. After

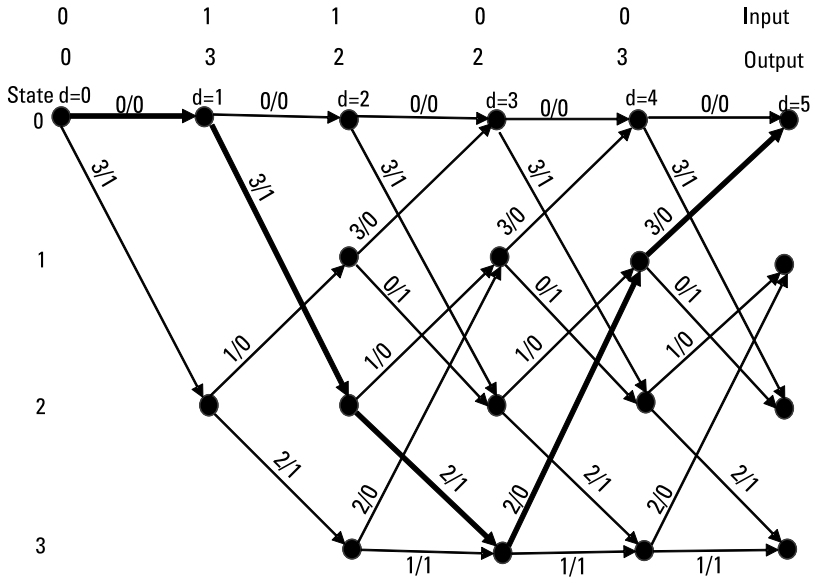


Figure 7.3 Trellis diagram of a 1/2 convolutional encoder.

depth 2, two major events are observed. The first is that the number of states remains at 4. The second is that there are two possible ways to reach any state. For example, at $d = 3$, state 1 can be reached from state 2 due to a bit 0 input and from state 3 also due to a bit 0 input. Following the transition path, an input sequence of 01100 generates an output baud sequence of 03223. The path traced through due to the input sequence is marked by dark lines. After $d = 3$, there are only two paths reaching each state, as will be discussed next.

7.4.2 Convolutional Decoder and Viterbi Algorithm

Figure 7.3 shows the possible number of paths through the trellis increases with the depth. There are two paths at $d = 1$, four paths at $d = 2$, and eight paths at $d = 3$, and, in general, there are 2^k paths at $d = k$. Assume the received semi-infinite symbol sequence is $Y = y_0 y_1 \ldots y_j \ldots$. Every possible path through the trellis diagram generates a possible output code sequence. Assume that the output code sequence of the nth path is $X_n = x_{n0} x_{n1} \ldots x_{nj} \ldots$. Based upon the received symbol sequence, the convolutional decoder finds a path with the highest probability of being followed through the trellis. The metric used is then to maximize $\ln(p(Y/X_k))$. The probability $p(Y/X_k)$ can be written as

$$p\left(\frac{Y}{X_k}\right) = \prod_{j=0}^{j=\infty} p\left(\frac{y_j}{x_{kj}}\right) \tag{7.36}$$

The metric then becomes

$$\ln\left(p\left(\frac{Y}{X_k}\right)\right) = \sum_{j=0}^{j=\infty} \ln\left(p\left(\frac{y_j}{x_{kj}}\right)\right) \tag{7.37}$$

Out of all the possible paths through the trellis, the code sequence X_k such that $\ln(p(Y/X_k))$ is the maximum is selected to be the code sequence input to the channel [3].

Example 7.12

From Figure 7.3, assume there are no channel errors and the received baud sequence is $Y = 03223$. The code sequence through the trellis that most likely matches with Y is the path k such that $X_k = 03223$. This path is marked with the dark line in Figure 7.3. Based upon X_k, the input message sequence is then found to be 01100. No other path can have a code sequence that matches so perfectly. For example, the code sequence corresponding to the binary input

message sequence 01110 is 03212, which does not match with Y as closely as X_k. In other words, ln $(p(Y/X_k))$ is the maximum.

In practice, it is not so easy to accurately determine $p(Y/X_k)$. For hard decision decoding, the metric used is to minimize the Hamming distance between Y and code sequence X_k. Assuming that X and Z are two binary sequences, the Hamming distance is then obtained by performing the exclusive operation of finding the number of places that they differ. For example, the Hamming distance between $X = 11011$ and $Z = 01111$ is 2. It can also be computed as the weight of $X + Z$. For example, the weight of $X + Z = 10100$ is also 2. Table 7.3 lists the Hamming distance between 2-bit bauds and the number inside the parentheses is the baud representation.

The received convolutional codes can be decoded using the Viterbi algorithm [3], which is illustrated in Figure 7.4 using the encoder given in Figure 7.1. Associated with each branch, there is a symbol in the form a/b/c. The additional letter c gives the accumulated Hamming distance at the ending state. The receiver has a replica of the encoder and knows all the possible transmitted codes. Before all four states are generated, all the possible paths are kept. Therefore, four paths are kept before $d = 2$. On each branch, the Hamming distance between branch symbol a and the received symbol is computed first. For example, the output state 1 at $d = 3$ is reached from the input state 2 at $d = 2$. The branch symbol is the encoder output baud $a = 1$ after a bit 0 input and the accumulated Hamming distance c needs an update. The Hamming distance between $a = 1$ and the received baud 2 is 2. The input state 2 is reached from state 0 with an accumulated Hamming distance 0. The cost at $d = 2$ and state 2 is then zero. The new accumulated Hamming distance from state 2 to state 1 is then $0 + 2 = 2$. The letter c is then set to 2. The branch symbol becomes 1/0/2. All the other branch symbols can be computed in a similar manner.

From depth $d = 2$ to $d = 3$, a total of eight possible paths exist. The path is expressed in terms of state sequence. To reach state 0, for example,

Table 7.3
Distance Between 2-Bit Bauds

	00(0)	**01(1)**	**10(2)**	**11(3)**
00(0)	0	1	1	2
01(1)	1	0	2	1
10(2)	1	2	0	1
11(3)	2	1	1	0

there are two paths 0000 and 0210. The other 6 paths are 0021, 0231, 0002, 0212, 0233, and 0023. If all the paths are kept, the number of paths increases exponentially. There are two paths to reach each state at $d = 3$ and only one path with a smaller Hamming distance is kept. All the kept paths are in dark colors. For example, to reach state 0, the path 0000 has accumulated Hamming distance of 3 while the path 0210 has an accumulated distance of 4. The path 0000 is then kept and the path 0210 is discarded. The discarded path will never be used again. After $d = 3$, the same algorithm is used to keep or discard paths. At $d = 3$ and state 1 for example, the path 0021 is kept and the branch symbol 1/0/2 is used to update the accumulated distance to transition from state 1 to state 0 and state 2. Doing it this way, a maximum of 4 paths is maintained at each depth. What has been discussed is one way of discarding paths although other possible ways exist.

At each depth and each state, there are only two paths leading to it. At the end of $d = 6$ and state 0, for example, the two possible paths are from state 0 and state 1 at $d = 5$. When all the symbols are received, the decoder finds the minimum accumulated distance. The minimum distance is zero and the path 0023100 is the final selected path. The encoder output baud sequence and the input bit sequence associated with this path can be found from the symbol on each branch. For example, the path 0023100 has an encoder output baud

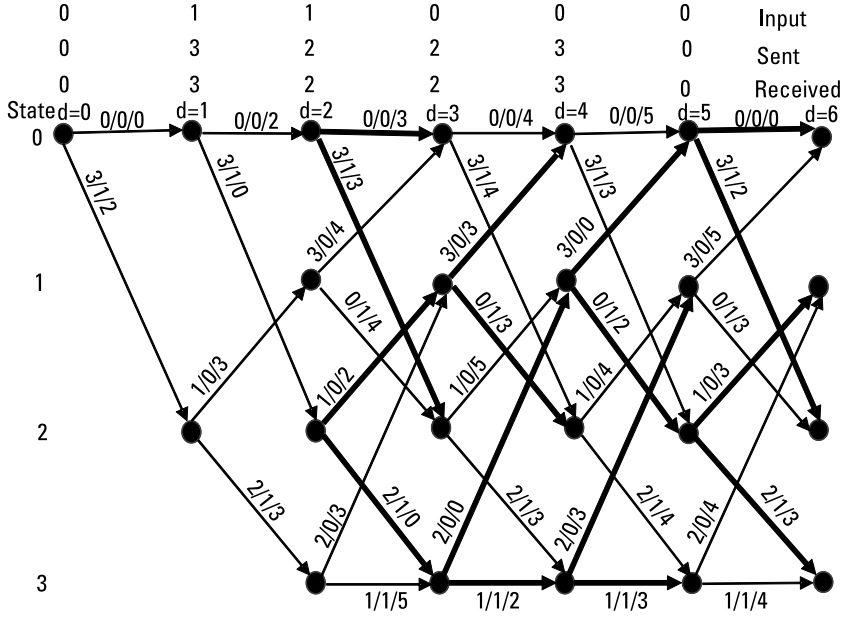


Figure 7.4 Illustration of the Viterbi decoding algorithm.

sequence 032230 and an input bit sequence of 011000. From Figure 7.4, the bit sequence of 011000 is indeed the transmitted message. Figure 7.5 shows the trellis diagram after removing the discarded paths.

Out of the many possible paths through the trellis diagram, the Viterbi algorithm selects the final path with minimum Hamming distance. In this sense, it performs the maximum likelihood decoding. To show how the Viterbi algorithm corrects errors, Figure 7.6 shows the trellis diagram with one transmission error. The input bit sequence is still 011000 and the transmitted encoder baud sequence is 032230. However, the received baud sequence is 033230 and there is a 1-bit error in the third baud. At each depth and state, only one path is chosen. If two paths have the same distance, any path can be selected. One such case is state 1 at depth 5. The branch from state 2 at depth 4 to state 1 has a distance of 3. The same is true for the branch from state 3 at depth 4 to state 1. In Figure 7.6, the branch from state 3 is arbitrarily selected. At $d = 6$, the selected path at states 0, 1, 2, and 3 has an accumulated distance of 1, 3, 3, and 3, respectively.

The minimum accumulated distance is 1 and the path 0023100 is selected. This is indeed the correct path and the input bit sequence is correctly decoded to be 011000. Figure 7.7 shows the trellis diagram after all the discarded paths are removed.

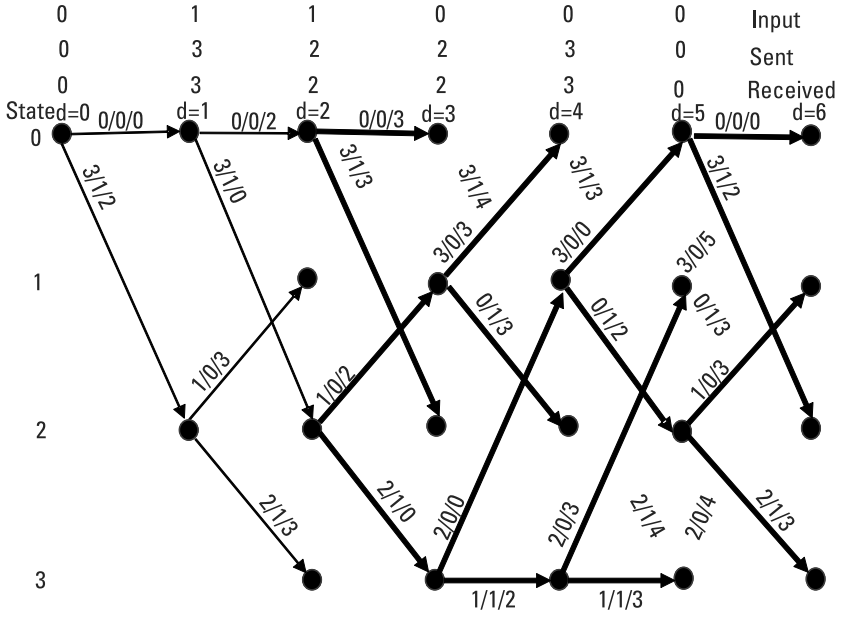


Figure 7.5 Illustration of Viterbi algorithm after removing the discarded paths.

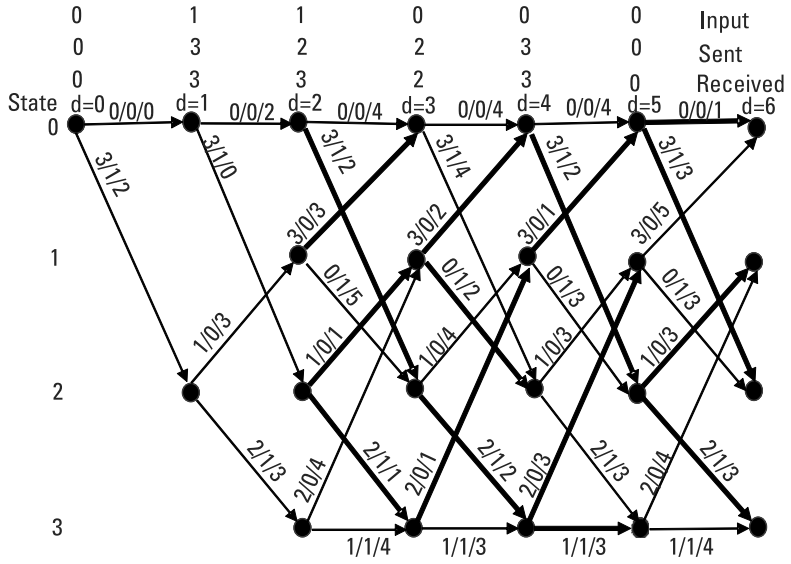


Figure 7.6 Illustration of Viterbi decoding with a single bit error.

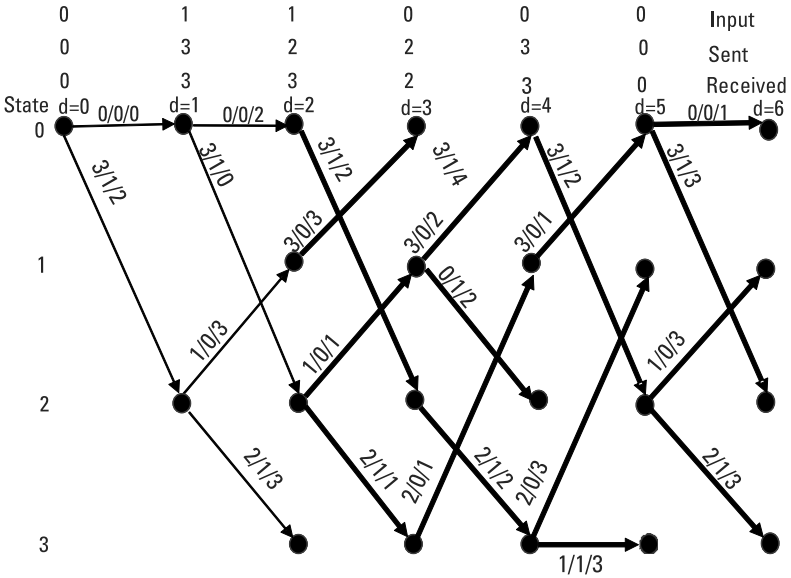


Figure 7.7 Illustration of Viterbi decoding with a single bit error after removing the discarded path.

In comparing Figure 7.5 with Figure 7.7, it is seen that not all the dark lines or paths are the same. This does not matter as long as the correct path has the minimum distance. If there is no transmission error, the correct path has a distance of 0. If there are indeed bit errors, the minimum distance is not 0. However, the minimum distance still selects the optimum path, which can decode the received code word to get the input message stream.

7.4.3 Convolutional Code in the IEEE 802.11a

In the IEEE 802.11a, the convolutional encoder is shown in Figure 7.8. For every single bit ($k = 1$) input, two output bits ($n = 2$) are generated. Therefore, this is still a 1/2 ($k/n = 1/2$) convolutional encoder. The total number of bits in the shift register is 6. Since $k = 1$, the constraint length is 6. The total number of states is then 64. The two generator polynomials are defined by the following equations:

$$g_1(x) = 1 + x + x^3 + x^4 + x^6 \tag{7.38}$$

$$g_2(x) = 1 + x^3 + x^5 + x^6 \tag{7.39}$$

One property of this encoder is that every output state can only be generated from two unique input states. Each input state generates two unique output states. One comes from a bit 0 input and the other comes from a bit

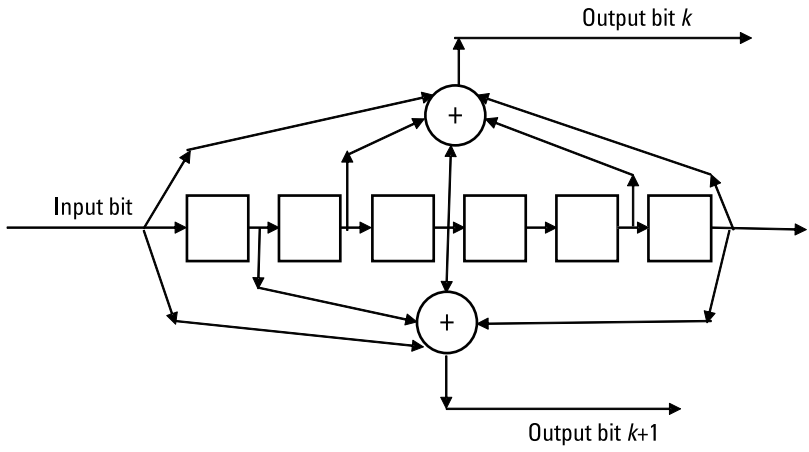


Figure 7.8 Convolutional encoder in the IEEE 802.11a.

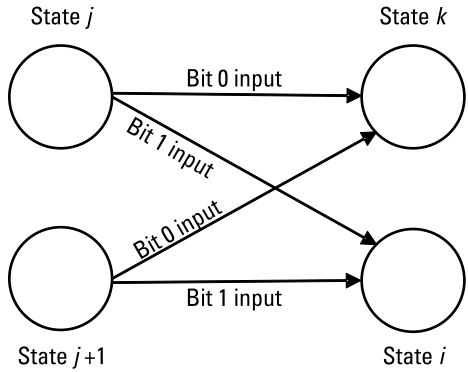


Figure 7.9 Relationship of input and output states.

1 input. Therefore, there exist two input states and two output states that depend upon each other. Figure 7.9 shows the relationship. In this figure, both state j and state $j + 1$ transition to state k after a bit 0 input. However, if the input bit is 1, both input state j and $j + 1$ transition to the same output state i. In other words, there exist two unique input states j and $j + 1$ that can only transition to state k and state i. For example, state 8 ($j = 8$) and state 9 ($j + 1 = 9$) are two input states that can only transition to state 4 ($k = 4$) and state 36 ($i = 36$). For this generator, the two input states differ by 1 while the two output states are generally not consecutive. Table 7.4 lists all the input and output state pairs. For example, input state pair 18 and 19 transitions to output state pair 9 and 41.

Table 7.4
Generation of Input and Output State Pairs

Input/Output	Input/Output	Input/Output	Input/Output
0,1 /0, 32	16,17/8,40	32,33/16,48	48,49/24,56
2,3/1,33	18,19/9,41	34,35/17,49	50,51/25,57
4,5/2,34	20,21/10,42	36,37/18,50	52,53/26,58
6,7/3,35	22,23/11,43	38,39/19,51	54,55/27,59
8,9/4,36	24,25/12,44	40,41/20,52	56,57/28,60
10,11/5,37	26,27/13,45	42,43/21,53	58,59/29,61
12,13/6,38	28,29/14,46	44,45/22,54	60,61/30,62
14,15/7,39	30,31/15,47	46,47 /23,55	62,63/31,63

In the Viterbi decoding algorithm, several different ways of discarding paths are possible. One way discussed before and also justified through simulation in Chapter 12 is summarized here:

- Generate the initial 64 states and keep all the paths through the trellis. During the generation process, compute the Hamming distance between the received baud and the encoder output baud for each branch, and update the accumulated Hamming distance according to the following formula

$$C_{s_{k+1}} = C_{s_k} + d\left(s_{k+1}, s_k\right) \tag{7.40}$$

  where s_k represents the state at depth k, d is the Hamming distance during a state transition, and C is the accumulated Hamming distance.
- After all the initial paths are generated, the total number of states remains at 64. There are two possible paths to reach a new state. Only one path with the minimum accumulated Hamming distance is selected. Mathematically, it is given as

$$C\left(s_{k+1}\right) = \min\left(C\left(s_{k+1}\right), \text{the two paths from } s_k \text{ to } s_{k+1}\right) \tag{7.41}$$

  If the two paths have the same Hamming distance, a path can be selected randomly.
- The process continues until the input message is exhausted. At each new depth, the number of states is 64 and only 64 paths are kept.

7.4.4 Punctured Convolutional Codes

Higher data rates can be generated by employing the puncturing technique. This means certain output bits are stolen without being transmitted. In the receiver, a dummy bit is inserted into the stolen bit location and its distance metric is ignored. The advantage is that the data rate increases by reducing the transmission overhead. The disadvantage is a slight increase in the bit error rate. If the channel is not too bad, the degradation is minimal.

There are two punctured encoders specified in the IEEE 802.11a. The first has a rate of $R = 2/3$ and the second has a rate of $R = 3/4$. Both punctured encoders use 1/2 convolutional coding as a base to generate the desired rate.

Table 7.5 shows the punctured coding at a rate of $R = 2/3$. Each column represents the two output bits for every single bit input. The symbol b_{ij} indicates

Table 7.5
Rate 2/3 Punctured Coding

b_{00}	b_{10}	b_{20}	b_{30}
b_{01}	$\boldsymbol{b_{11}}$	b_{21}	$\boldsymbol{b_{31}}$

the first ($j = 0$) and second ($j = 1$) output bit corresponding to the ith input bit. In other words, the first index is the input bit index while the second index is either the first output bit or the second output bit. For example, b_{30} is the first output bit due to the third input bit while b_{31} is the second output bit due to the same third bit input. The bold text is the output bit that will not be transmitted. Therefore, every fourth output bit is omitted. Table 7.5 shows a 4-bit input stream [4]. After the first 2-bit input, the output bit b_{11} is omitted and after the next 2-bit input, the output bit b_{31} is omitted. The transmitted bit stream is then $b_{00}b_{01}b_{10}b_{20}b_{21}b_{30}$. Instead of generating 2 bits for every single bit input for a rate $R = 1/2$, 3 bits are generated for every 2-bit input to generate a rate of $R = 2/3$.

Table 7.6 [4] shows the punctured coding at rate = 3/4. After every 3-bit input, the fourth and fifth output bits are omitted. Table 7.6 shows that bits b_{11} and b_{20} are omitted after the first 3-bit input and bits $b_{41}$and b_{50} are omitted after the next 3-bit input. The transmitted bit stream is then $b_{00}b_{01}b_{10}b_{21}b_{30}b_{31}b_{40}b_{51}$Therefore, for every 3-bit input, only 4 bits are transmitted and the rate becomes 3/4.

In the receiver, the convolutional decoding follows exactly the same procedure. However, the computation of the Hamming distance is slightly different. Assume after the dummy bits are inserted, b_{k0} and b_{k1} are the received bits and a_{k0} and a_{k1} are the encoder generated output bits before any bits are stolen. If b_{k1} is the inserted dummy bit, then both a_{k1} and b_{k1} are ignored and the Hamming distance is the exclusive or operation between b_{k0} and a_{k0}. However, if a_{k0} is omitted and b_{k0} is the inserted dummy bit, the Hamming distance is the exclusive or operation between a_{k1} and b_{k1}. For example, assume

Table 7.6
Rate 3/4 Punctured Coding

b_{00}	b_{10}	$\boldsymbol{b_{20}}$	b_{30}	b_{40}	$\boldsymbol{b_{50}}$
b_{01}	$\boldsymbol{b_{11}}$	b_{21}	b_{31}	$\boldsymbol{b_{41}}$	b_{51}

that 11 is the encoder generated output baud. Assume also that the received baud is 10 where 0 is the inserted dummy bit. The distance is then the exclusive operation between the first bit 1 of the encoder output baud (11) and the first bit 1 of the received baud (10) and the result is 0.

Except for the slight modification in computing the cost, the Viterbi algorithm can be followed in exactly the same way. The way the cost computation as modified will be justified in the simulation given in Chapter 12.

7.5 Interleaver

Even though the error correction discussed in the previous section can correct bit errors in digital communications, the error performance can be further enhanced through an interleaver. This is because the errors often occur in bursts and a group of continuous bits are in error. When this happens, it is rather difficult for the error correction codes to handle. An interleaver is used to rearrange the bits following certain rules to prevent a block of bit errors.

7.5.1 Illustration of an Interleaver

A block interleaver is commonly used in digital communications. Assume that $b_k, k = 0, N-1$ are the N input bits to the interleaver after error correction coding. Without an interleaver, these N bits are sent in sequence for data modulation. With an interleaver, these N bits are rearranged and sent in a sequence following a given rule.

For illustration purpose, consider a block of $N = 14$ bits. The bit index of the interleaver output is bit reversed from that of the input. Table 7.7 shows the sequence of the input and output bit index. It can be seen that the output bit index of the interleaver is no longer continuous. Following the third column, the bit transmission sequence is then b_0, b_8, b_4, b_{12}, and so forth.

Assume that the beginning two bits b_0 and b_8 are in error after reception. Before error correction, the bit stream is deinterleaved. The deinterleaving is just another bit reversal to restore the original bit stream index as given in the first column of Table 7.7. Note that b_0 and b_8 are no longer adjacent and are now separated by 8-bit positions. Assume that a Hamming (7,4) block code is used for error correction. In the first 7 bits, only one bit is in error and this error can be corrected. Without the interleaver, there are two bits in error and it is beyond the error correction capability of the Hamming code. The advantage of an interleaver can now be clearly seen.

Table 7.7
Bit Index of the Interleaver Input and Output

Decimal Input Bit Index	Binary Input Bit Index	Decimal Output Bit Index	Binary Output Bit Index
0	0000	0	0000
1	0001	8	1000
2	0010	4	0100
3	0011	12	1100
4	0100	2	0010
5	0101	10	1010
6	0110	6	0110
7	0111	14	1110
8	1000	1	0001
9	1001	9	1001
10	1010	5	0101
11	1011	13	1101
12	1100	3	0011
13	1101	11	1011

7.5.2 Interleaver Used in the IEEE 802.11a

In the IEEE 802.11a, the interleaver is defined by a two-step permutation. The first permutation converts the input bit index k to index i according to the following formula:

$$i = \frac{N_{\text{cbps}}}{16}(k \bmod 16) + \text{int}\left(\frac{k}{16}\right) \quad k = 0,1,\ldots,N_{\text{cbps}}\ldots 1 \tag{7.42}$$

where N_{cbps} is the number of coded bits per symbol, int denotes the integral part, and mod is the remainder after the division.

The second permutation converts the index i to index j according to the following formula:

$$j = n\left(\text{int}\left(\frac{i}{n}\right)\right) + \left(i + N_{\text{cbps}} - \text{int}\left(\frac{16i}{N_{\text{cbps}}}\right)\right) \bmod n \tag{7.43}$$

Table 7.8
Input Bit Indexes of an Interleaver in the IEEE 802.11a

0	6	12	18	24	30	36	42
1	7	13	19	25	31	37	43
2	8	14	20	26	32	38	44
3	9	15	21	27	33	39	45
4	10	16	22	28	34	40	46
5	11	17	23	29	35	41	47

where n depends upon the number of bits per subcarrier, N_{bpsc}, and is given here:

$$n = \max\left(\frac{N_{\text{bpsc}}}{2}, 1\right) \tag{7.44}$$

After the two-step permutation, the input bit index k is converted to bit index j.

For BPSK used in the IEEE 802.11a, each OFDM symbol has 24 data bits and 48 coded bits after 1/2 convolutional coding. Table 7.8 shows these

Table 7.9
Output Bit Indexes of an Interleaver in the IEEE 802.11a

0	18	36	7	25	43	14	32
3	21	39	10	28	46	17	35
6	24	42	13	31	2	20	38
9	27	45	16	34	5	23	41
12	30	1	19	37	8	26	44
15	33	4	22	40	11	29	47

48 input bit indexes before an interleaver. Inside each square is the bit index k. The bit stream is sent column-wise from the first column to the eighth column. Without an interleaver, the bit indexes are adjacent and continuous. With an interleaver, the bit indexes are no longer adjacent. Table 7.9 shows the output bit index j and each bit is reassigned. Bit b_k may be assigned to square m and $m \neq k$. For example, bit 1 is assigned to square 16. No two bits can be assigned to the same square. If b_k is assigned to square m and b_j is assigned to square i, then $m \neq i$. Table 7.9 clearly shows the bits are reassigned such that a block of adjacent bit errors can be prevented.

7.5.3 Deinterleaver Used in the IEEE 802.11a

For the IEEE 802.11a, the deinterleaver is the reverse process of the interleaver given in Section 7.5.2. It is completed after two permutations. The first permutation converts the index j to index i and is given by the following equation:

$$i = n\left(\text{int}\left(\frac{j}{n}\right)\right) + \left(j + \text{int}\left(\frac{16j}{N_{\text{cbps}}}\right)\right) \bmod n \qquad j = 0,1,\ldots,N_{\text{cbps}} - 1 \tag{7.45}$$

The second permutation converts the intermediate index i back to the original index k and is given by the following equation:

$$k = 16i - \left(N_{\text{cbps}} - 1\right)\text{int}\left(\frac{16i}{N_{\text{cbps}}}\right) \qquad i = 0,1,\ldots,N_{\text{cbps}} - 1 \tag{7.46}$$

Example 7.13

For BPSK defined in the IEEE 802.11a, the interleaver shown in Section 7.5.2 converts index $k = 1$ to index $i = 16$. In the deinterleaver, the reverse process converts the index $i = 16$ back to index $k = 1$. In other words, Table 7.9 is converted back to Table 7.8.

7.6 Summary

For the block codes, both linear block codes and cyclic codes were discussed. Associated with the linear block codes are a generator matrix and a parity

matrix. The generator matrix is orthogonal to the parity matrix. Every code word can be generated from the generator matrix. If there are transmission errors, a nonzero syndrome is created. Associated with every syndrome is a corresponding error pattern. Based upon the syndrome, the error can be corrected. A special (n, k) Hamming code was also defined such that $n = 2^m - 1$ and $k = n - m$.

For the cyclic code, a cyclic shift of every code word generates another code word. Associated with the cyclic code is a generator polynomial and every code word can be generated through this generator polynomial. For an (n, k) cyclic code, the generator polynomial is a factor of the polynomial $x^n + 1$. As with the linear block code, a syndrome polynomial was defined. If there are no transmission errors, the syndrome polynomial is zero. If there are transmission errors, every syndrome polynomial has a corresponding error pattern. Based upon the syndrome polynomial, the bit errors can be corrected.

For the convolutional code of rate k/n, every k information bits generate n coded bits. The input bit sequence can be infinite in length and the output bit sequence is generated through the contents of a shift register and input bits. The input and output bits can be traced through a trellis diagram. Each possible path is associated with a cost defined to be the Hamming distance between the received baud and the output coded baud. The Viterbi algorithm used for decoding is to find the best path through the trellis such that the cost between the received bits and the encoded output bits is at a minimum. A special convolutional code defined in the IEEE 802.11a was also discussed in more detail.

To reduce the transmission overhead, a punctured convolutional code was defined. It can be generated by taking away a certain number of output bits. For the 1/2 convolutional code, a punctured code of $R = 2/3$ and $R = 3/4$ can be generated. For $R = 2/3$, one output bit is dropped for every two input bits. For $R = 3/4$, two output bits are dropped for every 3 input bits. By ignoring the cost associated with the dropped bits, the Viterbi algorithm can be followed in exactly the same way.

An interleaver is also frequently applied to reduce burst errors. The idea is to distribute the input bits following a certain rule. In such a case, a block of consecutive bits is transmitted far apart.

Upon reception, these errors can then be easily corrected. A special interleaver used in the IEEE 802.11a was discussed.

In the next chapter, the OFDM signal detection using the preamble waveform is discussed. Both coarse detection and fine detection are covered.

References

[1] Lin, S., and D. Costello, *Error Control Coding*, Upper Saddle River, NJ: Prentice Hall, 1983.

[2] Proakis, J. G., *Digital Communications*, New York: McGraw-Hill, 1983.

[3] Viterbi, A. J. "Error Bounds for Convolutional Codes and an Asymptotically Optimum Decoding Algorithm," *IEEE Transactions on Information Theory*, Vol. 13, No. 2, pp. 260–269.

[4] Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications, IEEE Standard 802.11a, 2007.

8

Signal Acquisition

8.1 Introduction

For OFDM, a preamble is generally sent before the data transmission. The preamble normally consists of a repetition of signals. As shown in Chapter 4, the preamble in the IEEE 802.11a has a short sequence followed by a long sequence. The short sequence is a repetition of 16 segments of 16 samples each. The receiver must be able to first detect the presence of such a repetition of signals.

Since the preamble signal repeats, adjacent segments must have strong correlations. This property can be used to define several metrics for signal detection. Three metrics are given in this chapter. They are cross-correlation, normalized cross-correlation, and minimum mean square error (MMSE). Their performances are compared through examples from the simulation study. The simulation is based upon transmitted signals specified in the IEEE 802.11a. The results are compared running on a standard compliant C-program.

From the central limit theorem, each of the three metrics can be considered to have a Gaussian distribution. Using this assumption, a maximum likelihood detection approach is used for signal detection. This assumption is justified from the simulation data.

The front-end electronics must first convert the received signal into the in-phase I component and the quadrature-phase Q component. These

IQ signals are then digitized for further processing for signal detection and others. The traditional heterodyne receiver needs an intermediate frequency (IF) stage to recover the baseband signal. In this chapter, the received signal is directly converted without going through an IF stage.

The detection can be done in two steps. The first step is a coarse timing detection and the second step is a fine timing detection. In order for fast detection, the coarse detection is also further divided into two steps. The first is the segment detection and the second is the sample detection.

Each repetition of the preamble signal is called a segment. Initially, there is no way of knowing in which segment the signal starts. Instead of computing the correlation for each sample, a fast detection strategy is to first compute the correlation for each segment. After detecting the signal segment, the sample detection can be further performed using samples within the detected segment. Each segment in the IEEE 802.11a has 16 samples. The maximum number of correlations for the detected segment is then 15.

After the coarse detection, the sample at which the preamble starts is roughly known. To fine-tune the sample location, a fine detection algorithm is further applied. In the IEEE 802.11a, a long sequence follows immediately after the short sequence. The fine timing detection can be applied using either the long sequence or short sequence. The long sequence is applied for illustration.

The received signals may also suffer impairments from the front-end receiver. Only random noise is discussed in this chapter through simulation study. Others are deferred to Chapter 12.

The normalized cross-correlation is found to have the best performance and is selected for further studies across various SNR levels.

8.2 Direct Conversion to IQ Components

From (2.2), the transmitted signal can, in general, be written as

$$x(t) = \mathrm{Re}\left(s(t)e^{j2\pi f_c t}\right) \tag{8.1}$$

where f_c is the carrier frequency. Assuming that the baseband signal $s(t)$ has amplitude $a(t)$ and phase $\theta(t)$, we have

$$s(t) = a(t)e^{j\theta(t)} = x_I(t) + jx_Q(t) \tag{8.2}$$

The in-phase component $x_I(t)$ and quadrature-phase component $x_Q(t)$ are then given by

$$x_I(t) = a(t)\cos\big(\theta(t)\big) \tag{8.3a}$$

$$x_Q(t) = a(t)\sin\big(\theta(t)\big) \tag{8.3b}$$

Substituting (8.2) into (8.1), $x(t)$ can be rewritten as:

$$x(t) = a(t)\cos\Big(2\pi f_c t + \theta(t)\Big) \tag{8.4}$$

Using (8.3a) and (8.3b), $x(t)$ can be written in terms of $x_I(t)$ and $x_Q(t)$

$$\begin{aligned} x(t) &= a(t)\cos\big(\theta(t)\big)\cos\Big(2\pi f_c t\Big) - a(t)\sin\big(\theta(t)\big)\sin\Big(2\pi f_c t\Big) \\ &= x_I(t)\cos\Big(2\pi f_c t\Big) - x_Q(t)\sin\Big(2\pi f_c t\Big) \end{aligned} \tag{8.5}$$

Assume that the impulse response of the channel is given by $h(t)$. The received baseband signal $y(t)$ is then just the convolution between the transmitted signal, $s(t)$, and the channel impulse response, $h(t)$.

$$y(t) = y_I(t) + jy_Q(t) = \int \Big(x_I(\tau) + jx_Q(\tau)\Big)h(t-\tau)\,d\tau \tag{8.6}$$

where $y_I(t)$ and $y_Q(t)$ are just the convolution of $h(t)$ and $s(t)$

$$y_I(t) = \int x_I(\tau)h(t-\tau)\,d\tau \tag{8.7}$$

$$y_Q(t) = \int x_Q(\tau)h(t-\tau)\,d\tau \tag{8.8}$$

The received signal $r(t)$ is then given by

$$\begin{aligned} r(t) &= \mathrm{Re}\Big(y(t)e^{j2\pi f_c t}\Big) \\ &= y_I(t)\cos\Big(2\pi f_c t\Big) - y_Q(t)\sin\Big(2\pi f_c t\Big) \end{aligned} \tag{8.9}$$

In (8.9), we have ignored the additive random noise for analysis simplification, However, the noise impact will be simulated in later sections. In the RF front end of the receiver, $y_I(t)$ and $y_Q(t)$ can be recovered by passing the received signal through a mixing stage. Without any intermediate frequency, the bandpass signal can be recovered as shown in Figure 8.1. In this mixer, there are two locally generated signals. Assuming, for now, that there is no IQ

imbalance, the first is $l_1(t) = 2\cos(2\pi f_c t)$ and the other is $l_2(t) = -2\sin(2\pi f_c t)$. The signal $l_2(t)$ has exactly a 90° phase shift from $l_1(t)$. Multiplying (8.9) by $l_1(t)$, we have

$$\begin{aligned} r(t)l_1(t) &= 2y_I(t)\cos\left(2\pi f_c t\right)\cos\left(2\pi f_c t\right) - 2y_Q(t)\sin\left(2\pi f_c t\right)\cos\left(2\pi f_c t\right) \\ &= y_I(t)\left(1+\cos\left(4\pi f_c t\right)\right) - y_Q(t)\sin\left(4\pi f_c t\right) \end{aligned} \tag{8.10}$$

After passing through the lowpass filter (LPF), the double frequency terms are filtered. The output becomes $y_I(t)$. Similarly, multiplying (8.9) by $l_2(t)$, we have

$$\begin{aligned} r(t)l_2(t) &= -2y_I(t)\cos\left(2\pi f_c t\right)\sin\left(2\pi f_c t\right) + 2y_Q(t)\sin\left(2\pi f_c t\right)\sin\left(2\pi f_c t\right) \\ &= -y_I(t)\sin\left(4\pi f_c t\right) + y_Q(t)\left(1-\cos\left(4\pi f_c t\right)\right) \end{aligned} \tag{8.11}$$

Similarly, the double frequency term is filtered and the output becomes $y_Q(t)$. Therefore, the upper branch recovers the in-phase component $y_I(t)$ while the bottom branch recovers the quadrature-phase component $y_Q(t)$.

8.3 Detection Metric

The received samples after going through direct conversion becomes

$$y(t) = y_I(t) + jy_Q(t) \tag{8.12}$$

This analog signal then goes through an analog-to-digital converter to generate digitized samples for further processing. The kth such sample is written as $y_k = y_{Ik} + jy_{Qk}$ and y_k is, in general, complex.

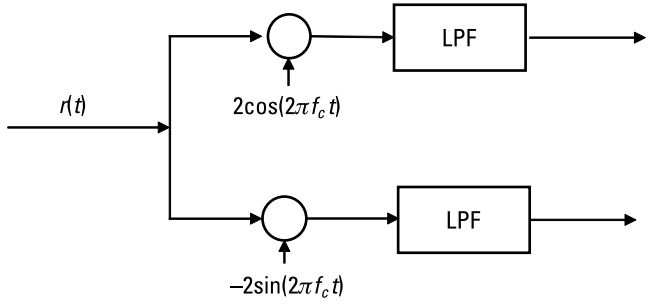


Figure 8.1 Direct conversion receiver without IQ imbalance.

Assume that each repetition segment of the preamble has M samples. In the IEEE 802.11a, the short sequence has $M = 16$ and there are 10 such segments. Since the signal repeats, the following must be true assuming no channel degradation

$$y_{i+k} = y_{i+k+M} \tag{8.13}$$

where i is the start sample index, k is the sample index, and M is the period in sample count. Based upon (8.13), several detection metrics can be utilized.

8.3.1 Cross-Correlation

In this scheme, two signals separated by M samples apart are correlated. The metric m is given here [1]:

$$m_i = \left| \sum_{k=0}^{k=M-1} y_{i+k} y_{i+k+M}^{*} \right| \tag{8.14}$$

In (8.14), the first signal is $R_1 = y_{i+k}$ while the second signal is $R_2 = y_{i+k+M}$. The separation between these two signals is M samples.

The correlation must be low if both signals are noise. The correlation increases if the signal appears in either R_1 or R_2. The correlation reaches a maximum when both R_1 and R_2 are signals. Assuming that there is no noise, then (8.13) applies and (8.14) becomes

$$m_i = \sum_{k=0}^{k=M-1} \left| y_{i+k} \right|^2$$

In this case, m_i is just the energy in one segment.

8.3.2 MMSE Metric

An MMSE metric utilizes the characteristics of (8.13). In other words, the difference between y_{i+k} and y_{i+k+M} must be a minimum when the signal appears. The metric can then be given as [2]

$$m_i = \sum_{k=0}^{k=M-1} \left| y_{i+k} - y_{i+k+M} \right|^2 \tag{8.15}$$

Expanding (8.15), the following is obtained:

$$\begin{aligned} m_i &= \sum_{k=0}^{k=M-1} \left(y_{i+k} - y_{i+k+M}\right)\left(y_{i+k} - y_{i+k+M}\right)^* \\ &= \sum_{k=0}^{k=M-1} \left|y_{i+k}\right|^2 + \sum_{k=0}^{k=M-1} \left|y_{i+k+M}\right|^2 - \sum_{k=0}^{k=M-1} y_{i+k} y_{i+k+M}^* - \sum_{k=0}^{k=M-1} y_{i+k}^* y_{i+k+M} \\ &= \sum_{k=0}^{k=M-1} \left|y_{i+k}\right|^2 + \sum_{k=0}^{k=M-1} \left|y_{i+k+M}\right|^2 - 2\sum_{k=0}^{k=M-1} \mathrm{Re}\left(y_{i+k} y_{i+k+M}^*\right) \end{aligned} \tag{8.16}$$

From (8.15), it can be seen that the metric is always positive. When the signal is present and without noise, (8.13) applies. Then from (8.15), $m_i = 0$ and this is also the minimum value of m_i.

8.3.3 Normalized Cross-Correlation

The metric in (8.16) can be normalized with respect to the signal energy given here:

$$E_i = \sum_{k=0}^{k=M-1} \left|y_{i+k}\right|^2 + \sum_{k=0}^{k=M-1} \left|y_{i+k+M}\right|^2 \tag{8.17}$$

Dividing m_i in (8.16) by (8.17), we have [2]

$$\frac{m_i}{E_i} = 1 - \frac{2\sum_{k=0}^{k=M-1} \mathrm{Re}\left(y_{i+k} y_{i+k+M}^*\right)}{E_i} \tag{8.18}$$

Minimizing m_i/E_i is equivalent to maximizing the second term of (8.18). The normalized metric is then defined below:

$$m_i(\text{normalized}) = \frac{2\sum_{k=0}^{k=M-1} \mathrm{Re}\left(y_{i+k} y_{i+k+M}^*\right)}{E_i} \tag{8.19}$$

Using (8.13) under the condition without noise, m_i in (8.19) is then equal to one. With a signal present, the metric is close to 1. With only noise

present, the correlation is even smaller due to the further division by the energy. Therefore, (8.19) is a good metric to detect the signal in noise.

For all three metrics given in (8.14), (8.16), and (8.19), the search is to find either the maximum or minimum when the signal appears. Since there is no way of knowing when the signal is transmitted, the metric is initially computed each time by advancing the index i by the period M. In other words, we compute m_i, m_{i+m}, m_{i+2M} until m_{i+lM} such that the signal segment is detected. In doing so, more precise signal timing is not known in exchange for fast detection.

8.4 Maximum Likelihood Detection

Even though the transmitter does not send m_i directly, it can be computed from the received samples. To detect the signals in noise, it becomes equivalent to detect the metric in noise. There are now two hypotheses. In the first hypothesis, H_1, the signal is present while in the second hypothesis, H_0, only noise is present.

Let y represent the received metric sample. Based on y, either H_1 is true or H_0 is true. If H_1 is true, the probability density function of y is $p(y/H_1)$. However, the probability density function of y is $p(y/H_0)$ if H_0 is true. The decision rule is then to choose H_1 if

$$\frac{p\left(y/H_1\right)}{p\left(y/H_0\right)} > \lambda \tag{8.20}$$

where λ is a detection threshold [3]. The ratio on the left side of (8.20) is called the likelihood ratio. A test based upon this ratio is called a likelihood ratio test.

Under hypothesis H_1 and H_0, the variable y is given below:

$$H_1:\ y = u + n \tag{8.21}$$

$$H_0:\ y = n \tag{8.22}$$

where n is the random noise and u is the average value of the metric. Assuming that the random noise n to be Gaussian distributed with mean zero and standard deviation σ, the probabilities $p(y/H_1)$ and $p(y/H_0)$ are then given here:

$$p\left(\frac{y}{H_1}\right) = \frac{1}{\sqrt{2\pi\sigma}} e^{\left[-(y-u)^2/2\sigma^2\right]} \tag{8.23}$$

$$p\left(\frac{y}{H_0}\right) = \frac{1}{\sqrt{2\pi\sigma}} e^{\left[-y^2/2\sigma^2\right]} \tag{8.24}$$

Substituting (8.23) and (8.24) into (8.20), we have

$$e^{(2yu-u^2)/2\sigma^2} > \lambda \tag{8.25}$$

Taking logarithm on both sides of (8.25), we have

$$y > \frac{u^2 + 2\sigma^2 \ln\lambda}{2u} \tag{8.26}$$

The decision rule can be to choose H_1 if $p(y/H_1) > p(y/H_0)$. In this case, $\lambda = 1$ and (8.26) becomes

$$y > \frac{u}{2} \tag{8.27}$$

Equation (8.27) is a simple decision rule. If the computed metric is greater than one-half of its average signal value, the signal is detected.

8.5 Coarse Timing Detection

The short sequence defined in Section 4.4.1 is used for the simulation of the coarse detection. The transmitted signal is assumed to go through no channel degradations and synchronization errors. That also means the channel impulse response $h(t) = 1$ in (8.7) and (8.8). However, the random noise impact on the accuracy of signal detection is provided through simulation study.

The coarse detection is done in two steps. The first is the segment detection to quickly locate the segment the signal appears. The second is the sample detection to find the onset of signal samples.

8.5.1 Segment Detection

In the segment detection, the goal is to find the segment where the signal is located. The segment size is M and the metric is computed by advancing the samples by M each time. In other words, only metrics m_0, m_{16}, m_{32}, m_{48}, and so forth are computed. In the following examples, two different noise levels are considered. One has SNR = 37.8 dB and the second has SNR = 2.8 dB.

Example 8.1

In this example, the SNR is set to be 37.8 dB. Figures 8.2 to 8.4 are plots of cross-correlation, normalized cross-correlation, and MMSE against the segment number. Each segment of the short sequence has 16 samples. The index i of the metric m_i is advanced by 16 each time and has numbers 16, 32, 48, and so forth. The segment number is just $i/16$ and has numbers 1, 2, 3, and so forth. The short sequence starts at segment 26.

Figure 8.2 shows that the noise correlation is negligibly small. The cross-correlation is 0.1 at segment number 26 and reaches a maximum of 0.2 for segment number ≥ 27. Since the short sequence samples start within segment 26, the cross-correlation is not at maximum. The average signal level u in (8.27) is then 0.2 and the threshold is $u/2 = 0.1$. Clearly, the signal is correctly detected at segment 26. Figure 8.3 shows that the normalized cross-correlations

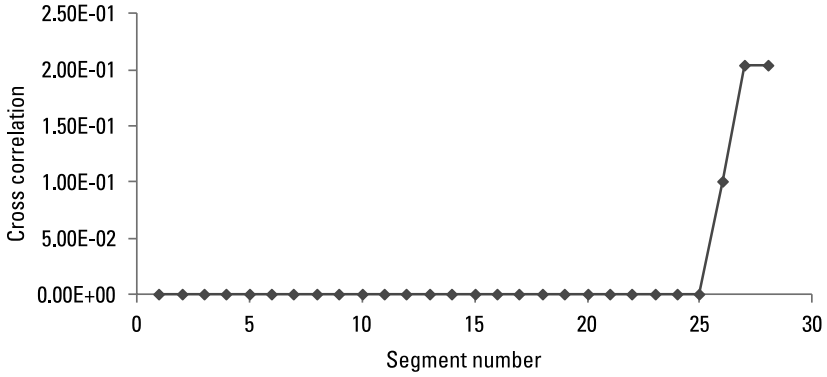


Figure 8.2 Cross-correlation against segment number at SNR = 37 dB.

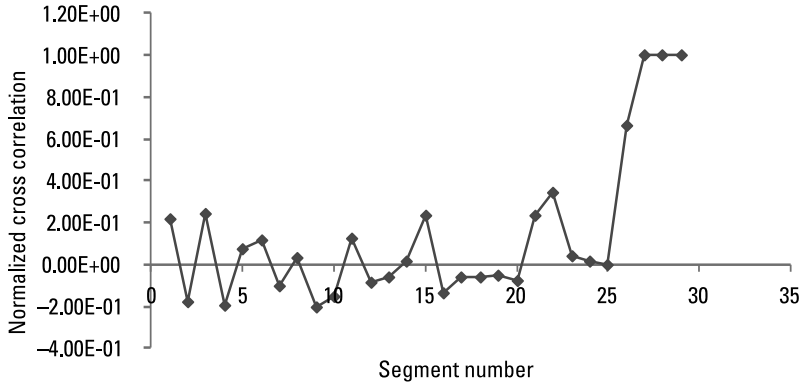


Figure 8.3 Normalized cross-correlation against segment number at SNR = 37.8 dB.

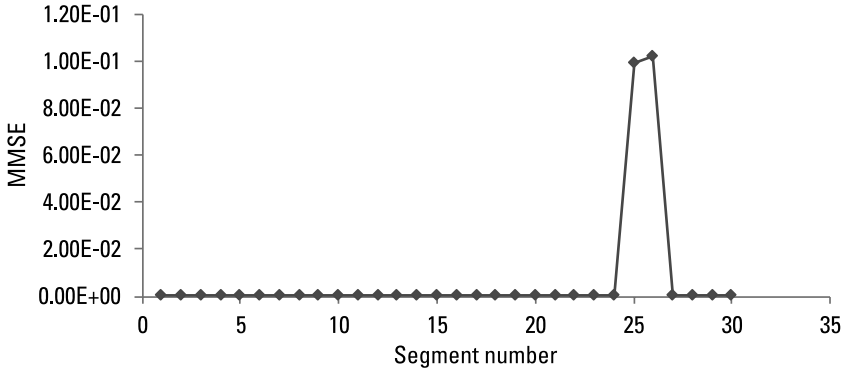


Figure 8.4 MMSE against the segment number at SNR = 37.8 dB.

maximum is 1. The threshold is then 0.5. The signal is also correctly detected at segment 26 with a normalized cross-correlation of 0.66. Since the signal is normalized, the noise correlation is not small anymore and the value can be either positive or negative as expected.

Figure 8.4 shows the MMSE and the detection is to find the minimum and (8.27) does not apply. Because of signal and noise mismatch at segments 25 and 26, the MMSE has a larger value. When the signal matches again starting in segment 27, the MMSE does reach a minimum. Since there is a peak at segment 26, it cannot be detected as the starting signal segment. Another issue is that the minimum is almost indistinguishable for noise, and thererfore, confusion can arise.

Example 8.2

In this example, the SNR is set to be 2.8 dB. Figures 8.5 to 8.7 are plots of the three metrics against the segment number. For cross-correlation, Figure 8.5 shows at segment 26, the metric is 0.16 and is greater than the threshold 0.1. Even with significant noise, the signal is still correctly detected. At segment 25, the metric is 0.069 and there is no false alarm. Figure 8.6 shows the normalized cross-correlation. At segment number 25, the metric is −0.029, and at segment number 26, the metric is 0.586. Therefore, the signal is still correctly detected at segment 26. Figure 8.7 shows the MMSE. At segment number 25, the signal has a peak and drops immediately starting at segment 26. Again, the miniumum is almost indistinguishable from the noise.

The simulation justifies the maximum likelihood detection strategy. Both cross-correlation and normalized cross-correlation work well. However,

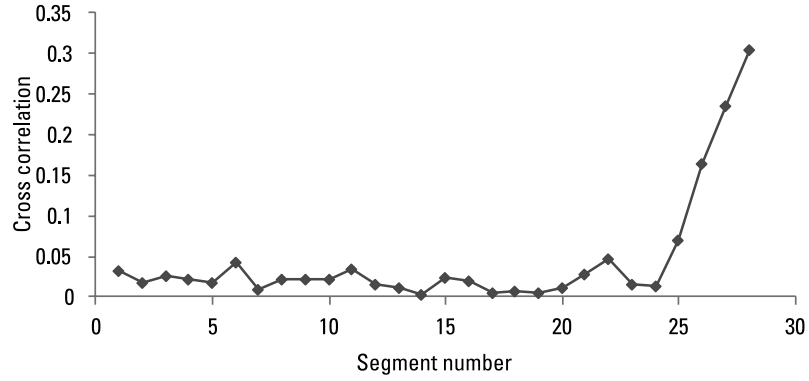


Figure 8.5 Cross-correlation against the segment number at SNR = 2.8 dB.

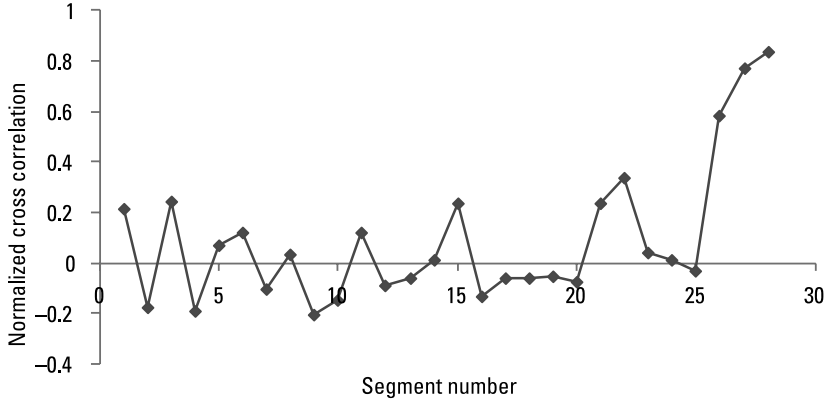


Figure 8.6 Normalized cross-correlation against segment number at SNR = 2.8 dB.

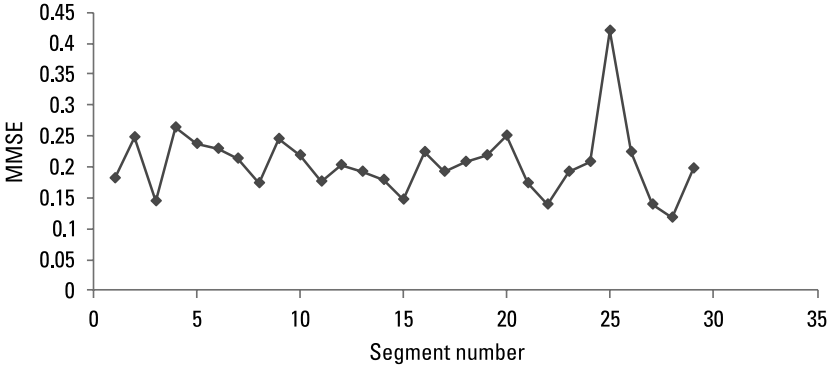


Figure 8.7 MMSE against the segment number at SNR = 2.8 dB.

these are by no means the only methods to detect the signal in noise. Another standard method is to estimate the noise level and its standard deviation. For signal detection, the signal level must be greater than the noise level by at least the noise standard deviation.

For cross-correlation, the threshold parameter u in (8.27) can be set at its maximum value. As will become clear later, this value is difficult to set in advance. For normalized cross-correlation, it is theoretically set at 1 and is a constant. In practice, it can be set at slightly less than 1 in order not to miss the sample onset due to interference and others.

8.5.2 Sample Detection

After locating the signal segment, the next step is to locate the starting sample. In the segment detection, the index i of the metric m_i increases by 16 each time. In the sample detection, i has to advance by 1. In other words, only m_1, m_2, m_3, and so forth are computed. The question is what should be the value of M or the number of samples used for computing the correlation. If M is set to the maximum of 16, the correlation can be too large to distinguish between noise and signal. This is because too many signal samples are included in the computation. This means that M should be set small.

The impact of SNR and M are investigated through the following two examples. Again, the SNR is either 37.8 dB or 2.8 dB. Two different M values are used. One is 16 and the other is 2. Since the MMSE has the worst performance, only cross-correlation and normalized cross-correlation are used for the simulation study.

Example 8.3

From previous examples, the signal segment is 26. The starting sample is $16 * 25 = 400$ since the first segment is 1. Figure 8.8 is a plot of cross-correlation against sample number within segment 26 at SNR = 37.8 dB. The top figure uses $M = 16$ for correlation while the bottom figure uses $M = 2$ for correlation. Since the signal appears somewhere after a certain sample number, the signal u can be set at a sample number of 15. The threshold of the top figure is then $0.5m_{15} = 0.1$. However, the correlation is greater than 0.1 for all the sample numbers and it is not clear where the signal starts. In the bottom figure, the correlation is much smaller since only 2 samples are used in the correlation. The threshold is then $0.5m_{15} = 0.5 * 0.022 = 0.011$. The curve shows the correlation is much less than 0.011 until sample number 8. Since sample number 8 has a correlation greater than the threshold 0.011, it is correctly identified as the signal starting sample. Figure 8.9 is a plot of normalized cross-correlation

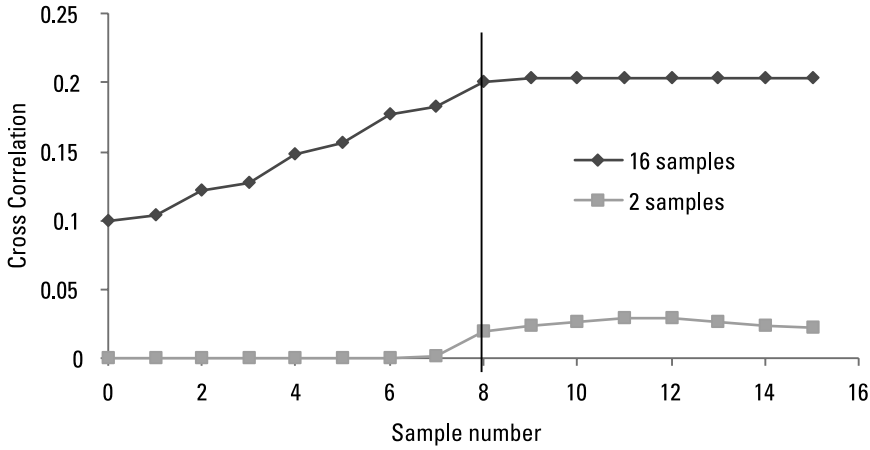


Figure 8.8 Plot of cross-correlation against the sample number at SNR = 37.8 dB.

against a sample number. Since the maximum correlation is 1 and the threshold is 0.5. Again, for $M = 16$, it is not clear where the signal starts. For $M = 2$, the correlation is greater than 0.5 for sample numbers greater than 8. Again, sample number 8 is correctly identified as the signal starting point. It is also obvious that the percentage jump at signal onset for the normalized cross-correlation is much higher than that for the cross-correlation.

Example 8.4

To see the impact of noise, the SNR is set to be 2.8 dB. Figure 8.10 is a plot of cross-correlation. The top figure uses $M = 16$. To estimate the detection

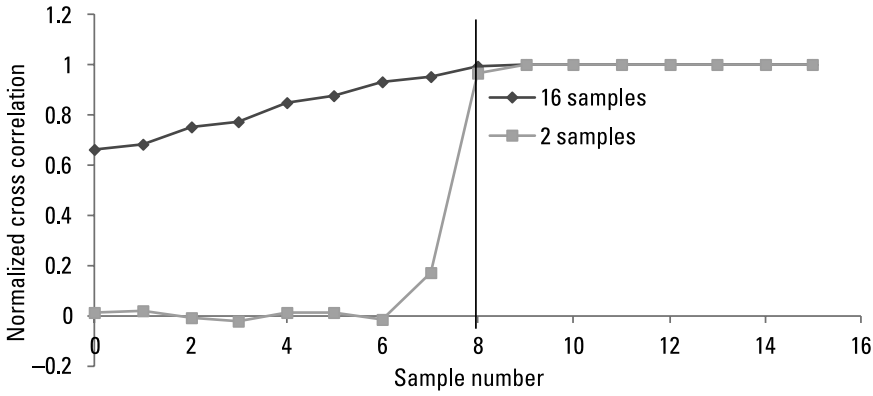


Figure 8.9 Normalized cross-correlation against the sample number at SNR = 37.8 dB.

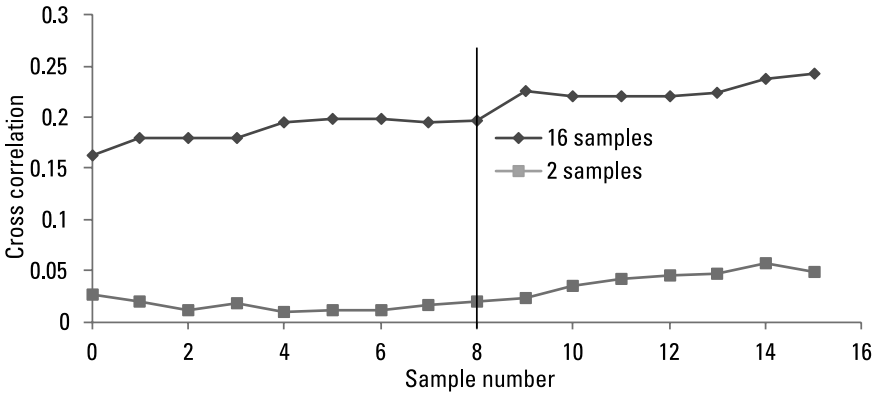


Figure 8.10 Cross-correlation against the sample number at SNR = 2.8 dB.

threshold, the sample value at m_{15} is used. The threshold is then approximately given by $m_{15}/2 = 0.24/2 = 0.12$. Figure 8.10 shows that the cross-correlation is always greater than 0.12 and it is not clear again where the signal starts. The bottom figure uses $M = 2$ and the threshold is approximately set at 0.024. The correlation is greater than the threshold at sample 0 and at sample number greater than 10. Sample 0 cannot be the signal starting point since future correlations can be smaller than the threshold. Therefore, sample 10 is selected as the signal onset and there is an error of 2 samples. Since the threshold has to be set before the detection can start, this shows the disadvantage of the cross correlation using the maximum likelihood detection strategy.

Figure 8.11 is a plot of normalized cross-correlation. The threshold is always 0.5. The top figure is for $M = 16$ and it is not clear where the signal

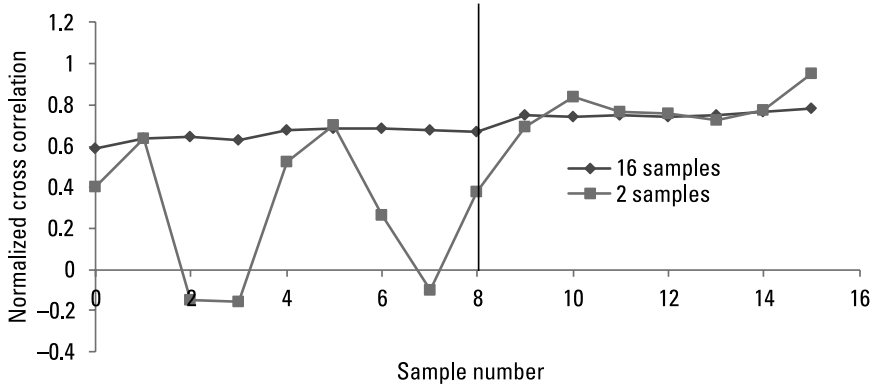


Figure 8.11 Normalized cross-correlation against the sample number at SNR = 2.8 dB.

starts. The bottom figure is for $M = 2$. The correlation is greater than 0.5 at sample 1, 5 and all samples greater than 9. Sample 1 and sample 5 cannot be the choice since future correlations can drop below 0.5. Therefore, sample 9 is the choice and there is an error of 1 sample.

8.6 Fine Timing Detection

After the starting signal sample location is roughly detected, the next step is to make some potential refinements. Since the preamble is a known signal sent from the transmitter, the receiver has a duplicate copy of this signal. For the IEEE 802.11a, the preamble consists of two parts. The first part is the short sequence, while the second part is the long sequence. Since the short sequence is used for coarse signal detection, the long sequence can be used for fine signal detection.

Assume that the clean preamble waveform is given by c_k where $k = 0$, …, $M - 1$ and M is the sample size. This clean waveform then starts at $k = 0$. After a coarse timing estimate, assume that y_L is aligned with c_0. Using the normalized cross-correlation as the metric, the following are defined:

$$E_\delta = \sum_{k=0}^{k=M-1} \left|c_k\right|^2 + \sum_{k=0}^{k=M-1} \left|y_{k+\delta+L}\right|^2 \tag{8.28a}$$

$$m_\delta(\text{normalized}) = \frac{2\sum_{k=0}^{k=M-1} \text{Re}\left(c_k y_{k+\delta+L}^*\right)}{E_\delta} \tag{8.28b}$$

where the parameter δ is the sample deviation index from $k = 0$ of the clean preamble waveform. If there is a perfect match, $\delta = 0$ and the following is true:

$$c_k = y_{k+L} \qquad k = 0,\ldots,M-1 \tag{8.29}$$

Applying (8.29) to (8.28b), we have

$$E_0 = 2\sum_{k=0}^{k=M-1} \left|c_k\right|^2 \tag{8.30}$$

$$m_0(\text{normalized}) = \frac{2\sum_{k=0}^{k=M-1} \text{Re}\left(c_k y_{k+L}^*\right)}{E_0} = 1 \tag{8.31}$$

This means the metric has the maximum value of 1 with a perfect match.

After the coarse timing estimate is performed, the value δ should not be too far away from zero. The value δ can be assumed to have the range from $-\alpha$ to α. In other words, the metric given in (8.26) is computed a total of $2\alpha + 1$ times to find the best match. The index δ that generates the maximum metric matches with clean preamble waveform.

Example 8.5

Using the IEEE 802.11a waveform, the long sequence has a total of 160 samples at a 20-MHz channel spacing. It starts with 32 guard samples followed by the first symbol of 64 samples and the second symbol of 64 samples. At SNR = 37.8 dB, the detection of short sequence finds that it starts at sample 408. The second long symbol then starts at 408 + 32 + 64 = 504. The parameter L is then set to 504. In other words, the received sample index 504 is assumed to match with sample index 0 of the clean waveform. At SNR = 2.8 dB, the short sequence is detected to start at sample index 409. In this case, sample index 505 is assumed to match with sample 0 of the clean waveform.

Figure 8.12 is a plot of the normalized cross-correlation against the sample deviation index δ. The dark curve is for SNR = 37.8 dB and the gray curve

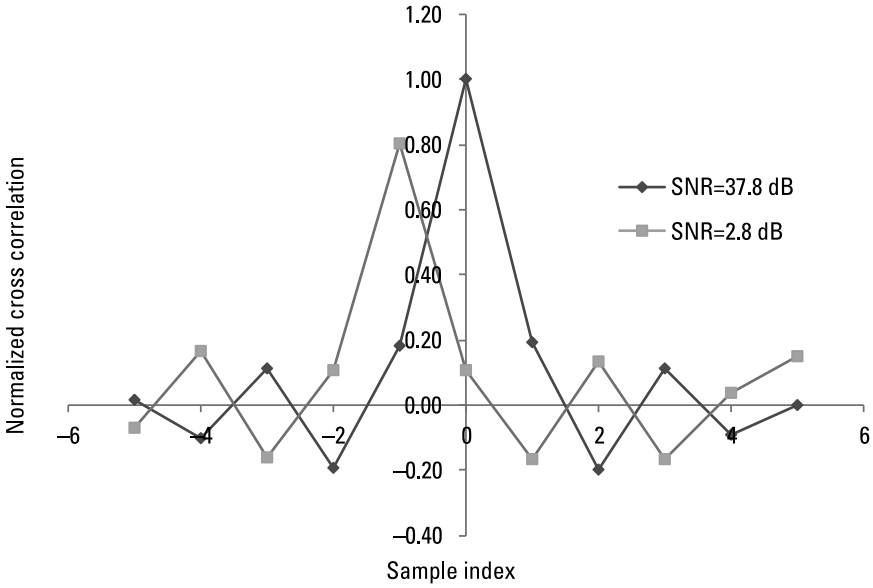


Figure 8.12 Normalized cross-correlation with clean waveform against sample deviation.

is for SNR = 2.8 dB. The parameter α is set to 5 and a total of 11 searches are performed. As expected, the maximum for SNR = 37.8 dB is 1 appearing at $\delta = 0$. The first OFDM symbol of the long sequence then starts at sample index 504. At SNR = 2.8 dB, the maximum appears at sample deviation index −1. The maximum is not 1 due to the background random noise. Since the value $\delta = 0$ corresponds to the received sample index 505, the first OFDM symbol of the long sequence then starts at received sample index 504.

Even though the coarse timing estimate may misalign the starting sample by 1 at SNR = 2.8 dB, the fine timing estimate can indeed correct the error and again make the correct adjustment.

Example 8.6

Based upon the simulation studies, the normalized cross-correlation has the best performance. This metric is then selected for further studies in various SNR levels. Figure 8.13 is a plot of the performance of coarse sample detection in random noise. The vertical axis shows the number of samples deviated from the correct signal sample onset. The horizontal axis shows the SNR in decibels. For SNR above 3.1 dB, there is no sample detection error and the sample error count is 0. For SNR between −2.08 dB and 3.01 dB, there is one sample detection error. For SNR between −4.18 dB and −2.18 dB, there

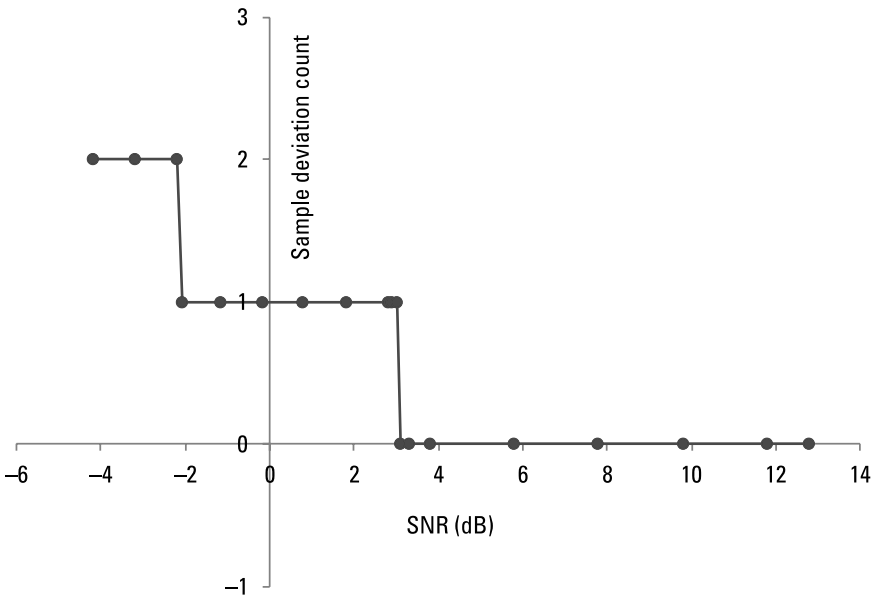


Figure 8.13 The noise performance of coarse sample detection.

are two sample detection errors. For SNR below −4.18 dB, the signal segment cannot be detected and the signal acquisition fails.

For segment detection, the segment detection error is always zero, as long as the SNR is above −4.18 dB. As is mentioned before, no signal segment can be detected for SNR below −4.18 dB. Even though there is either a one-sample or two-sample detection error from the coarse sample detection, the fine sample detection can always correct the error. That means there is no sample detection error for the fine detection as long as the SNR is above −4.18 dB.

8.7 Summary

The received radio waveform is first processed to remove the carrier. After going through a direct conversion, in-phase and quadrature-phase signals are obtained. These two signals can then be digitized for further processing.

For OFDM waveform, a periodic preamble signal is normally sent to help the receiver for signal detection. This preamble waveform used in the IEEE 802.11a is a short sequence followed by a long sequence. The short sequence has 16 periodic segments and is used for coarse timing detection. The long sequence has two periodic segments and can be used for fine timing detection.

For any detection algorithm, the detection metric must be defined. Three such metrics were discussed. They are cross-correlation, normalized cross-correlation, and MMSE. The cross-correlation and normalized cross-correlation find the correlation maximum while MMSE finds the correlation minimum. A detection algorithm based upon maximum likelihood criterion was derived. The simple algorithm shows a signal detected if the metric is greater than a threshold. For normalized cross-correlation, this threshold is simply set at 0.5 theoretically.

The detection algorithm follows a two-step procedure. The first step is coarse timing detection followed by fine timing detection. Since the preamble is periodic, we develop further a two-step approach for fast detection of a coarse timing. The first step is the segment detection in which the correlation is advanced by a segment. For the IEEE 802.11a, the short sequence has 16 samples in each segment. In other words, the correlation between two consecutive segments is 16 samples apart. The idea is to find which segment the signal starts. After segment detection, the sample detection in which the correlation is performed on a sample basis follows immediately. The idea is to locate which sample the signal starts. From the simulation study, it was found that the signal sample is difficult to locate if the correlation size is large. It was found that the best correlation size is $M = 2$.

The three metrics were compared from the simulation study. The MMSE metric was found to have the worst performance since the minimum can be indistinguishable from noise. The cross-correlation somehow works; however, the detection threshold is difficult to set and depends upon the correlation size and SNR. The normalized cross-correlation not only has the best performance, but is also the easiest metric from the computational point of view. The decision threshold from the theoretical analysis is simply 0.5 and is independent of SNR and correlation size.

After the coarse detection is completed, the fine detection follows. The fine detection computes the correlation between a clean preamble and that from the received waveform. Since the transmitted preamble is known to the receiver, this known waveform can be stored in the receiver for the fine detection. After the fine detection, the signal sample onset can be accurately determined. Since normalized correlation has the best performance, it is only used for the simulation study. Based upon the simulation in the examples, the coarse detection could accurately locate the signal sample at SNR = 37.8 dB, but with one sample error at SNR = 2.8 dB. However, the fine detection can accurately correct the error.

Lastly, the signal detection performance in random noise was investigated. As long as the SNR is above −4.18 dB, there is no sample detection error after the fine sample detection. For SNR below −4.18 dB, no signal segment can be located and the detection fails.

The next chapter centers on synchronization issues in the receiver. The topics covered include DC offset, carrier frequency offset, frame timing offset, sampling clock offset, and IQ imbalance.

References

[1] Schmidt, T., and D. Cox, "Robust Frequency and Timing Synchronization for OFDM," *IEEE Transactions on Communications*, Vol. 45, No. 12, December 1997, pp. 1631–1621.

[2] Minn, H., V. Bhargava, and K. Letaief, "A Robust Timing and Frequency Synchronization for OFDM Systems," *IEEE Transactions on Wireless Communications*, Vol. 2, No. 4, 2003, pp. 822–839.

[3] Whalen, A. D., *Detection of Signals in Noise*, San Diego: Academic Press, 1971.

9

Synchronization

9.1 Introduction

The received OFDM signal normally suffers two types of degradations. The first is due to the propagation effects from the channel and is unavoidable. The second is due to the imperfect front-end electronics that can cause frequency and phase mismatches between the transmitter and the receiver. Since the circuit parts can never completely match, some compensation schemes must be taken to minimize its impact. The synchronization deals with both issues.

Regarding the channel effect, the received OFDM signal may suffer Doppler spread. It can be caused by relative motion between the transmitter and the receiver. The received carrier frequency is then different from the transmitted frequency by the Doppler shift. The second is due to the channel variations that cause frequency dispersion or time-selective fading. The fading can be fast if the coherence time is less than the signal duration. Another distortion that the OFDM transmission suffers is frequency-selective fading. This is because signals with different delays are often received due to reflections from the ground or obstacles. This delay spread can be longer than the OFDM guard interval or the channel coherence bandwidth can be shorter than the signal bandwidth to cause serious signal distortions.

Regarding the imperfect circuit parts, the local receiver clock can run slightly differently in comparison with the transmitter clock. This can cause

some problems. Because of the different clocks, the local oscillator (LO) cannot generate the same carrier frequency and phase as the received signal. First, it causes the carrier frequency offset (CFO). Second, it causes IQ imbalance because the two LOs used to recover the in-phase and quadrature-phase components of the received signal run at a different frequency and phase. The received signals can also be sampled at different time instants in comparison with the transmitter to cause misalignments of the frame boundary. Another issue is the DC bias caused by the leakage current from the LO to the front-end linear amplifier.

All these synchronization issues including DC offset, CFO, sampling clock offset (SCO), sampling time offset, and IQ imbalance are analyzed in detail in this chapter. The parameters associated with each synchronization issue are estimated. A number of examples from computer simulations are also given to show the accuracy of the estimate.

9.2 DC Offset

After passing through the RF front end, the digitized baseband signals frequently have a DC component. The DC signal is not random and can be wrongly detected as the transmitted signal, thereby causing a false alarm. Therefore, it is important to remove the DC bias before further processing.

In an OFDM receiver, direct conversion of the RF band signals to baseband is frequently used. However, one of the drawbacks is DC bias. In this section, the DC bias is analyzed and the compensation scheme is also presented.

9.2.1 Algorithm Analysis

In a direct-conversion receiver, the converted IQ signal may have a DC offset. This phenomenon is depicted in Figure 9.1 [1, 2]. The received signal passes through a linear amplifier (LNA) and mixes with an LO. The output from

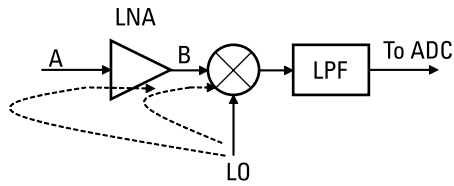


Figure 9.1 LO leakage.

the mixer is lowpass filtered and subsequently digitized using an analog-to-digital converter.

If the isolation between the LO and the mixer input or LNA input is not perfect, a leakage path may exist from LO to A (LNA input) or from LO to B (mixer input). The leakage then mixes with LO again to generate the DC offset. Assume that the LO signal is $\cos(2\pi f_c t)$ and the leakage is $\alpha \cos(2\pi f_c t)$. The mixing then produces the following:

$$\begin{aligned}\cos\left(2\pi f_c t\right)\left(\alpha\cos\left(2\pi f_c t\right)\right) &= \alpha\cos^2\left(2\pi f_c t\right) \\ &= \frac{\alpha}{2}+\frac{\alpha}{2}\cos\left(4\pi f_c t\right)\end{aligned} \tag{9.1}$$

The first term of (9.1) is the DC offset.

Besides the leakage to the LNA or mixer input, the leakage can also go to the antenna. The leakage radiated from the antenna can be reflected from the surrounding obstacles and received again by the antenna. The received reflection then mixes with the LO to generate the DC offset.

For OFDM, the data demodulation is done in the frequency domain. The DC offset only generates an extra zero frequency term in the frequency domain. Therefore, as long as the signal detection can still find the correct signal onset location, the DC offset cannot affect the accuracy of data demodulation.

On signal detection, the DC offset should have minimal impact on the MMSE metric. Assume that the DC offset is the constant c. Then the MMSE metric becomes

$$\begin{aligned}m_i &= \sum_{k=0}^{k=M-1}\left|\left(z_{i+k}+c\right)-\left(z_{i+k+M}+c\right)\right|^2 \\ &= \sum_{k=0}^{k=M-1}\left|z_{i+k}-z_{i+k+M}\right|^2\end{aligned} \tag{9.2}$$

The DC offset term is cancelled out and the metric is unchanged.

Using the metric of normalized cross correlation for signal detection, the impact can be significant. With the addition of a DC offset, the noise correlation between adjacent segments becomes highly correlated. The background noise can be mistakenly detected to be the signal to cause false alarm. From (8.17) to (8.19), we have the following by considering the case without any input signal

$$y_j = n_j + d \tag{9.3}$$

where d is the DC offset. The correlation between two segments of M samples apart becomes

$$\begin{aligned} R_i &= 2\sum_{k=0}^{k=M-1} \mathrm{Re}\left(y_{i+k} y_{i+k+M}^{*}\right) \\ &= 2\sum_{k=0}^{k=M-1} \mathrm{Re}\left(n_{i+k}+d\right)\left(n_{i+k+M}^{*}+d\right) \\ &= 2\sum_{k=0}^{k=M-1} \mathrm{Re}\left(n_{i+k} n_{i+k+M}^{*}\right) + 2d\sum_{k=0}^{k=M-1} \mathrm{Re}\left(n_{i+k}+n_{i+k+M}^{*}\right) + 2\sum_{k=0}^{k=M-1} d^2 \end{aligned} \tag{9.4}$$

Since the noise average is zero and there is no correlation between two different noise segments, the first and second terms of (9.4) are both zero. We then have the following:

$$R_i = 2Md^2 \tag{9.5}$$

Similarly, the energy from (8.17) can be shown as given here:

$$\begin{aligned} E_i &= \sum_{k=0}^{k=M-1} \left|y_{i+k}\right|^2 + \sum_{k=0}^{k=M-1} \left|y_{i+k+M}\right|^2 \\ &= 2Md^2 + 2E_N \end{aligned} \tag{9.6}$$

where E_N is the noise energy and is defined here:

$$E_N = \sum_{k=0}^{k=M-1} n_k^2 \tag{9.7}$$

The normalized correlation metric then becomes

$$m_i = \frac{R_i}{E_i} = \frac{Md^2}{Md^2 + E_N} \tag{9.8}$$

If the background noise energy is very small, then m_i is close to 1. Under this condition, the noise will be wrongly detected to be the received signal.

To avoid this problem, the DC bias needs to be removed. In the analog domain, the AC coupling can be used [2]. In the digital domain, one way is to estimate the DC bias and to remove it. However, the estimation can be

difficult if there are other channel related degradations such as multipath and fading. The easiest way is to pass the received signal through a highpass filter to remove the DC bias. The FIR filter has the advantage of linear phase. Assume that this filter has length N and coefficients, h_k, $k = 0, \ldots, N-1$. The filter output z_k due to an input y_k is then given here:

$$z_n = \sum_{k=0}^{N-1} h_k y_{n-k} \tag{9.9}$$

9.2.2 Simulation Examples

The preamble waveform from the IEEE 802.11a was used for the study. At 20 MHz, the short sequence has 10 segments of 160 samples and each segment has 16 samples. The long sequence has also 160 samples, which start with 2 guard intervals of 32 samples and are followed by two OFDM symbols of 128 samples. There are no channel degradations and other synchronization issues. However, background noise is mixed with the preamble waveform to the generated SNR at 37.8 dB and 2.8 dB. The coarse detection is done using the short sequence while the fine detection is done using the long sequence. As expected, the noise was detected as the received signal without using a highpass filter. In Example 9.1, the advantage of using a highpass filter to filter out the DC bias is illustrated.

Example 9.1

To design a highpass filter, the frequency sampling design was used [3]. This filter has 65 coefficients with $h_0 = 0.984615$ while h_1 to h_{64} all have the same value of -0.015385. The frequency response is shown in Figure 9.2 against the frequency index. A frequency index of 65 corresponds to 2π. Clearly, the frequency response is 0 at DC and 1 at all the other frequency indexes.

The computation of the output response can be simplified by utilizing the fact that all filter coefficients are the same except h_0. Assuming that $h_k = h_c$, $k = 1, \ldots, N-1$, y_k is the input, and z_k is the output, we have

$$\begin{aligned} z_n &= \sum_{k=0}^{k=N-1} h_k y_{n-k} \\ &= h_0 y_n + \left(y_{n-1} + \cdots + y_{n-N+2} + y_{n-N+1} \right) h_c \end{aligned} \tag{9.10}$$

Similarly, we have the following for z_{n+1}:

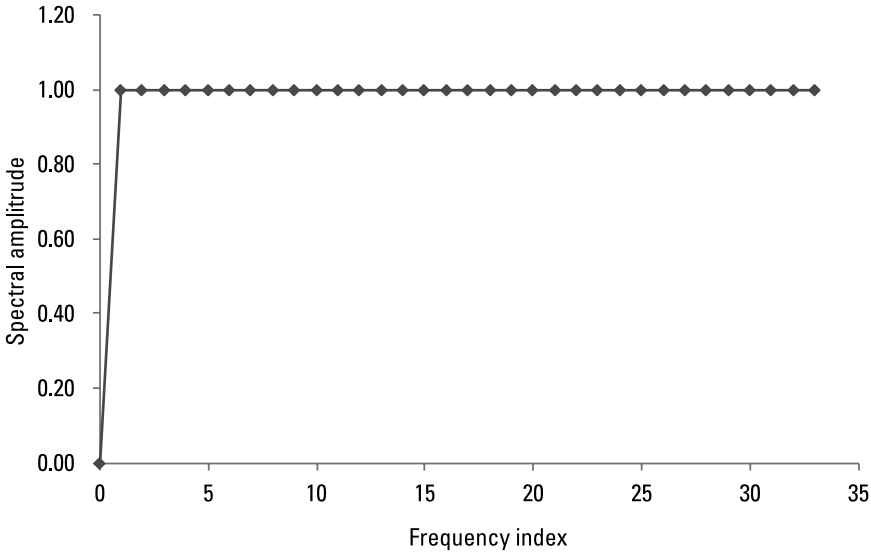


Figure 9.2 Highpass filter frequency response.

$$\begin{aligned} z_{n+1} &= h_0 y_{n+1} + \left(y_n + y_{n-1} + \ldots + y_{n-N+2}\right) h_c \\ &= h_0 y_{n+1} + y_n h_c - y_{n-N+1} h_c + \left(y_{n-1} + \ldots + y_{n-N+2} + y_{n-N+1}\right) h_c \\ &= z_n - h_0 y_n + h_0 y_{n+1} + y_n h_c - y_{n-N+1} h_c \\ &= z_n + h_0 \left(y_{n+1} - y_n\right) + h_c \left(y_n - y_{n-N+1}\right) \end{aligned} \tag{9.11}$$

Instead of having N additions and multiplications, (9.11) only has 4 additions and 2 multiplications.

Assuming that the ith preamble signal is y_i, the ith random noise sample is n_i, and the DC offset is d_i, then the output signal z_i becomes

$$z_i = y_i + n_i + d_i \tag{9.12}$$

where the random noise has zero mean. The DC offset is assumed to be a constant and has the following form:

$$d_i = c + jc \tag{9.13}$$

where c is a constant. At SNR = 37.8 dB and 2.8 dB, the DC offset d ranging from 0 to 0.059 is used. The short sequence has an average amplitude of 0.112,

Table 9.1
Deviation of Coarse and Fine Detection at SNR = 37.8 dB

Average Clean Short Sequence Amplitude	DC Offset Amplitude	DC Offset Percentage	Coarse Detection Sample Deviation	Fine Detection Sample Deviation
0.112	0.000	0%	0	0
0.112	0.020	17.9%	0	0
0.112	0.04	35.7%	0	0
0.112	0.059	52.7%	0	0

which is the square root of the average short sequence energy. It is assumed that the addition of a DC offset does not saturate the power amplifier. The average DC offset amplitude is then $\sqrt{2}c$. Tables 9.1 and 9.2 list the results at SNR = 37.8 dB and 2.8 dB, respectively. The SNR is computed without adding DC bias. In both tables, the DC offset percentage is the ratio of DC offset amplitude against the average short sequence amplitude. The coarse detection deviation is the number of the sample difference between the coarse detected sample and the actual sample onset. A plus sign means the detected position is larger than the true position, while a minus sign means the contrary. The fine detection deviation is the deviation due to the fine detection.

Both Tables 9.1 and 9.2 show that the DC bias has minimum impact on the signal detection after highpass filtering.

Table 9.2
Deviation of Coarse and Fine Detection at SNR = 2.8 dB

Average Clean Short Sequence Amplitude	DC Offset Amplitude	DC Offset Percentage	Coarse Detection Sample Deviation	Fine Detection Sample Deviation
0.112	0.000	0%	0	0
0.112	0.020	17.9%	0	0
0.112	0.040	35.7%	0	0
0.112	0.059	52.7%	0	0

9.3 CFO

In the front end of the OFDM receiver, the received carrier frequency may be different from the transmitted carrier frequency. This happens because of the relative motion between the transmitter and the receiver. The net result is a Doppler shift and was discussed in Chapter 6.

Besides the Doppler shift, there are also interferences from the neighbor subcarriers. Equation (3.14) and Figure 3.1 show such an effect. It is clear from Figure 3.1 that interferences exist in the region between peak subcarriers. However, at the peak subcarrier frequency, there are no interferences from the rest of subcarriers.

This frequency offset due to Doppler shift can degrade system performance and must be removed. The acquisition signal in OFDM normally has a repetitive structure that can be used to find the frequency offset. In this section, the estimate of frequency offset due to Doppler shift is discussed.

9.3.1 Algorithm Analysis

There are two possibilities to generate the CFO. One is the Doppler shift caused by the relative motion between the transmitter and the receiver. This happens even if there is a perfect match between the transmitter and receiver frequency. The other is that the LO frequency of the receiver may deviate from the transmitted carrier frequency.

Assuming that the frequency deviation is given by f_δ, the received carrier frequency is then $f_c + f_\delta$. The received signal $r(t)$ given in (8.9) then becomes

$$\begin{aligned} r(t) &= \mathrm{Re}\left(y(t) e^{j2\pi (f_c + f_\delta) t} \right) \\ &= \mathrm{Re}\left(z(t) e^{j2\pi f_c t} \right) \end{aligned} \tag{9.14}$$

where $z(t)$ is given by

$$z(t) = y(t) e^{j2\pi f_\delta t} \tag{9.15}$$

The baseband signal that is actually received is then $z(t)$.

Assuming that the sampling frequency is f_s and the FFT size is N, the subcarrier frequency spacing is then

$$\Delta_f = \frac{f_s}{N} \tag{9.16}$$

Assuming also that the ratio of f_δ against Δ_f is ϵ, the following then results:

$$\epsilon = \frac{Nf_\delta}{f_s} \tag{9.17}$$

After analog-to-digital conversion, the nth sample of $z(t)$ then becomes

$$z_n = y_n e^{j2\pi\epsilon n/N} \tag{9.18}$$

As shown in Section 1.8.3 of Chapter 1, a multiplication by a phase term in the time domain is equivalent to a frequency shift in the frequency domain. We have the following:

$$Z_k = Y_{k-\epsilon} \tag{9.19}$$

where Z_k is the discrete Fourier transform of z_n and Y_k is the discrete Fourier transform of y_n.

To estimate ϵ, the repetitive structure of the preamble can be utilized. The frame size M of the preamble can be set to the FFT size N. Since the period is then N, the following is true:

$$y_{n+N} = y_n \qquad n = 0,\ldots,N-1 \tag{9.20}$$

Setting n to $n + N$ in (9.18), we have

$$\begin{aligned} z_{n+N} &= y_{n+N} e^{j2\pi\epsilon(n+N)/N} \\ &= y_n e^{j2\pi\epsilon n/N} e^{j2\pi\epsilon} \\ &= z_n e^{j2\pi\epsilon} \end{aligned} \tag{9.21}$$

where we have used (9.18) and (9.20). Both z_{n+N} and z_n are the known received samples. Therefore, the CFO ϵ can be estimated based upon (9.21).

The estimate can be inaccurate in noise if just one sample is used. A better way is to form a cross-correlation between two consecutive segments. We then have the following:

$$\begin{aligned} \sum_{n=0}^{n=N-1} z_n^* z_{n+N} &= \sum_{n=0}^{n=N-1} z_n^* z_n e^{j2\pi\epsilon} \\ &= \left(\sum_{n=0}^{n=N-1} |z_n|^2 \right) e^{j2\pi\epsilon} \end{aligned} \tag{9.22}$$

From (9.22), the CFO ϵ is estimated as follows [1, 4]:

$$\epsilon = \frac{1}{2\pi}\measuredangle\left(\sum_{n=0}^{n=N-1} z_n^* z_{n+N}\right) \text{ where } \measuredangle(z) = \tan^{-1}\left(\frac{\text{Im}(z)}{\text{Re}(z)}\right) \tag{9.23}$$

In (9.23), we have assumed that the beginning sample index is 0. Also, (9.23) can be used only if the frequency offset is less than the subcarrier spacing.

In (9.23), ϵ is estimated using just one segment of the preamble. Using a concept similar to BLUE [5], a better estimate can be obtained using several such segments. For the mth and lth segment, we have from (9.18) the following:

$$z_{n+mN} = y_{n+mN} e^{j2\pi\epsilon(n+mN)/N} \qquad n = 0,\ldots,N-1 \tag{9.24}$$

$$z_{n+lN} = y_{n+lN} e^{j2\pi\epsilon(n+lN)/N} \qquad n = 0,\ldots,N-1 \tag{9.25}$$

Because of the repetitive structure, the following is true:

$$y_{n+lN} = y_{n+mN} \qquad n = 0,\ldots,N-1 \tag{9.26}$$

Using (9.26), (9.25) can then be further simplified as follows:

$$\begin{aligned} z_{n+lN} &= \left[y_{n+mN} e^{j2\pi\epsilon(n+mN)/N} \right] e^{j2\pi\epsilon(l-m)N/N} \\ &= z_{n+mN} e^{j2\pi\epsilon(l-m)} \qquad n = 0,\ldots,N-1 \end{aligned} \tag{9.27}$$

Performing the cross-correlation between the lth and mth segment, we have

$$\sum_{n=0}^{n=N-1} z_{n+mN}^* z_{n+lN} = \left[\sum_{n=0}^{n=N-1} \left| z_{n+mN} \right|^2 \right] e^{j2\pi\epsilon(l-m)} \tag{9.28}$$

where we have used (9.27). From (9.28), the frequency offset φ is estimated as follows:

$$\varphi(m,l) = \frac{1}{2\pi(l-m)}\measuredangle\left(\sum_{n=0}^{n=N-1} z_{n+mN}^* z_{n+lN}\right) \tag{9.29}$$

To reduce the number of computations, any two consecutive segments are used for averaging. In this casem $l = m + 1$ and we have

$$\varphi(m,m+1) = \frac{1}{2\pi}\measuredangle\left(\sum_{n=0}^{n=N-1} z^{*}_{n+mN} z_{n+(m+1)N}\right) \tag{9.30}$$

Assuming that there are L segments, the estimate of ϵ then becomes the following:

$$\epsilon = \frac{1}{L-1}\sum_{m=0}^{L-2}\varphi(m,m+1) \tag{9.31}$$

9.3.2 Simulation Examples

The simulation has been performed to understand the effect of SNR on the accuracy of the frequency offset estimate. Except for random noise, CFO, and DC offset, it is assumed that there are no other channel degradations and synchronization issues in the following examples. Both the short sequence and the long sequence from the IEEE 802.11a are used for simulation at 20-MHz channel spacing.

Example 9.2

In this example, the effect of SNR on the accuracy of the estimate of frequency offset is investigated. No DC offset is added; however, three different SNR levels at 37.8 dB, 7.8 dB, and 2.8 dB are used for simulation. From (9.17), the frequency offset is normalized to the subcarrier spacing and is set to 0.1. The short sequence is used for simulation and each segment has 16 samples. Equation (9.31) is used for averaging and $L = 9$. Only coarse detection is applied and there is no fine detection. Figure 9.3 is a plot of ϵ against m in (9.30). At SNR = 37.8 dB, the estimate is almost flat at $\epsilon = 0.1$. As the noise level increases, the estimate also fluctuates. The fluctuation is much higher as SNR decreases from 7.8 dB to 2.8 dB. At SNR = 37.8 dB and 7.8 dB, there is no sample misalignment. At SNR = 2.8 dB, there is one sample misalignment. Therefore, at SNR = 2.8 dB, there are two factors contributing to the estimate inaccuracy. One is the noise and the other is the sample misalignment. Even though the individual estimate for a segment may be inaccurate

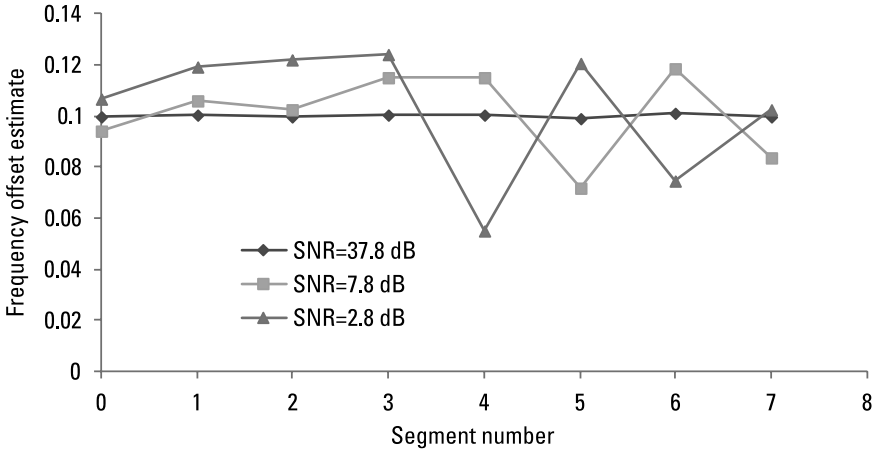


Figure 9.3 Effect of SNR on the coarse frequency offset estimate.

Table 9.3
Effect of Averaging on the Frequency Offset Estimate

	SNR = 37.8 dB	**SNR = 7.8 dB**	**SNR = 2.8 dB**
Average ϵ	0.10005	0.10089	0.1022

as SNR drops, the average can have significant improvement. After averaging by using (9.31), Table 9.3 shows the results.

It is clear from Table 9.3 that averaging does improve the estimate. Even at SNR = 2.8 dB, the estimate is close.

Example 9.3

This example is a continuation of the previous example using the short sequence. In this example, the impact of detection accuracy on the estimate of ϵ is investigated. Figure 9.4 shows the plot of ϵ against m at SNR = 2.8 dB. The gray curve uses coarse detection only, while the dark curve follows the coarse detection with another fine detection. After fine detection, it is found that there is no sample misalignment. In other words, the only factor contributing to the estimate inaccuracy is random noise after fine detection. Even though the individual estimate may still show a large swing, the average estimate improves from $\epsilon = 0.1022$ using coarse detection to $\epsilon = 0.1008$

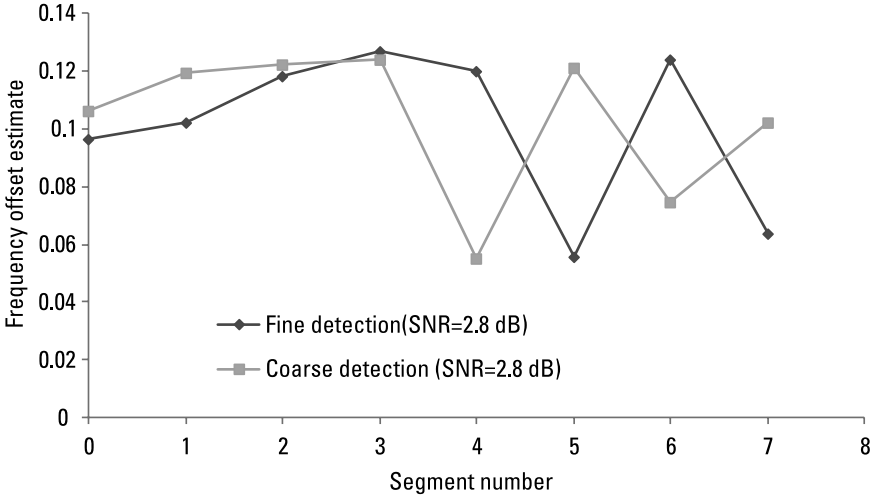


Figure 9.4 Effect of detection accuracy on the frequency offset estimate.

after fine detection. The estimate after fine detection is also the fine estimate of frequency offset.

Example 9.4

In the previous examples, the estimate is done using the short sequence. However, the long sequence can also be used for the fine estimate of frequency offset. Fine detection using the long sequence can still be performed after the coarse detection using the short sequence. Immediately after fine detection, the fine estimate of ϵ is continued. Table 9.4 shows the results.

In Table 9.4, the ϵ deviation percentage is the computation of the estimate deviation from the true value and is computed as $(\epsilon - 0.1)/0.1$. Quite clearly, the short sequence is a better choice. This is because the short sequence has 10 segments, while the long sequence has only two segments, even though

Table 9.4
Comparison of ϵ Estimate Using Short and Long Sequence

	ϵ **(SNR = 37.8 dB)**	ϵ **(SNR = 7.8 dB)**	ϵ **Deviation Percentage (SNR = 37.8 dB)**	ϵ **Deviation Percentage (SNR = 7.8 dB)**
Long	0.0998	0.0941	–0.20%	–5.90%
Short	0.100052	0.10089	0.05%	0.89%

the segment size of the long sequence (64 samples) is larger than that of the short sequence (16 samples).

Example 9.5

In this example, a range of the SNR values are scanned to find the effect of frequency offset on the detection accuracy. The SNR ranges from −3.1 dB to 37.8 dB and the frequency offset ϵ is set at 0.1. Only the short sequence is used for both the coarse detection and fine detection. Figure 9.5 is a plot of ϵ against the SNR under these two conditions. For the coarse detection, there is no sample error for the SNR above 5.8 dB, 1 sample error for the SNR from 1.81 dB to 4.8 dB, 5 sample errors for the SNR from −1.18 dB to 0.8 dB, and 6 sample errors for the SNR at −2.18 dB. In other words, for the SNR above 5.8 dB, only noise affects the estimation accuracy of ϵ. However, for the SNR below 4.8 dB, both the SNR and detection have the impacts. The sudden transition on the curve is due to the change on the number of sample errors.

The fine detection follows immediately after the coarse detection. After the fine detection, there is no sample detection error for the SNR above −1.18 dB. However, there is only one sample error for the SNR at −2.18 dB and that is why the curve has a sudden transition. For the SNR above −1.18 dB, only noise affects the estimation accuracy of ϵ.

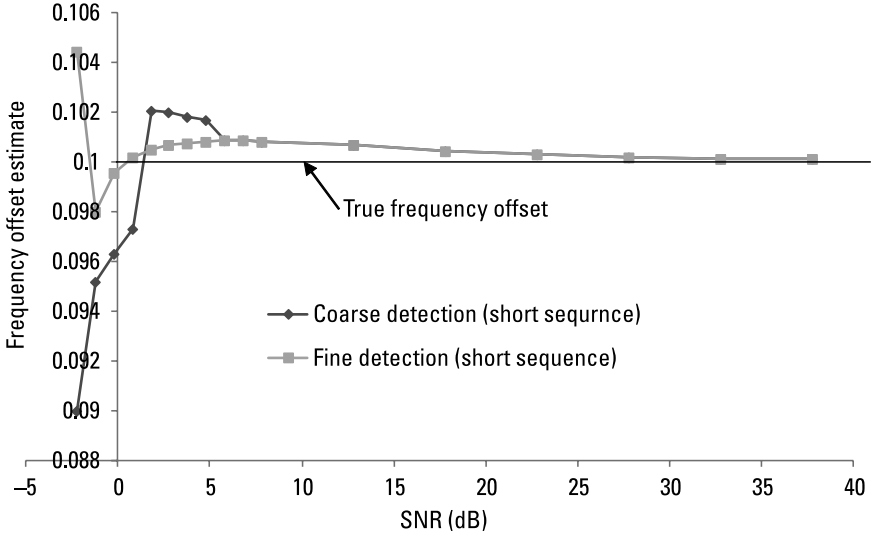


Figure 9.5 Effect of both SNR and detection accuracy on the frequency offset estimate.

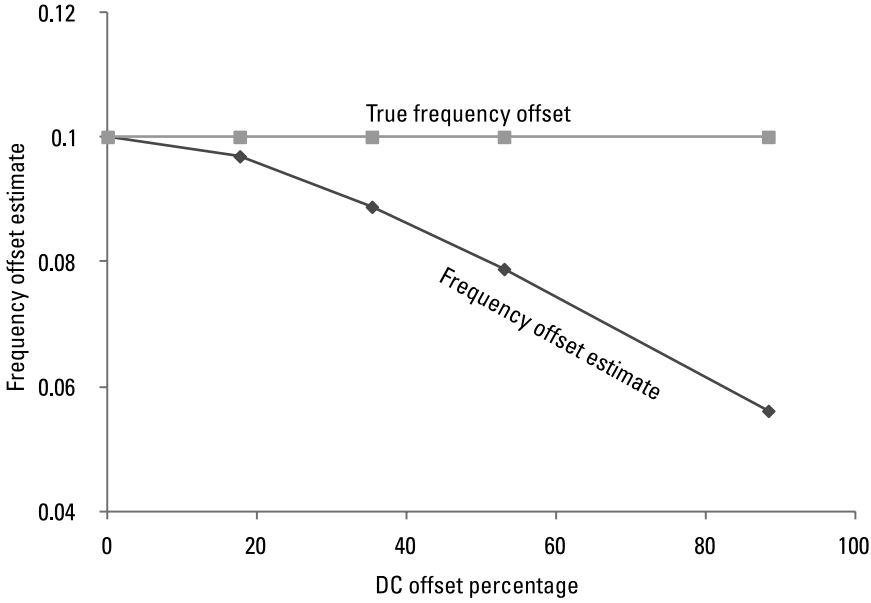


Figure 9.6 Effect of DC bias on the frequency offset estimate.

Example 9.6

In the previous examples, no DC bias is applied in the simulation. Since DC bias can also be considered as a periodic function, its presence can definitely affect the accuracy in estimating the frequency offset. To numerically show the impact of the DC bias, Figure 9.6 is the plot of two curves using the short sequence. The top curve is the true frequency offset and is the same for all of the DC bias. The bottom curve shows the estimate of the frequency offset. The SNR and frequency offset ϵ are set at 37.8 dB and 0.1, respectively. As the

Table 9.5
The Frequency Offset Estimate After Highpass Filtering

DC Offset Percentage	SNR = 37.8 dB	SNR = 2.8 dB
0	0.1001	0.1007
17.7%	0.1001	0.1007
35.3%	0.1001	0.1007
53%	0.1001	0.1007

DC offset percentage increases, the estimate accuracy also drops. This example indicates DC bias has to be removed in order to estimate the frequency offset.

Example 9.7

This example is a continuation of Example 9.6. The goal is to find the impact of applying highpass filtering. The result is shown in Table 9.5.

Quite clearly, the frequency offset can be correctly determined after highpass filtering to remove the DC offset. Since the DC offset is completely removed, it has no impact on the estimate accuracy.

9.4 Frame Timing Offset

The frame boundary of the received OFDM symbol can be different from its true location. Figure 9.7 is a timing diagram showing this situation. Between two consecutive OFDM symbols, a cyclic prefix is inserted. The OFDM symbol has N samples while the cyclic prefix has N_g samples. Assuming that the duration of the channel response lasts L samples, then $D = N_g - L$.

Figure 9.7 also shows two windows of an OFDM symbol. One window is correctly aligned, while the other window is misaligned by δ samples. As long as δ is less than D, there is no ISI [6]. The effect is just a time delay of the received waveform. Since the delay is given by δ samples, the received preamble signal, z_n, can be written as

$$z_n = y_{n-\delta} \tag{9.32}$$

Taking the discrete Fourier transform on both sides of (9.32) or from (1.47), we have

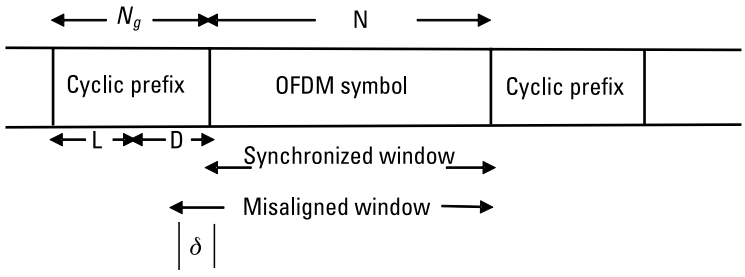


Figure 9.7 Timing diagram of frame synchronization.

$$Z_k = Y_k e^{-j2\pi k\delta/N} \tag{9.33}$$

where Z_k and Y_k are discrete Fourier transforms of z_n and y_n, respectively. Therefore, the impact of symbol timing error is equivalent to a phase shift in the frequency domain. However, if δ is within L, the ISI occurs. In this case, the received preamble suffers amplitude attenuation in addition to a phase shift [6].

Looking at this from a different angle, the timing delay can also be considered as the channel response from the multipath effect. It can be removed by first estimating the channel response. This channel equalization technique will be discussed in the next chapter.

9.5 SCO

Besides CFO and frame timing offset, the receiver may also suffer from SCO. This is because the receiver clock and the transmitter clock may not be aligned. This translates to a slightly different sampling period between the transmitter and the receiver. Assume that the transmitter sampling period is T, the receiver sampling period is T', and the deviation ratio is δ. Then T and T' are related by the following equation

$$T' = T + \delta T = T(1+\delta) \tag{9.34}$$

Suppose δ is 100 PPM or 10^{-4}; then there is one sample misalignment after 10,000 sampling periods. If the sampling clock is 20 MHz, one sample is misaligned after 0.5 ms. This shows that the SCO can be significant in the receiver performance.

9.5.1 Algorithm Development

Let $y_n$and Y_k represent the time domain and the corresponding frequency domain samples, respectively. Assuming no CFO, they are related by the following FFT for the lth symbol [7, 8]:

$$y_n = \frac{1}{N}\sum_{k=0}^{k=N-1} Y_k e^{j2\pi k[t_n-(N_g+lN_s)T]/NT} \tag{9.35}$$

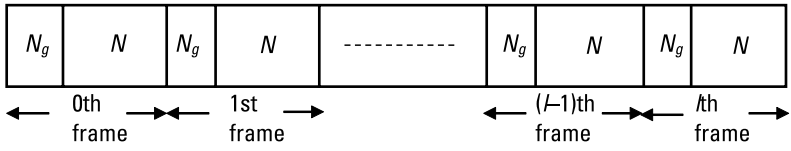


Figure 9.8 Frame and sample relationship.

where N_g is the number of guard samples and N_s is given here:

$$N_s = N_g + N \tag{9.36}$$

Figure 9.8 shows the relationship between the frames and samples. The time t_n is the time of the nth sample in the lth frame. If there is no SCO, t_n can be written as

$$t_n = \left(N_g + lN_s + n\right)T \tag{9.37}$$

Substituting (9.37) into (9.35), the following familiar inverse FFT equation results:

$$y_n = \left(\frac{1}{N}\right)\sum_{k=0}^{k=N-1} Y_k e^{j2\pi kn/N} \tag{9.38}$$

If there is a CFO of ϵ, it results in a shift of the frequency index k and (9.35) becomes

$$z_{n,l} = e^{\frac{j2\pi\epsilon t_n}{NT}}\left(\frac{1}{N}\right)\sum_{k=0}^{k=N-1} Y_{k,l} e^{j2\pi k[t_n-(N_g+lN_s)T]/NT} \tag{9.39}$$

In (9.39), we replaced y_n with $z_{n,l}$ and Y_k with $Y_{k,l}$ to show the spectral response due to the lth symbol. In the receiver, the sampling clock has a period of T' and (9.37) becomes

$$t_n = \left(N_g + lN_s + n\right)(1+\delta)T \tag{9.40}$$

Substituting (9.40) into (9.39), we have

$$z_{n,l} = e^{\frac{j2\pi\epsilon\left(N_g+lN_s+n\right)(1+\delta)}{N}}\left(\frac{1}{N}\right)\sum_{k=0}^{k=N-1} Y_{k,l}e^{j2\pi k[\delta(N_g+lN_s)+n(1+\delta)]/N} \tag{9.41}$$

Taking the FFT of $z_{n,l}$ generates the frequency response of $Z_{k,l}$

$$Z_{k,l} = \sum_{n=0}^{n=N-1} z_{n,l}e^{-j2\pi kn/N} \tag{9.42}$$

Substituting (9.41) into (9.42),

$$\begin{aligned} Z_{k,l} = \frac{1}{N}\sum_{q=0}^{q=N-1} Y_{q,l}e^{[j2\pi\epsilon(N_g+lN_s)(1+\delta)]/N}e^{j2\pi q(N_g+lN_s)\delta/N} \\ \sum_{n=0}^{n=N-1} e^{[j2\pi n\left(q-k+q\delta+(1+\delta)\epsilon\right)]/N} \end{aligned} \tag{9.43}$$

Equation (9.43) can be rewritten in the following form:

$$Z_{k,l} = \frac{1}{N}Y_{k,l}e^{[j2\pi\epsilon(N_g+lN_s)(1+\delta)]/N}e^{[j2\pi k(N_g+lN_s)\delta]/N}\sum_{n=0}^{n=N-1} e^{[j2\pi n\left(k\delta+(1+\delta)\epsilon\right)]/N} + \text{ICI} \tag{9.44}$$

where intercarrier interference (ICI) is given by

$$\begin{aligned} \text{ICI} = \frac{1}{N}\sum_{q=0,q\neq k}^{q=N-1} Y_{q,l}e^{[j2\pi\epsilon(N_g+lN_s)(1+\delta)]/N}e^{[j2\pi q(N_g+lN_s)\delta]/N} \\ \sum_{n=0}^{n=N-1} e^{[j2\pi n\left(q-k+q\delta+(1+\delta)\epsilon\right)]/N} \end{aligned} \tag{9.45}$$

The phase term inside the summation of (9.44) is very small and the following approximation is made:

$$e^{j2\pi n\left(k\delta+(1+\delta)\epsilon\right)/N} \approx 1 \tag{9.46}$$

Substituting (9.46) into (9.44) and neglecting ICI, we have

$$Z_{k,l} = Y_{k,l} e^{[j2\pi\epsilon(N_g + lN_s)(1+\delta)]/N} e^{[j2\pi k(N_g + lN_s)\delta]/N} \tag{9.47}$$

Assume the phase of $Z_{k,l}$ is θ and the phase of $Y_{k,l}$ is φ, the following is then obtained:

$$\theta_{k,l} = \varphi_{k,l} + \left(\frac{1}{N}\right)2\pi\epsilon\left(N_g + lN_s\right)(1+\delta) + \left(\frac{1}{N}\right)2\pi k\left(N_g + lN_s\right)\delta \tag{9.48}$$

For the symbol $l + 1$, (9.48) then becomes

$$\begin{aligned}\theta_{k,l+1} = \varphi_{k,l+1} &+ \left(\frac{1}{N}\right)2\pi\epsilon\left(N_g + lN_s + N_s\right)(1+\delta) \\ &+ \left(\frac{1}{N}\right)2\pi k\left(N_g + lN_s + N_s\right)\delta\end{aligned} \tag{9.49}$$

Subtracting (9.48) from (9.49), we have

$$\theta_{k,l+1} - \theta_{k,l} = \varphi_{k,l+1} - \varphi_{k,l} + \frac{2\pi\epsilon N_s(1+\delta)}{N} + \frac{2\pi k\delta N_s}{N} \tag{9.50}$$

Since the product of ϵ and δ is small, (9.50) can be further simplified to be

$$\theta_{k,l+1} - \theta_{k,l} = \varphi_{k,l+1} - \varphi_{k,l} + \frac{2\pi\epsilon N_s}{N} + \frac{2\pi k\delta N_s}{N} \tag{9.51}$$

The phase difference between two consecutive symbols can be given as below:

$$\Delta\theta_k = \theta_{k,l+1} - \theta_{k,l} \tag{9.52}$$

$$\Delta\varphi_k = \varphi_{k,l+1} - \varphi_{k,l} \tag{9.53}$$

Using (9.52) and (9.53), (9.51) then becomes

$$\Delta\theta_k = \Delta\varphi_k + \frac{2\pi\epsilon N_s}{N} + \frac{2\pi k\delta N_s}{N} \tag{9.54}$$

Table 9.6
Pilot Phases of the First 5 OFDM Symbols from IEEE 802.11a

Symbol Number	$k=-21$	$k=-7$	$k=7$	$k=21$
1	1	1	1	–1
2	1	1	1	–1
3	1	1	1	–1
4	–1	–1	–1	1
5	–1	–1	–1	1

For the CFO, the phase shift is the same for all of the subcarriers. However, for SCO, the phase increases linearly with respect to the subcarrier frequency index.

9.5.2 Lease Square Estimation in the Tracking Mode

The OFDM symbols are transmitted during the tracking mode. Usually in the frequency domain, there are a set of pilot subcarriers. The phases of these pilot subcarriers are known. At a given frequency index, two consecutive symbols may not have the same phase. For the IEEE 802.11a, each OFDM symbol has four pilot subcarriers at frequency indexes of –21, –7, 7, and 21. Table 9.6 shows the phases of these four pilot subcarriers in the first five data symbols. From symbol 3 to symbol 4, there is a sign change. This means $\Delta\varphi_k$ is not necessarily 0, but it is a known quantity. Moving $\Delta\varphi_k$ to the left in (9.54), we have

$$\Delta\varnothing_k = \frac{2\pi\epsilon N_s}{N} + \frac{2\pi k\delta N_s}{N} \tag{9.55}$$

where $\Delta\varnothing_k$ is defined here:

$$\Delta\varnothing_k = \Delta\theta_k - \Delta\varphi_k \tag{9.56}$$

Before trying to solve (9.55), it is rewritten as here:

$$v_{k=}\frac{N}{2\pi N_s}\Delta\varnothing_k = \delta k + \epsilon \tag{9.57}$$

Since pilot locations are not continuous, let k_j ($j = 1, M$) represent the pilot locations. For the IEEE 802.11a, k_j takes the four possible values –21,

–7, 7, and 21. Equation (9.57) has only two unknowns, but M data points and it can be solved using the method of least squares.

For each data point, the error is given here:

$$\varepsilon_j = v_{k_j} - \delta k_j - \epsilon \qquad j = 1, M \tag{9.58}$$

Summing up the square of all errors, we have

$$E = \sum_{j=1}^{M} \varepsilon_j^2 = \sum_{j=1}^{M} \left(v_{k_j} - \delta k_j - \epsilon \right)^2 \tag{9.59}$$

To find the minimum E, the two derivatives of E against ϵ and δ are set to 0. We then have

$$M\epsilon + \left(\sum_{j=1}^{M} k_j \right) \delta = \sum_{j=1}^{M} v_{k_j} \tag{9.60}$$

$$\left(\sum_{j=1}^{M} k_j \right) \epsilon + \left(\sum_{j=1}^{M} k_j^2 \right) \delta = \sum_{j=1}^{M} v_{k_j} k_j \tag{9.61}$$

Equations (9.60) and (9.61) have two unknowns and can be solved to give

$$\epsilon = \frac{\left(\sum_{j=1}^{M} v_{k_j} \right) \left(\sum_{j=1}^{M} k_j^2 \right) - \left(\sum_{j=1}^{M} k_j \right) \left(\sum_{j=1}^{M} v_{k_j} k_j \right)}{J} \tag{9.62}$$

$$\delta = \frac{\mathrm{M}\left(\sum_{j=1}^{M} v_{k_j} k_j \right) - \left(\sum_{j=1}^{M} v_{k_j} \right) \left(\sum_{j=1}^{M} k_j \right)}{J} \tag{9.63}$$

where J is given by

$$J = M \left(\sum_{j=1}^{M} k_j^2 \right) - \left(\sum_{j=1}^{M} k_j \right)^2 \tag{9.64}$$

Substituting $v_{k_j} = (N/2\pi_{\mathrm{Ns}})\Delta\varnothing_{k_j}$ into (9.62) and (9.63), we have

$$\epsilon = \left(\frac{N}{2\pi N_s}\right)\frac{\left(\sum_{j=1}^{M}\Delta\varnothing_{k_j}\right)\left(\sum_{j=1}^{M}k_j^2\right)-\left(\sum_{j=1}^{M}k_j\right)\left(\sum_{j=1}^{M}\Delta\varnothing_{k_j}k_j\right)}{J} \tag{9.65}$$

$$\delta = \left(\frac{N}{2\pi N_s}\right)\frac{\mathrm{M}\left(\sum_{j=1}^{M}\Delta\varnothing_{k_j}k_j\right)-\left(\sum_{j=1}^{M}\Delta\varnothing_{k_j}\right)\left(\sum_{j=1}^{M}k_j\right)}{J} \tag{9.66}$$

The definition of N_s and $\Delta\varnothing_{k_j}$ are rewritten here for quick reference

$$\Delta\varnothing_{k_j} = \Delta\theta_{k_j} - \Delta\varphi_{k_j} \qquad j = 1, M \tag{9.67}$$

$$N_s = N_g + N \tag{9.68}$$

Equations from (9.65) to (9.66) are then joint estimations of both CFO and SCO in the tracking mode using the received data symbols.

9.5.3 Estimation in the Acquisition Mode

Previously, pilot subcarriers in the received data symbols were used to derive a joint estimate of the CFO and SCO. Similar formulas can also be used for their estimates in the acquisition mode. In the acquisition mode, the repetitive acquisition sequence can be used.

Assume that there are L such repetitive segments. For the short sequence in the IEEE 802.11a, there are $L = 10$ segments and each segment has 16 samples. Each segment repeats itself and can be considered similar to one OFDM symbol. There are two major differences. The first is that there is no guard interval and the second is a different signal transmission format.

Equations (9.65) and (9.66) can still be used with two modifications. The first is $N_g = 0$ and $N_s = N$. For the IEEE 802.11a, each segment has $M = 12$ subcarriers and each subcarrier can be considered as pilot subcarriers. Since the segment repeats, two consecutive segments have the same phases at each subcarrier. Therefore, the second modification is $\Delta\varphi_k = 0$. Incorporating these two modifications, the following equations result:

$$\epsilon = \left(\frac{1}{2\pi}\right)\frac{\left(\sum_{j=1}^{M}\Delta\varnothing_{k_j}\right)\left(\sum_{j=1}^{M}k_j^2\right)-\left(\sum_{j=1}^{M}k_j\right)\left(\sum_{j=1}^{M}\Delta\varnothing_{k_j}k_j\right)}{J} \tag{9.69}$$

$$\delta = \left(\frac{1}{2\pi}\right)\frac{\mathrm{M}\left(\sum_{j=1}^{M}\Delta\varnothing_{k_j}k_j\right)-\left(\sum_{j=1}^{M}\Delta\varnothing_{k_j}\right)\left(\sum_{j=1}^{M}k_j\right)}{J} \tag{9.70}$$

$$\Delta\varnothing_{k_j} = \Delta\theta_{k_j} \qquad j = 1, M \tag{9.71}$$

$$N_s = N \tag{9.72}$$

$$J = M\left(\sum_{j=1}^{M}k_j^2\right)-\left(\sum_{j=1}^{M}k_j\right)^2 \tag{9.73}$$

As we have shown previously, the CFO can be estimated independently. Assuming that the CFO is known already, then we have from (9.57):

$$\frac{1}{2\pi}\Delta\theta_{k_j} - \epsilon = \delta k_j \tag{9.74}$$

Summing across all subcarriers, (9.74) becomes

$$\delta\left(\sum_{j=1}^{M}k_j\right) = \frac{1}{2\pi}\left(\sum_{j=1}^{M}\Delta\theta_{k_j}\right) - M\epsilon \tag{9.75}$$

Solving for δ in (9.75), we obtain

$$\delta = \frac{\frac{1}{2\pi}\left(\sum_{j=1}^{M}\Delta\theta_{k_j}\right) - M\epsilon}{\left(\sum_{j=1}^{M}k_j\right)} \tag{9.76}$$

For the long sequence used in the IEEE 802.11a, there are two consecutive symbols with no guard intervals in between. These two symbols are identical to each other. Therefore, the same equations applied to the short sequence are also valid. The only differences are the parameter values, which are L and M. The parameter L is 2 because there are only two symbols. The number of subcarriers in each symbol including DC is 53. Excluding DC, then $M = 52$.

9.5.4 Estimation Under Special Conditions

More insight to the estimation can be obtained under two special conditions. The first is when there is no SCO and the second is when there is no CFO. Under both cases, the estimation is significantly simplified.

Assume that first there is no SCO and $\delta = 0$. Setting $\delta = 0$ in (9.66), we have

$$\left(\sum_{j=1}^{M} \Delta\varnothing_{k_j} k_j\right) = \frac{\left(\sum_{j=1}^{M} \Delta\varnothing_{k_j}\right)\left(\sum_{j=1}^{M} k_j\right)}{M} \tag{9.77}$$

Substituting (9.77) into (9.65), the CFO becomes

$$\epsilon = \frac{\left(\frac{N}{2\pi N_s}\right)\left(\sum_{j=1}^{M} \Delta\varnothing_{k_j}\right)}{M} \tag{9.78}$$

$$\Delta\varnothing_{k_j} = \Delta\theta_{k_j} - \Delta\varphi_{k_j} \qquad j = 1,\ldots,M \tag{9.79}$$

$$N_s = N_g + N \tag{9.80}$$

Equation (9.78) is the frequency-domain estimate of CFO in the tracking mode. At any given subcarrier pilot index k_j, it depends upon the phase difference between two consecutive OFDM symbols. It does not depend upon a particular subcarrier pilot index. The average across all possible pilot subcarrier indexes minimizes the error.

In the acquisition mode, $N_s = N$ and $\Delta\varphi_j = 0$ and the following equation results:

$$\epsilon = \frac{\left(\sum_{j=1}^{M} \Delta\theta_{k_j}\right)}{(2\pi M)} \tag{9.81}$$

In this case, $\Delta\theta_{k_j}$ is just the phase difference between two consecutive segments in the repetitive acquisition sequence for a given subcarrier index k_j. It is very similar to the frequency-domain estimate given in [4]. The average across all possible subcarrier indexes improves the estimate.

Assume next that there is no CFO. By setting $\epsilon = 0$ in (9.65), we then have

$$\sum_{j=1}^{M} \Delta\varnothing_{k_j} k_j = \frac{\left(\sum_{j=1}^{M} \Delta\varnothing_{k_j}\right)\left(\sum_{j=1}^{M} k_j^2\right)}{\left(\sum_{j=1}^{M} k_j\right)} \tag{9.82}$$

Substituting (9.82) into (9.66), the following equation is obtained:

$$\delta = \left(\frac{N}{2\pi N_s}\right)\frac{\left(\sum_{j=1}^{M} \Delta\varnothing_{k_j}\right)}{\left(\sum_{j=1}^{M} k_j\right)} \tag{9.83}$$

Equation (9.82) can also be written in the following form:

$$\sum_{j=1}^{M} \Delta\varnothing_{k_j} = \frac{\left(\sum_{j=1}^{M} \Delta\varnothing_{k_j} k_j\right)\left(\sum_{j=1}^{M} k_j\right)}{\left(\sum_{j=1}^{M} k_j^2\right)} \tag{9.84}$$

Substituting (9.84) into (9.66), δ then becomes

$$\delta = \left(\frac{N}{2\pi N_s}\right)\frac{\left(\sum_{j=1}^{M} \Delta\varnothing_{k_j} k_j\right)}{\left(\sum_{j=1}^{M} k_j^2\right)} \tag{9.85}$$

Both (9.78) and (9.85) are similar to those given in [1].

Equation (9.83) is the frequency-domain estimate of δ in the tracking mode. Another direct way of getting this formula is by setting $\epsilon = 0$ in (9.57). It is seen from there that the phase difference between two consecutive OFDM symbols is linearly proportional to the pilot subcarrier index. Equation (9.83) just finds the average slope to improve the estimate.

Again, in the acquisition mode, $N_s = N$ and $\Delta\varphi_{k_j} = 0$. Equations (9.83) and (9.85) are further simplified to give the following

$$\delta = \left(\frac{1}{2\pi}\right)\frac{\left(\sum_{j=1}^{M} \Delta\theta_{k_j}\right)}{\left(\sum_{j=1}^{M} k_j\right)} \tag{9.86}$$

or

$$\delta = \left(\frac{1}{2\pi}\right)\frac{\left(\sum_{j=1}^{M}\Delta\theta_{k_j}k_j\right)}{\left(\sum_{j=1}^{M}k_j^2\right)} \tag{9.87}$$

If it is known in advance that either CFO = 0 or SCO = 0, the estimate can be greatly simplified. If there is no such knowledge, then both assumptions are not true and the original equations given earlier should be used.

9.5.5 Simulation Examples

The short sequence defined in the IEEE 802.11a is used for the simulation. It is assumed that there are no channel degradations and IQ imbalances. However, there is possible CFO, SCO, and background noise.

Example 9.8

Assume that the CFO and SCO are $\epsilon = 0.05$ and $\delta = 0.001$, respectively. The joint estimate is applied at SNR = 37.8 dB. Assume also that there is no DC bias. Because of the SCO, the received samples are actually sampled at $nT + n\delta T$ where n is an integer. These time-shifted samples are obtained through interpolation. The DFT is then applied to find the phase at each subcarrier

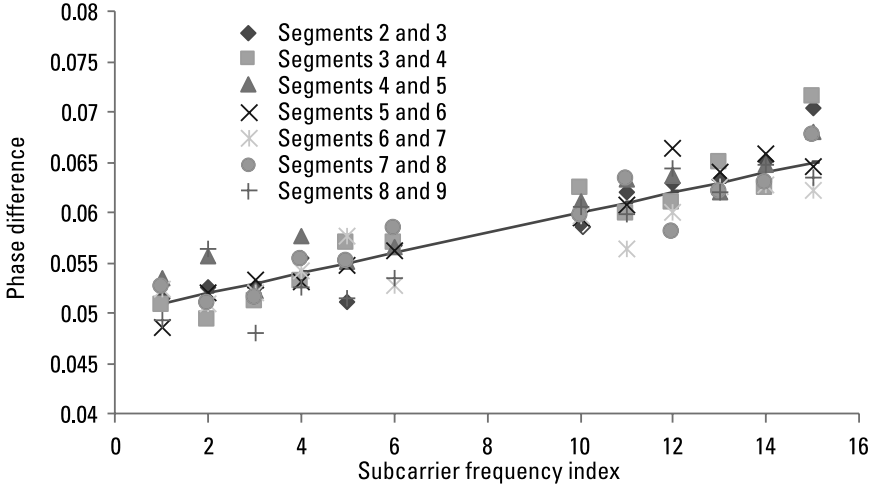


Figure 9.9 A plot of phase difference against subcarrier index.

index k_j. The phase difference given by (9.52) is subsequently evaluated between two consecutive segments l and $l+1$ where l ranges from 2 to 8. For a short sequence, there are 12 subcarriers from 1 to 6 and from 10 to 15. Figure 9.9 is a scatter plot of phase difference against subcarrier index. Also superimposed on the figure is an ideal curve. From (9.74), it is given as

$$y = 0.001x + 0.05$$

where $y = (1/2\pi)\Delta\theta_{k_j}$ and $x = k_j$. It can be seen that the data points are scattered along the straight line of this ideal curve. This also means an average from all two consecutive segments shall improve the estimate.

Example 9.9

Under the same conditions given in Example 9.8, (9.69) and (9.70) are then applied to compute the CFO and SCO for the SNR from 7.8 dB to 37.8 dB. First, they are computed for each two consecutive segments l and $l+1$ where l ranges from 2 to 8. A total of seven such values are generated. The average CFO and SCO are subsequently computed and plotted in Figures 9.10 and 9.11. Both Figures 9.10 and 9.11 show that the estimate converges to the desired value as the SNR increases. In Example 9.2, we showed the estimate of the CFO without the SCO through time-domain analysis. It is clear that the

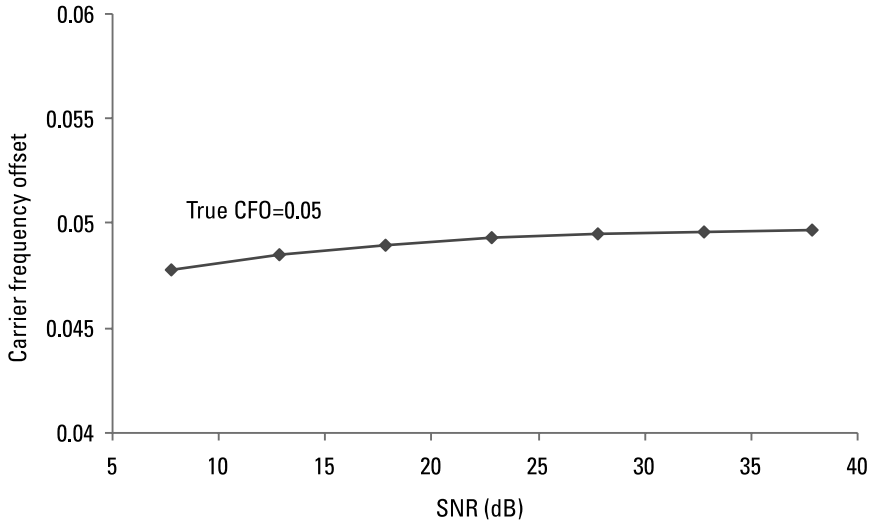


Figure 9.10 A plot of the CFO against the SNR.

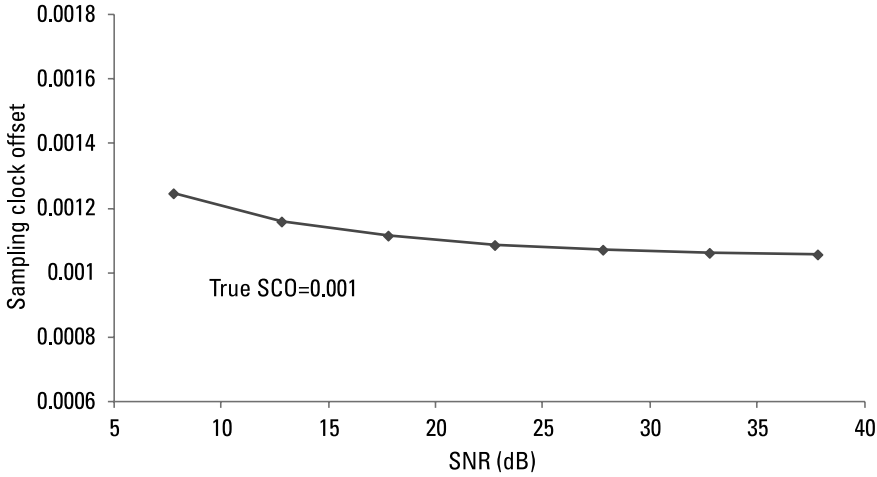


Figure 9.11 A plot of the SCO against the SNR.

estimate accuracy of the CFO goes down if there is an SCO. This is because there are approximations involved during the development of the joint estimate.

Example 9.10

Assume that there is no CFO. Then both (9.86) and (9.87) can be used to compute the SCO. Figure 9.12 is a plot of both against the SNR. The dark

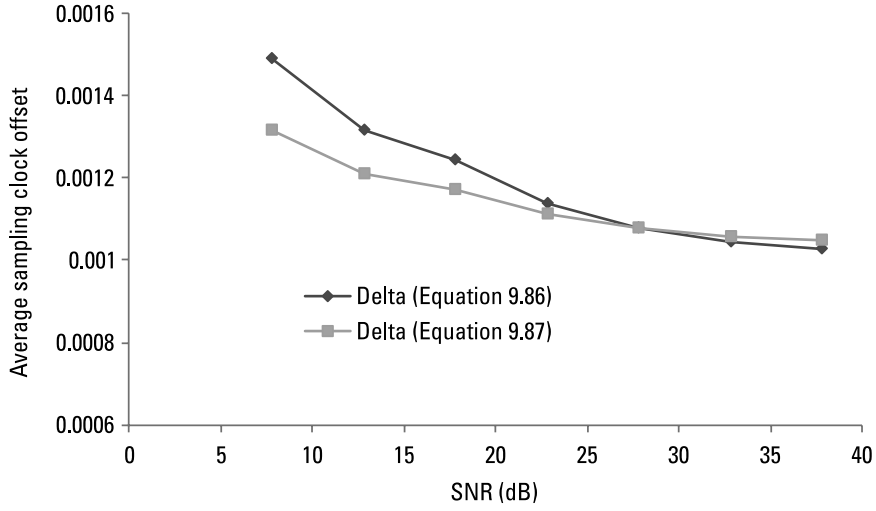


Figure 9.12 Comparisons of the SCO against the SNR.

curve is a plot using (9.86) while the gray curve is a plot using (9.87). All the other conditions are the same as in previous examples. As can be seen from the plot, the difference from both computations is small. However, the computation from (9.87) is slightly better.

9.6 IQ Imbalance

As was shown in Section 8.2, the in-phase and quadrature-phase components need to be extracted by mixing the received signal with two locally generated sinusoidal signals. Under ideal conditions, these two signals differ in phase by 90°. In reality, the circuit parts may not be perfect and there are phase and amplitude disturbances. The IQ imbalance [9–11] refers to the mismatch of these two LOs.

Because of the mismatch, the channel estimate is no longer accurate and the system performances can be affected significantly. Using the training samples in the acquisition mode, the compensation scheme is discussed in the following sections.

9.6.1 IQ Model

Assume the received complex amplitude to be $y(t)$, the transmitted complex amplitude to be $x(t)$ and the channel response to be $h(t)$, then the following equations result from (8.7) to (8.9):

$$y(t) = y_I(t) + jy_Q(t) \tag{9.88}$$

$$x(t) = x_I(t) + jx_Q(t) \tag{9.89}$$

$$\begin{aligned} r(t) &= \mathrm{Re}\left(y(t)e^{j2\pi f_c t}\right) \\ &= y_I(t)\cos\left(2\pi f_c t\right) - y_Q(t)\sin\left(2\pi f_c t\right) \end{aligned} \tag{9.90}$$

$$y_I(t) = \int x_I(\tau)h(t-\tau)\,d\tau \tag{9.91}$$

$$y_Q(t) = \int x_Q(\tau)h(t-\tau)\,d\tau \tag{9.92}$$

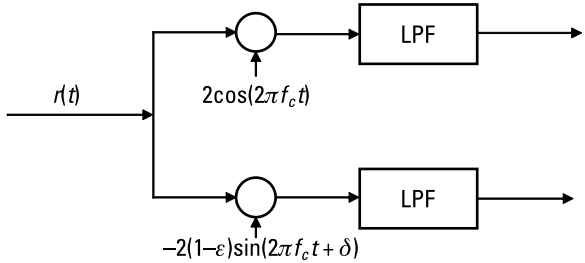


Figure 9.13 Direct conversion receiver with IQ imbalance.

where $r(t)$ is the received signal and f_c is the carrier frequency

Figure 9.13 shows a direct conversion receiver in which the two LOs are not a perfect match in both amplitude and phase. Assume that the phase deviation is given by δ and the amplitude deviation is given by ε. The two LOs can be written as

$$l_1(t) = 2\cos\left(2\pi f_c t\right) \tag{9.93}$$

$$l_2(t) = -2(1-\varepsilon)\sin\left(2\pi f_c t + \delta\right) \tag{9.94}$$

$$= -2\alpha\sin\left(2\pi f_c t + \delta\right) \tag{9.95}$$

where we have defined $\alpha = 1 - \varepsilon$. If $\delta = 0$ and $\varepsilon = 0$, then the two LOs are perfectly matched and Figure 9.13 reduces to Figure 8.1.

Multiplying (9.90) by (9.93) and following the same analysis as given in Chapter 8, we recover $y_I(t)$ and

$$r_I(t) = y_I(t) \tag{9.96}$$

Similarly, multiplying (9.90) by (9.95), we have $r_Q(t)$ in the lower branch as given here:

$$\begin{aligned} r_Q(t) &= -2\alpha\Big(y_I(t)\sin\left(2\pi f_c t + \delta\right)\cos\left(2\pi f_c t\right) - y_Q(t)\sin\left(2\pi f_c t\right)\sin\left(2\pi f_c(t) + \delta\right)\Big) \\ &= \alpha\Big(-y_I(t)\sin\left(4\pi f_c t + \delta\right) - y_I(t)\sin(\delta) - y_Q(t)\cos\left(4\pi f_c(t) + \delta\right) + y_Q(t)\cos(\delta)\Big) \end{aligned} \tag{9.97}$$

Again, $r_Q(t)$ in (9.97) after passing through an LPF becomes

$$r_Q(t) = \alpha y_Q(t)\cos(\delta) - \alpha y_I(t)\sin(\delta) \tag{9.98}$$

If both ε and δ are 0 and $\alpha = 1$, then $r_Q(t)$ becomes $y_Q(t)$.

9.6.2 Estimation in the Acquisition Mode

Equation (9.96) and (9.98) can also be written in the following matrix form [11]:

$$\begin{bmatrix} r_I(t) \\ r_Q(t) \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ -\alpha\sin(\delta) & \alpha\cos(\delta) \end{bmatrix} \begin{bmatrix} y_I(t) \\ y_Q(t) \end{bmatrix} \tag{9.99}$$

After the analog-to-digital conversion, the ith digitized samples are written as

$$r_{I,i} = y_{I,i} \tag{9.100}$$

$$r_{Q,i} = -\alpha\sin(\delta)y_{I,i} + \alpha\cos(\delta)y_{Q,i} \tag{9.101}$$

The amplitude imbalance α is estimated by first taking the square on both sides of (9.101) and summing across all the acquisition samples. Assuming that there are L such samples, we have

$$\begin{aligned} \sum_{i=0}^{L-1} r_{Q,i}^2 = {} & \alpha^2\sin^2(\delta)\sum_{i=0}^{i=L-1} y_{I,i}^2 - 2\alpha^2\sin(\delta)\cos(\delta)\sum_{i=0}^{i=L-1} y_{I,i}y_{Q,i} \\ & + \alpha^2\cos^2(\delta)\sum_{i=0}^{L-1} y_{Q,i}^2 \end{aligned} \tag{9.102}$$

During the acquisition mode, the training samples can be designed in such a way that the following relationship is satisfied:

$$\sum_{i=0}^{i=L-1} y_{I,i}^2 = \sum_{i=0}^{L-1} y_{Q,i}^2 \tag{9.103}$$

Similarly, $y_{I,i}$ can be uncorrelated with $y_{Q,i}$ and the following relationship can be made:

$$\sum_{i=0}^{i=L-1} y_{I,i} y_{Q,i} = 0 \tag{9.104}$$

Both (9.103) and (9.104) will be justified in Example 9.11 using the short sequence of the IEEE 802.11a.

Substituting (9.103) and (9.104) into (9.102), we have

$$\begin{aligned} \sum_{i=0}^{L-1} r_{Q,i}^2 &= \alpha^2 \left(\sin^2(\delta) + \cos^2(\delta) \right) \sum_{i=0}^{i=L-1} y_{I,i}^2 \\ &= \alpha^2 \sum_{i=0}^{i=L-1} y_{I,i}^2 \end{aligned} \tag{9.105}$$

Summing across all the L samples in (9.100), we have

$$\sum_{i=0}^{L-1} r_{I,i}^2 = \sum_{i=0}^{i=L-1} y_{I,i}^2 \tag{9.106}$$

Dividing (9.105) by (9.106), we have an estimate for the amplitude imbalance α:

$$\alpha = \sqrt{\frac{\sum_{i=0}^{L-1} r_{Q,i}^2}{\sum_{i=0}^{L-1} r_{I,i}^2}} \tag{9.107}$$

From the definition of α, the amplitude deviation ε is then given by

$$\varepsilon = 1 - \alpha \tag{9.108}$$

$$= 1 - \sqrt{\frac{\sum_{i=0}^{L-1} r_{Q,i}^2}{\sum_{i=0}^{L-1} r_{I,i}^2}} \tag{9.109}$$

To estimate the phase imbalance δ, the cross-correlation of $r_{I,i}$ and $r_{Q,i}$ are formed. From (9.100) and (9.101), the following is obtained:

$$\begin{aligned}\sum_{i=0}^{L-1} r_{I,i} r_{Q,i} &= -\alpha \sin(\delta) \sum_{i=0}^{i=L-1} y_{I,i}^2 + \alpha \cos(\delta) \sum_{i=0}^{i=L-1} y_{I,i} y_{Q,i} \\ &= -\alpha \sin(\delta) \sum_{i=0}^{i=L-1} y_{I,i}^2\end{aligned} \tag{9.110}$$

where we have used the same approximation given by (9.104). From (9.110) and (9.100), δ is solved to give

$$\sin(\delta) = -\frac{\sum_{i=0}^{L-1} r_{I,i} r_{Q,i}}{\alpha \sum_{i=0}^{i=L-1} r_{I,i}^2} = -\frac{P_{\text{est}}}{\alpha} \tag{9.111}$$

where we have defined the factor P_{est} as follows:

$$P_{\text{est}} = \frac{\sum_{i=0}^{L-1} r_{I,i} r_{Q,i}}{\sum_{i=0}^{L-1} r_{I,i}^2} \tag{9.112}$$

From (9.111), the solution of δ depends upon α. However, α is independently determined from (9.107). Once α is computed, it is used in (9.111) to solve for δ.

9.6.3 Compensation of IQ Imbalance

Once both the amplitude imbalance α and the phase imbalance δ are estimated, they can be used for restoring the received IQ signals. Multiplying (9.100) by $\alpha \sin(\delta)$ and adding to (9.101), we have the following:

$$r_{Q,i} + \alpha \sin(\delta) r_{I,i} = \alpha \cos(\delta) y_{Qi} \tag{9.113}$$

From (9.113), $y_{Q,i}$ is solved to give

$$y_{Q,i} = \frac{r_{Q,i} + \alpha \sin(\delta) r_{I,i}}{\alpha \cos(\delta)} \tag{9.114}$$

Substituting (9.111) into (9.114), the following equation results:

Table 9.7
The 12 Subcarriers of the Short Sequence in the IEEE 802.11a

	Subcarrier	Index *k*	Subcarrier	Index *k*	Subcarrier
1	$-1.472(1+j)$	5	$1.472(1+j)$	12	$1.472(1+j)$
2	$-1.472(1+j)$	6	$1.472(1+j)$	13	$-1.472(1+j)$
3	$1.472(1+j)$	10	$1.472(1+j)$	14	$-1.472(1+j)$
4	$1.472(1+j)$	11	$-1.472(1+j)$	15	$1.472(1+j)$

$$y_{Q,i} = \frac{r_{Q,i} - P_{\text{est}} r_{Ii}}{\sqrt{\alpha^2 - P_{\text{est}}^2}} \tag{9.115}$$

From (9.100), $y_{I,i}$ is obtained as

$$y_{I,i} = r_{I,i} \tag{9.116}$$

Both (9.115) and (9.116) can be used to recover the original in-phase and quadrature-phase samples.

9.6.4 Simulation Examples

The short sequence from acquisition defined in the IEEE 802.11a will be used for simulation study. The background noise will be mixed to find the performance of the imbalance estimate in noise. It is assumed that there are no other degradations.

Example 9.11

There are two major assumptions given in (9.103) and (9.104). Both assumptions will be justified using the short sequence of the IEEE 802.11a. There are 12 subcarriers, which are given in Table 9.7.

After performing 16-point IFFT, the time-domain samples are further divided by 4 for further normalization. The powers of the real part and the imaginary part are computed to obtain

Real part power = Imaginary part power = 0.101568
Correlation = 0

Table 9.8
The Original and Impaired Short Sequence Subcarriers

Y_I	Y_Q	Z_I	Z_Q
–1.00	–1.00	–1.04	–1.05
–1.00	–1.00	–1.06	–0.87
1.00	1.00	0.91	0.95
1.00	1.00	0.94	0.77
1.00	1.00	0.91	0.95
1.00	1.00	0.94	0.77
1.00	1.00	0.94	0.77
–1.00	–1.00	–1.04	–1.05
1.00	1.00	0.94	0.77
–1.00	–1.00	–1.04	–1.05
–1.00	–1.00	–1.06	–0.87
1.00	1.00	0.91	0.95

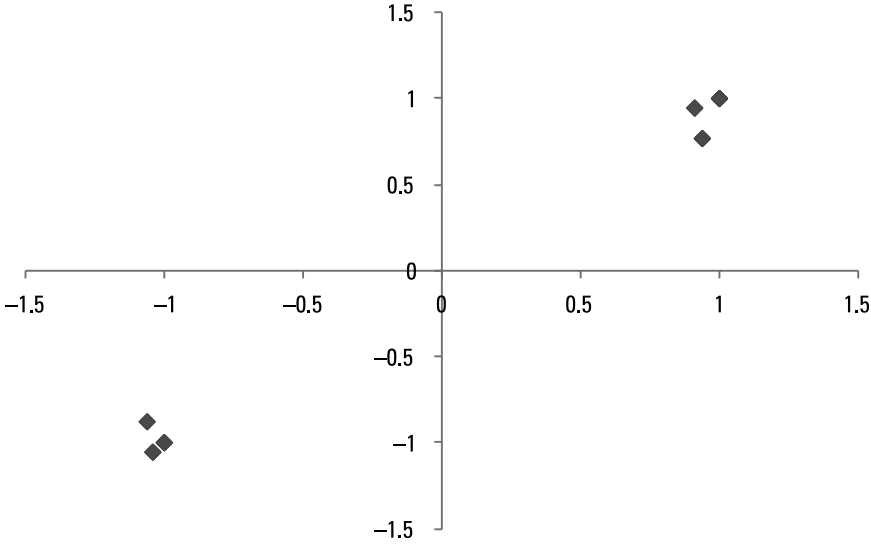


Figure 9.14 The constellation of the original and impaired short sequence.

Clearly, both (9.103) and (9.104) are satisfied

Example 9.12

Assume that the amplitude imbalance $\alpha = 0.9$ ($\varepsilon = 0.1$) and the phase imbalance $\delta = 5° = 0.08727$ radian. First, perform the FFT of the short sequence generated in Example 9.11 and then divide by the normalization factor 1.472 to obtain the original spectrum $Y_i + jY_Q$. Next, (9.100) and (9.101) are applied to generate the time-domain samples with IQ imbalance. The FFT is again applied to obtain the impaired spectrum $Z_i + jZ_Q$. Table 9.8 lists all the subcarriers after dividing the spectrum by the normalization factor 1.472. As Table 9.8 shows, the original spectrum has only two different subcarriers. It is either $1 + j$ or $-1 - j$. After the application of IQ imbalance, the impaired spectrum has now four different subcarriers. They are $0.91 + 0.95j$, $0.94 + 0.77j$, $-1.04 - 1.05j$, and $-1.06 - 0.87j$.

Figure 9.14 is a plot of the constellation of the short sequence with IQ imbalance. Without IQ imbalance, there are only two spectral points. With IQ imbalance, there are four modified spectral points. The total number of unique spectral components then increases from 2 to 4. From Figure 9.14, all these 6 spectral components are clearly seen. Therefore, the occurrence of IQ imbalance can cause spectral distortion.

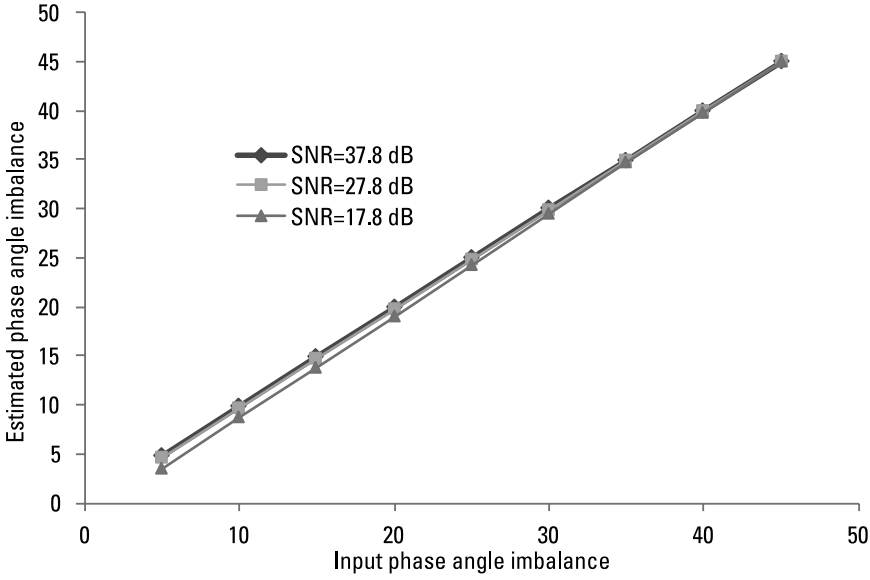


Figure 9.15 Estimate of phase angle imbalance in random noise.

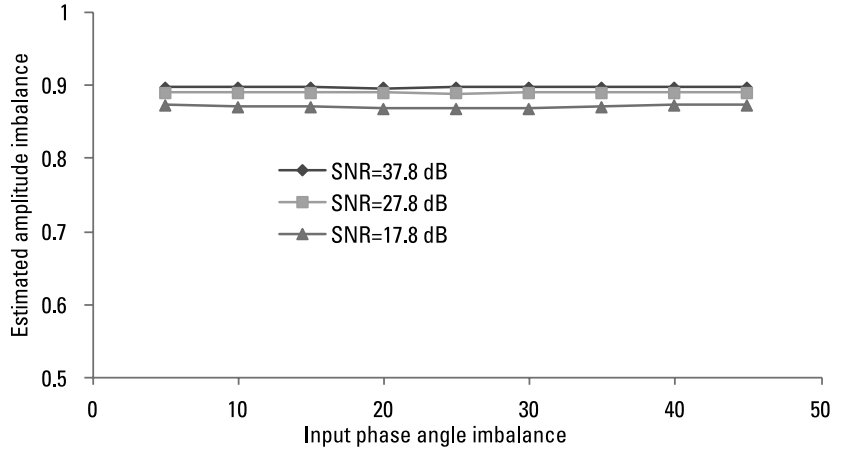


Figure 9.16 Estimate of amplitude imbalance in random noise.

Example 9.13

Example 9.11 shows that both (9.103) and (9.104) are satisfied. This means that the estimate accuracy does not depend upon the magnitude of the imbalance. Assume first that the amplitude imbalance is fixed at $\alpha = 0.9$. The phase angle imbalance δ is allowed to change from 5° to 45° in 5° increments.

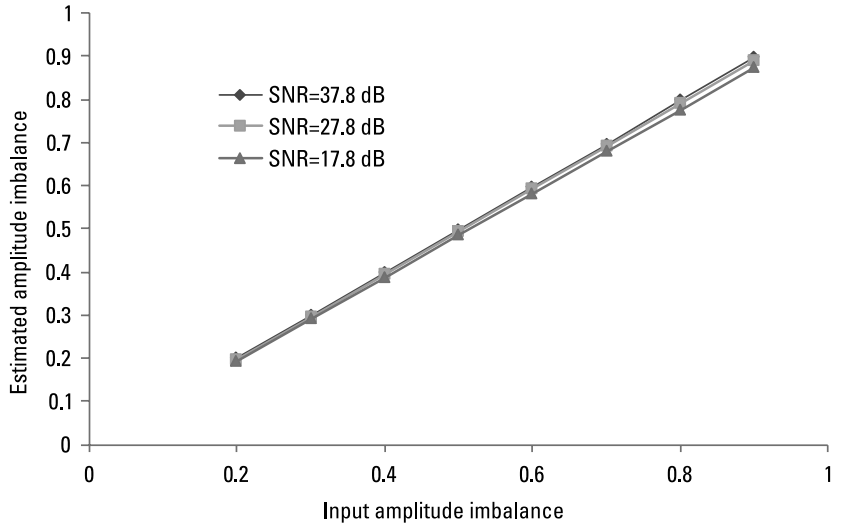


Figure 9.17 Estimate of amplitude imbalance against its true input.

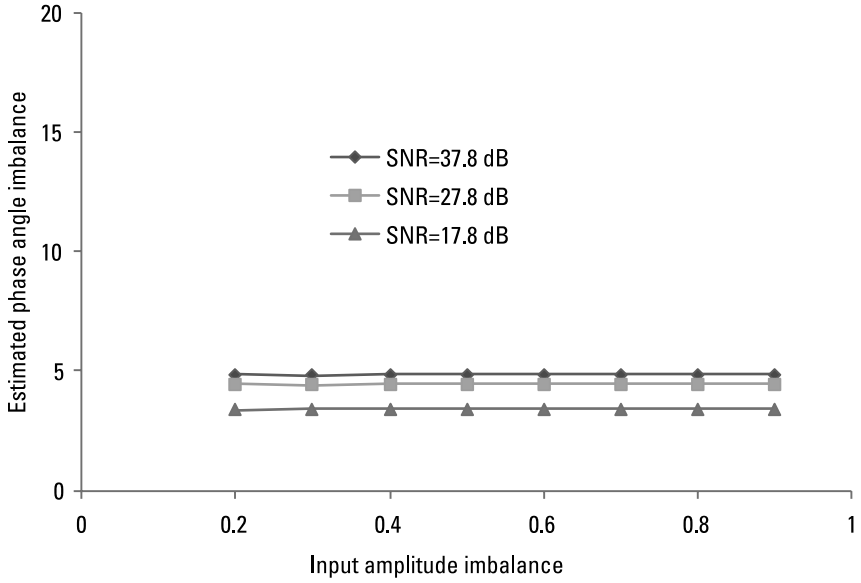


Figure 9.18 Estimate of phase angle imbalance against amplitude imbalance.

The imbalanced samples are generated based upon (9.100) and (9.101). The amplitude imbalance α is then estimated from (9.107) while the phase angle imbalance is estimated from (9.111). Short sequence samples from the second segment to the ninth segment are used for estimation. The total number of samples is $L = 128$. The SNR of 37.8 dB, 27.8 dB, and 17.8 dB is used to obtain the performance in random noise.

Figure 9.15 is a plot of estimated phase angle imbalance against its true input. It can be seen that the estimate of phase angle imbalance is even better at larger angles than with smaller angles. The straight-line curve demonstrates the superiority of the estimate. As the SNR drops, the estimate accuracy also drops because (9.103) and (9.104) are no longer valid.

Figure 9.16 is a plot of estimated amplitude imbalance in random noise. The horizontal line shows that the estimated amplitude imbalance is independent of the input phase angle imbalance. However, as the SNR drops, the estimation accuracy also drops but is still not too far away from the true input of 0.9.

Example 9.14

This example is similar to Example 9.13. However, the input phase angle imbalance δ is fixed at 5° while the amplitude imbalance α is allowed to

vary. Just as before, the SNRs of 37.8 dB, 27.8 dB, and 17.8 dB are used to get the performance of the estimation in noise. Figure 9.17 is a plot of the estimated amplitude imbalance against the input amplitude imbalance. Again, the straight-line curve shows that the estimate is excellent in noise. As the SNR drops, the accuracy also drops slightly when the input amplitude imbalance is large.

Figure 9.18 is a plot of phase angle imbalance against the input amplitude imbalance. The horizontal line shows the estimate is independent of the magnitude of the input amplitude imbalance. However, as the SNR drops, the estimate accuracy also drops.

9.7 Summary

Five major synchronization issues in the receiver front end have been analyzed in detail in this chapter. They are DC offset, CFO, SCO, sampling time offset, and IQ imbalance.

The DC offset is caused by leakage from the receiver LO to the input of either the amplifier or mixer. Its appearance can cause a false alarm through the use of the normalized cross-correlation metric in the signal detection. However, it can also be easily removed though highpass filtering after the analog-to-digital conversion.

The CFO can be caused by the relative motion between the transmitter and the receiver. Using the periodic nature of the preamble signal, this offset can be estimated through correlation between periodic segments. The estimate improves through the averaging of estimates from adjacent segments.

The sampling time offset can cause the frame boundary difference between the transmitted and the received signal. This offset is simply a phase shift in the frequency domain. The effect is similar to the signal delay in a multipath channel and can be removed through channel estimate and adaptation discussed in the later chapters.

The SCO is caused by the difference in sampling time period between the transmitter and receiver clocks. The effect can be analyzed through a joint estimate of both SCO and CFO in the frequency domain from adjacent symbols. In the tracking mode or during data transmission, the known phases of pilot subcarriers are used. In the acquisition modes or during the preamble transmission, the periodic nature of adjacent segments and the zero guard period can be used for simplification. The least square optimization is used to estimate both SCO and CFO since there are more than two pilots in the tracking mode and in the acquisition mode. Under the special condition where either SCO or CFO is 0, a much simpler formula was derived.

The IQ imbalance is caused by defects of circuit parts during the direct conversion process. This results in amplitude and phase disturbances in the LOs of the in-phase and quadrature-phase components. Both can be estimated in the acquisition mode much easier if the preamble waveform satisfies two special conditions. The first is that the in-phase and quadrature-phase signals have the same power. The second is that the in-phase and quadrature-phase signals are not correlated. The short sequence specified in the IEEE 802.11a meets these two conditions. After both phase and amplitude disturbances are estimated, they can be compensated and the original in-phase and quadrature-phase signals are restored.

In the next chapter, both block-type and comb-type pilot patterns for the channel estimation are presented, and channel tracking using the LMS algorithm is also covered.

References

[1] Chiueh, T. -D., P. Y. Tsai, and I. W. Lai, *Baseband Receiver Design for Wireless MIMO-OFDM Communications*, New York: Wiley, 2012.

[2] Razavi, B., "Design Considerations for Direct-Conversion Receivers," *IEEE Transactions on Circuits and Systems*, Vol. 44, No. 6, June 1997, pp. 428–435.

[3] Oppenheim, A. V., and R. W. Schafer, *Digital Signal Processing*, Upper Saddle River, NJ: Prentice Hall, 1975.

[4] Moose, P. H., "A Technique for Orthogonal Frequency Division Multiplexing Frequency Offset Correction," *IEEE Transactions on Communications*, Vol. 42, No. 10, 1994, pp. 2908–2914.

[5] Morelli, M., and U. Mangali, "An Improved Frequency Offset Estimator for OFDM Applications," *IEEE Communication Letters*, 1999.

[6] Speth, M., F. Classen, and H. Meyer, "Frame Synchronization of OFDM Systems in Frequency Selective Fading Channels," *1997 IEEE 47th Vehicular Technology Conference. Technology in Motion*, Vol. 3, pp. 1807–1811.

[7] Speth, M., S. Stefan, and A. Fechtel, "Optimum Receiver Design for Wireless Broad-Band Systems Using OFDM-Part I," *IEEE Transactions on Communications*, Vol. 47, No. 11, November 1999, pp. 1668–1677.

[8] Liu, S. Y., and J. W. Chong, "A Study of Joint Tracking Algorithms of Carrier Frequency Offset and Sampling Clock Offset for OFDM-Based WLANs," *IEEE International Conference on Communications, Circuits and Systems*, Vol. 1, June 2002.

[9] Tubbax, J., et al., "Compensation of Transmitter Imbalance for OFDM Systems," *2004 IEEE International Conference on Acoustics, Speech, and Signal Processing (2004 ICASSP)*, Vol. 2, 2004.

[10] Tseng, H. -Y., et al., "Compensation of Imbalance and DC Offset for OFDM Transmission over Frequency Selective Channels," *2008 IEEE International Conference on Communications*, 2008.

[11] Held, I., et al., "Low Complexity Digital IQ Imbalance Correction in OFDM WLAN Receivers," *2004 IEEE 59th Vehicular Technology Conference, VTC 2004-Spring*, Vol. 2, pp. 1172–1176.

10

Channel Estimation and Tracking

10.1 Introduction

In a radio channel, there are normally reflecting objects and scatters in the signal propagation path. These obstacles include mountains, buildings, and even the ground. The directly transmitted signal is then reflected. The receiver may then receive multiple versions of the original signal with a different amplitude, phase, and arrival time. These multipath signals add together and may cause significant distortion of the received signal.

For OFDM, the signal bandwidth is normally larger than the coherence bandwidth of the channel. When this happens, the received signal suffers frequency selective fading. In the time domain, this also means the delay spread is longer than the reciprocal of the signal bandwidth. This is also why a guard interval is added after each OFDM symbol to minimize the ISI.

The channel equalization technique for OFDM is quite simplified in the frequency-selective channel. The channel estimation is normally performed with the transmission of training symbols before the actual signal transmission. These training symbols are designed using subcarriers with a known amplitude and phase to the receiver. Based upon the received amplitude and phase of each subcarrier, the channel is easily estimated.

After the channel estimation, the original signal can be restored. If the channel is steady and does not change fast over time, the use of training

symbols to aid the channel estimation in the data acquisition mode is normally enough. However, the channel may not be steady and it is necessary to estimate the channel frequently. The latter is done by inserting pilot subcarriers besides the data subcarriers to aid in the estimation.

If the channel varies fast, it can also be tracked for each received data symbol. Based upon the estimation error and the received signal, the channel at each data subcarrier is tracked using the method of steep descent. For OFDM, only one tap is necessary and the process is also greatly simplified.

In the following sections, the possible pilot structures are introduced first. For each pilot structure, the channel estimation is presented. After that, the tracking algorithm is analyzed. The simulation examples are also given to understand the effectiveness of each algorithm.

10.2 Pilot Patterns

For OFDM, the pilot patterns are usually given in a time-frequency diagram. Both time and frequency are represented in a group of small circles. Each circle represents an OFDM symbol in the time axis. In other words, a circle along the time axis has the time duration of an equivalent OFDM symbol. The same circle along the frequency axis has the equivalence of a subcarrier. If a circle is filled, it carries pilot subcarriers.

Many different pilot patterns can be designated. However, the most commonly used patterns are the block type and comb type. Both types are given in the following sections.

10.2.1 Block-Type Pilot Pattern

Figure 10.1 shows a typical block type pilot pattern [1]. Along the time axis, there is a filled node for every four empty nodes. Along the frequency axis, all the nodes are filled at a given time period. In terms of OFDM terminology, it means after for every four OFDM symbols, a symbol is transmitted with all its subcarriers designated as pilot subcarriers.

Using the block type pattern, pilot subcarriers are periodically transmitted for the new channel estimation. Since pilot subcarriers do not carry data, the data transmission rate is reduced in exchange for better bit error rate performance. The period of pilot transmission depends upon the system requirement and the channel bandwidth.

Assuming that T_g is the guard interval, T_s is the symbol interval, T_{bpsc} is the number of data bits per subcarrier, N_{sd} is the number of data subcarriers

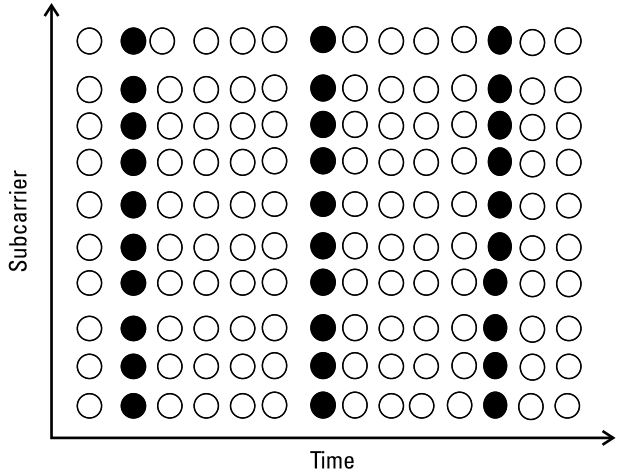


Figure 10.1 Block-type pilot pattern.

per symbol, and M is the data symbol count between pilot symbols, then the data transmission rate R is given here:

$$R = \frac{MN_{\text{bpsc}}N_{\text{sd}}C}{(M+1)\left(T_g + T_s\right)} \tag{10.1}$$

where C is the coding rate. If there is no pilot transmission, then the following equation results in:

$$R = \frac{N_{\text{bpsc}}N_{\text{sd}}C}{\left(T_g + T_s\right)} \tag{10.2}$$

Example 10.1

Assume that the modulation is QPSK and the channel spacing is 20 MHz. From Table 3.1, $T_g = 0.8$ μs, $T_s = 3.2$ μs, $N_{\text{bpsc}} = 2$, $N_{\text{sd}} = 48$, and $C = 1/2$ for the convolutional code. If there is no pilot transmission, the data rate is then

$$R = \frac{2 * 48 * 0.5}{3.2 + 0.8} = 12 \text{ Mbps}$$

Assuming that now that one pilot symbol is sent every 4 data symbols, then the new data rate becomes

$$R = \frac{4 * 2 * 48 * 0.5}{5 * 4} = 9.6 \text{ Mbps}$$

With the transmission of a pilot symbol after 4 data symbols, the data rate then reduces from 12 Mbps to 9.6 Mbps.

During the time between pilot transmissions, the existing channel estimation is used. Therefore, the block-type pilot pattern is normally used when the channel is not fast varying. However, it can still be slightly updated through tracking as will be discussed later.

10.2.2 Comb-Type Pilot Pattern

Figure 10.2 shows a comb-type pilot pattern [2, 3]. In this case, pilot subcarriers are sent during every data symbol transmission. However, inside each data symbol, only certain subcarriers are designated as pilot subcarriers. In Figure 10.2, subcarriers 0, 4, and 8 are pilot subcarriers and the others are data subcarriers. These three pilot subcarriers exist during every symbol transmission.

Strictly speaking, the data rate is still reduced because the pilot subcarriers for each data symbol do not carry data. The data rate is computed according to (10.2). Since the pilot subcarriers have the known amplitude and phase to the receiver, the channel can still be estimated at these pilot subcarriers. The channel at the data subcarriers then needs to be estimated through other means such as interpolation.

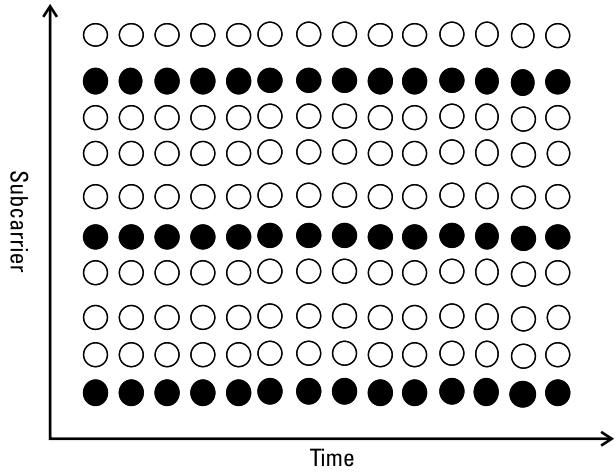


Figure 10.2 Comb-type pilot pattern.

Since the channel is estimated at each data symbol, this type of pilot pattern is suitable for a fast varying channel. If the number of pilot carriers is not large enough, the accuracy of channel estimation at data subcarriers will suffer. The IEEE 802.11a specifies four subcarriers as pilot signals. They can be used for frequency offset and sampling clock offset estimation as discussed in Chapter 9. If necessary, they can also be used for channel estimation.

10.3 Channel Estimation

There are two major types of pilot patterns. One is the block type and the other is the comb type. The channel estimation for both types is given here.

10.3.1 Channel Estimation for the Block-Type Pilot Pattern

To send the OFDM signal, the frequency-domain subcarriers X_k is first transformed to the time-domain samples x_n following the inverse discrete Fourier transform:

$$x_n = \frac{1}{N}\sum_{k=0}^{N-1} X_k e^{j2\pi kn/N} \tag{10.3}$$

After propagating through the multipath channel, the signal x_n is corrupted. Assume that the impulse response is h_n. Then the received sample y_n is given by the following convolution of x_n and x_n:

$$y_n = \sum_{i=0}^{N-1} x_i h_{n-i} + w_n \tag{10.4}$$

where w_n is an additive Gaussian noise. The noise type could be thermal noise, interference noise, or others. To recover the transmitted subcarriers, the time-domain samples y_n are again transformed back to the frequency domain following the discrete Fourier transform

$$Y_k = \sum_{n=0}^{N-1} y_n e^{-j2\pi nk/N} \tag{10.5}$$

Substituting (10.4) into (10.5), we have

$$Y_k = \sum_{n=0}^{N-1}\sum_{i=0}^{N-1} x_i h_{n-i} e^{-j2\pi nk/N} + \sum_{n=0}^{N-1} w_n e^{-j2\pi nk/N} \tag{10.6}$$

The second term of (10.6) is simply W_k representing the kth noise sample in the frequency domain. Assuming that the frequency response of the channel is given by H_k, then it is related to h_n by the following inverse Fourier transform

$$h_i = \frac{1}{N}\sum_{l=0}^{N-1} H_l e^{j2\pi li/N} \tag{10.7}$$

Replacing i by $n - i$ in (10.7), we have

$$h_{n-i} = \frac{1}{N}\sum_{l=0}^{N-1} H_l e^{j2\pi l(n-i)/N} \tag{10.8}$$

Substituting (10.3) and (10.8) into (10.6), the following is obtained:

$$\begin{aligned} Y_k &= \sum_{n=0}^{N-1}\sum_{i=0}^{N-1}\frac{1}{N}\sum_{m=0}^{N-1} X_m e^{j2\pi mi/N}\frac{1}{N}\sum_{l=0}^{N-1} H_l e^{j2\pi l(n-i)/N} e^{-j2\pi nk/N} + W_k \\ &= \sum_{m=0}^{N-1}\sum_{l=0}^{N-1} X_m H_l \left[\frac{1}{N}\sum_{n=0}^{N-1} e^{j2\pi n(l-k)/N}\right]\left[\frac{1}{N}\sum_{i=0}^{N-1} e^{j2\pi i(m-l)/N}\right] + W_k \end{aligned} \tag{10.9}$$

However, the summation inside the bracket of (10.9) has the following relationship:

$$\sum_{k=0}^{N-1} e^{-j2\pi rk/N} = \begin{cases} N & \text{for } r = jN, j \text{ an integer} \\ 0 & \text{otherwise} \end{cases} \tag{10.10}$$

Using (10.10), (10.9) then becomes

$$\begin{aligned} Y_k &= \sum_{l=0}^{N-1} X_l H_l \left[\frac{1}{N}\sum_{n=0}^{N-1} e^{j2\pi n(l-k)/N}\right] + W_k \\ &= X_k H_k + W_k \end{aligned} \tag{10.11}$$

Equation (10.11) says that the discrete Fourier transform of the received samples is the addition of noise spectrum and the product of the transmitted data spectrum and channel spectrum. This relationship is true for every subcarrier index k. From (10.11), the channel response is given here:

$$H_k = \frac{\widehat{Y_k}}{X_k} \quad \text{where } \widehat{Y_k} = Y_k - W_k \tag{10.12a}$$

If the random noise is negligible small or 0, then H_k is exactly recovered as given here:

$$H_k = \frac{Y_k}{X_k} \quad \text{if } W_k = 0 \tag{10.12b}$$

Using the pilot subcarriers, X_k is known in the receiver. Therefore, from the received frequency domain samples, Y_k, the channel frequency response H_k is then obtained.

For block-type pilot patterns, (10.12) gives the newest channel estimation for every M data symbols. Based upon this estimation, the data subcarriers can be recovered before any new channel estimation as given here:

$$X_k = \frac{Y_k}{H_k} \tag{10.13}$$

10.3.2 Channel Estimation for Comb-Type Pilot Pattern

For the comb-type pilot pattern, there are only a finite number of pilot subcarriers. Even though the channel response can be estimated at the pilot subcarriers, there is no information available to estimate that at the data subcarriers. One popular method is to interpolate at the data subcarriers based upon the known channel response at the pilot subcarriers.

Depending upon the number of points used for interpolation, the interpolation can be linear, parabolic, and cubic. The number of points used is 2, 3, and 4 for linear, parabolic, and cubic interpolators, respectively. The Lagrange interpolation formula is used for each of these interpolators

10.3.2.1 Linear Interpolation

The easiest way is to use only two points for straight-line interpolation. The Lagrange interpolation formula is given by the following equation:

$$H(k) = \frac{\left(k - k_1\right)}{\left(k_0 - k_1\right)} H\left(k_0\right) + \frac{k - k_0}{\left(k_1 - k_0\right)} H\left(k_1\right) \tag{10.14}$$

where k_0 and k_1 are the two pilot subcarriers indexes and k is the data subcarrier index to be interpolated. The function values $H(k_0)$ and $H(k_1)$ are channel responses at the pilot subcarrier index k_0 and k_1. From (10.14), it is clear $H(k) = H(k_0)$ when $k = k_0$ and $H(k) = H(k_1)$ when $k = k_1$.

Assume that the distance between k_0 and k_1 is D and $k_1 > k_0$. Assuming that k is larger than k_0, then k can be written as

$$k = k_0 + \mu D \tag{10.15}$$

$$k_1 = k_0 + D \tag{10.16}$$

Substituting (10.15) and (10.16) into (10.14), we have [1]

$$H(k) = (1-\mu)H\left(k_0\right) + \mu H\left(k_1\right) \tag{10.17a}$$

$$k = k_0 + \mu D \tag{10.17b}$$

Equation (10.17) is the linear interpolation at point k using the two known data points at k_0 and k_1. For $k < k_0$, $\mu < 0$ and (10.17) still applies.

10.3.2.2 Parabolic Interpolation

If there are three data points available, then it is parabolic interpolation. Assuming that these three points are k_0, k_1, and k_2, then the Lagrange interpolation formula is given here:

$$\begin{aligned} H(k) = {} & \frac{\left(k-k_1\right)\left(k-k_2\right)}{\left(k_0-k_1\right)\left(k_0-k_2\right)}H\left(k_0\right) \\ & + \frac{\left(k-k_0\right)\left(k-k_2\right)}{\left(k_1-k_0\right)\left(k_1-k_2\right)}H\left(k_1\right) + \frac{\left(k-k_0\right)\left(k-k_1\right)}{\left(k_2-k_0\right)\left(k_2-k_1\right)}H\left(k_2\right) \end{aligned} \tag{10.18}$$

Let the distance between consecutive data points be a constant such that $k_{m+1} - k_m = D$ for $m = 0, 1$. Then we have the following:

$$k = k_0 + \mu D \tag{10.19}$$

$$k_1 = k_0 + D \tag{10.20}$$

$$k_2 = k_0 + 2D \tag{10.21}$$

Substituting (10.19), (10.20), and (10.21) into (10.18), we have [3]

$$H(k) = \frac{(\mu-1)(\mu-2)}{2}H\left(k_0\right) + \mu(2-\mu)H\left(k_1\right) + \frac{\mu(\mu-1)}{2}H\left(k_2\right) \tag{10.22a}$$

$$k = k_0 + \mu D \tag{10.22b}$$

Equation (10.22) is the second-order Lagrange interpolation using three data points k_0, k_1, and k_2.

10.3.2.3 Cubic Interpolation

The interpolation order can go higher to use four data points, k_0, k_1, k_2, and k_3. The Lagrange interpolation formula is then given here:

$$H(k) = \frac{(k-k_1)(k-k_2)(k-k_3)}{(k_0-k_1)(k_0-k_2)(k_0-k_3)} H(k_0) + \frac{(k-k_0)(k-k_2)(k-k_3)}{(k_1-k_0)(k_1-k_2)(k_1-k_3)} H(k_1)$$
$$+ \frac{(k-k_0)(k-k_1)(k-k_3)}{(k_2-k_0)(k_2-k_1)(k_2-k_3)} H(k_2) + \frac{(k-k_0)(k-k_1)(k-k_2)}{(k_3-k_0)(k_3-k_1)(k_3-k_2)} H(k_3) \tag{10.23}$$

Again, the distance between consecutive data points is a constant D. That means $k_{m+1} - k_m = D$ for $m = 0$, 1, and 2. However, we let the interpolation data point k be computed starting from k_1. We then have the following:

$$k = k_1 + \mu D \tag{10.24}$$

$$k_0 = k_1 - D \tag{10.25}$$

$$k_2 = k_1 + D \tag{10.26}$$

$$k_3 = k_1 + 2D \tag{10.27}$$

Substituting from (10.24) to (10.27) into (10.23), the following is obtained:

$$H(k) = C_0 H(k_0) + C_1 H(k_1) + C_2 H(k_2) + C_3 H(k_3) \tag{10.28a}$$

$$k = k_1 + \mu D \tag{10.28b}$$

where C_0, C_1, C_2, and C_3 are given here [4]:

$$C_0 = -\frac{\mu(\mu-1)(\mu-2)}{6} \tag{10.29a}$$

$$C_1 = \frac{(\mu^2 - 1)(\mu - 2)}{2} \tag{10.29b}$$

$$C_2 = -\frac{\mu(\mu + 1)(\mu - 2)}{2} \tag{10.29c}$$

$$C_3 = \frac{\mu(\mu^2 - 1)}{6} \tag{10.29d}$$

The choice of linear, parabolic, or cubic interpolation will certainly affect interpolation accuracy. There are also three other factors that will have significant impact on the accuracy. The first is the number of pilot subcarriers available for interpolation. The second is the degree of channel spectral variations. If the channel frequency response has too many spectral cycles, then more pilot subcarriers are needed for interpolation. The last is the effects due to random noise. The dependence of interpolation accuracy on all of these factors is illustrated in the following examples.

Example 10.2

To illustrate the dependence of interpolation accuracy on the number of pilot subcarriers, channel spectral variations, and the interpolation method, a simple multipath model is used. Assume that the receiver receives signals from two major paths. One is the main signal x_n and the other is the reflected signal bx_{n-m} where b is the reflected amplitude and m is the sample delay. The received signal y_n is then the sum of these two signals:

$$y_n = x_n + bx_{n-m} \tag{10.30}$$

Equation (10.30) is a special case of the more general formula given in (12.11).

Taking DFT on both sides of (10.30), we have

$$\begin{aligned} Y_k &= X_k + bX_k e^{-j2\pi mk/N} \\ &= X_k\left(1 + be^{-j2\pi mk/N}\right) \end{aligned} \tag{10.31}$$

From (10.31), the frequency channel response H_k is then given below:

$$\begin{aligned} H_k &= 1 + be^{-j2\pi mk/N} \\ &= \left(1 + b\cos\left(\frac{2\pi mk}{N}\right)\right) - jb\sin\left(\frac{2\pi mk}{N}\right) \end{aligned} \tag{10.32}$$

The real part H_{kr} and the imaginary part H_{ki} of H_k are then written below:

$$H_{kr} = 1 + b\cos\left(\frac{2\pi mk}{N}\right) \tag{10.33}$$

$$-H_{ki} = b\sin\left(\frac{2\pi mk}{N}\right) \tag{10.34}$$

In this example, the channel frequency response is a sine wave. The period is obtained by setting the cos argument to 2π. We then have

$$\text{Period} = \frac{N}{m} \tag{10.35}$$

Clearly, as the delay m increases, the period decreases and it becomes more difficult to interpolate. This is illustrated in the next few examples.

Example 10.3

In this example, we illustrate how the number of pilot subcarriers may affect the interpolation accuracy. Parabolic interpolation is used and the number of pilot subcarriers is 8, 16, and 32. However, the total number of subcarriers is set to 64. Assuming that $m = 3$, $N = 64$, and $b = 0.9$, Figures 10.3, 10.4, and 10.5 are the plots of both the original H_{kr} and $-H_{ki}/b$ and their interpolation against the subcarrier index k.

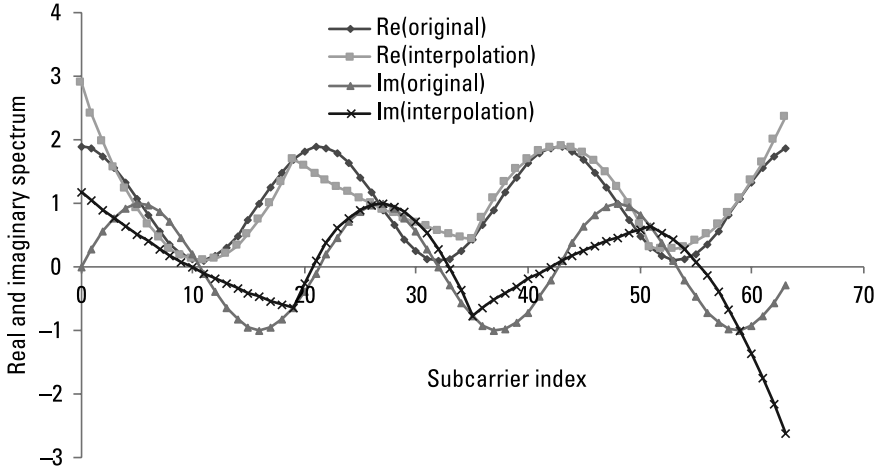


Figure 10.3 Parabolic interpolation using eight pilot subcarriers.

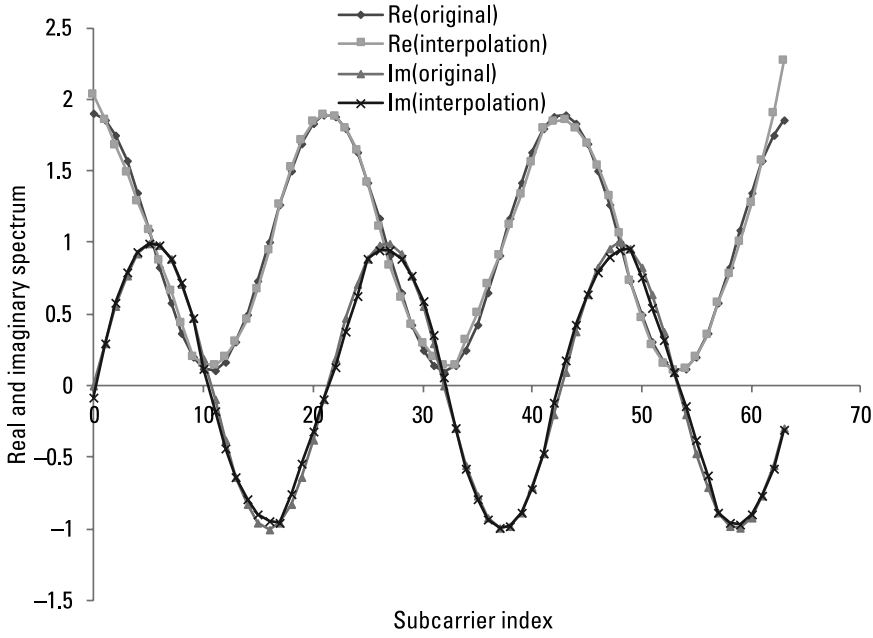


Figure 10.4 Parabolic interpolation using 16 pilot subcarriers.

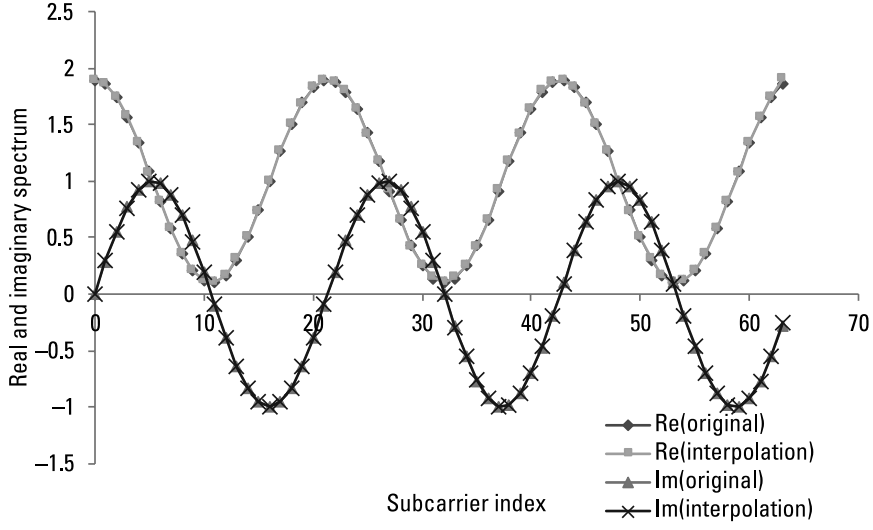


Figure 10.5 Parabolic interpolation using 32 pilot subcarriers.

It is clear from the plots that the interpolation accuracy improves as the number of pilot subcarriers M increases. When $M = 32$, the interpolated curve is almost indistinguishable from the original. However, when $M = 8$, the interpolation is not considered as being good enough. The period is $64/3 \cong 21$ and is confirmed from the figure. Roughly, each cycle requires that at least 4 pilot subcarriers and 12 pilot subcarriers are needed in this example. This also shows when $M = 16$, the interpolation significantly improves.

Example 10.4

In this example, we show how the interpolation method affects the interpolation accuracy. When the number of pilot subcarriers is insufficient, the interpolation cannot have a good match to the original. Under this condition, the interpolation method really does not matter. It is not surprising that the linear interpolation may even be better than the higher-order interpolation. If there are a sufficient number of pilot subcarriers that exist, then the higher-order interpolation shows some advantage in exchange for more computation time. However, if there are more than enough pilot carriers, then even linear interpolation can work well.

Figure 10.6 shows the linear interpolation using 16 pilot subcarriers and $m = 3$. The interpolation is not that bad. However, it is certainly not as good as the parabolic interpolation shown in Figure 10.4. Figure 10.7 shows linear interpolation using 32 pilot subcarriers and $m = 3$. The interpolation also has a fairly close match with the original. This also suggests that a lower-order interpolation is acceptable when more than enough numbers of pilot subcarriers exist.

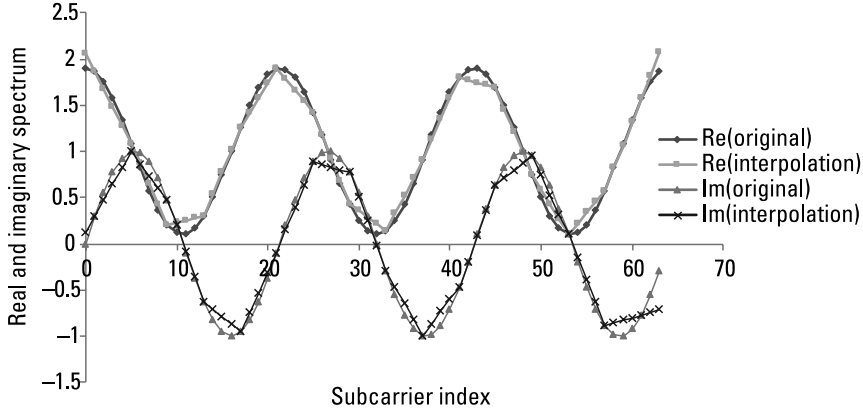


Figure 10.6 Linear interpolation using 16 pilot subcarriers.

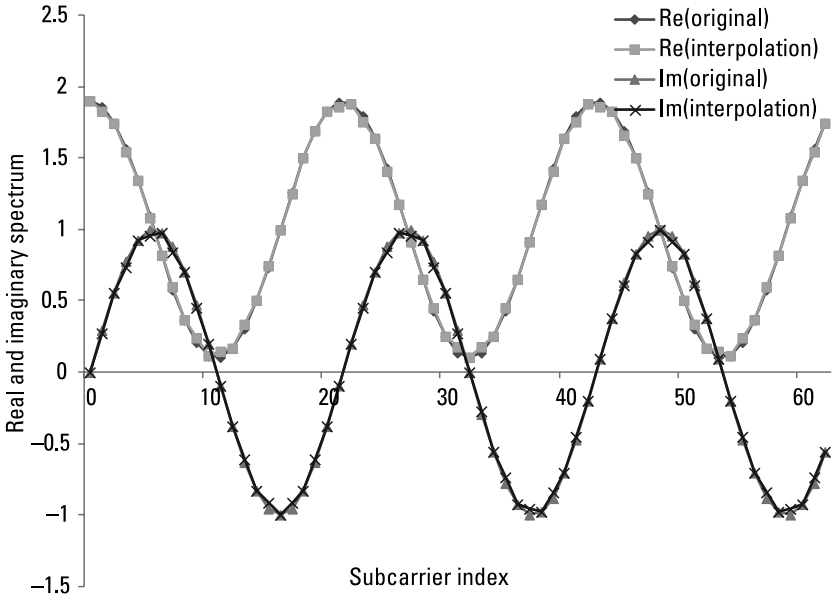


Figure 10.7 Linear interpolation using 32 pilot subcarriers.

Example 10.5

In this example we show how the delay may affect the interpolation. If the delay decreases, the spectral period increases. Then the number of pilot subcarriers required also decreases. Figure 10.8 shows the parabolic interpolation with the delay reduced to $m = 1$ and the number of pilot subcarriers used is 8. In this case, there is only one spectral cycle and the match is much better than that given in Figure 10.3.

Figure 10.9 is another plot with the number of subcarriers reduced to four. The pilot subcarriers selected are at $k = 7$, 23, 39, and 55. Because of the lack of pilot subcarriers near the beginning and end of the symbol, the match is not good there. However, the overall match is not bad. This is also the early statement that a minimum of 4 pilot subcarriers are required for each spectral cycle.

Example 10.6

In this example, the effects of random noise on the accuracy of parabolic interpolation are investigated. Four different numbers of subcarriers are considered

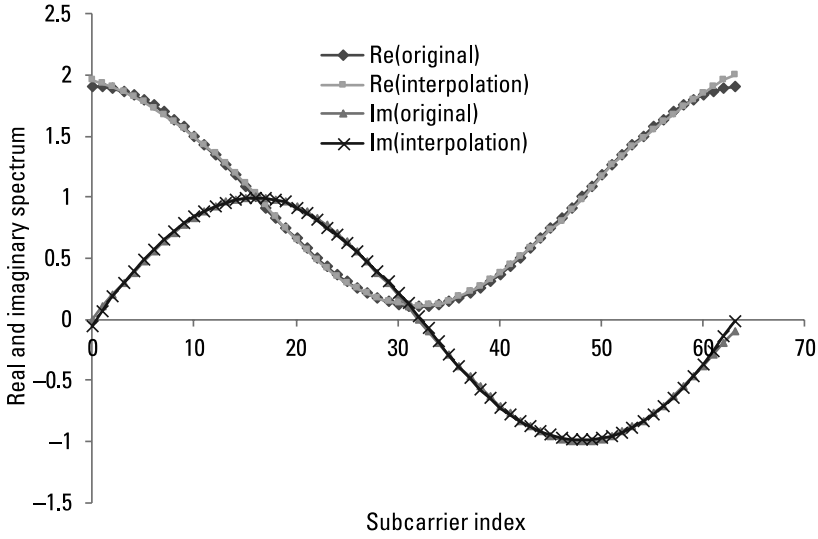


Figure 10.8 Parabolic interpolation with delay = 1 sample and 8 pilot subcarriers.

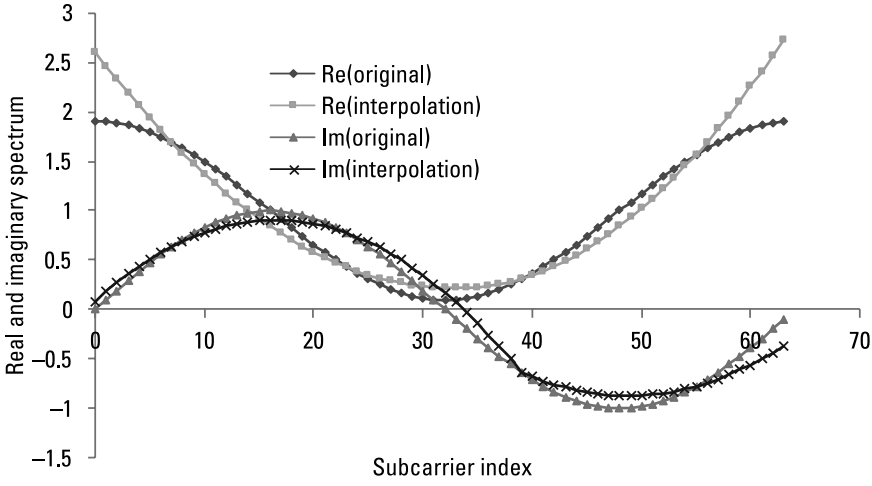


Figure 10.9 Parabolic interpolation with delay = 1 sample and 4 pilot subcarriers.

and they are $M = 4$, 8, 16, and 32. The channel parameters used are the same as previous examples. They are $m = 3$, $N = 64$, and $b = 0.9$. For simplicity, assuming that $X_k = 1$, then $Y_k = H_k$. To compute SNR, the signal amplitude is then the same as the channel amplitude. Both the channel amplitude and

noise amplitude at these pilot subcarriers can be averaged. The SNR in decibels is then computed based upon these average values.

Assume that H_{kr} and H_{ki} represent the true channel response at subcarrier k. Assume that also $\widehat{H_{kr}}$ and $\widehat{H_{kı}}$ represent the estimated channel response at subcarrier k. Then the average deviation δ per subcarrier is computed as follows:

$$\delta = \left(\frac{1}{8}\right)\sqrt{\sum_{k=0}^{63}\left[\left(H_{kr} - \widehat{H_{kr}}\right)^2 + \left(H_{ki} - \widehat{H_{kı}}\right)^2\right]} \tag{10.36}$$

Assume also that the peak channel amplitude at a subcarrier is $|H_{\text{peak}}|$. Then the deviation percentage ε is computed as

$$\varepsilon = \frac{\delta}{|H_{\text{peak}}|} \tag{10.37}$$

Figure 10.10 is a plot of ε in percentage against the SNR and four different numbers of pilot subcarriers. One common trend is the deviation

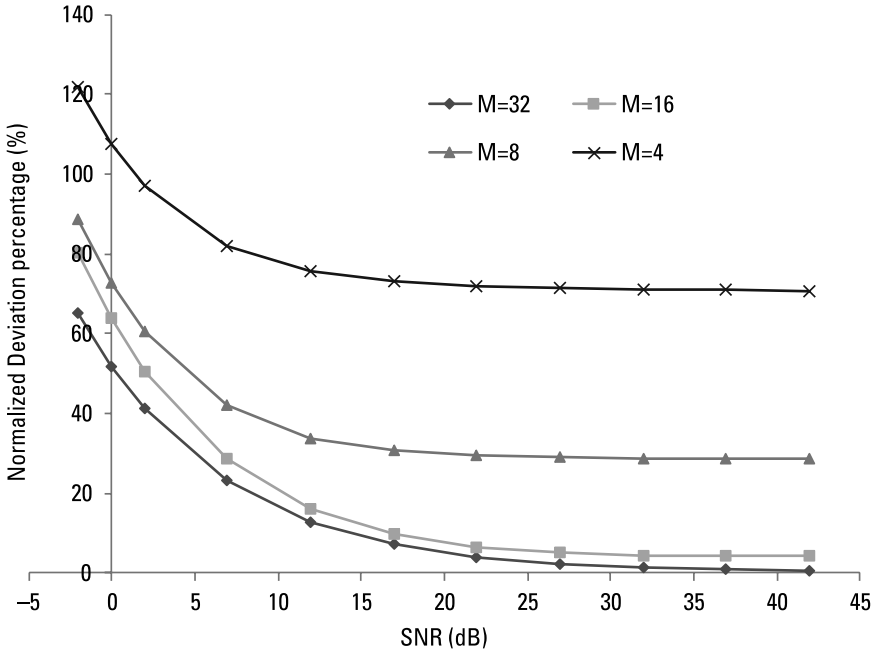


Figure 10.10 The deviation percentage of channel estimation against the SNR and the number of pilot subcarriers.

percentage increases as either the SNR decreases or M decreases. At SNR = 42 dB, the ε is 28.7% for $M = 8$ and 70.7% for $M = 4$. The high percentage deviation at such a high SNR shows that both $M = 4$ and $M = 8$ are unacceptable. For both $M = 32$ and $M = 16$, the deviation percentage is acceptable at a high SNR but not at a low SNR. If we would like the deviation percentage to be less than 10%, then the SNR has to be higher than 17 dB.

Also, in this example, it is assumed that $m = 3$ or 3 sample delay. This is why $M = 16$ works reasonably well. As m increases, the number of pilot subcarriers has to increase further. However, a higher number of pilot subcarriers takes away the data bandwidth and reduces the date rate. This could be a design problem for using the comb-type pattern.

In summary, the number of pilot subcarriers required for interpolation increases with increasing delay. If the number of subcarriers is insufficient, then the accuracy of interpolation also suffers.

10.4 Channel Tracking

As discussed previously, the channel frequency response can be exactly solved using the block-type pilot pattern. The simplicity of (10.12) shows the advantage of OFDM against the single-carrier system over the frequency-selective channel. However, this pattern requires the use of all subcarriers to be pilot subcarriers and cannot be used frequently. The comb-type pilot pattern is used within each OFDM data symbol and is suitable for a fast-varying channel. Since only a finite number of pilot subcarriers are selected, the channel at other data subcarriers is estimated through interpolation. The accuracy may not be great depending upon the multipath delay and the number of pilot subcarriers.

10.4.1 The LMS Algorithm

Another method is to adaptively adjust the channel response based upon the LMS algorithm using a gradient search [5, 6]. The technique intends to minimize the mean-square error between the information sequence and its estimation. At the kth subcarrier, assume that the information data is X_k and the received data is Y_k. The information data is estimated to be G_kY_k and G_k is the weighting coefficient. From (10.13), the weight G_k is simply the inverse of the channel frequency response. The difference between the desired and estimated response is the error signal ε_k:

$$\varepsilon_k = X_k - G_kY_k \tag{10.38}$$

In (10.38), all quantities are complex in general. The G_k is determined by minimizing $|\varepsilon_k|^2$ with respect to G_k. This is done by setting $(\partial|\varepsilon_k|^2)/\partial G_k$ to zero. The partial derivative is given here:

$$\frac{\partial|\varepsilon_k|^2}{\partial G_k} = -2\varepsilon_k Y_k^* \tag{10.39}$$

Since Y_k is not 0, the only way for the left side of (10.39) to be 0 is to set ε_k to 0. After doing that in (10.38), the G_k is solved to give

$$G_k = \frac{X_k}{Y_k} \tag{10.40}$$

Clearly, G_k is the inverse of channel frequency response H_k given in (10.12). This is just another way of interpreting how H_k is obtained from the point of view of minimizing the MSE.

Since X_k is not always available, a weight-adjusting procedure is available to solve the problem. This procedure is the least mean square (LMS) algorithm using the method of steepest descent. The weight can be adjusted adaptively without any knowledge of the information sequence. Equation (10.40) applies to every subcarrier index k. The one-tap equalizer is then given by the following equation:

$$G_{j+1,k} = G_{j,k} + \mu\nabla\left(\left|\varepsilon_{j,k}\right|^2\right) \tag{10.41}$$

where

$G_{j,k}$: weight vector at the current or the jth iteration;

$G_{j+1,k}$: the weight vector at the next or the $j + 1$th iteration;

$\nabla(|\varepsilon_{j,k}|^2)$: the gradient or partial derivative of the error $|\varepsilon_{j,k}|^2$ against $G_{j,k}$;

μ: a scalar constant that controls the rate of convergence;

$\varepsilon_{j,k}$: error between the estimation $G_{j,k}Y_k$ and the reference X_k.

Substituting (10.39) into (10.41), we have

$$G_{j+1,k} = G_{j,k} - 2\mu\varepsilon_{j,k}Y_k^* \tag{10.42}$$

The adaptation index j can also be considered as the jth received OFDM symbol. Equation (10.42) shows the adaptation depends upon the estimation error and the received subcarrier Y_k^*.

10.4.2 The Condition for Convergence

The adaptation given by (10.42) needs to be convergent or else it will fail. To derive the condition for convergence, the error $\varepsilon_{j,k}$ can be written in terms of $G_{j,k}$ given by (10.38) [6]:

$$G_{j+1,k} = G_{j,k} - 2\mu\left(X_k - G_{j,k}Y_k\right)Y_k^* \tag{10.43}$$

$$= G_{j,k}\left(1 + 2\mu\left|Y_k\right|^2\right) - 2\mu X_k Y_k^* \tag{10.44}$$

Taking the expected value on both sides of (10.44), we have

$$E\left(G_{j+1,k}\right) = E\left(G_{j,k}\right)\left(1 + 2\mu\left|Y_k\right|^2\right) - 2\mu X_k Y_k^* \tag{10.45}$$

Setting $j = 0$, 1, and 2 and expressing the $G_{j+1,k}$ in terms of $G_{0,k}$, we have

$$E\left(G_{1,k}\right) = E\left(G_{0,k}\right)\left(1 + 2\mu\left|Y_k\right|^2\right) - 2\mu X_k Y_k^* \tag{10.46}$$

$$E\left(G_{2,k}\right) = E\left(G_{0,k}\right)\left(1 + 2\mu\left|Y_k\right|^2\right)^2 - 2\mu X_k Y_k^* - 2\mu X_k Y_k^*\left(1 + 2\mu\left|Y_k\right|^2\right) \tag{10.47}$$

$$\begin{aligned} E\left(G_{3,k}\right) = {} & E\left(G_{0,k}\right)\left(1 + 2\mu\left|Y_k\right|^2\right)^3 - 2\mu X_k Y_k^* - 2\mu X_k Y_k^*\left(1 + 2\mu\left|Y_k\right|^2\right) \\ & - 2\mu X_k Y_k^*\left(1 + 2\mu\left|Y_k\right|^2\right)^2 \end{aligned} \tag{10.48}$$

Repeating the process given in (10.46), (10.47), and (10.48), we have for the $n + 1$th iteration,

$$E\left(G_{n+1,k}\right) = E\left(G_{0,k}\right)\left(1 + 2\mu\left|Y_k\right|^2\right)^{n+1} - \sum_{i=0}^{n} 2\mu X_k Y_k^*\left(1 + 2\mu\left|Y_k\right|^2\right)^i \tag{10.49}$$

If n goes to infinity, the second term converges to the following:

$$\sum_{i=0}^{\infty} 2\mu X_k Y_k^* \left(1+2\mu\left|Y_k\right|^2\right)^i = 2\mu X_k Y_k^* \sum_{i=0}^{\infty}\left(1+2\mu\left|Y_k\right|^2\right)^i = \frac{2\mu X_k Y_k^*}{-2\mu\left|Y_k\right|^2} = -\frac{X_k}{Y_k} \tag{10.50}$$

Equation (10.50) shows that the second term of (10.49) converges to a finite number. In order for $E(G_{m+1,k})$to converge, the first term of (10.49) must be 0 and the following condition must be satisfied:

$$-1 < 1+2\mu\left|Y_k\right|^2 < 1 \tag{10.51}$$

The condition for convergence is then the following:

$$\frac{-1}{\left|Y_k\right|^2} < \mu < 0 \tag{10.52}$$

Equation (10.52) shows that for convergence, the scalar constant μ is related to the total power of the received kth subcarrier and it must be less than 0.

Substituting (10.50) and (10.52) to (10.49) and setting n to ∞, we finally have

$$E\left(G_{\infty,k}\right) = \frac{X_k}{Y_k} \tag{10.53}$$

Equation (10.53) is the same as (10.40). This shows that the gain adaptation given by (10.42) converges to the exact or Wiener-Hopf equation given by (10.40).

Equation (10.53) is important in the sense that the ideal solution can be reached by simply performing the adaptation given by (10.42) as long as μ is chosen to satisfy (10.52). The error $\varepsilon_{j,k}$ given by (10.38) requires the knowledge of the reference X_k which is unknown. However, it can be estimated from the previous $(j-1)$th iteration. Based upon this estimation, the gain can be estimated in the next iteration.

The adaptation speed depends upon the coefficient μ. If it is too small, it may take a long time to converge. If it is too large, it may not converge at all. One way to choose the adaptation coefficient μ is in the following:

$$\mu = \frac{-\alpha}{\left|Y_{k,\max}\right|^2} \tag{10.54}$$

where $Y_{k,\max}$ is the maximum of the kth subcarrier. Substituting (10.54) into (10.52), we have

$$0 < \alpha < \frac{\left|Y_{k,\max}\right|^2}{\left|Y_k\right|^2} \tag{10.55}$$

The parameter α given in (10.55) controls the convergence behavior and a proper selection is necessary.

10.4.3 Examples

In the following examples, we concentrate on a single subcarrier k using QPSK. Assume the multipath channel is given by (10.33) and (10.34). From the training sequence, the initial gain G is then given here:

$$G_k = \frac{X_k}{Y_k} = \frac{1}{H_{kr} + jH_{ki}} = \frac{H_{kr} - jH_{ki}}{H_{kr}^2 + H_{ki}^2} \tag{10.56}$$

Later, assume the channel suffers gradual fading which is determined by the factor b in (10.33) and (10.34). Since b is changed, the initial value of G_k given by (10.56) is no longer valid. If it is not updated, the receiver performance will degrade.

Using (10.42) for adaptation, the error given by (10.38) can be used to determine how good the convergence is.

Example 10.7

Let $k = 24$, $b = 0.9$, $N = 64$, and $m = 3$, and then from (10.33) and (10.34) we can set some initial values given in (10.57a) to (10.57d):

$$X_{24} = 0.707(1 + j) \tag{10.57a}$$

$$H_{24} = 1.636 + 0.707\,j \tag{10.57b}$$

$$Y_{24} = 0.657 + 1.657\,j \tag{10.57c}$$

$$G_{24} = 0.515 - 0.223\,j \tag{10.57d}$$

If the channel does not change, the G_{24} given by (10.57d) can be used to achieve a perfect demodulation. Assume now that b drops to 0.8 and is maintained for at least 20 symbol periods. Since H_{24} is changed, the gain G_{24} is also changed. If the gain is not updated, an error always occurs and cannot be removed.

Assume now that the gain is updated according to (10.42). Depending upon the selection of α, there is a different convergence behavior. Figure 10.11 is a plot of the error magnitude $|\varepsilon_k|$ given in (10.38) against the iteration number. The plot starts at iteration 1 when b drops to 0.8.

Depending upon the adaptation and α, there are four different curves. If there is no adaptation, the error remains the same and can never be reduced. For both $\alpha = 0.1$ and 0.5, the adaptation converges. However, the convergence speed at $\alpha = 0.5$ is much faster than at $\alpha = 0.1$. For $\alpha = 0.1$, the error is close to 0 at the end of iteration. For $\alpha = 0.5$, the error is exactly 0 at iteration number 5. It has to be careful to continue increasing α. When α is increased to 1.1, the error continues increasing because the convergence condition given by (10.55) is not satisfied.

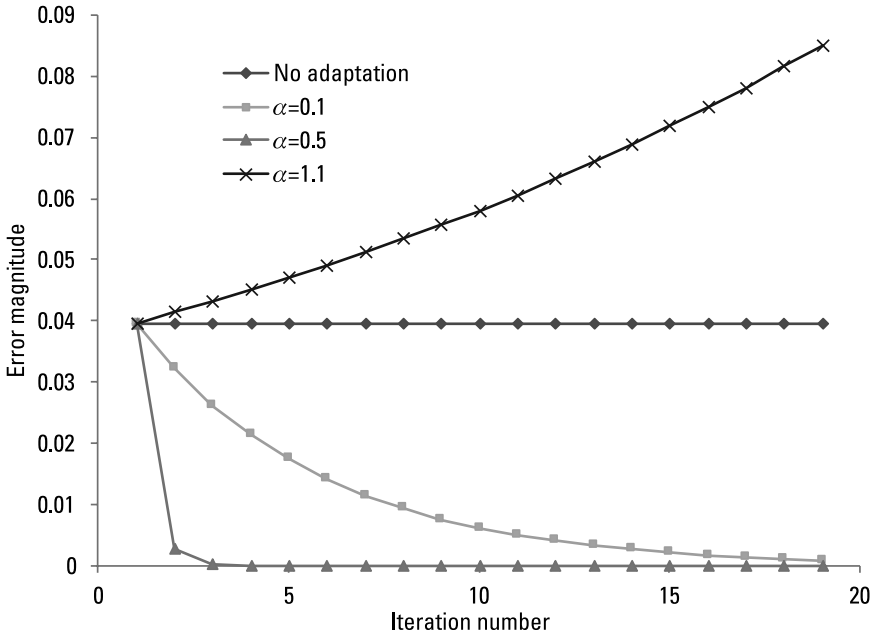


Figure 10.11 The demodulation error against the iteration number without fading.

Example 10.8

In this example, the fading factor b is assumed to drop from 0.9 to 0.8 in 20 steps. In other words, b drops according to the following formula:

$$b = 0.9 - 0.005n \tag{10.58}$$

where n is the iteration number. Besides the change of b values, all the other conditions follow the same as given in Example 10.6. At $b = 0.9$, all the initial parameter values are given in (10.57).

Figure 10.12 is a plot of the error magnitude $|\varepsilon_k|$ against the iteration number. There are four curves. One is without adaptation. The other three curves have $\alpha = 0.1$, 0.5, and 1.12. The error magnitude is the same at iteration number 1 for all. This is because at iteration number 0, the error magnitude is zero. Without adaptation, the error continues increasing with increasing iteration number. With adaptation, the error depends upon the choice of α. The error does not go to 0 because the channel frequency response changes due to the b variations at every iteration number. For $\alpha = 0.5$, the error is better than that for $\alpha = 0.2$. At $\alpha = 1.12$, the error is small initially and diverges eventually, because the convergence condition given by (10.55) is not satisfied.

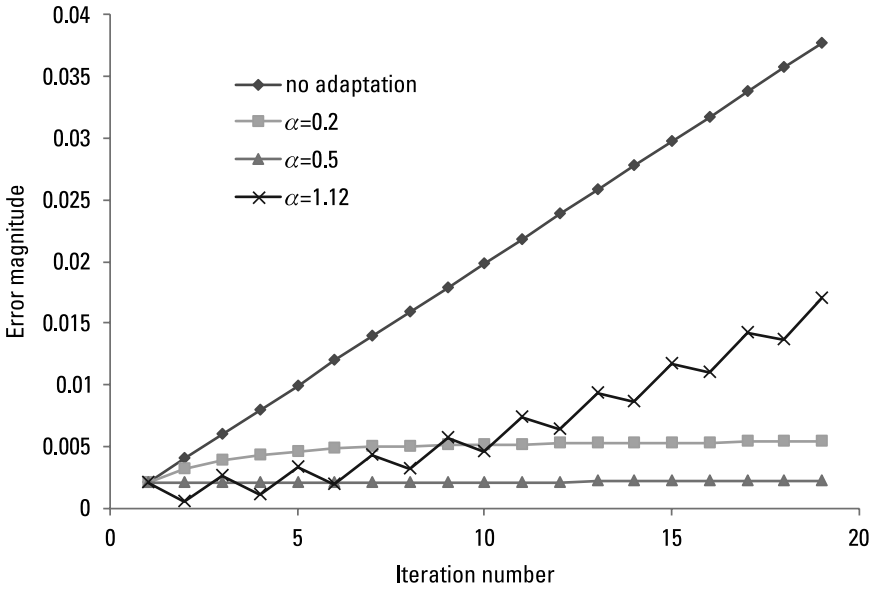


Figure 10.12 The error magnitude against iteration number with fading.

10.5 Summary

Two pilot patterns were used to estimate the channel. One is the block type and the other is the comb type. The pilot patterns are usually given in a time-frequency diagram.

For the block-type pattern, the pilot subcarriers are periodically inserted at all subcarriers. This type is used when the channel variation is slow. At the insertion symbol time, there are no data subcarriers. Therefore, the channel is accurately estimated at all subcarriers. Since the transmission of the pilot symbol is an overhead, the price paid is a reduction in the data rate.

For the comb-type pattern, the pilots are sent all the time but only a finite number of subcarriers are dedicated to be pilots. This type is used when the channel is fast varying. Even though the channel at the pilot subcarriers can be exactly determined, the channel at the data subcarriers has to be estimated through interpolation. Lagrange interpolation using polynomials is used. Depending upon the interpolation order, linear, parabolic, and cubic interpolation formulas were derived. The interpolation accuracy depends upon the sample delay, interpolation order, the number of pilot subcarriers, and random noise. Clearly, the interpolation accuracy increases with an increasing number of available pilot subcarriers. Since the number of pilot subcarriers is limited and there is uncertainty on the sample delay, the interpolation may not generate a good channel estimate.

The channel tracking provides an alternative to comb pilot pattern. The LMS algorithm uses a gradient search to adaptively adjust the channel response. The channel at every data subcarrier can be independently tracked and only one tap is necessary. The tracking depends upon the received subcarrier samples and the mean-square error between the information sequence and its estimate. As long as the adaptation coefficient is selected to meet the convergence condition, the channel estimation error is significantly reduced in comparison to without any tracking. Another advantage is adaptive channel tracking does not need pilot subcarriers, and the data rate will not additionally suffer.

The next chapter concentrates on the decoding process. Both hard decision decoding and soft decision decoding are discussed.

References

[1] Chiueh, T.-D., P. Y. Tsai, and I. W. Lai, *Baseband Receiver Design for Wireless MIMO-OFDM Communications*, New York: Wiley, 2012.

[2] Rinne, J., and M. Renfors, "Pilot Spacing in Orthogonal Frequency Division Multiplexing Systems on Practical Channels," *IEEE Transactions on Consumer Electronics*, Vol. 42, No. 4, 1996, pp. 959–962.

[3] Hsieh, M. H., and C. H. Wei, "Channel Estimation for OFDM Systems Based upon Comb-Type Pilot Arrangement in Frequency Selective Fading Channels," *IEEE Transactions on Consumer Electronics*, Vol. 44, No. 1, 1998, pp. 217–225.

[4] Eurp, L., F. M. Gardner, and R. A. Harris, "Interpolation in Digital Modems, Part II, Implementation and Performance," *IEEE Transactions on Communications*, Vol. 41, No. 6, 1993, pp. 998–1008.

[5] Proakis, J. G., *Digital Communications*, New York: McGraw-Hill, 1983.

[6] Widrow, B., et al., "Adaptive Antenna Systems," *IEEE Proceedings*, Vol. 55, No. 12, 1967, pp. 2143–2159.

11

Data Decoding

11.1 Introduction

In previous chapters, we have discussed signal detection, synchronization, channel estimation, and tracking. After the completion of those steps, each OFDM symbol should be correctly identified. The next step is the demodulation after channel removal to recover the in-phase and quadrature-phase samples for each subcarrier. These IQ samples are then demapped. Both hard decision decoding and soft decision decoding are covered.

For the hard decision decoding, both a brute-force approach and a simplified approach are discussed. The brute-force approach has to compute all the Euclidean distances against all the constellation points. For the simplified approach, each bit is determined individually based upon a simple metric for the I bits and Q bits. After demapping and subsequent deinterleaving, Viterbi decoding follows and the distance metric is based upon computing the Hamming distance. The Viterbi decoder output is then descrambled to recover the data bit sequence.

For the soft decision decoding, the IQ samples are used together with Viterbi decoding for demapping and error correction. The difference from the hard decision decoding is that the distance metric is based upon computing the Euclidean distance in each branch of the Viterbi decoder. A simplified log likelihood ratio (LLR) for soft decoding is also discussed. The transmitter and

receiver block diagrams are also slightly modified for the purpose of discussing soft decision decoding.

Both hard decision decoding and soft decision decoding are discussed in the following sections.

11.2 Demodulation

Figure 4.13 shows a typical receiver block diagram. After ADC and cyclic prefix removal, the FFT is performed to transform the samples from the time domain to the frequency domain. The channel effects can then be removed using the techniques discussed in Chapter 10. After the in-phase I samples and quadrature-phase Q samples are recovered, the next step is demodulation.

On demodulation, there are two different ways data can be decoded. One is hard decision decoding and the other is soft decision decoding. In the hard decision decoding, the I and Q samples are first demapped before passing to the convolutional decoder. The Hamming distance is used to find the right path through the trellis diagram to correct errors. In the soft decision decoding, the I and Q samples are not demapped initially. They are used to compute the Euclidean distance in the Viterbi decoder to find the right path for error correction. Since the I and Q samples have a range of possible values, more information is carried. This additional information provides the improved error correction capability of the Viterbi decoder.

Both hard decision and soft decision decoding are covered in the following sections.

11.3 Hard Decision Decoding

In the hard decision decoding, the first step is to transform the I and Q samples into data bits. There are two ways to do it. One way is to use the conventional Euclidean distance, which is time-consuming. The other is a simplified demapper without loss of accuracy [1]. Both approaches are discussed in the following sections.

11.3.1 Conventional Demapper

Let Y_k represent the kth received subcarrier of an OFDM symbol as given here:

$$Y_k = Y_{I,k} + jY_{Q,k} \tag{11.1}$$

where $Y_{I,k}$ and $Y_{Q,k}$ represent the in-phase and quadrature-phase sample of the kth subcarrier. Let H_k represent the channel estimation of the kth subcarrier. After removing the channel degradation, the restored subcarrier Z_k is then given according to (10.13):

$$Z_k = \frac{Y_k}{H_k} \tag{11.2}$$

Assume also that there are M constellation points for a given modulation. Let the ith constellation point be given as

$$A_i = \frac{A_{I,i} + jA_{Q,i}}{N}$$

where N is the normalization factor. The Euclidean distance between Z_k and A_i is then given here:

$$\begin{aligned} D_i &= \left|Z_k - A_i\right|^2 \\ &= \frac{\left[\left(NZ_{I,k} - A_{I,i}\right)^2 + \left(NZ_{Q,k} - A_{Q,i}\right)^2\right]}{N^2} \end{aligned} \tag{11.3}$$

If there are M constellation points, there are M such computations according to (11.3). Since N is a constant, the decision rule can be based upon the following modified D_i:

$$D_i = \left(NZ_{I,k} - A_{I,i}\right)^2 + \left(NZ_{Q,k} - A_{Q,i}\right)^2 \tag{11.4}$$

The lth constellation point A_l is selected if D_l is the minimum. Once A_l is determined, the corresponding bit sequence is obtained.

Example 11.1

Figure 11.1 shows the constellation of a QPSK modulation. The normalization factor is $N = \sqrt{2}$. The four constellation points are then $A_1 = (1 + j)/\sqrt{2}$, $A_2 = (1 - j)/\sqrt{2}$, $A_3 = (-1 + j)/\sqrt{2}$, and $A_4 = (-1 - j)/\sqrt{2}$. The coordinates of the constellation points are plotted as NA_i. Assume that the received kth subcarrier after removing the channel frequency response is $Z_k = (0.8/\sqrt{2})(1 + j)$. From (11.4), the decision metric is then the distance from $0.8 + 0.8j$ to $1 + j$, $1 - j$, $-1 + j$, and $-1 - j$. They are given by $D_1 = 0.08$, $D_2 = 3.28$, $D_3 = 3.28$,

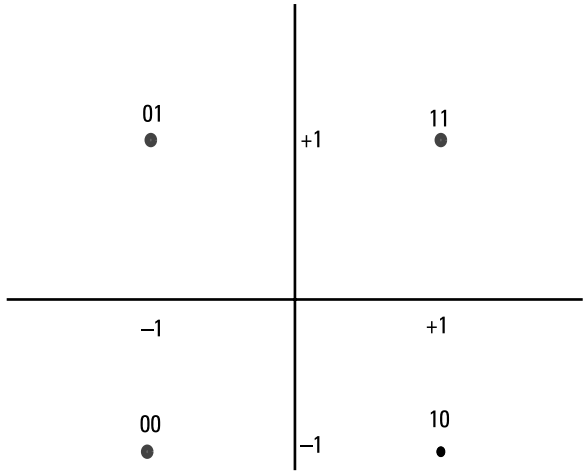


Figure 11.1 QPSK constellation.

and $D_4 = 6.48$. Since D_1 is the minimum, the constellation point selected is A_1. Based upon A_1, the data bit sequence is mapped to be 11.

For QPSK, only four Euclidean distances need to be computed. For 64-QAM, the number of distance computations jumps to 64 for each subcarrier. Therefore, it is necessary to simplify the computations.

11.3.2 Simplified Demapper

The approach given in the previous section is to first find the best constellation point. Based upon this constellation point, all the sequence bits are obtained at the same time. In the simplified approach [1], each bit is decoded separately. This is possible due to the way the bits sequence is assigned to the constellation point.

To simplify the presentation, let $U_{I,k}$ and $U_{Q,k}$ be defined here:

$$U_{I,k} = NZ_{I,k} \tag{11.5}$$

$$U_{Q,k} = NZ_{Q,k} \tag{11.6}$$

Substituting (11.5) and (11.6) into (11.4), we have

$$D_i = \left(U_{I,k} - A_{I,i}\right)^2 + \left(U_{Q,k} - A_{Q,i}\right)^2 \tag{11.7}$$

To further simplify the presentation, the index i is dropped and we have

$$\begin{aligned} D &= \left(U_{I,k} - A_I\right)^2 + \left(U_{Q,k} - A_Q\right)^2 \\ &= D_I + D_Q \end{aligned} \tag{11.8}$$

where D_I and D_Q are given here:

$$D_I = \left(U_{I,k} - A_I\right)^2 \tag{11.9}$$

$$D_Q = \left(U_{Q,k} - A_Q\right)^2 \tag{11.10}$$

Assume that each constellation point is mapped to $2m$ bits. As will become clear later, the first m bits are in-phase bits, while the second m bits are quadrature-phase bits. Each bit is either $b_{I,i}$ or $b_{Q,i}$ where i ranges from 1 to m. The in-phase bits are determined by D_I while the quadrature-phase bits are determined by D_Q.

Each bit is either 0 or 1 and has its own decision region. For the in-phase bits, they are either $R^0_{I,i}$ or $R^1_{I,i}$ where i is the bit number. For the quadrature-phase bits, they are either $R^0_{Q,i}$ or $R^1_{Q,i}$.

11.3.2.1 Simplified QPSK Demodulation

For the QPSK, each constellation point has 2 bits and $m = 1$. The 2-bit sequence is represented by $b_{I,1}b_{Q,1}$. As can be seen from Figure 11.1, the decision region is to the left of the y-axis if the first bit is 0 and to the right of the y-axis if the first bit is 1. In other words, $R^0_{I,1}$ is for $U_{I,k} < 0$ while $R^1_{I,1}$ is for $U_{I,k} > 0$. Since the decision boundary is the y-axis, the first bit is the in-phase bit and is determined by D_I.

For the second bit, the decision region is the x-axis. The decision region is above the x-axis if the second bit is 1 and below the x-axis if the second bit is 0. In other words, $R^0_{Q,1}$ is for $U_{Q,k} < 0$ while $R^1_{Q,1}$ is for $U_{I,k} > 0$. Since the decision region is the x-axis, the second bit is the quadrature-phase bit and is completely determined by D_Q.

Using D_I, the first bit is 1 if the following is true:

$$\left(U_{I,k} - 1\right)^2 < \left(U_{I,k} + 1\right)^2 \tag{11.11}$$

From (11.11), we have $U_{I,k} > 0$ if the first bit is 1 and $U_{I,k} < 0$ if the first bit is 0. This quantitatively justifies the previous argument.

Using D_Q, the second bit is one if the following is true:

$$\left(U_{Q,k}-1\right)^2 < \left(U_{Q,k}+1\right)^2 \tag{11.12}$$

From (11.12), we similarly have $U_{Q,k} > 0$ if the first bit is 1 and $U_{Q,k} < 0$ if the first bit is 0.

In summary, the decision rule is given here:

$$b_{I,1} = \begin{cases} 1 & \text{if} \quad U_{I,k} > 0 \\ 0 & \text{if} \quad U_{I,k} < 0 \end{cases} \tag{11.13a}$$

$$b_{Q,1} = \begin{cases} 1 & \text{if} \quad U_{Q,k} > 0 \\ 0 & \text{if} \quad U_{Q,k} < 0 \end{cases} \tag{11.13b}$$

Example 11.2

In Example 11.1, the received kth subcarrier is $Z_k = (0.8/\sqrt{2})(1+j)$. Multiplying by the normalization factor $\sqrt{2}$, we have $U_k = (0.8)(1+j)$. The in-phase and quadrature-phase components are then $U_{I,k} = 0.8$ and $U_{Q,k} = 0.8$. From (11.13), the bits are decoded as $b_{I,1} = 1$ and $b_{Q,1} = 1$. The sequence of 2 bits is then 11.

Equation (11.13) shows that the computation of Euclidean distance is completely removed. A simple comparison of the in-phase and quadrature-phase component against 0 determines the baud bits.

11.3.2.2 Simplified 16-QAM Demodulation

For 16-QAM, each constellation point has 4 bits. The first two bits are in-phase bits and the last two bits are quadrature-phase bits. The bit sequence is given by $b_{I,1}b_{I,2}b_{Q,1}b_{Q,2}$. Figure 11.2 shows the partitions of the first in-phase bit $b_{I,1}$. As can be seen, the partition boundary is the y-axis. The partition $R^0_{I,1}$ is to the left of the y axis while $R^1_{I,1}$ is to the right of the y-axis. Therefore, the shortest distance is determined by D_I.

In each partition, there are two constellation points on the x-axis. The first step is to choose the constellation point in each partition. In the partition $R^1_{I,1}$, the two constellation points are 3 and 1. The constellation point 3 is chosen if the following is true:

$$\left(U_{I,k}-3\right)^2 < \left(U_{I,k}-1\right)^2 \tag{11.14}$$

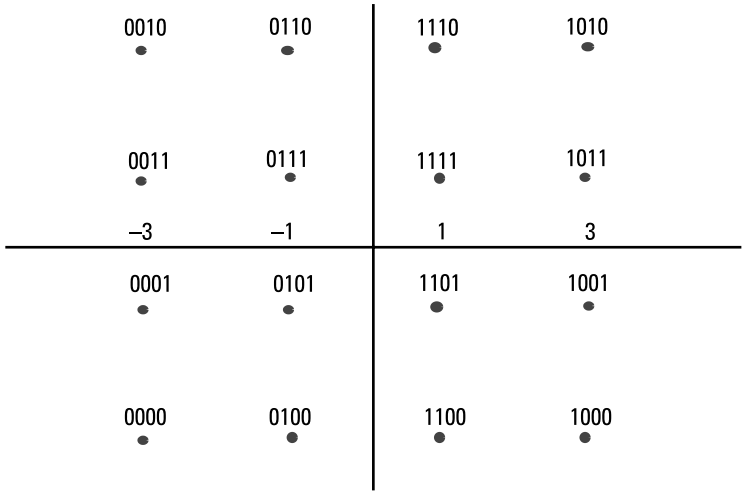


Figure 11.2 The partitions of the first in phase bit of 16-QAM.

After some simple algebra, (11.14) leads to the condition $U_{I,k} > 2$. Similarly, constellation point 1 is chosen if $U_{I,k} < 2$.

In the partition $R^0_{I,1}$, there are also two constellation points −1 and −3. The constellation point −3 is chosen if the following is true:

$$\left(U_{I,k}+3\right)^2 < \left(U_{I,k}+1\right)^2 \tag{11.15}$$

Equation (11.15) leads to the condition $U_{I,k} < -2$. Similarly, the condition becomes $U_{I,k} > -2$ if the constellation point −1 is chosen.

For $U_{I,k} > 2$, the constellation point in partition $R^1_{I,1}$ is 3 and the closest constellation point in partition $R^0_{I,1}$ is −1. Therefore, $b_{I,1} = 1$ if the following is true:

$$\left(U_{I,k}-3\right)^2 < \left(U_{I,k}+1\right)^2 \text{ or } U_{I,k} > 1 \tag{11.16}$$

Following the similar logic, the following are obtained:

$$U_{I,k} < -2 \text{ and } b_{I,1} = 0 \text{ if } \left(U_{I,k}+3\right)^2 < \left(U_{I,k}-1\right)^2 \text{ or } U_{I,k} < -1 \tag{11.17}$$

$$U_{I,k} < 2 \text{ and } b_{I,1} = 1 \text{ if } \left(U_{I,k}-1\right)^2 < \left(U_{I,k}+1\right)^2 \text{ or } U_{I,k} > 0 \tag{11.18}$$

$$U_{I,k} > -2 \text{ and } b_{I,1} = 0 \text{ if } \left(U_{I,k}+1\right)^2 < \left(U_{I,k}-1\right)^2 \text{ or } U_{I,k} < 0 \tag{11.19}$$

From (11.16) to (11.19), $b_{I,1}$ is decoded based upon the following criteria:

$$U_{I,k} - 1 \qquad U_{I,k} > 2 \tag{11.20a}$$

$$U_{I,k} \qquad \left|U_{I,k}\right| < 2 \tag{11.20b}$$

$$U_{I,k} + 1 \qquad U_{I,k} < -2 \tag{11.20c}$$

For each criterion, the first term determines whether the bit is 1 or 0. For example, the first criterion is to set $b_{I,1} = 1$ for $U_{I,k} - 1 > 0$ and $b_{I,1} = 0$ for $U_{I,k} - 1 < 0$ under the condition of $U_{I,k} > 2$.

Figure 11.3 shows the partitions of the second in-phase bit $b_{I,2}$. The two vertical lines y_1 and y_2 generate three partitions. The partition $R^0_{I,2}$ is to the left of y_1 and to the right of y_2 while the partition $R^1_{I,2}$ is between y_1 and y_2. The constellation point is 3 if the partition $R^0_{I,2}$ is to the right of y_2 and −3 if the partition $R^0_{I,2}$ is to the left of y_1.

For constellation $R^1_{I,2}$, there are two possible constellation points −1 and 1 on the x-axis. To choose 1 as the constellation point, the following inequality must be satisfied:

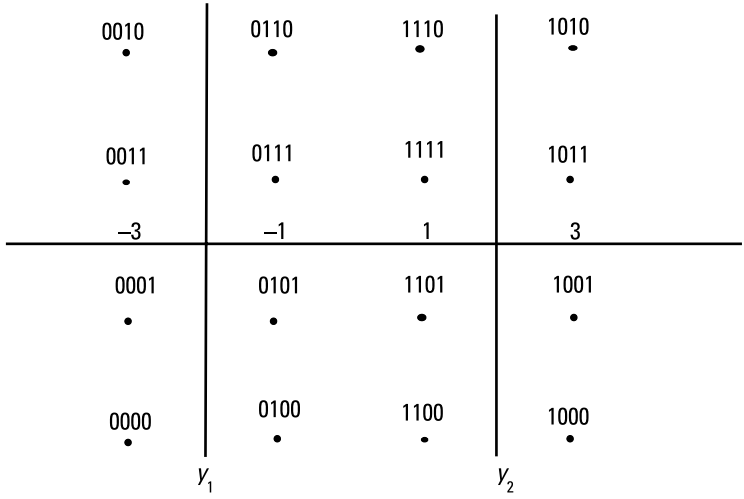


Figure 11.3 The partitions of the second in-phase bit of 16-QAM.

$$\left(U_{I,k}-1\right)^2 < \left(U_{I,k}+1\right)^2 \text{ or } U_{I,k} > 0 \tag{11.21}$$

Similarly, to choose −1 as the constellation point, the condition becomes $U_{I,k} < 0$.

Based upon the same logic as before, $b_{I,2} = 0$ if the following inequalities are true:

$$\left(U_{I,k}-3\right)^2 < \left(U_{I,k}-1\right)^2 \text{ or } -U_{I,k}+2 < 0 \text{ for } U_{I,k} > 0 \tag{11.22a}$$

$$\left(U_{I,k}+3\right)^2 < \left(U_{I,k}+1\right)^2 \text{ or } U_{I,k}+2 < 0 \text{ for } U_{I,k} < 0 \tag{11.22b}$$

Similarly, $b_{I,2} = 1$ if the following inequalities are true:

$$\left(U_{I,k}-1\right)^2 < \left(U_{I,k}-3\right)^2 \text{ or } -U_{I,k}+2 > 0 \text{ for } U_{I,k} > 0 \tag{11.23a}$$

$$\left(U_{I,k}+1\right)^2 < \left(U_{I,k}+3\right)^2 \text{ or } U_{I,k}+2 > 0 \text{ for } U_{I,k} < 0 \tag{11.23b}$$

Equations (11.22) and (11.23) can be combined to have the simple criterion $-|U_{I,k}| + 2$. The interpretation is then $b_{I,2} = 1$ if $-|U_{I,k}| + 2 > 0$ and $b_{I,2} = 0$ if $-|U_{I,k}| + 2 < 0$.

Equation (11.20) can be further simplified. For criterion (1), $U_{I,k} - 1$ is always greater than 0 if $U_{I,k} > 2$. Therefore, the criterion is similar to $b_{I,1} = 1$ for $U_{I,k} > 0$. Similarly, for criterion (3), $U_{I,k} + 1$ is always less than zero if $U_{I,k} < -2$. The criterion is then the same as $b_{I,1} = 0$ for $U_{I,k} < 0$. Based upon these arguments, (11.20) and (11.23) can be combined to give the following criteria:

$$b_{I,1}\colon\ U_{I,k} \tag{11.24a}$$

$$b_{I,2}\colon\ -\left|U_{I,k}\right| + 2 \tag{11.24b}$$

Similarly, the quadrature-phase bits have the following criteria:

$$b_{Q,1}\colon\ U_{Q,k} \tag{11.25a}$$

$$b_{Q,2}\colon\ -\left|U_{Q,k}\right| + 2 \tag{11.25b}$$

Equations (11.24) and (11.25) are significantly simplified in comparison with the computations of 16 Euclidean distances to decode one subcarrier.

Example 11.3

Assume 16-QAM and the received kth subcarrier is $Z_k = (-0.5 + 3j)/\sqrt{10}$. The normalization factor is $\sqrt{10}$. After multiplying by $\sqrt{10}$, the received $Z_k = (-0.5 + 3j)$. Using the brute-force approach, it is necessary to compute 16 Euclidean distances. Figure 11.4 is a plot of these 16 Euclidean distances against the decimal equivalent of the bit sequence. The minimum is 0.25 corresponding to the decimal number 6. The bit sequence is then 0110. The two in-phase bits are 01 and the two quadrature-phase bits are 10.

Equation (11.24) and (11.25) can be used to quickly get the same answer. The in-phase and quadrature-phase components are $U_{I,k} = -0.5$ and $U_{Q,k} = 3$. To decode the in-phase bits, we have from (11.24), $U_{I,k} < 0$ while $-|U_{I,k}| + 2 = 1.5 > 0$. The two in-phase bits are then 01. To decode the two quadrature-phase bits, we have from (11.25), $U_{Q,k} > 0$ while $-|U_{Q,k}| + 2 = -1 < 0$. The two quadrature-phase bits are then 10. The whole bit sequence is then 0110, which is the same as that from the brute-force computation.

11.3.2.3 Simplified 64-QAM Demodulation

Figure 2.6 shows the constellation for 64-QAM. The constellation points are 1, 3, 5, and 7 on the positive x-axis and −1, −3, −5, and −7 on the negative x-axis. The same is true on the y-axis. Each constellation point has 6 bits. There are 3 in-phase bits and 3 quadrature-phase bits. The bit sequence can

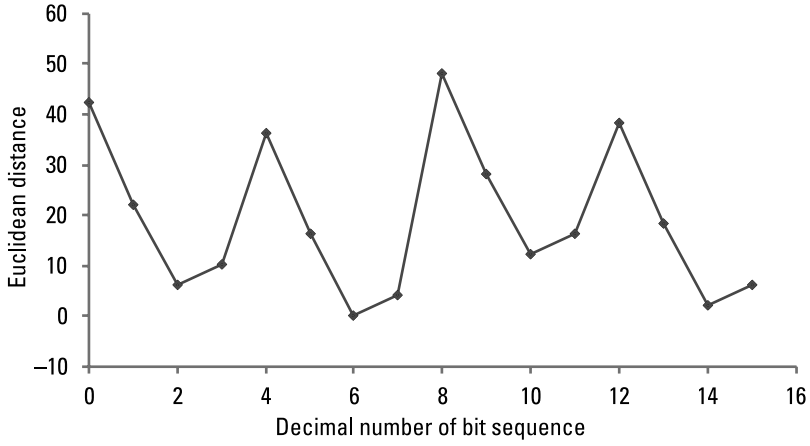


Figure 11.4 Plot of Euclidean distances for QAM-16 and $U_k = -0.5 + 3j$.

be written as $b_{I,1}b_{I,2}b_{I,3}b_{Q,1}b_{Q,2}b_{Q,3}$. Following the same procedure as given in the previous two sections for QPSK and 16-QAM, the results are given here:

$$
\begin{aligned}
b_{I,1}\text{:}\quad & (1)\ U_{I,k} && |U_{I,k}| \le 2 \\
& (2)\ U_{I,k} - 1 && 2 < U_{I,k} \le 4 \\
& (3)\ U_{I,k} - 2 && 4 < U_{I,k} \le 6 \\
& (4)\ U_{I,k} - 3 && U_{I,k} > 6 \\
& (5)\ U_{I,k} + 1 && -4 \le U_{I,k} < -2 \\
& (6)\ U_{I,k} + 2 && -6 \le U_{I,k} < -4 \\
& (7)\ U_{I,k} + 3 && U_{I,k} < -6
\end{aligned}
\tag{11.26}
$$

$$
\begin{aligned}
b_{I,2}\text{:}\quad & (1)\ -|U_{I,k}| + 3 && |U_{I,k}| \le 2 \\
& (2)\ 4 - |U_{I,k}| && 2 < |U_{I,k}| \le 6 \\
& (3)\ -|U_{I,k}| + 5 && |U_{I,k}| > 6
\end{aligned}
\tag{11.27}
$$

$$
\begin{aligned}
b_{I,3}\text{:}\quad & (1)\ |U_{I,k}| - 2 && |U_{I,k}| \le 4 \\
& (2)\ -|U_{I,k}| + 6 && |U_{I,k}| > 4
\end{aligned}
\tag{11.28}
$$

Equation (11.26) can be further simplified. Criteria (2), (3), and (4) are always positive under the specified positive values of $U_{I,k}$. This is the same as $b_{I,1} = 1$ for $U_{I,k} > 0$. Similarly, criteria (5), (6), and (7) are always negative under the specified negative values of $U_{I,k}$. Again, this is the same as $b_{I,1} = 0$ for $U_{I,k} < 0$. Criterion (1) also suggests $b_{I,1} = 1$ for $U_{I,k} > 0$ and $b_{I,1} = 0$ for $U_{I,k} < 0$. In other words, $b_{I,1}$ is completely determined by $U_{I,k}$.

Equation (11.27) can also be further simplified. Criterion (1) is always positive for $|U_{I,k}| \le 2$. Criterion (3) always generates negative values for $|U_{I,k}| > 6$. Both are either always positive or always negative under the second criterion. In other words, $b_{I,2}$ is completely determined by $4 - |U_{I,k}|$ without any conditions.

Equation (11.28) shows that there are two criteria to specify $b_{I,3}$ under two separate conditions. These two criteria can be replaced by a single criterion $-||U_{I,k}| - 4| + 2$ without any attached conditions [1]. By tracing through all possible values of $U_{I,k}$, it can be seen that this single criterion is equivalent to (11.28).

From the above arguments, the further simplified criteria to determine the in-phase bits are summarized here [1]:

$$\begin{aligned} b_{I,1}&: \; U_{I,k} \\ b_{I,2}&: \; 4-\left|U_{I,k}\right| \\ b_{I,3}&: \; -\left|\left|U_{I,k}\right|-4\right|+2 \end{aligned} \tag{11.29}$$

Similarly, the quadrature-phase bits are determined by the following criteria:

$$\begin{aligned} b_{Q,1}&: \; U_{Q,k} \\ b_{Q,2}&: \; 4-\left|U_{Q,k}\right| \\ b_{Q,3}&: \; -\left|\left|U_{Q,k}\right|-4\right|+2 \end{aligned} \tag{11.30}$$

Example 11.4

Assume the modulation is 64-QAM and the received $Z_k = (1.5 + j)/\sqrt{42}$. The normalization factor is $\sqrt{42}$. Multiplying by $\sqrt{42}$, we have $U_k = 1.5 + j$. Figure 11.5 is a plot of Euclidean distance against all 64 possible bit sequences. The minimum is 54 and the corresponding bit sequence is 110110.

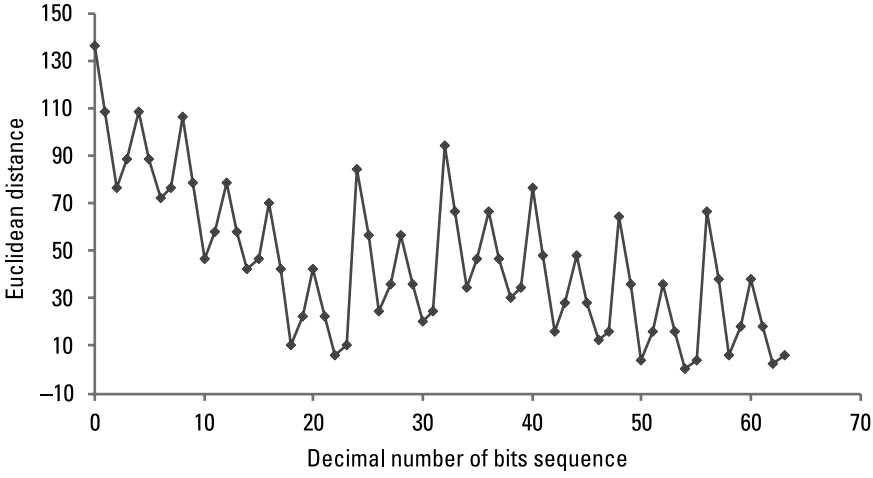


Figure 11.5 Plot of Euclidean distance for 64-QAM and $U_k = 1.5 + j$.

Using the simplified approach, the in-phase and quadrature-phase components are $U_{I,k} = 1.5$ and $U_{Q,k} = 1$, respectively. The three criteria for the in-phase bits from (11.29) are 1.5, 2.5, and −0.5, respectively. The in-phase bit sequence is then 110. For the quadrature-phase bits, the three criteria generate 1, 3, and −1. The quadrature phase bit sequence is then also 110. The complete bit sequence is then 110110. The simplified approach generates the same answer but with much less computation.

11.3.3 Deinterleaver and Viterbi Decoding

After the I and Q samples are demapped, the bit sequence must first be deinterleaved. The deinterleaver process is discussed in Chapter 7. The output bit sequence from the deinterleaver is then fed into the Viterbi decoder. The details of the Viterbi decoder are also discussed in Chapter 7.

The Viterbi decoder in Chapter 7 is illustrated using Hamming distance as the distance metric. The received baud is compared with the baud from each trellis branch to get the distance. For a 1/2 convolutional decoder, each baud has 2 bits. Table 7.3 lists the Hamming distance for all the possible bauds. In essence, this is hard decision decoding.

11.3.4 Descrambler

The last step in the decoding process is to pass the corrected bit sequence from the Viterbi decoder to a descrambler to recover the data bits. The scrambler in the transmitter is necessary for data security. The scrambler using the shift register sequence is discussed in Chapter 5.

Since the receiver knows exactly how the data is scrambled in the transmitter, the descrambling process is pretty easy and is illustrated in Figure 5.7.

11.4 Soft Decision Decoding

For soft decision decoding, two slightly modified transmitter and receiver block diagrams are given first. After understanding the overall architectures, the soft decision processes then follows.

11.4.1 Transmitter and Receiver Block Diagram

For the soft decision decoding, a typical transmitter block diagram is shown in Figure 11.6. In comparing with Figure 4.1, the only change is that the error

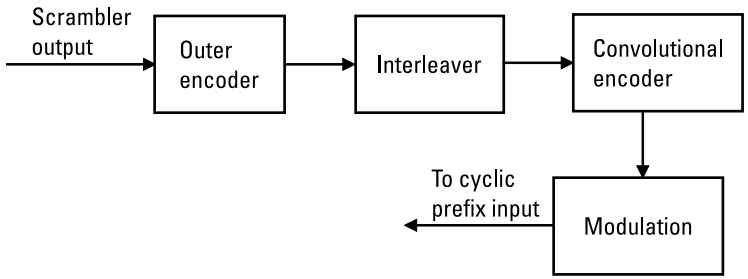


Figure 11.6 A typical soft decision encoder block diagram.

correction encoder and interleaver are replaced by the outer encoder, interleaver, and convolutional encoder. The outer encoder could be a Reed-Solomon encoder, which can be applied to reduce block errors [2, 3]. However, it can be another error correction encoder depending on the specific application. For example, the outer encoder could be a quaternary convolutional encoder [4]. The output from the outer encoder then passes through an interleaver and then a convolutional encoder before getting into the modulation process. After the modulator, it follows what is given in Figure 4.1.

A typical receiver block diagram is shown in Figure 11.7. In comparing with Figure 4.13, the demodulation, deinterleaver, and error correction decoder are replaced by Viterbi decoder, deinterleaver, and outer decoder. After the channel effects are removed, the I and Q samples have a range of values and include valuable information. They will not be demapped immediately. Instead, they will be used in combination with a Viterbi decoder to generate the bit sequence.

The output from the Viterbi decoder then goes through deinterleaver and outer decoder before getting into the descrambler to recover the data bit sequence.

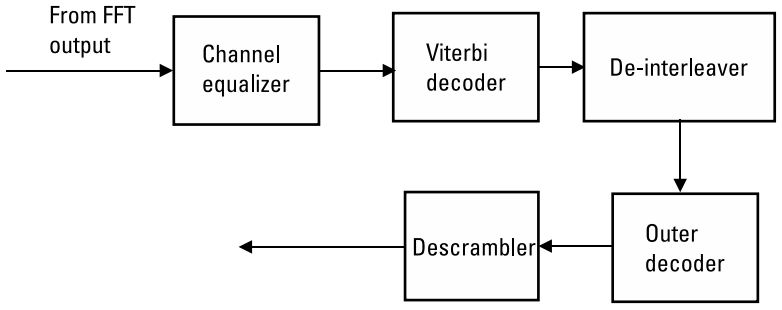


Figure 11.7 A typical soft decision decoder block diagram.

11.4.2 Soft Decoding Using the Euclidean Distance

Using the soft decision in the Viterbi decoding, the only difference from the hard decision is the distance metric. As was mentioned before, the metric used is the Euclidean distance. For simplicity in the illustration, assume that QPSK is used for the modulation. For each branch in the Viterbi trellis diagram, let the received sample be given as $I + jQ$. Each branch output baud is 00, 01, 11, or 10. Their equivalent values in the constellation are given in Table 2.2. Let each branch baud be represented as $B_I + jB_Q$. Then the Euclidean distance is simply given here:

$$D = \sqrt{\left(I - B_I\right)^2 + \left(Q - B_Q\right)^2} \tag{11.31a}$$

To save computation time, D can also be computed as follows:

$$D = \left|I - B_I\right| + \left|Q - B_Q\right| \tag{11.31b}$$

In each branch, the Hamming distance for hard decision decoding is then replaced by D given in (11.31).

Following the same process as in hard decision decoding, the path score is accumulated for each additional branch. At the last branch, the path with the minimum accumulated distance is the optimum path. The input bit sequence is then determined from this optimum path.

11.4.3 Soft Decoding Using LLR

The simplified demodulation discussed in Section 11.3.2 simplifies the hard decision demodulation. However, the soft bit values defined are actually derived from LLR, which is very similar to what has been presented in Section 8.4. Assume that the constellation point is a in the decision region $R^1_{I,j}$ and b in the decision region $R^0_{I,j}$ for the jth I bit. Then point a is selected if the following LLR is true

$$p\left(\frac{y}{R^1_{I,j}}\right) > p\left(\frac{y}{R^0_{I,j}}\right) \tag{11.32}$$

where y is the received in-phase sample. Assuming that y is Gaussian distributed with standard deviation σ, (11.32) then becomes

$$\frac{1}{\sqrt{2\pi\sigma}} e^{\left[-\frac{(y-a)^2}{2\sigma^2}\right]} > \frac{1}{\sqrt{2\pi\sigma}} e^{\left[-\frac{(y-b)^2}{2\sigma^2}\right]} \tag{11.33}$$

Removing the common term and taking logarithm on both sides of (10.33), we have

$$(y-a)^2 < (y-b)^2 \quad \text{if } a \text{ is selected} \tag{11.34}$$

Comparing the Euclidean distance given in (11.34) is the basis for the discussion of the hard decision decoding presented in Section 11.3. However, it is actually derived from the LLR.

Based upon the LLR, the soft bit values are defined for QPSK in (11.13), for 16-QAM in (11.24) and (11.25) and for 64-QAM in (11.29) and (11.30). It has been shown these soft bit values can be applied for soft Viterbi decoding successfully [1].

11.4.4 Further Decoding Process

In reference to Figure 11.7, further processes after soft Viterbi decoder are shown. The corrected data bit sequence is then deinterleaved and further decoded through the outer decoder. The output bit sequence from the output decoder is then descrambled to recover the original data bits.

11.5 Summary

The data demodulation starts after the receiver has completed the process of IQ recovery, synchronization and frequency offset estimation. There are two different ways for data demodulation. One is hard decision decoding and the other is soft decision decoding.

On the hard decision decoding, the received I and Q samples have to be demapped first. There are two different approaches. One is the brute-force approach by finding the minimum Euclidean distance against all the constellation points. The other is a simplified approach by decoding a single bit each time based upon it is I bits or Q bits. The decision rules for QPSK, 16-QAM, and 64-QAM are derived. This greatly simplifies the required computations and speeds up the demodulation process. After the demapping is complete, the transmitted bit stream is recovered after additionally going through the process of deinterleaving, FEC decoding, and descrambling.

On the soft decision decoding, the transmitter and receiver block diagrams are slightly modified and discussed. The soft Viterbi decoding is then described. The I and Q samples are demapped in combination with the Viterbi decoding. The difference from the hard Viterbi decoding is the computation of distance metric. The soft decision uses Euclidean distance while the hard decision uses the Hamming distance.

The soft bit values given in the hard decision for the simplified demodulation are derived from the LLR by comparing two different decision regions for bit 1 and bit 0. These LLR values can be applied successfully to the soft Viterbi decoding. Following Figure 11.7, the input data bit sequence can eventually be recovered.

The next chapter presents the simulation results of the OFDM performance in a multipath channel.

References

[1] Tosato, F., and P. Bisaglia, "Simplified Soft-Output Demapper for Binary Interleaved COFDM with Application to HIPPERLAN/2," *Proceedings of IEEE International Conference on Communications*, 2002, pp. 664–668.

[2] Lin, S., and D. Costello, *Error Control Coding*, Upper Saddle River, NJ: Prentice Hall, 1983.

[3] Peterson, W. W., and E. J. Weldon, *Error Correcting Codes*, Cambridge, MA: MIT Press, 1972.

[4] Singh, M., and I. Wassell, "Comparison Between Soft and Hard Decision Decoding Using Quaternary Convolutional Encoders and the Decomposed CPM Model," *IEEE VTS 53rd Vehicular Technology Conference*, Vol. 2, 2001, pp. 1347–1351.

12

Simulation Study of a Multipath Channel on OFDM

12.1 Introduction

Up to this point, all the major components in the transmitter and the receiver have been presented. They were provided in Figures 4.1 and 4.13 for the transmitter and receiver block diagrams. The complete system is then simulated to find out the effectiveness of the OFDM to combat the impact of a multipath channel and random Gaussian noise.

Both the transmitter and receiver software were written in C language running on a laptop. The Ubuntu Linux operating system was used to generate the results. However, the commands and operations are not different from the Linux operating system.

The transmitter waveform follows exactly what was specified in the IEEE 802.11a. The receiver algorithms have been discussed in the previous chapters. Another goal for the simulation is to justify these algorithms. On the Viterbi decoding, only hard decision decoding is used. The unique algorithm is the maximum likelihood detection based upon normalized cross-correlation for signal acquisition. Other algorithms are the block pilot pattern inserted at a given period for channel estimation, the convolutional codes including

punctured convolutional codes for error corrections, and the LMS algorithm using the method of steep descent for channel tracking.

In the following, a multipath channel model is presented first. Simulation examples for OFDM performances over both an AWGN and a multipath channel are then given.

12.2 Characterization of Multipath Channel

From (8.1), the transmitted signal $x(t)$ with a carrier frequency of f_c is given here:

$$x(t) = \mathrm{Re}\left(s(t)e^{j2\pi f_c t}\right) \tag{12.1}$$

where $s(t)$ is the source signal. To characterize the multipath channel, assume that there are $L(t)$ paths. The number of paths at any given time is time-dependent. This means that, at any time, the number of signals the receiver receives is not a constant. Corresponding to each path, assume the delay is $\tau_n(t)$, the Doppler shift is $\in_n(t)$, and the attenuator factor is $\alpha_n(t)$. The received signal $r(t)$ is then given by

$$r(t) = \sum_{n=1}^{L(t)} \alpha_n(t)x\left(t-\tau_n(t)\right)e^{j2\pi t\in_n(t)} \tag{12.2}$$

The Doppler shift, $\in_n(t)$, from each path is also time-dependent. Assuming that the angle between the direction of motion and the signal propagation path is $\theta_n(t)$, then the Doppler shift can be written as here:

$$\in_n(t) = vf_c\cos\left(\theta_n(t)\right)/c \tag{12.3}$$

where c is the light velocity and v is the velocity of motion. Substituting (12.1) into (12.2), the following equation results:

$$r(t) = \mathrm{Re}\left\{\left[\sum_{n=1}^{L(t)} \alpha_n(t)s\left(t-\tau_n(t)\right)e^{-j2\pi f_c\tau_n(t)}e^{+j2\pi\in_n(t)t}\right]e^{j2\pi f_c t}\right\} \tag{12.4}$$

The quantity inside the bracket is the equivalent lowpass signal $z(t)$ and is given by

$$z(t) = \sum_{n=1}^{L(t)} \alpha_n(t)s\left(t-\tau_n(t)\right)e^{-j2\pi(f_c\tau_n(t)-\in_n(t)t)} \tag{12.5}$$

Then the phase term in (12.5) can be combined to give

$$\varphi_n(t) = -2\pi\left(f_c\tau_n(t) - \epsilon_n(t)t\right) \tag{12.6}$$

Substituting (12.6) into (12.5), the following results [1]:

$$z(t) = \sum_{n=1}^{L(t)} \alpha_n(t) e^{j\varphi_n(t)} s\left(t - \tau_n(t)\right) \tag{12.7}$$

Equation (12.7) shows that there is an amplitude perturbation $\alpha_n(t)$ and a phase perturbation $\varphi_n(t)$ corresponding to each path. The phase term $\varphi_n(t)$ can cause the signal fading. Depending upon the magnitude of phase term $\varphi_n(t)$, the signals from different paths may add constructively or destructively. If it is the constructive addition, the combined signal strength increases. However, if it is the destructive addition, the combined signal strength decreases. These signal variations are called signal fading.

Equation (12.7) can be further simplified by defining the weighting factor here:

$$w_n(t) = \alpha_n(t) e^{j\varphi_n(t)} \tag{12.8}$$

Substituting (12.8) into (12.7), we have

$$z(t) = \sum_{n=1}^{L(t)} w_n(t) s\left(t - \tau_n(t)\right) \tag{12.9}$$

From (12.9), the received lowpass signal $z(t)$ is the summation of the delayed version of the original signal $s(t)$ multiplied by a time-dependent weighting factor $w_n(t)$. This weighting factor is, in general, complex and has both amplitude and phase variations.

From (12.9), the equivalent channel impulse response $h(\tau, t)$ at time t is given here:

$$h(\tau,t) = \sum_{n=1}^{L(t)} w_n(t)\delta\left(\tau - \tau_n(t)\right) \tag{12.10}$$

The received signal $z(t)$ is the output of passing the source signal $s(t)$ through the channel characterized by $h(\tau, t)$. Equation (12.10) is justified by the following convolutional integral:

$$\begin{aligned} z(t) &= \int h(\tau,t)s(t-\tau)\,d\tau \\ &= \int \sum_{n=1}^{L(t)} w_n(t)\delta\left(\tau-\tau_n(t)\right)s(t-\tau)d\tau \\ &= \sum_{n=1}^{L(t)} w_n(t)s\left(t-\tau_n(t)\right) \end{aligned} \tag{12.11}$$

From the derivation of (12.11), the channel impulse response given by (12.10) is justified. Equation (12.9) is used as the basis for the simulation of a multipath channel.

12.3 Computation of SNR

In a multipath channel, only one is the direct path and the others are due to reflections. For the direct path, the delay is $\tau_1(t)$. Equation (12.11) can then be rewritten as

$$z(t) = w_1(t)s\left(t-\tau_1(t)\right) + \sum_{n=2}^{L(t)} w_n(t)s\left(t-\tau_n(t)\right) \tag{12.12}$$

In the receiver, the received signal after adding the noise $n(t)$ becomes

$$y(t) = z(t) + n(t) \tag{12.13a}$$

Taking the discrete Fourier transform of (12.13a), we have the following:

$$Y_k = Z_k + N_k \tag{12.13b}$$

The k_jth subcarrier of the mth symbol in the frequency domain is denoted as Z_{mk_j}. Assume that the total number of data subcarriers is N_{SD}. The average signal energy of these data subcarriers for the mth symbol is then

$$E_m = \left(\frac{1}{N_{\mathrm{SD}}}\right)\sum_{j=1}^{N_{\mathrm{SD}}}\left|Z_{mk_j}\right|^2 \tag{12.14}$$

The noise energy in the frequency domain can be computed in a similar fashion and is given here:

$$N_m = \left(\frac{1}{N_{\text{SD}}}\right)\sum_{j=1}^{N_{\text{SD}}}\left|N_{mk_j}\right|^2 \tag{12.15}$$

where N_{mk_j} is the noise energy at the subcarrier index k_j of the mth symbol and N_m represents the average noise energy of the mth symbol in the frequency domain.

Assuming that there are a total of N symbols, the average bit SNR E_b/N_0 is then given by

$$\frac{E_b}{N_0} = \left(\frac{1}{N}\right)\sum_{m=1}^{N}\frac{E_m}{N_m} \tag{12.16}$$

12.4 Transmitter Architecture for Simulation

For simulation, the transmitter architecture is the same as given in Figure 4.1. The transmitter waveform follows exactly the specification of the IEEE 802.11a. The channel spacing is 20 MHz. Four different modulation techniques were studied and they are BPSK, QPSK, 16-QAM, and 64-QAM. The error control coding selected is the convolutional coder. The three possible coding rates are 1/2, 3/4, and 2/3. The data scrambling and interleaving follow those specified in Chapters 5 and 7.

The preamble includes both short sequence and long sequence. The short sequence has 10 segments and each segment lasts 0.8 μs. The total number of samples is then 160. The long sequence has two data symbols of 3.2 μs each and two guard intervals of 0.8 μs each. The short sequence then lasts 8 μs and the same is true for the long sequence. The total preamble period is then 16 μs.

Each data symbol has a duration of 3.2 μs and a cyclic prefix of 0.8 μs. The total signal duration for each symbol is then 4 μs. The number of subcarriers is 48 for data and 4 for pilots and the resulting total becomes 52. The size of IFFT is 64. This means that there are 12 unused subcarriers and their complex amplitudes are set to zero. Before the data transmission, a signal symbol of 4 μs is also inserted.

The channel is tracked in two ways. The first is that block type pilot patterns are inserted at a given period. The second is to the use the LMS algorithm to track the channel at every data subcarrier. For the block-type pilot pattern, the actual data rate is reduced according to (10.1) due to the overhead transmission. However, there is the advantage of reducing the bit error rate (BER).

12.5 Receiver Architecture for Simulation

The receiver architecture follows that were given in Figure 4.13. Before the data demodulation, there will be synchronization and signal detection. The detection algorithm used is the maximum likelihood detection based upon the simple rule according to (8.27). The decision metric is the normalized cross-correlation.

The long sequence has two data symbols. The second data symbol is used for the initial estimation of the channel. The equalization starts at the signal symbol and continues with the following data symbols. The multipath channel can have fading and the $w_n(t)$ given in (12.10) varies with time. This means that the channel must also be tracked and the LMS algorithm based upon (10.42) is used.

The effectiveness of transmitting the block pilot patterns is also simulated. In other words, the channel is estimated every M data symbols. At the end of the new channel estimation, it is then tracked until the next pilot period.

After the channel equalization, the sequence of demodulation, deinterleaving, descrambling, and convolutional decoding follow to restore the originally transmitted data.

The performance metric is the BER, defined here:

$$\text{BER} = \frac{\text{Total number of bits in error}}{\text{Total number of transmitted bits}}$$

The unit is normally in percentage.

12.6 Simulation Studies

The simulation of OFDM is grouped into two categories. The first is the addition of noise, but there are no multipath degradations. The second is with both noise and multipath degradations.

The details are given in the following sections.

12.6.1 Performance over AWGN

In the following, the transmitter and receiver architecture described in Sections 12.4 and 12.5 are simulated. The signal detection accuracy can impact the BER performance. Therefore, the beginning data sample number detected is checked for every run. As described in Chapter 8, the detection is performed in two steps. The first step is coarse detection and the second step is

fine detection. It was found that the beginning sample location is accurately determined without error for $E_b/N_0 > 5.2$ dB. For E_b/N_0 between 2.38 dB and 5.2 dB, there is one sample error for the coarse detection. However, the fine detection can accurately correct the error. In other words, the signal detection plays no role in the BER performance.

Figure 12.1 shows the BER performance in decibels over AWGN. For comparison purposes, four modulation schemes, BPSK, QPSK, 16-QAM, and 64-QAM, are included. The error correction coding applied is the convolutional coder of coding rate = 1/2. The detailed structure is given in Section 7.4.3. As expected, BPSK has the best performance while 64-QAM has the worst performance.

For BPSK, the BER is 0 at $E_b/N_0 = 8.71$ dB, 0.000385 at $E_b/N_0 = 7.71$ dB, 0.000866 at $E_b/N_0 = 6.71$ dB, and 0.00182 at $E_b/N_0 = 5.71$ dB. The inclusion of error correction coding and interleaver significantly improve the error rate performance.

The simulation results also show that the BER is 0 at $E_b/N_0 = 8.71$ dB for BPSK, 12.71 dB for QPSK, 18.81 dB for 16-QAM, and 26.72 dB for 64-QAM. As the modulation becomes more complicated to increase the data rate, there is also a requirement for a higher SNR.

Table 12.1 lists the E_b/N_0 for BPSK, QPSK, 16-QAM and 64-QAM at around the same BER. There is about 5.9-dB improvement in E_b/N_0 using BPSK instead of QPSK. If QPSK is used instead of 16-QAM, the improvement of E_b/N_0 is about 7.1 dB.

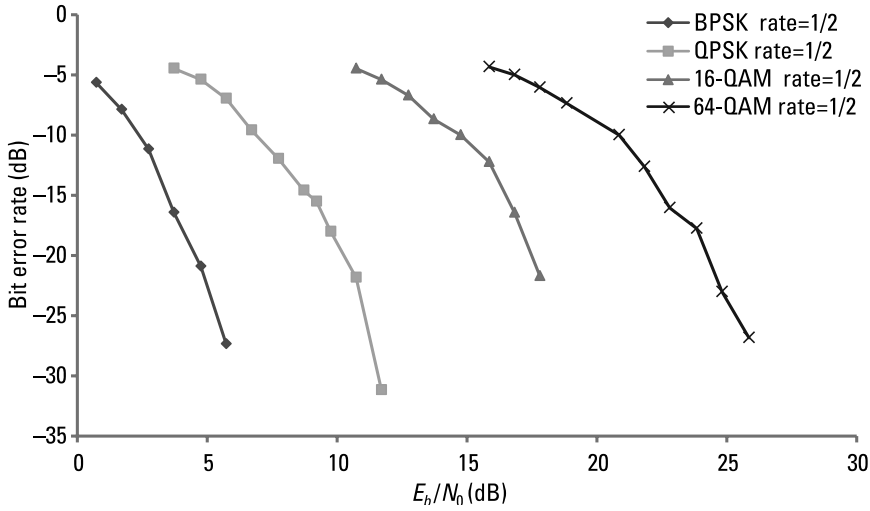


Figure 12.1 BER performance over AWGN using a convolutional of coding rate = 1/2.

Table 12.1
E_b/N_0 Comparisons of Different Modulations at Around the Same BER

Modulation	Coding Rate	BER	E_b/N_0
BPSK	½	0.71%	4.82
QPSK	½	0.65%	10.71
16-QAM	½	0.68%	17.81
64-QAM	½	0.50%	24.81

The punctured convolutional coder has been discussed in Chapter 7. It is generated based upon the coder rate = 1/2 by dropping bits at a specified pattern. The net result is an increase in date rate at a cost of a higher BER. Figure 12.2 a plot of BER performance over AWGN by comparing a rate = 1/2 and a rate = 3/4 convolutional coder. Both QPSK and BPSK modulations are used in the comparisons.

As expected, the BER of the rate = 1/2 coder is always lower than the punctured coder of rate = 3/4. At the same BER, that means a higher E_b/N_0 is required for rate = 3/4 coder. An example is shown in Table 12.2 by listing the E_b/N_0 at around the BER of 0.012.

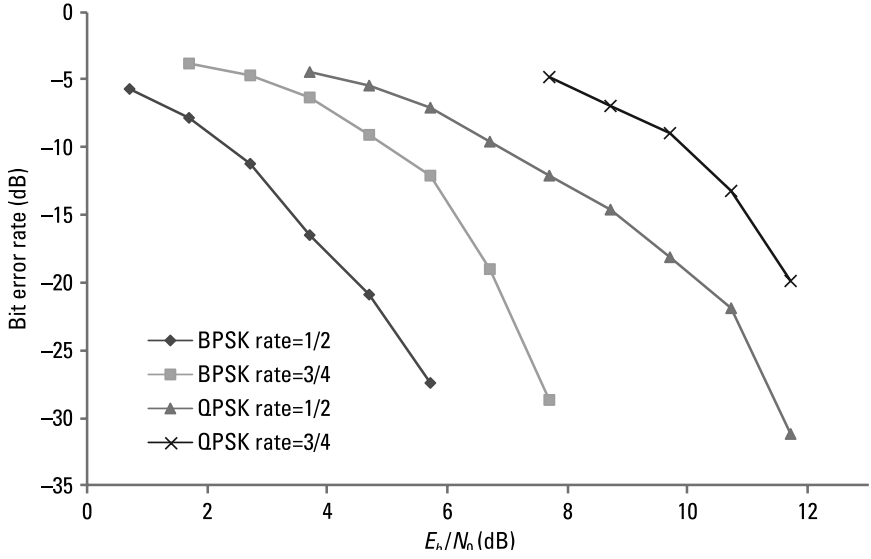


Figure 12.2 BER comparisons between QPSK and BPSK.

Table 12.2
E_b/N_0 Comparisons Between Rate = 1/2 and 3/4 at Around the Same BER

Modulation	Coding Rate	BER	E_b/N_0
QPSK	1/2	1.30%	10.01
QPSK	3/4	1.28%	11.51
BPSK	1/2	1.3%	4.22
BPSK	3/4	1.24%	6.71

For QPSK, the E_b/N_0 increase in using the coding rate of 3/4 instead of 1/2 is 1.5 dB to achieve about the same BER. However, from Table 4.1, the data rate increases from 48 Mbps to 72 Mbps using a 20-MHz channel spacing. For BPSK, there is an increase of E_b/N_0 by 2.5 dB in using coding rate of 3/4 instead of 1/2. Again, from Table 4.1, the data rate increases from 24 Mbps to 36 Mbps using a 20-MHz channel spacing. If a higher data rate is required, then a slight loss in E_b/N_0 is not a bad option. The BER becomes zero at $E_b/N_0 = 12.71$ dB for QPSK and at $E_b/N_0 = 8.72$ dB for BPSK using a bit coding rate of 3/4.

12.6.2 Performance over a Multipath Channel

In the previous section, the BER performance over background noise was given. In a real environment, there can be additional multipath delay and signal fading. To minimize the impact of these degradations, two techniques were discussed in Chapter 10. The first is the LMS algorithm to track the channel response. The second is to add a block-type pilot pattern. The efficiency of both techniques is simulated in this section.

Before the simulation data is given, the simulated multipath channel and simulation parameters are first presented.

12.6.2.1 Multipath Channel Model for Simulation

A multipath channel characterized by (12.11) is written here

$$z(t) = \sum_{n=1}^{L(t)} w_n(t) s\left(t - \tau_n(t)\right) \tag{12.17}$$

In (12.17), $w_n(t)$ is the signal fading and can have both amplitude and phase perturbations and $\tau_n(t)$ is the path delay. For the simulation study, we consider two multipath channels or $L(t) = 2$. One path is the main path and

the other path is the reflected path due to obstacles. For the main path or $n = 1$, the propagation delay is assumed negligible and is set to zero. For the reflected path or $n = 2$, $\tau_2(t)$ is assumed to be a constant, τ. Based upon these two simplifications, (12.17) becomes

$$z(t) = w_1(t)s(t) + \omega_2(t)s(t-\tau) \tag{12.18}$$

In (12.18), the weighting coefficients $\omega_1(t)$ and $\omega_2(t)$ are complex and time variant. In the simulation, they are further given by the following two equations:

$$w_1(t) = w_{1p} + \delta_1(t) \tag{12.19}$$

$$w_2(t) = w_{2p} + \delta_2(t) \tag{12.20}$$

where $\delta_1(t)$ and $\delta_2(t)$ are small perturbations around their principal values. For the main path, the principal value is w_{1p} and for the reflected path, the principal value is w_{2p}. Both $\delta_1(t)$ and $\delta_2(t)$ are complex and can be written as follows:

$$\delta_n(t) = \delta_{nr}(t) + j\delta_{ni}(t) \quad n = 1,2 \tag{12.21}$$

Both $\delta_{nr}(t)$ and $\delta_{ni}(t)$ are assumed to be Gaussian random variables with a zero mean and standard deviation of σ_n. This standard deviation can be considered as the volatility or strength of the fading. The larger the standard deviation, the stronger is the signal fading.

For each path, a set of two random variables are generated every M data symbols. One is for the real part, $\delta_{nr}(t)$, and the other is for the imaginary part, $\delta_{ni}(t)$. We assume that the channel cannot change that fast. Between the beginning and ending symbols, the channel is estimated through linear interpolation at these M symbols. Let $t = 0$ and $t = MT$ represent the beginning and ending time of any M symbol period where T is the symbol duration. The symbol duration here includes the guard interval. Assume further that $\delta_n(0)$ represents the channel perturbation at $t = 0$ and $\delta_n(MT)$ represents the channel perturbations at $t = MT$. Then at $t = kT$, the channel perturbation, $\delta_n(kT)$, is given by

$$\delta_n(kT) = \delta_n(0) + \frac{k}{M}\left(\delta_n(MT) - \delta_n(0)\right) \tag{12.22}$$

For each symbol, the channel remains the same without any change throughout the entire duration. After computing $\delta_n(kT)$, (12.19) and (12.20) can be used to compute $w_n(t)$.

12.6.2.2 Key Parameter Values for Simulation

For the simulation study, the transmitted waveform given by the IEEE 802.11a is used. The channel spacing is 20 MHz. The modulation is QPSK using 1/2 convolutional error correction coding. The data scrambling and interleaving are all included in the simulation. Each symbol lasts 4 μs, which includes 3.2 μs for data and 0.8 μs for the guard interval.

Since the LMS algorithm is used to track the channel, the performance depends strongly on how the tracking coefficient μ defined in (10.54) is used. If the μ is too large, the channel estimation error diverges and system performance degrades. However, if it is too small, the convergence speed is too slow. To obtain an appropriate value, some test runs were applied and the parameter α given in (10.54) is set to 0.01.

On the block pilot pattern, the known data is inserted with three different periods. They are 5, 10, and 20. If the period is N, that means a known pilot symbol is inserted after every N symbols. Table 4.1 shows that the data rate for QPSK using the 1/2 coding rate and 20-MHz channel spacing is 12 Mbps. For periods of 5, 10, and 20, the new data rate is reduced to 10 Mbps, 10.9 Mbps, and 11.42 Mbps, respectively.

On the channel model, the parameter M defined in (12.22) is set to 20 and the delay for the reflected path is set to 0.2 μs. The principal values for w_{1p} and w_{2p} are set as 1 and 0.5, respectively. The fading strength is given by its standard deviation, σ_n. This means that $w_1(t)$ in (12.19) may have values ranging from $1 - \sigma_n$ to $1 + \sigma_n$ and $w_2(t)$ in (12.20) may have values ranging from $0.5 - \sigma_n$ to $0.5 + \sigma_n$. Three different values of σ_n, 0.1, 0.25, and 0.3, are used in the simulation.

12.6.2.3 Simulation Results

As was mentioned before, two approaches are used to minimize the impact of a multipath channel with signal fading. One is the LMS algorithm and the other is the block pilot pattern. On the block pilot pattern, the channel at the data symbols between the inserted pilot patterns is still tracked using the LMS algorithm.

Figure 12.3 is a plot of BER in percentage against the E_b/N_0 (dB) when the fading standard deviation is 0.1. Three different noise seeds of 93, 40, and 60 are used and the average BER from these three runs is performed. There are four curves. One curve is for the LMS algorithm without any block pilot pattern. The other three curves have a block pilot pattern with pilot periods of 5, 10, and 20 symbols. The LMS algorithm is still used to track the channel at the data symbols. The LMS algorithm tracks the channel very well as is evidenced by the negligible error rate percentage when the E_b/N_0 is above

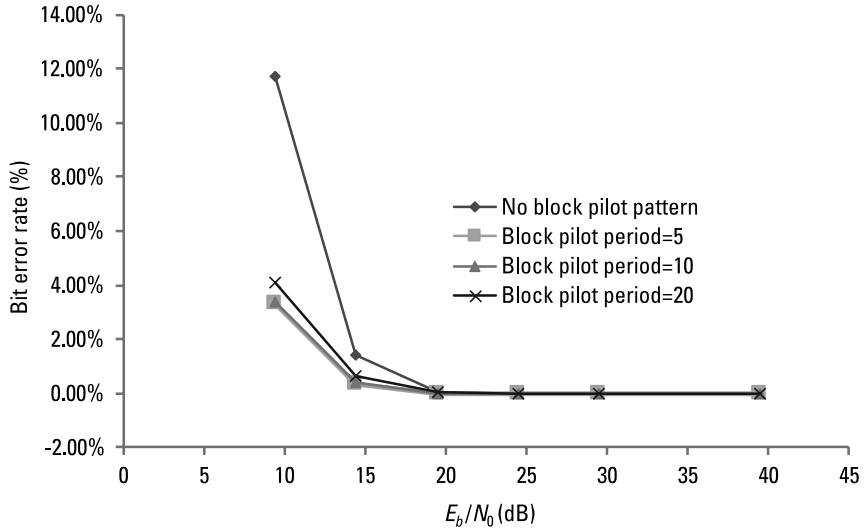


Figure 12.3 BER comparisons with fading standard deviation = 0.1.

20 dB. When the E_b/N_0 is below 20 dB, the performance advantage of adding a block pilot pattern is obvious. As the period of pilot pattern increases, the error percentage also increases. When the pilot period is 20 symbols, the data rate only decreases to 11.42 Mbps from the original 12 Mbps without a

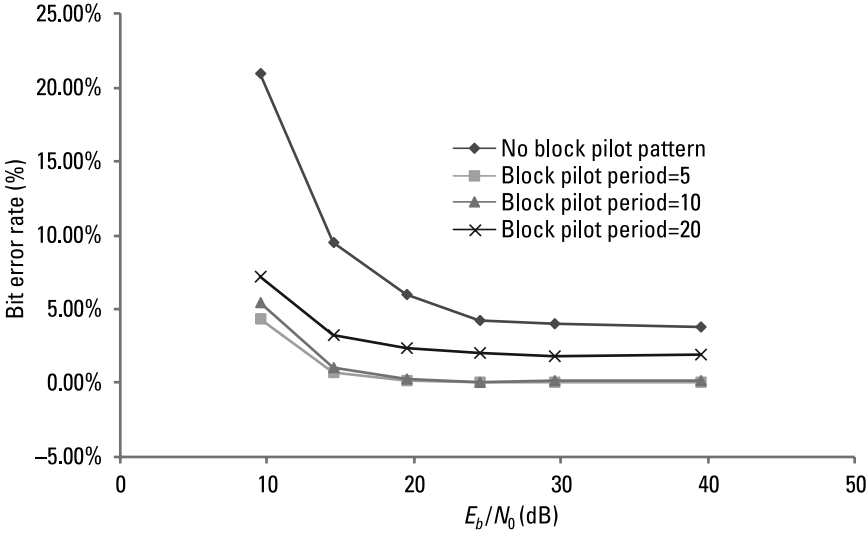


Figure 12.4 BER comparisons with fading standard deviation = 0.25.

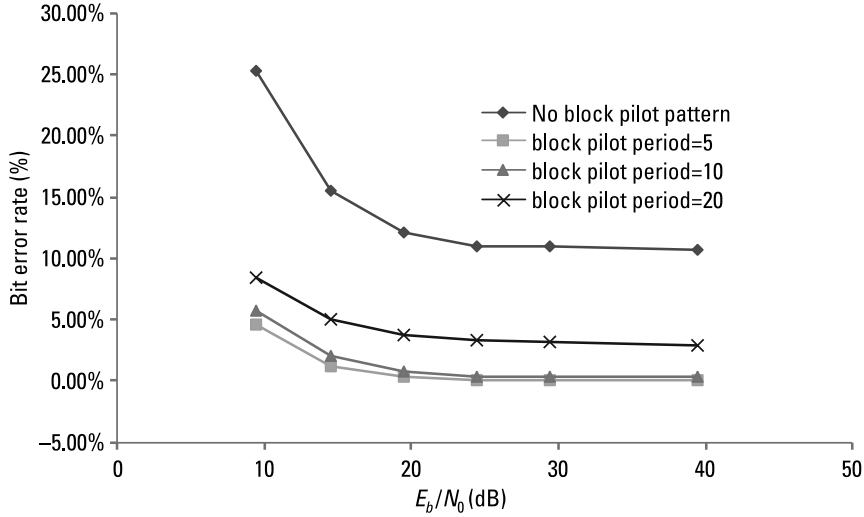


Figure 12.5 BER comparisons with fading standard deviation = 0.3.

block pilot pattern. However, the BER is significantly improved and especially at a low E_b/N_0.

Figures 12.4 and 12.5 are similar plots except the fading standard deviation is increased to 0.25 and 0.3. The advantage of adding a block pilot pattern can clearly be seen. There is a small price paid due to data rate decrease, but the big gain in BER performance is still a worthy approach.

12.7 Summary

A multipath channel was characterized and simulated. First, the BER in AWGN was compared for four different modulations, BPSK, QPSK, 16-QAM, and 64-QAM. As the constellation bit encoding becomes more complex, the performance also degrades. The BPSK has the best performance and there is a gain of around 5.9 dB in E_b/N_0against QPSK while 64-QAM has the worst performance. Under the same modulation, punctured convolutional coding can gain in data rate increase with a slight price paid on BER performance.

For QPSK and 1/2 convolutional encoding, the BER performance in a multipath channel was further performed. Two approaches were studied. One is the LMS algorithm only. The other is the same LMS algorithm with an added block pilot pattern. The result shows that the addition of block pilots

can significantly improves the BER, especially at a low E_b/N_0. The price paid is only a slight decrease in date rate.

Reference

[1] Proakis, J. G., *Digital Communications*, New York: McGraw-Hill, 1983.

About the Author

Y. J. Liu received his electrical engineering B.S. from the National Taiwan University, his M.S. from the University of Rochester, and his Ph.D. from the Ohio State University. He has been working in the industry since 1978. Initially, he worked for the Lear Siegler Instrument division in LORAN and OMEGA navigational signal processing and isolated word recognition under a noisy environment. He then joined Nartron Corporation in developing an isolated word recognizer for a passenger car. Next, he worked at ITT Exelis for almost 28 years. He was involved in numerous projects related to the development of handheld, wireless, and digital combat tactical radios. Specific working areas include ad hoc networking and modem and low-rate speech coding. He has also had many papers published and patents awarded.

Index

Recent Titles in the Artech House Mobile Communications Series

William Webb, Series Editor

3G CDMA2000 Wireless System Engineering, Samuel C. Yang

3G Multimedia Network Services, Accounting, and User Profiles, Freddy Ghys, Marcel Mampaey, Michel Smouts, and Arto Vaaraniemi

5G New Radio: Beyond Mobile Broadband, Amitav Mukherjee

5G Spectrum and Standards, Geoff Varrall

802.11 WLANs and IP Networking: Security, QoS, and Mobility, Anand R. Prasad and Neeli R. Prasad

Achieving Interoperability in Critical IT and Communications Systems, Robert I. Desourdis, Peter J. Rosamilia, Christopher P. Jacobson, James E. Sinclair, and James R. McClure

Advances in 3G Enhanced Technologies for Wireless Communications, Jiangzhou Wang and Tung-Sang Ng, editors

Advances in Mobile Information Systems, John Walker, editor

Advances in Mobile Radio Access Networks, Y. Jay Guo

Applied Satellite Navigation Using GPS, GALILEO, and Augmentation Systems, Ramjee Prasad and Marina Ruggieri

Artificial Intelligence in Wireless Communications, Thomas W. Rondeau and Charles W. Bostian

Broadband Wireless Access and Local Network: Mobile WiMax and WiFi, Byeong Gi Lee and Sunghyun Choi

CDMA for Wireless Personal Communications, Ramjee Prasad

CDMA Mobile Radio Design, John B. Groe and Lawrence E. Larson

CDMA RF System Engineering, Samuel C. Yang

CDMA Systems Capacity Engineering, Kiseon Kim and Insoo Koo

CDMA Systems Engineering Handbook, Jhong S. Lee and Leonard E. Miller

Cell Planning for Wireless Communications, Manuel F. Cátedra and Jesús Pérez-Arriaga

Cellular Communications: Worldwide Market Development, Garry A. Garrard

Cellular Mobile Systems Engineering, Saleh Faruque

Cognitive Radio Interoperability through Waveform Reconfiguration, Leszek Lechowicz and Mieczyslaw M. Kokar

Cognitive Radio Techniques: Spectrum Sensing, Interference Mitigation, and Localization, Kandeepan Sithamparanathan and Andrea Giorgetti

The Complete Wireless Communications Professional: A Guide for Engineers and Managers, William Webb

Digital Communication Systems Engineering with Software-Defined Radio, Di Pu and Alexander M. Wyglinski

EDGE for Mobile Internet, Emmanuel Seurre, Patrick Savelli, and Pierre-Jean Pietri

Emerging Public Safety Wireless Communication Systems, Robert I. Desourdis, Jr., et al.

From LTE to LTE-Advanced Pro and 5G, Moe Rahnema and Marcin Dryjanski

The Future of Wireless Communications, William Webb

Geographic Information Systems Demystified, Stephen R. Galati

Geospatial Computing in Mobile Devices, Ruizhi Chen and Robert Guinness

GPRS for Mobile Internet, Emmanuel Seurre, Patrick Savelli, and Pierre-Jean Pietri

GPRS: Gateway to Third Generation Mobile Networks, Gunnar Heine and Holger Sagkob

GSM and Personal Communications Handbook, Siegmund M. Redl, Matthias K. Weber, and Malcolm W. Oliphant

GSM Networks: Protocols, Terminology, and Implementation, Gunnar Heine

GSM System Engineering, Asha Mehrotra

Handbook of Land-Mobile Radio System Coverage, Garry C. Hess

Handbook of Mobile Radio Networks, Sami Tabbane

High-Speed Wireless ATM and LANs, Benny Bing

Inside Bluetooth Low Energy, Second Edition, Naresh Gupta

Interference Analysis and Reduction for Wireless Systems, Peter Stavroulakis

Interference and Resource Management in Heterogeneous Wireless Networks, Jiandong Li, Min Sheng, Xijun Wang, and Hongguang Sun

Internet Technologies for Fixed and Mobile Networks, Toni Janevski

Introduction to 3G Mobile Communications, Second Edition, Juha Korhonen

Introduction to 4G Mobile Communications, Juha Korhonen

Introduction to Communication Systems Simulation, Maurice Schiff

Introduction to Digital Professional Mobile Radio, Hans-Peter A. Ketterling

Introduction to GPS: The Global Positioning System, Ahmed El-Rabbany

An Introduction to GSM, Siegmund M. Redl, Matthias K. Weber, and Malcolm W. Oliphant

Introduction to Mobile Communications Engineering, José M. Hernando and F. Pérez-Fontán

I*ntroduction to OFDM Receiver Design and Simulation*, Y. J. Liu

Introduction to Radio Propagation for Fixed and Mobile Communications, John Doble

Introduction to Wireless Local Loop, Broadband and Narrowband, Systems, Second Edition, William Webb

IS-136 TDMA Technology, Economics, and Services, Lawrence Harte, Adrian Smith, and Charles A. Jacobs

Location Management and Routing in Mobile Wireless Networks, Amitava Mukherjee, Somprakash Bandyopadhyay, and Debashis Saha

LTE Air Interface Protocols, Mohammad T. Kawser

Metro Ethernet Services for LTE Backhaul, Roman Krzanowski

Mobile Data Communications Systems, Peter Wong and David Britland

Mobile IP Technology for M-Business, Mark Norris

Mobile Satellite Communications, Shingo Ohmori, Hiromitsu Wakana, and Seiichiro Kawase

Mobile Telecommunications Standards: GSM, UMTS, TETRA, and ERMES, Rudi Bekkers

Mobile-to-Mobile Wireless Channels, Alenka Zajić

Mobile Telecommunications: Standards, Regulation, and Applications, Rudi Bekkers and Jan Smits

Multiantenna Digital Radio Transmission, Massimiliano Martone

Multiantenna Wireless Communications Systems, Sergio Barbarossa

Multi-Gigabit Microwave and Millimeter-Wave Wireless Communications, Jonathan Wells

Multipath Phenomena in Cellular Networks, Nathan Blaunstein and Jørgen Bach Andersen

Multiuser Detection in CDMA Mobile Terminals, Piero Castoldi

OFDMA for Broadband Wireless Access, Slawomir Pietrzyk

Personal Wireless Communication with DECT and PWT, John Phillips and Gerard MacNamee

Practical Wireless Data Modem Design, Jonathon Y. C. Cheah

Prime Codes with Applications to CDMA Optical and Wireless Networks, Guu-Chang Yang and Wing C. Kwong

Quantitative Analysis of Cognitive Radio and Network Performance, Preston Marshall

QoS in Integrated 3G Networks, Robert Lloyd-Evans

Radio Engineering for Wireless Communication and Sensor Applications, Antti V. Räisänen and Arto Lehto

Radio Propagation in Cellular Networks, Nathan Blaunstein

Radio Resource Management for Wireless Networks, Jens Zander and Seong-Lyun Kim

Radiowave Propagation and Antennas for Personal Communications, Third Edition, Kazimierz Siwiak and Yasaman Bahreini

RDS: The Radio Data System, Dietmar Kopitz and Bev Marks

Resource Allocation in Hierarchical Cellular Systems, Lauro Ortigoza-Guerrero and A. Hamid Aghvami

RF and Baseband Techniques for Software-Defined Radio, Peter B. Kenington

RF and Microwave Circuit Design for Wireless Communications, Lawrence E. Larson, editor

RF Positioning: Fundamentals, Applications, and Tools, Rafael Saraiva Campos and Lisandro Lovisolo

Sample Rate Conversion in Software Configurable Radios, Tim Hentschel

Signal Processing Applications in CDMA Communications, Hui Liu

Signal Processing for RF Circuit Impairment Mitigation, Xinping Huang, Zhiwen Zhu, and Henry Leung

Smart Antenna Engineering, Ahmed El Zooghby

Software-Defined Radio for Engineers, Travis F. Collins, Robin Getz, Di Pu, and Alexander M. Wyglinski

Software Defined Radio for 3G, Paul Burns

Spread Spectrum CDMA Systems for Wireless Communications, Savo G. Glisic and Branka Vucetic

Technical Foundations of the IoT, Boris Adryan, Dominik Obermaier, and Paul Fremantle

Technologies and Systems for Access and Transport Networks, Jan A. Audestad

Third-Generation and Wideband HF Radio Communications, Eric E. Johnson, Eric Koski, William N. Furman, Mark Jorgenson, and John Nieto

Third Generation Wireless Systems, Volume 1: Post-Shannon Signal Architectures, George M. Calhoun

Traffic Analysis and Design of Wireless IP Networks, Toni Janevski

Transmission Systems Design Handbook for Wireless Networks, Harvey Lehpamer

UMTS and Mobile Computing, Alexander Joseph Huber and Josef Franz Huber

Understanding Cellular Radio, William Webb

Understanding Digital PCS: The TDMA Standard, Cameron Kelly Coursey

Understanding GPS: Principles and Applications, Second Edition, Elliott D. Kaplan and Christopher J. Hegarty, editors

Understanding WAP: Wireless Applications, Devices, and Services, Marcel van der Heijden and Marcus Taylor, editors

Universal Wireless Personal Communications, Ramjee Prasad

WCDMA: Towards IP Mobility and Mobile Internet, Tero Ojanperä and Ramjee Prasad, editors

Wireless Communications in Developing Countries: Cellular and Satellite Systems, Rachael E. Schwartz

Wireless Communications Evolution to 3G and Beyond, Saad Z. Asif

Wireless Intelligent Networking, Gerry Christensen, Paul G. Florack and Robert Duncan

Wireless LAN Standards and Applications, Asunción Santamaría and Francisco J. López-Hernández, editors

Wireless Sensor and Ad Hoc Networks Under Diversified Network Scenarios, Subir Kumar Sarkar

Wireless Technician's Handbook, Second Edition, Andrew Miceli

For further information on these and other Artech House titles,including previously considered out-of-print books now available through our In-Print-Forever® (IPF®) program, contact:

Artech House
685 Canton Street
Norwood, MA 02062
Phone: 781-769-9750
Fax: 781-769-6334
e-mail: artech@artechhouse.com

Artech House
16 Sussex Street
London SW1V 4RW UK
Phone: +44 (0)20 7596-8750
Fax: +44 (0)20 7630-0166
e-mail: artech-uk@artechhouse.com

Find us on the World Wide Web at: www.artechhouse.com