A First Course in Digital Communications

Committers:

Jean-Francois Chamberland

Krishna Narayanan

Henry Pfister

Spring 2012

Copyright

Contents

Preface **ix**

1 Digital Communication **1**

1.1 System Components 1

1.1.1 The Input-Output Blocks 2

1.1.2 The Transmitter-Receiver Pair 4

1.2 Common Channels and Applications 6

2 Sources of Digital Information **9**

2.1 Discrete Memoryless Sources 10

2.2 Entropy 12

2.3 Variable-Length Compression Codes 14

2.3.1 Kraft Inequality 16

2.3.2 Entropy Bounds on Prefix-Free Codes 19

2.3.3 Huffman Codes 21

2.4 Joint Encoding of Source Symbols 24

2.5 Sources with Memory 26

2.6 Universal Source-Coding Algorithms 26

2.7 Fixed-Length Compression Codes* 29

2.7.1 Asymptotic Equipartition Property 29

3 Discrete-Time Communication **33**

3.1 A Simple Channel Model 34

3.2 A Simple Modulation Scheme 35

3.3 Quadrature Amplitude Modulation 36

3.4 Optimal Symbol Detection 38

3.4.1 Hypothesis Testing 39
3.4.2 Multiple Hypothesis Testing 40

4 Fourier Analysis and Sampling 43
4.1 Fourier Series 43
4.2 Fourier Transforms 45
4.2.1 The Dirac Delta Function 46
4.2.2 Periodic Signals 47
4.2.3 Spectral Density 48
4.2.4 Linear Time-Invariant Filters 50
4.3 Sampling Deterministic Signals 50
4.3.1 The Sampling Theorem 52
4.3.2 Imperfect Sampling and Reconstruction 54
4.4 Sampling Bandlimited Processes* 55
4.5 Bandpass Signals and Processes* 56
4.6 Stochastic Signals 58
4.6.1 Power Spectral Density 60
4.6.2 Filtering Stochastic Processes 61
4.6.3 Gaussian Processes 63

5 Quantization 65
5.1 Scalar Quantizers 65
5.2 Distortion Measures 67
5.2.1 Mean Squared Error 68
5.2.2 Signal to Quantization-Noise Ratio 69
5.2.3 Dithering 72
5.2.4 Non-Uniform Quantization via Companding 73
5.3 Predictive Quantization 75
5.3.1 Delta Modulation 75
5.3.2 Estimation and Prediction 76
5.4 Optimal Quantization 77
5.4.1 Uniform Quantizers 77
5.4.2 Non-Uniform Quantizers 79
5.4.3 Lloyd-Max Algorithm 81

5.5 Vector Quantizers 82
5.6 Analysis-Synthesis Algorithms 85

6 Channel Coding 87
6.1 Introduction 87
6.1.1 What is channel coding and why do we use it? 87
6.1.2 Channels and Error Models 88
6.2 The Basics of Coding 90
6.2.1 Codes 90
6.2.2 Decoding 91
6.3 Binary Linear Codes 92
6.3.1 Basic Properties 92
6.3.2 Generator and Parity-Check Matrices 94
6.3.3 Decoding 97
6.3.4 Manipulating Linear Codes 100
6.4 Performance Analysis 101
6.5 Cyclic Codes 103
6.5.1 Basic Properties 103
6.5.2 Systematic Encoding 106
6.5.3 Cyclic Redundancy Check 107

7 Waveform Communication 109
7.1 A Single Digital Symbol 109
7.1.1 Orthogonal Waveforms 110
7.1.2 General Waveforms 112
7.1.3 The Matched Filter 114
7.2 Time-Shift Waveforms 115
7.2.1 Demodulation 116
7.2.2 The Nyquist Criterion 117
7.2.3 Linear Time-Invariant Channels 122
7.2.4 Passband Communication 123

Index 125

Preface

These notes provide an introduction to the fundamental concepts of digital communication systems. The material emphasizes the unifying principles of communication theory, taking a mathematical approach to system design. The main topics covered in these notes include sampling, quantization, data compression, channel coding, Shannon capacity and modulation theory. Possessing some programming skills will also help in order to appreciate and use the computing material and examples contained in this document.

Major Goals

1. Identify the various components of a digital communication system. Discuss the purpose of source coding, channel coding, modulation, and equalization. Become familiar with commonly encountered digital communication systems, and discuss how these systems can be decomposed into the same abstract constituent parts.

2. Review basic notions from Fourier analysis, including Fourier series and Fourier transforms. Define the power spectrum of stochastic signals, and explore how it is affected by linear filtering.

3. Explore methods to convert an analog signal into a digital format through sampling and quantization. Define the mean squared error and explain its role in assessing the performance of a digital communication system.

4. Discuss the purpose of information theory, and calculate the entropy of simple information sources. Understand fundamental compression limits and survey efficient source coding algorithms.

5. Introduce the notions of channel capacity and error protection. Understand how simple block codes work and compute the probability of decoding failure for simples codes.

6. Present simple modulation schemes, signal waveforms, and their vector space representations. Characterize the structure of optimal receivers, and compute the probabilities of symbol and bit errors at the output of the demodulator.

7. Explore the properties of bandlimited channels. Study the causes and implications of intersymbol interference, and derive the Nyquist criterion for no interference. Review simple channel equalization schemes and go over the advantages of orthogonal frequency-division multiplexing.

Chapter 1

Digital Communication

The advent of data transmissions over physical channels has transformed the modern communication landscape. The term *digital communication* broadly refers to the transfer of information using digital messages or bit streams. There are notable advantages to transmitting data using discrete messages. It allows for enhanced signal processing and quality control. In particular, errors caused by noise and interference can be detected and corrected systematically. Furthermore, digital communication makes the networking of heterogeneous systems possible, with the Internet being the most obvious such example. These advantages, and many more, explain the widespread adoption and constantly increasing popularity of digital media.

1.1 System Components

The operation of a typical digital communication system can be represented by the functional block diagram depicted in Figure 1.1. It is composed of five basic components. The input block contains the source, which produces data such as voice, emails, or images; and it also includes all the operations that are required to convert the original signal waveform into a format suitable for transmission. The transmitter takes bits from the input block and sends them over a channel using electromagnetic signals or some alternate means. It may add a protective layer to the intended message to shield it from noise and interference. Communication channels come in many flavors. For instance, a transmission can take place over an ethernet cable, a coaxial cable, or free space (wireless communication). There are also more esoteric channels like a hard drive platter, a compact disc, or a memory stick. The role of the receiver is to recover the sent message from a collection of measurements. This unit may need to extract the signal from noise and, possibly, correct errors that may have occurred during transmission. Finally, the output block takes the received information and puts it back into a format that is appropriate for the end-users.

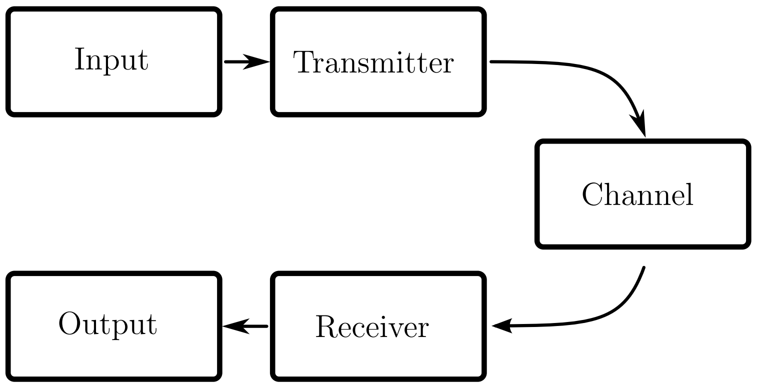

Figure 1.1: A digital communication system contains five basic components. These building blocks appear in this diagram.

Figure 1.1 also alludes to the natural symmetry present in digital communication systems. Operations that take place on the transmitter side must often be undone at the destination. As such, complementary steps are frequently studied in pairs. Our treatment of digital communication follows this general approach.

1.1.1 The Input-Output Blocks

The role of the input block is to take information in its original form and to convert it into a digital format. Depending on the nature of the source, two operations may be necessary to digitize the information waveform. *Sampling* is a signal processing technique that transforms a continuous-time function to a discrete-time signal. The conversion of a sound wave into a sequence of samples is a common application of sampling. The second action that may be needed is *quantization*, which maps a continuous-space signal into a discrete set of possible values. The combination of these two operations will transform an analog signal into a digital message.

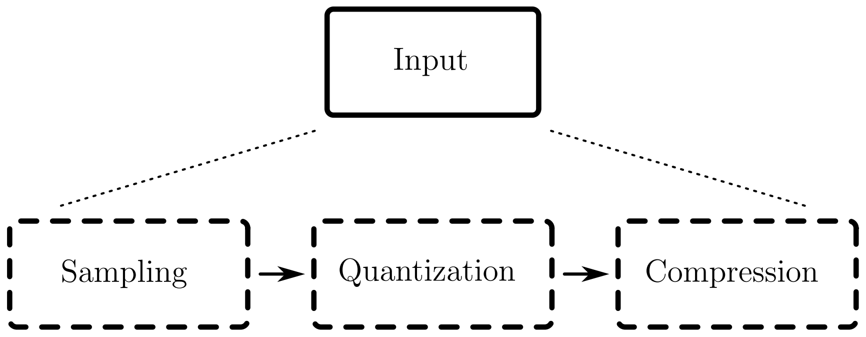

Figure 1.2: The input block can be further divided into three operations: sampling, quantization and data compression. The first two operations are necessary to transform an analog waveform into a digital signal; whereas the last block is optional and provides a means to reduce the size of a digitized message.

The optional step of *data compression*, or source coding, is often employed at the origin to reduce the size of the message to be transmitted. This, in turn, brings down the consumption of expensive physical resources such as power, spectral bandwidth and hard disk space. On the downside, data processing entails additional computations and delay, and the compressed data must be expanded before being accessed. This involves using extra processing on the receiver side as well. Some compression schemes reduce the quality of the data. These schemes usually feature better compression ratios at the expense of introducing small variations in the data. The MPEG standards, including the MP3 audio layer, provide examples of lossy compression schemes.

At the output of the system, the data must be put back into a format that is acceptable to the end-user. The received message must be decompressed. Note that for the destination to be able to recover the original data, it must understand the encoding scheme utilized by the sender. In other words, the decoding method must be known at the receiver. If the original signal is a continuous-time waveform, then an interpolator can be used to reconstruct the waveform from its sample values. The quantization block present at the input

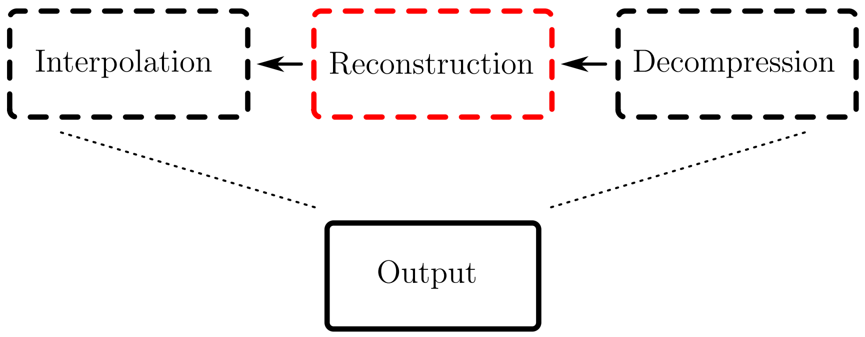

Figure 1.3: The operations executed at the transmitter must typically be undone at the destination. When passed through the output block, the received data is first decompressed and put in a format that suitable for signal processing. Furthermore, an analog signal must be reconstructed from the sampled data point. While these two steps can reversed without any loss, the quantization step cannot. This is because quantization loses a small part of the original information. Instead, the reconstruction step reverses quantization step in an approximate way to minimize the distortion.

does not have an exact counterpart at the output. This deficiency follows from the fact that quantization cannot be perfectly undone because information is typically lost when a continuous-valued signal is discretized. The level of *distortion* associated with quantization can, however, be controlled by choosing an appropriate scheme and reconstructing the signal in a sensible way. In most cases, the impact of quantization is minimal; indeed, quantizers are often designed to induce negligible distortion.

1.1.2 The Transmitter-Receiver Pair

A channel code is used at the transmitter to shield data sent over the channel against errors due to noise and interference. This level of protection is achieved by adding redundancy to the information bits in a structured manner. Audio compact discs make use of a *Reed-Solomon code* to protect digital music from scratches and dust, whereas a low-latency *convolutional code* is employed in cellphones to carry voice signals. The second task of the transmission unit is to modulate the digital bit-stream onto an analog carrier prior to transmission.

The possible waveforms emitted by the transmitter are chosen from a finite number of symbols. The transmitter must produce a signal that remains confined to its assigned spectral bandwidth.

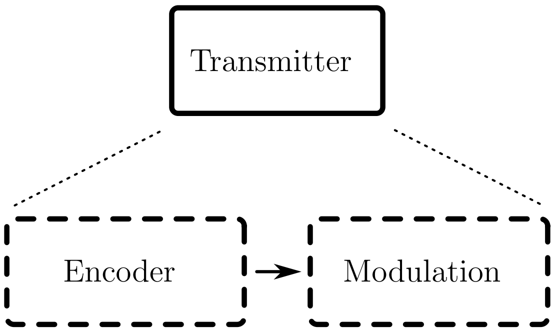

Figure 1.4: At the transmitter, redundancy is added to the digital message to protect information bits against noise and interference. Once this is completed, the bit stream is modulated onto an analog carrier and transmitted to the destination.

The message then propagates through the communication channel. This is the physical medium that bridges the gap between the transmitter and the receiver. In most applications, the channel acts as to transfer the data to a different place. However, in hard disk drives and compact discs, the information is stored simply to be accessible at a later time. Most communication channels cause signal degradation. The message may be subject to attenuation, interference and noise corruption. That is, the communication channel may be unreliable; and its environment, hard to characterize. Recovering the original data from a set of measurements available at the receiver is one of the many challenges of digital communication systems.

The receiver is tasked with extracting the original symbols from noisy measurements. The first step consists of estimating which symbols were sent by the transmitter over the channel. This procedure is termed *demodulation.* These symbol estimates are then translated back into information bits by the decoder. Redundancy is removed from the data, and the original format of the information bits is restored. When channel conditions are harsh and the local measurement noisy, this process may fail and the corresponding data is lost. Scratching a compact disc repetitively will illustrate this point well; after sub-

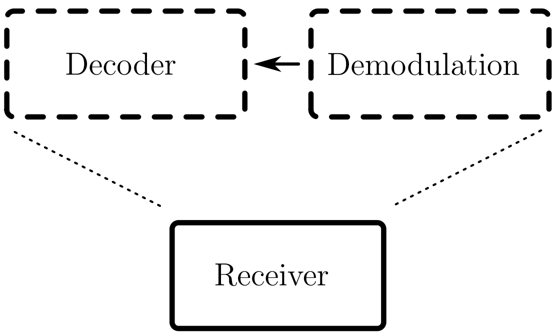

Figure 1.5: On the receiver side, the sent symbols are extracted from the received signal. Error-correction techniques are then applied to insure the integrity of the acquired data. Finally, the digital message is restored to its original format.

stantial disc abuse, a player is no longer able to reconstruct the music. At this point, the receiver may elect to notify the sender and request a retransmission of the desired message.

1.2 Common Channels and Applications

Various communication channels can provide a connection between a transmitter and its destination. Common wireline channels include coaxial cables, ethernet cables, and twisted-pair wires. Optical fiber offers a high-capacity solution for heavy applications. Compact discs and DVDs are also based on optics and they can be employed to store information on discs. Finally, wireless electromagnetic channels are popular for their convenience and broadcast capabilities.

Digital communication occupies a central role in almost every aspect of contemporary life. Most businesses rely directly or indirectly on networked computers and the Internet for day-to-day operations, and digital technologies have become a staple of the entertainment industry. Below, we provide a few examples of digital communication systems that you may be familiar with.

Cable Modem: A cable modem enables point-to-point communication over the cable television infrastructure. They are primarily employed to deliver broadband Internet access, taking advantage of unused bandwidth on a cable television network. With the emergence of voice over Internet protocol (VoIP) telephony, cable modems can also be used to provide telephone service.

Wi-Fi Technology: Wi-Fi is a global set of standards that allows wireless inter-networking. In particular, it includes the IEEE 802.11 protocol suite (e.g., 802.11b, 802.11g and 802.11n). Wireless access points, also called *hotspots*, often provide users access to the Internet. Wi-Fi products can be used as an enabling technology for *mesh networks*, which offer connectivity to large urban communities.

Bluetooth: Bluetooth is a wireless protocol designed specifically for short-range communication. It is used primarily to create personal area networks. Bluetooth provides a means to exchange information between devices such as mobile phones, personal computers, digital cameras, and a myriad of small accessories.

Hard Disk Drive: A hard disk drive is a non-volatile storage device that keeps digitally encoded data on rapidly rotating platters with magnetic surfaces. Today, hard drives can be found in computers, digital audio players, personal digital assistants, game consoles and other embedded computing devices. Data on a hard disk drive is recorded by magnetizing ferromagnetic material directionally, and is read back by detecting the magnetization of the material.

Compact Disc: The compact disc (CD) is an optical disc used to store digital data, and remains one of the popular playback media for commercial audio recordings. A standard compact disc can store approximately 650 Megabytes of data. The data is stored on the disc as a sequence of bumps that are stamped into the polycarbonate during production. A reflective layer is added so that the data can be read by using a laser to distinguish between *pits* and *lands*.

In a recordable compact disc (CD-R), a photosensitive dye is used instead. The write laser of a CD recorder changes the color of the dye to allow a standard CD player to read the data, just as it would with a standard stamped disc. A re-recordable disc medium (CD-RW) uses a metallic alloy instead of a dye. The write laser in this case is used to heat and alter the properties of the alloy, and hence change its reflectivity. A CD-RW does not possess as great a difference in reflectivity as a stamped compact disc, and so many earlier audio players cannot read CD-RW discs, although most later CD audio players and stand-alone DVD players can.

Chapter 2

Sources of Digital Information

Digital systems use sequences of symbols (e.g., binary systems use 0's and 1's) to represent information. A *source* of digital information is assumed to produce a succession of symbols, each drawn from a *discrete alphabet*. The three goals of this chapter are to understand the nature of digital information, find an adequate measure of information for digital systems and to describe compression algorithms that can be employed to represent the said information in a succinct manner. *Data compression*, also known as **source coding**, is important because it reduces the consumption of expensive resources such as hard disk space or transmission bandwidth. Alternatively, it can be applied to lower the cost of communication, reduce latency or improve the quality of the received messages.

This chapter offers an introductory treatment of *lossless compression* algorithms, whereby the original message can be recovered perfectly from the compressed data. This is in contrast to *lossy data compression*, which provides improved compression ratios at the expense of introducing some distortion in the message. In the latter case, part of the information may be lost and the original data need not be perfectly recoverable, although the reconstructed message may be quite close to the original one. For instance, the JPEG algorithm can be employed as a lossy compression scheme to reduce the size of a digital photograph.

In lossless data compression, two strategies are employed to reduce the expected length of a message. Highly probable symbols are assigned short descriptions, and less likely symbols are encoded using longer binary representations. Second, the statistical redundancy contained in the input signal over time is removed, leading to a more concise description of the digital data. Data compression algorithms are explained more thoroughly below.

As we will see, finding a pertinent measure of information is key in assessing the performance and limitations of compression algorithms. While the general notion of information may be quite broad, it has a precise definition in the context of digital communication systems. To describe this specific meaning, we first need to develop a rigorous mathematical model for digital information sources.

2.1 Discrete Memoryless Sources

As mentioned above, a digital source produces a sequence of symbols drawn from a countable alphabet. It can accordingly be modeled as a discrete-time random process. Because of their indeterminate nature, random signals and stochastic processes can be difficult to characterize. Later in this document, we will discuss random processes in more detail. A thorough discussion of the subject requires advanced concepts from probability theory, a topic that interested readers may wish to pursue on their own. For the sake of simplicity, we focus on a class of elementary information sources that are collectively known as discrete memoryless sources. These sources can be described simply as a sequence of independent and identically-distributed discrete random variables. Furthermore, discrete memoryless sources provide valuable insights into the design of efficient compression algorithms for more general settings.

Definition 2.1.1. *A **discrete memoryless source** is a digital information source that produces a sequence of independent and identically distributed symbols over time. Mathematically, it consists of an alphabet $\mathcal{X}$ and a probability mass function $p_X(\cdot)$ such that, at any time t, the probability that the source outputs symbol $x \in \mathcal{X}$ is equal to $p_X(x)$, irrespective of the past and future.*

To completely characterize the statistical properties of a discrete memoryless source, it suffices to define the probability mass function of individual symbols. Since the source generates independent and indentically distributed symbols, the higher-order statistics need not be specified explicitly. Instead, they can be ontained from

$$\Pr(X_{t_1} = x_{t_1}, \ldots, X_{t_n} = x_{t_n}) = \prod_{k=1}^{n} p_X(x_{t_k}) \tag{2.1}$$

where $x_{t_1}, \ldots, x_{t_n} \in \mathcal{X}$. In (2.1), the random variable X_{t_i} denotes the output of the source at time t_i. We provide two examples of memoryless sources below to further illustrate their form.

Example 2.1.2 (Binary Source). *The simplest possible information source is a discrete memoryless source where $p_X(\cdot)$ is the probability mass function of a Bernoulli random variable,*

$$p_X(x) = \begin{cases} (1-p), & x = 0 \\ p, & x = 1 \end{cases}$$

with $p \in [0, 1]$. This source can be employed, for instance, to model the successive flipping of a biased coin, where heads is obtained with probability p and tails is obtained with probability $1 - p$.

Example 2.1.3. *To construct a slightly more elaborate example, consider a collection of experiments where a fair coin is flipped repetitively until heads is observed. The outcome of each experiment is reported as a source output. The source alphabet in this case is $\mathcal{X} = \{1, 2, \ldots\}$, the positive integers, and the marginal probability mass function associated with individual outcomes becomes*

$$p_X(x) = \frac{1}{2^x}, \quad x = 1, 2, \ldots$$

Thus, the distribution of the source output at time t is a geometric random variable with parameter $\frac{1}{2}$.

The independence, over time, of symbols from a discrete memoryless source makes them convenient for analysis. However, it should also be pointed out that many realistic sources are more complicated than memoryless sources. In particular, their outputs may be correlated over time, which can have a major impact on information rates. Handling more complicated sources typically requires heavy mathematical machinery, and is beyond the scope of this document. The results derived using more these are, nevertheless, similar in nature to the ones presented below. This partially explains why we choose not to study more difficult information sources in this document.

Having constructed a suitable abstraction for digital sources, we turn to the subject of digital information. From an intuitive point of view, the data rate of a discrete memoryless source should be equal to the amount of information it produces at every time instant. In other words, the amount of information created by a discrete memoryless source at time t should be computable based on $\mathcal{X}$ and $p_X(\cdot)$ exclusively. This is indeed the case. Before we can make this statement precise, we need a rigorous mathematical characterization of information. We address this issue by introducing entropy, a concept closely related to the notion of information.

2.2 Entropy

The **entropy** can be viewed as a measure of uncertainty in a random variable. In the context of digital communications, it provides a lower bound on the expected number of bits required to describe the output of a discrete memoryless source. As we will see shortly, this lower bound is tight and can be approached by practical encoders

Definition 2.2.1 (Entropy)**.** *Let X be a discrete random variable drawn from alphabet $\mathcal{X}$ according to probability mass function $p_X(\cdot)$. The entropy of X, denoted $\mathrm{H}[X]$, is given by*

$$\mathrm{H}[X] = -\sum_{x \in \mathcal{X}} p_X(x) \log_2(p_X(x)). \tag{2.2}$$

Under this definition, entropy is described in bits. When writing $\mathrm{H}[X]$, we use the convention

$$0 \cdot \log_2 \left(\frac{1}{0} \right) = \lim_{\epsilon \to 0} \epsilon \log_2 \left(\frac{1}{\epsilon} \right) = 0.$$

Alternatively, the entropy of X can be interpreted as the expectation of a logarithmic function,

$$\mathrm{H}[X] = \mathrm{E}\left[\log_2 \left(\frac{1}{p_X(X)} \right) \right].$$

The entropy as described in (2.2) has interesting properties. The value $\mathrm{H}[X]$ does not depend on the actual symbols themselves, it only depends on the probability mass function of the possible outcomes. For instance, in Example 2.2.2, the entropy of X remains the same whether we represent the flipping of a coin by a single bit or through a string of letters, heads or tails. More generally, the way we choose to designate the possible outcomes of a random experiment has no bearing over the entropy of the corresponding source, only the respective probabilities of the possible symbols matter.

Example 2.2.2. *Let X be an abstract representation of the flipping of a (possibly biased) coin. The probability mass function of X is then equal to*

$$p_X(x) = \begin{cases} (1-p), & x = 0 \\ p, & x = 1 \end{cases}$$

with zero denoting tails and one for heads. We can compute the entropy of X as follows,

$$\mathrm{H}[X] = -(1-p)\log_2(1-p) - p\log_2(p).$$

If the coin is fair, $p = \frac{1}{2}$, then the entropy of X becomes one bit. Hence, the minimum expected number of bits needed to describe the outcome of a fair coin toss is one. This seems quite reasonable.

The entropy of pair of two independent random variables is the sum of the individual entropies. Suppose that X is a vector random variable given by $X = (U, V)$, where U and V are independent. Then, we can write

$$p_X(x) = p_X((u,v)) = p_U(u)p_V(v)$$

and the entropy of X can be computed as

$$\begin{aligned} \mathrm{H}[X] &= -\sum_{x\in\mathcal{X}} p_X(x)\log_2(p_X(x)) \\ &= -\sum_{(u,v)\in\mathcal{U}\times\mathcal{V}} p_X((u,v))\log_2(p_X((u,v))) \\ &= -\sum_{u\in\mathcal{U}}\sum_{v\in\mathcal{V}} p_U(u)p_V(v)\log_2(p_U(u)p_V(v)) \\ &= -\sum_{u\in\mathcal{U}} p_U(u)\log_2(p_U(u)) - \sum_{v\in\mathcal{V}} p_V(v)\log_2(p_V(v)) \\ &= \mathrm{H}[U] + \mathrm{H}[V]. \end{aligned}$$

This corresponds to our intuitive understanding; the amount of information contained in two unrelated events should be the sum of the information pertaining to each individual event.

It is important to recognize that $\mathrm{H}[X]$ is computed based on the probability mass function $p_X(\cdot)$, it is not a function of the random variable X itself. As such, $\mathrm{H}[X]$ is a deterministic quantity and does not depend on the actual realization of X. Furthermore, we note that $\mathrm{H}[X]$ is continuous in the weights of the distribution $p_X(\cdot)$. A small change in the distribution of X only results in a small variation in its entropy. It is therefore possible to construct accurate entropy estimates based on empirical measurements of the source outputs.

2.3 Variable-Length Compression Codes

A **code** is a rule for converting a symbol (or a group of symbols) into a string of bits called a **codeword**. Mathematically, an encoder is a mapping $c : \mathcal{X} \mapsto \mathcal{C}$ from the input alphabet $\mathcal{X}$ to the collection of possible codewords $\mathcal{C}$. The goal of a **compression code** is, of course, to provide a more concise representation of the information signal. In lossless compression, the function c must be invertible when restricted to the support of X. Without this one-to-one relationship, decoding errors are guaranteed to happen. Encoding schemes can be partitioned into two categories based on the structure of their codebooks. If the codewords all share the same bit-length, then the corresponding code is called a **fixed-length code**. This section focuses on codes in the second category, variable-length codes, which are often used in lossless data compression.

As the name suggests, a **variable-length code** is an encoding function that maps source symbols to a variable number of bits. This is a beneficial feature for many compression schemes, as the greater flexibility sometimes leads to better compression ratio. The motivation behind variable-length encoding is the intuition that data compression can be achieved by assigning short bit strings to likely symbols, and necessarily longer bit strings to less probable ones. In dealing with variable-length codes, it is essential to recognize that they are inherently more tricky than fixed-length ones. With variable-length coding, it may be impossible to know where codewords begin in a compressed binary file without knowing the content of the file. This is in stark contrast with fixed-length codes where codewords are positioned at regular intervals and, therefore, easy to distinguish. To ensure that the binary output of a variable-length encoder can be recovered unambiguously, the code needs specific properties.

The extension of a code c is obtained by concatenating its codewords when c is applied to a multitude of source symbols. Given the string of symbols $x_1, x_2, \ldots, x_n$, the extension of c produces the output bit string

$$c(x_1)c(x_2)\cdots c(x_n).$$

An extension of c is a proper encoding scheme because it takes a group of symbols as its argument and produces a string of bits as its output.

Variable-length codes can be nested in order of decreasing generality as non-singular, uniquely decodable and instantaneous. A code is **non-singular** if each source symbol is mapped to a different bit string. That is, the mapping c from $\mathcal{X}$ to $\mathcal{C}$ is one-to-one. Rather, if two symbols map to the same codeword, then it is intuitively clear that the original message cannot be recovered with certainty. A code is said to be **uniquely decodable** if its extensions are non-singular.

It is important to recognize that successive codewords in a message are communicated as an undifferentiated sequence of bits. There is no separation marker or frame between adjacent codewords, no commas or spaces. The decoder, given a starting point, must infer the boundaries of every codeword from the data. This process is called *parsing*. The third and final property of variable-length encoding is related to parsing. A code is *instantaneous*, or **prefix-free**, if no codeword in $\mathcal{C}$ is a prefix of a any other encoded symbol in $\mathcal{C}$. This property guarantees that each encoded symbol can be identified with no further delay once the corresponding string of bits is received or read.

Example 2.3.1. *Suppose that a source produces three possible symbols, $\mathcal{X} = \{x_1, x_2, x_3\}$. We consider four encoding functions (c_1, c_2, c_3, c_4), each with different properties. The encoding schemes are defined as follows.*

Symbol	*Codeword*			
x	$c_1(x)$	$c_2(x)$	$c_3(x)$	$c_4(x)$
x_1	*0*	*0*	*0*	*0*
x_2	*1*	*1*	*01*	*10*
x_3	*0*	*01*	*11*	*11*

The first scheme is not injective because it maps two different source symbols to the same codeword, $c_1(x_1) = c_1(x_3)$. Thus, individual codewords cannot be decoded unambiguously. The second code is one-to-one; however, it is not uniquely decodable. The encoded message 01 can be generated by either input string $x_1 x_2$ or input symbol x_3. Clearly, the compressed message is not uniquely decodable. The third code, $c_3(\cdot)$ is uniquely decodable, but not instantaneous. After receiving a zero, it not immediately clear whether x_1 produced this output or if this zero consists of the first half of codeword $c(x_2)$. While $c_4(\cdot)$ is a prefix code where every symbol can be decoded immediately after reading the corresponding bits.

The measure of a good prefix code is the expected length of its encoded symbols. Suppose that a discrete memoryless source $(\mathcal{X}, p_X)$ is given along with a code c. We denote the length in bits of codeword $c(x)$ by $\ell_c(x)$. The expected number of bits produced by the source at each time instant is given by

$$\mathrm{E}[\ell_c(X)] = \sum_{x \in \mathcal{X}} p_X(x) \ell_c(x). \tag{2.3}$$

We emphasize that the expected length is a function of both the statistics of the source and the structure of compression code employed. Under the assumption that the source outcomes are independent and identically distributed over time, $\mathrm{E}[\ell_c(X)]$ also represents the average data rate produced by the source, in bits per source symbol.

2.3.1 Kraft Inequality

When building a compression code, it is obvious from (2.3) that assigning short codewords is better than long codewords. Yet, it is clear that we cannot describe every symbol using a very small number of bits, for otherwise the prefix condition will be violated. The collection of possible length assignments for a prefix-free code is characterized by the following inequality.

Theorem 2.3.2 (Kraft Inequality)**.** *Let $\mathcal{X}$ be a finite alphabet. Any binary prefix-free code $c : \mathcal{X} \mapsto \mathcal{C}$ satisfies the inequality*

$$\sum_{x \in \mathcal{X}} 2^{-\ell_c(x)} \leq 1. \tag{2.4}$$

where $\ell_c(x)$ is the bit length of codeword $c(x)$. Conversely, if we first assign the codeword lengths such that (2.4) *is satisfied, then there exists an instantaneous code with these codeword lengths.*

Proof. We wish to give necessary and sufficient conditions about the existence of a prefix-free code with a specific length assignment. We employ simple combinatorial arguments to a binary tree structure to establish this result. Let

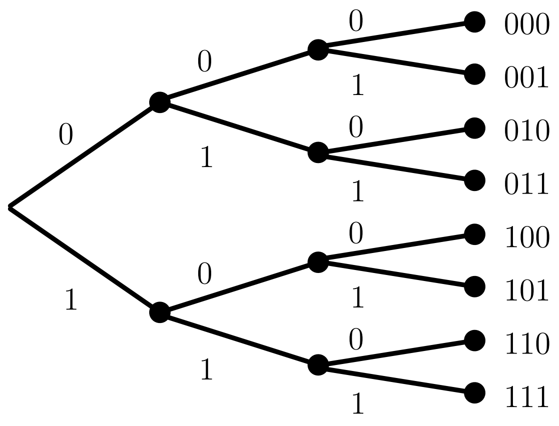

Figure 2.1: This figure shows a binary tree with depth $L_c = 3$ and eight leaves. The branches from every node correspond to zero or one.

$L_c = \max_{x \in \mathcal{X}} \ell_c(x)$ be the length of the longest codeword. Code $c : \mathcal{X} \mapsto \mathcal{C}$ can be defined using a binary tree of depth L_c, where branches from every node correspond either to zero or one. Each codeword consists of a unique path from the root to a leaf at depth $\ell_c(x)$, following its binary string expansion. The prefix condition ensures that no codeword is a descendant of any other codeword in the binary tree. For the codewords in the tree, let S_x be the set of descendants that $c(x)$ would have in a full binary tree of depth L_c. The sets S_x

are disjoint because of the prefix-free nature of the code, and $|S_x| = 2^{L_c - \ell_c(x)}$. Since the total number of nodes at depth L_c is 2^{L_c}, we have

$$\left| \bigcup_{x \in \mathcal{X}} S_x \right| = \sum_{x \in \mathcal{X}} |S_x| = \sum_{x \in \mathcal{X}} 2^{L_c - \ell_c(x)} \leq 2^{L_c}.$$

By dividing both sides by 2^{L_c}, we conclude that (2.4) holds. That is, a binary prefix-free code c over finite alphabet $\mathcal{X}$ satisfies the Kraft inequality.

Conversely, suppose that we have a code assignment such that (2.4) is satisfied. Without loss of generality, we assume that the codeword lengths $\ell_c(x_i)$ are increasing in i,

$$\ell_c(x_1) \leq \ell_c(x_2) \leq \ell_c(x_3) \leq \cdots$$

We can construct a prefix code with matching codeword lengths by pruning subtrees from a full binary tree of depth L_c. First, choose any node from the full tree at depth ℓ_1 and remove all of its descendants. This removes $2^{L_c - \ell_1}$ leafs from the original binary tree. Next, select any available node from the resulting tree at depth ℓ_2, and remove all of its descendants. This time, an additional $2^{L_c - \ell_2}$ leafs are taken away from the original binary tree. Continue this procedure with the other codeword lengths. After m iterations, the total number of leafs removed from the original binary tree is equal to

$$\sum_{i=1}^{m} 2^{L_c - \ell_i} = 2^{L_c} \sum_{i=1}^{m} 2^{-\ell_i}.$$

Since the Kraft inequality holds for the codeword length assignment, this implies that all the codewords can be placed at different positions on the binary graph. Then, following the binary structure of the graph, the binary string of the codes can be inferred from the graph. □

Example 2.3.3 (Code on a Tree). *Suppose that we intend to construct a prefix code for $\mathcal{X} = \{x_1, \ldots, x_5\}$, with code lengths*

$$\ell_c(x_1) = \ell_c(x_2) = 2 \qquad \ell_c(x_3) = \ell_c(x_4) = 3 \qquad \ell_c(x_5) = 2.$$

First, we check the Kraft inequality to make sure that such an assignment is feasible,

$$\sum_{i=1}^{5} 2^{-\ell_c(x_1)} = \frac{1}{4} + \frac{1}{4} + \frac{1}{8} + \frac{1}{8} + \frac{1}{4} = 1.$$

The inequality is fulfilled, we can therefore use a binary tree construction to design the desired instantaneous code. The process is illustrated in Figure 2.2, and the resulting code is shown below.

Source Symbol	*Codeword*	*Source Symbol*	*Codeword*
x_1	*00*	x_4	*101*
x_2	*01*	x_5	*11*
x_3	*100*		

Since the Kraft inequality is met with equality, we know that it is impossible to get a better code by shortening one of the codewords.

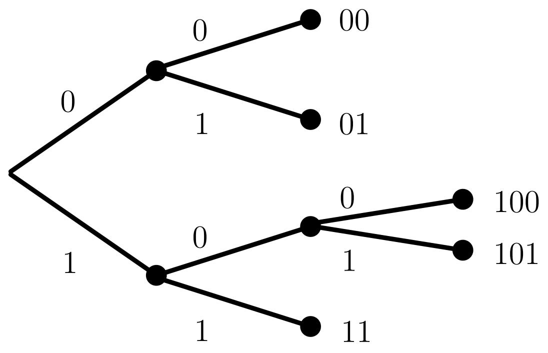

Figure 2.2: Construction of a prefix code with a binary tree.

2.3.2 Entropy Bounds on Prefix-Free Codes

Now that we know how to build instantaneous, we consider the problem of finding good prefix-free codes. Recall from (2.3) that our objective is to find a prefix-free code with the smallest possible expected length. As seen earlier, this codeword length assignment is subject to the Kraft inequality. Putting these two requirements together, we can formulate the optimization problem as follows,

$$\min_{\ell(x)} \sum_{x \in \mathcal{X}} p_X(x)\ell(x) \quad \text{subject to} \quad \sum_{x \in \mathcal{X}} 2^{-\ell(x)} \leq 1.$$

We note that, for a code to exist, the function $\ell(x)$ must take values in the positive integers. It turns out that this problem is difficult to solve.

To gain insight into the problem, we relax the integer constrain on $\ell(x)$. This added flexibility will provide a lower bound on $\mathrm{E}[\ell(X)]$; having more choices can only lead to better results. We use the method of Lagrange multipliers to solve the latter version of the problem. The objective function, with Lagrange multiplier λ, becomes

$$\sum_{x \in \mathcal{X}} p_X(x)\ell(x) + \lambda \left(\sum_{x \in \mathcal{X}} 2^{-\ell(x)} - 1 \right).$$

Note that this function is twice differentiable in $\ell(x)$. Taking a partial derivative with respect to $\ell(x)$ and setting it to zero, we get

$$p_X(x) - \lambda \ln(2) 2^{-\ell(x)} = 0.$$

The optimal value for $\ell(x)$, which we denote by $\ell^{\star}(x)$, must therefore satisfy $2^{-\ell^{\star}(x)} = p_X(x)/(\lambda \ln(2))$. Computing the derivative with respect to λ yields

$$\sum_{x \in \mathcal{X}} 2^{-\ell^{\star}(x)} = 1,$$

which in turn implies $\lambda = 1/\ln(2)$. Putting these results together, we gather that the optimal values for $\{\ell(x) : x \in \mathcal{X}\}$ are given by

$$\ell^{\star}(x) = -\log_2(p_X(x))$$

for $x \in \mathcal{X}$. Thus, by construction, we obtain

$$\mathrm{E}[\ell_c(X)] \geq - \sum_{x \in \mathcal{X}} p_X(x) \log_2(p_X(x)) = \mathrm{H}[X]$$

for any prefix-code c. In other words, the entropy is a lower bound on the expected length of any prefix-free code.

It is equally easy to obtain an upper bound on the expected length of an optimal prefix-free code. Observe that $\lceil -\log_2(p_X(x)) \rceil$ is an integer, with

$$-\log_2(p_X(x)) \leq \lceil -\log_2(p_X(x)) \rceil \leq -\log_2(p_X(x)) + 1.$$

The Kraft inequality asserts that we can build a code $c : \mathcal{X} \mapsto \mathcal{C}$ such that $\ell_c(x) = \lceil -\log_2(p_X(x)) \rceil$, as

$$\sum_{x \in \mathcal{X}} 2^{-\ell_c(x)} \leq \sum_{x \in \mathcal{X}} 2^{-\ell^{\star}(x)} = 1.$$

As such, there exists a code c such that

$$\mathrm{E}[\ell_c(X)] \leq \mathrm{H}[X] + 1.$$

We collect and formalize these important results in the form of a theorem.

Theorem 2.3.4. *Consider a discrete memoryless source $(\mathcal{X}, p_X(\cdot))$ over a finite alphabet. If symbols are encoded individually using an optimal prefix-free code $c : \mathcal{X} \mapsto \mathcal{C}$, then the expected length of the codewords satisfies*

$$\mathrm{H}[X] \leq \mathrm{E}[\ell_c(X)] \leq \mathrm{H}[X] + 1.$$

2.3.3 Huffman Codes

Theorem 2.3.4 identifies properties of an optimal prefix-code. However, it does not provide an algorithmic methodology to design such a code. This is addressed by the **Huffman code**, which provides an efficient variable-length code for lossless data compression. Not too surprisingly, the underlying strategy in this scheme is to assign short strings of bits to likely symbols, and necessarily longer ones to less probable source outputs. The encoding is specifically crafted so that the code table forms a prefix-free code. Huffman codes are the most efficient compression mapping for individual source symbols. The expected length of the compressed data achieved with this technique will be no greater than the expected message length of any other prefix-free code that operates on individual source symbols.

The insight behind Huffman coding is based on three properties of optimal prefix-codes. Suppose that we wish to encode outputs from discrete memoryless source $(\mathcal{X}, p_X)$, and let $c^\star : \mathcal{X} \mapsto \mathcal{C}$ be an optimal prefix-code for this source. If $p_X(x_1) > p_X(x_2)$, then $\ell_{c^\star}(x_1) \leq \ell_{c^\star}(x_2)$. For any of the longest codewords, its sibling (the bit string that differs only in the last digit) must also be codeword; otherwise, the original codeword can be shortened by removing the last bit. Finally, the code tree associated with an optimal code must be full. A binary tree is *full* if every node has either zero or two children. Again, if this condition fails, then some codewords in the codebook can be shortened.

The Huffman algorithm creates a code by building a binary tree. The algorithm proceeds as follows. First, every source symbol x is assigned to an individual node. Then, the simple recursion outlined below is applied.

1. Sort the nodes in decreasing order of probabilities.
2. Merge the two least probable nodes into a single one, whose probability equals the sum of its constituents.
3. Arbitrarily assign zero or one to the branches emerging from this new node.
4. Repeat the previous three steps with the new collection of nodes and their corresponding probabilities until only one node remains.

The Huffman encoding algorithm is best grasped through simple examples.

Example 2.3.5. *Suppose that a discrete memoryless source $(\mathcal{X}, p_X)$ with alphabet $\mathcal{X} = \{x_1, x_2, x_3\}$ has probability mass function*

$$p_X(x) = \begin{cases} \frac{2}{3}, & x = x_1 \\ \frac{1}{6}, & x \in \{x_2, x_3\}. \end{cases}$$

We wish to obtain an optimal prefix-code for this source, and thus we apply the Huffman algorithm. For the given source, code design proceeds as follows.

Stage 3	*Stage 2*	*Stage 1*	*Symbol*	*Codeword*
$\Pr\{x_1, x_2, x_3\} = 1$	$\Pr\{x_1\} = \frac{2}{3}$	$p_X(x_1) = \frac{2}{3}$	x_1	*0*
	$\Pr\{x_2, x_3\} = \frac{1}{3}$	$p_X(x_2) = \frac{1}{6}$	x_2	*10*
		$p_X(x_3) = \frac{1}{6}$	x_3	*11*

From this successive re-ordering of probabilities, we use a binary tree to build the actual code. This is illustrated in Figure 2.3.

Example 2.3.6. *A source $(\mathcal{Y}, p_Y)$ generates four different symbols $\{y_1, y_2, y_3, y_4\}$ with probabilities $\{0.35, 0.25, 0.2, 0.2\}$. A binary tree is generated from right to left, by merging the two less probable symbols at every step. Once this is complete, the code can then be form by assigning different bits to every pair of branches emerging from a node. The table below shows the different stages of the iterative procedure where probabilities are first sorted in decreasing order, then the two least probable nodes are merged into a single one.*

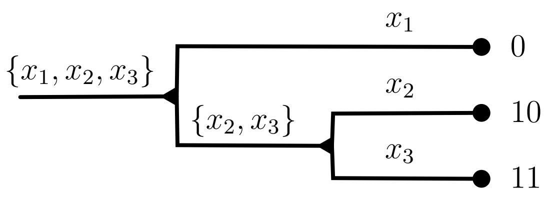

Figure 2.3: A graphical representation for the construction of a simple Huffman code. The source alphabet in this case is $\mathcal{X} = \{x_1, x_2, x_3\}$ and the probabilities of individual symbols are $\{2/3, 1/6, 1/6\}$, respectively.

Stage 4	*Stage 3*	*Stage 2*	*Stage 1*
$0.6 + 0.4 = 1$	$0.35 + 0.25 = 0.6$	$0.2 + 0.2 = 0.4$	$p_X(x_1) = 0.35$
	0.4	0.35	$p_X(x_2) = 0.25$
		0.25	$p_X(x_3) = 0.2$
			$p_X(x_4) = 0.2$

The ensuing Huffman code is obtained by moving from left to right in the corresponding binary tree. The binary tree and the resulting Huffman code are shown in Figure 2.4.

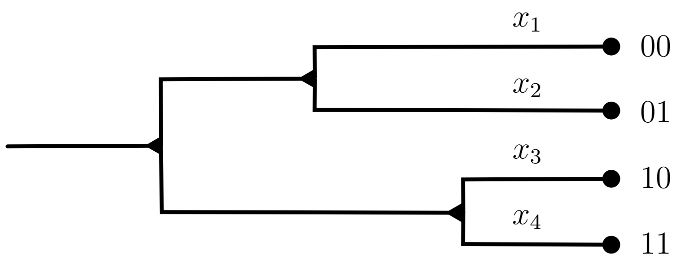

Figure 2.4: This figure depicts a Huffman code construction for an alphabet of size four.

Although Huffman coding is optimal for a symbol-by-symbol encoding with a known input probability mass function, it can be outperformed when these two conditions are not known. For instance, if the input distribution $p_X(\cdot)$ is not known, then it must be inferred from the available data prior to applying Huffman coding. Small errors in the estimated probability mass function can then lead to inefficiency, which in turn renders Huffman coding suboptimal.

We will soon see an encoding algorithm that does not require the input distribution $p_X(\cdot)$. However, before we can present this algorithm, we need to consider the join encoding of source symbols.

2.4 Joint Encoding of Source Symbols

Under special circumstances, namely when the probability of every source symbol is an exponent base two, the expected length of a Huffman code is equal to the entropy of the source. However, in many situations, this is not the case, and there exists a gap between the expected codeword length and the entropy of a source output. An efficient way to encode data, where the expected number of coded bits per source symbol approaches the entropy, is to consider blocks of source symbols and to encode them jointly. Although more complicated, this process leads to better performance and typically leads to expected message lengths that are shorter than that of a symbol-by-symbol Huffman code.

Consider a sequence $X_1, X_2, \ldots$ of symbols at the output of a discrete memoryless source. Instead of using a code that operates on individual symbols, we can design a more elaborate code that takes as input a group of n symbols, $c : \mathcal{X}^n \mapsto \mathcal{C}$. Since the outputs of a discrete memoryless source are independent and identically distributed random variables, we know from the additive property of the entropy that

$$\mathrm{H}[X_1, \ldots, X_n] = n\mathrm{H}[X].$$

Then, by applying Theorem 2.3.4, we get that an optimal prefix code, which operates on $\mathcal{X}^n$, yields

$$n\mathrm{H}[X] \leq \mathrm{E}[\ell_c(X_1, \ldots, X_n)] \leq n\mathrm{H}[X] + 1.$$

Then, the expected message length per source output becomes

$$\mathrm{H}[X] \leq \frac{\mathrm{E}[\ell_c(X_1, \ldots, X_n)]}{n} \leq \mathrm{H}[X] + \frac{1}{n}.$$

Thus, the expected number of bits per symbol produced by a source can be made arbitrarily close to $\mathrm{H}[X]$ by jointly encoding strings of symbols.

Example 2.4.1. *Let $(\mathcal{X}, p_X)$ be a binary discrete memoryless source, as described in Example 2.1.2. Furthermore, assume that Bernoulli parameter p is equal to $\frac{1}{4}$. Then, the entropy of the source can be calculated as*

$$\mathrm{H}[X] = -\frac{3}{4}\log_2\left(\frac{3}{4}\right) - \frac{1}{4}\log_2\left(\frac{1}{4}\right) \approx 0.811.$$

Since there are only two source symbols, a code generated by the Huffman algorithm is the identity code, where a source output is represented by its binary value. In this case, the expected codeword length is equal to one.

Suppose instead that two symbols are encoded at a time. In this case, the possible inputs to the encoder are $\{00, 01, 10, 11\}$, with respective probabilities $\left\{\frac{9}{16}, \frac{3}{16}, \frac{3}{16}, \frac{1}{16}\right\}$. The Huffman code specified by

Symbol	*Codeword*
00	*0*
01	*10*
10	*110*
11	*111*

has an expected length of

$$\mathrm{E}[\ell_c(X_1, X_2)] = 1 \cdot \frac{9}{16} + 2 \cdot \frac{3}{16} + 3 \cdot \frac{3}{16} + 3 \cdot \frac{1}{16} = \frac{27}{16}.$$

The expected message length per source output becomes

$$\frac{\mathrm{E}[\ell_c(X_1, X_2)]}{2} = \frac{27}{32} \approx 0.844,$$

which is much closer to the entropy of individual source symbols. Repeating this procedure with an Huffman code that takes three symbols as its input would lead to an expected codeword length per symbol of approximately 0.823.

Example 2.4.1 illustrates well how encoding several source symbol at a time can lead to a decrease in the expected codeword length per symbol. The joint encoding of source symbols works even better for sources that are correlated over time. Although we will not discuss the specifics of this scenario, it is informative to mention that joint source coding is instrumental in approaching the entropy rate of correlated sources. This is especially important considering the fact that symbol-by-symbol encoding may perform very poorly for correlated sources.

2.5 Sources with Memory

Until now, we have focused on the compression of discrete memoryless sources because they produce i.i.d. symbols. The majority of real-world sources generate symbol sequences where the probability of the next symbol depends on the previously observed symbols. To model these sources, we use the mathematical concept of a **discrete-time random process**.

A discrete-time random process is an infinite sequence $\dots, X_{-1}, X_0, X_1, \dots$ of random variables defined on a common sample space Ω. This approach treats each random variable X_n as a mapping $X_n : \Omega \to \mathbb{R}$. Using this, each $\omega \in \Omega$ defines a realization $\dots, X_{-1}(\omega), X_0(\omega), X_1(\omega), \dots$ of the random process. While random processes can be quite complicated in their full generality, the random processes used in this class have relatively simple descriptions. For example, the discrete memoryless source introduced earlier in this chapter defines a very simple random process.

In this section, we introduce a simple random process with memory known as a **Markov chain**. A Markov chain assumes that the probability of the n-th symbol depends only on the value of the previous symbol. Mathematically, this condition can be written as

$$P_{X_n|X_{n-1},X_{n-2},\dots,X_0}(x_n|x_{n-1},x_{n-2},\dots,x_0) = P_{X_n|X_{n-1}}(x_n|x_{n-1}).$$

If the conditional probability is time-invariant (i.e., it doesn't depend on n), then the Markov chain is called **stationary**. Mathematically, stationarity implies that

$$P_{X_n|X_{n-1}}(x_n|x_{n-1}) = P_{X_1|X_0}(x_n|x_{n-1}).$$

For a stationary Markov chain, the probability simplifies to

$$P_{X_n,X_{n-1},\dots,X_0}(x_n,x_{n-1},\dots,x_0) = P_{X_0}(x_0)\prod_{i=1}^{n} P_{X_1|X_0}(x_i|x_{i-1}).$$

2.6 Universal Source-Coding Algorithms

Huffman coding has two important drawbacks. First, the source statistics are used to design a Huffman code. If one only has access to the source outputs, the design procedure requires two passes through the data, one to estimate the statistics of the source, and a second one for encoding. To overcome this, one can use adaptive Huffman codes where the code is updated dynamically to match the statistics of the sequence as it is observed. This is a problem because The second problem is that one must jointly encode multiple symbols to take advantage of source memory and reduce length rounding loss. In this case, one finds that the complexity increases exponentially with the number of symbols that are encoded together. To provide a partial solution to these drawbacks, we study an example of a universal source-coding algorithm, namely the Lempel-Ziv algorithm. This type of **universal data compression** is the basis for standard file compression algorithms (e.g., winzip, gzip).

The basic idea behind the **Lempel-Ziv algorithm** is to parse the input sequence into non-overlapping strings of different lengths while constructing a dictionary of the strings seen thus far. There are many versions of this algorithm and we discuss the variant known as LZ78 that was described in a 1978 paper by Lempel and Ziv. The encoding algorithm works as follows. First, initialize the dictionary to contain all strings of length one and set the input pointer to the beginning of the string. Then, apply the following iterative procedure.

1. Starting at the input pointer, find the longest substring w that is already in the dictionary.

2. Concatenate w with the next symbol y in the string and add wy to the first empty location in the dictionary.

3. Encode the pair by sending the dictionary index of w and the value of y.

4. Set the input pointer to the symbol after y.

There are a number of practical variants of this algorithm that improve performance and/or reduce the implementation complexity.

Decompression works in the reverse fashion. Each received index and symbol can be immediately decoded and used to build a copy of the dictionary at the receiver. In this fashion, one can resolve the input without ambiguity.

Example 2.6.1. *Suppose that we are to use a Lempel-Ziv algorithm with dictionary size* $2^3 = 8$*. The dictionary is initialized to contain* 0 *and* 1 *in the first* *two positions. Then, the source sequence is sequentially parsed into strings that have not appeared so far. For example,*

$$10110101000101\ldots \rightarrow 10, 11, 01, 010, 00, 101\ldots$$

The dictionary table at this point has eight elements.

Index	*Dictionary String*	*Encoded Index*	*Added Bit*
000	*0*	*N/A*	*N/A*
001	*1*	*N/A*	*N/A*
010	*10*	*001*	*0*
011	*11*	*001*	*1*
100	*01*	*000*	*1*
101	*010*	*100*	*0*
110	*00*	*000*	*0*
111	*101*	*010*	*1*

Each phrase (the bit string contained between two commas) is coded by giving the location of its prefix in the dictionary table, and the value of the additional bit. This results in the coded sequence

$$10, 11, 01, 010, 00, 101 \rightarrow (001, 0)(001, 1)(000, 1)(100, 0)(000, 0)(010, 1),$$

where the first number of each pair gives the index of the prefix in the table and the second number gives the last bit of the new phrase. When applied to sequences generated by any stationary ergodic source, the Lempel-Ziv coding algorithm asymptotically achieves the optimal encoding rate (known as the entropy rate).

Most readers will notice that this algorithm, as stated, requires prior knowledge of the total number of phrases in the dictionary. In fact, this problem can be solved easily and the solution actually requires fewer transmitted bits. The key point is that both the transmitter and receiver know the number of phrases currently in the dictionary. Let M be the current number of phrases in the dictionary. Then, the transmitter can be simply send the $\lceil \log_2 M \rceil$ least significant bits of the index. Since the receiver also knows M, there will be no confusion. In this case, the encoded sequence will be

$$10, 11, 01, 010, 00, 101 \rightarrow (1, 0)(01, 1)(00, 1)(100, 0)(000, 0)(010, 1).$$

2.7 Fixed-Length Compression Codes*

In the previous sections, the joint encoding of multiple source symbols was shown to perform well, with the average number of bits per symbol produced by a source approaching $\mathrm{H}[X]$. Below, we explore how the joint encoding of symbols together with fixed-length codes can be used to produce good compression ratios. Fixed-length compression codes have several advantages. They are simple to encode and easy to decode, yielding unambiguous messages. Furthermore, all fixed-length codes are prefix-free, and encoded symbols can therefore be recovered instantaneously. However, fixed-length codes cannot be used to compress data by assigning short descriptions to most frequent symbols and longer descriptions to the less likely ones. Data compression in fixed-length coding methods is only possible for large blocks of data, and any compression beyond the logarithm of the total number of possibilities comes with a finite, though perhaps small, probability of decoding failure.

The minimum number of binary strings in lossless fixed-length symbol-by-symbol encoding is $\lceil \log_2(|\mathcal{X}|) \rceil$, where $|\mathcal{X}|$ is the size of the source alphabet and $\lceil \cdot \rceil$ is the ceiling function, which returns the smallest integer greater than or equal to its argument. More generally, the minimum number of binary strings necessary to encode a group of n symbols is $\lceil n \log_2(|\mathcal{X}|) \rceil$. This strategy alone, encoding multiple source symbols at a time, is not powerful enough to compress data using fixed-length codes. To design effective fixed-length codes, two components are necessary. First, we need to relax the assumption that the data compression scheme should be lossless, rather we allow a small probability of encoding failure. In particular, we assume that the probability of encoding failure, where data cannot be decoded properly, is $\delta > 0$, where δ is implicitly very small. The second ingredient to fixed-length compression is the asymptotic equipartition property, which we review next.

2.7.1 Asymptotic Equipartition Property

The **asymptotic equipartition property** (**AEP**) is a general property of the output samples of discrete memoryless sources. This property implies that, given a very long sequence of n source symbols, the probability that

$(X_1, \ldots, X_n)$ belongs to a set of typical sample strings is almost one. It takes a few steps to make this statement precise.

Theorem 2.7.1. *Let $(\mathcal{X}, p_X)$ be a discrete memoryless source, which produces a sequence of symbols $X_1, X_2, \ldots$ Furthermore, assume that the output alphabet $\mathcal{X}$ is finite. The asymptotic equipartition probability asserts that*

$$\lim_{n \to \infty} -\frac{1}{n} \log_2 \left(p_{X^n}(X_1, \ldots X_n) \right) = \mathrm{H}[X].$$

Proof. We can proof this theorem through an application of the weak law of large numbers. First, we observe that

$$\begin{aligned} \log_2 \left(p_{X^n}(X_1, \ldots X_n) \right) &= \log_2 \left(\prod_{k=1}^{n} p_X(X_k) \right) \\ &= \sum_{k=1}^{n} \log_2 p_X(X_k). \end{aligned}$$

That is, $\log_2 \left(p_{X^n}(X_1, \ldots X_n) \right)$ is a sum of independent and identically distributed random variables, with bounded second moment. It follows, by the law of large numbers, that

$$-\frac{1}{n} \log_2 \left(p_{X^n}(X_1, \ldots X_n) \right)$$

converges in probability to $\mathrm{E}\left[-\log_2(p_X(X)) \right] = \mathrm{H}[X]$. In particular, we have

$$\Pr\left(\left| \frac{-\log_2 \left(p_{X^n}(X_1, \ldots X_n) \right)}{n} - \mathrm{H}[X] \right| \geq \epsilon \right) \leq \frac{\sigma^2}{n\epsilon^2}$$

where σ^2 is the variance of random variable $-\log_2(p_X(X))$. □

Drawing intuition from the proof of Theorem 2.7.1, we define the **typical set** $T_\epsilon^{(n)}$ as

$$T_\epsilon^{(n)} = \left\{ \mathbf{x} \in \mathcal{X}^n : \left| \frac{-\log_2 \left(p_{X^n}(\mathbf{x}) \right)}{n} - \mathrm{H}[X] \right| < \epsilon \right\}.$$

The probability that the first n source symbols belongs to the typical set $T_\epsilon^{(n)}$ is bounded below by

$$\Pr\left((X_1, \ldots, X_n) \in T_\epsilon^{(n)} \right) \geq 1 - \frac{\sigma^2}{n\epsilon^2}. \tag{2.5}$$

Thus, as n increases, the probability that the source produces a typical sequence approaches one. We note that an equivalent definition of typical set is

$$T_\epsilon^{(n)} = \left\{ \mathbf{x} \in \mathcal{X}^n : 2^{-n(\mathrm{H}[X]+\epsilon)} < p_{X^n}(\mathbf{x}) < 2^{-n(\mathrm{H}[X]-\epsilon)} \right\}.$$

Using the second definition of $T_\epsilon^{(n)}$, we can bound the number of elements contained in a typical set. First, recall that the sum of the probability of disjoint events cannot exceed one. As a consequence, the number of elements in $T_\epsilon^{(n)}$ is bounded by

$$\left| T_\epsilon^{(n)} \right| < 2^{n(\mathrm{H}[X]+\epsilon)}.$$

Similarly, using (2.5) and the second definition of $T_\epsilon^{(n)}$, we get

$$\left| T_\epsilon^{(n)} \right| > \left(1 - \frac{\sigma^2}{n\epsilon^2} \right) 2^{n(\mathrm{H}[X]-\epsilon)}.$$

We collect these results in the following theorem.

Theorem 2.7.2 (Asymptotic Equipartition Property)**.** *Let $(\mathcal{X}, p_X)$ be a discrete memoryless source with finite alphabet $\mathcal{X}$ and output sequence $X_1, X_2, \ldots$, each with entropy $\mathrm{H}[X]$. For any $\delta > 0$ and all n sufficiently large, we have*

$$\Pr\left((X_1, \ldots, X_n) \in T_\epsilon^{(n)} \right) \geq 1 - \delta$$

and the size of the typical set $T_\epsilon^{(n)}$ is bounded by

$$(1-\delta) 2^{n(\mathrm{H}[X]-\epsilon)} < \left| T_\epsilon^{(n)} \right| < 2^{n(\mathrm{H}[X]+\epsilon)}.$$

The intuition behind the asymptotic equipartition property is that a compression scheme can focus on encoding only the symbol strings that belong to $T_\epsilon^{(n)}$. Under such a strategy, at most $\lceil n(\mathrm{H}[X]+\epsilon) \rceil$ codewords are needed. Although not lossless, this fixed-length coding scheme results in a decoding failure with a probability no greater than δ.

Theorem 2.7.3 (Source Coding Theorem)**.** *Let $(\mathcal{X}, p_X)$ be a discrete memoryless source with finite alphabet $\mathcal{X}$ and entropy $\mathrm{H}[X]$. For any $\delta > 0$, $\epsilon > 0$ and n sufficiently large, there exists a fixed-length compression scheme such that the probability of failure is less than δ and the expected number of bits per symbol is*

$$\frac{\mathrm{E}[\ell_c(X_1, \ldots, X_n)]}{n} \leq \mathrm{H}[X] + \epsilon + \frac{1}{n}.$$

Chapter 3

Discrete-Time Communication

Most digital communication systems operate by converting digital data into continuous waveforms that can be conveyed through some physical medium to a receiver. For example, digital communication through wires (e.g., Ethernet or USB) is based on moving electrons back and forth in the wire. In contrast, underwater wireless communication uses acoustic transmission through the water. While radio communication relies on the propagation of electromagnetic waves through air.

The process by which a string of bits is converted into a waveform suitable for transmission is known as **modulation**. The reverse operation, called **demodulation**, is performed at the destination and involves extracting the information symbols from the received signal. The mapping between the transmitted waveform and the received waveform is known as the **channel**.

Precise models of the physical channel can be very complicated and many of the key ideas in digital communication do not depend on the exact details. For this reason, one can model and design communication systems based a simplified model that separates the communication problem from the physical models. In this chapter, we develop some basic concepts of digital communication using this simplified model. The goal is to build some intuition about how things work without getting lost in the mathematical details.

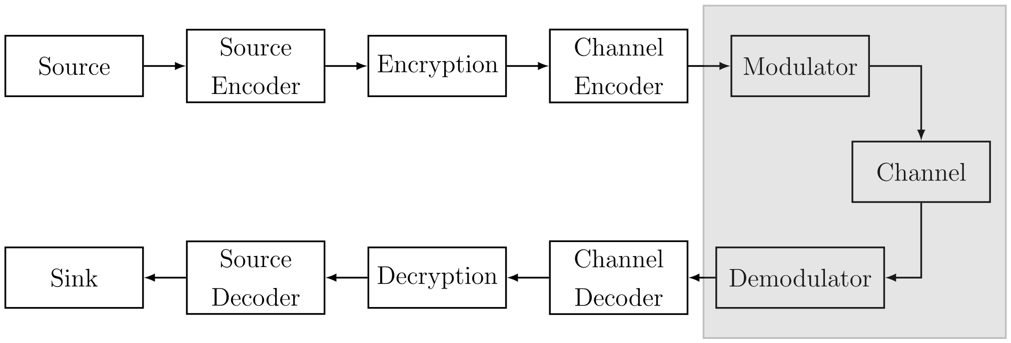

Figure 3.1: The block diagram of a digital communication system where the blocks comprising the discrete-time channel are shaded.

3.1 A Simple Channel Model

In this section, we introduce the **discrete-time channel model** for digital communication systems. We will see later that, for some communication systems, this model is exactly equivalent to the more complicated waveform model. The first channel model we discuss is where the nth output of the channel, Y_n, is equal to the nth input to the channel, x_n, corrupted by an additive noise term Z_n so that

$$Y_n = x_n + Z_n.$$

For this model, it is typical to assume that the noise sequence $Z_1, Z_2, \ldots$ consists of independent identically distributed (i.i.d.) zero-mean Gaussian random variables with variance σ^2. This implies that each Y_n is a Gaussian random variable with mean x_n and variance σ^2, so that

$$f_{Y_n}(y_n) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-(y_n - x_n)^2/(2\sigma^2)}.$$

This model is commonly referred to as discrete-time communication in **additive white Gaussian noise** (AWGN) noise.

As an example, consider a system where the transmitter sets the voltage on end of a wire and the receiver measures the voltage on the other end of the wire. It turns out that the thermal agitation of electrons causes voltage fluctuations known as Johnson noise. So, the receiver ends up measuring the transmitted voltage corrupted by noise fluctuations. In fact, Johnson noise is quite well approximated by AWGN. So, the simple model described above already gives a relatively accurate picture of reality.

3.2 A Simple Modulation Scheme

Once a channel model has been defined, the next step is choosing how to transmit digital data through the channel. A common approach is to choose a small set $\mathcal{U}$ of information symbols and represent each one by a distinct channel input in the set $\mathcal{X}$. This is the discrete-time model of a scheme known as **pulse-amplitude modulation** (PAM). Let $u_n \in \mathcal{U}$ be the information symbol transmitted during the nth time interval and $x_n = M(u_n)$ be the nth input to the channel, where $M : \mathcal{U} \to \mathcal{X}$ is called the symbol mapping function. For example, one can transmit binary information symbols by mapping "0" to +1V and "1" to −1V; mathematically, this is done by choosing $\mathcal{U} = \{0, 1\}$, $\mathcal{X} = \{-1, 1\}$, and $M(u) = 1 - 2u$. This particular type of PAM is called **binary phase-shift keying** (BPSK) or 2-PAM.

Suppose a BPSK signal is transmitted through our discrete-time AWGN channel model. The detector must measure the voltage and decide whether a 0 or 1 one was transmitted. A natural choice is to define a decoder function that associates positive voltages with 0 and negative voltages with 1. Let $\hat{U}_n = D(Y_n)$ be the output of the detector function

$$D(y) = \begin{cases} 0 & \text{if } y \geq 0 \\ 1 & \text{if } y < 0 \end{cases}.$$

This detector is optimal if 0's and 1's are transmitted with equal probability.

One of the main challenges in communication systems is providing reliable data transmission. In this example, the noise variance σ^2 is proportional to the power of the thermal noise in the wire. Notice that, if a 1 is transmitted, the detector described above will make an incorrect decision with probability

$$\Pr(Y_n \geq 0 | x_n = 1) = \Pr(Z_n \geq 1) = \int_1^\infty \frac{1}{\sqrt{2\pi\sigma^2}} e^{-y^2/(2\sigma^2)} dy = Q\left(\frac{1}{\sigma}\right).$$

This probability can be reduced by increasing the transmitted voltage, which increases the power dissipated due to resistive losses, or by reducing the ther-

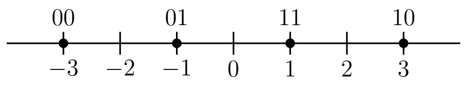

Figure 3.2: The symbol set $\mathcal{X}$ for 4-PAM with gray coded binary labels.

mal noise. For this reason, receivers used for large satellite dish receivers often use liquid nitrogen to cool the first-stage amplifier to reduce this thermal noise. On the other hand, the maximum transmitted power is typically limited by physical constraints (e.g., the wire thickness) or FCC regulations.

To send more information per channel use, one can use larger sets of PAM symbols. For example, 4-PAM uses the 4 symbols $\mathcal{X} = \{-3, -1, 1, 3\}$ while 8-PAM uses the 8 symbols $\mathcal{X} = \{-7, -5, -3, -1, 1, 3, 5, 7\}$. Notice that these two sets of symbols are centered around 0 to minimize the transmitted energy.

The mapping function $M(\cdot)$ determines the information symbol associated with each channel input value. In general, sets of input values with 2^m elements are associated with binary strings. There are still some choices to be made, however, because one can map the 4-PAM symbol set to binary strings in either the standard binary order $\{00, 01, 10, 11\}$ or with a Gray code $\{00, 01, 11, 10\}$.

For the 4-PAM symbol set with the mapping function $M(\cdot)$ that maps $\{00, 01, 10, 11\}$ (in order) to $\{-3, -1, 1, 3\}$, the natural decision function is

$$D(y) = \begin{cases} 00 & \text{if } y < -2 \\ 01 & \text{if } -2 \leq y < 0 \\ 10 & \text{if } 0 \leq y < 2 \\ 11 & \text{if } y \geq 2 \end{cases}.$$

3.3 Quadrature Amplitude Modulation

In most communication systems, the baseband waveform is modulated onto a high-frequency carrier to enable better propagation. This is because low-frequency signals often do not propagate well through physical media. While this process will be discussed later in more detail, the following key detail affects the discrete-time model. High frequency modulation allows two independent signals to be modulated onto the same carrier frequency; one onto the sine wave and the other onto the cosine wave. This allows one to treat the

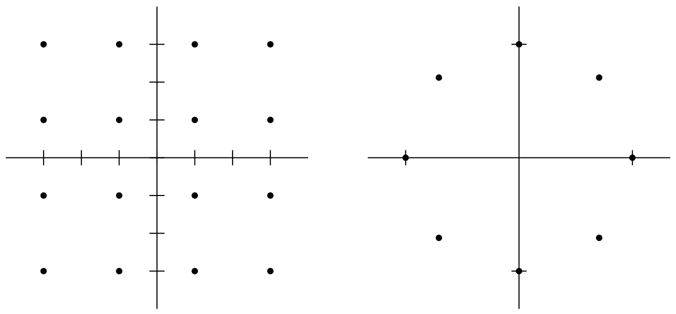

Figure 3.3: The symbol constellations $\mathcal{X}$ for 16-QAM (left) and 8-PSK (right).

transmitted value x_n and received value Y_n as points in 2-dimensional space. The set $\mathcal{X}$ of possible transmitted points in 2-dimensional space in called the **symbol constellation**.

For mathematical convenience, points in these two-dimensional symbol constellations are represented by complex numbers. The set of complex numbers is $\mathbb{C}$ and the constellation is a subset $\mathcal{X} \subset \mathbb{C}$. Likewise, the transmitted symbol is $x_n \in \mathbb{C}$ and the received value is $Y_n \in \mathbb{C}$. The noise term Z_n now consists of two i.i.d. Gaussian random variables (one in each direction). The probabilistic observation model is formed by treating the real and imaginary parts separately, and is given by

$$\begin{aligned} f_{Y_n^{(r)},Y_n^{(i)}}(y_n^{(r)}, y_n^{(i)}) &= \left(\frac{1}{\sqrt{2\pi\sigma^2}} e^{-\left(y_n^{(r)}-x_n^{(r)}\right)^2/(2\sigma^2)}\right)\left(\frac{1}{\sqrt{2\pi\sigma^2}} e^{-\left(y_n^{(i)}-x_n^{(i)}\right)^2/(2\sigma^2)}\right) \\ &= \frac{1}{2\pi\sigma^2} e^{-|y_n-x_n|^2/(2\sigma^2)}. \end{aligned}$$

Therefore, the probability of receiving a y_n value is simply a function of its Euclidean distance $|y_n - x_n|^2$ from the actual transmitted symbol. This leads to a nice geometric characterization of the optimal decision regions for the detector.

The **signal-to-noise ratio** (SNR) of a communication system is typically denoted by E_s/N_0 where E_s is the average **energy per channel input symbol** and N_0 is the **noise spectral density**. For equiprobable signaling, one finds that

$$E_s = \frac{1}{|\mathcal{X}|} \sum_{x \in \mathcal{X}} |x|^2.$$

The noise spectral density measures how much the AWGN is affects the channel and later we will see that $N_0 = 2\sigma^2$ for our discrete-time model.

Constellations are typically defined by first choosing the set of channel input values $\mathcal{X}$, and then choosing the mapping function $M : \mathcal{U} \to \mathcal{X}$. This second step is called **labeling** the constellation. While the labeling does not affect the symbol error rate of the system, it generally does affect the bit error rate. Therefore, one can optimize the mapping function for a particular application.

Constellations can be chosen and optimized for a variety of reasons. Still, there are a few very common choices:

- M-ary PAM (M-PAM) is M points equally spaced along a line or
$$\mathcal{X} = \left\{2a - (M-1) \,\middle|\, a \in \{0, 1, \ldots, M-1\}\right\} \subset \mathbb{C}.$$
- M^2-ary QAM (M^2-QAM) is an M by M square grid of points or
$$\mathcal{X} = \left\{(2a - (M-1)) + (2b - (M-1))\, i \,\middle|\, a, b \in \{0, 1, \ldots, M-1\}\right\} \subset \mathbb{C}.$$
- M-ary PSK (M-PSK) is M points equally spaced around a circle or
$$\mathcal{X} = \left\{e^{2\pi i k/M} \,\middle|\, k \in \{0, 1, \ldots, M-1\}\right\} \subset \mathbb{C}.$$

Example 3.3.1. *The standard QAM constellation with 16 points (known as 16-QAM) is given by*
$$\mathcal{X} = \left\{a + bi \,\middle|\, a, b \in \{-3, -1, 1, 3\}\right\} \subset \mathbb{C}.$$
The average energy of this constellation is given by
$$E_s = \frac{1}{16} \sum_{a,b \in \{-3,-1,1,3\}} (a^2 + b^2) = \frac{8}{16} \sum_{a \in \{-3,-1,1,3\}} a^2 = 10.$$

3.4 Optimal Symbol Detection

In this section, we consider the problem of designing a symbol detector that minimizes the probability of error. This is known as an **optimal detection** problem and has an elegant solution that is related to the classical problem of **hypothesis testing** in statistics.

3.4.1 Hypothesis Testing

Sir Ronald Fisher, one of the founders of statistical decision theory, was at a tea party when Ms. Bristol mentioned that she preferred tea poured into milk over milk poured into tea. Fisher commented that surely she could not tell the difference, but his colleague William Roach suggested that they design an experiment. At that point, they prepared eight cups of tea: four milk-into-tea and four tea-into-milk. The cups were presented in a random order and she correctly identified enough (all eight cups by some accounts) to prove her point.

The mathematics behind this type of hypothesis test is based on defining a **null hypothesis**, which states that the difference in preparation has no effect on the outcome. In this example, the null hypothesis is $H_0 =$"the order of pouring the tea and milk does not affect Ms. Bristol's answer". Notice that there are $\binom{8}{4} = 70$ ways that Ms. Bristol can divide the cups of tea into two categories, but that only one identifies all of them correctly. Since the cups are presented in a random order, we can compute

$$\Pr(\text{she identifies all cups correctly}|H_0) = \frac{1}{70}.$$

This number is small enough so that it is reasonable to conclude that H_0 is false and the order does influence Ms. Bristol's decisions.

There are more subtle issues, however, than the case where Ms. Bristol succeeds due to luck. What is more problematic is listing all other hypotheses that can lead to the same observation. For example, variations in the temperature or composition of the tea could help Ms. Bristol guess correctly. Fisher, in his essay on this experiment, argues that proper use of randomization can eliminate the effect of these variations. On the other hand, Ms. Bristol may pass by cheating but still be unable to distinguish between the two drinks. If all possible hypotheses are not explicitly considered, then one can come to an incorrect conclusion. For this reason, it commonly understood that a scientific test of an hypothesis can only *disprove* that hypothesis.

3.4.2 Multiple Hypothesis Testing

Hypothesis testing in communication theory often has the luxury that one of the hypotheses must be true. This leads to the more well-defined problem of **multiple hypothesis testing**. Instead of testing a single hypothesis to see if it is false, one can compare multiple hypothesis to see which is most supported by the observation.

Let $H_0, H_1, \ldots, H_{m-1}$ be m different hypotheses that affect a random observation Y. The probability of a hypothesis before the observation, $\Pr(H_i)$, is called the **a priori probability**. For each hypothesis, the connection with Y is defined by the **observation probability** $\Pr(Y = y \,|\, H_i)$.

The goal is to choose a decision function $D(y)$ which, for any observation, minimizes the decision error probability. Of course, this is equivalent to maximizing the probability that the decision is correct. Notice that, if $Y = y$, then the probability that hypothesis H_i is correct is given by its **a posteriori probability** $\Pr(H_i \,|\, Y = y)$. Therefore, one finds that the optimal choice is the **maximum a posteriori probability** (MAP) decision rule

$$D(y) = \operatorname*{arg\,max}_{i \in \{0,\ldots,m-1\}} \Pr(H_i \,|\, Y = y).$$

In practice, these probabilities can be computed with Bayes' rule using only the a priori probabilities and observation probabilities. This gives

$$\Pr\left(H_i | Y = y\right) = \frac{\Pr(H_i) \Pr(Y = y | H_i)}{\sum_{j=0}^{m-1} \Pr(H_j) \Pr(Y = y | H_j)}.$$

Since the denominator of this expression is the same for all i, the MAP rule can be simplified to

$$D(y) = \operatorname*{arg\,max}_{i \in \{0,\ldots,m-1\}} \Pr(H_i) \Pr(Y = y | H_i).$$

Example 3.4.1. *Consider a system which transmits BPSK over an AWGN channel. Let H_0 be the hypothesis that a zero (i.e., $+1$) was sent and H_1 be the hypothesis that a one (i.e., -1) was sent. For binary hypothesis problems, the MAP decision rule can be written as*

$$\Pr(H_0) \Pr(Y = y | H_0) \underset{H_1}{\overset{H_0}{\gtrless}} \Pr(H_0) \Pr(Y = y | H_0,$$

where this notation implies that one should pick H_0 if the LHS is greater than the RHS and H_1 otherwise. If $\Pr(H_0) = 1-p$ and $\Pr(H_1) = p$, then one can substitute formulae to rewrite this as

$$(1-p)\frac{1}{\sqrt{2\pi\sigma^2}}e^{-(y-1)^2/(2\sigma^2)} \overset{H_0}{\underset{H_1}{\gtrless}} p\frac{1}{\sqrt{2\pi\sigma^2}}e^{-(y+1)^2/(2\sigma^2)}.$$

After a little algebra, taking the logarithm of both sides simplifies this to

$$y \overset{H_0}{\underset{H_1}{\gtrless}} \frac{\sigma^2}{2}\ln\frac{p}{1-p}.$$

Another popular rule is the **maximum likelihood** (ML) decision rule

$$D(y) = \underset{i\in\{0,\ldots,m-1\}}{\arg\max}\ \Pr(Y=y|H_i),$$

which ignores the a priori probability. When all the hypotheses have the same a priori probability, these two rules are identical. In communication systems, this is often the case.

Example 3.4.2. *Consider a system which transmits 4-PAM (i.e, $\mathcal{X} = \{-3,-1,1,3\}$) over an AWGN channel. If all channel inputs are equiprobable, then the optimum detector is*

$$\begin{aligned} D(y) &= \underset{x\in\{-3,-1,1,3\}}{\arg\max}\left(\frac{1}{\sqrt{2\pi\sigma^2}}e^{-(y-x)^2/(2\sigma^2)}\right) \\ &= \underset{x\in\{-3,-1,1,3\}}{\arg\max}\left(-\frac{1}{2}\ln(2\pi\sigma^2) - \frac{1}{2\sigma^2}(y-x)^2\right) \\ &= \underset{x\in\{-3,-1,1,3\}}{\arg\min}\ (y-x)^2. \end{aligned}$$

Therefore, the optimum detector chooses the constellation point closest to the channel observation. Moreover, this statement remains true for any signal constellation with equiprobable signalling and AWGN.

Chapter 4

Fourier Analysis and Sampling

Fourier analysis refers to a collection of tools that can be applied to express a function in terms of complex sinusoids, called basis elements, of different frequencies. The result of the decomposition is the amplitude and the phase to be imparted to each basis element in the reconstruction. This decomposition is termed the **frequency domain** representation of the original signal.

Fourier analysis is extremely useful in engineering, with a myriad of applications. Part of its appeal lies in the fact that basis elements are characteristic functions of linear time-invariant systems. This property, which may seem nebulous at this point, is instrumental in solving many challenging problems, and makes Fourier analysis a powerful methodology for the design of communication systems. We assume that the reader is familiar with basic Fourier analysis, and only review details that are pertinent to our treatment of communication systems. This is not intended to be a comprehensive treatment of the subject.

4.1 Fourier Series

Fourier series can be employed to express, as weighted sums of sinusoidal components, either periodic functions or functions that are time-limited. Suppose that the signal $s(t)$ is zero for all $|t| \geq \frac{T}{2}$, is integrable and satisfies

$$\int_{\mathbb{R}} |s(t)|^2 dt = \int_{-\frac{T}{2}}^{\frac{T}{2}} |s(t)|^2 dt < \infty.$$

Then, $s(t)$ possesses a **Fourier series** representation, which is defined by

$$s(t) = \begin{cases} \sum_{k=-\infty}^{\infty} \hat{s}_k e^{2\pi i \frac{k}{T} t}, & \text{if } |t| \leq \frac{T}{2} \\ 0, & \text{otherwise} \end{cases} \tag{4.1}$$

where the Fourier series coefficients $\{\hat{s}_k : k \in \mathbb{Z}\}$ are given by

$$\hat{s}_k = \frac{1}{T} \int_{-\frac{T}{2}}^{\frac{T}{2}} s(t) e^{-2\pi i \frac{k}{T} t} dt.$$

We can use the standard rectangular function $\mathrm{rect}(\cdot)$, defined by

$$\mathrm{rect}(t) = \begin{cases} 1, & \text{if } |t| < 0.5 \\ 0, & \text{otherwise} \end{cases} \tag{4.2}$$

to simplify (4.1), and rewrite the Fourier representation of $s(t)$ as

$$s(t) = \sum_{k=-\infty}^{\infty} \hat{s}_k e^{2\pi i \frac{k}{T} t} \mathrm{rect}\left(\frac{t}{T}\right). \tag{4.3}$$

If $s(t)$ is periodic with $s(t+T) = s(t)$, instead of being zero for $|t| > \frac{T}{2}$, then the same result holds without the rectangular window function.

From a vector space perspective, (4.3) asserts that $s(t)$ can be expressed as a linear combination of basis elements $\{\theta_k(t) : k \in \mathbb{Z}\}$, where

$$\theta_k(t) = e^{2\pi i \frac{k}{T} t} \mathrm{rect}\left(\frac{t}{T}\right).$$

Furthermore, note that the collection of functions $\{\theta_k(t) : k \in \mathbb{Z}\}$ forms an orthogonal set under the standard inner product; that is,

$$\begin{aligned} \langle \theta_k(t), \theta_n(t) \rangle = \int_{-\infty}^{\infty} \theta_k(t) \theta_n^*(t) dt &= \int_{-\frac{T}{2}}^{\frac{T}{2}} e^{2\pi i \frac{k}{T} t} e^{-2\pi i \frac{n}{T} t} dt \\ &= \int_{-\frac{T}{2}}^{\frac{T}{2}} e^{2\pi i \frac{(k-n)}{T} t} dt = 0 \end{aligned}$$

for all $k \neq n$. An interesting and important aspect of Fourier series is that time-limited functions can be characterized using a discrete set of coefficients. This fact provides insight into the sampling theorem, which we will review shortly.

4.2 Fourier Transforms

The **Fourier transform** applies to functions that are not necessarily time-limited. A signal $x(t)$ is **square integrable** (or an **energy-type signal**) if

$$\|x(t)\|^2 \triangleq \int_{\mathbb{R}} |x(t)|^2 dt < \infty. \tag{4.4}$$

Then, we can express $x(t)$ using its frequency domain representation. The Fourier transform of $x(t)$, which we denote by $\hat{x}(f)$ or $\mathcal{F}[x(t)]$, is defined by

$$\hat{x}(f) = \mathcal{F}[x(t)] \triangleq \int_{\mathbb{R}} x(t) e^{-2\pi i f t} dt. \tag{4.5}$$

Using the inverse Fourier transform, the original function can also be expressed in terms of its decomposition with

$$x(t) = \mathcal{F}^{-1}[x(t)] \triangleq \int_{\mathbb{R}} \hat{x}(f) e^{2\pi i f t} df. \tag{4.6}$$

It is interesting to point out the duality between the Fourier transform and its inverse, $\mathcal{F}\left[\hat{x}(t)\right] = x(-f)$. This relation is rooted in the striking similarity between (4.5) and (4.6).

Definition 4.2.1. *The* ***sinc function*** *is defined by*

$$\operatorname{sinc}(t) \triangleq \frac{\sin(\pi t)}{\pi t}.$$

Example 4.2.2 (Rectangular Pulse)**.** *The rectangular pulse* $\operatorname{rect}(\cdot)$*, defined in* (4.2)*, can be used to constrain various signals in time or frequency. For* $\alpha > 0$*, one has* $\|\operatorname{rect}(\alpha t)\|^2 = 1/\alpha < \infty$*, which guarantees that Fourier analysis can be applied to this function. The Fourier transform of* $\operatorname{rect}(\alpha t)$ *can be computed as follows,*

$$\begin{aligned}
\mathcal{F}\left[\operatorname{rect}(\alpha t)\right] &= \int_{\mathbb{R}} \operatorname{rect}(\alpha t) e^{-2\pi i f t} dt = \int_{-\frac{1}{2\alpha}}^{\frac{1}{2\alpha}} e^{-2\pi i f t} dt \\
&= \frac{1}{\pi f} \left(\frac{e^{\pi i f/\alpha} - e^{-\pi i f/\alpha}}{2i} \right) = \frac{\sin(\pi f/\alpha)}{\pi f} \\
&= \frac{1}{\alpha} \operatorname{sinc}\left(\frac{f}{\alpha} \right).
\end{aligned}$$

Thus, the Fourier transform of $\operatorname{rect}(t)$ *is the aforementioned* $\operatorname{sinc}(f)$ *function, which plays a central role in the sampling and reconstruction of information signals.*

The Fourier transform $\hat{x}(f)$ of a square-integrable signal $x(t)$ also allows one to pose the question: how much signal energy is contained in the spectral band between frequencies f_0 and f_1? The answer, for $f_0 \leq f_1$, is given by the integral

$$\int_{f_0}^{f_1} |\hat{x}(f)|^2 \, df.$$

This integral allows us to interpret the quantity $|\hat{x}(f)|^2$ as the **energy spectral density** of $x(t)$. For technical reasons, we actually define the energy spectral density later in more detail. In practice, the answer to the above question also depends on whether the signal is real or complex. For real signals, the integral is typically computed over the range $f_0 < |f| < f_1$.

When condition (4.4) is not satisfied, it may be hazardous to use Fourier analysis and frequency domain representations. Strictly speaking, the Fourier transform of a function may not exist if the function behaves wildly. Casually taking the Fourier transforms of arbitrary signals should be avoided. Having said that, there will be instances where we discuss the Fourier transforms of functions that do not fulfill (4.4). In such circumstances, the argument to the Fourier transform is carefully selected to remain meaningful from an engineering viewpoint; one such example appears below.

4.2.1 The Dirac Delta Function

The **Dirac delta function** $\delta(t)$ can be defined in a naive fashion with the operational rule

$$x(t) = \int_{\mathbb{R}} \delta(t - \tau) x(\tau) d\tau. \tag{4.7}$$

A more rigorous approach, based on generalized functions, is out of the scope of this class. So, these notes adopt a somewhat cavalier attitude towards the Fourier transform of $\delta(t)$ and rely on experience to avoid pitfalls. The benefit of this approach to Fourier analysis is that it rapidly leads to valuable engineering insight. On the downside, the reader is left with the burden of deciding whether a signal has a proper spectral representation, or if the definition of the Fourier transform is being applied loosely.

Starting with signal $x(t)$, we can write

$$\begin{aligned} x(t) &= \int_{\mathbb{R}} \hat{x}(f) e^{2\pi i f t} df = \int_{\mathbb{R}} \left[\int_{\mathbb{R}} x(\tau) e^{-2\pi i f \tau} d\tau \right] e^{2\pi i f t} df \\ &= \int_{\mathbb{R}} \left[\int_{\mathbb{R}} e^{2\pi i f (t-\tau)} df \right] x(\tau) d\tau, \end{aligned} \tag{4.8}$$

where the second equality follows from (4.5) and the third equality is obtained by changing the order of integration. Since (4.8) holds for any time t, it follows from (4.7) that

$$\delta(t) = \int_{\mathbb{R}} e^{2\pi i f t} df$$

is one representation of $\delta(t)$ and hence the (cavalier) Fourier transform of the δ-function is $\mathcal{F}[\delta(t)] = 1$.

4.2.2 Periodic Signals

We can develop (cavalier) Fourier transform representations for periodic signals as well, thereby providing a unified treatment of periodic and aperiodic functions. Indeed, we can construct the Fourier transform of a periodic signal directly from its Fourier series representation. Let $x(t)$ be a signal with Fourier transform $\hat{x}(f) = \delta(f - f_0)$. To recover the signal $x(t)$, we can apply the inverse Fourier transform

$$x(t) = \mathcal{F}^{-1}[\delta(f - f_0)] = \int_{\mathbb{R}} \delta(f - f_0) e^{2\pi i f t} df = e^{2\pi i f_0 t}.$$

More generally, if $\hat{x}(f)$ is a linear combination of impulses equally spaced in frequency

$$\hat{x}(f) = \sum_{k=-\infty}^{\infty} \hat{s}_k \delta(f - k f_0), \tag{4.9}$$

then its inverse Fourier transform becomes

$$x(t) = \sum_{k=-\infty}^{\infty} \hat{s}_k e^{2\pi i k f_0 t}. \tag{4.10}$$

Note that (4.10) corresponds to the Fourier series representation of a periodic signal. Thus, the Fourier transform of a periodic signal with Fourier series coefficients $\{\hat{s}_k : k \in \mathbb{Z}\}$ can be interpreted as a train of impulses in the frequency domain.

A signal that will be useful in our analysis of sampling is the impulse train (or Dirac comb)

$$x(t) = \sum_{k=-\infty}^{\infty} \delta(t - kT).$$

This is a special case of a periodic function, with period T. We can therefore apply a methodology similar to the one derived above to compute its Fourier transform. The Fourier series coefficients for the impulse train are obtained as

$$\hat{s}_k = \frac{1}{T} \int_{-\frac{T}{2}}^{\frac{T}{2}} x(t) e^{-2\pi i \frac{k}{T} t} dt = \frac{1}{T}.$$

Using (4.9), we get

$$\hat{x}(f) = \frac{1}{T} \sum_{k=-\infty}^{\infty} \delta\left(f - \frac{k}{T}\right). \tag{4.11}$$

Surprisingly, an impulse train in the time domain can be regarded as an impulse train in the frequency domain. A second representation for $x(t)$ is given by (4.10),

$$x(t) = \sum_{k=-\infty}^{\infty} \delta(t - kT) = \frac{1}{T} \sum_{k=-\infty}^{\infty} e^{2\pi i \frac{k}{T} t}. \tag{4.12}$$

Which representation to use depends on the problem at hand.

4.2.3 Spectral Density

The energy of a deterministic signal $x(t)$ is given by (4.4). If the energy of $x(t)$ is finite, i.e. $\|x(t)\|^2 < \infty$, then we define its **autocorrelation function** by

$$R_x(\tau) \triangleq \int_{\mathbb{R}} x(t) x^*(t - \tau) dt.$$

Using this notation, we see that the energy of $x(t)$ is also given by $R_x(0)$.

Definition 4.2.3. *The* ***energy spectral density*** *of an energy-type signal $x(t)$, denoted by $\mathcal{G}_x(f)$, is defined to be the Fourier transform of its autocorrelation function,*

$$\mathcal{G}_x(f) = \mathcal{F}[R_x(\tau)] = |\hat{x}(f)|^2.$$

Intuitively, the energy spectral density captures the frequency content of a signal and helps identify how its energy is distributed across frequencies.

A signal $x(t)$ is a **power-type signal** if the limit

$$P_x = \lim_{T \to \infty} \frac{1}{T} \int_{-\frac{T}{2}}^{\frac{T}{2}} |x(t)|^2 dt$$

exists and $0 < P_x < \infty$. The autocorrelation function of a power-type signal is defined accordingly as

$$R_x(\tau) \triangleq \lim_{T \to \infty} \frac{1}{T} \int_{-\frac{T}{2}}^{\frac{T}{2}} x(t) x^*(t - \tau) dt.$$

We note that the Fourier transform of a power-type signal $x(t)$ need not exist because it may fail to satisfy condition (4.4). Nevertheless, if truncated versions of $x(t)$, defined by

$$x_T(t) \triangleq x(t) \mathrm{rect}\left(\frac{t}{T}\right),$$

are energy-type signals, then we can define the Fourier transforms

$$\hat{x}_T(f) \triangleq \mathcal{F}\left[x(t) \mathrm{rect}\left(\frac{t}{T}\right)\right].$$

From this, the **power spectral density**, which represents the power (per unit of bandwidth) present at each frequency of the signal, can be defined by

$$\mathcal{S}_x(f) \triangleq \lim_{T \to \infty} \frac{1}{T} |\hat{x}_T(f)|^2.$$

Intuitively, the power spectral density captures the frequency content of a signal and helps identify how its power is distributed across frequencies.

Notice how the truncated signal is used to overcome the difficulty of dealing with infinite-energy signals. This is a common and valuable trick.

Definition 4.2.4. *The power spectral density of a power-type signal $x(t)$ is also given by the Fourier transform of its autocorrelation function,*

$$\mathcal{S}_x(f) = \mathcal{F}[R_x(\tau)].$$

These two definitions of power spectral density are equivalent under mild conditions on $x(t)$.

The **spectral bandwidth** of a signal $x(t)$ is the smallest value of W such that its spectral density is zero for all $|f| > W$. An energy-type signal $x(t)$

is **bandwidth-limited** to W if it can be obtained as the inverse Fourier transform of a function $\hat{x}(f)$, where $\hat{x}(f)$ is identically zero for all $|f| > W$. Likewise, a power-type signal is bandwidth-limited to W if its power-spectral density $S_x(f)$ is identically zero for all $|f| > W$.

4.2.4 Linear Time-Invariant Filters

The importance of the Fourier transform comes, partly, from its ability to capture the effects of linear time-invariant filters on deterministic signals. Suppose that the input to a linear time-invariant filter is $x(t)$, then its output is given by

$$y(t) = x(t) * h(t),$$

where $h(t)$ is the impulse response of the linear filter and $*$ denotes the convolution operator. If we use $\hat{h}(f)$ to represent the Fourier transform of impulse response $h(t)$, then the output signal in the frequency domain becomes

$$\hat{y}(f) = \hat{x}(f)\hat{h}(f).$$

That is, convolution in the time domain becomes multiplication in the frequency domain, a much simpler operation. The output signal can then be recovered by taking the inverse Fourier transform of $\hat{y}(f)$,

$$y(t) = \mathcal{F}^{-1}[\hat{y}(f)] = \mathcal{F}^{-1}[\hat{x}(f)\hat{h}(f)].$$

This also implies that the spectral density, $\mathcal{G}_y(f)$, of $y(t)$ satisfies

$$\mathcal{G}_y(f) = \mathcal{G}_x(f)\mathcal{G}_h(f).$$

4.3 Sampling Deterministic Signals

A great deal of inuition about sampling can be gained using the Fourier transform representation for periodic signals developed in Section 4.2.2. Let $x_{\mathrm{s}}(t)$ denote the result of sampling $x(t)$ by impulses at times $\{nT : n \in \mathbb{Z}\}$,

$$x_{\mathrm{s}}(t) = x(t) \sum_{n=-\infty}^{\infty} \delta(t - nT) = \sum_{n=-\infty}^{\infty} x(nT)\delta(t - nT).$$

Looking at the sampled signal in the frequency domain, we get

$$\begin{aligned}
\hat{x}_{\mathrm{s}}(f) &= \hat{x}(f) * \mathcal{F}\left[\sum_{n=-\infty}^{\infty} \delta(t-nT)\right] = \hat{x}(f) * \frac{1}{T} \sum_{n=-\infty}^{\infty} \delta\left(f - \frac{n}{T}\right) \\
&= \frac{1}{T} \sum_{n=-\infty}^{\infty} \hat{x}(f) * \delta\left(f - \frac{n}{T}\right) = \frac{1}{T} \sum_{n=-\infty}^{\infty} \int_{\mathbb{R}} \hat{x}(\xi) * \delta\left(f - \xi - \frac{n}{T}\right) d\xi \\
&= \frac{1}{T} \sum_{n=-\infty}^{\infty} \hat{x}\left(f - \frac{n}{T}\right),
\end{aligned}$$

where we have used (4.11) to express the Fourier transform of an impulse train. When the sampling rate is fast enough, the translated copies of $\hat{x}(f)$ contained in the transform $\hat{x}_{\mathrm{s}}(f)$ do not overlap, and the original signal can be recovered using an ideal lowpass filter. However, when the sampling period T is too small, the various copies of $\hat{x}(f)$ overlap and the content of the original is partially destroyed. This is know as **aliasing**.

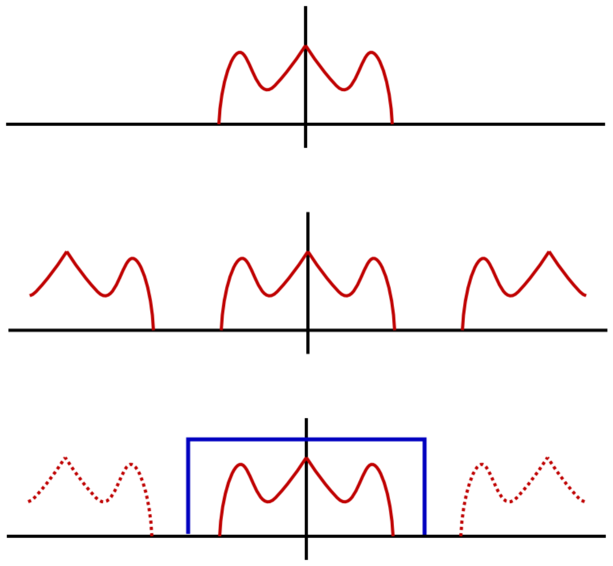

Figure 4.1: The sampling and reconstruction of a bandwidth-limited signal. When the sampling rate exceeds twice the bandwidth of the original signal, this signal can be reconstituted from its sampled values.

A succession of power spectral densities can be found in Figure 4.1. The top component shows the power spectral density of the original signal. The density of the sampled signal appears below. Finally, the reconstruction operation where a lowpass filter is employed to recovered the original function is illustrated at the bottom of the figure. In contrast, Figure 4.2 exhibits a case where the sampling frequency is too low. Aliasing in the frequency domain

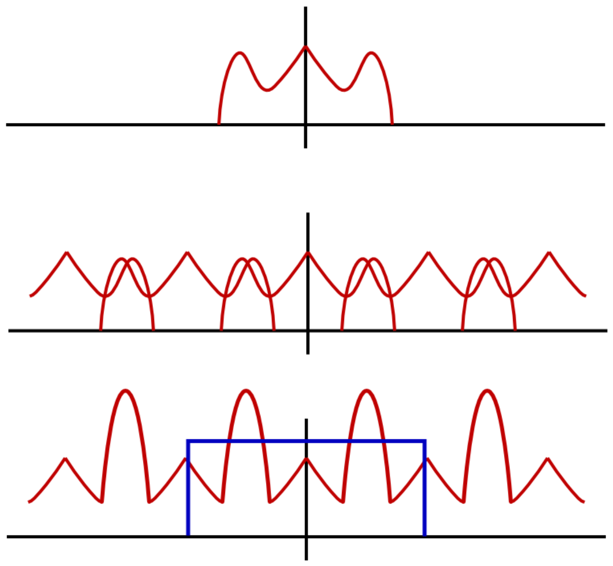

Figure 4.2: A low sampling frequency leads to aliasing, thereby preventing reconstruction of the original signal.

prevents the original signal from being retrieved.

Sampling and aliasing are also important in film and video. The illusion of a moving image in video is achieved by displaying a rapid succession of still pictures over time. Films are typically shot at a rate of twenty-four frames per second, whereas the minimum frame rate required to create the appearance of a moving image is about fifteen frames per second. The human eye acts as a lowpass filter and transforms the succession of images into a live video. High-speed cameras are used to record slow-motion playback movies. As a consequence, they must run at much higher frame-rates than normal cameras.

4.3.1 The Sampling Theorem

The sampling theorem is one of the most significant results in digital communication and signal processing. Many digital communication systems rely on the validity of this theorem and on the design insights it provides for proper operation.

The basic idea behind the sampling theorem can be summarized in a few words. If a signal $x(t)$ is bandwidth-limited to W, then this signal can be reconstructed from a collection of samples so long as the samples are taken at periodic intervals of $T \leq \frac{1}{2W}$. A formal version of the sampling theorem appears below.

Theorem 4.3.1 (Sampling Theorem)**.** *Let signal $x(t)$ be a bandwidth-limited function with bandwidth W. If $x(t)$ is sampled at times $\{nT : n \in \mathbb{Z}\}$ where $T \leq \frac{1}{2W}$, then it is possible to reconstruct the original signal $x(t)$ from its sampled points $\{x(nT) : n \in \mathbb{Z}\}$. Specifically, if $T \leq \frac{1}{2W}$ then*

$$x(t) = \sum_{n=-\infty}^{\infty} x(nT) \operatorname{sinc}\left(\frac{t}{T} - n\right). \tag{4.13}$$

Proof. The signal $x(t)$ is bandwidth-limited with bandwidth W. It follows that $x(t)$ is the inverse Fourier transform of a function $\hat{x}(f)$, where $\hat{x}(f) = 0$ for all frequencies such that $|f| > W$. For convenience, we define $F = \frac{1}{T}$ and we stress that $W \leq \frac{1}{2T} = \frac{F}{2}$. Thus, $\hat{x}(f) = 0$ whenever $|f| > \frac{F}{2}$. We can apply the theory of Fourier series introduced in Section 4.1 to express $\hat{x}(f)$ as

$$\hat{x}(f) = \sum_{k=-\infty}^{\infty} s_k e^{2\pi i \frac{k}{F} f} \operatorname{rect}\left(\frac{f}{F}\right)$$

where the coefficients $\{s_k : k \in \mathbb{Z}\}$ are equal to

$$s_k = \frac{1}{F} \int_{-\frac{F}{2}}^{\frac{F}{2}} \hat{x}(f) e^{-2\pi i \frac{k}{F} f} df.$$

Special care should be taken when reading these equations because we are applying Fourier series analysis to a function in the frequency domain. This can get confusing.

We can then write $x(t)$ in terms of basis elements,

$$\begin{aligned}
x(t) = \mathcal{F}^{-1}\left[\hat{x}(f)\right] &= \mathcal{F}^{-1}\left[\sum_{k=-\infty}^{\infty} s_k e^{2\pi i \frac{k}{F} f} \operatorname{rect}\left(\frac{f}{F}\right)\right] \\
&= \sum_{k=-\infty}^{\infty} s_k \mathcal{F}^{-1}\left[e^{2\pi i \frac{k}{F} f} \operatorname{rect}\left(\frac{f}{F}\right)\right] \\
&= \sum_{k=-\infty}^{\infty} \frac{s_k}{T} \operatorname{sinc}\left(\frac{t}{T} + k\right).
\end{aligned}$$

Above, we have successively used the scaling and time-shift properties of the Fourier transform. We can obtain the values of $\{s_k : k \in \mathbb{Z}\}$ explicitly by exploiting the characteristics of the sinc$(\cdot)$ function,

$$x(nT) = \sum_{k=-\infty}^{\infty} \frac{s_k}{T} \operatorname{sinc}\left(\frac{nT}{T} + k\right) = \sum_{k=-\infty}^{\infty} \frac{s_k}{T} \operatorname{sinc}(n+k) = \frac{s_{-n}}{T}.$$

Thus, we have $s_n = Tx(-nT)$ and formula (4.13) follows. The sampling rate $F = 2W$ associated with the sampling period $T = \frac{1}{2W}$ is the minimum rate at which perfect reconstruction is possible. It is called the **Nyquist rate** in honor of Swedish-American engineer Harry Nyquist. □

4.3.2 Imperfect Sampling and Reconstruction

In practice, it is impossible to measure (i.e., sample) a signal $x(t)$ instantaneously. A more realistic model is to assume that the sample value is obtained through the integral

$$u(t) = \int_{\mathbb{R}} x(\tau) p(\tau - t) d\tau,$$

for some sampling waveform $p(t)$. In this case, the resulting samples are identical to the perfect sampling of the filtered waveform $u(t) = x(t) * p(-t)$. That is, the sample values are given by

$$u(nT) = \int_{\mathbb{R}} x(\tau) p(\tau - nT) d\tau.$$

It also follows that, if no aliasing occurs, this degradation can be eliminated completely using a discrete-time filter to equalize the resulting samples.

A similar imperfection occurs during reconstruction. In practice, it is not possible to weight each sample by the exact sinc interpolation waveform. Instead, the samples $x(nT)$ are weighted by a pulse shape $q(t)$. In this case, the reconstruction output is given by

$$y(t) = \sum_{n=-\infty}^{\infty} x(nT) q(t - nT).$$

Since $q(t-nT) = \delta(t-nT) * q(t)$, we can see this instead as perfect reconstruction followed by filtering and write

$$y(t) = \left(\sum_{n=-\infty}^{\infty} x(nT) \delta(t - nT) \right) * q(t) = x(t) * q(t).$$

Again, it follows that this imperfection can be eliminated completely by post-filtering. In practice, the main advantage of this observation is that one can jointly optimize $p(t)$, $q(t)$ and the post-filter to provide good performance while using inexpensive components.

4.4 Sampling Bandlimited Processes*

We know from Theorem 4.3.1 that a bandwidth-limited signal can be perfectly reconstructed from its samples provided that the sampling rate exceeds twice the bandwidth of the original signal. At this point, one may wonder whether it is possible to extend the sampling theorem to bandwidth-limited stochastic processes. This question is answered in the affirmative below.

Theorem 4.4.1. *Suppose that $X(t)$ is a wide-sense stationary bandwidth-limited process with bandwidth W and power spectral density $\mathcal{S}_X(f)$. Let $\tilde{X}(t)$ be an approximation for $X(t)$ built from the sampled values $\{X(nT) : n \in \mathbb{Z}\}$,*

$$\tilde{X}(t) = \sum_{n=-\infty}^{\infty} X(nT) \mathrm{sinc}(2W(t - nT)),$$

where $T = \frac{1}{2W}$ denotes the sampling interval. Then the mean-squared error between the original random process and the reconstructed version vanishes,

$$\left\| X(t) - \tilde{X}(t) \right\|^2 = \mathrm{E}\left[\left| X(t) - \sum_{n=-\infty}^{\infty} X(nT) \mathrm{sinc}(2W(t - nT)) \right|^2 \right] = 0. \tag{4.14}$$

The expectation in (4.14) *is over all possible realizations of* $X(t)$.

Proof. To establish this result, we expand the mean-squared error of (4.14),

$$\left\|X(t) - \tilde{X}(t)\right\|^2 = \mathrm{E}\left[\left|X(t) - \sum_{n=-\infty}^{\infty} X(nT)\mathrm{sinc}(2W(t-nT))\right|^2\right]$$

$$= R_X(0) - \sum_{n=-\infty}^{\infty} [R_X(t-nT) + R_X(t-nT)]\mathrm{sinc}(2W(t-nT))$$

$$+ \sum_{n=-\infty}^{\infty}\sum_{m=-\infty}^{\infty} R_X((m-n)T)\mathrm{sinc}(2W(t-mT))\mathrm{sinc}(2W(t-nT)).$$

The double summation above can be rewritten as

$$\sum_{n=-\infty}^{\infty}\sum_{k=-\infty}^{\infty} R_X(kT)\mathrm{sinc}(2W(t-kT-nT))\mathrm{sinc}(2W(t-nT))$$

$$\sum_{n=-\infty}^{\infty}\left(\sum_{k=-\infty}^{\infty} R_X(kT)\mathrm{sinc}(2W(t-kT-nT))\right)\mathrm{sinc}(2W(t-nT))$$

$$= \sum_{n=-\infty}^{\infty} R_X(t-nT)\mathrm{sinc}(2W(t-nT)),$$

where the last equality follows from the sampling theorem for deterministic signals (Theorem 4.3.1). Putting these results together, we get

$$\left\|X(t) - \tilde{X}(t)\right\|^2 = R_X(0) - \sum_{n=-\infty}^{\infty} R_X^*(t-nT)\mathrm{sinc}(2W(t-nT)).$$

Applying Theorem 4.3.1 one more time and noticing that $R_X(0) = R_X^*(0)$, we obtain $\|X(t) - \tilde{X}(t)\|^2 = 0$, as desired. □

Theorem 4.4.1 is important because it confirms that the design insights gained from analyzing deterministic signals hold for random signals as well.

4.5 Bandpass Signals and Processes*

One possible application of sampling is to take a continuous-time signal and to transform it into a discrete-time signal. For instance, this operation gives the information coming out of a source a format more suitable for digital communications. This prime application of sampling served as the original motivation for our study of the subject. A second possible application of sampling is the processing of received waveforms at the output of communication channels. In digital communications, the data often assumes the form of an analog carrier signal modulated by a digital bit stream. Mathematically, this situation is captured by the equation

$$y(t) = x(t) \cos(2\pi f_{\mathrm{c}} t).$$

The signal $y(t)$ is a special form of a **bandpass signal**. Its Fourier transform $\hat{y}(f)$ is non-zero only for frequencies contained in a small neighborhood of carrier frequency f_{c}. That is, $\hat{y}(f) = 0$ for all frequencies such that $|f - f_{\mathrm{c}}| \geq W$. To apply the sampling tools derived above to the information bearing signal $x(t)$, we need to shift the corresponding spectrum to the origin.

The Fourier transform of $y(t)$ is given by

$$\hat{y}(f) = \frac{1}{2}\hat{x}(f + f_{\mathrm{c}}) + \frac{1}{2}\hat{x}(f - f_{\mathrm{c}}).$$

Our strategy is to first eliminate $\frac{1}{2}\hat{x}(f + f_{\mathrm{c}})$ from $\hat{y}(f)$, and then to scale and shift $\frac{1}{2}\hat{x}(f + f_{\mathrm{c}})$ back to the origin. Define the **step function** by

$$\mathrm{step}(t) = \frac{1}{2} + \frac{1}{2}\mathrm{sign}(t).$$

Taking the (cavalier) Fourier transform of step(t), we get

$$\begin{aligned} \mathcal{F}[\mathrm{step}(t)] &= \mathcal{F}\left[\frac{1}{2} + \frac{1}{2}\mathrm{sign}(t)\right] \\ &= \frac{1}{2}\delta(f) - \frac{1}{2}\int_{-\infty}^{0} e^{-2\pi i f t} dt + \frac{1}{2}\int_{0}^{\infty} e^{-2\pi i f t} dt \\ &= \frac{1}{2}\delta(f) + \frac{1}{2\pi i f}. \end{aligned}$$

Using the duality property of the Fourier transform, we get

$$\mathcal{F}^{-1}[\mathrm{step}(f)] = \frac{1}{2}\delta(t) + \frac{i}{2\pi t}.$$

And, by construction, we obtain $\hat{x}(f - f_{\mathrm{c}}) = 2\mathrm{step}(f)\hat{y}(f)$. We can therefore recover the original lowpass signal $x(t)$ using the frequency-shift property of the Fourier transform,

$$x(t) = \left[y(t) * \left(\delta(t) + \frac{i}{\pi t} \right) \right] e^{-2\pi i f_c t} = \left[y(t) + i \left(y(t) * \frac{1}{\pi t} \right) \right] e^{-2\pi i f_c t}.$$

The second component of this signal,

$$y(t) * \frac{1}{\pi t},$$

is called the **Hilbert transform** of $y(t)$. Once $x(t)$ is brought back to baseband, the standard sampling theorem applies and a discrete-time version of the signal can be produced.

4.6 Stochastic Signals

A **random process** (or **stochastic process**) is an extension of the concept of random variable to the situation where the values of a signal are not known beforehand. Mathematically, a stochastic process can be viewed in two different ways. First, the process can be thought of as an instantiation of a random experiment where the outcome is selected from a collection of time functions. Alternatively, a stochastic process can be viewed as a collection of random variables indexed by time. If the index set corresponds to the real numbers, then the process is a **continuous-time random process**. Whereas if the index set is discrete, then it is a **discrete-time random process**. The viewpoint where a stochastic process is regarded as a collection of random variables tends to prevail in the study of digital communications.

Random processes are frequently employed in the design of communication systems. For example, they can be used to model the data originating from a source, channel variations, noise and interference. Their importance will become evident as we progress through these notes. In general, it is difficult to provide a complete mathematical description for a random process. For now, we restrict our attention to **stationary** and **ergodic** random processes.

Definition 4.6.1 (Stationarity). *A random process $X(t)$ is wide-sense stationary (WSS) if its* ***mean***

$$m_X(t) \triangleq \mathrm{E}[X(t)]$$

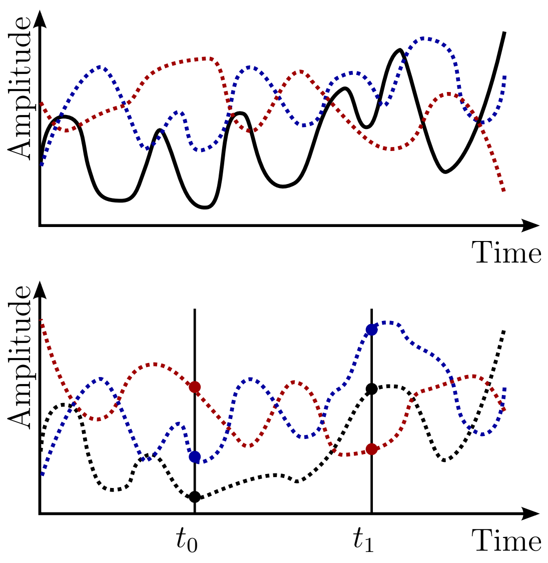

Figure 4.3: Two distinct abstractions of a random process. It can be viewed as the output of an experiment where a function is selected at random. Alternatively, a random process may be taken as a set of random variables indexed by time.

is independent of time, and its ***autocorrelation function****, defined by*

$$R_X(t_1, t_2) \triangleq \mathrm{E}[X(t_1)X^*(t_2)],$$

only depends on the difference between t_1 and t_2. With a slight abuse of notation, we can denote the mean and autocorrelation of a stationary process respectively by m_X and $R_X(\tau)$, where $\tau = t_1 - t_2$.

Definition 4.6.2 (Ergodic)**.** *An* ***ergodic theorem*** *asserts that, under certain conditions, the time average of a random process*

$$\langle g(X(t)) \rangle \triangleq \lim_{T \to \infty} \frac{1}{T} \int_{-\frac{T}{2}}^{\frac{T}{2}} g(X(t)) dt$$

exists and is equal to the ensemble average $\mathrm{E}[g(X(t))]$ for almost all trajectories of the random process. When a stochastic process fulfills these conditions, it is called ***ergodic****.*

One of the important characteristics of an ergodic process is that it suffices to look at one realization of the process to infer many of its statistical

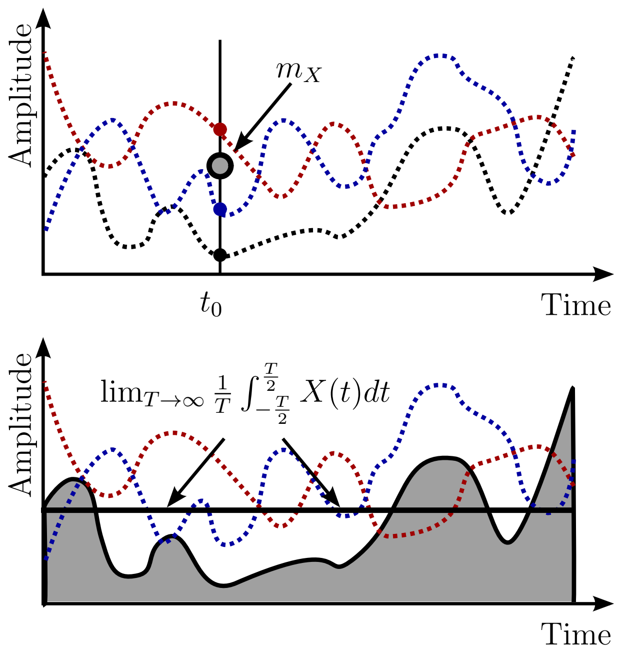
Amplitude
m_X
t_0
Time
Amplitude
$\lim_{T\to\infty} \frac{1}{T} \int_{-\frac{T}{2}}^{\frac{T}{2}} X(t) dt$
Time

Figure 4.4: For ergodic processes, the time average of a function along a trajectory is equal to the ensemble average.

attributes. Ergodicity is a very strong property, and it is hard to test and validate. Rather, it is frequently taken as a premise in the design of communication systems. For instance, most information sources are assumed to be stationary and ergodic. Such a postulate appears reasonable, especially considering the many successful communication systems implemented based on this assumption.

4.6.1 Power Spectral Density

For a stochastic signal, the definition of the **power spectral density** is somewhat more intricate because it must account for uncertainty in the process. Let $X(t)$ be a wide-sense stationary random process with $R_X(0) < \infty$. Then, we can define

$$\hat{X}_T(f) \triangleq \mathcal{F}\left[X(t)\mathrm{rect}\left(\frac{t}{T}\right)\right]$$

and

$$\mathcal{S}_X(f) \triangleq \lim_{T\to\infty} \frac{1}{T}\mathrm{E}\left[|\hat{X}_T(f)|^2\right].$$

As we will soon see, the power spectral density plays an instrumental role in the sampling theorem for random signals. First, we provide a means to compute $\mathcal{S}_X(f)$ from its statistical attributes.

Theorem 4.6.3 (Wiener-Khinchin). *The power spectral density $\mathcal{S}_X(f)$ of a wide-sense stationary random process $X(t)$ is equal to the Fourier transform of its autocorrelation function, $\mathcal{S}_X(f) = \mathcal{F}[R_X(\tau)]$.*

Proof. For a wide-sense stationary process, we have

$$\begin{aligned}
\mathcal{S}_X(f) &= \lim_{T\to\infty} \frac{1}{T}\mathrm{E}\left[|\hat{X}_T(f)|^2\right] = \lim_{T\to\infty} \frac{1}{T}\mathrm{E}\left[\hat{X}_T(f)\hat{X}_T^*(f)\right] \\
&= \lim_{T\to\infty} \frac{1}{T}\mathrm{E}\left[\int_{-\frac{T}{2}}^{\frac{T}{2}} X(t_1)e^{-2\pi i f t_1}dt_1 \int_{-\frac{T}{2}}^{\frac{T}{2}} X^*(t_2)e^{2\pi i f t_2}dt_2\right] \\
&= \lim_{T\to\infty} \frac{1}{T}\int_{-\frac{T}{2}}^{\frac{T}{2}}\int_{-\frac{T}{2}}^{\frac{T}{2}} \mathrm{E}\left[X(t_1)X^*(t_2)\right]e^{-2\pi i f(t_1-t_2)}dt_1dt_2 \\
&= \int_{\mathbb{R}} R_X(\tau)e^{-2\pi i f\tau}d\tau = \mathcal{F}[R_X(\tau)].
\end{aligned}$$

The fourth equality is obtained by interchanging the expectation and the integrals, while the sixth equality follows from a change of variables and the fact that $X(t)$ is wide-sense stationary. To guarantee that the former operation is legitimate, $\tau R_X(\tau)$ must remain finite for all τ. □

In some cases, it may be possible to estimate the power spectral density from a single realization of $X(t)$. For example, the autocorrelation of $X(t)$ is ergodic if it satisfies

$$\langle X(t)X^*(t-\tau)\rangle \triangleq \lim_{T\to\infty}\frac{1}{T}\int_{-\frac{T}{2}}^{\frac{T}{2}} X(t)X^*(t-\tau)dt = \mathrm{E}\left[X(t)X^*(t-\tau)\right].$$

In this case, the expectation in the theorem above can be replaced by a time average and the result can be computed from a single realization.

4.6.2 Filtering Stochastic Processes

We discussed in Section 4.2.4 how the Fourier transform can simplify the analysis of the effects of linear time-invariant filters on deterministic signals. In this section, we consider the operation of such filters in the context of random processes.

Theorem 4.6.4. *If a wide-sense stationary process $X(t)$ with mean m_X and autocorrelation function $R_X(\tau)$ is passed through a linear time-invariant filter with impulse response $h(t)$, then the output process $Y(t)$ has mean*

$$m_Y = m_X \int_{\mathbb{R}} h(t) dt$$

and its autocorrelation is equal to

$$R_Y(\tau) = R_X(\tau) * h(\tau) * h^*(-\tau).$$

Proof. The output process at time t is given by $Y(t) = \int_{\mathbb{R}} X(t-\xi)h(\xi)d\xi$. We can therefore obtain the expectation of $Y(t)$ as follows,

$$\begin{aligned} m_Y(t) &= \mathrm{E}\left[\int_{\mathbb{R}} X(t-\xi)h(\xi)d\xi\right] = \int_{\mathbb{R}} \mathrm{E}\left[X(t-\xi)\right] h(\xi)d\xi \\ &= m_X \int_{\mathbb{R}} h(\xi)d\xi. \end{aligned}$$

We emphasize that m_Y is independent of time.

To derive the autocorrelation function for $Y(t)$, we first compute the cross-correlation between $X(t)$ and $Y(t)$,

$$\begin{aligned} \mathrm{E}[X(t_1)Y^*(t_2)] &= \mathrm{E}\left[X(t_1)\int_{\mathbb{R}} X^*(\xi)h^*(t_2-\xi)d\xi\right] \\ &= \int_{\mathbb{R}} \mathrm{E}\left[X(t_1)X^*(\xi)\right] h^*(t_2-\xi)d\xi \\ &= \int_{\mathbb{R}} R_X(t_1-\xi)h^*(t_2-\xi)d\xi \\ &= R_X(\tau) * h^*(-\tau). \end{aligned} \tag{4.15}$$

This shows that the cross-correlation between $X(t)$ and $Y(t)$ depends only on τ; we can therefore express it as $R_{XY}(\tau)$. We are now ready to compute the autocorrelation function for $Y(t)$.

$$\begin{aligned} \mathrm{E}[Y(t_1)Y^*(t_2)] &= \mathrm{E}\left[\int_{\mathbb{R}} X(\xi)h(t_1-\xi)d\xi Y^*(t_2)\right] \\ &= \int_{\mathbb{R}} \mathrm{E}\left[X(\xi)Y^*(t_2)\right] h(t_1-\xi)d\xi \\ &= \int_{\mathbb{R}} R_{XY}(\xi-t_2)h(t_1-\xi)d\xi \\ &= R_{XY}(\tau) * h(\tau). \end{aligned} \tag{4.16}$$

Substituting $R_{XY}(\tau)$ by the equivalent expression $R_X(\tau) * h^*(-\tau)$ from (4.15), we get the desired result. We observe that the autocorrelation of the process $Y(t)$ only depends on the difference between t_1 and t_2, and hence $Y(t)$ is also wide-sense stationary. □

Obtaining an expression for the autocorrelation function corresponding to the output of a linear time-invariant filter allows us to characterize the power spectral density of the output process. In terms of the frequency representation, we get $m_Y = m_X \hat{h}(0)$ and

$$\begin{aligned} \mathcal{S}_Y(f) &= \mathcal{F}[R_Y(\tau)] \\ &= \mathcal{F}\left[R_X(\tau) * h(\tau) * h^*(-\tau)\right] \\ &= \mathcal{S}_X(f)|\hat{h}(f)|^2. \end{aligned}$$

A linear time-invariant filter can be employed to shape the spectrum of a stochastic process, and to constrain its bandwidth. This is an important result, as linear filters can be used to reduce the bandwidth of a random signal before sampling or to reconstruct a random signal from its samples.

4.6.3 Gaussian Processes

A stochastic process $X(t)$ is a **Gaussian process** if, for any finite set of sampling times $t_1, t_2, \ldots, t_n$, the resulting random variables $X(t_1), X(t_2), \ldots, X(t_n)$ are jointly Gaussian random variables. The distribution of jointly Gaussian random variables is completely determined by their mean vector and correlation matrix. Therefore, a Gaussian process is completely determined by its mean function $m_X(t)$ and its autocorrelation function $R_{XX}(t_1, t_2)$. Of course, if the process is also wide-sense stationary, then this reduces to its mean value m_X and its autocorrelation function $R_X(\tau)$.

More generally, one finds that, for any finite set of energy-type signals $s_1(t), s_2(t), \ldots, s_n(t)$, the random variables

$$Z_i = \int_{-\infty}^{\infty} s(t) X(t) dt$$

are jointly Gaussian. The intuition behind this is that each integral is defined as the limit of a sequence of Riemann sums. Since each Riemann sum is simply the sum of Gaussian random variables, it is also Gaussian. This argument can be extended to show that the mean vector and correlation matrix of the vector $(Z_1, Z_2, \ldots, Z_n)$ is also jointly Gaussian. In fact, combining this with Theorem 4.6.4, implies that the output of linear time-invariant system, whose input is a wide-sense stationary Gaussian process, is also a wide-sense stationary Gaussian process.

The most common random process in signal processing and communications is the **Gaussian white-noise** process $N(t)$. This process is defined to be a zero-mean wide-sense stationary Gaussian process whose power spectral density $S_N(f)$ is constant (e.g., 1). Since this process has infinite power, it is not particularly well-defined mathematically. Still, we can use the cavalier Fourier transform $\mathcal{F}[1] = \delta(t)$ to argue that the autocorrelation function of $N(t)$ should be $R_N(\tau) = \delta(\tau)$. In practice, this is not a problem because $N(t)$ is only used as the input to a linear filter whose output process is guaranteed to have finite power.

Chapter 5

Quantization

As mentioned in the introduction, two operations are necessary to transform an analog waveform into a digital signal. The first action, sampling, consists of converting a continuous-time input into a discrete-time function. The second operation is the process of approximating continuous-space sample values by a discrete set of possible points. This process, termed **quantization**, is also essential to transmit an analog signal over digital media. Quantization invariably induces a loss in signal quality. The **distortion** between the original and quantized functions is usually unwanted, and cannot be reversed. Yet, for a specific application, the level of signal degradation can be controlled.

In this chapter, we focus primarily on the quantization of real numbers. The techniques described here can easily be extended to complex numbers by quantizing the real and imaginary parts separately. In a more abstract sense, the quantization of a complex number is equivalent to vector quantization for a pair of real numbers.

5.1 Scalar Quantizers

Quantizers can generally be designed to be very robust for a large class of signals. In scalar quantization, each source value is processed individually; the input value is mapped to an output taking one of finitely many values. The number of quantization levels is typically chosen to be a power-of-2 because the outputs are usually represented using binary strings. Mathematically, a quantizer is a function taking value in a finite set. The input to the quantizer is a real number and the output belongs to set $\mathcal{Q}$. Then, one can define the quantizer as a function $Q : \mathbb{R} \mapsto \mathcal{Q}$ with output

$$x_{\mathrm{q}} = Q(x).$$

This is perhaps best seen through an example.

Example 5.1.1. *Let $Q : \mathbb{R} \mapsto \mathcal{Q}$ be a quantizer with four possible outputs labeled q_1 through q_4. The output x_{q} of the quantizer belongs to set $\mathcal{Q}$, and it must therefore be equal to one of the four possible points listed above. Figure 5.1 shows the functional representation of a four-level quantization scheme. The*

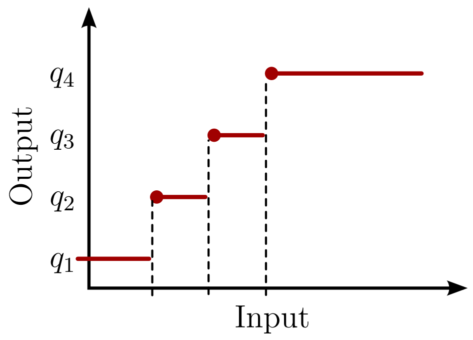

Figure 5.1: This is a functional representation of a quantizer where the input is converted to one of four possible values.

output of the quantizer in this example is determined according to a nearest neighbor rule and this implies that

$$Q(x) = \begin{cases} q_1 & \text{if } x < \frac{q_1+q_2}{2} \\ q_2 & \text{if } \frac{q_1+q_2}{2} \leq x < \frac{q_2+q_3}{2} \\ q_3 & \text{if } \frac{q_2+q_3}{2} \leq x < \frac{q_3+q_4}{2} \\ q_4 & \text{if } x \geq \frac{q_3+q_4}{2} \end{cases}.$$

We note that this quantization function is not one-to-one and, therefore, does not have an inverse.

5.2 Distortion Measures

To compare different quantizers, a performance metric must be established. A common distortion measure with desirable properties is the square of the error

$$d(x, x_{\mathrm{q}}) = |x - Q(x)|^2, \tag{5.1}$$

where $x - x_{\mathrm{q}}$ is called the **quantization error**. The expression in (5.1) measures how close the output x_{q} is to the input for a specific value of x. It is popular in communications and signal processing because it is proportional to the increase in noise power caused by quantization.

When a quantizer is employed to discretize the value of a sampled waveform, it is appropriate to evaluate the performance of the quantizer in terms of the difference between the original function and the reconstructed version rather than the distortion between the sample values themselves. Suppose that $x(t)$ is a bandwidth-limited process with bandwidth W. We know that this process can be accurately recovered from the sampled values $\{x(nT) : n \in \mathbb{Z}\}$ using the formula

$$x(t) = \sum_{n=-\infty}^{\infty} x(nT) \mathrm{sinc}\left(\frac{t}{T} - n\right)$$

where $T \leq \frac{1}{2W}$. Assume that the sampled values are quantized before reconstruction, with

$$x_{\mathrm{q}}(nT) = Q(x(nT)).$$

A legitimate question is, how close is the approximation $x_{\mathrm{q}}(t)$ to the original signal $x(t)$ using the quantized data

$$x_{\mathrm{q}}(t) = \sum_{n=-\infty}^{\infty} x_{\mathrm{q}}(nT) \mathrm{sinc}\left(\frac{t}{T} - n\right),$$

is anywhere close to $x(t)$. A useful performance criterion is the energy in the qunatization error signal,

$$\int_{\mathbb{R}} |x(t) - x_{\mathrm{q}}(t)|^2 \, dt,$$

which quantifies the total distortion between the original and reconstructed signals. Using Parseval's identity, we get

$$\begin{aligned}
\int_{\mathbb{R}} |x(t) - x_{\mathrm{q}}(t)|^2 \, dt &= \int_{\mathbb{R}} |\hat{x}(f) - \hat{x}_{\mathrm{q}}(f)|^2 \, df \\
&= \int_{\mathbb{R}} \left| \sum_{n=-\infty}^{\infty} \frac{(x(nT) - x_{\mathrm{q}}(nT))}{2W} e^{2\pi i \frac{n}{2W} f} \mathrm{rect}\left(\frac{f}{2W} \right) \right|^2 df \\
&= \sum_{n=-\infty}^{\infty} |x(nT) - x_{\mathrm{q}}(nT)|^2 .
\end{aligned}$$

Above, we have used the fact that the basis elements

$$\left\{ e^{2\pi i \frac{n}{2W} f} \mathrm{rect}\left(\frac{f}{2W} \right) : n \in \mathbb{Z} \right\}$$

are orthogonal. In some sense, minimizing the quantization error for individual samples turns out to also minimize the overall squared error between the original and reconstructed waveforms $x(t)$ and $x_{\mathrm{q}}(t)$.

5.2.1 Mean Squared Error

Since the quantizer is designed to operate on an information signal, a more relevant assessment of performance weighs in the accuracy of the quantizer over all possible realizations of the random input, as opposed to a specific realization. An appropriate distortion measure for the quantization of a stochastic signal is provided by the **mean squared error** (MSE),

$$\mathrm{E}[d(X, X_{\mathrm{q}})] = \mathrm{E}\left[(X - Q(X))^2 \right] . \tag{5.2}$$

Note that this performance metric depends on the distribution of the input signal, and hence is tied to a specific application. A quantizer can be said to work well in a particular context. However, describing the performance of a quantization scheme without specifying the distribution of its input signal is meaningless.

Example 5.2.1. *Suppose that the values of a discrete-time information signal are uniformly distributed over* $[0, 16]$. *Furthermore, assume that the quantizer implements a nearest neighbor rule with quantization levels equal to* $q_m = 2m - 1$ *for* $m = 1, 2, \ldots, 8$. *We wish to find the mean squared error of this quantization scheme.*

Being uniformly distributed, the probability density function of the input is

$$f_X(x) = \frac{1}{16}$$

for $x \in [0, 16]$, *and zero otherwise. The decision regions corresponding to the various quantization points are given by* $[0, 2], (2, 4], \ldots, (14, 16]$. *We can therefore compute the mean squared error as follows,*

$$\begin{aligned} \mathrm{E}[d(X, X_{\mathrm{q}})] = \mathrm{E}\left[(X - Q(X))^2\right] &= \int_0^{16} \frac{(x - Q(x))^2}{16} dx \\ &= \sum_{m=1}^{8} \int_{2m-2}^{2m} \frac{(x - (2m-1))^2}{16} dx \\ &= \frac{1}{2} \int_0^2 (x-1)^2 dx = \frac{1}{3}. \end{aligned}$$

That is, the mean squared error associated with this input distribution and quantization scheme is $\mathrm{E}\left[(X - Q(X))^2\right] = \frac{1}{3}$.

5.2.2 Signal to Quantization-Noise Ratio

One of the criticisms about the MSE of (5.2) is that it has a tendency to assign a larger distortion value to signal input with larger second moments. Indeed, an information process likely to feature large amplitudes is bound to yield outputs with a large mean squared error. On the other hand, under this absolute metric, most quantizers may appear to work well for minute signals as their mean squared errors are destined to remain small. A normalized version of this criterion that takes into consideration the power of the original signal is the **signal-to-quantization-noise ratio** (SQNR),

$$\mathrm{SQNR} = \frac{\mathrm{E}\left[X^2\right]}{\mathrm{E}\left[(X - Q(X))^2\right]}. \tag{5.3}$$

Because of the implicit normalization associated with (5.3), this latter measure is more suitable to compare quantizer performance for different input processes. Like many quantities in communications and signal processing, the value is often specified in decibels by applying the function $10 \log_{10}(\cdot)$.

Example 5.2.2. *Assume that the values of a discrete-time information signal are uniformly distributed over* $[0, 2]$. *We denote the input signal in the present* *problem using Y. We wish to obtain the mean squared error associated with the quantizer of Example 5.2.1, applied to the signal at hand. Also, we wish to compare the SQNR of the quantization scheme of Example 5.2.1 with the SQNR of the scenario described in this example.*

To derive the mean squared error, we follow the same steps as before

$$\begin{aligned} \mathrm{E}[d(Y, Y_{\mathrm{q}})] &= \int_0^2 \frac{(y - Q(y))^2}{2} dy \\ &= \frac{1}{2} \int_0^2 (y-1)^2 dy = \frac{1}{3}. \end{aligned}$$

We notice that the MSE is the same as the one derived in Example 5.2.1. Nevertheless, the quantization scheme seems more suited to the signal described in the previous example. The SQNR of the current scheme is given by

$$SQNR = \frac{\mathrm{E}[Y^2]}{\mathrm{E}[d(Y, Y_{\mathrm{q}})]} = 3\mathrm{E}[Y^2] = 4 \approx 6.02\, dB.$$

This can be compared with the SQNR of the problem featured in Example 5.2.1, which is given by

$$SQNR = \frac{\mathrm{E}[X^2]}{\mathrm{E}[d(X, X_{\mathrm{q}})]} = 3\mathrm{E}[X^2] = 256 \approx 24.08\, dB. \tag{5.4}$$

Obviously, the SQNR is much better in the case of Example 5.2.1. Can you think of an eight-level quantization scheme that would perform better for the current problem, perhaps rivaling the SQNR of (5.4)*?*

Both the mean squared error and the signal-to-quantization-noise ratio are valid and meaningful ways to present performance results for quantization schemes. Still, they must be put in proper context, especially when comparing scenarios where the distributions of the input signals differ. For a fixed input process, a quantization scheme that minimizes the MSE will invariably maximize the SQNR.

Example 5.2.3 (Quantization Rule). *Consider a signal whose amplitude is bounded between -1 and 1 and whose amplitude distribution has a uniform distribution over the same range. Suppose that n bits are represent 2^n uniformly spaced quantization levels $q_m = -1 - 2^{-n} + m2^{-n+1}$ for $m = 1, \ldots, 2^n$. The performance as a function of n is typically approximated by the rule that the* ***SQNR increases by 6.02 dB per bit*** *for additional quantization bits.*

We can understand this by analyzing the quantizer function, which is

$$Q(x) = \begin{cases} -1 + 2^{-n} & \text{if } x < -1 \\ \frac{1}{2^{n-1}} \left(\lfloor 2^{n-1} x \rfloor + \frac{1}{2} \right) & \text{if } -1 \le x < 1 \\ 1 - 2^{-n} & \text{if } x \ge 1. \end{cases}$$

By symmetry, the distribution of the quantization error is the same for each quantization cell so it suffices to calcluate

$$\mathrm{E}\left[(X - Q(X))^2 \,\middle|\, Q(X) = q \right] = \int_{q-2^{-n}}^{q+2^{-n}} f_{X|Q(X)=q}(x)(x-q)^2 dx$$

$$= \int_{-2^{-n}}^{2^{-n}} 2^{n-1} x^2 dx = 2^{n-1} \frac{2^{-3n+1}}{3} = \frac{2^{-2n}}{3}.$$

The SQNR in decibels is therefore given by

$$SQNR = 10 \log_{10} \left(\frac{3\, E[X^2]}{2^{-2n}} \right) = P_X + 10 \log_{10} 3 + 20n \log_{10} 2$$

$$\approx P_X + 4.77 + 6.02n \ dB.$$

Since $E[X^2] = 2/3$, we have $P_X \approx -1.76$ dB and the SQNR simplifies to

$$SQNR \approx 3.01 + 6.02n \ dB.$$

What is surprising is that, for large n, this rule holds for any continuous amplitude distribution supported on $[-1, 1]$. This can be understood by repeating the above computation for an arbitrary continuous amplitude distribution $f_X(x)$. As n increases, the width of each quantization cell (i.e., 2^{-n+1}) decreases rapidly and the amplitude distribution becomes essentially constant over each quantization cell. Therefore, the conditional amplitude distribution, given the reconstruction point, satisfies

$$f_{X|Q(X)=q}(x|q) \approx \begin{cases} 2^{-n+1} & \text{if } q - 2^{-n} \le x < q + 2^{-n} \\ 0 & \text{otherwise.} \end{cases}.$$

5.2.3 Dithering

While the SQNR accurately measures the increase in noise power caused by quantization, it does not give any information about the spectral content of the quantization noise. For many applications, it is also important that the quantization noise be white (i.e., uncorrelated in time). To see the problem with standard quantization, consider the periodic signal x_n that satisfies $x_{n+N} = x_n$. In this case, the quantized version $y_n = Q(x_n)$ and the quantization error $e_n = y_n - x_n$ are also periodic. Therefore, the spectral energy of the quantization noise is concentrated in the harmonics of the fundamental frequency.

Since the quantizer affects only one value at a time, one may wonder how the quantization noise becomes correlated. The mechanism for this phenomenon can be explained through the fact that the quantization noise is correlated with the input value. For example, one can compute this correlation for the quantizer in Example 5.2.1 and obtain

$$\begin{aligned}
\mathrm{E}\left[X\left(X-Q(X)\right)\right] &= \int_0^{16} \frac{x(x-Q(x))}{16}dx \\
&= \sum_{m=1}^{8} \int_{2m-2}^{2m} \frac{x(x-(2m-1))}{16}dx \\
&= \sum_{m=1}^{8} \int_{2m-2}^{2m} \frac{x(x-(2m-1))}{16}dx \\
&= \frac{1}{2}\int_0^2 (x+(2m-2))(x-1)dx \\
&= \frac{1}{2}\int_0^2 x(x-1)dx = \frac{1}{3}.
\end{aligned}$$

From this, we see that the correlation is the same as the MSE. This means that quantizing a pure sinusoid will typically create new spurious harmonics whose powers are proportional to the power in the original sinusoid.

The process of adding a small amount of noise before quantization is called **dithering**. Of course, the added noise increases the overall noise power in the system by a small amount. But, if the noise sequence is chosen to be independent and uniformly distributed over one quantization interval, then correlation becomes exactly zero. To see this, we use the quantizer $Q(x) =$

$2\lfloor x/2 \rfloor + 1$ and let Z be a uniform random variable on $[-1, 1]$. In this case, we get

$$\begin{aligned}
\mathrm{E}\left[Q(X+Z)|X=x\right] &= \int_{-1}^{1} \frac{1}{2} Q(x+z) dz = \int_{-1}^{1} \frac{1}{2} \left(2 \left\lfloor \frac{x+z}{2} \right\rfloor + 1 \right) dz \\
&= \int_{\frac{x-1}{2}}^{\frac{x+1}{2}} \left(\lfloor y \rfloor + \frac{1}{2} \right) 2\, dy = 1 + 2 \int_{\frac{x-1}{2}}^{\frac{x+1}{2}} \lfloor y \rfloor \, dy \\
&= 1 + 2 \left(\left\lfloor \frac{x-1}{2} \right\rfloor \left(1 - \left\{ \frac{x-1}{2} \right\} \right) + \left(\left\lfloor \frac{x-1}{2} \right\rfloor + 1 \right) \left\{ \frac{x-1}{2} \right\} \right) \\
&= 1 + 2 \left(\left\lfloor \frac{x-1}{2} \right\rfloor + \left\{ \frac{x-1}{2} \right\} \right) \\
&= 1 + 2 \frac{x-1}{2} = x,
\end{aligned}$$

where $\{x\} \triangleq x - \lfloor x \rfloor$. Using this, we can compute

$$\begin{aligned}
\mathrm{E}\left[X\left(X - Q(X+Z)\right)\right] &= \int_{-\infty}^{\infty} f_X(x) \int_{-1}^{1} f_Z(z) x (x - Q(x+z))\, dz\, dx \\
&= \int_{-\infty}^{\infty} f_X(x) x (x - x)\, dx = 0.
\end{aligned}$$

This implies that the quantization noise is uncorrelated with the signal. With a little more work, one can also show that it is white (i.e., uncorrelated with time-shifts of itself).

5.2.4 Non-Uniform Quantization via Companding

Uniform quantizers are quite robust because their performance is relatively insensitive to the input distribution. When the input statistics are not uniform, it is possible to design a non-uniform quantizer with the same number of levels and lower SQNR. One way of doing this is by quantizing the signal after it is processed by a **compressor**. After reconstruction, an **expander** is used to reverse the effect of the compresor. Since quantization occurs after compression, the quanitization levels are effectively remapped by the compressor. This whole process is called **companding**.

Assuming the amplitude distribution is supported on $[-x_0, x_0]$, we can define the compressor $g(x)$ to be a strictly increasing function mapping $[-x_0, x_0]$

to $[-1, 1]$. Then, one has $X_q = g^{-1}\left(Q\left(g(X)\right)\right)$, where $Q(\cdot)$ is a uniform quantizer with step size Δ. When Δ is small, the step size Δ_x after compression can be approximated by

$$\Delta_x \approx \frac{\Delta}{g'(x)}.$$

This allows one to approximate the quantization noise variance by

$$E\left[(X_q - X)^2\right] \approx \int_{-x_0}^{x_0} f_X(x) \frac{\Delta_x^2}{12} dx = \frac{\Delta^2}{12} \int_{-x_0}^{x_0} \frac{f_X(x)}{g'(x)^2} dx.$$

This equation for the noise variance is an instance of the Euler-Lagrange equation. Therefore, it can be minimized over $g(x)$ using the calculus of variations and the optimum compressor is defined by

$$g(x) \propto \int_{-x_0}^{x} \sqrt[3]{f_X(z)} dz - \int_{x}^{x_0} \sqrt[3]{f_X(z)} dz.$$

For signals with a wide dynamic range, the SQNR depends heavily on the local average signal power. Using a logarithmic compressor allows one efficiently quantize these signal so that the SQNR is independent of local signal power. Since larger quantization cells are used for large signal values, the quantization noise naturally scales with the signal power. For audio signals, the perceptual effect of noise also depends more on the SQNR than the noise power. Therefore, logarithmic compression is natural choice for such signals.

In fact, transforms of this type have been standardized for use in the telephone system. The standard transform in the United States is called μ-law companding and maps uniformly spaced 14-bit samples into non-uniformly spaced 8-bit samples. The resulting distortion is not easily detectable by the human ear. The μ-law companding algorithm maps $[-1, 1]$ to $[-1, 1]$ using the rule, for some $\mu > 0$,

$$g(x) = \operatorname{sgn}(x) \frac{\ln\left(1 + \mu|x|\right)}{\ln(1 + \mu)}.$$

The benefits of companding are actually due to two different mechanisms. First, the amplitude distribution of speech decays rapidly, so it makes sense to include more quantization levels in the small signal range. Adjusting for this correctly actually increases the SQNR. Second, human perception of audio depends on the SQNR instead of the noise power. So, matching the quantization noise to the perceptual measure improves perceived quality. In many cases, however, it is possible that the SQNR decreases while the perceived quality increases.

5.3 Predictive Quantization

5.3.1 Delta Modulation

When a signal has a wide dynamic range but changes relatively slowly, it can be more effective to quantize the difference between adjacent sample samples. In this case, one can use **differential quantization** to construct the quantized sequence $\widetilde{X}_n$ from the input sequence X_n using the rule

$$\widetilde{X}_n = \alpha \widetilde{X}_{n-1} + Q\left(X_n - \alpha \widetilde{X}_{n-1}\right),$$

for any positive $\alpha \leq 1$. In this system, only the sequence $Y_n = Q\left(X_n - \alpha \widetilde{X}_{n-1}\right)$ must be transmitted or stored. The receiver simply reconstructs the quantized sequence using $\widetilde{X}_n = \alpha \widetilde{X}_{n-1} + Y_n$.

This system is called **delta modulation** when a one-bit quantizer,

$$Q(x) = \begin{cases} +\Delta & \text{if } x \geq 0 \\ -\Delta & \text{if } x < 0 \end{cases},$$

is used. In this case, the signal is typically sampled at a rate much higher than the Nyquist rate (i.e., oversampled) to handle rapid changes in the signal. If the sample rate is F, then the maximum signal slope that can be tracked is ΔF. The distortion caused by the input signal changing faster than this rate is called **slope overload noise**. For a sinusoidal signal $x(t) = A\cos(2\pi f t)$, this effect can be avoided by choosing Δ so that

$$\max_t |x'(t)| = 2\pi A f \leq \Delta F.$$

For a slowly varying signal, the noise due to the quantization step size Δ is called **granular noise**. If the difference in signal amplitude is uniform over $[-\Delta, \Delta]$, then the granular noise power is given by

$$\sigma_G^2 = \int_{-\Delta}^{0} \frac{1}{2\Delta}(x+\Delta)^2 \, dx + \int_0^{\Delta} \frac{1}{2\Delta}(x-\Delta)^2 \, dx = \frac{\Delta^2}{3}.$$

Choosing $\Delta \geq 2\pi Af/F$ to avoid slope overload noise implies that

$$\sigma_G^2 \geq \frac{4\pi^2 A^2 f^2}{3F^2}.$$

Since the signal power in the sinusoid $x(t)$ is $A^2/2$, this gives

$$SQNR = \frac{A^2}{2\sigma_G^2} \leq \frac{3F^2}{8\pi^2 f^2}.$$

Comparing the bit rate versus noise power of this system to standard quantization shows that delta modulation is sometimes better for moderate to low SQNR.

The differential quantization scheme described here is a particular case of a more general approach where the next sample is predicted from the previous samples and subtracted before quantzation. Using better prediction allows one to quantize a residual signal that has less variation.

5.3.2 Estimation and Prediction

Let X, Y be joint random variables defined on common sample space. If X, Y are not independent, then it is often possible to estimate X from Y. For example, let $\hat{X} = g(Y)$ be an estimate of X from Y based on the arbitrary function $g : \mathbb{R} \to \mathbb{R}$. If $Y = y$, then the MSE of this estimate is given by

$$E\left[(X - g(y))^2 | Y = y\right] = E\left[X^2 | Y = y\right] - 2g(y)E\left[X | Y = y\right] + g(y)^2.$$

One can minimize the MSE, for the case where $Y = y$, by taking the derivative w.r.t. to $g(y)$. This shows that $g(y) = E\left[X|Y = y\right]$ minimizes the MSE and leads to a general rule that says "The **MMSE estimate** (i.e., the minimum mean-squared error estimate) is given by the **conditional mean** of the random variable given the observation". Computing the conditional mean, however, can be difficult in many cases.

It turns out that less information is required if we restrict our attention to linear predictors of the form $g(y) = ay + b$. Computing the MSE for this choice of $g(\cdot)$ gives

$$\begin{aligned} E\left[(X - g(Y))^2\right] &= E\left[(X - aY - b)^2\right] \\ &= E\left[X^2\right] - 2E\left[X(aY + b)\right] + E\left[(aY + b)^2\right] \\ &= E\left[X^2\right] - 2aE\left[XY\right] - 2bE\left[X\right] + a^2 E\left[Y^2\right] + 2abE\left[Y\right] + b^2. \end{aligned}$$

In this case, one can minimize the MSE by taking derivatives w.r.t. a, b. This results in

$$a = \frac{E[X]E[Y] - E[XY]}{E[Y]^2 - E[Y^2]} \qquad b = E[X] - aE[Y].$$

This is known as the **LMMSE estimate** (i.e., the linear minimum mean-square error estimate).

More generally, **linear prediction** can be applied to random processes $X_0, X_1, X_2, \ldots$. In this case, the m-th order linear prediction is given by

$$\hat{X}_n = \sum_{i=1}^{m} a_i X_{n-i}.$$

Optimizing the coefficients $a_1, \ldots, a_m$ leads to the matrix equation $Ra = p$, where $R_{ij} = E\left[X_{n-i}X_{n-j}\right]$ and $p_i = E\left[X_n X_{n-i}\right]$. This type of linear prediction is often used to analyze stationary signals. In this case, the matrix R and the vector p become independent of n.

Linear prediction is also the basis of many speech analysis techniques (e.g., speech compression and speech recognition). Since speech signals appear stationary when viewed in 5-10 ms blocks, the optimizing the prediction coefficients for a short block creates a prediction filter for that block. These prediction filters are typically used to remove the effect of the throat-tongue-mouth filter and leave only the glottal pulses generated by the vocal chords. Most of the information in a speech signal is actually carried by the linear prediction coefficients themselves because the glottal pulses define only the pitch and timbre of a voice.

5.4 Optimal Quantization

5.4.1 Uniform Quantizers

Finding an optimal quantizer is not an easy task. An eight-level quantizer has eight degrees of freedom, and the overall performance of the quantizer is jointly determined by the positions of the quantization points. A means to reduce the difficulty of identifying an optimal quantization scheme is to constrain the possible candidates. This can be achieved, for instance, by imposing a rule on the respective position of the quantization points. Restricting the search space to uniform quantizers is one possible way to ensure that an optimal quantizer can be found.

A **uniform quantizer** is a function where the locations of successive outputs are situated at a fixed interval, $q_m - q_{m-1} = \Delta$ for all the possible values. That is, the distance between two quantization points is the same for all neighbors. This scheme is one of the simplest quantizer designs. The quantization function considered in Figure 5.1 is in fact a uniform quantizer. If the objective function of the optimization process is the mean squared error, then optimal locations for the quantization points can be found in a straightforward manner. First, note that we can write the position of the quantization points as

$$q_m = q_1 + (m-1)\Delta$$

for $m = 1, 2, \ldots, M$ where M is the number quantization levels. As usual, the MSE is given by

$$\mathrm{E}[d(X, X_{\mathrm{q}})] = \int_{\mathbb{R}} (x - Q(x))^2 f_X(x) dx.$$

We emphasize that the performance of the quantizer is optimized by minimizing the value of the integrand at each point. We deduce that the decision regions corresponding to $q_1, q_2, \ldots, q_M$ must be equal to

$$\left(-\infty, q_1 + \frac{\Delta}{2}\right], \left(q_2 - \frac{\Delta}{2}, q_2 + \frac{\Delta}{2}\right], \ldots, \left(q_M - \frac{\Delta}{2}, \infty\right),$$

respectively. The objective function then becomes

$$\begin{aligned} \mathrm{MSE} = & \int_{-\infty}^{q_1 + \frac{\Delta}{2}} (x - q_1)^2 f_X(x) dx + \sum_{m=2}^{M-1} \int_{q_m - \frac{\Delta}{2}}^{q_m + \frac{\Delta}{2}} (x - q_m)^2 f_X(x) dx \\ & + \int_{q_M - \frac{\Delta}{2}}^{\infty} (x - q_M)^2 f_X(x) dx. \end{aligned} \tag{5.5}$$

The resulting optimization process has two degrees of freedom, namely q_1 and Δ. For a suitable probability density function $f_X(\cdot)$, an optimal solution can be obtained explicitly using standard optimization methods.

Example 5.4.1. *We revisit Example 5.2.1 in the context of uniform quantizers. Again, suppose that the values of the discrete-time input process are* *uniformly distributed over* $[0, 16]$*. We wish to find the optimal eight-level uniform quantizer associated with this input distribution.*

For simplicity, we assume that the quantization points $\{q_1, q_2, \dots, q_8\}$ *are contained in the interval* $[0, 16]$*. This implies that* $q_1 > 0$ *and* $q_8 < 16$*. The mean squared error as a function of* q_1 *and* Δ *is given by* (5.5)*, which we can rewrite as*

$$MSE\ (q_1, \Delta) = \int_{-q_1}^{\frac{\Delta}{2}} \frac{\xi^2}{16} d\xi + 6 \int_{-\frac{\Delta}{2}}^{\frac{\Delta}{2}} \frac{\xi^2}{16} d\xi + \int_{-\frac{\Delta}{2}}^{16-q_8} \frac{\xi^2}{16} d\xi$$

Recall that, by construction, we can write $q_8 = q_1 + 7\Delta$*. Taking first derivatives with respect to* q_1 *and* Δ*, we obtain*

$$\frac{\partial}{\partial q_1} MSE\ (q_1, \Delta) = \frac{q_1^2}{16} - \frac{(16 - q_1 - 7\Delta)^2}{16}$$
$$\frac{\partial}{\partial \Delta} MSE\ (q_1, \Delta) = \frac{7\Delta^2}{64} - \frac{7(16 - q_1 - 7\Delta)^2}{16}.$$

Setting these derivatives equal to zero, we get $q_1 = 1$ *and* $\Delta = 2$*. A second derivative test ensures that this corresponds to a local minimum. Since this point is the only inflection point of the function MSE* (q_1, Δ) *within our search space, we gather that the quantization scheme of Example 5.2.1 coincide with the optimal uniform quantizer.*

Although the search space for uniform quantizers is much smaller than the set of all possible quantizers, finding an optimal uniform quantizer remains a strenuous task in most situations. The resulting minimization problem need not have a closed-form solution, in contrast to Example 5.4.1. This task is often accomplished by discretizing the search space and applying numerical techniques to identify the best candidate.

5.4.2 Non-Uniform Quantizers

For a **non-uniform quantizer**, the restriction that neighboring quantization points should be equidistant is relaxed. As such, these points can be located anywhere on the real line, and the decision region corresponding to each quantization point need not have a simple structure. The collection of non-uniform quantizers is much larger then the set of uniform quantizers described in Section 5.4.1. This greater flexibility in choosing a quantizer often results in better overall performance. However, it also comes with added complexity in designing the quantization scheme. In this section, we explore some properties of an optimal quantizer for the mean squared error criterion and we present an algorithm that can be employed to obtain a non-uniform quantizer.

First, suppose that the quantization points $\{q_m : m = 1, \ldots, M\}$ are fixed. An optimal quantizer for these points is a function $Q^\star : \mathbb{R} \mapsto \mathcal{Q}$ that minimizes the corresponding MSE,

$$\begin{aligned}\min_Q \mathrm{E}[d(X, X_\mathrm{q})] &= \min_Q \int_\mathbb{R} (x - Q(x))^2 f_X(x) dx \\ &= \int_\mathbb{R} \min_{Q(x)} (x - Q(x))^2 f_X(x) dx.\end{aligned}$$

The last equality follows from the fact that a quantization point can be selected independently for every possible input value. It should then be clear that the value of the function $Q^\star(\cdot)$ evaluated at x is the point q_m that is closest to x,

$$Q^\star(x) = \arg\min_{\{q_m\}} (x - q_m)^2,$$

where ties can be broken arbitrarily. In particular, an optimal M-level scalar quantizer always has M decision regions, each being an interval containing its quantization point.

Also, we can assume that the decision intervals are given. Let the boundaries of these intervals be denoted by $b_1, b_2, \ldots, b_{M-1}$. Then, the corresponding MSE is given by

$$\begin{aligned}&\int_{-\infty}^{b_1} (x - q_1)^2 f_X(x) dx + \sum_{m=2}^{M-1} \int_{b_{m-1}}^{b_m} (x - q_m)^2 f_X(x) dx \\ &+ \int_{b_{M-1}}^{\infty} (x - q_M)^2 f_X(x) dx.\end{aligned} \tag{5.6}$$

The value of the MSE can be minimized by selecting the quantization points $\{q_m : m = 1, \ldots, M\}$ appropriately. Note that each quantization point can be optimized individually, as the integrals in (5.6) have a nice additive structure. Solving one instance of the decoupled problem, we get

$$q_m^\star = \min_{q_m} \int_{b_{m-1}}^{b_m} (x - q_m)^2 f_X(x) dx.$$

To obtain the solution, we differentiate the objective function with respect to q_m and set the derivative equal to zero;

$$\frac{d}{dq_m}\int_{b_{m-1}}^{b_m}(x-q_m)^2 f_X(x)dx = -\int_{b_{m-1}}^{b_m} 2(x-q_m)f_X(x)dx = 0.$$

This implies that

$$q_m^\star = \mathrm{E}[X|b_{m-1} < X \leq b_m].$$

Once the decision intervals are picked, the optimal quantization point $q_m^\star$ is equal to the expectation of X conditioned on X falling inside interval m.

Putting these two observations together, we can summarized the properties of an optimal M-level quantizer as follows. Suppose that $Q^\star : \mathbb{R} \mapsto \mathcal{Q}$ is an optimal quantizer with respect to the mean squared error criterion (or the signal-to-quantization-noise ratio). Then, the decision regions corresponding to the quantization points $\{q_m : m = 1, \ldots, M\}$ form a partition of the real line where each decision set is an interval. Given the positions of the various quantization points $\{q_m : m = 1, \ldots, M\}$, the boundaries of the decision intervals are given by

$$b_m = \frac{q_m + q_{m+1}}{2}$$

where $m = 1, 2, \ldots, M-1$. Furthermore, the quantization point corresponding to decision interval $(b_{m-1}, b_m]$ is equal to the expectation of X conditioned on $b_{m-1} < X \leq b_m$. These properties are collectively known as the Lloyd-Max conditions.

5.4.3 Lloyd-Max Algorithm

The Lloyd-Max conditions enumerated above define a set of rules that are necessarily fulfilled by optimal quantizers. Yet, they do not provide an explicit means to compute the exact positions of the quantization points in an optimal quantizer. Furthermore, it may not be possible to find these points analytically.

A method that is frequently used to find a non-uniform scalar quantizer is the **Lloyd-Max algorithm**. This algorithm starts with an initial assignment for the quantization points which, in this case, we label $q_1^{(0)}, q_2^{(0)}, \ldots, q_M^{(0)}$. The ensuring procedure is to alternate iteratively between the following two steps.

1. Compute the boundaries of the decision intervals according to

$$b_m^{(t)} = \frac{q_m^{(t)} + q_{m+1}^{(t)}}{2}, \quad m = 1, 2, \ldots, M-1.$$

2. Find the updated values of the quantization points,

$$q_m^{(t+1)} = \mathrm{E}[X | b_{m-1} < X \leq b_m], \quad m = 1, 2, \ldots, M.$$

We adopt the simplifying notation $b_0 = -\infty$ and $b_M = \infty$ to express the conditional expectation in a consistent manner. The MSE of the quantization scheme decreases at every step, thereby becoming smaller than the MSE of all the previous assignments. This insures that performance improves with every iteration. The algorithm terminates when a satisfactory level of performance has been achieved, or when the quantization points have converged to their final values.

5.5 Vector Quantizers

A quantizer can work either on single-source outputs or on blocks of source outputs. So far, we have studied the former approach by focusing on scalar quantizers. Although more involved, the second method where multiple outputs are aggregated prior to being quantized typically yields better results. This latter approach, called **vector quantization**, is especially powerful for sources producing signals that are strongly correlated over time. We do not explore the details of vector quantization in this document, however we motivate its purpose through a simple example.

Example 5.5.1. *Assume that a source produces two correlated symbols, denoted X and Y. We are tasked with designing a vector quantizer for this source with a total of four quantization points. The joint probability distribution function associated with this source is known to be*

$$\begin{aligned} f_{X,Y}(x, y) = &\frac{1}{4} g(x + 1.5, y + 1.5) + \frac{1}{4} g(x + 0.5, y + 0.5) \\ &+ \frac{1}{4} g(x - 0.5, y - 0.5) + \frac{1}{4} g(x - 1.5, y - 1.5), \end{aligned} \tag{5.7}$$

where the function $g(\cdot,\cdot)$ is given by

$$g(x,y) = \begin{cases} 1, & |x|,|y| < 0.5 \\ 0, & \textit{otherwise.} \end{cases}$$

For illustrative purposes, $f_{X,Y}(\cdot,\cdot)$ is shown in Figure 5.2. The four quanti-

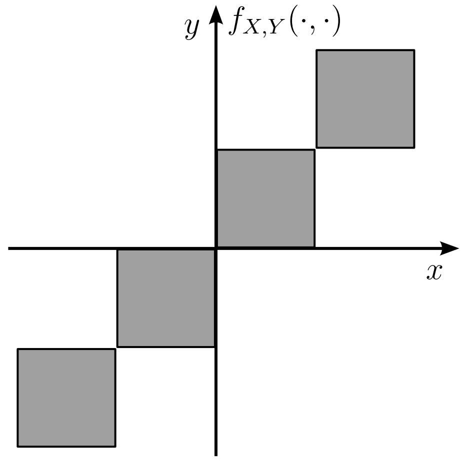

Figure 5.2: A graphical rendering of the joint probability density function defined in (5.7).

zation points for the vector quantizer are located at $q_1 = (-1.5,-1.5)$, $q_2 = (-0.5,-0.5)$, $q_3 = (0.5, 0.5)$, and $q_4 = (1.5, 1.5)$. Let $Q(x,y)$ the the quantizer function that maps each (x,y) point to its nearest q-quantization point. Then, the MSE associated with this vector quantizer can be computed as

$$\begin{aligned} &\mathrm{E}\left[\|(X,Y) - Q(X,Y)\|^2\right] \\ &= \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} \|(x,y) - Q(x,y)\|^2 f_{X,Y}(x,y)dxdy \\ &= 4\int_{-0.5}^{0.5}\int_{-0.5}^{0.5} \frac{x^2+y^2}{4} dxdy = \frac{1}{6}. \end{aligned}$$

That is, the MSE per pair of symbols associated with our vector quantizer is $\frac{1}{6}$. Note that we have used an extended version of the mean squared error that accounts for the 2-D aspect of the problem.

Suppose instead that we wish to use an optimal scalar quantizer instead of the aforementioned vector quantizer. We are then left with two quantization *points per axis. The marginal distributions of the source symbols are given by*

$$f_X(\varphi) = f_Y(\varphi) = \begin{cases} \frac{1}{4}, & |\varphi| < 2 \\ 0, & \textit{otherwise.} \end{cases}$$

The optimal scalar quantization scheme in this case is to put one point at -1 *and the other at* 1*. Once combined, the two scalar quantizers are equivalent to having points at* $r_1 = (1,1)$, $r_2 = (-1,1)$, $r_3 = (-1,-1)$, *and* $r_4 = (1,-1)$ *in the plane. Let* $R(x,y)$ *the the quantizer function that maps each* (x,y) *point to its nearest* r*-quantization point. Then, the resulting MSE per pair of symbols becomes*

$$\begin{aligned}
&\mathrm{E}\left[\|(X,Y) - R(X,Y)\|^2\right] \\
&= \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} \|(x,y) - R(x,y)\|^2 f_{X,Y}(x,y)dxdy \\
&= 4\int_0^1\int_0^1 \frac{x^2+y^2}{4}dxdy = \frac{2}{3}.
\end{aligned}$$

Comparing with our previous result, we notice that the MSE is much larger when using the scalar approach.

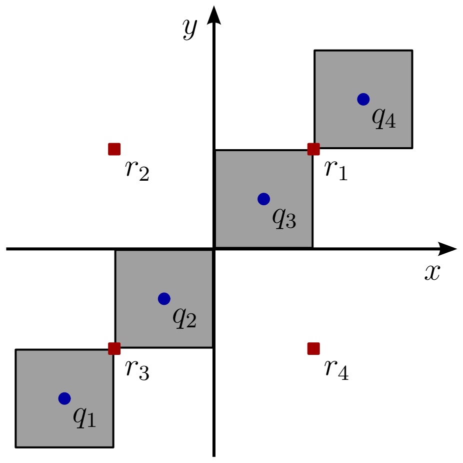

Figure 5.3: A graphical comparison of vector and scalar quantization schemes applied to the two-dimensional problem of Example 5.5.1.

Although this example may not be the most realistic scenario, it provides a good illustration of the potential benefits associated with using vector quantizers. The improved performance comes from the greater flexibility in positioning the various quantization points in a high-dimensional setting together with the ability to exploit correlation among consecutive source symbols. On the downside, the mathematical treatment of a quantization problem becomes more intricate in higher-dimensional spaces.

5.6 Analysis-Synthesis Algorithms

All the quantizers described up to this point are generic schemes designed to work well with abstract sources. In practice, many quantization algorithms are tailored to specific applications. These quantizers are called **analysis-synthesis** coders, and their design is typically fairly intricate. They require advanced models and are often evaluated according to complex performance metrics. These criteria are rooted in human perception rather than conventional mathematics. When developed properly, model-based quantization schemes achieve better compression ratios than the **waveform coders** presented hitherto. A serious treatment of analysis-synthesis algorithms is beyond the scope of this document. Nevertheless, we mention two popular schemes below for illustrative purposes.

Speech Coding: The quantization mechanism employed in speech coding provides a nice example of an analysis-synthesis scheme. Speech coders are widely used in mobile telephony and VoIP. Human speech is modeled as a much simpler random process than most other audio signals, and there is an abundance of knowledge about its statistical properties and the way voice is generated. As a result, some auditory information which is relevant in generic audio signals becomes inconsequential in the context of speech. The primary performance criteria for voice signals are intelligibility and pleasantness of the received signal. In addition, most speech applications require low delay, as long delays interfere with speech interaction in real-time applications. The **Code Excited Linear Prediction** (CELP) is a class of algorithms developed for human speech. The basic idea behind this approach is to model human speech production using a time-varying linear filter. The speech samples are subsequently separated in two distinct parts. The first component contains the current parameters governing the operation of the linear filter; these parameters are selected from a finite set of possible values. The second component captures the residual error, the difference between the predicted signal and the actual one. This second signal is quantized using standard waveform coding techniques. The overall operation of the system works quite well for conversations. However, a speech coder applied to music fails to provide an adequate rendition of the original signal.

Joint Photographic Experts Group (JPEG): The JPEG algorithm is a file format designed to store photographs and paintings digitally. The acronym JPEG is derived from the name of the committee that created this standard. A JPEG file can actually be created in various ways. A commonly used procedure specified in this standard is the **JPEG file interchange format** (JFIF), which we describe briefly. The encoding process consists of several steps. First, an image is represented using YCbCr, an encoding scheme that specifies every pixel (sample point) in the image according to a light intensity component (Y) and two chroma (Cb and Cr) for colors. This scheme, as opposed to RGB, is interesting because it parallels the way the human visual system perceives image elements. The image is then split into blocks of 8×8 pixels; and for each block, the Y, Cb, and Cr data undergoes a two-dimensional cosine transform. This step is similar to a Fourier transform in the sense that it produces a spatial frequency spectrum. The amplitudes of the resulting frequency components are quantized. The resolution of the chroma data is reduced, compared to the light intensity component. This reflects the fact that the human eye is less sensitive to fine color details than to fine brightness details. Furthermore, human perception is much more sensitive to small variations in color or brightness over large areas than to the strength of high-frequency brightness variations. Thus, the magnitudes of the high-frequency components are stored with a lower accuracy than the low-frequency components. The resulting data for all 8×8 blocks is further compressed with a lossless algorithm that is a variant of the **Huffman code**. The important concept exposed in this example is how JPEG is built with a specific application in mind, and therefore quantizes sample data as to minimize the perceived distortion. This is a very good illustration of an analysis-synthesis quantizer.

Chapter 6

Channel Coding

6.1 Introduction

6.1.1 What is channel coding and why do we use it?

Channel coding is the art of adding redundancy to a message in order to make it more robust against noise. It is used because noise and errors are essentially unavoidable in many systems (e.g., wireless communications and magnetic storage). Coding allows one to trade-off rate for reliability and usually provides large gains in overall system efficiency. In contrast, source coding (or compression) is used to remove the redundancy from sources (e.g., zip files JPEG pictures). Channel coding carefully adds redundancy to a message so that it can be transmitted reliably over noisy channels.

Example 6.1.1 (Repeat Code). *Consider the 1 bit message $u \in \{0,1\}$ mapped to a codeword of length $2t+1$ by repeating the message bit. This gives a binary code with two codewords:*

$$\mathcal{C} = \{\underbrace{00\ldots00}_{2t+1}, \underbrace{11\ldots11}_{2t+1}\}.$$

If fewer than t errors, then received sequence is closer to correct codeword. Therefore, a decoder which chooses the codeword closest to the received sequence will decode successfully. For a binary code, the code rate is defined to be

$$R = \frac{\#\ \textit{information bits}}{\#\ \textit{code bits}},$$

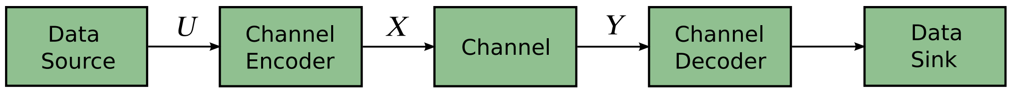

Figure 6.1: Block diagram of digital communication from a coding perspective.

and this gives $\frac{1}{2t+1}$ *for the repeat code.*

Example 6.1.2 (Credit Card Numbers)**.** *Credit card numbers use a check digit to detect single errors and adjacent transpositions. Let* $\underline{x}$ *be a credit card number whose digits are given by* $\underline{x} = (x_1, x_2, \ldots, x_{16})$*, then*

$$\left[\sum_{i=1}^{8} x_{2i} + \sum_{i=1}^{8} (2x_{2i-1} \bmod 9)\right] \bmod 10 = 0$$

Consider the number 5420 1213 7904 9210. In this case, the first sum gives $4+0+2+3+9+4+2+0 = 24$ *and the second sum gives:* $1+4+2+2+5+0+0+2 = 16$*. So, we have the overall sum* $[24+16] \bmod 10 = 0$*. The code detects single errors because, for* $i = 1, \ldots, 8$*, changing* x_{2i} *to* x'_{2i} *changes the check value by* $x'_{2i} - x_{2i}$ *and changing* x_{2i-1} *to* x'_{2i-1} *changes the check value by* $2(x'_{2i} - x_{2i}) \bmod 9$*. Likewise, swapping* x_1, x_2 *changes the check by*

$$[(x_1 - x_2) + (2x_2 \bmod 9 - 2x_1 \bmod 9)] \bmod 10.$$

Coding is used in many systems and devices including:

- CD / DVD players : Modulation code + Reed-Solomon (RS) code
- Digital Video Broadcasting (DVB): Convolutional Code + RS code
- Deep Space Communications: Advanced Turbo and LDPC codes
- Cell Phones: Convolutional codes for voice and Turbo/LDPC codes for data

6.1.2 Channels and Error Models

When designing and analyzing channel codes, one often uses a simple model of a communications channel known as a **discrete memoryless channel** (DMC). The channel input X is an element of input alphabet $\mathcal{X}$ and the channel output Y is an element of the output alphabet $\mathcal{Y}$. The main assumption is that each channel use is indepedent and governed by the probability law

$$W(y|x) \triangleq \Pr(Y = y|X = x).$$

[Add figure with transition diagrams]

The **binary symmetric channel** (BSC) with error probability p is defined by $\mathcal{X} = \mathcal{Y} = \{0, 1\}$ and

$$W(y|x) = \begin{cases} p & \text{if } x \neq y \\ 1-p & \text{if } x = y \end{cases}$$

The **binary erasure channel** (BEC) with erasure probability ϵ is defined by $\mathcal{X} = \{0, 1\}$, $\mathcal{Y} = \{0, 1, ?\}$ and

$$W(y|x) = \begin{cases} \epsilon & \text{if } y = ? \\ 1-\epsilon & \text{if } x = y \\ 0 & \text{if } x \neq y \end{cases}$$

The **binary-input AWGN channel** (BIAWGN) with $\sigma^2 = \frac{N_0}{2}$ is defined by $\mathcal{X} = \{-1, 1\}$, $\mathcal{Y} = \mathbb{R}$ and

$$W(y|x) = \frac{1}{\sqrt{\pi N_0}} e^{-|y-x|^2/N_0},$$

where N_0 is the noise spectral density at the receiver.

The SNR of communication system is defined in terms of the **energy per information bit**,E_b, and the average **energy per transmitted symbol**, E_s. The conversion between these two quantities is based on keeping track of the units

$$E_s = \frac{\#\text{ information bits}}{\#\text{ transmitted symbols}} \frac{\text{Energy}}{\text{information bit}} = R' E_b.$$

The information rate R' (bits/channel use) is equal to the code rate R for binary-input channels. To make a fair comparisons, one must use the rate-normalized quantity E_b/N_0 (pronounced ebb-no). The normalization adjusts for the extra energy used to send parity symbols. The **coding gain** is the reduction in required E_b/N_0 to achieve a particular error rate. In other cases, it more convenient to use the quantity E_s/N_0 (pronounced ess-no).

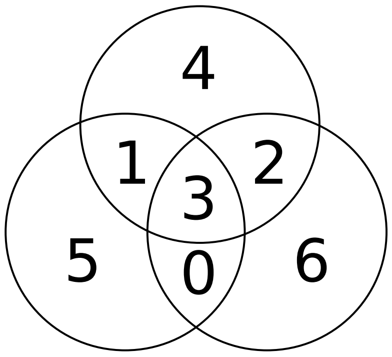

Figure 6.2: Venn diagram representation of the (7,4) binary Hamming code.

For example, a bit-error rate (BER) of 10^{-5} is achieved by uncoded BPSK transmission with $E_b/N_0 = 9.6$ dB. Whereas, a rate-$\frac{1}{2}$ code with moderate decoding complexity (Viterbi decoding of a convolutional code) has a BER of 10^{-5} at $E_b/N_0 = 4.2$ dB. The coding gain in this case is $9.6 - 4.2 = 5.4$ dB

6.2 The Basics of Coding

6.2.1 Codes

Let $\mathcal{X}$ be the input alphabet of a channel (assume $|\mathcal{X}| = q$)

Definition 6.2.1. *A length-n* ***code*** *over the alphabet $\mathcal{X}$ is simply a subset $\mathcal{C} \subseteq \mathcal{X}^n$ of all input sequences.*

If a binary code has $M = |\mathcal{C}|$ codewords, then the code rate is $R = \frac{\log_2 M}{n}$. This means we can send k information bits when $M = 2^k$.

For example, the binary repeat code of length 5 is defined by $\mathcal{X} = \{0, 1\}$ and

$$\mathcal{C} = \{00000, 11111\} \subset \{0,1\}^5 .$$

Likewise, the binary even parity code of length 3 is

$$\mathcal{C} = \{000, 110, 101, 011\} \subset \{0,1\}^3 .$$

Definition 6.2.2. *The* ***Hamming distance*** *$d(\underline{x}, \underline{y})$ is equal to the number of places where the vectors differ. It can be defined mathematically by*

$$d(\underline{x}, \underline{y}) = \sum_{i=1}^{n} (1 - \delta_{x_i,y_i}),$$

where $\delta_{a,b}$ is Kronecker delta function

$$\delta_{a,b} = \begin{cases} 0 & \text{if } a \neq b \\ 1 & \text{if } a = b \end{cases}.$$

The distance between codewords is typically measured with the Hamming distance. Using this metric, the set $\mathcal{X}^n$ forms a discrete metric space Another important code parameter is the minimum distance d_{min} between any two codewords is

$$d_{min} \triangleq \min_{\underline{x},\underline{y} \in \mathcal{C}, \underline{x} \neq \underline{y}} d(\underline{x}, \underline{y}).$$

Example 6.2.3 (Hamming Code)**.** *The (7,4) binary Hamming Code has $n = 7$, $M = 16$, and $d_{min} = 3$. The code can be defined in terms of a Venn diagram showing three partially overlapping sets. Each of the seven subregions represent a code bit and the three circles represent even parity costraints. Encoding can be done by choosing $x_0, \ldots, x_3$ arbitrarily and then computing the last three parity bits. Any single error can be corrected by observing each bit error gives a unique pattern of parity violations. The codewords can be listed as follows:*

0000000	0100110	1000011	1100101
0001111	0101001	1001100	1101010
0010101	0110011	1010110	1110000
0011010	0111100	1011001	1111111

6.2.2 Decoding

Consider the decoding problem for binary codes with $\mathcal{X} = \mathcal{Y} = \{0, 1\}$ and $\mathcal{C} \subseteq \mathcal{X}^n$. The channel input is $\underline{x} \in \mathcal{C}$, the received sequence is $\underline{y}$, and the number of errors is $t = d(\underline{x}, \underline{y})$ It is not hard to verify that minimum distance decoding, which returns the codeword closest to the channel output, is optimal. Breaking ties arbitrarily, one can write

$$\hat{\underline{x}} = \arg\min_{\underline{w} \in \mathcal{C}} d(\underline{w}, \underline{y})$$

The following implies that the minimum distance decoder can always correct t errors if $d_{min} \geq 2t + 1$.

Proposition 6.2.4. *For any received sequence $\underline{y}$, there is at most one codeword $\underline{w}$ such that $d(\underline{y},\underline{w}) \leq \frac{d_{min}-1}{2}$.*

Proof. Suppose there are codewords $\underline{w},\underline{z}$ where $d(\underline{y},\underline{w})$ and $d(\underline{y},\underline{z})$ are $\leq \frac{d_{min}-1}{2}$. Then, the triangle inequality implies $d(\underline{w},\underline{z}) \leq d(\underline{y},\underline{w}) + d(\underline{y},\underline{z}) \leq d_{min}-1$ and contradicts the definition of d_{min}. Therefore, if $t \leq \frac{d_{min}-1}{2}$, then $\underline{x}$ is the unique codeword such that $d(\underline{x},\underline{y}) \leq \frac{d_{min}-1}{2}$. □

The following allows simultaneous error correction of t errors and detection of d errors.

Proposition 6.2.5. *If $d_{min} \geq t+d+1$ and $d \geq t$, then a single decoder can both correct t and detect d errors.*

Proof. Assume that each codeword is surrounded a inner ball of radius t and an outer ball of radius d. If the received vector is in an inner ball, decode to the codeword at the center. Otherwise, declare decoding failure.

From the previous result, we see that no two inner balls overlap and that the inner ball of one codeword does not overlap the outer ball of any codeword. If the number of errors is at most t, then received vector will be in the inner ball of the transmitted codeword and will be decoded correctly. If the number of errors is between $t+1$ and d, then received vector will not be in the inner ball of any codeword and failure will be declared. □

Proposition 6.2.6. *If $d_{min} \geq e+1$, then there is a decoder which corrects all patterns of e erasures.*

Proof. Make a list of all codewords and then erase any e positions. Each erasure reduces the minimum distance between any two codewords by at most one. After e steps, the new $d_{min} \geq e+1-e = 1$. This implies that the codewords, defined by the remaining symbols, are all unique. □

6.3 Binary Linear Codes

6.3.1 Basic Properties

This chapter focuses almost exclusively on binary linear codes, which are the simplest and most important class of error-correcting codes. The restriction

+	0	1
0	0	1
1	1	0

*	0	1
0	0	0
1	0	1

Table 6.1: The addition and multiplication operations for the binary field.

to linear codes can be motivated by two things: simplicity and performance. We will see later that linear codes are much simpler to describe, encode, and analyze. Moreover, there are very few cases where non-linear codes are better than linear codes. So, there is essentially no performance penalty for this simplicity.

Linear codes, like linear algebra, make use of matrices and vectors of elements that can be added, subtracted, multipled, and divided. A set of numbers which obey all the standard rules of arithmetic is an algebraic object known as a **field**. For example, the real and complex numbers are both fields.

There are also fields which have a finite number of elements. Let a, d be positive integers so that the division of a by d gives the equation $a = dq + r$, where q is quotient and $0 \leq r \leq d-1$ is the remainder. The modulo operation is defined to return the remainder from division and is denoted $r = a \bmod d$. It turns out that the binary alphabet, $\{0,1\}$, with standard arithmetic $(+, -, *, /)$ performed modulo 2 is also a field. The operations are shown explicitly in Table 6.1.

For linear algebra over a field, the scalar (i.e., field) operations are used to define vector and matrix operations. Vector addition is defined element-wise, so that $\left[\underline{x}+\underline{y}\right]_i = x_i + y_i$. An (n,k) **binary linear code** is $\mathcal{C} \subseteq \{0,1\}^n$ with $|\mathcal{C}| = 2^k$ where $\underline{x}, \underline{y} \in \mathcal{C}$ implies $\underline{x}+\underline{y} \in \mathcal{C}$ Since $\underline{x}+\underline{x} = \underline{0}$, this implies all zero vector $\underline{0} \in \mathcal{C}$. For example, the $n = 3$ "even parity" code is a (3,2) binary linear code with codewords $\mathcal{C} = \{000, 110, 101, 011\}$.

Definition 6.3.1. *The* ***Hamming weight*** $w(\underline{x})$ *is the number of non-zero elements in* $\underline{x}$ *or*

$$w(\underline{x}) = \sum_{i=1}^{n} (1 - \delta_{x_i,0}).$$

For binary vectors, this also implies that the Hamming distance is given by

$$d(\underline{x}, \underline{y}) = w(\underline{x} - \underline{y}).$$

Linear codes also have a simplified distance structure. Instead of considering the minimum distance between all codewords, it suffices to focus only on the all-zero codeword.

Proposition 6.3.2. *The minimum distance of a linear code is given by*

$$d_{min} = \min_{\underline{x} \in \mathcal{C}, \underline{x} \neq \underline{0}} w(\underline{x} - \underline{y}).$$

Proof. The linear property of the code allows one to translate computations involving the distance between two codewords to expressions involving the Hamming weight of one codeword. This gives

$$\begin{aligned} d_{min} &\triangleq \min_{\underline{x}, \underline{y} \in \mathcal{C}, \underline{x} \neq \underline{y}} d(\underline{x}, \underline{y}) \\ &= \min_{\underline{x}, \underline{y} \in \mathcal{C}, \underline{x} \neq \underline{y}} w(\underline{x} - \underline{y}) \\ &= \min_{\underline{x} \in \mathcal{C}, \underline{x} \neq \underline{0}} w(\underline{x} - \underline{y}), \end{aligned}$$

where the last step follows from the fact that

$$\left\{ \underline{x} - \underline{y} \,|\, \underline{x}, \underline{y} \in \mathcal{C}, \underline{x} \neq \underline{y} \right\} = \left\{ \underline{x} \in \mathcal{C} \,|\, \underline{x} \neq \underline{0} \right\}.$$

□

6.3.2 Generator and Parity-Check Matrices

Linear codes can be represented compactly using matrices. The generator defines the code by allowing one to list all the codewords.

Definition 6.3.3. *The* ***generator matrix*** $\underline{G}$ *of an* (n, k) *binary linear code is a* $k \times n$ *binary matrix such that all codewords,* $\underline{x} \in \mathcal{C}$*, can be written as* $\underline{u} \cdot \underline{G} = \underline{x}$ *for some message vector* $\underline{u} \in \{0, 1\}^k$*. Therefore, the code is the row space of* $\underline{G}$*).*

Example 6.3.4. *The generator matrix*

$$\underline{G} = \begin{bmatrix} 1 & 0 & 1 & 1 & 0 \\ 1 & 1 & 1 & 0 & 1 \end{bmatrix}$$

defines the $(5,2)$ *code*

$$\mathcal{C} = \{00000, 10110, 01011, 11101\}.$$

Encoding $\underline{u} = [11]$ *gives* $\underline{u} \cdot \underline{G} = [0\ 1\ 0\ 1\ 1]$

If $\underline{G}$ has full rank k (over the binary field), then the code has 2^k distinct codewords. Otherwise, some non-zero messages encode to the all-zero codeword and there are at most 2^{k-1} codewords.

Definition 6.3.5. *The generator matrix is in **systematic form** if* $\underline{G} = [\underline{P}\ \ \underline{I}_k]$*, where* $\underline{I}_k$ *is the* $k \times k$ *identity matrix. The matrix* $\underline{P}$ *is called the **parity generator** of the code because* $\underline{u} \cdot \underline{P}$ *computes the parity bits for* $\underline{u}$*. For a generator matrix in systematic form, the message vector appears in codeword*

$$\underline{u} \cdot \underline{G} = \underline{u} \cdot [\underline{P}\ \ \underline{I}_k] = [\underline{u} \cdot \underline{P}\ \ \underline{u}].$$

The parity-check (PC) matrix defines the code by listing the parity-check equations that each codeword must satisfy.

Definition 6.3.6. *The **parity-check matrix*** $\underline{H}$ *of an* (n,k) *binary linear code is an* $(n-k) \times n$ *binary matrix such that* $\underline{x} \cdot \underline{H}^T = \underline{0}$ *for all* $\underline{x} \in \mathcal{C}$*. Therefore, the code is the null space of* $\underline{H}$*.*

While the generator matrix defines the code and an encoder, the parity-check matrix defines only the code; there is no implied encoder. There is also a relationship between and generator and parity-check matrix for the same code. Recall that, for all codewords $\underline{x}$, there is a message $\underline{u}$ such that $\underline{x} = \underline{u} \cdot \underline{G}$. This means that $\underline{G} \cdot \underline{H}^T = \underline{0}$.

Example 6.3.7. *For the* $(5,2)$ *code we saw previously, one possible parity-check matrix is*

$$\underline{H} = \begin{bmatrix} 1 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 1 & 1 \end{bmatrix}.$$

Notice that it satisfies $\underline{G} \cdot \underline{H}^T = 0$ *for the previous* $\underline{G}$*.*

In general, we assume that $\underline{H}$ has full rank. Otherwise, there are redundant constraints and some row can eliminated without changing the code.

A parity-check matrix is in **systematic form** if $H = \left[\underline{I}_{n-k} \quad -\underline{P}^T\right]$. When both the generator and parity-check matrices are in systematic form, we can write

$$\begin{aligned}
\underline{G} \cdot \underline{H}^T &= [\underline{P} \;\; \underline{I}_k] \cdot \left[\underline{I}_{n-k} \quad -\underline{P}^T\right]^T \\
&= [\underline{P} \;\; \underline{I}_k] \cdot \begin{bmatrix} \underline{I}_{n-k} \\ -\underline{P} \end{bmatrix} \\
&= \underline{P} - \underline{P} \\
&= \underline{0}.
\end{aligned}$$

Example 6.3.8. *A* ***single parity-check code*** *is an $(n, n-1)$ binary linear code with parity-check matrix*

$$H = [\underbrace{1, 1, \ldots, 1}_{n \text{ times}}].$$

For all codewords $\underline{x}$, the parity-check constraint $\underline{x} \cdot \underline{H}^T = \underline{0}$ implies that $\sum_{i=1}^n x_i \bmod 2 = 0$ (i.e., the codeword has an even number of ones). The minimum distance is $d_{min} = 2$ and the generator matrix is given by

$$G = \begin{bmatrix} 1 & 1 & 0 & \cdots & 0 \\ 1 & 0 & 1 & \cdots & 0 \\ \vdots & 0 & \ldots & \ddots & 0 \\ 1 & 0 & 0 & 0 & 1 \end{bmatrix}.$$

Next, we consider the minimum distance of a code in terms of its generator and parity check matrices. In general, it is very difficult to compute the minimum distance without enumerating all codewords. But, one can get upper and lower bounds on the minimum distance much more easily. In this way, the minimum distance can be found approximately.

Since the minimum distance is equal to the minimum of the Hamming weight overall codewords, it is clearly upper bounded by the Hamming weight of any particular non-zero codeword. This gives, for any non-zero codewords $\underline{x}$,

$$d_{min} \leq w(\underline{x}).$$

Likewise, the parity-check matrix can be used to lower bound the minimum distance.

Proposition 6.3.9. *The minimum distance of a code with parity-check matrix*

$$\underline{H} = [\underline{h}_1, \underline{h}_2, \ldots, \underline{h}_n]$$

is equal to the minimum number of columns that sum to zero or

$$d_{min} = \min \left\{ w(\underline{x}) | \underline{x} \cdot \underline{H}^T = \underline{0}, \underline{x} \in \{0,1\}^n \backslash \underline{0} \right\}.$$

Proof. Notice that

$$\underline{x} \cdot \underline{H}^T = \sum_{i=1}^{n} x_i \underline{h}_i = \sum_{i:x_i=1} \underline{h}_i,$$

where $\underline{h}_i$is the ith column of $\underline{H}$. Therefore, the statement that $\underline{x} \cdot \underline{H}^T = \underline{0}$ (i.e., $\underline{x}$ is a codeword) is equivalent to the statement that the sum of $w(\underline{x})$ columns is $\underline{0}$. Taking the minimum over all non-zero codewords gives the minimum distance. □

One can also bound the minimum distance in terms of error-correction ability. Recall that a code corrects all error patterns of weight t if and only if $d_{min} \geq 2t + 1$. This gives a simple lower bound on the minimum distance.

Example 6.3.10. *Correcting all single errors requires* $\frac{d_{min}-1}{2} = 1$ *or* $d_{min} = 3$. *Let us try to find the longest code, for a fixed number of parity bits* m*, that corrects all single errors. In this case,* $\underline{H}$ *is matrix with* m *rows and we can add columns, one at a time, until it is not possible to add a column without losing the ability to correct a single error. How many columns can we choose? The maximum value is* $n = 2^m - 1$ *and the resuting optimal code is called the binary* ***Hamming code*** *of length* n*.*

6.3.3 Decoding

Assume a codeword $\underline{x} \in \mathcal{C}$ is transmitted over a channel and $\underline{r} = \underline{x} + \underline{e}$ is received, where $\underline{e}$ is the **error pattern** with $w(\underline{e})$ errors. For any received sequence $\underline{r}$, a **decoder** either returns a codeword $\hat{\underline{x}} = D(\underline{r})$ or declares failure. The decoded message $\hat{\underline{u}}$ associated with $\hat{\underline{x}}$ is the unique message satisfying $\hat{\underline{x}} = \underline{\hat{u}G}$. A decoder makes

- a **block error** (or word error) if $\underline{\hat{x}} \neq \underline{x}$ and P_B is used to denote the probability of block error,
- b **code bit errors** if $w(\underline{\hat{x}} - \underline{x}) = b$ and P_b is used to denote the probability that a randomly chosen bit in $\underline{\hat{x}}$ is in error,
- and b **message bit errors** if $w(\underline{\hat{u}} - \underline{u}) = b$ and P_b is used to denote the probability that a randomly chosen bit in $\underline{\hat{u}}$ is in error.

From this, we see that the probability of bit error P_b can have multiple meanings. The correct meaning can usually be inferred from the context. Decoding can also be simplified for linear codes.

Definition 6.3.11. *Let $\underline{s} = \underline{r} \cdot \underline{H}^T$ be the* ***syndrome*** *of the received vector $\underline{r}$.*

It turns out that $\underline{s}$ depends only on the error pattern. Since $\underline{x} \in \mathcal{C}$, we have

$$\underline{s} = \underline{r} \cdot \underline{H}^T = (\underline{x} + \underline{e}) \cdot \underline{H}^T = \underline{x} \cdot \underline{H}^T + \underline{e} \cdot \underline{H}^T = \underline{e} \cdot \underline{H}^T.$$

A **syndrome decoder** $\underline{\hat{e}} = D_s(\underline{s})$ maps the syndrome $\underline{s}$ to an estimated error pattern $\underline{\hat{e}}$. Let us define the equivalence relation $\sim$ by $\underline{x} \sim \underline{y}$ iff $\underline{x} \cdot \underline{H}^T = \underline{y} \cdot \underline{H}^T$. This means that two binary vectors are equivalent if they have the same syndrome. A syndrome decoder can be designed correct exactly one error pattern in each equivalence class. The best choice for correction is the most-likely error pattern in that equivalence class. For most channels, these vectors are chosen to be the minimum weight vector in the equivalence class.

The **standard array** is way of listing all vectors of length-n that exposes the connectiion between syndromes, codewords, and error correction. In general, it is a $2^{n-k} \times 2^k$ table that contains each length-n binary vector exactly once. The main idea behind this table is that, when one chooses to correct a particular error pattern, the decoder is automatically defined for all received sequences equal to that error pattern plus a codeword. Of course, this limits one's ability choose correctable error patterns.

Each row is indexed by a syndrome $\underline{s}$ and contains all binary vectors $\underline{x}$ that satisfy the equation $\underline{s} = \underline{x} \cdot \underline{H}^T$. The first row is reserved for $\underline{s} = \underline{0}$ and contains the all-zero codeword $\underline{c}_1$ in the first column followed the remaining codewords $\underline{c}_2, \ldots, \underline{c}_{2^k}$ in any order. The first column contains the correctable error patterns $\underline{e}_1, \ldots, \underline{e}_{2^{n-k}}$ where $\underline{e}_1$ is the all-zero sequence. The row-i column-j entry always contains $\underline{e}_i + \underline{c}_j$, and is therefore defined by the first row and first column of the table. The column associated with a particular codeword can be seen as all the received vectors that will be decoded to that codeword.

Using the parity-check matrix for our $(5, 2)$ our example code, the process is as follows

1. Start with 8 by 5 table and list the zero syndrome and all codewords on first row
2. Pick a minimum weight vector of length n that is not already in table and compute its syndrome
3. Add a new row by writing syndrome followed by the minimum weight vector plus each codeword
4. Repeat from step 2 until table is complete.

The resulting standard array is

syn\cw	00000	10110	01011	11101
100	10000	00110		01101
010	01000		00011	
001	00100	10010		11001
101	00010		01001	
111	00001	10111		11100
110	11000	01110	10011	
011	01100	11010		10001

To see if you understand this example, try filling in the missing entries. One can prove that this process always enumerates all 2^n binary vectors, so you can test your answers by checking if all binary vectors appear in the table exactly once.

Example 6.3.12. *To see syndrome decoding in action, let* $\underline{x} = 10110$ *and* $\underline{e} = 11000$*. Then,* $\underline{r} = 01110$ *and* $\underline{s} = 110$*. Looking in the syndrome table, we find that* $\hat{\underline{e}} = 11000$ *and* $\hat{\underline{x}} = 10110$*.*

6.3.4 Manipulating Linear Codes

It follows from linear algebra that any $\underline{G}, \underline{H}$ can be put in systematic form using elementary row operations and (possibly) a column permutation. For the parity-check matrix H, the basic idea is to use elemtary row operations to form an identity matrix in the first few columns (i.e., put it in reduced row-echelon form). For the generator matrix G, elementary row operations are used to form an identity matrix in the last few columns. Sometimes an identity cannot be formed in the desired columns and a column permutation is required to complete the process. For this reason, two codes are called **equivalent** if and only if they differ only in the order of the code symbols.

Definition 6.3.13. *For a matrix, an* ***elementary row operation*** *is any one of the following operations:*

1. *interchaging any two rows,*
2. *scaling any row by a non-zero constant,*
3. *and adding a multiple of one row to another row.*

Example 6.3.14. *In this example, we consider a parity-check matrix*

$$\underline{H} = \begin{bmatrix} 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 & 1 & 1 \end{bmatrix}$$

that requires a column permutation for systematic form. Let us put $\underline{H}$ is reduced row-echelon form and then find a column permutation to achieve an identity in the first few rows. The first step gives

$$\underline{H} \rightarrow \begin{bmatrix} 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 \end{bmatrix} \rightarrow \begin{bmatrix} 1 & 0 & 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 \end{bmatrix} \rightarrow \begin{bmatrix} 1 & 0 & 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 \end{bmatrix},$$

and a column permutation gives

$$\tilde{\underline{H}} = \begin{bmatrix} 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 \end{bmatrix}.$$

Now, we can compute parity generator and generator associated with $\underline{\tilde{H}}$ to get

$$\underline{\tilde{P}} = \begin{bmatrix} 1 & 1 & 0 \\ 0 & 1 & 1 \\ 1 & 0 & 1 \end{bmatrix} \quad \underline{\tilde{G}} = \begin{bmatrix} 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 & 1 \end{bmatrix}.$$

Finally, reversing the column permutation gives, for the original code, a generator

$$\underline{G} = \begin{bmatrix} 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 & 0 & 1 \end{bmatrix}$$

that contains an identity in a subset of its columns.

6.4 Performance Analysis

Consider an (n,k) binary code of rate $R = k/n$ with $d_{min} \geq 2t^* + 1$ and a decoder that corrects all error patterns of weight at most t^*. It is relatively easy to upper bound the probability of block error when this code and decoder are used on a BSC, because a block error cannot occur unless there are more than t^* errors. This gives

$$P_B \leq 1 - \sum_{i=0}^{t^*} \binom{n}{i} p^i (1-p)^{n-i} = \sum_{i=t^*+1}^{n} \binom{n}{i} p^i (1-p)^{n-i}.$$

If p is small enough, then we can approximate this by the first term in the sum

$$P_B \approx \binom{n}{t^*+1} p^{t^*+1}.$$

If the code bits are transmitted over an AWGN channel using BPSK followed by a hard-decision detector, then we also have

$$p = Q\left(\sqrt{\frac{2E_s}{N_0}}\right) = Q\left(\sqrt{\frac{2RE_b}{N_0}}\right).$$

The latter expression allows us to make relatively fair comparisons between coding systems with different rates.

Definition 6.4.1. *The **coding gain** achieved by a channel code is the reduction in E_b/N_0 required achieve a particular error rate. For example, if the uncoded system achieves an error rate of 10^{-5} at $E_b/N_0 = 9.6$ dB and the coded system achieves an error rate of 10^{-5} at $E_b/N_0 = 6.6 dB$, then the coding gain is $9.6 - 6.6 = 3$ dB at the error rate 10^{-5}. The coding gain often approaches a limit, for low error rates, known as the **asymptotic coding gain**.*

Now, we compute the asymptotic coding gain, in terms of t^* and R, of hard-decision decoding of a block code. For the uncoded system, let γ_u be the E_b/N_0 so that the probability of block error is given by

$$P_B^{(u)}(\gamma_u) = 1 - \left(1 - Q\left(\sqrt{2\gamma_u}\right)\right)^k .$$

For the coded system, let γ_c be the E_b/N_0 so that the probability of block error is

$$P_B^{(c)}(\gamma_c) \approx \binom{n}{t^*+1} Q\left(\sqrt{2\gamma_c}\right)^{t^*+1} .$$

The exponential decay rate of these error probabilities with E_b/N_0 is given by

$$\lim_{\gamma \to \infty} \frac{1}{\gamma} \ln\left(P_B(\gamma)\right) .$$

For large x, the Q-function is very well approximated by

$$Q(x) \approx \frac{1}{\sqrt{2\pi x^2}} e^{-x^2/2} .$$

This implies that the exponential decay rate of $P_B^{(u)}$ is γ_u and the exponential decay rate of $P_B^{(c)}$ is $(t^*+1)R\gamma_c$. Matching these exponential decay rates gives the equation

$$\frac{\gamma_u}{\gamma_c} = (t^*+1)R,$$

and converting to dB shows that the asymptotic coding gain is

$$10 \log_{10}\left((t^*+1)R\right) .$$

Example 6.4.2. *Consider the $(15, 11)$ binary Hamming code with $d_{min} = 3$ and $t^* = 1$. This code achieves an asymptotic coding gain of*

$$10 \log_{10}\left(2 \frac{11}{15}\right) \approx 1.66 \text{ dB}.$$

For syndrome decoding, one can compute exactly the probability of block error (including decoding failure) by observing that the decoder only returns the correct codeword if the error vector is a coset leader. In this case, we can define A_h be the number of coset leaders with Hamming weight h and write

$$P_B = 1 - \sum_{h=0}^{n} A_h p^h (1-p)^{n-h}.$$

6.5 Cyclic Codes

6.5.1 Basic Properties

In this section, we consider a subset of linear codes which have even more algebraic structure. A cyclic (or circular) shift of a vector is another vector with the same elements in the same order, but starting from an different index. For example, the left circular shift of $(x_0, x_1, \ldots, x_{n-1})$ by 1 position gives the vector $(x_{n-1}, x_0, x_1, \ldots, x_{n-2})$.

Definition 6.5.1. *A* ***cyclic code*** *is a linear code where any cyclic shift of a codeword is also a codeword.*

Example 6.5.2. *Consider the* $(7,3)$ *binary linear code whose 8 codewords are*

$$\mathcal{C} = \{0000000, 1011100, 0101110, 1110010, 0010111, 1001011, 0111001, 1100101\}.$$

Since all non-zero codewords are circular shifts of a single codeword, it is a cyclic code. Its cyclic structure can also be exposed by choosing the generator matrix to be

$$\underline{G} = \begin{bmatrix} 1 & 0 & 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 & 1 \end{bmatrix}$$

and the parity-check matrix to be

$$\underline{H} = \begin{bmatrix} 1 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 1 \end{bmatrix}.$$

The benefits of cyclic codes follow largely from a close conection to polynomial arithmetic. This allows the encoding and decoding of cyclic codes to make use of efficient algorithms and hardware for polynomial arithmetic. Consider an (n,k) cyclic code and assume each codeword vector $\underline{c} = (c_0, c_1, \ldots, c_{n-1})$ and message vector $\underline{m} = (m_0, m_1, \ldots, m_{k-1})$ is associated with a polynomial

$$c(x) = \sum_{i=0}^{n-1} c_i x^i \qquad m(x) = \sum_{i=0}^{k-1} m_i x^i.$$

It turns out that every (n,k) cyclic code is uniquely defined by its generator polynomial $g(x)$, which allows all codewords $c(x) = m(x)g(x)$ to be generated by multiplication with some message polynomial $m(x)$. The generator polynomial for the code in the previous example is $g(x) = 1 + x^2 + x^3 + x^4$.

Before proceeding any further, we must define a few mathematical terms asociated with polynomial arithmetic. The **degree** of a polynomial $c(x)$, denoted $\deg(c(x))$, is the maximum power of x that appears in the polynomial. For the example $g(x)$, we have $\deg(g(x)) = 4$. In fact, for all (n,k) cyclic codes, we have $\deg(c(x)) \leq n-1$, $\deg(m(x)) \leq k-1$, and $\deg(g(x)) = n-k$.

For a polynomial $a(x)$ and a divisor $d(x)$, the **remainder** $r(x)$ and **quotient** $q(x)$ are uniqely defined by $\deg(r(x)) < \deg(d(x))$ and

$$a(x) = q(x)d(x) + r(x).$$

In discrete mathematicss, the **modulo polynomial** operation

$$r(x) = a(x) \bmod d(x)$$

is used to compactly represent the remainder $r(x)$ of division by $d(x)$.

Proposition 6.5.3. *For an (n,k) cyclic code defined by its generator polynomial $g(x)$, the parity-check polynomial $h(x)$ is defined uniquely by $g(x)h(x) = x^n - 1$ and satisfies*

$$h(x)c(x) \bmod x^n - 1 = 0,$$

for all codeword polynomials $c(x)$.

Proof. First, we observe that $h(x)$ can be computed by dividing $x^n - 1$ by $g(x)$. Therefore, one can prove this statement by observing that $c(x) = m(x)g(x)$ for some $m(x)$ and therefore

$$h(x)c(x) = m(x)g(x)h(x) = m(x)(x^n - 1).$$

Since $x^n - 1$ divides $h(x)c(x)$ without remainder, the result follows. $\square$

The parity-check polynomial for the code in the previous example is $h(x) = 1 + x^2 + x^3$.

In general, the generator matrix of an (n, k) cyclic code with generator $g(x) = g_0 + g_1 x + g_2 x^2 + \cdots g_{n-k} x^{n-k}$ can be written in the form

$$\underline{G} = \begin{bmatrix} g_0 & g_1 & \cdots & g_{n-k} & 0 & 0 & 0 \\ 0 & g_0 & g_1 & \cdots & g_{n-k} & 0 & 0 \\ \vdots & \vdots & \ddots & \ddots & \ddots & \ddots & 0 \\ 0 & 0 & 0 & g_0 & \cdots & g_{n-k-1} & g_{n-k} \end{bmatrix}.$$

Likewise, the parity-check polynomial $h(x) = h_0 + h_1 x + h_2 x^2 + \cdots h_k x^k$ gives rise to a parity-check matrix of the form

$$\underline{H} = \begin{bmatrix} h_k & h_{k-1} & \cdots & h_0 & 0 & 0 & 0 \\ 0 & h_k & h_{k-1} & \cdots & h_0 & 0 & 0 \\ \vdots & \vdots & \ddots & \ddots & \ddots & \ddots & 0 \\ 0 & 0 & 0 & h_k & \cdots & h_1 & h_0 \end{bmatrix}.$$

The encoder mapping $\underline{c} = \underline{m}\,\underline{G}$ is actually identical to the encoder mapping $c(x) = m(x)g(x)$. If the rows and columns of $\underline{G}$ are numbered from zero, then this can be seen by associating the ith row of $\underline{G}$ with x^i and the jth column of $\underline{G}$ with x^j. Observing that $[\underline{G}]_{ij} = g_{j-i}$ for $i \leq j \leq n-1$ and 0 otherwise, we find that

$$\begin{aligned} c(x) &= \sum_{j=0}^{n-1} x^j c_j = \sum_{j=0}^{n-1} x^j \sum_{i=0}^{k-1} m_i [\underline{G}]_{ij} = \sum_{j=0}^{n-1} x^j \sum_{i=0}^{k-1} m_i g_{j-i} \\ &= \sum_{i=0}^{k-1} m_i x^i \sum_{j=i}^{n-1} x^{j-i} g_{j-i} = \sum_{i=0}^{k-1} m_i x^i g(x) = m(x)g(x). \end{aligned}$$

Example 6.5.4. *The binary Hamming codes are a family of (n, k) cyclic codes, with $n = 2^r - 1$ and $k = n - r$, that can correct all single errors. The generator polynomials for $k = 3, 4, 5$ are given by*

$$\begin{aligned} (7,4) &\qquad g(x) = 1 + x^2 + x^3 \\ (15,11) &\qquad g(x) = 1 + x^3 + x^4 \\ (31,26) &\qquad g(x) = 1 + x^3 + x^5. \end{aligned}$$

6.5.2 Systematic Encoding

For cyclic codes, the basic encoding process $c(x) = m(x)g(x)$ does not provide systematic encoding. One can overcome this limitation by observing that $c(x) \bmod g(x) = 0$ for all codewords. Using this equation instead, one can choose the codeword have the format

$$c(x) = p(x) + x^{n-k}m(x),$$

where the message $m(x)$ appears as the last k bits in the codeword and the parity polynomial $p(x) = p_0 + p_1 x + \cdots p_{n-k-1}x^{n-k-1}$ has degree $\deg(p(x)) = n-k-1$. Applying the modulo $g(x)$ operation to both sides of this equation shows that

$$0 = c(x) \bmod g(x) = p(x) \bmod g(x) + x^{n-k}m(x) \bmod g(x).$$

Solving for $p(x)$ gives

$$p(x) = -x^{n-k}m(x) \bmod g(x),$$

where $p(x) = p(x) \bmod g(x)$ follows from the fact that $\deg(p(x)) < \deg(g(x))$. For cyclic codes, this process defines the standard **systematic encoder** used in practice. It turns out that that $p(x)$ can be calculated efficiently using a shift register circuit. So, systematic encoders for cyclic codes are easy to implement in practice.

Example 6.5.5. *Consider our example $(7,3)$ code with $g(x) = 1+x^2+x^3+x^4$. The message $\underline{m} = [0\ 1\ 1]$ is associated with the message polynomial $m(x) = x + x^2$ and the systematic encoding equation says that*

$$p(x) = -x^4 m(x) \bmod g(x) = -x^5 - x^6 \bmod 1 + x^2 + x^3 + x^4.$$

To compute $p(x)$, we proceed by using polynomial long division

$$
\begin{array}{rllllll}
 & & & & & -x^2 & +1\\ \cline{2-7}
x^4+x^3+x^2+1 \big) & -x^6 & -x^5 & & & & \\
 & x^6 & +x^5 & +x^4 & & +x^2 & \\ \cline{2-6}
 & & & x^4 & & +x^2 & \\
 & & & -x^4 & -x^3 & -x^2 & -1\\ \cline{4-7}
 & & & & -x^3 & & -1
\end{array}
$$

This implies that $p(x) = x^3 + 1$ *and* $c(x) = 1 + x^3 + x^5 + x^6$. *One can verify that this is a codeword by observing that* $(1 + x^2)(1 + x^2 + x^3 + x^4) \bmod 2 = 1 + x^3 + x^5 + x^6$.

6.5.3 Cyclic Redundancy Check

A **cyclic redundancy check** (CRC) code calculates its check bits using a process very similar to the systematic encoding of cyclic codes. The main difference is that the message polynomial $m(x)$ is not constrained to be k bits in length and it is not multiplied by x^{n-k} before the modulo operation. Let $g(x)$ be the CRC polynomial and $m(x)$ be the message polynomial, then the CRC encoder generates the check polynomial

$$p(x) = m(x) \mod g(x).$$

Since $\deg(p(x)) < \deg(g(x))$, the resulting check sequence is represented by exactly $\deg(g(x)) - 1$ bits. Many of the CRCs defined in standards (or commonly used in practice) have subtle variations such as

- extra bits are appended to the start or end of the message,
- the shift register state is initialized to a non-zero value,
- the message sequence is XOR'd with a known sequence before division,
- and the bit (or byte) ordering may differ between standards.

In most specifications, the CRC polynomial is not described as a polynomial but instead given as the hexadecimal value of the binary representation of $g(x)$. Moreover, the topmost bit is not included in this value because the length is know and it must a 1.

Example 6.5.6. *The standard unix CRC32 polynomial is often listed in hexadecimal as* `04C11DB7`. *Writing this in binary gives*

$$0000\ 0100\ 1100\ 0001\ 0001\ 1101\ 1011\ 0111.$$

Counting from the right (i.e., the LSB), we find ones in the positions

$$0, 1, 2, 4, 5, 7, 8, 10, 11, 12, 16, 22, 23, 26.$$

Therefore, it follows that

$$g(x) = x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^{8} + x^{7} + x^{5} + x^{4} + x^{2} + x + 1.$$

Notice the x^{32} is added based on the known length of the CRC and is not described by the hexadecimal value.

The CRC32 algorithm prepends the data sequence by 32 ones, performs the modulo, and then inverts the resulting parity sequence. The ASCII sequence “123456789” has the hexadecimal representation `31 32 33 34 35 36 37 38 39` and is commonly used to test CRC implementations. For this sequence, the correct CRC result, in hexadecimal, is `CBF43926`.

Chapter 7

Waveform Communication

In Chapter 3, we introduced digital communication using a discrete-time model that allowed us to neglect the details of the more realistic continuous-time model. The goal of this chapter is to delve into the details of waveform-based communication and see that the discrete-time model can be derived from the continuous-time model under some conditions. The **waveform channel** defines the probabilistic mapping between the deterministic channel input waveform $s(t)$ and the random-process $R(t)$ observed at the output of the channel. The basic model is that $R(t) = s(t) + N(t)$, where $N(t)$ is Gaussian white-noise random process with autocorrelation function $R_N(\tau) = \delta(\tau) N_0/2$.

7.1 A Single Digital Symbol

Suppose that one would like to send a single digital symbol taking M different values to a receiver through a waveform channel. Then, one can associate a waveform $s_m(t)$ with each symbol $m = 1, \ldots, M$ and transmit the waveform assigned with the desired message. For mathematical convenience, we assume that the energy integral exists and is finite for each waveform.

The demodulation process is based on mathematical operation, known as an inner product, that maps any two signals to a complex number. The **inner product** $\langle s(t), r(t) \rangle$ between the signals $s(t)$ and $r(t)$ is defined by

$$\langle s(t), r^*(t) \rangle = \int_{-\infty}^{\infty} s(t) r^*(t) dt,$$

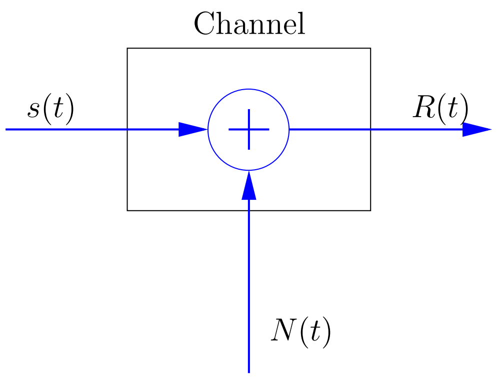

Figure 7.1: Block diagram of a basic waveform channel for input signal $s(t)$, noise process $N(t)$, and received signal $R(t)$.

where $r^*(t)$ is the complex conjugate of $r(t)$. This is the same as the cross-correlation between $r(t)$ and $s(t)$ evaluated at lag-time 0. Mathematically, we are treating the set of all finite-energy signals as a **vector space** and using this inner product to define distances and angles in this space. The energy (or length squared) of a signal $s(t)$ is given by

$$\int_{-\infty}^{\infty} |s(t)|^2 \, dt = \langle s(t), s(t) \rangle.$$

Two signals $s_1(t), s_2(t)$ are said to be **orthogonal** if $\langle s_1(t), s_2(t) \rangle = 0$. A signal $s(t)$ is said to be **normalized** if $\langle s(t), s(t) \rangle = 1$. A set of signals is said to be **orthonormal** if every signal in the set is orthogonal to every other signal in the set and all signals in the set are normalized. The set of all linear combinations of signal waveforms is called the **signal space**.

7.1.1 Orthogonal Waveforms

The simplest scenario occurs when the set of signal waveforms $\{s_1(t), \ldots, s_M(t)\}$ are orthogonal and each has energy E_s. In this case, one can demodulate the received signal $R(t)$ by computing

$$R_j = \langle R(t), s_j(t) \rangle,$$

for each $j = 1, \ldots, M$. This is the same as correlating the received signal with each of the transmitted waveforms. Due to the noise process, the resulting values $R_1, \ldots, R_M$ are Gaussian random variables. Their expected values depend on which waveform was actually transmitted. If $s_m(t)$ was transmitted, then $R(t) = s_m(t) + N(t)$, $E[R(t)] = s_m(t)$, and we have

$$\begin{aligned} E\left[R_j\right] &= E\left[\int_{-\infty}^{\infty} R(t) s_j^*(t) dt\right] \\ &= \int_{-\infty}^{\infty} E\left[R(t)\right] s_j^*(t) dt \\ &= \int_{-\infty}^{\infty} s_m(t) s_j^*(t) dt \\ &= E_s \delta_{m,j}. \end{aligned}$$

The random variables $R_1, \ldots, R_M$ also have some other nice properties. They are uncorrelated because the the waveforms are orthogonal and they have variance $\frac{1}{2} E_s N_0$. Proving this statement will be left as a homework exercise.

This transforms the waveform detection problem into a detection problem for a length-M vector of Gaussian random variables. If $s_m(t)$ was transmitted, then the mean of the random vector is scaled unit vector with E_s in the mth position. Let $(r_1, \ldots, r_M)$ be the realization of the random vector $(R_1, \ldots, R_M)$. Then, based on the techniques from Chapter 3, we know that the maximum-likelihood decision rule chooses the symbol whose unit vector is closest to $(r_1, \ldots, r_M)$ in Euclidean distance. This implies that we should choose the signal $s_{\hat{m}}(t)$ if

$$\sum_{i=1}^{M} (r_i - \delta_{\hat{m},i} E_s)^2 \leq \sum_{i=1}^{M} (r_i - \delta_{m,i} E_s)^2 \quad \text{for } m = 1, \ldots, M.$$

Expanding both sides of this equation shows that this is equivalent to

$$r_{\hat{m}} \geq r_m \quad \text{for } m = 1, \ldots, M.$$

This whole process is known as a correlation-based receiver.

Example 7.1.1 (Frequency-Shift Keying). *For a fixed time-interval T, consider the collection of waveforms given by*

$$s_m(t) = \frac{1}{\sqrt{T}} e^{2\pi i \frac{m}{T} t} \text{rect}\left(\frac{t}{T}\right).$$

for $m = 1, \ldots, M$. This set of waveforms is known as M-ary frequency-shift keying (or M-FSK). We wish to show that these waveforms are orthonormal.

To prove that they are orthogonal, we consider the inner product of $s_m(t)$ and $s_n(t)$ when $m \neq n$,

$$\begin{aligned}\langle s_m(t), s_n(t)\rangle &= \int_{\mathbb{R}} s_m(t)s_n^*(t)dt = \frac{1}{T}\int_{-\frac{T}{2}}^{\frac{T}{2}} e^{2\pi i \frac{m}{T}t}e^{-2\pi i \frac{n}{T}t}dt \\ &= \frac{1}{T}\int_{-\frac{T}{2}}^{\frac{T}{2}} e^{2\pi i \frac{(m-n)}{T}t}dt = 0.\end{aligned}$$

Next, we show that these basis elements have unit energy,

$$\begin{aligned}\|s_m(t)\|^2 &= \int_{\mathbb{R}} s_m(t)s_m^*(t)dt = \frac{1}{T}\int_{-\frac{T}{2}}^{\frac{T}{2}} e^{2\pi i \frac{m}{T}t}e^{-2\pi i \frac{m}{T}t}dt \\ &= \frac{1}{T}\int_{-\frac{T}{2}}^{\frac{T}{2}} dt = 1.\end{aligned}$$

7.1.2 General Waveforms

When the signal waveforms are not orthonormal, the detection problem is solved most easily by constructing a set of orthonormal basis vectors for the signal space. In general, this can be accomplished by applying the Gram-Schmidt orthogonalization process to the signal waveforms. Rather than focusing on this process, we assume that $\{\phi_1(t), \ldots, \phi_N(t)\}$ is an orthonormal set of waveforms such that the coefficients $a_{m,k}$ allow us to write

$$s_m(t) = \sum_{k=1}^{N} a_{m,k}\phi_k(t).$$

This implies that the orthonormal set spans the signal space.

In this case, the received signal $R(t)$ can be demodulated by computing

$$R_j = \langle R(t), \phi_j(t)\rangle,$$

for each $j = 1, \ldots, N$. This is the same as correlating the received signal with each of the basis waveforms. Again, the noise process implies that $R_1, \ldots, R_N$ are Gaussian random variables whose means are determined by which waveform was actually transmitted. For $j = 1, \ldots, N$, the noise component of R_j

is independent of the transmitted signal, and is given by

$$N_j = \langle N(t), \phi_j(t)\rangle = \int_{-\infty}^{\infty} N(t)\phi_j^*(t)dt.$$

Therefore, $R_j = E[R_j] + N_j$ and, assuming $s_m(t)$ was transmitted, the mean is given by

$$\begin{aligned} E\left[R_j\right] &= E\left[\int_{-\infty}^{\infty} R(t)\phi_j^*(t)dt\right] \\ &= \int_{-\infty}^{\infty} E\left[R(t)\right]\phi_j^*(t)dt \\ &= \int_{-\infty}^{\infty} s_m(t)\phi_j^*(t)dt \\ &= \int_{-\infty}^{\infty} \left(\sum_{k=1}^{N} a_{m,k}\phi_k(t)\right)\phi_j^*(t)dt \\ &= \sum_{k=1}^{N} a_{m,k}\delta_{k,j}dt \\ &= a_{m,j}. \end{aligned}$$

Again, we have transformed the waveform detection problem into a detection problem for a vector of Gaussian random variables. In this case, however, there are M different mean vectors of dimension N. Let $(r_1, \ldots, r_N)$ be the realization of the random vector $(R_1, \ldots, R_N)$. Based on the techniques from Chapter 3, the maximum-likelihood decision rule chooses the symbol $\hat{m}$ whose coordinate vector $(a_{\hat{m},1}, \ldots .a_{\hat{m},N})$ is closest to $(r_1, \ldots, r_N)$ in Euclidean distance. This implies that we should choose the signal $s_{\hat{m}}(t)$ if

$$\sum_{i=1}^{N}(r_i - a_{\hat{m},i})^2 \le \sum_{i=1}^{N}(r_i - a_{m,i})^2 \quad \text{for } m = 1, \ldots, M.$$

This approach can be seen as first projecting the infinite-dimensional received waveform onto the finite-dimensional signal-subspace and then using the optimal detector assuming only the projection was observed. This overall approach is optimal only if the projection is a **sufficient statistic** for maximum-likelihood decision problem. While this is indeed true, we disregard this detail for now.

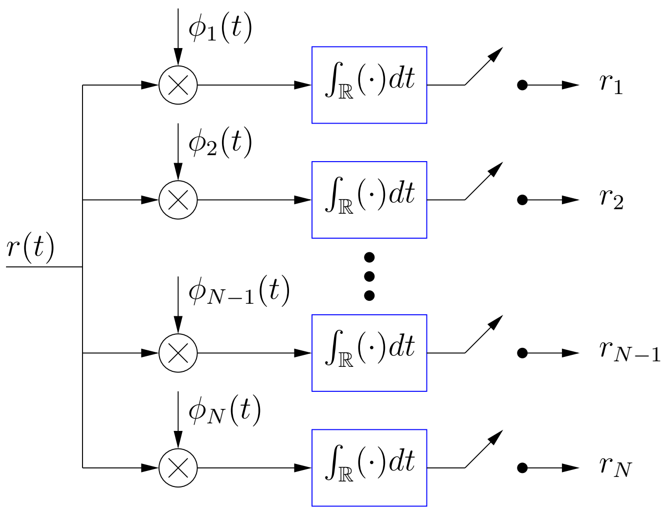

Figure 7.2: For a particular realization $r(t)$ of the received signal, the projection $(r_1, \ldots, r_N)$ onto the signal-space basis vectors can be computed by a bank of correlators.

7.1.3 The Matched Filter

Let $s(t)$ be a finite-energy signal and $R(t) = s(t) + N(t)$ be the received waveform. Suppose that the receiver computes $R = \langle R(t), z(t) \rangle$ with an arbitrary finite-energy signal $z(t)$. One's ability to detect $s(t)$ correctly is largely dependent on the SNR of R. Since the noise is zero mean, we have

$$SNR \triangleq \frac{|E[R]|^2}{\operatorname{Var}(R)} = \frac{\left| \int_{-\infty}^{\infty} s(t) z^*(t) dt \right|^2}{\frac{N_0}{2} \int_{-\infty}^{\infty} |z(t)|^2}.$$

Rewriting this in inner product notation shows that

$$SNR = \frac{2}{N_0} \frac{|\langle s(t), z(t) \rangle|^2}{\langle z(t), z(t) \rangle} = \frac{2}{N_0} \left| \left\langle s(t), \frac{z(t)}{\|z(t)\|} \right\rangle \right|^2.$$

Our intuition about vectors tells us that we can maximize this inner product by choosing the unit vector $z(t)/\|z(t)\|$ to point in the same direction as $s(t)$. Therefore, choosing $z(t) = c\, s(t)$, for $c \neq 0$, maximizes this function and gives

$$SNR = \frac{\left| c \int_{-\infty}^{\infty} s(t) s^*(t) dt \right|^2}{\frac{N_0}{2} \int_{-\infty}^{\infty} |c\, s(t)|^2} = \frac{c^2 E_s^2}{\frac{N_0}{2} c^2 E_s} = \frac{2E_s}{N_0},$$

where E_s is the energy in the waveform $s(t)$. This is one of the reasons that receivers based on the inner product (i.e., cross-correlation) can be optimal.

Using a bank of correlators requires timing information to zero each integrator at the correct time. In many cases, it is more convenient to build a filter $h(t)$ matched to the transmitted waveform and sample this filter as needed. The **matched filter** $h(t) = s^*(-t)$ is the time-reversed complex-conjugate of the transmitted waveform. Then, $y(t) = h(t) * R(t)$ can be sampled at time 0 to get

$$\begin{aligned} y(0) &= \int_{-\infty}^{\infty} R(\tau)h(t-\tau)d\tau \bigg|_{t=0} \\ &= \int_{-\infty}^{\infty} R(\tau)s^*(\tau - t)d\tau \bigg|_{t=0} \\ &= \int_{-\infty}^{\infty} R(\tau)s^*(\tau)d\tau \\ &= \langle R(t), s(t) \rangle. \end{aligned}$$

This shows that correlating against $s(t)$ is mathematically identical to filtering by $h(t) = s^*(-t)$ followed by sampling.

7.2 Time-Shift Waveforms

In practical communication systems, a succession of symbols is transmitted to the destination. Not only can waveforms interfere with one another in signal space, they can also disrupt signal quality across time. Suppose that a different symbol is sent every T seconds using time shifts of a basic pulse waveform $p(t)$. The transmitted signal, accounting for the different values of $\{x_n\}$, is equal to

$$s(t) = \sum_{n=-\infty}^{\infty} x_n p(t - nT).$$

Note that in this case, the available waveforms are simply translated versions of one another. Ideally, we want the collection $\{p(t - nT)\}$ to be orthonormal. This would greatly simplify system implementation and decision making at the receiver. However, we cannot use standard techniques such as the Gram-Schmidt procedure to construct a set of orthogonal waveforms, because the elements of $\{p(t-nT)\}$ are constrained to be translated version of one another. If an orthonormal set is needed, it is typically chosen by design.

7.2.1 Demodulation

As before, the received signal is the random process $R(t) = s(t) + N(t)$ and $N(t)$ is zero-mean Gaussian white-noise process. One can detect the symbol x_m by correlating the received signal with the waveform $p(t-mT)$. This gives

$$\begin{aligned}
R_m &= \langle R(t), p(t-mT)\rangle \\
&= \int_{-\infty}^{\infty} R(t)p^*(t-mT)dt \\
&= \int_{-\infty}^{\infty} N(t)p^*(t-mT)dt + \sum_{n=-\infty}^{\infty} x_n \int_{-\infty}^{\infty} p(t-nT)p^*(t-mT)dt \\
&= \int_{-\infty}^{\infty} N(t)p^*(t-mT)dt + \sum_{n=-\infty}^{\infty} x_n v\left((m-n)T\right) \\
&= \underbrace{Z_m}_{\text{noise}} + \underbrace{x_m v(0)}_{\text{signal}} + \sum_{n=-\infty, n\neq m}^{\infty} \underbrace{x_n v\left((m-n)T\right)}_{\text{intersymbol interference}},
\end{aligned}$$

where $v(\tau)$ is the autocorrelation function of $p(t)$ and

$$Z_m = \int_{-\infty}^{\infty} N(t)p^*(t-mT)dt.$$

The x_m term in the R_m expansion contains the desired symbol. The sum of the remaining x_n terms is called *intersymbol interference (ISI)*; it contains the contributions from all the other time-shifted waveforms. To retrieve the information sequence unambiguously, we wish to have $R_m = x_m + Z_m$, irrespective of the values in the sequence $\{x_n\}$. This will be achieved provided that

$$v(nT) = \begin{cases} 1, & n = 0 \\ 0, & \text{otherwise.} \end{cases} \tag{7.1}$$

Notice also that the autocorrelation of the noise sequence is given by

$$\begin{aligned}
E[Z_i Z_j^*] &= E\left[\left(\int_{-\infty}^{\infty} N(t)p^*(t-iT)dt\right)\left(\int_{-\infty}^{\infty} N^*(u)p(u-jT)du\right)\right] \\
&= \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} E[N(t)N^*(u)]p^*(t-iT)p(u-jT)dudt \\
&= \frac{N_0}{2}\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} \delta(t-u)p^*(t-iT)p(u-jT)dudt \\
&= \frac{N_0}{2}\int_{-\infty}^{\infty} p^*(t-iT)p(t-jT)dt \\
&= \frac{N_0}{2}v((i-j)T).
\end{aligned}$$

If (7.1) holds, then it follows that the noise $\{Z_m\}$ is a sequence of independent and identically distributed Gaussian random variables.

Example 7.2.1. *Therefore, if the condition in (7.1) holds, then we find that*

$$R_m = x_m + Z_m,$$

and we have recovered exactly the discrete-time model of communication discussed in Chapter 3.

Alternatively, one can use a matched filter $h(t) = p^*(-t)$ and define the filter output to be $Y(t) = R(t) * h(t)$. In this case, sampling $Y(t)$ gives

$$\begin{aligned}
Y(mT) &= \int_{-\infty}^{\infty} R(\tau)h(mT-\tau)d\tau \\
&= \int_{-\infty}^{\infty} R(\tau)p^*(\tau-mT)d\tau \\
&= R_m.
\end{aligned}$$

This shows that sampling the matched filter is equivalent to the correlation-based receiver.

7.2.2 The Nyquist Criterion

To understand how the condition (7.1) impacts our choice of a functions $p(t)$ and $h(t)$, we let $g(t) = p(t) * h(t)$ and use the frequency representation of $g(t)$.

Looking at the inverse Fourier transform of $\hat{g}(f)$, we get

$$\begin{aligned} g(\tau) = \mathcal{F}^{-1}\left[\hat{g}(f)\right] &= \int_{\mathbb{R}} \hat{g}(f) e^{2\pi i f \tau} df \\ &= \sum_{m=-\infty}^{\infty} \int_{-\frac{F}{2}}^{\frac{F}{2}} \hat{g}(f - Fm) e^{2\pi i (f - mF)\tau} df \end{aligned}$$

where, for reasons that will soon be obvious, we have judiciously selected $F = \frac{1}{T}$. The value of $g(\tau)$ at the sample points $\{\tau = nT : n \in \mathbb{Z}\}$ can then be expressed as

$$\begin{aligned} g(-nT) &= \sum_{m=-\infty}^{\infty} \int_{-\frac{F}{2}}^{\frac{F}{2}} \hat{g}\left(f - \frac{m}{T}\right) e^{-2\pi i \left(f - \frac{m}{T}\right) nT} df \\ &= \int_{-\frac{F}{2}}^{\frac{F}{2}} \left(\sum_{m=-\infty}^{\infty} \hat{g}\left(f - \frac{m}{T}\right) \right) e^{-2\pi i n T f} df \qquad (7.2) \\ &= \frac{1}{F} \int_{-\frac{F}{2}}^{\frac{F}{2}} \hat{z}(f) e^{-2\pi i \frac{n}{F} f} df, \end{aligned}$$

where we have defined

$$\hat{z}(f) = F \sum_{m=-\infty}^{\infty} \hat{g}\left(f - \frac{m}{T}\right) \mathrm{rect}\left(\frac{f}{F}\right).$$

Notice the similarity between (7.2) and the Fourier series representation of a time-limited function. Specifically, $\{g(nT) : n \in \mathbb{Z}\}$ can be viewed as the Fourier series coefficients of the frequency-limited function $\hat{z}(f)$. Under condition (7.1), and using the reconstruction formula for Fourier series, we get

$$\hat{z}(f) = \sum_{n=-\infty}^{\infty} g(-nT) e^{2\pi i \frac{n}{F} f} \mathrm{rect}\left(\frac{f}{F}\right) = \mathrm{rect}\left(\frac{f}{F}\right) \qquad (7.3)$$

because $g(nT) = 0$ whenever $n \neq 0$. Thus, the system exhibits no intersymbol interference if and only if (7.3) holds.

Equivalently, condition (7.1) is satisfied whenever

$$\sum_{m=-\infty}^{\infty} \hat{g}\left(f - \frac{m}{T}\right) = T. \qquad (7.4)$$

We formalize this key result, known as the *Nyquist pulse-shaping criterion*, in the theorem below.

Theorem 7.2.2 (Nyquist). *Let $g(\tau)$ and $\hat{g}(f)$ be square integrable functions that are Fourier transforms of each other. Furthermore, assume that the function*

$$\hat{z}(f) = F \sum_{m=-\infty}^{\infty} \hat{g}\left(f - \frac{m}{T}\right) \text{rect}(fT)$$

has finite energy. Then, a necessary and sufficient condition for

$$g(nT) = \begin{cases} 1, & n = 0 \\ 0, & \textit{otherwise} \end{cases}$$

is that the following equality holds for all values of $f \in \mathbb{R}$,

$$\sum_{m=-\infty}^{\infty} \hat{g}\left(f - \frac{m}{T}\right) = T. \tag{7.5}$$

It is important to note that the Nyquist criterion is applied not the transmitted pulse $p(t)$, but to the overall response $g(t)$. But, optimum detection in white-noise requires that $h(t)$ is matched to $p(t)$. Therefore, the best solution is to *split* the $g(t)$ filter between the transmitter and receiver. If $\hat{g}(f)$ is real and positive, then we can simply choose

$$\hat{p}(f) = \hat{h}(f) = \sqrt{\hat{g}(f)}.$$

This is known as a **square-root Nyquist pulse**. In this case, $\hat{p}(f)$ is real and $p(t) = p^*(-t)$ and the matched filter is $h(t) = p^*(-t) = p(t)$. Since $\hat{g}(f)$ satisfies the Nyquist criterion, this allows one to achieve optimum detection without ISI.

Example 7.2.3. *One of the simplest possible choices for waveforms $p(t)$ and $h(t)$ is*

$$p(t) = h(t) = \frac{1}{\sqrt{T}} \text{rect}\left(\frac{t}{T}\right).$$

In this case, we get

$$\begin{aligned} g(\tau) = \langle p(t), h(t-\tau) \rangle &= \frac{1}{T} \int_{\mathbb{R}} \text{rect}\left(\frac{t}{T}\right) \text{rect}\left(\frac{t-\tau}{T}\right) dt \\ &= \frac{1}{T} \int_{\mathbb{R}} \text{rect}\left(\frac{t}{T}\right) \text{rect}\left(\frac{\tau - t}{T}\right) dt = \frac{1}{T} \text{rect}\left(\frac{t}{T}\right) * \text{rect}\left(\frac{t}{T}\right). \end{aligned}$$

Obviously, this selection leads to the desired property with $g(0) = 1$, *and* $g(nT) = 0$ *for any non-zero integer* n. *One of the design issues with the rectangular pulse is that its bandwidth is infinite and decays slowly. This becomes a problem in most practical systems where spectral bandwidth comes at a premium. Notice also that this is the square-root Nyquist pulse associated with a triangular* $g(t)$.

Example 7.2.4. *Consider the pulse-shaping criterion applied to*

$$p(t) = h(t) = \frac{1}{\sqrt{T}} \operatorname{sinc}\left(\frac{t}{T}\right).$$

We wish to show that this choice of waveforms satisfies the Nyquist criterion and leads to a set of orthogonal time-shift waveforms.

First, we find an expression for $g(\tau)$,

$$\begin{aligned} g(\tau) = \langle p(t), h(t-\tau) \rangle &= \frac{1}{T} \int_{\mathbb{R}} \operatorname{sinc}\left(\frac{t}{T}\right) \operatorname{sinc}\left(\frac{t-\tau}{T}\right) dt \\ &= \frac{1}{T} \int_{\mathbb{R}} \operatorname{sinc}\left(\frac{t}{T}\right) \operatorname{sinc}\left(\frac{\tau-t}{T}\right) dt = \frac{1}{T} \operatorname{sinc}\left(\frac{t}{T}\right) * \operatorname{sinc}\left(\frac{t}{T}\right). \end{aligned}$$

In the frequency domain, we have

$$\begin{aligned} \hat{g}(f) = \mathcal{F}[g(\tau)] &= \frac{1}{T} \mathcal{F}\left[\operatorname{sinc}\left(\frac{t}{T}\right)\right] \mathcal{F}\left[\operatorname{sinc}\left(\frac{t}{T}\right)\right] \\ &= T\operatorname{rect}(Tf)\operatorname{rect}(Tf) = T\operatorname{rect}(Tf). \end{aligned}$$

We can therefore verify condition (7.5) *as*

$$\sum_{m=-\infty}^{\infty} \hat{g}\left(f - \frac{m}{T}\right) = \sum_{m=-\infty}^{\infty} T\operatorname{rect}(Tf - m) = T.$$

That is, the conditions of Theorem 7.2.2 hold and, consequently, $g(0) = 1$ *and* $g(nT) = 0$ *for any non-zero integer. One of the positive attributes of the* $\operatorname{sinc}(\cdot)$ *waveform is that it is bandwidth-limited. However, this pulse is not time-limited and it decays quite slowly in the time-domain. It is therefore somewhat impractical, as using* $\operatorname{sinc}(\cdot)$ *waveforms without truncation would entail infinite delay at the destination. Notice that this is the square-root Nyquist pulse associated with* $\hat{g}(f) = T\operatorname{rect}(Tf)$ *and that the square root has no effect.*

Although, there are many choices for $\hat{g}(f)$ that satisfy Theorem 7.2.2, we are primarily interested in waveforms that are approximately both bandwidth-limited and time-limited. The *Nyquist bandwidth* associated with a signal

$$s(t) = \sum_{n=-\infty}^{\infty} x_n p(t - kT)$$

is defined by $\frac{1}{2T}$; it represents the smallest possible bandwidth for which it is possible to prevent intersymbol interference. The spectral bandwidth of $\hat{g}(f)$ is the smallest possible value of W such that $\hat{g}(f) = 0$ for $|f| > W$.

Example 7.2.5. *The* ***raised-cosine Nyquist pulse*** *introduces some bandwidth expansion in order to achieve a more rapid pulse decay in the time-domain. It is the basis for the one of the most popular pulse shaping filters used in digital communications. In the frequency domain, it has the representation*

$$\hat{g}(f) = \begin{cases} T & \text{if } |f| \leq \frac{1-\beta}{2T} \\ T\cos^2\left(\frac{\pi T}{2\beta}\left[|f| - \frac{1-\beta}{2T}\right]\right) & \text{if } \frac{1-\beta}{2T} < |f| \leq \frac{1+\beta}{2T} \\ 0 & \text{otherwise,} \end{cases}$$

where β is the roll-off factor that measures the excess bandwidth (beyond the Nyquist minimum) used by the filter.

To compute the correct pulse-shaping filter, we must take the square-root in the frequency domain and the inverse Fourier transform. This results in the ***root raised-cosine pulse*** *given by*

$$p(t) = \begin{cases} 1 - \beta + 4\dfrac{\beta}{\pi} & \text{if } t = 0 \\ \dfrac{\beta}{\sqrt{2}}\left[\left(1 + \dfrac{2}{\pi}\right)\sin\left(\dfrac{\pi}{4\beta}\right) + \left(1 - \dfrac{2}{\pi}\right)\cos\left(\dfrac{\pi}{4\beta}\right)\right] & \text{if } t = \pm\dfrac{T_s}{4\beta} \\ \dfrac{\sin\left[\pi\dfrac{t}{T_s}(1-\beta)\right] + 4\beta\dfrac{t}{T_s}\cos\left[\pi\dfrac{t}{T_s}(1+\beta)\right]}{\pi\dfrac{t}{T_s}\left[1 - \left(4\beta\dfrac{t}{T_s}\right)^2\right]} & \text{otherwise.} \end{cases}$$

In this case, the slow $1/t$ decay rate of the sinc function has been improved to a decay rate of $1/t^2$.

7.2.3 Linear Time-Invariant Channels

Now, we can also generalize the system model to include an LTI channel response $h_c(t)$ and a LTI receive filter $h_r(t)$. Since we assume that the dominant source of noise is the *low-noise amplifier*, the noise is added after the channel filter but before the receive filter. This implies that the received signal is given by $R(t) = s(t) * h_c(t) + N(t)$ and the output of the receive filter is given by

$$\begin{aligned} Y(t) &= s(t) * h_c(t) * h_r(t) + N(t) * h_r(t) \\ &= \left(\sum_{n=-\infty}^{\infty} x_n p(t - nT) \right) * h_c(t) * h_r(t) + N(t) * h_r(t) \\ &= \sum_{n=-\infty}^{\infty} x_n g(t - nT) + \int_{-\infty}^{\infty} N(\tau) h_r(t - \tau), \end{aligned}$$

where $g(t) = p(t) * h_c(t) * h_r(t)$ and $y_s(t)$ is the deterministic portion of $Y(t)$.

Let $R_m = Y(mT)$ be the sampled output of the matched filter. Then, $R_m = y_s(mT) + Z_m$ with

$$y_s(mT) = x_m g(0) + \sum_{n=-\infty, n\neq m}^{\infty} x_n g\left((m-n)T\right)$$

and

$$Z_m = \int_{-\infty}^{\infty} N(\tau) h_r(mT - \tau).$$

If we choose $h_r(t)$ such that its autocorrelation function $v(\tau)$ satisfies (7.1), then the noise $\{Z_m\}$ is a sequence of independent and identically distributed Gaussian random variables.

Example 7.2.6. *If $g(nT) \neq 0$ for some $n \neq 0$, then we say that system has ISI. For example, consider the case where*

$$g(nT) = \begin{cases} 1 & \text{if } n = 0 \\ \frac{1}{4} & \text{if } n = 1 \\ 0 & \text{otherwise} \end{cases} .$$

This gives a discrete-time model of communication with ISI where

$$Y_m = x_m + \frac{1}{4} x_{m-1} + Z_m.$$

7.2.4 Passband Communication

Until now, our study of waveform communication has focused on **baseband signals**, which have the majority of their energy near DC. Signals which occupy a narrow frequency band well separated from DC are known as **passband signals**. All of wireless communications is based on passband signals because low-frequency electromagnetic waves require enormous antennas for efficient transmission. Typical frequency ranges for wireless communications start around 100 MHz and short-range communication is possible up until about 60 GHz.

The process of shifting a baseband signal in frequency to make it passband is know as mixing. Let $s(t)$ be a baseband signal with bandwidth W and define

$$x(t) = e^{2\pi j f_0 t} s(t).$$

From the modulation property of the Fourier transform, one sees that

$$\hat{x}(f) = \hat{s}(f - f_0).$$

This implies that $x(t)$ is a passband signal centered at frequency f_0 with bandwidth $2W$.

The resulting $x(t)$ must be complex because it does not have negative-frequency conjugate-symmetry. But, any signal represented by a single time-varying voltage on a single wire is necessarily real. Therefore, we must take do something before coupling $x(t)$ to the antenna. It turns out that the real part of $x(t)$ gives a physically realizable passband signal that works for our purposes. The result

$$\begin{aligned} \Re\left\{x(t)\right\} &= \frac{1}{2} e^{2\pi j f_0 t} s(t) + \frac{1}{2} e^{-2\pi j f_0 t} s^*(t) \\ &= \Re\left\{s(t)\right\} \cos(2\pi f_0 t) - \Im\left\{s(t)\right\} \sin(2\pi f_0 t) \end{aligned}$$

shows us how to mix a complex baseband signal $s(t)$ with real oscillators (i.e., for the sine and cosine) to generate the passband signal. This mixing process is known as **quadrature amplitude modulation**.

In wireless communication, the real passband signal is coupled to an antenna to generate radio waves that propagate through space. At the receiver, another antenna couples the received radio waves into the the time-varying voltage $r(t)$. Demodulation of the received waveform $r(t) = \Re\left\{x(t)\right\}$ is done using a similar process with

$$\begin{aligned} y(t) &= r(t)e^{-2\pi j f_0 t} \\ &= \frac{1}{2}\Re\left\{s(t)\right\}\left(\cos^2(2\pi f_0 t) - j\cos(2\pi f_0 t)\sin(2\pi f_0 t)\right) \\ &\quad + \frac{j}{2}\Im\left\{s(t)\right\}\left(\sin^2(2\pi f_0 t) - j\cos(2\pi f_0 t)\sin(2\pi f_0 t)\right) \\ &= \frac{1}{2}\Re\left\{s(t)\right\}\left(1 + \cos(4\pi f_0 t) - j\sin(4\pi f_0 t)\right) \\ &\quad + \frac{j}{2}\Im\left\{s(t)\right\}\left(1 - \cos(4\pi f_0 t) - j\sin(4\pi f_0 t)\right. . \end{aligned}$$

If $f_0 + W < 2f_0 - W$, then we can use an ideal low-pass filter $h(t)$ (with cutoff $f_0 + W$) to remove the *images* centered at $2f_0$ without affecting the signal. This gives

$$y(t) * h(t) = \frac{1}{2}s(t).$$

So this approach allows us to upconvert $s(t)$ to a passband signal and then downconvert it back to a bandpass signal.

Index

channel coding, 87
- binary erasure channel, 89
- binary linear code, 93
- binary symmetric channel, 89
- binary-input AWGN channel, 89
- coding gain, 89, 102
- cyclic code, 103
- cyclic redundancy check, 107
- discrete memoryless channel, 88
- generator matrix, 94
- Hamming code, 97
- Hamming distance, 90
- Hamming weight, 93
- modulo polynomial, 104
- parity generator, 95
- parity-check matrix, 95
- standard array, 98
- syndrome decoder, 98
- systematic encoder, 106
- systematic form, 95, 96

digital communication, 1, 33, 109
- additive white Gaussian noise, 34
- baseband signals, 123
- binary phase-shift keying, 35
- channel, 33
- demodulation, 33
- discrete-time channel model, 34
- inner product, 109
- matched filter, 115
- modulation, 33
- noise spectral density, 37
- optimal detection, 38
- orthogonal, 110
- passband signals, 123
- pulse-amplitude modulation, 35
- raised-cosine Nyquist pulse, 121
- root raised-cosine pulse, 121
- signal space, 110
- signal-to-noise ratio, 37
- square-root Nyquist pulse, 119
- symbol constellation, 37
- vector space, 110
- waveform channel, 109

Fourier analysis, 43
- aliasing, 51
- autocorrelation function, 48
- bandpass signal, 57
- bandwidth-limited, 50
- Dirac delta function, 46
- energy spectral density, 46, 48
- energy-type signal, 45
- Fourier series, 44
- Fourier transform, 45
- frequency domain, 43

Hilbert transform, 58
Nyquist rate, 54
power spectral density, 49
power-type signal, 49
sinc function, 45
spectral bandwidth, 49
square integrable, 45
step function, 57

hypothesis testing, 38
a posteriori probability, 40
a priori probability, 40
maximum a posteriori, 40
maximum likelihood, 41
multiple hypothesis testing, 40
null hypothesis, 39
observation probability, 40

information theory
AEP, 29
entropy, 12
typical set, 30

quantization, 65
analysis-synthesis, 85
Code Excited Linear Prediction, 85
companding, 73
conditional mean, 76
delta modulation, 75
differential quantization, 75
distortion, 65
dithering, 72
JPEG file interchange format, 86
linear prediction, 77
Lloyd-Max algorithm, 81
LMMSE estimate, 77
mean squared error, 68
MMSE estimate, 76
non-uniform quantizer, 79
quantization error, 67
signal-to-quantization-noise ratio, 69
uniform quantizer, 78
vector quantization, 82
waveform coders, 85

random process, 58
autocorrelation function, 59
continuous-time, 58
discrete-time, 26, 58
ergodic, 58, 59
Gaussian process, 63
Gaussian white-noise, 64
Markov chain, 26
mean, 58
power spectral density, 60
stationary, 26, 58
stochastic process, 58

source coding
code, 14
codeword, 14
compression code, 14
discrete memoryless source, 10
fixed-length code, 14
Huffman code, 21, 86
Lempel-Ziv algorithm, 27
non-singular, 15
prefix-free, 15
uniquely decodable, 15

universal data compression, 27
variable-length code, 14