Chapter 1

A Context for Error Correction Coding

I will make weak things become strong unto them ...

—Ether 12:27

... he denies that any error in the machine is responsible for the so-called errors in the answers. He claims that the Machines are self correcting and that it would violate the fundamental laws of nature for an error to exist in the circuits of relays.

—Isaac Asimov
I, Robot

1.1 Purpose of This Book

Error control coding in the context of digital communication has a history dating back to the middle of the twentieth century. In recent years, the field has been revolutionized by codes which are capable of approaching the theoretical limits of performance, the *channel capacity*. This has been impelled by a trend away from purely combinatoric and discrete approaches to coding theory toward codes which are more closely tied to a physical channel and soft decoding techniques. The purpose of this book is to present error correction and detection coding covering both traditional concepts thoroughly as well as modern developments in soft-decision and iteratively decoded codes and recent decoding algorithms for algebraic codes. An attempt has been made to maintain some degree of balance between the mathematics and their engineering implications by presenting both the mathematical methods used in understanding the codes as well as the algorithms which are used to efficiently encode and decode.

1.2 Introduction: Where Are Codes?

Error correction coding is the means whereby errors which may be introduced into digital data as a result of transmission through a communication channel can be corrected based upon received data. Error detection coding is the means whereby errors can be detected based upon received information. Collectively, error correction and error detection coding are *error control coding*. Error control coding can provide the difference between an operational communications system and a dysfunctional system. It has been a significant enabler in the telecommunications revolution, portable computers, the Internet, digital media, and space exploration. Error control coding is nearly ubiquitous in modern, information-based society. Every compact disc, CD-ROM, or DVD employs codes to protect the data embedded in the plastic disk. Every flash drive (thumb drive) and hard disk drive employs correction coding. Every phone call made over a digital cellular phone employs it. Every frame of digital television is protected by error correction coding. Every packet transmitted over the Internet has a protective coding "wrapper" used to determine if the packet has been received correctly. Even everyday commerce takes advantage of error detection coding, as the following examples illustrate.

Error Correction Coding: Mathematical Methods and Algorithms, Second Edition. Todd K. Moon

Companion website: www.wiley.com/go/Moon/ErrorCorrectionCoding

Example 1.1 The ISBN (international standard book number) is used to uniquely identify books. An ISBN such as 0-201-36186-8 can be parsed as

$$\underbrace{0}_{\text{country}} - \underbrace{20}_{\text{publisher}} - \underbrace{1-36186}_{\text{book no.}} - \underbrace{8}_{\text{check}}.$$

Hyphens do not matter. The first digit indicates a country/language, with 0 for the United States. The next two digits are a publisher code. The next six digits are a publisher-assigned book number. The last digit is a check digit, used to validate if the code is correct using what is known as a weighted code. An ISBN is checked as follows: the cumulative sum of the digits is computed, then the cumulative sum of the cumulative sum is computed. For a valid ISBN, the sum-of-the-sum must be equal to 0, modulo 11. The character X is used for the check digit 10. For this ISBN, we have

Digit	Cumulative Sum	Cumulative Sum
0	0	0
2	2	2
0	2	4
1	3	7
3	6	13
6	12	25
1	13	38
8	21	59
6	27	86
8	35	121

The final sum-of-the-sum is 121, which is equal to 0 modulo 11 (i.e., the remainder after dividing by 11 is 0). □

Example 1.2 The Universal Product Codes (UPC) employed on the bar codes of most merchandise employ a simple error detection system to help ensure reliability in scanning. In this case, the error detection system consists of a simple parity check. A UPC consists of a 12-digit sequence, which can be parsed as

$$\underbrace{0\ \ 16000}_{\substack{\text{manufacturer}\\ \text{identification}\\ \text{number}}} \quad \underbrace{66610}_{\substack{\text{item}\\ \text{number}}} \quad \underbrace{8}_{\substack{\text{parity}\\ \text{check}}}.$$

Denoting the digits as $u_1, u_2, \ldots, u_{12}$, the parity digit u_{12} is determined such that

$$3(u_1 + u_3 + u_5 + u_7 + u_9 + u_{11}) + (u_2 + u_4 + u_6 + u_8 + u_{10} + u_{12})$$

is a multiple of 10. In this case,

$$3(0 + 6 + 0 + 6 + 6 + 0) + (1 + 0 + 0 + 6 + 1 + 8) = 70.$$

If, when a product is scanned, the parity condition does not work, the operator is flagged so that the object may be re-scanned. □

1.3 The Communications System

Appreciation of the contributions of coding and understanding its limitations require some awareness of information theory and how its major theorems delimit the performance of a digital communication system. Information theory is increasingly relevant to coding theory, because with recent advances in theory it is now possible to achieve the performance bounds of information theory. By contrast, in the past, the bounds were more of a backdrop to the action on the stage of coding research and practice. Part of this success has come by placing the coding problem more fully in its communications context,

marrying the coding problem more closely to the signal detection problem instead of treating the coding problem mostly as one of discrete combinatorics.

Information theory treats *information* almost as a physical quantity which can be measured, transformed, stored, and moved from place to place. A fundamental concept of information theory is that information is conveyed by the resolution of uncertainty. Information can be measured by the amount of uncertainty resolved. For example, if a digital source always emits the same value, say 1, then no information is gained by observing that the source has produced, yet again, the output 1, since there was no uncertainty about the outcome to begin with. Probabilities are used to mathematically describe the uncertainty. For a discrete random variable X (i.e., one which produces discrete outputs, such as $X = 0$ or $X = 1$), the information conveyed by observing an outcome x is defined as $-\log_2 P(X = x)$ bits. (If the logarithm is base 2, the units of information are in **bits**. If the natural logarithm is employed, the units of information are in **nats**.) For example, if $P(X = 1) = 1$ (the outcome 1 is certain), then observing $X = 1$ yields $-\log_2(1) = 0$ bits of information. On the other hand, observing $X = 0$ in this case yields $-\log_2(0) = \infty$: total surprise at observing an impossible outcome. The *entropy* is the *average information*. For a binary source X having two outcomes occurring with probabilities p and $1 - p$, the binary entropy function, denoted as either $H_2(X)$ (indicating that it is the entropy of the source) or $H_2(p)$ (indicating that it is a function of the outcome probabilities) is

$$H_2(X) = H_2(p) = E[-\log_2 P(X)] = -p \log_2(p) - (1 - p) \log_2(1 - p) \text{ bits.}$$

A plot of the binary entropy function as a function of p is shown in Figure 1.1. The peak information of 1 bit occurs when $p = \frac{1}{2}$.

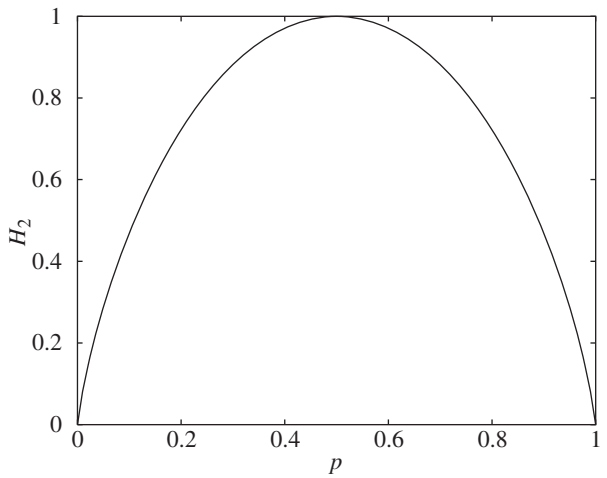


Figure 1.1: The binary entropy function $H_2(p)$.

Example 1.3 A fair coin is tossed once per second, with the outcomes being "head" and "tail" with equal probability. Each toss of the coin generates an event that can be described with $H_2(0.5) = 1$ bit of information. The sequence of tosses produces information at a rate of 1 bit/second. An unfair coin, with $P(\text{head}) = 0.01$ is tossed. The average information generated by each throw in this case is $H_2(0.01) = 0.0808$ bits. Another unfair coin, with $P(\text{head}) = 1$ is tossed. The information generated by each throw in this case is $H_2(1) = 0$ bits. □

For a source X having M outcomes $x_1, x_2, \ldots, x_M$, with probabilities $P(X = x_i) = p_i, i = 1, 2, \ldots, M$, the entropy is

$$H(X) = E[-\log_2 P(X)] = -\sum_{i=1}^{M} p_i \log_2 p_i \text{ bits.} \tag{1.1}$$

Note: The "bit" as a measure of entropy (or information content) is different from the "bit" as a measure of storage. For the unfair coin having $P(\text{head}) = 1$, the actual information content determined by a toss of the coin is 0: there is no information gained by observing that the outcome is again 1. For this process with this unfair coin, the entropy rate — that is, the amount of actual information it generates — is 0. However, if the coin outcomes were for some reason to be stored directly, without the benefit of some kind of coding, each outcome would require 1 bit of storage (even though they don't represent any new information).

With the prevalence of computers in our society, we are accustomed to thinking in terms of "bits" — e.g., a file is so many bits long, the register of a computer is so many bits wide. But these are "bits" as a measure of storage size, not "bits" as a measure of actual information content. Because of the confusion between "bit" as a unit of information content and "bit" as an amount of storage, the unit of information content is sometimes — albeit rarely — called a *Shannon*, in homage to the founder of information theory, Claude Shannon.[1]

A digital communication system embodies functionality to perform physical actions on information. Figure 1.2 illustrates a fairly general framework for a single digital communication link. In this link, digital data from a *source* are encoded and modulated (and possibly encrypted) for communication over a *channel*. At the other end of the channel, the data are demodulated, decoded (and possibly decrypted), and sent to a sink. The elements in this link all have mathematical descriptions and theorems from information theory which govern their performance. The diagram indicates the realm of applicability of three major theorems of information theory.

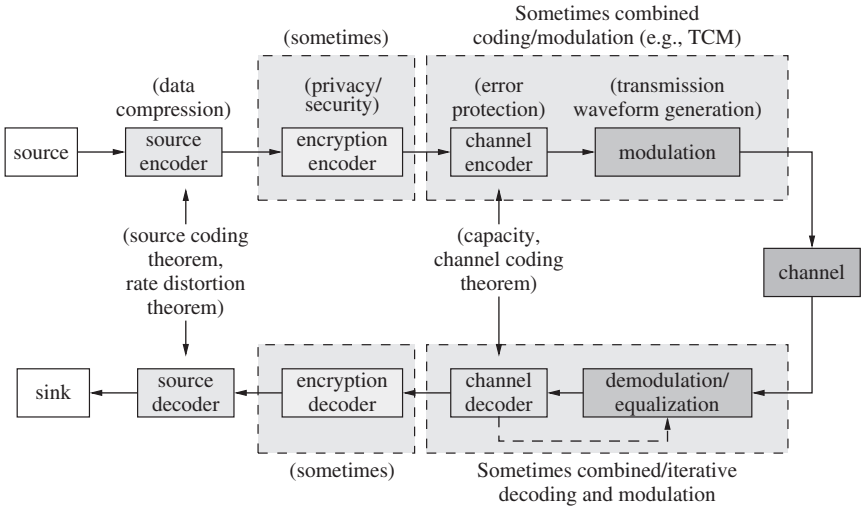


Figure 1.2: A general framework for digital communications.

There are actually many kinds of codes employed in a communication system. In the description below, we point out where some of these codes arise. Throughout the book we make some connections between these codes and our major focus of study, error correction codes.

The source is the data to be communicated, such as a computer file, a video sequence, or a telephone conversation. For our purposes, it is represented in digital form, perhaps as a result of an

[1] This mismatch of object and value is analogous to the physical horse, which may or may not be capable of producing one "horsepower" of power, 550 ft-lbs/second. Thermodynamicists can dodge the issue by using the SI unit of Watts for power, information theorists might sidestep confusion by using the Shannon. Both of these units honor founders of their respective disciplines.

analog-to-digital conversion step. Information-theoretically, sources are viewed as streams of random numbers governed by some probability distribution.

Every source of data has a measure of the information that it represents, which (in principle) can be exactly quantified in terms of entropy.

The source encoder performs data compression by removing redundancy.

As illustrated in Example 1.3, the number of bits used to store the information from a source may exceed the number of bits of actual information content. That is, the number of bits to represent the data may exceed the number of mathematical bits — Shannons — of actual information content.

The amount a particular source of data can be compressed without any loss of information (*lossless* compression) is governed theoretically by the *source coding theorem* of information theory, which states that a source of information can be represented without any loss of information in such a way that the amount of storage required (in bits) is equal to the amount of information content — the entropy — in bits or Shannons. To achieve this lower bound, it may be necessary for long blocks of the data to be jointly encoded.

Example 1.4 For the unfair coin with $P(\text{head}) = 0.01$, the entropy is $H(0.01) = 0.0808$. Therefore, 10,000 such (independent) tosses convey 808 bits (Shannons) of information, so theoretically the information of 10,000 tosses of the coin can be represented exactly using only 808 (physical) bits of information. □

Thus a bit (in a computer register) in principle *can* represents an actual (mathematical) bit of information content, if the source of information is represented correctly.

In compressing a data stream, a source encoder removes redundancy present in the data. For compressed binary data, 0 and 1 occur with equal probability in the compressed data (otherwise, there would be some redundancy which could be exploited to further compress the data). Thus, it is frequently assumed at the channel coder that 0 and 1 occur with equal probability.

The source encoder employs special types of codes to do the data compression, called collectively source codes or data compression codes. Such coding techniques include Huffman coding, run-length coding, arithmetic coding, Lempel–Ziv coding, and combinations of these, all of which fall beyond the scope of this book.

If the data need to be compressed below the entropy rate of the source, then some kind of distortion must occur. This is called lossy data compression. In this case, another theorem governs the representation of the data. It is possible to do lossy compression in a way that minimizes the amount of distortion for a given rate of transmission. The theoretical limits of lossy data compression are established by the *rate–distortion theorem* of information theory. One interesting result of rate–distortion theory says that for a binary source having equiprobable outcomes, the minimal rate to which the data can be compressed with the average distortion per bit equal to p is

$$r = 1 - H_2(p) \quad p \leq \frac{1}{2}. \tag{1.2}$$

Lossy data compression uses its own kinds of codes as well.

Example 1.5 In the previous example, suppose that a channel is available that only provides 800 bits of information to convey 10,000 tosses of the biased coin. Since this number of bits is less than allowed by the source coding theorem, there must be some distortion introduced. From above, the source rate is 0.0808 bits

of information per toss. The rate we are sending over this restricted channel is $800/10\,000 = 0.08$ bits/toss. The rate at which tosses are coded is

$$r = \frac{800}{808} = 0.9901.$$

The distortion is

$$p = H_2^{-1}(1 - r) = H_2^{-1}(1 - 0.9901) = H_2^{-1}(0.0099) = 0.00085.$$

That is, out of 10,000 coin tosses, transmission at this rate would introduce distortion in (0.00085)(10,000) = 8.5 bits.

□

The encrypter hides or scrambles information so that unintended listeners are unable to discern the information content. The codes used for encryption are generally different from the codes used for error correction.

Encryption is often what the layperson frequently thinks of when they think of "codes," as in secret codes, but as we are seeing, there are many other different kinds of codes.

As we will see, however, the mathematical tools used in error correction coding can be applied to some encryption codes. In particular, we will meet RSA public key encryption as an outgrowth of number theoretic tools to be developed, and McEliece public key encryption as a direct application of a particular family of error correction codes.

The channel coder is the first step in the error correction or error detection process.

The channel coder adds redundant information to the stream of input symbols in a way that allows errors which are introduced into the channel to be corrected. **This book is primarily dedicated to the study of the channel coder and its corresponding channel decoder.**

It may seem peculiar to remove redundancy with the source encoder, then turn right around and add redundancy back in with the channel encoder. However, the redundancy in the source typically depends on the source in an unstructured way and may not provide uniform protection to all the information in the stream, nor provide any indication of how errors occurred or how to correct them. The redundancy provided by the channel coder, on the other hand, is introduced in a structured way, precisely to provide error control capability.

Treating the problems of data compression and error correction separately, rather than seeking a jointly optimal source/channel coding solution, is asymptotically optimal (as the block sizes get large). This fact is called the *source–channel separation theorem* of information theory. (There has been work on combined source/channel coding for finite — practical — block lengths, in which the asymptotic theorems are not invoked. This work falls outside the scope of this book.)

Because of the redundancy introduced by the channel coder, there must be more symbols at the output of the coder than at the input. Frequently, a channel coder operates by accepting a block of k input symbols and producing at its output a block of n symbols, with $n > k$. The **rate** of such a channel coder is

$$R = k/n,$$

so that $R < 1$.

The input to the channel coder is referred to as the *message symbols* (or, in the case of binary codes, the *message bits*). The input may also be referred to as the *information symbols* (or bits).

The modulator converts symbol sequences from the channel encoders into signals appropriate for transmission over the channel. Many channels require that the signals be sent as a continuous-time voltage, or an electromagnetic waveform in a specified frequency band. The modulator provides the appropriate channel-conforming representation.

Included within the modulator block one may find codes as well. Some channels (such as magnetic recording channels) have constraints on the maximum permissible length of runs of 1s. Or they might have a restriction that the sequence must be DC-free. Enforcing such constraints employs special codes. Treatment of such runlength-limited codes appears in [275]; see also [209].

Some modulators employ mechanisms to ensure that the signal occupies a broad bandwidth. This *spread-spectrum* modulation can serve to provide multiple-user access, greater resilience to jamming, low probability of detection, and other advantages. (See, e.g., [509].) Spread-spectrum systems frequently make use of pseudorandom sequences, some of which are produced using linear feedback shift registers as discussed in Appendix 4.A.

The channel is the medium over which the information is conveyed. Examples of channels are telephone lines, internet cables, fiber-optic lines, microwave radio channels, high-frequency channels, cell phone channels, etc. These are channels in which information is conveyed between two distinct places. Information may also be conveyed between two distinct times, for example, by writing information onto a computer disk, then retrieving it at a later time. Hard drives, CD-ROMs, DVDs, and solid-state memory are other examples of channels.

As signals travel through a channel, they are corrupted. For example, a signal may have noise added to it; it may experience time delay or timing jitter, or suffer from attenuation due to propagation distance and/or carrier offset; it may be reflected by multiple objects in its path, resulting in constructive and/or destructive interference patterns; it may experience inadvertent interference from other channels, or be deliberately jammed. It may be filtered by the channel response, resulting in interference among symbols. These sources of corruption in many cases can all occur simultaneously.

For purposes of analysis, channels are frequently characterized by mathematical models, which (it is hoped) are sufficiently accurate to be representative of the attributes of the actual channel, yet are also sufficiently abstracted to yield tractable mathematics. Most of our work in this book will assume one of two idealized channel models, the binary symmetric channel (BSC) and the additive white Gaussian noise channel (AWGNC), which are described in Section 1.5. While these idealized models do not represent all of the possible problems a signal may experience, they form a starting point for many, if not most, of the more comprehensive channel models. The experience gained by studying these simpler channel models forms a foundation for more accurate and complicated channel models. (As exceptions to the AWGN or BSC rule, in Section 14.7, we comment briefly on convolutive channels and turbo equalization, while in Chapter 19, coding for quasi-static Rayleigh flat fading channels are discussed.)

As suggested by Figure 1.2, the channel encoding and modulation may be combined in what is known as *coded modulation*.

Channels have different information-carrying capabilities. For example, a dedicated fiber-optic line is capable of carrying more information than a plain-old-telephone-service (POTS) pair of copper wires. Associated with each channel is a quantity known as the **capacity**, C, which indicates how much information it can carry **reliably**.

The information a channel can reliably carry is intimately related to the use of error correction coding. The governing theorem from information theory is Shannon's **channel coding theorem**, which states essentially this: Provided that the rate R of transmission is less than the capacity C, *there exists* a code such that the probability of error can be made arbitrarily small.

The demodulator/equalizer receives the signal from the channel and converts it into a sequence of symbols. This typically involves many functions, such as filtering, demodulation, carrier synchronization, symbol timing estimation, frame synchronization, and matched filtering, followed

by a detection step in which decisions about the transmitted symbols are made. We will not concern ourselves in this book with these very important details (many of which are treated in textbooks on digital communication, such as [367]), but will focus on issues related to channel encoding and decoding.

The channel decoder exploits the redundancy introduced by the channel encoder to correct any errors that may have been introduced. As suggested by the figure, demodulation, equalization, and decoding may be combined. Particularly in recent work, turbo equalizers (introduced in Chapter 14) are used in a powerful combination.

The decrypter removes any encryption.

The source decoder provides an uncompressed representation of the data.

The sink is the ultimate destination of the data.

As this summary description has indicated, there are many different kinds of codes employed in communications. This book treats only error correction (or detection) codes. There is a certain duality between some channel coding methods and some source coding methods. So, studying error correction does provide a foundation for other aspects of the communication system.

1.4 Basic Digital Communications

The study of modulation/channel/demodulation/detection falls in the realm of "digital communications," and many of its issues (e.g., filtering, synchronization, carrier tracking) lie beyond the scope of this book. Nevertheless, some understanding of digital communications is necessary here because modern coding theory has achieved some of its successes by careful application of detection theory, in particular in maximum *a posteriori* (MAP) and maximum likelihood (ML) receivers. Furthermore, performance of codes is often plotted in terms of signal-to-noise ratio (SNR), which is understood in the context of the modulation of a physical waveform. Coded modulation relies on signal constellations beyond simple binary modulation, so an understanding of them is important.

The material in this section is standard for a digital communications course. However, it is germane to our treatment here because these concepts are employed in the development of soft-decision decoding algorithms.

1.4.1 Binary Phase-Shift Keying

In digital communication, a stream of bits (i.e., a sequence of 1s and 0s) is mapped to a waveform for transmission over the channel. Binary phase-shift keying (BPSK) is a form of amplitude modulation in which a bit is represented by the sign of the transmitted waveform. (It is called "phase-shift" keying [PSK] because the sign change represents a 180° phase shift.) Let $\{\ldots, b_{-2}, b_{-1}, b_0, b_1, b_2, \ldots\}$ represent a sequence of bits, $b_i \in \{0, 1\}$ which arrive at a rate of 1 bit every T seconds. The bits are assumed to be randomly generated with probabilities $P_1 = P(b_i = 1)$ and $P_0 = P(b_i = 0)$. While typically 0 and 1 are equally likely, we will initially retain a measure of generality and assume that $P_1 \neq P_0$ necessarily. It will frequently be of interest to map the set $\{0, 1\}$ to the set $\{-1, 1\}$. We will denote $\tilde{b}_i$ as the ± 1-valued bit corresponding to the $\{0, 1\}$-valued bit b_i. Either of the mappings

$$\tilde{b}_i = (2b_i - 1) \quad \text{or} \quad \tilde{b}_i = -(2b_i - 1)$$

may be used in practice, so some care is needed to make sure the proper mapping is understood.

Here, let $a_i = \sqrt{E_b}(2b_i - 1) = -\sqrt{E_b}(-1)^{b_i} = \sqrt{E_b}\tilde{b}_i$ be a mapping of bit b_i (or $\tilde{b}_i$) into a transmitted signal amplitude. This signal amplitude multiplies a waveform $\varphi_1(t)$, where $\varphi_1(t)$ is a unit-energy

signal,

$$\int_{-\infty}^{\infty} \varphi_1(t)^2 \, dt = 1,$$

which has support[2] over $[0, T)$. Thus, a bit b_i arriving at time iT can be represented by the signal $a_i\varphi_1(t - iT)$. The energy required to transmit a single bit b_i is

$$\int_{-\infty}^{\infty} (a_i\varphi_1(t))^2 \, dt = E_b.$$

Thus, E_b is the energy expended per bit.

It is helpful to think of the transmitted signals $\sqrt{E_b}\varphi_1(t)$ and $-\sqrt{E_b}\varphi_1(t)$ as points $\sqrt{E_b}$ and $-\sqrt{E_b}$ in a one-dimensional **signal space**, where the coordinate axis is the "$\varphi_1(t)$" axis. The two points in the signal space are plotted with their corresponding bit assignment in Figure 1.3. The points in the signal space employed by the modulator are called the **signal constellation**, so Figure 1.3 is a signal constellation with two points (or signals).

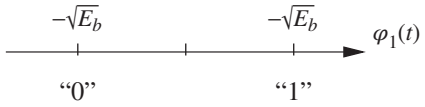


Figure 1.3 Signal constellation for BPSK.

A sequence of bits to be transmitted can be represented by a juxtaposition of $\varphi_1(t)$ waveforms, where the waveform representing bit b_i starts at time iT. Then the sequence of bits is represented by the signal

$$s(t) = \sum_i a_i\varphi_1(t - iT). \tag{1.3}$$

Example 1.6 With $\varphi_1(t)$ as shown in Figure 1.4(a) and the bit sequence $\{1, 1, 0, 1, 0\}$, the signal $s(t)$ is as shown in Figure 1.4(b).

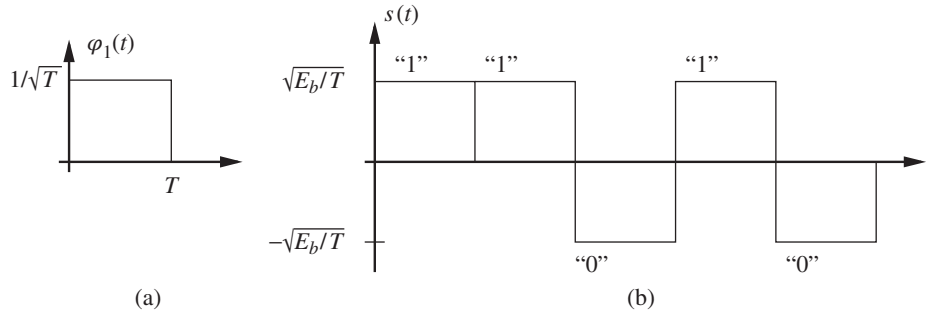


Figure 1.4: Juxtaposition of signal waveforms.

□

1.4.2 More General Digital Modulation

The concept of signal spaces generalizes immediately to higher dimensions and to larger signal constellations; we restrict our attention here to no more than two dimensions. Let $\varphi_2(t)$ be a unit-energy

[2] Strictly speaking, functions not having this limited support can be used, but assuming support over $[0, T)$ makes the discussion significantly simpler. Also, the signal $\varphi_i(t)$ can in general be complex, but we restrict attention here to real signals.

function which is orthogonal to $\varphi_1(t)$. That is,

$$\int_{-\infty}^{\infty} \varphi_2(t)^2 \, dt = 1 \quad \text{and} \quad \int_{-\infty}^{\infty} \varphi_1(t)\varphi_2(t) \, dt = 0.$$

In this case, we are defining "orthogonality" with respect to the inner product

$$\langle \varphi_1(t), \varphi_2(t) \rangle = \int_{-\infty}^{\infty} \varphi_1(t)\varphi_2(t) \, dt.$$

We say that $\{\varphi_1(t), \varphi_2(t)\}$ form an *orthonormal set* if they both have unit energy and are orthogonal:

$$\langle \varphi_1(t), \varphi_1(t) \rangle = 1 \quad \langle \varphi_2(t), \varphi_2(t) \rangle = 1 \quad \langle \varphi_1(t), \varphi_2(t) \rangle = 0.$$

The orthonormal functions $\varphi_1(t)$ and $\varphi_2(t)$ define the coordinate axes of a two-dimensional signal space, as suggested by Figure 1.5. Corresponding to every point (x_1, y_1) of this two-dimensional signal space is a signal (i.e., a function of time) $s(t)$ obtained as a linear combination of the coordinate functions:

$$s(t) = x_1\varphi_1(t) + y_1\varphi_2(t).$$

That is, there is a one-to-one correspondence between "points" in space and their represented signals. We can represent this as

$$s(t) \leftrightarrow (x_1, y_1).$$

The geometric concepts of distance and angle can be expressed in terms of the signal space points. For example, let

$$\begin{aligned} s_1(t) &= x_1\varphi_1(t) + y_1\varphi_2(t) \quad (\text{i.e., } s_1(t) \leftrightarrow (x_1, y_1)) \\ s_2(t) &= x_2\varphi_1(t) + y_2\varphi_2(t) \quad (\text{i.e., } s_2(t) \leftrightarrow (x_2, y_2)) \end{aligned} \tag{1.4}$$

We define the squared distance between $s_1(t)$ and $s_2(t)$ as

$$d^2(s_1(t), s_2(t)) = \int_{-\infty}^{\infty} (s_1(t) - s_2(t))^2 \, dt, \tag{1.5}$$

and the inner product between $s_1(t)$ and $s_2(t)$ as

$$\langle s_1(t), s_2(t) \rangle = \int_{-\infty}^{\infty} s_1(t)s_2(t) \, dt. \tag{1.6}$$

Rather than computing distance using the integral (1.5), we can equivalently and more easily compute using the coordinates in signal space (see Figure 1.5):

$$d^2(s_1(t), s_2(t)) = (x_1 - x_2)^2 + (y_1 - y_2)^2. \tag{1.7}$$

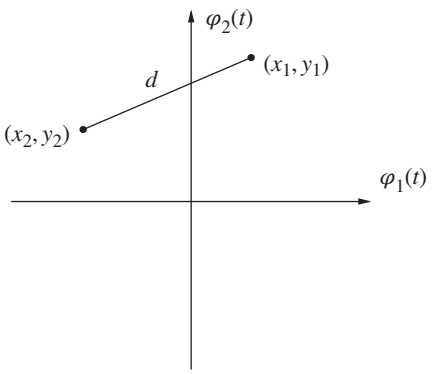


Figure 1.5: Two-dimensional signal space.

This is the familiar squared **Euclidean distance** between the points (x_1, y_1) and (x_2, y_2). Also, rather than computing the inner product using the integral (1.6), we equivalently compute using the coordinates in signal space:

$$\langle s_1(t), s_2(t) \rangle = x_1 x_2 + y_1 y_2. \tag{1.8}$$

This is the familiar inner product (or dot product) between the points (x_1, y_1) and (x_2, y_2):

$$\langle (x_1, y_1), (x_2, y_2) \rangle = x_1 x_2 + y_1 y_2.$$

The significance is that we can use the signal space geometry to gain insight into the nature of the signals, using familiar concepts of distance and angle.

We can use this two-dimensional signal space for digital information transmission as follows. Let $M = 2^m$, for some integer m, be the number of points in the signal constellation. M-ary transmission is obtained by placing M points $(a_{1k}, a_{2k}), k = 0, 1, \ldots, M-1$, in this signal space and assigning a unique pattern of m bits to each of these points. These points are the signal constellation. Let

$$\mathcal{S} = \{(a_{1k}, a_{2k}), k = 0, 1, \ldots, M-1\}$$

denote the set of points in the signal constellation.

Example 1.7 Figure 1.6 shows eight points arranged in two-dimensional space in a constellation known as 8-PSK. Each point has a 3-bit designation. The signal corresponding to the point (a_{1k}, a_{2k}) is selected by three input bits and transmitted. Thus, the signal

$$s_k(t) = a_{1k}\varphi_1(t) + a_{2k}\varphi_2(t), \quad (a_{1k}, a_{2k}) \in \mathcal{S}$$

carries 3 bits of information.

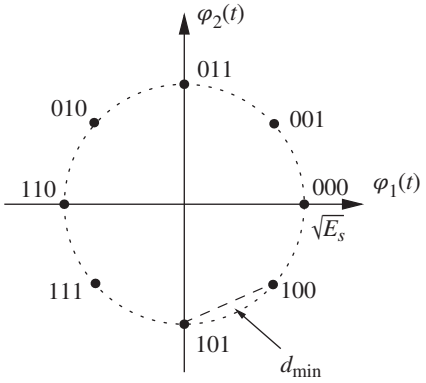


Figure 1.6: 8-PSK signal constellation.

Note that the assignments of bits to constellation points in Figure 1.6 is such that adjacent points differ by only 1 bit. Such an assignment is called Gray code order. Since it is most probable that errors will move from a point to an adjacent point, this reduces the probability of bit error. □

Associated with each signal $s_k(t) = a_{1k}\varphi_1(t) + a_{2k}\varphi_2(t)$ and signal constellation point $(a_{1k}, a_{2k}) \in \mathcal{S}$ is a signal energy,

$$E_k = \int_{-\infty}^{\infty} (s_k(t))^2 \, dt = a_{1k}^2 + a_{2k}^2.$$

The *average signal energy* E_s is obtained by averaging all the signal energies, usually by assuming that each signal point is used with equal probability:

$$E_s = \frac{1}{M}\sum_{k=0}^{M-1}\int_{-\infty}^{\infty} s_k(t)^2\, dt = \frac{1}{M}\sum_{k=0}^{M-1}(a_{1k}^2 + a_{2k}^2).$$

The average energy per signal E_s can be related to the average energy per bit E_b by

$$E_b = \frac{\text{energy per signal}}{\text{number of bits/signal}} = \frac{E_s}{m}.$$

To send a sequence of bits using M-ary modulation, the bits are partitioned into blocks of m successive bits, where the data rate is such that m bits arrive every T_s seconds. The ith m-bit set is then used to index a point $(a_{1i}, a_{2i}) \in \mathcal{S}$. This point corresponds to the signal which is transmitted over the signal interval for the m bits. These signals are juxtaposed to form the transmitted signal:

$$s(t) = \sum_i a_{1i}\varphi_1(t - iT_s) + a_{2i}\varphi_2(t - iT_s), \quad (a_{1i}, a_{2i}) \in \mathcal{S}. \tag{1.9}$$

The point (a_{1i}, a_{2i}) is thus the point set at time i. Equation (1.9) can be expressed in its signal space vector equivalent, by simply letting $\mathbf{s}_i = [a_{1i}, a_{2i}]^T$ denote the vector transmitted at time i.

In what follows, we will express the operations in terms of the two-dimensional signal space. Restricting to a one-dimensional signal space (as for BPSK transmission), or extending to higher-dimensional signal spaces is straightforward.

In most channels, the signal $s(t)$ is mixed with some carrier frequency before transmission. However, for simplicity, we will restrict attention to the baseband transmission case.

1.5 Signal Detection

1.5.1 The Gaussian Channel

The signal $s(t)$ is transmitted over the channel. Of all the possible disturbances that might be introduced by the channel, we will deal primarily with additive white Gaussian noise (AWGN), resulting in the received signal

$$r(t) = s(t) + n(t). \tag{1.10}$$

In an AWGN channel, the signal $n(t)$ is a white Gaussian noise process, having the properties that

$$E[n(t)] = 0 \quad \forall t,$$

$$R_n(\tau) = E[n(t)n(t-\tau)] = \frac{N_0}{2}\delta(\tau),$$

and all sets of samples are jointly Gaussian distributed. The quantity $N_0/2$ is the (two-sided) noise power spectral density.

Due to the added noise, the signal $r(t)$ is typically not a point in the signal constellation, nor, in fact, is $r(t)$ probably even *in* the signal space — it cannot be expressed as a linear combination of the basis functions $\varphi_1(t)$ and $\varphi_2(t)$. The detection process to be described below corresponds to the geometric operations of (1) projecting $r(t)$ onto the signal space; and (2) finding the closest point in the signal space to this projected function.

At the receiver, optimal detection requires first passing the received signal through a filter "matched" to the transmitted waveform. This is the projection operation, projecting $r(t)$ onto the signal space. To detect the ith signal starting at iT_s, the received signal is correlated with the waveforms

$\varphi_1(t - iT_s)$ and $\varphi_2(t - iT_s)$ to produce the point (R_{1i}, R_{2i}) in signal space[3]:

$$\begin{aligned} R_{1i} &= \langle r(t), \varphi_1(t - iT_s)\rangle = \int_{-\infty}^{\infty} r(t)\varphi_1(t - iT_s)\, dt, \\ R_{2i} &= \langle r(t), \varphi_2(t - iT_s)\rangle = \int_{-\infty}^{\infty} r(t)\varphi_2(t - iT_s)\, dt. \end{aligned} \tag{1.11}$$

The integral is over the support of $\varphi_1(t)$ and $\varphi_2(t)$. For example, if $\varphi_1(t)$ has support over $[0, T_s]$, then the limits of integration in (1.11) are $\int_{iT_s}^{(i+1)T_s}$. The processing in (1.11) is illustrated in Figure 1.7. Using (1.9) and (1.10), it is straightforward to show that

$$R_{1i} = a_{1i} + N_{1i} \quad \text{and} \quad R_{2i} = a_{2i} + N_{2i}, \tag{1.12}$$

where (a_{1i}, a_{2i}) is the transmitted point in the signal constellation for the ith symbol. The point (a_{1i}, a_{2i}) is not known at the receiver — it is, in fact, what the receiver needs to decide — so at the receiver a_{1i} and a_{2i} are random variables.

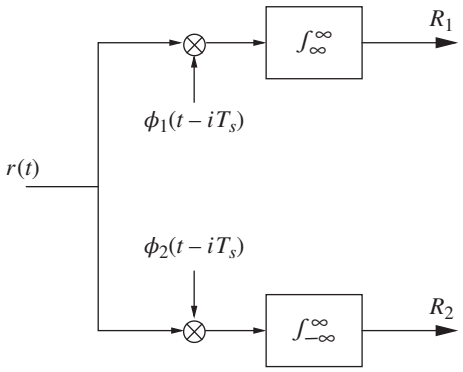


Figure 1.7: Correlation processing (equivalent to matched filtering).

The noise random variables N_{1i} and N_{2i} defined by

$$N_{1i} = \int_{iT_s}^{(i+1)T_s} \varphi_1(t - iT_s)n(t)\, dt \quad \text{and} \quad N_{2i} = \int_{iT_s}^{(i+1)T_s} \varphi_2(t - iT_s)n(t)\, dt$$

have the following properties: N_{1i} and N_{2i} are Gaussian random variables, with

$$E[N_{1i}] = 0 \quad \text{and} \quad E[N_{2i}] = 0 \tag{1.13}$$

and[4]

$$\text{var}[N_{1i}] \triangleq \sigma^2 = \frac{N_0}{2} \quad \text{and} \quad \text{var}[N_{2i}] \triangleq \sigma^2 = \frac{N_0}{2} \tag{1.14}$$

and

$$E[N_{1i}N_{2i}] = 0; \tag{1.15}$$

[3] The operation in (1.11) can equivalently be performed by passing $r(t)$ through filters with impulse response $\varphi_1(-t)$ and $\varphi_2(-t)$ and sampling the output at time $t = iT_s$. This is referred to as a *matched filter*. The matched filter implementation and the correlator implementation provide identical outputs.

[4] The symbol $\triangleq$ means "is defined to be equal to."

that is, N_{1i} and N_{2i} are uncorrelated and hence, being Gaussian, are independent. The probability density function (pdf) for N_{1i} or N_{2i} is

$$p_N(n_i) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{1}{2\sigma^2} n_i^2}.$$

It will be convenient to express (1.12) in vector form. Let $\mathbf{R}_i = [R_{1i}, R_{2i}]^T$ (received vector), $\mathbf{S}_i = [a_{1i}, a_{2i}]^T$ (sent vector), and $\mathbf{N}_i = [N_{1i}, N_{2i}]^T$ (noise vector). Then

$$\mathbf{R}_i = \mathbf{S}_i + \mathbf{N}_i.$$

Then $\mathbf{N}_i$ is jointly Gaussian distributed, with 0 mean and covariance matrix

$$E[\mathbf{N}_i \mathbf{N}_i^T] = \sigma^2 \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} = \sigma^2 I = R_N.$$

Explicitly, the pdf of the vector $\mathbf{N}_i$ is

$$p_{\mathbf{N}}(\mathbf{n}) = \frac{1}{2\pi\sqrt{\det(R_N)}} \exp\left[-\frac{1}{2}\mathbf{n}^T R_N^{-1} \mathbf{n}\right] = \frac{1}{2\pi\sigma^2} \exp\left[-\frac{1}{2\sigma^2}(n_1^2 + n_2^2)\right].$$

1.5.2 MAP and ML Detection

Let $\mathbf{S}$ denote the transmitted value, where $\mathbf{S} \in \mathcal{S}$ is chosen with prior probability $P(\mathbf{S} = \mathbf{s})$, or, more briefly, $P(\mathbf{s})$. The receiver uses the received point $\mathbf{R} = \mathbf{r}$ to make a decision about what the transmitted signal $\mathbf{S}$ is. Let $P(\mathbf{s}|\mathbf{r})$ be used to denote $P(\mathbf{S} = \mathbf{s}|\mathbf{R} = \mathbf{r}) = P_{\mathbf{S}|\mathbf{R}}(\mathbf{S} = \mathbf{s}|\mathbf{R} = \mathbf{r})$ for an observed value of the random variable $\mathbf{R} = \mathbf{r}$. Conditioned on knowing that the transmitted signal is $\mathbf{S} = \mathbf{s}$, $\mathbf{R}$ is a Gaussian random variable with conditional density

$$\begin{aligned} p_{R|S}(\mathbf{r}|\mathbf{s}) &= p_N(\mathbf{r} - \mathbf{s}) = \frac{1}{2\pi\sqrt{\det(R_N)}} \exp\left[-\frac{1}{2}(\mathbf{r} - \mathbf{s})^T R_N^{-1} (\mathbf{r} - \mathbf{s})\right] \\ &= C \exp\left[-\frac{1}{2\sigma^2} \|\mathbf{r} - \mathbf{s}\|^2\right], \end{aligned} \tag{1.16}$$

where $\|\mathbf{r} - \mathbf{s}\|^2$ is the squared Euclidean distance between $\mathbf{r}$ and $\mathbf{s}$ and C is a quantity that does not depend on either $\mathbf{R}$ or $\mathbf{S}$. The quantity $p_{R|S}(\mathbf{r}|\mathbf{s})$ is called the *likelihood function*. The likelihood function $p_{R|S}(\mathbf{r}|\mathbf{s})$ is typically viewed as a function of the *conditioning* argument, with the observed values $\mathbf{r}$ as fixed parameters.

The signal point $\mathbf{s} \in \mathcal{S}$ depends uniquely upon the transmitted bits mapped to the signal constellation point. Conditioning upon knowing the transmitted signal is thus equivalent to conditioning on knowing the transmitted bits. Thus, the notation $p(\mathbf{r}|\mathbf{s})$ is used interchangeably with $p(\mathbf{r}|\mathbf{b})$, when $\mathbf{s}$ is the signal used to represent the bits $\mathbf{b}$. For example, for BPSK modulation, we could write either $p(r|s = \sqrt{E_b})$ or $p(r|b = 1)$ or even $p(r|\tilde{b} = 1)$, since by the modulation described above the amplitude $\sqrt{E_b}$ is transmitted when the input bit is $b = 1$ (or $\tilde{b} = 1$).

Let us denote the estimated decision as $\hat{\mathbf{s}} = [\hat{a}_1, \hat{a}_2]^T \in \mathcal{S}$. We will use the notation $P(\mathbf{s}|\mathbf{r})$ as a shorthand for $P(\mathbf{S} = \mathbf{s}|\mathbf{r})$.

Theorem 1.8 *The decision rule which minimizes the probability of error is to choose* $\hat{\mathbf{s}}$ *to be that value of* $\mathbf{s}$ *which maximizes* $P(\mathbf{S} = \mathbf{s}|\mathbf{r})$*, where the possible values of* $\mathbf{s}$ *are those in the signal constellation* $\mathcal{S}$*. That is,*

$$\hat{\mathbf{s}} = \arg\max_{\mathbf{s} \in \mathcal{S}} P(\mathbf{s}|\mathbf{r}). \tag{1.17}$$

Proof Let us denote the constellation as $\mathcal{S} = \{\mathbf{s}_i, i = 1, 2, \ldots, M\}$. Let $p(\mathbf{r}|\mathbf{s}_i)$ denote the pdf of the received signal when $\mathbf{S} = \mathbf{s}_i$ is the transmitted signal. Let Ω denote the space of possible received values; in the current case $\Omega = \mathbb{R}^2$. Let us partition the space Ω into regions Ω_i, where the decision rule is expressed as: set $\hat{\mathbf{s}} = \mathbf{s}_i$ if $\mathbf{r} \in \Omega_i$. That is,

$$\Omega_i = \{\mathbf{r} \in \Omega \ : \ \text{decide } \hat{\mathbf{s}} = \mathbf{s}_i\}.$$

The problem, then, is to determine the partitions Ω_i. By the definition of the partition, the conditional probability of a correct answer when $\mathbf{S} = \mathbf{s}_i$ is sent is

$$P(\hat{\mathbf{s}} = \mathbf{s}_i|\mathbf{S} = \mathbf{s}_i) = \int_{\Omega_i} p(\mathbf{r}|\mathbf{s}_i)\, d\mathbf{r}.$$

Denote the conditional probability of error when signal $\mathbf{S} = \mathbf{s}_i$ is sent as $P_i(\mathcal{E})$:

$$P_i(\mathcal{E}) = P(\hat{\mathbf{s}} \neq \mathbf{s}_i|\mathbf{S} = \mathbf{s}_i).$$

Then we have

$$P_i(\mathcal{E}) = 1 - P(\hat{\mathbf{s}} = \mathbf{s}_i|\mathbf{S} = \mathbf{s}_i) = 1 - \int_{\Omega_i} p(\mathbf{r}|\mathbf{s}_i)\, d\mathbf{r}.$$

The average probability of error is

$$\begin{aligned} P(\mathcal{E}) &= \sum_{i=1}^{M} P_i(\mathcal{E}) P(\mathbf{S} = \mathbf{s}_i) = \sum_{i=1}^{M} P(\mathbf{S} = \mathbf{s}_i) \left[1 - \int_{\Omega_i} p(\mathbf{r}|\mathbf{s}_i)\, d\mathbf{r}\right] d\mathbf{r} \\ &= 1 - \sum_{i=1}^{M} \int_{\Omega_i} p(\mathbf{r}|\mathbf{s}_i) P(\mathbf{S} = \mathbf{s}_i)\, d\mathbf{r}. \end{aligned}$$

The probability of a correct answer is

$$P(C) = 1 - P(\mathcal{E}) = \sum_{i=1}^{M} \int_{\Omega_i} p(\mathbf{r}|\mathbf{s}_i) P(\mathbf{S} = \mathbf{s}_i)\, d\mathbf{r} = \sum_{i=1}^{M} \int_{\Omega_i} P(\mathbf{S} = \mathbf{s}_i|\mathbf{r}) p(\mathbf{r})\, d\mathbf{r}.$$

Since $p(\mathbf{r}) \geq 0$, to maximize $P(C)$, the region of integration Ω_i should be chosen precisely so that it covers the region where $P(\mathbf{S} = \mathbf{s}_i|\mathbf{r})$ is the largest possible. That is,

$$\Omega_i = \{\mathbf{r} \ : \ P(\mathbf{S} = \mathbf{s}_i|\mathbf{r}) > P(\mathbf{S} = \mathbf{s}_j|\mathbf{r}), i \neq j\}.$$

This is equivalent to (1.17). □

Using Bayes' rule we can write (1.17) as

$$\hat{\mathbf{s}} = \arg\max_{\mathbf{s}\in\mathcal{S}} P(\mathbf{s}|\mathbf{r}) = \arg\max_{\mathbf{s}\in\mathcal{S}} \frac{p_{R|S}(\mathbf{r}|\mathbf{s}) P(\mathbf{s})}{p_R(\mathbf{r})}.$$

Since the denominator of the last expression does not depend on $\mathbf{s}$, we can further write

$$\boxed{\hat{\mathbf{s}} = \arg\max_{\mathbf{s}\in\mathcal{S}} p_{R|S}(\mathbf{r}|\mathbf{s}) P(\mathbf{s}).} \tag{1.18}$$

This is called the MAP decision rule. In the case that all the prior probabilities are equal (or are assumed to be equal), this rule can be simplified to

$$\boxed{\hat{\mathbf{s}} = \arg\max_{\mathbf{s}\in\mathcal{S}} p_{R|S}(\mathbf{r}|\mathbf{s}).}$$

This is called the ML decision rule.

Note: We will frequently suppress the subscript on a pdf or distribution function, letting the arguments themselves indicate the intended random variables. We could thus write $p(\mathbf{r}|\mathbf{s})$ in place of $p_{R|S}(\mathbf{r}|\mathbf{s})$.

Once the decision $\hat{\mathbf{s}}$ is made, the corresponding bits are determined by the bit-to-constellation mapping. The output of the receiver is thus an estimate of the bits.

By the form of (1.16), we see that the ML decision rule for the Gaussian noise channel selects that point $\hat{\mathbf{s}} \in \mathcal{S}$ which is closest to $\mathbf{r}$ in squared **Euclidean distance**, $\|\mathbf{r} - \hat{\mathbf{s}}\|^2$.

1.5.3 Special Case: Binary Detection

For the case of binary transmission in a one-dimensional signal space, the signal constellation consists of the points $\mathcal{S} = \{\sqrt{E_b}, -\sqrt{E_b}\}$, corresponding, respectively, to the bit $b = 1$ or $b = 0$ (respectively, using the current mapping). The corresponding likelihood functions are

$$p(r|s = \sqrt{E_b}) = \frac{1}{\sqrt{2\pi}} e^{-\frac{1}{2\sigma^2}(r-\sqrt{E_b})^2} \quad p(r|s = -\sqrt{E_b}) = \frac{1}{\sqrt{2\pi}} e^{-\frac{1}{2\sigma^2}(r+\sqrt{E_b})^2}.$$

These densities are plotted in Figure 1.8(a). We see $r|s = \sqrt{E_b}$ is a Gaussian with mean $\sqrt{E_b}$. The MAP decision rule compares the weighted densities $p(r|s = \sqrt{E_b})P(s = \sqrt{E_b})$ and $p(r|s = -\sqrt{E_b})P(s = -\sqrt{E_b})$. Figure 1.8(b) shows these densities in the case that $P(s = -\sqrt{E_b}) > P(s = \sqrt{E_b})$. Clearly, there is a threshold point τ at which

$$p(r|s = \sqrt{E_b})P(s = \sqrt{E_b}) = p(r|s = -\sqrt{E_b})P(s = -\sqrt{E_b}).$$

In this case, the decision rule (1.18) simplifies to

$$\hat{s} = \begin{cases} \sqrt{E_b} (\text{i.e., } b_i = 1) & \text{if } r > \tau \\ -\sqrt{E_b} (\text{i.e., } b_i = 0) & \text{if } r < \tau. \end{cases} \tag{1.19}$$

The threshold value can be computed explicitly as

$$\tau = \frac{\sigma^2}{2\sqrt{E_b}} \ln \frac{P(s = -\sqrt{E_b})}{P(s = \sqrt{E_b})}. \tag{1.20}$$

In the case that $P(s = \sqrt{E_b}) = P(s = -\sqrt{E_b})$, the decision threshold is at $\tau = 0$, as would be expected.

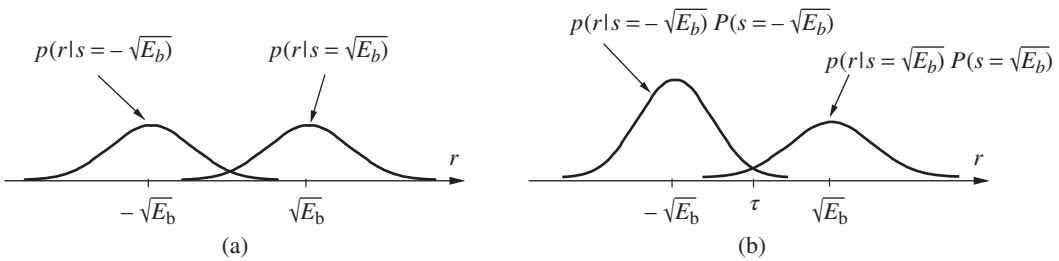


Figure 1.8: Conditional densities in BPSK modulation. (a) Conditional densities; (b) weighted conditional densities.

Binary detection problems are also frequently expressed in terms of *likelihood ratios*. For binary detection, the problem is one of determining, say, if $b = 1$ or if $b = 0$. The detection rule (1.18) becomes a test between

$$p(r|b = 1)P(b = 1) \quad \text{and} \quad P(r|b = 0)P(b = 0).$$

This can be expressed as a *ratio*,

$$\frac{p(r|b=1)P(b=1)}{p(r|b=0)P(b=0)}.$$

In the case of equal priors, we obtain the likelihood ratio

$$L(r) = \frac{p(r|b=1)}{p(r|b=0)}.$$

For many channels, it is more convenient to use the *log likelihood ratio*

$$\Lambda(r) = \log \frac{p(r|b=1)}{p(r|b=0)},$$

where the natural logarithm is usually used. The decision is made that $\hat{b} = 1$ if $\Lambda(r) > 0$ and $\hat{b} = 0$ if $\Lambda(r) < 0$.

For the Gaussian channel with BPSK modulation, we have

$$\Lambda(r) = \log \frac{p(r|a=\sqrt{E_b})}{p(r|a=-\sqrt{E_b})} = \log \frac{\exp(-\frac{1}{2\sigma^2}(r-\sqrt{E_b})^2)}{\exp(-\frac{1}{2\sigma^2}(r+\sqrt{E_b})^2)} = \frac{2\sqrt{E_b}}{\sigma^2} r = L_c r, \tag{1.21}$$

where $L_c = \frac{2\sqrt{E_b}}{\sigma^2}$ is called the **channel reliability**.[5]

The quantity $\Lambda(r) = L_c r$ can be used as *soft information* in a decoding system. The quantity $\text{sign}(\Lambda(r))$ is referred to as *hard information* in a decoding system. Most early error correction decoding algorithms employed hard information — actual estimated bit values — while there has been a trend toward increasing use of soft information decoders, which generally provide better performance.

Note: Beginning in Section 1.7, error correction coding is used, and the transmitted BPSK amplitudes are $\pm\sqrt{E_c}$, where $E_c = RE_b$. Using these amplitudes,

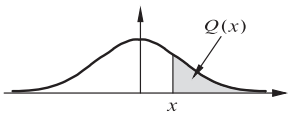
$$\Lambda(r) = \log \frac{p(r|a=\sqrt{E_c})}{p(r|a=-\sqrt{E_c})} = \frac{2\sqrt{E_c}}{\sigma^2} r$$

so that the channel reliability is $L_c = 2\sqrt{E_c}/\sigma^2$.

1.5.4 Probability of Error for Binary Detection

Even with optimum decisions at the demodulator, errors can still be made with some probability (otherwise, error correction coding would not ever be needed). For binary detection problems in Gaussian noise, the probabilities can be expressed using the $Q(x)$ function, which is the probability that a unit Gaussian $N \sim \mathcal{N}(0,1)$ exceeds x:

$$Q(x) = P(N > x) = \frac{1}{\sqrt{2\pi}} \int_x^\infty e^{-n^2/2}\, dn.$$



The Q function has the properties that

$$Q(x) = 1 - Q(-x) \quad Q(0) = \frac{1}{2} \quad Q(-\infty) = 1 \quad Q(\infty) = 0.$$

[5] In some sources (e.g., [178]) the channel reliability L_c is expressed alternatively as equivalent to $2E_b/\sigma^2$. This is in some ways preferable, since it is unitless.

For a Gaussian random variable Z with mean μ and variance σ^2, $Z \sim \mathcal{N}(\mu, \sigma^2)$, it is straightforward to show that

$$P(Z > x) = \frac{1}{\sqrt{2\pi}\sigma} \int_x^\infty e^{-(z-\mu)^2/2\sigma^2}\, dz = Q\left(\frac{x-\mu}{\sigma}\right).$$

Suppose there are two points a and b along an axis, and that

$$R = s + N,$$

where s is one of the two points, and $N \sim \mathcal{N}(0, \sigma^2)$. The distributions $P(R|s=a)P(s=a)$ and $P(R|s=b)P(s=b)$ are plotted in Figure 1.9. A decision threshold τ is also shown. When a is sent, an error is made when $R > \tau$. Denoting $\mathcal{E}$ as the error event, this occurs with probability

$$P(\mathcal{E}|s=a) = P(R > \tau) = \frac{1}{\sqrt{2\pi}\sigma} \int_\tau^\infty e^{-\frac{1}{2\sigma^2}(r-a)^2}\, dr = Q\left(\frac{\tau-a}{\sigma}\right).$$

When b is sent, an error is made when $R < \tau$, which occurs with probability

$$P(\mathcal{E}|s=b) = P(R < \tau) = 1 - P(R > \tau) = 1 - Q\left(\frac{\tau-b}{\sigma}\right) = Q\left(\frac{b-\tau}{\sigma}\right).$$

The overall probability of error is

$$\begin{aligned} P(\mathcal{E}) &= P(\mathcal{E}|s=a)P(s=a) + P(\mathcal{E}|s=b)P(s=b) \\ &= Q\left(\frac{\tau-a}{\sigma}\right)P(s=a) + Q\left(\frac{b-\tau}{\sigma}\right)P(s=b). \end{aligned} \tag{1.22}$$

An important special case is when $P(s=a) = P(s=b) = \frac{1}{2}$. Then the decision threshold is at the midpoint $\tau = (a+b)/2$. Let $d = |b-a|$ be the distance between the two signals. Then (1.22) can be written

$$\boxed{P(\mathcal{E}) = Q\left(\frac{d}{2\sigma}\right).} \tag{1.23}$$

Even in the case that the signals are transmitted in multidimensional space, provided that the covariance of the noise is of the form $\sigma^2 I$, the probability of error is still of the form (1.23). That is, if

$$\mathbf{R} = \mathbf{s} + \mathbf{N}$$

are n-dimensional vectors, with $\mathbf{N} \sim \mathcal{N}(\mathbf{0}, \sigma^2 I)$, and $\mathbf{S} \in \{\mathbf{a}, \mathbf{b}\}$ are two equiprobable transmitted vectors, then the probability of decision error is $P(\mathcal{E}) = Q\left(\frac{d}{2\sigma}\right)$, where $d = \| \mathbf{a} - \mathbf{b} \|$ is the Euclidean distance between vectors. This formula is frequently used in characterizing the performance of codes.

For the particular case of BPSK signaling, we have $a = -\sqrt{E_b}$, $b = \sqrt{E_b}$, and $d = 2\sqrt{E_b}$. The probability $P(\mathcal{E})$ is denoted as P_b, the "probability of a bit error." Thus,

$$P_b = Q\left(\frac{\tau + \sqrt{E_b}}{\sigma}\right)P(-\sqrt{E_b}) + Q\left(\frac{\sqrt{E_b} - \tau}{\sigma}\right)P(\sqrt{E_b}). \tag{1.24}$$

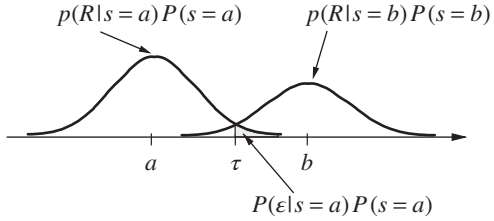
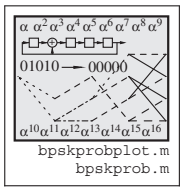


Figure 1.9: Distributions when two signals are sent in Gaussian noise.

When $P(\sqrt{E_b}) = P(-\sqrt{E_b})$, then $\tau = 0$. Recalling that the variance for the channel is expressed as $\sigma^2 = \frac{N_0}{2}$, we have for BPSK transmission

$$P_b = Q(\sqrt{E_b}/\sigma) = Q\left(\sqrt{\frac{2E_b}{N_0}}\right). \tag{1.25}$$



The quantity E_b/N_0 is frequently called the (bit) SNR.

Figure 1.10 shows the probability of bit error for a BPSK as a function of the SNR in dB (decibel), where

$$E_b/N_0 \text{ dB} = 10 \log_{10} E_b/N_0,$$

for the case $P(\sqrt{E_b}) = P(-\sqrt{E_b})$.

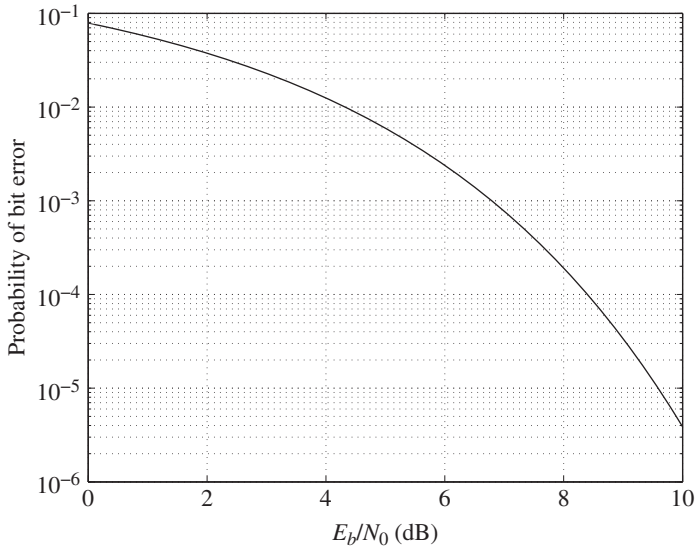


Figure 1.10: Probability of error for BPSK signaling.

1.5.5 Bounds on Performance: The Union Bound

For some signal constellations, exact expressions for the probability of error are difficult to obtain. In many cases it is more convenient to obtain a *bound* on the probability of error using the union bound. (See Box 1.1.) Consider, for example, the 8-PSK constellation in Figure 1.11. If the point labeled $\mathbf{s}_0$ is transmitted, then an error occurs if the received signal falls in either shaded area. Let A be the event that the received signal falls on the incorrect side of threshold line L_1 and let B be the event that the received signal falls on the incorrect side of the line L_2. Then

$$\Pr(\text{symbol decoding error}|\mathbf{s}_0 \text{ sent}) = P(A \cup B).$$

The events A and B are not disjoint, as is apparent from Figure 1.11. The exact probability computation is made more difficult by the overlapping region. Using the union bound, however, the probability of error can be bounded as

$$\Pr(\text{symbol decoding error}|\mathbf{s}_0 \text{ sent}) \leq P(A) + P(B).$$

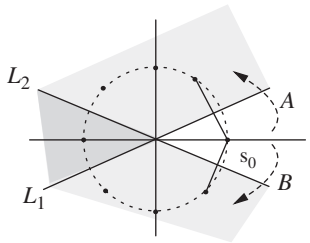
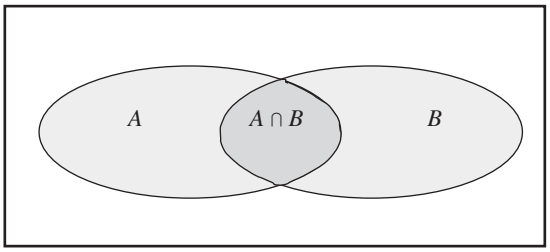


Figure 1.11: Probability of error bound for 8-PSK modulation.

Box 1.1: The Union Bound

For sets A and B, we have $P(A \cup B) = P(A) + P(B) - P(A \cap B)$.



Then, since $P(A \cap B) \geq 0$, clearly $P(A \cup B) \leq P(A) + P(B)$.

The event A occurs with the probability that the transmitted signal falls on the wrong side of the line L_1; similarly for B. Assuming that the noise is independent Gaussian with variance σ^2 in each coordinate direction, this probability is

$$P(A) = Q\left(\frac{d_{\min}}{2\sigma}\right),$$

where $d_{\min}$ is the minimum distance between signal points. Denote the probability of a symbol error by P_s. Assuming that all symbols are sent with equal probability, we have $P_s = \Pr(\text{symbol decoding error} | \mathbf{s}_0 \text{ sent})$, where the probability is bounded by

$$P_s \leq 2Q\left(\frac{d_{\min}}{2\sigma}\right). \tag{1.26}$$

The factor 2 multiplying the Q function is the number of nearest neighbors around each constellation point. The probability of error is dominated by the minimum distance between points: better performance is obtained with larger distance. As E_s/N_0 (the symbol SNR) increases, the probability of falling in the intersection region decreases and the bound (1.26) becomes increasingly tight. More generally, the probability of detection error for a symbol s which has K neighbors in signal space at a distance $d_{\min}$ from it can be bounded by

$$P_s \leq KQ\left(\frac{d_{\min}}{2\sigma}\right), \tag{1.27}$$

and the bound becomes increasingly tight as the SNR increases.

For signal constellations larger than BPSK, it is common to plot the probability of a *symbol* error *vs.* the SNR in E_s/N_0, where E_s is the average signal energy. However, when the bits are assigned in

Gray code order, then a symbol error is likely to be an adjacent symbol. One symbol thus results in 1 bit error, out of the $\log_2 M$ bits the symbol represents, so

$$P_b \approx \frac{1}{\log_2 M} P_s \quad \text{for sufficiently large SNR.} \tag{1.28}$$

1.5.6 The Binary Symmetric Channel

The BSC is a simplified channel model which contemplates only the transmission of bits over the channel; it does not treat details such as signal spaces, modulation, or matched filtering. The BSC accepts 1 bit per unit of time and transmits that bit with a probability of error p. A representation of the BSC is shown in Figure 1.12. An incoming bit of 0 or 1 is transmitted through the channel unchanged with probability $1 - p$, or flipped with probability p. The sequence of output bits in a BSC can be modeled as

$$R_i = S_i \oplus N_i, \tag{1.29}$$

where $R_i \in \{0, 1\}$ are the output bits, $S_i \in \{0, 1\}$ are the input bits, $N_i \in \{0, 1\}$ represents the possible bit errors, where N_i is 1 if an error occurs on bit i. The addition in (1.29) is **modulo 2 addition** (without carry), according to the addition table

$$0 \oplus 0 = 0 \quad 0 \oplus 1 = 1 \quad 1 \oplus 1 = 0,$$

so that if $N_i = 1$, then R_i is the bit complement of S_i. The BSC is an instance of a *memoryless channel*. This means each of the errors N_i is statistically independent of all the other N_i. The probability that bit i has an error is $P(N_i = 1) = p$, where p is called the BSC crossover probability. The sequence $\{N_i, i \in \mathbb{Z}\}$ can be viewed as an independent and identically distributed (i.i.d.) Bernoulli(p) random process.

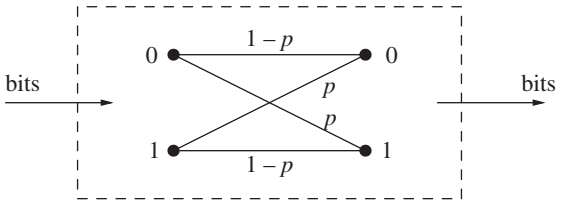


Figure 1.12: A binary symmetric channel.

Suppose that S is sent over the channel and R is received. The likelihood function $P(R|S)$ is

$$P(R|S) = \begin{cases} 1 - p & \text{if } R = S \\ p & \text{if } R \neq S. \end{cases} \tag{1.30}$$

Now suppose that the sequence $\mathbf{s} = [s_1, s_2, \ldots, s_n]$ is transmitted over a BSC and that the received sequence is $\mathbf{R} = [r_1, r_2, \ldots, r_n]$. Because of independent noise samples, the likelihood function factors,

$$P(\mathbf{R}|\mathbf{S}) = \prod_{i=1}^{n} P(R_i|S_i). \tag{1.31}$$

Each factor in the product is of the form (1.30). Thus, there is a factor $(1 - p)$ every time R_i agrees with S_i, and a factor p every time R_i differs from S_i. To represent this, we introduce the *Hamming distance*.

Definition 1.9 The **Hamming distance** between a sequence $\mathbf{x} = [x_1, x_2, \ldots, x_n]$ and a sequence $\mathbf{y} = [y_1, y_2, \ldots, y_n]$ is the number of positions that the corresponding elements differ:

$$\boxed{d_H(\mathbf{x}, \mathbf{y}) = \sum_{i=1}^{n} [x_i \neq y_i].} \tag{1.32}$$

Here we have used the notation (Iverson's convention [166])

$$[x_i \neq y_i] = \begin{cases} 1 & \text{if } x_i \neq y_i \\ 0 & \text{if } x_i = y_i. \end{cases}$$

□

Using the notation of Hamming distance, we can write the likelihood function (1.31) as

$$P(\mathbf{R}|\mathbf{S}) = \underbrace{(1-p)^{n-d_H(\mathbf{R},\mathbf{S})}}_{\substack{\text{number of places} \\ \text{they are the same}}} \underbrace{p^{d_H(\mathbf{R},\mathbf{S})}}_{\substack{\text{number of places} \\ \text{they differ}}}.$$

The likelihood function can also be written as

$$P(\mathbf{R}|\mathbf{S}) = \left(\frac{p}{1-p}\right)^{d_H(\mathbf{R},\mathbf{S})} (1-p)^n.$$

Consider now the detection problem of deciding if the sequence $\mathbf{S}_1$ or the sequence $\mathbf{S}_2$ was sent, where each occur with equal probability. The ML decision rule says to choose that value of $\mathbf{S}$ for which $\frac{p}{1-p}^{d_H(\mathbf{R},\mathbf{S})}(1-p)^n$ is the largest. Assuming that $p < \frac{1}{2}$, this corresponds to choosing that value of $\mathbf{S}$ for which $d_H(\mathbf{R}, \mathbf{S})$ is the smallest, that is, the vector $\mathbf{S}$ nearest to $\mathbf{R}$ in Hamming distance.

We see that for detection in a Gaussian channel, the *Euclidean* distance is the appropriate distance for detection. For the BSC, the *Hamming* distance is the appropriate distance for detection.

1.5.7 The BSC and the Gaussian Channel Model

At a sufficiently coarse level of detail, the modulator/demodulator system with the AWGN channel can be viewed as a BSC. The modulation, channel, and detector collectively constitute a "channel" which accepts bits at the input and emits bits at the output. The end-to-end system viewed at this level, as suggested by the dashed box in Figure 1.13(b), forms a BSC. The crossover probability p can be computed based on the system parameters,

$$p = P(\text{bit out} = 0|\text{bit in} = 1) = P(\text{bit out} = 1|\text{bit in} = 0) = P_b = Q(\sqrt{2E_b/N_0}).$$

In many cases the probability of error is computed using a BSC with an "internal" AWGN channel, so that the probability of error is produced as a function of E_b/N_0.

1.6 Memoryless Channels

A memoryless channel is one in which the output r_n at the nth symbol time depends only on the input at time n. Thus, given the input at time n, the output at time n is statistically independent of the outputs at other times. That is, for a sequence of received signals

$$\mathbf{R} = (R_1, R_2, \ldots, R_m)$$

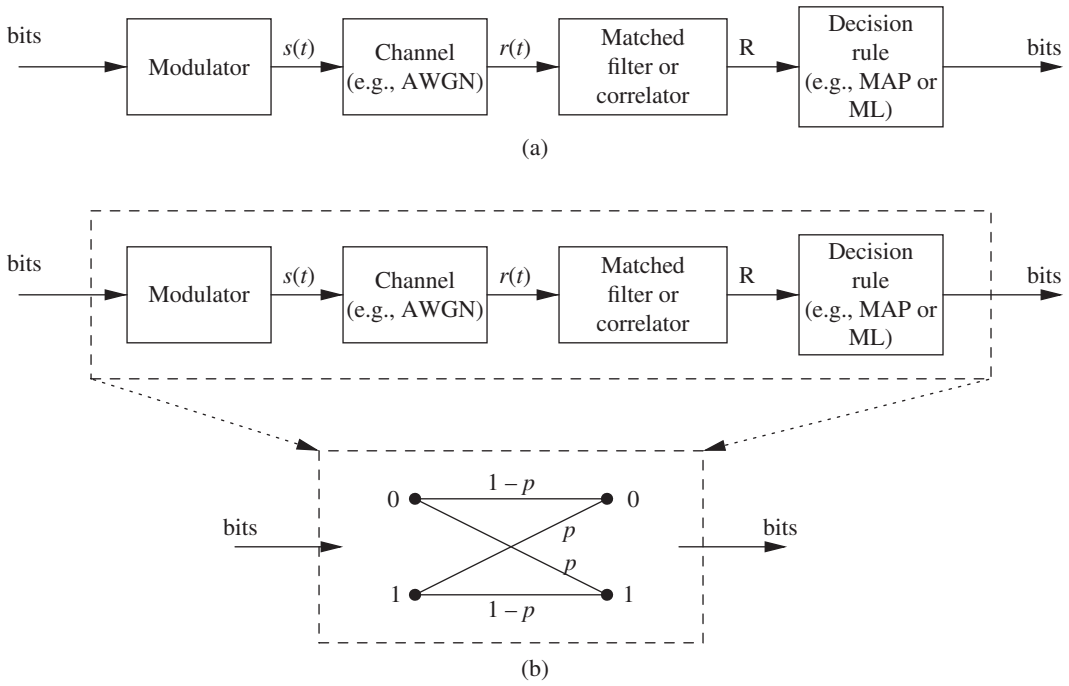


Figure 1.13: (a) System diagram showing modulation, channel, and demodulation; (b) BSC equivalent.

depending on transmitted signals $S_1, S_2, \ldots, S_m$, the likelihood function

$$p(R_1, R_2, \ldots, R_m | S_1, S_2, \ldots, S_m)$$

can be factored as

$$p(R_1, R_2, \ldots, R_m | S_1, S_2, \ldots, S_m) = \prod_{i=1}^{m} p(R_i | S_i).$$

Both the additive Gaussian channel and the BSC that have been introduced are memoryless channels. We will almost universally assume that the channels are memoryless channels. The bursty channels discussed in Chapter 10 and the convolutive channel introduced in Chapter 14 are exceptions to this.

1.7 Simulation and Energy Considerations for Coded Signals

In channel coding with rate $R = k/n$, k input bits yield n coded bits at the output of the encoder, where $n > k$. A transmission budget which allocates E_b Joules/bit for the uncoded data must spread that energy over more coded bits. Let

$$E_c = RE_b$$

denote the "energy per coded bit." We thus have $E_c < E_b$. Consider the framework shown in Figure 1.14. From point "a" to point "b," there is conventional (uncoded) BPSK modulation passing through an AWGN channel, except that the energy per bit is E_c, so that the amplitudes in the BPSK modulator are $\pm\sqrt{E_c}$, instead of $\pm\sqrt{E_b}$, as they would be in the uncoded case.

Thus, at point "b" the probability of error of the coded bits appearing at point "a" can be computed as

$$P_{b,\text{coded}} = Q\left(\sqrt{\frac{2E_c}{N_0}}\right).$$

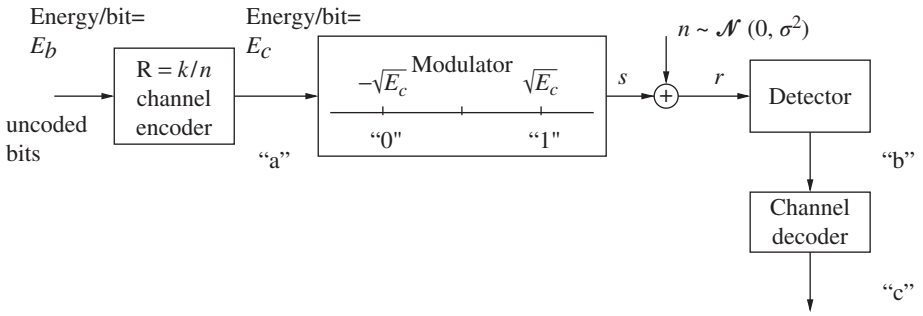


Figure 1.14: Energy for a coded signal.

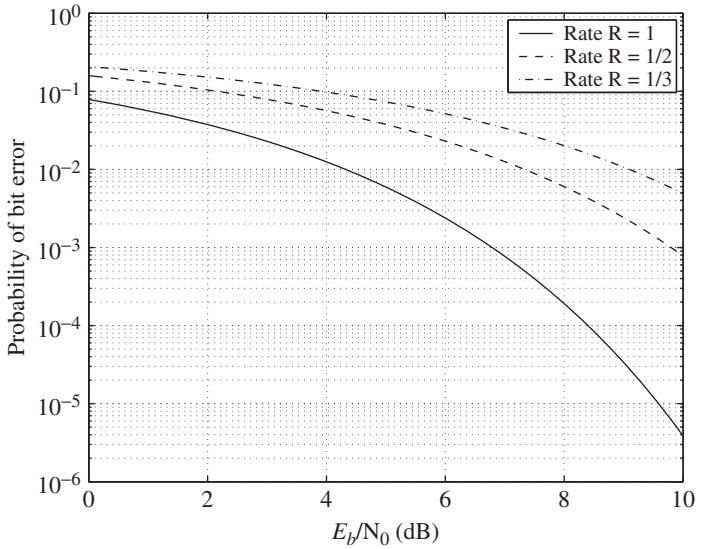


Figure 1.15: Probability of error for coded bits, before correction.

Since $E_c < E_b$, this is *worse* performance than uncoded BPSK would have had. Figure 1.15 shows the probability of error of coded bits for $R = 1/2$ and $R = 1/3$ error correction codes at point "b" in Figure 1.14. At the receiver, the detected coded bits are passed to the channel decoder, the error correction stage, which attempts to correct errors. Clearly, in order to be of any value, the code must be strong enough so that the bits emerging at point "c" of Figure 1.14 can compensate for the lower energy per bit in the channel, plus correct other errors. Fortunately, we will see that this is in fact the case.

Now consider how this system might be simulated in software. It is common to simulate the modulator at point "a" of Figure 1.14 as having fixed amplitudes and to adjust the variance σ^2 of the noise n in the channel. One of the primary considerations, therefore, is how to set σ^2.

Frequently it is desired to simulate performance at a particular SNR, expressed in dB, denoted as $(E_b/N_0)_{\text{dB}}$). Let $\gamma = E_b/N_0$ denote the desired SNR at which to simulate. The SNR in dB is converted to γ by

$$\gamma = 10^{(\text{SNR})_{\text{dB}}/10}$$

Recalling that $\sigma^2 = N_0/2$, and knowing γ, we have

$$\gamma = \frac{E_b}{2\sigma^2},$$

so

$$\sigma^2 = \frac{E_b}{2\gamma}.$$

Since $E_b = E_c/R$, we have

$$\sigma^2 = \frac{E_c}{2R\gamma}.$$

It is also common in simulation to normalize so that the simulated signal amplitude is $E_c = 1$.

1.8 Some Important Definitions and a Trivial Code: Repetition Coding

In this section, we introduce the important coding concepts of code rate, Hamming distance, minimum distance, Hamming spheres, and the generator matrix. These concepts are introduced by means of a simple, even trivial, example of an error correction code, the repetition code.

Let $\mathbb{F}_2$ denote the set (field) with two elements in it, 0 and 1. In this field, arithmetic operations are defined as:

$$0+0=0 \quad 0+1=1 \quad 1+0=1 \quad 1+1=0$$

$$0\cdot 0=0 \quad 0\cdot 1=0 \quad 1\cdot 0=0 \quad 1\cdot 1=1.$$

Let $\mathbb{F}_2^n$ denote the (vector) space of n-tuples of elements of $\mathbb{F}_2$.

Suppressing for the moment a few details, here is a useful definition: an (n, k) binary code is a set of 2^k distinct points in $\mathbb{F}_2^n$. Another way of putting this: an (n, k) binary code is a code that accepts k bits as input and produces n bits as output. (We will be interested in linear codes, as the details unfold.)

Definition 1.10 The **rate** of an (n, k) code is

$$R = \frac{k}{n}.$$

□

The $(n, 1)$ **repetition code**, where n is odd, is the code obtained by repeating the 1-bit input n times in the output codeword. That is, the codeword representing the input 0 is a block of n 0s and the codeword representing the input 1 is a block of n 1s. The *code* $\mathcal{C}$ consists of the set of two codewords

$$\mathcal{C} = \{[0, 0, \ldots, 0], [1, 1, \ldots, 1]\} \subset \mathbb{F}_2^n.$$

Letting m denote the message, the corresponding codeword is

$$\mathbf{c} = \underbrace{[m, m, m, \ldots, m]}_{n \text{ copies}}.$$

This is a rate $R = 1/n$ code.

Encoding can be represented as a matrix operation. Let G be the $1 \times n$ **generator matrix** given by

$$G = \begin{bmatrix} 1 & 1 & \cdots & 1 \end{bmatrix}.$$

Then the encoding operation is

$$\mathbf{c} = mG.$$

1.8.1 Detection of Repetition Codes Over a BSC

Let us first consider decoding of this code when transmitted through a BSC with crossover probability $p < 1/2$. Denote the output of the BSC by

$$\mathbf{r} = \mathbf{c} \oplus \mathbf{n},$$

where the addition is modulo 2 and $\mathbf{n}$ is a binary vector of length n, with 1 in the positions where the channel errors occur. Assuming that the codewords are selected with equal probability, ML decoding is appropriate. As observed in Section 1.5.6, the ML decoding rule selects the codeword in C which is closest to the received vector $\mathbf{r}$ in Hamming distance. For the repetition code, this decoding rule can be expressed as a majority decoding rule: if the majority of received bits are 0, decode a 0; otherwise, decode a 1. For example, take the $(7, 1)$ repetition code and let $m = 1$. Then the codeword is $\mathbf{c} = [1, 1, 1, 1, 1, 1, 1]$. Suppose that the received vector is

$$\mathbf{r} = [1, 0, 1, 1, 0, 1, 1].$$

Since 5 out of the 7 bits are 1, the decoded value is

$$\hat{m} = 1.$$

An error *detector* can also be established. If the received vector $\mathbf{r}$ is not one of the codewords, we *detect* that the channel has introduced one or more errors into the transmitted codeword.

The codewords in a code C can be viewed as points in n-dimensional space. For example, Figure 1.16(a) illustrates the codewords as points $(0, 0, 0)$ and $(1, 1, 1)$ in three-dimensional space. (Beyond three dimensions, of course, the geometric viewpoint cannot be plotted, but it is still valuable conceptually.) In this geometric setting, we use the **Hamming distance** to measure distances between points.

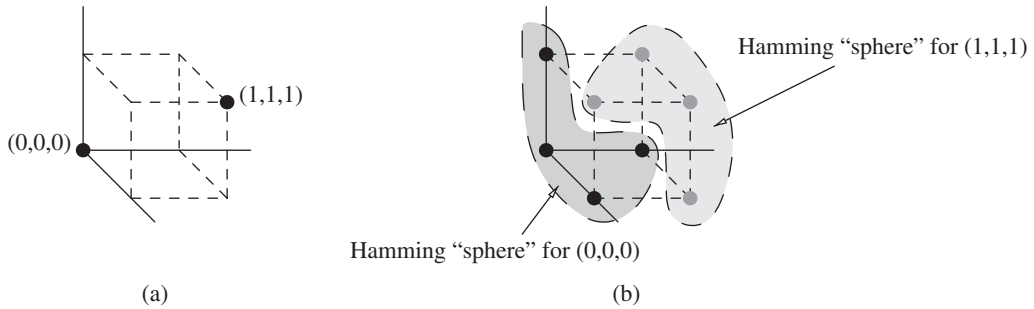


Figure 1.16: A (3, 1) binary repetition code. (a) The code as points in space; (b) the Hamming spheres around the points.

Definition 1.11 The **minimum distance** $d_{\min}$ of a code C is the smallest Hamming distance between any two codewords in the code:

$$d_{\min} = \min_{\mathbf{c}_i, \mathbf{c}_j \in C, \mathbf{c}_i \neq \mathbf{c}_j} d_H(\mathbf{c}_i, \mathbf{c}_j).$$

□

The two codewords in the $(n, 1)$ repetition code are clearly (Hamming) distance n apart.

In this geometric setting, the ML decoding algorithm may be expressed as: Choose the codeword $\hat{\mathbf{c}}$ which is closest to the received vector $\mathbf{r}$. That is,

$$\hat{\mathbf{c}} = \arg \min_{\mathbf{c} \in C} d_H(\mathbf{r}, \mathbf{c}).$$

Definition 1.12 The **Hamming sphere** of radius t around a codeword $\mathbf{c}$ consists of all vectors which are at a Hamming distance $\leq t$ from $\mathbf{c}$. □

For example, for the $(3, 1)$ repetition code, the codewords and the points in their Hamming spheres are

Codeword	Points in its sphere
(0,0,0)	(0,0,0),(0,0,1),(0,1,0),(1,0,0)
(1,1,1)	(1,1,1),(1,1,0),(1,0,1),(0,1,1),

as illustrated in Figure 1.16(b).

When the Hamming spheres around each codeword are all taken to have the same radius, the largest such radius producing nonoverlapping spheres is determined by the separation between the *nearest* two codewords in the code, $d_{\min}$. The radius of the spheres in this case is $t = \lfloor (d_{\min} - 1)/2 \rfloor$, where the notation $\lfloor x \rfloor$ means to take the greatest integer $\leq x$. Figure 1.17 shows the idea of these Hamming spheres. The black squares represent codewords in n-dimensional space and black dots represent other vectors in n-dimensional space. The dashed lines indicate the boundaries of the Hamming spheres around the codewords. If a vector $\mathbf{r}$ falls inside the sphere around a codeword, then it is closer to that codeword than to any other codeword. By the ML criterion, $\mathbf{r}$ should decode to that codeword inside the sphere. When all the spheres have radius $t = \lfloor (d_{\min} - 1)/2 \rfloor$, this decoding rule is referred to as bounded distance decoding.

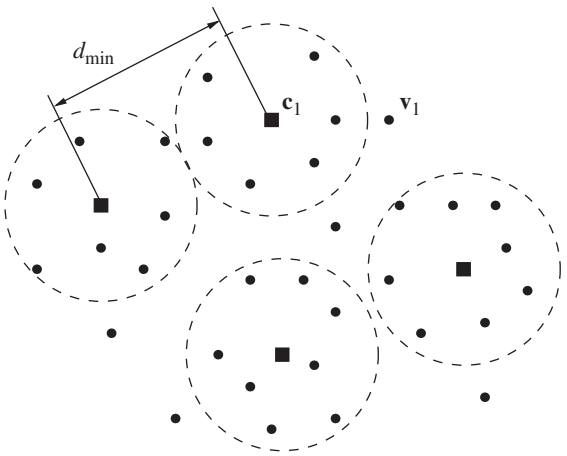


Figure 1.17: A representation of decoding spheres.

The decoder will make a decoding error if the channel noise moves the received vector $\mathbf{r}$ into a sphere other than the sphere the true codeword is in. Since the centers of the spheres lie a distance at least $d_{\min}$ apart, the decoder is guaranteed to *decode* correctly provided that no more than t errors occur in the received vector $\mathbf{r}$. The number t is called the **random error correction capability** of the code. If $d_{\min}$ is even and two codewords lie exactly $d_{\min}$ apart and the channel introduces $d_{\min}/2$ errors, then the received vector lies right on the boundary of two spheres. In this case, given no other information, the decoder must choose one of the two codewords arbitrarily; half the time it will make an error.

Note from Figure 1.17 that in a bounded distance decoder there may be vectors that fall outside the Hamming spheres around the codewords, such as the vector labeled $\mathbf{v}_1$. If the received vector $\mathbf{r} = \mathbf{v}_1$, then the nearest codeword is $\mathbf{c}_1$. A bounded distance decoder, however, would not be able to decode if $\mathbf{r} = \mathbf{v}_1$, since it can only decode those vectors that fall in spheres of radius t. The decoder might have to declare a decoding failure in this case.

A true ML decoder, which chooses the nearest codeword to the received vector, would be able to decode. Unfortunately, ML decoding is computationally very difficult for large codes. Most of the algebraic decoding algorithms in this book are only bounded distance decoders. An interesting

exception are the decoders presented in Chapters 7 and 11, which actually produce *lists* of codeword candidates. These decoders are called *list decoders*.

If the channel introduces fewer than $d_{\min}$ errors, then these can be *detected*, since $\mathbf{r}$ cannot be another codeword in this case. In summary, for a code with minimum distance $d_{\min}$:

$$\begin{array}{rc} \text{Guaranteed error correction capability:} & t = \lfloor (d_{\min} - 1)/2 \rfloor \\ \text{Guaranteed error detection capability:} & d_{\min} - 1 \end{array}$$

Having defined the repetition code, let us now characterize its probability of error performance as a function of the BSC crossover probability p. For the $(n, 1)$ repetition code, $d_{\min} = n$, and $t = (n-1)/2$ (remember n is odd). Suppose in particular that $n = 3$, so that $t = 1$. Then the decoder will make an error if the channel causes either 2 or 3 bits to be in error. Using P_e^n to denote the probability of decoding error for a code of length n, we have

$$\begin{aligned} P_e^3 &= \text{Prob(2 channel errors)} + \text{Prob(3 channel errors)} \\ &= 3p^2(1-p) + p^3 = 3p^2 - 2p^3. \end{aligned}$$

If $p < \frac{1}{2}$, then $P_e^3 < p$, that is, the decoder will have fewer errors than using the channel without coding.

Let us now examine the probability of decoding error for a code of length n. Note that it doesn't matter what the transmitted codeword was; the probability of error depends only on the error introduced by the channel. Clearly, the decoder will make an error if more than half of the received bits are in error. More precisely, if more than t bits are in error, the decoder will make an error. The probability of error can be expressed as

$$P_e^n = \sum_{i=t+1}^{n} \text{Prob}(i \text{ channel errors occur out of } n \text{ transmitted bits}).$$

The probability of exactly i bits in error out of n bits, where each bit is drawn at random with probability p is[6]

$$\binom{n}{i} p^i (1-p)^{n-i},$$

so that

$$\begin{aligned} P_e^n &= \sum_{i=t+1}^{n} \binom{n}{i} p^i (1-p)^{n-i} \\ &= \binom{n}{t+1} (1-p)^n \left(\frac{p}{1-p} \right)^{t+1} + \text{terms of higher degree in } p. \end{aligned}$$

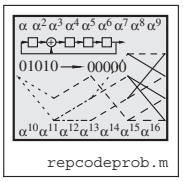
It would appear that as the code length increases, and thus t increases, the probability of decoder error decreases. (This is substantiated in Exercise 1.16b.) Thus, it is possible to obtain *arbitrarily small* probability of error, but at the cost of a very low rate: $R = 1/n \to 0$ as $P_e^N \to 0$.

Let us now consider using this repetition code for communication over the AWGN channel. Let us suppose that the transmitter has $P = 1$ Watt (W) of power available and that we want to send information at 1 bit/second. There is thus $E_b = 1$ Joule (J) of energy available for each bit of information. Now the information is coded using an $(n, 1)$ repetition code. To maintain the information rate of 1 bit/second, we must send n coded bits/second. With n times as many bits to send, there is still only 1 W of power available, which must be shared among all the coded bits. The energy available for each coded bit, which we denote as E_c, is $E_c = E_b/n$. Thus, because of coding, there is less energy available for each bit to convey information! The probability of error for the AWGN channel (i.e., the

[6] The binomial coefficient is $\binom{n}{i} = \frac{n!}{i!(n-i)!}$.

binary crossover probability for the effective BSC) is

$$p = Q(\sqrt{2E_c/N_0}) = Q(\sqrt{2E_b/nN_0}).$$



The crossover probability p is higher as a result of using a code! However, the hope is that the error decoding capability of the overall system is better. Nevertheless, for the repetition code, this hope is in vain.

Figure 1.18 shows the probability of error for repetition codes (here, consider only the hard-decision decoding). The coded performance is worse than the uncoded performance, and the performance gets worse with increasing n.

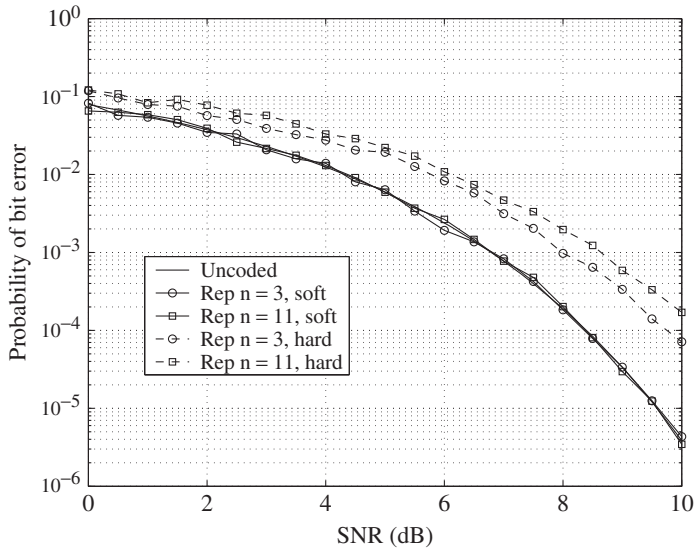


Figure 1.18: Performance of the $(3, 1)$ and $(11, 1)$ repetition code over BSC using both hard- and soft-decision decoding.

1.8.2 Soft-Decision Decoding of Repetition Codes Over the AWGN

Let us now consider decoding over the AWGN using a *soft-decision* decoder. Since the repetition code has a particularly simple codeword structure, it is straightforward to describe the soft-decision decoder and characterize its probability of error.

The likelihood function is

$$p(\mathbf{r}|\mathbf{c}) = \prod_{i=1}^{n} p(r_i|c_i),$$

so that the log likelihood ratio

$$\Lambda(\mathbf{r}) = \log \frac{p(\mathbf{r}|m=1)}{p(\mathbf{r}|m=0)}$$

can be computed using (1.21) as

$$\Lambda(\mathbf{r}) = L_c \sum_{i=1}^{n} r_i.$$

Then the decoder decides $\hat{m} = 1$ if $\Lambda(\mathbf{r}) > 0$, or $\hat{m} = 0$ if $\Lambda(\mathbf{r}) < 0$. Since the threshold is 0 and L_c is a positive constant, the decoder decides

$$\hat{m} = \begin{cases} 1 & \text{if } \sum_{i=1}^{n} r_i > 0 \\ 0 & \text{if } \sum_{i=1}^{n} r_i < 0. \end{cases}$$

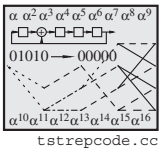
The soft-decision decoder performs superior to the hard-decision decoder. Suppose the vector $(-\sqrt{E_c}, -\sqrt{E_c}, \ldots, -\sqrt{E_c})$ is sent (corresponding to the all-zero codeword). If one of the r_i happens to be greater than 0, but other of the r_i are correspondingly less than 0, the erroneous positive quantities might be canceled out by the other symbols. In fact, it is straightforward to show (see Exercise 1.18) that the probability of error for the $(n, 1)$ repetition code with soft-decision decoding is

$$P_b = Q(\sqrt{2E_b/N_0}). \tag{1.33}$$

That is, it is the same as for uncoded transmission — still not effective as a code, but better than hard-decision decoding.

While these simple-minded repetition codes do not, by themselves, make good error correction codes, repetition codes will be shown in Chapter 15 to be a component of very powerful codes, the repeat accumulate codes.

1.8.3 Simulation of Results



tstrepcode.cc
repcoderes.m

While it is possible for these simple codes to compute explicit performance curves, it is worthwhile to consider how the performance might also be simulated, since other codes that we will examine may be more difficult to analyze. The program here illustrates a framework for simulating the performance of codes. The probability of error is estimated by running codewords through a simulated Gaussian channel until a specified number of errors has occurred. Then the estimated probability of error is the number of errors counted divided by the number of bits generated.

Figure 1.18 shows the probability of error for uncoded transmission and both hard- and soft-decision decoding of (3, 1) and (11, 1) codes.

1.8.4 Summary

This lengthy example on a nearly useless code has introduced several concepts that will be useful for other codes:

- The concept of minimum distance of a code.
- The probability of decoder error.
- The idea of a generator matrix.
- The fact that not every code is good!
- Recognition that soft-decision decoding is superior to hard-input decoding in terms of probability of error.

Prior to the proof of Shannon's channel coding theorem and the research it engendered, communication engineers were in a quandary. It was believed that to obtain totally reliable communication, it would be necessary to transmit using very slow rates, essentially employing repetition codes to catch any errors and using slow symbol rate to increase the energy per bit. However, Shannon's theorem dramatically changed this perspective, indicating that it is not necessary to slow the rate of communication to zero. It is only necessary to use better codes. It is hardly an overstatement that information-theoretic understanding has driven the information age!

1.9 Hamming Codes

As a second example we now introduce are Hamming codes. These are codes which are much better than repetition codes and were the first important codes discovered. Hamming codes lie at the intersection of many different kinds of codes, so we will also use them to briefly introduce

several important themes which will be developed throughout the course of this book. These themes include: generator matrices; parity check matrices; syndromes; generator polynomials; parity check polynomials; trellises; and Tanner graphs. These themes, touched on here, will be fully developed in later chapters.

A $(7, 4)$ Hamming code produces $n = 7$ bits of output for every $k = 4$ bits of input. Hamming codes are *linear block codes*, which means that the encoding operation can be described in terms of a 4×7 generator matrix, such as

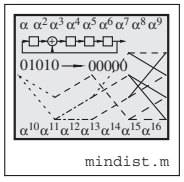
$$G = \begin{bmatrix} 1 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 1 \end{bmatrix}. \tag{1.34}$$

The codewords are obtained as linear combination of the *rows* of G, where all the operations are computed modulo 2 in each vector element, that is, in the field $\mathbb{F}_2$. The code is the *row space* of G. For a message vector $\mathbf{m} = [m_1, m_2, m_3, m_4]$ the codeword is

$$\mathbf{c} = \mathbf{m}G.$$

For example, if $\mathbf{m} = [1, 1, 0, 0]$, then

$$\mathbf{c} = \mathbf{m}G = [1, 1, 0, 1, 0, 0, 0] \oplus [0, 1, 1, 0, 1, 0, 0] = [1, 0, 1, 1, 1, 0, 0].$$



It can be verified that the minimum distance of the Hamming code is $d_{\min} = 3$, so the code is capable of correcting 1 error in every block of n bits.

The codewords for this code are

$$\begin{array}{l} [0,0,0,0,0,0,0],[1,1,0,1,0,0,0],[0,1,1,0,1,0,0],[1,0,1,1,1,0,0] \\ [0,0,1,1,0,1,0],[1,1,1,0,0,1,0],[0,1,0,1,1,1,0],[1,0,0,0,1,1,0] \\ [0,0,0,1,1,0,1],[1,1,0,0,1,0,1],[0,1,1,1,0,0,1],[1,0,1,0,0,0,1] \\ [0,0,1,0,1,1,1],[1,1,1,1,1,1,1],[0,1,0,0,0,1,1],[1,0,0,1,0,1,1]. \end{array} \tag{1.35}$$

The Hamming decoding algorithm presented here is slightly more complicated than for the repetition code. (There are other decoding algorithms.)

Every (n, k) linear block code has associated with it a $(n - k) \times n$ matrix H called the *parity check matrix*, which has the property that $GH^T = \mathbf{0}$ (all the operations are performed modulo 2). This means that a vector $\mathbf{v} \in \mathbb{F}_2^n$ (a vector of binary elements)

$$\mathbf{v}H^T = \mathbf{0} \text{ if and only if the vector } \mathbf{v} \text{ is a codeword.} \tag{1.36}$$

The parity check matrix is not unique. For the generator G of (1.34), the parity check matrix can be written as

$$H = \begin{bmatrix} 1 & 0 & 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 & 1 \end{bmatrix}. \tag{1.37}$$

The matrix H can be expressed in terms of its columns as

$$H = \begin{bmatrix} \mathbf{h}_1 & \mathbf{h}_2 & \mathbf{h}_3 & \mathbf{h}_4 & \mathbf{h}_5 & \mathbf{h}_6 & \mathbf{h}_7 \end{bmatrix}.$$

It may be observed that (for the Hamming code) the columns of H consist of the binary representations of the numbers 1 through $7 = n$, though not in numerical order. On the basis of this observation, we can generalize to other Hamming codes. Hamming codes of length $n = 2^m - 1$ and dimension $k = 2^m - m - 1$ exist for every $m \geq 2$, having parity check matrices whose columns are binary representations of the numbers from 1 through n.

1.9.1 Hard-Input Decoding Hamming Codes

Suppose that a codeword $\mathbf{c}$ is sent and the vector through a BSC is

$$\mathbf{r} = \mathbf{c} \oplus \mathbf{n} \text{ (addition modulo 2).}$$

The first decoding step is to compute the *syndrome*

$$\mathbf{s} = \mathbf{r}H^T = (\mathbf{c} \oplus \mathbf{n})H^T = \mathbf{n}H^T$$

(all operations in $\mathbb{F}_2$). Because of Property (1.36), the syndrome depends only on the error $\mathbf{n}$ and not on the transmitted codeword. The codeword information is "projected away."

Since a Hamming code is capable of correcting only a single error, suppose that $\mathbf{n}$ is all zeros except at a single position,

$$\mathbf{n} = [n_1, n_2, n_3, \ldots, n_7] = [0, \ldots, 0, 1, 0, \ldots, 0],$$

where the 1 is equal to n_i. (That is, the error is in the ith position.)

Let us write H^T in terms of its rows:

$$H^T = \begin{bmatrix} \mathbf{h}_1^T \\ \mathbf{h}_2^T \\ \vdots \\ \mathbf{h}_n^T \end{bmatrix}.$$

Then the syndrome is

$$\mathbf{s} = \mathbf{r}H^T = \mathbf{n}H^T = \begin{bmatrix} n_1 & n_2 & \ldots & n_n \end{bmatrix} \begin{bmatrix} \mathbf{h}_1^T \\ \mathbf{h}_2^T \\ \vdots \\ \mathbf{h}_n^T \end{bmatrix} = \mathbf{h}_i^T.$$

The error position i is the column i of H that is equal to the (transpose of) the syndrome $\mathbf{s}^T$.

Algorithm 1.1 Hamming Code Decoding

1. For the received binary vector $\mathbf{r}$, compute the syndrome $\mathbf{s} = \mathbf{r}H^T$. If $\mathbf{s} = \mathbf{0}$, then the decoded codeword is $\hat{\mathbf{c}} = \mathbf{r}$.
2. If $\mathbf{s} \neq \mathbf{0}$, then let i denote the column of H which is equal to $\mathbf{s}^T$. There is an error in position i of $\mathbf{r}$. The decoded codeword is $\hat{\mathbf{c}} = \mathbf{r} + \mathbf{n}_i$, where $\mathbf{n}_i$ is a vector which is all zeros except for a 1 in the ith position.

This decoding procedure fails if more than one error occurs.

Example 1.13 Suppose that the message

$$\mathbf{m} = [m_1, m_2, m_3, m_4] = [0, 1, 1, 0]$$

is encoded, resulting in the codeword

$$\mathbf{c} = [0, 1, 1, 0, 1, 0, 0] + [0, 0, 1, 1, 0, 1, 0] = [0, 1, 0, 1, 1, 1, 0].$$

When $\mathbf{c}$ is transmitted over a BSC, the vector

$$\mathbf{r} = [0, 1, 1, 1, 1, 1, 0]$$

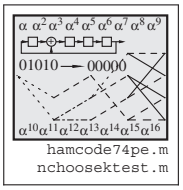
is received. The decoding algorithm proceeds as follows:

1. The syndrome $\mathbf{s} = [0, 1, 1, 1, 1, 1, 0]H^T = [1, 0, 1]$ is computed.
2. This syndrome corresponds to column 3 of H. The decoded value is therefore

$$\hat{\mathbf{c}} = \mathbf{r} + [0, 0, 1, 0, 0, 0, 0] = [0, 1, 0, 1, 1, 1, 0],$$

which is the transmitted codeword.

□



The expression for the probability of bit error is significantly more complicated for Hamming codes than for repetition codes. We defer on the details of these computations to the appropriate location (Section 3.7) and simply plot the results here. The available energy per encoded bit is

$$E_c = E_b(k/n) = 4E_b/7,$$

so, as for the repetition code, there is less energy available per bit. This represents a loss of $10\log_{10}(4/7) = -2.4$ dB of energy per transmitted bit compared to the uncoded system. The decrease in energy per bit is not as great as for the repetition code, since the rate is higher. Figure 1.19 shows the probability of bit error for uncoded channels (the solid line), and for the coded bits — that is, the bits coded with energy E_c per bit — (the dashed line). The figure also shows the probability of bit error for the bits *after* they have been through the decoder (the dash-dot line). In this case, the decoded bits *do* have a lower probability of error than the uncoded bits. For the uncoded system, to achieve a probability of error of $P_b = 10^{-6}$ requires an SNR of 10.5 dB, while for the coded system, the same probability of error is achieved with 10.05 dB. The code was able to overcome the 2.4 dB of loss due to rate, and add another 0.45 dB of improvement. We say that the *coding gain* of the system (operated near 10 dB) is 0.45 dB: we can achieve the same performance as a system expending 10.5 dB SNR per bit, but with only 10.05 dB of expended transmitter energy per bit.

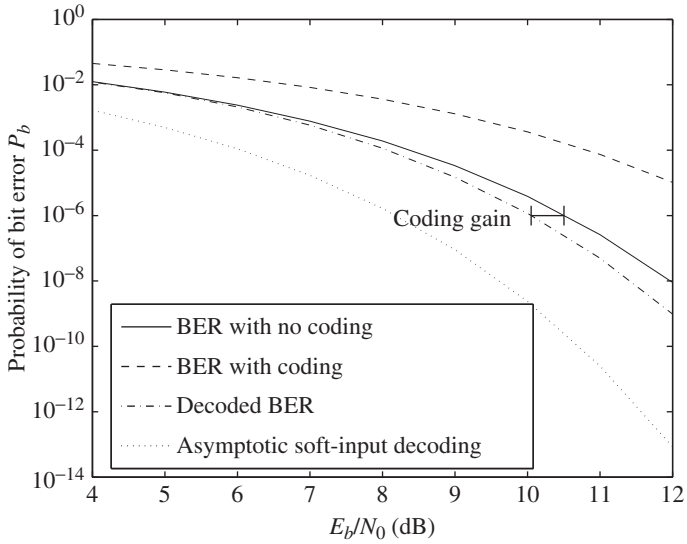


Figure 1.19: Performance of the $(7, 4)$ Hamming code in the AWGN channel.

Also shown in Figure 1.19 is the asymptotic (most accurate for large SNR) performance of soft-decision decoding. This is somewhat optimistic, having better performance than might be achieved in practice. But it does show the potential that soft-decision decoding has: it is significantly better than the hard-input decoding.

1.9.2 Other Representations of the Hamming Code

In the brief introduction to the Hamming code, we showed that the encoding and decoding operations have matrix representations. This is because Hamming codes are **linear block codes**, which will be explored in Chapter 3. There are other representations for Hamming and other codes. We briefly introduce these here as bait and lead-in to further chapters. As these representations show, descriptions of codes involve algebra, polynomials, graph theory, and algorithms on graphs, in addition to the linear algebra we have already seen.

1.9.2.1 An Algebraic Representation

The columns of the parity check matrix H can be represented using a symbol for each column. For example, we could write

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

as

$$H = \begin{bmatrix} \boldsymbol{\beta}_1 & \boldsymbol{\beta}_2 & \boldsymbol{\beta}_3 & \boldsymbol{\beta}_4 & \boldsymbol{\beta}_5 & \boldsymbol{\beta}_6 & \boldsymbol{\beta}_7 \end{bmatrix},$$

where each β_i represents a 3-tuple. Then the syndrome $\mathbf{s} = \mathbf{r}H^T$ can be represented as

$$\mathbf{s} = \sum_{i=1}^{7} r_i \boldsymbol{\beta}_i.$$

Then $\mathbf{s} = \boldsymbol{\beta}_j$ for some j, which indicates the column where the error occurred. This turns the decoding problem into a straightforward algebra problem.

A question we shall take up later is how to generalize this operation. That is, can codes be defined which are capable of correcting more than a single error, for which finding the errors can be computed using algebra? In order to explore this question, we will need to carefully define how to perform algebra on discrete objects (such as the columns of H) so that addition, subtraction, multiplication, and division are defined in a meaningful way. Such algebraic operations are defined in Chapters 2 and 5.

1.9.2.2 A Polynomial Representation

Examination of the codewords in (1.35) reveals an interesting fact: if $\mathbf{c}$ is a codeword, then so is every cyclic shift of $\mathbf{c}$. For example, the codeword $[1, 1, 0, 1, 0, 0, 0]$ has the cyclic shifts

$$[0, 1, 1, 0, 1, 0, 0], [0, 0, 1, 1, 0, 1, 0], [0, 0, 0, 1, 1, 0, 1], [1, 0, 0, 0, 1, 1, 0],$$

$$[0, 1, 0, 0, 0, 1, 1], [1, 0, 1, 0, 0, 0, 1],$$

which are also codewords. Codes for which all cyclic shifts of every codeword are also codewords are called **cyclic** codes. As we will find in Chapter 4, Hamming codes, like most block codes of modern interest, are cyclic codes. In addition to the representation using a generator matrix, cyclic codes can also be represented using polynomials. For the $(7, 4)$ Hamming code, there is a *generator polynomial* $g(x) = x^3 + x + 1$ and a corresponding *parity-check polynomial* $h(x) = x^4 + x^2 + x + 1$, which is a polynomial such that $h(x)g(x) = x^7 + 1$. In general, for a cyclic code of length n, a generator polynomial $g(x)$ must divide $x^n + 1$ (without remainder, operations in $\mathbb{F}_2$), and the quotient is the parity check polynomial,

$$h(x) = \frac{x^n + 1}{g(x)},$$

where the operations are done in $\mathbb{F}_2$. The encoding operation can be represented using polynomial multiplication (with coefficient operations modulo 2). For this reason, the study of polynomial operations and the study of algebraic objects built out of polynomials is of great interest. The parity

check polynomial can be used to check if a polynomial is a code polynomial: A polynomial $r(x)$ is a code polynomial if and only if $r(x)h(x)$ is a multiple of $x^n + 1$, so that

$$\frac{r(x)h(x)}{x^n + 1}$$

divides with remainder 0. This provides a parity check condition: for this Hamming code ($n = 7$) compute $r(x)h(x)$ modulo $x^7 + 1$. If this is not equal to 0, then $r(x)$ is not a code polynomial.

Example 1.14 The message $\mathbf{m} = [m_0, m_1, m_2, m_3] = [0, 1, 1, 0]$ can be represented as a polynomial as

$$m(x) = m_0 + m_1 x + m_2 x^2 + m_3 x^3 = 0 \cdot 1 + 1 \cdot x + 1 \cdot x^2 + 0 \cdot x^3 = x + x^2.$$

The code polynomial is obtained by $c(x) = m(x)g(x)$, or

$$\begin{aligned} c(x) &= (x + x^2)(1 + x + x^3) = (x + x^2 + x^4) + (x^2 + x^3 + x^5) \\ &= x + 2x^2 + x^3 + x^4 + x^5 = x + x^3 + x^4 + x^5, \end{aligned}$$

(where $2x^2 = 0$ modulo 2), which corresponds to the code vector $\mathbf{c} = [0, 1, 0, 1, 1, 1, 0]$. □

1.9.2.3 A Trellis Representation

As we will see in Chapter 12, there is a graph associated with a block code. This graph is called the *Wolf trellis* for the code. We shall see that paths through the graph correspond to vectors $\mathbf{v}$ that satisfy the parity check condition $\mathbf{v}H^T = \mathbf{0}$. For example, Figure 1.20 shows the trellis corresponding to the parity check matrix (1.37). The trellis states at the kth stage are obtained by taking all possible binary linear combinations of the first k columns of H. In Chapter 12, we will develop a decoding algorithm which essentially finds the best path through the graph. One such decoding algorithm is called the *Viterbi algorithm*. Such decoding algorithms will allow us to create soft-decision decoding algorithms for block codes.

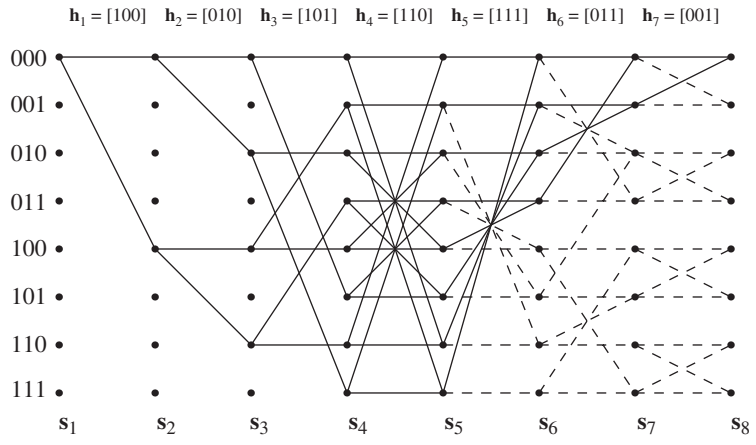


Figure 1.20: The trellis of a $(7, 4)$ Hamming code.

The Viterbi algorithm is also used for decoding codes which are defined using graphs similar to that of Figure 1.20. Such codes are called *convolutional codes*.

1.9.2.4 The Tanner Graph Representation

Every linear block code also has another graph which represents it called the *Tanner graph*. For a parity check matrix, the Tanner graph has one node to represent each column of H (the "bit nodes," denoted by c_i in the figure) and one node to represent each row of H (the "check nodes," denoted by z_i in the figure). Edges occur only between bit nodes and check nodes. There is an edge between a bit node and a check node if there is a 1 in the parity check matrix at the corresponding location. For example, for the parity check matrix of (1.37), the Tanner graph representation is shown in Figure 1.21. Algorithms to be presented in Chapter 15 describe how to propagate information through the graph in order to perform decoding. These algorithms are usually associated with codes which are iteratively decoded, such as turbo codes and low-density parity-check codes. These modern families of codes have very good behavior, sometimes nearly approaching capacity.

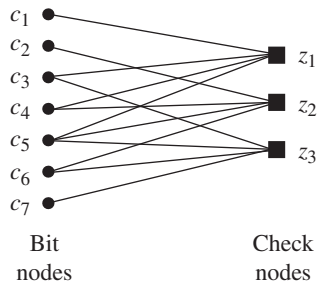


Figure 1.21: The Tanner graph for a $(7, 4)$ Hamming code.

1.10 The Basic Questions

The two simple codes we have examined so far bring out issues relevant for the codes we will investigate:

1. How is the code described and represented?
2. How is encoding accomplished?
3. How is decoding accomplished? (This frequently takes some cleverness!)
4. How are codewords to be represented, encoded, and decoded, in a computationally tractable way?
5. What is the performance of the code? What are the properties of the code? (e.g., How many codewords? What are the weights of the codewords?)
6. Are there other families of codes which can provide better coding gains?
7. How can these codes be found and described?
8. Are there constraints on allowable values of n, k, and $d_{\min}$?
9. Is there some limit to the amount of coding gain possible?
10. For a given available SNR, is there a lower limit on the probability of error that can be achieved?

Questions of this nature shall be addressed throughout the remainder of this book, presenting the best answers available at this time.

1.11 Historical Milestones of Coding Theory

We present in Table 1.1 a brief summary of major accomplishments in coding theory and some of the significant contributors to that theory, or expositors who contributed by bringing together the significant contributions to date. Some dates and contributions may not be exactly as portrayed here; it is difficult to sift through the sands of recent history. Also, significant contributions to coding are made *every month*, so this cannot be a complete list.

1.12 A Bit of Information Theory

This section serves as an appendix to this introductory chapter, providing background on the basic definitions and concepts of information theory. As information-theoretic concepts are invoked throughout this book, readers may want to refer to this section for this background material.

1.12.1 Information- Theoretic Definitions for Discrete Random Variables

1.12.1.1 Entropy and Conditional Entropy

Definition 1.15 Let X be a discrete random variable taking values in a set $\mathcal{X} = \{x_1, x_2, \ldots, x_m\}$ with probabilities $P(X = x_i)$, $x_i \in \mathcal{X}$. Different notations for these probabilities are $p_X(x_i)$, $p(x_i)$, or p_i. The **entropy** of X is

$$H(X) = E[-\log_2 P(X)] = -\sum_{x\in\mathcal{X}} P(X = x)\log_2 P(X = x) = -\sum_{i=1}^{m} p_i \log_2 p_i$$
$$= -\sum_{x\in\mathcal{X}} p_X(x)\log_2 p_X(x).$$

When logarithm base is 2, the units of entropy are in *bits*. When the logarithm base is e (natural log), the units of entropy are in *nats* . □

The entropy represents the uncertainty there is about X prior to its measurement; equivalently, it is the amount of information gained when X is measured. The entropy of X depends on the *probabilities* with which X takes on its values, not the particular *values* that X takes on, so it is a function of p_X, not $\mathcal{X}$.

When the base of logarithms is 2 (as above), then the entropy is in units of *bits*. When the base of logarithms is e, the entropy is in units of *nats*. Writing the base of the entropy as $H_b(X)$, it is straightforward to show that

$$H_b(X) = (\log_b a)H_a(X).$$

For example,

$$H_e(X) = (\log_e 2)H_2(X).$$

Thus, to convert from bits to nats, multiply by $\log_e 2$.

It is straightforward to show that for any distribution, $H(X) \geq 0$.

Example 1.16 Let

$$X = \begin{cases} 1 & \text{with probability } p \\ 0 & \text{with probability } 1-p. \end{cases}$$

Then

$$H(X) = -p\log p - (1-p)\log(1-p).$$

This is also denoted as $H(p)$ (since it depends only on p).

For example, when $p = 0.1$, then $H(X) = 0.469$ bits (using $\log_2$) and $H(X) = 0.3251$ nats (using $\log_e$). □

Table 1.1: Historical Milestones

Year	Milestone
1948	Shannon publishes "A Mathematical Theory of Communication" [404]
1950	Hamming describes Hamming codes [181]
1954	Reed [362] and Muller [321] both present Reed–Muller codes and their decoders
1955	Elias introduces convolutional codes [109]
1957	Prange introduces cyclic codes [348]
1959	A. Hocquenghem [201] and ...
1960	Bose and Ray-Chaudhuri [48] describe BCH codes Reed and Solomon produce eponymous codes [364] Peterson provides a solution to BCH decoding [336]
1961	Peterson produces his book [337], later extended and revised by Peterson and Weldon [338]
1962	Gallager introduces LDPC codes [148] 2400 BPS modem commercially available (4-PSK) (see [138])
1963	The Fano algorithm for decoding convolutional codes introduced [113] Massey unifies the study of majority logic decoding [297]
1966	Forney produces an in-depth study of concatenated codes [126] introduces generalized minimum distance decoding [127]
1967	Berlekamp introduces a fast algorithm for BCH/Reed–Solomon decoding [36] Rudolph initiates the study of finite geometries for coding [385] 4800 BPS modem commercially available (8-PSK) (see [138])
1968	Berlekamp produces *Algebraic Coding Theory* [33] Gallager produces *Information theory and reliable communication* [147]
1969	Jelinek describes the stack algorithm for decoding convolutional codes [219] Massey introduces his algorithm for BCH decoding [295] Reed–Muller code flies on *Mariner* deep space probes using Green machine decoder
1971	Viterbi introduces the algorithm for ML decoding of convolutional codes [468] 9600 BPS modem commercially available (16-QAM) (see [138])
1972	The BCJR algorithm is described in the open literature [21]
1973	Forney elucidates the Viterbi algorithm [128]
1975	Sugiyama et al. propose the use of the Euclidean algorithm for decoding [424]
1977	MacWilliams and Sloane produce the encyclopedic *The Theory of Error Correcting Codes* [292] *Voyager* deep space mission uses a concatenated RS/convolutional code (see [305])
1978	Wolf introduces a trellis description of block codes [488]
1980	14,400 BPS modem commercially available (64-QAM) (see [138]) Sony and Phillips standardize the compact disc, including a shortened Reed–Solomon code
1981	Goppa introduces algebraic-geometry codes [163, 164]
1982	Ungerboeck describes trellis-coded modulation [452]
1983	Lin and Costello produce their engineering textbook [271] Blahut publishes his textbook [45]
1984	14,400 BPS TCM modem commercially available (128-TCM) (see [138])
1985	19,200 BPS TCM modem commercially available (160-TCM) (see [138])
1993	Berrou, Glavieux, and Thitimajshima announce turbo codes [39]
1994	The $\mathbb{Z}_4$ linearity of families of nonlinear codes is announced [182]
1995	MacKay resuscitates LDPC codes [290] Wicker publishes his textbook [483]
1996	33,600 BPS modem (V.34) modem is commercially available (see [136])
1998	Alamouti describes a space-time code [8]
1999	Guruswami and Sudan present a list decoder for RS and AG codes [172]
2000	Aji and McEliece [6] (and others [255]) synthesize several decoding algorithms using message passing ideas
2002	Hanzo, Liew, and Yeap characterize turbo algorithms in [187]
2003	Koetter and Vardy extend the GS algorithm for soft-decision decoding of RS codes [251]
2004	Lin and Costello second edition [272]
2009	Arıkan invents polar codes [16]
2020	Moon produces what is hoped to be a valuable 2nd Edition!

Now suppose that $Y = f(X)$ for some probabilistic function $f(X)$. (For example, Y might be the output of a noisy channel that has X as the input, such that $Y = X + N$ for a discrete random variable N.) Let $\mathcal{Y}$ denote the set of possible Y outcomes. Let $P_Y(y) = P(Y = y)$. X and Y are jointly distributed with joint probability $P_{XY}(x, y)$. The conditional probability of X, given that $Y = y$ is

$$P(X|Y = y) = \frac{P_{XY}(X, y)}{P_Y(y)}.$$

$H(X|y)$ is the uncertainty remaining about X when Y is measured as $Y = y$:

$$H(X|y) = E[-\log_2 P_{X|Y}(X|y)] = -\sum_{x\in\mathcal{X}} P_{X|Y}(x|y) \log_2 P_{X|Y}(x|y) \text{ (bits)}.$$

Then the average uncertainty in X averaged over the outcomes Y is called the *conditional entropy*, $H(X|Y)$, computed as

$$\begin{aligned} H(X|Y) &= \sum_{y\in\mathcal{Y}} H(X|y)P_Y(y) = -\sum_{y\in\mathcal{Y}}\sum_{x\in\mathcal{X}} P_{X|Y}(x|y)P_Y(y) \log_2 P_{X|Y}(x|y) \\ &= -\sum_{y\in\mathcal{Y}}\sum_{x\in\mathcal{X}} P_{XY}(x, y) \log_2 P_{X|Y}(x|y) \text{ (bits)}. \end{aligned}$$

The **joint entropy** is defined by

$$H(X, Y) = -\sum_{x\in\mathcal{X}}\sum_{y\in\mathcal{Y}} P_{XY}(x, y) \log_2 P_{XY}(x, y).$$

It is straightforward to show that

$$H(X, Y) = H(X) + H(Y|X).$$

This is referred to as the *chain rule* for entropy. This has the interpretation: The uncertainty in the joint variables (X, Y) is the uncertainty in X plus the uncertainty remaining in Y when X is given. By symmetry,

$$H(X, Y) = H(Y) + H(X|Y).$$

This rule also holds if all arguments are conditioned on the same event:

$$H(X, Y|Z) = H(X|Z) + H(Y|X, Z).$$

The chain rule for conditional entropy can be extended to multiple variables:

$$\begin{aligned} H(X_1, X_2, \ldots, X_n) &= \sum_{i=1}^{n} H(X_i|X_{i-1}, \ldots, X_1) \\ &= H(X_1) + H(X_2|X_1) + H(X_3|X_2, X_1) + \cdots + H(X_n|X_{n-1}, \ldots, X_1). \end{aligned}$$

1.12.1.2 Relative Entropy, Mutual Information, and Channel Capacity

Definition 1.17 Let $P(x)$ and $Q(x)$ be two probability mass functions on the outcomes in $\mathcal{X}$. The **Kullback–Leibler distance** $D(P||Q)$ between two probability mass functions, also known as the **relative entropy** or the **cross entropy** is

$$D(P||Q) = E_P\left[\log \frac{P(x)}{Q(x)}\right] = \sum_{x\in\mathcal{X}} P(x) \log \frac{P(x)}{Q(x)}.$$

□

Lemma 1.18 *$D(P||Q) \geq 0$, with equality if and only if $p = q$; that is, if the two distributions are the same.*

Proof We use the inequality $\log x \leq x - 1$, with equality only at $x = 1$ (where the log is $\log_e$). This inequality appears so frequently in information theory it has been termed the information theory inequality. Then

$$\begin{aligned} D(P||Q) &= \sum_{x\in\mathcal{X}} P(x) \log \frac{P(x)}{Q(x)} = -\sum_{x\in\mathcal{X}} P(x) \log \frac{Q(x)}{P(x)} \\ &\geq \sum_{x\in\mathcal{X}} P(x) \left[1 - \frac{Q(x)}{P(x)}\right] \quad \text{(information theory inequality)} \\ &= \sum_{x\in\mathcal{S}} P(x) - Q(x) = 0. \end{aligned}$$

□

Definition 1.19 Let X be a random variable taking values in the set $\mathcal{X}$ and let Y be a random variable taking values in the set $\mathcal{Y}$, and let X and Y be jointly and marginally distributed as $P_{XY}(x, y)$, $P_X(x)$, and $P_Y(y)$, respectively. The **mutual information** between X and Y is the Kullback–Leibler distance between the joint distribution $P_{XY}(x, y)$ and the product of the marginals $P_X(x)P_Y(y)$:

$$I(X;Y) = D(P_{XY}||P_X P_Y) = \sum_{x\in\mathcal{X}} \sum_{y\in\mathcal{Y}} P_{XY}(x,y) \log \frac{P_{XY}(x,y)}{P_X(x)P_Y(y)}. \tag{1.38}$$

□

If X and Y are independent, so that $P_{XY}(x, y) = P_X(x)P_Y(y)$, then $I(X; Y) = 0$. That is, Y tells no information at all about X. From the definition,

$$I(X;Y) = I(Y;X) \quad \text{and} \quad I(X;X) = H(X).$$

Using the definitions, mutual information can also be written as

$$I(X;Y) = H(X) - H(X|Y).$$

This is shown as follows:

$$\begin{aligned} I(X;Y) &= \sum_{x,y} P_{XY}(x,y) \log \frac{P_{XY}(x,y)}{P_X(x)P_Y(y)} \\ &= \sum_{x,y} P_{XY}(x,y) \log \frac{P_{X|Y}(x|y)}{P_X(x)} \\ &= -\sum_{x,y} P_{XY}(x,y) \log P_X(x) + \sum_{x,y} P_{XY}(x,y) \log P_{X|Y}(x|y) \\ &= -\sum_{x} P_X(x) \log P_X(x) - \left(-\sum_{x,y} P_{XY}(x,y) \log P_{X|Y}(x|y)\right) \\ &= H(X) - H(X|Y). \end{aligned}$$

The mutual information is the difference between the average uncertainty in X and the uncertainty in X there still is after measuring Y. Thus, it quantifies how much information Y tells about X. Since the Definition (1.38) is symmetric, also

$$I(X;Y) = H(Y) - H(Y|X).$$

We can also write

$$I(X;Y) = H(X) + H(Y) - H(X,Y).$$

In light of Lemma 1.18, we see that mutual information $I(X;Y)$ can never be negative.

Definition 1.20 The *conditional mutual information* of random variables X and Y given Z is

$$I(X;Y|Z) = H(X|Z) - H(X|Y,Z). \tag{1.39}$$

□

Similar to entropy, mutual information has a chain rule:

$$\begin{aligned} I(X_1, X_2, \ldots, X_n; Y) &= \sum_{i=1}^{n} I(X_i; Y|X_{i-1}, X_{i-2}, \ldots, X_1) \\ &= I(X_1;Y) + I(X_2;Y|X_1) + I(X_3;Y|X_2,X_1) \\ &\quad + \cdots + I(X_n;Y|X_{n-1}, X_{n-2}, \ldots, X_1). \end{aligned} \tag{1.40}$$

1.12.2 Data Processing Inequality

Let X, Y, and Z be random variables described by mass functions (or pdfs, in the case of continuous random variables). By the properties of conditional probability, for any three such random variables, the joint density factors as

$$p(x,y,z) = p(x)p(y|x)p(z|x,y).$$

(Here, the argument is used to also denote the random variables involved). In the particular case the X, Y, and Z form a Markov chain (in that order), the joint probability mass function (or pdf) can be factored as

$$p(x,y,z) = p(x)p(y|x)p(z|y),$$

that is, $p(z|x,y) = p(z|y)$, so that Z is conditionally independent of X given Y. Variables X, Y, and Z forming a Markov chain are conditionally independent, given Y, since

$$p(x,z|y) = \frac{p(x,y,z)}{p(y)} = \frac{p(x)p(y|x)p(z|y)}{p(y)} = \frac{p(x,y)p(z|y)}{p(y)} = p(x|y)p(z|y).$$

The following theorem is known as the data processing inequality.

Theorem 1.21 *If X, Y and Z form a Markov chain (in that order), then*

$$I(X;Y) \geq I(X;Z).$$

Proof Expand the mutual information using the chain rule in two different ways:

$$\begin{aligned} I(X;Y,Z) &= I(X;Z) + I(X;Y|Z) \\ &= I(X;Y) + I(X;Z|Y). \end{aligned}$$

Since, as observed above, X and Z are conditionally independent given Y, $I(X;Z|Y) = 0$, so

$$I(X;Y) = I(X;Z) + I(X;Y|Z).$$

Since $I(X;Y|Z) \geq 0$, it follows that

$$I(X;Y) \geq I(X;Z).$$

□

An interpretation of this can be given in light of Figure 1.22. X is an input to a channel (or system) which produces an output Y. There is some information that Y provides about X, as determined by $I(X;Y)$. The Y data is processed by some processing block (which is not limited in any way) to produce an output Z. What the data processing inequality teaches is that Z cannot provide more information about X than was already available from Y. (It might provide a more useful representation, but it cannot create information that is not there originally.)

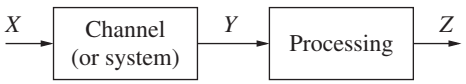


Figure 1.22: Illustration of data processing inequality.

1.12.3 Channels

From an information theoretic view, a channel is a model of the medium by which an input is conveyed to an output, often in a setting of communication. Since communication in the presence of noise involves random perturbations to the input, a channel is described by a transition probability, as follows.

Definition 1.22 A **discrete channel** is a system with an input alphabet $\mathcal{X}$ and an output alphabet $\mathcal{Y}$, with a transition probability $P_{Y|X}(y|x)$ (or, more briefly, $p(y|x)$), which is the probability of observing the output symbol $y \in \mathcal{Y}$ when the input symbol is $x \in \mathcal{X}$. □

A channel is said to be **memoryless** if the probability distribution of the output depends only on the input at the corresponding time, and is conditionally independent of channel inputs or outputs at other times. Let $\mathbf{x} = x_1, x_2, \ldots, x_n$ be a sequence of inputs and $\mathbf{y} = y_1, y_2, \ldots, y_n$ be the corresponding sequence of outputs. Then for a memoryless channel, the joint conditional probability factors:

$$p(\mathbf{y}|\mathbf{x}) = \prod_{i=1}^{n} p(y_i|x_i).$$

1.12.3.1 Binary Symmetric Channel

An important example of a channel is the **BSC** . This models a situation in which bits, having alphabet $\mathcal{X} = \{0, 1\}$, may be flipped as they traverse the channel. A transmitted 0 is received (correctly) as a 0 with probability $1 - p$, and as a 1 (incorrectly) with probability p, and correspondingly for a transmitted 1. Thus, p is the probability that a bit is corrupted as it passed through the channel. p is referred to as the crossover probability. A model for the BSC is shown in Figure 1.23(a).

For the BSC, the mutual information between the input and the output is

$$\begin{aligned} I(X;Y) &= H(Y) - H(Y|X) \\ &= H(Y) - \sum_{x\in\{0,1\}} p(x)H(Y|X=x) \\ &= H(Y) - \sum_{x} p(x)H(P) = H(Y) - H(p) \\ &\leq 1 - H(p). \end{aligned}$$

The last inequality is because Y is a binary random variable (maximum entropy = 1). When the probability distribution on the input is $P_X(0) = P_X(1) = \frac{1}{2}$, then by symmetry $P_Y(0) = P_Y(1) = \frac{1}{2}$, and the mutual information is maximized.

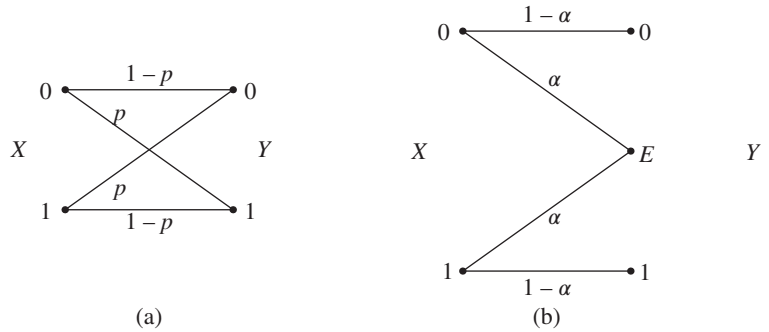


Figure 1.23: BSC and BEC models. (a) Binary symmetric channel with crossover; (b) binary erasure channel with erasure probability.

A useful representation of the channel transition probabilities $p(y|x)$ is as a **transition matrix** whose xth row and yth column represents $p(y|x)$. For the BSC, the transmission matrix is

$$p(y|x) = \begin{bmatrix} 1-p & p \\ p & 1-p \end{bmatrix}.$$

1.12.3.2 Binary Erasure Channel

Another important channel is the **binary erasure channel** (BEC). In the BEC, bits are not corrupted but some bits are lost. This might model, for example, an internet connection in which when packets are received, the bits in the channel are (essentially) reliably received, but some packets may be lost, resulting in lost bits.

A model for the BEC is shown in Figure 1.23(b). The input alphabet is binary, $\mathcal{X} = \{0, 1\}$, and the output alphabet has three symbols, 0, 1, and a symbol denoting an erasure, $\mathcal{Y} = \{0, 1, \mathsf{E}\}$. The probability that a symbol is erased is α:

$$P(Y = E|X = 0) = P(Y = E|X = 1) = \alpha.$$

The transition matrix is

$$p(y|x) = \begin{bmatrix} 1-\alpha & \alpha & 0 \\ 0 & \alpha & 1-\alpha \end{bmatrix}.$$

The mutual information between the input and the output is

$$I(X;Y) = H(Y) - H(Y|X) = H(Y) - H(\alpha)$$

where $H(Y|X) = H(\alpha)$ (the entropy for a binary channel with crossover probability p). It can be shown that $H(Y)$ is maximized when $P(X = 1) = P(X = 0) = \frac{1}{2}$, and in that case,

$$I(X;Y) = 1 - \alpha.$$

1.12.3.3 Noisy Typewriter

Less practical, but helpful to think about the nature of channels, is the noisy typewriter channel. In this case, the input set and output sets are $\mathcal{X} = \mathcal{Y} = \{A, B, C, \ldots, Z\}$. The channel input is either

received unchanged, or the next letter in the alphabet is received, with probability $\frac{1}{2}$. The mutual information is

$$I(X;Y) = H(Y) - H(Y|X).$$

For every given X, there are two possible outcomes Y, so $H(Y|X) = 1$ bit. When all input symbols are equally likely, then all output symbols are equally likely and $H(Y) = \log_2 26$ bits. For the uniform input probabilities, then,

$$I(X;Y) = \log_2 26 - 1 = \log_2 13.$$

1.12.3.4 Symmetric Channels

Definition 1.23 [82, p. 190] A channel is said to be **symmetric** if the rows of the channel transition matrix are permutations of each other and the columns are permutations of each other.

A channel is said to be **weakly symmetric** if every row of the transition matrix is a permutation of every other row, and all the column sums $\sum_x p(y|x)$ are equal. □

The BSC introduced above is an example of a symmetric channel, but the BEC is not a symmetric channel. However, the BEC is weakly symmetric.

1.12.4 Channel Capacity

Definition 1.24 The **channel capacity** C of a channel with input X and output Y is defined as the maximum mutual information between X and Y, where the maximum is taken over all possible input distributions.

$$C = \max_{P_X(x)} I(X;Y).$$

□

For a (weakly) symmetric channel, let $\mathbf{r}$ denote the probabilities of one of the rows and let $H(\mathbf{r})$ be the entropy computed using these probabilities. The mutual information is

$$I(X;Y) = H(Y) - H(Y|X) = H(Y) - H(\mathbf{r}).$$

The entropy of Y is bounded by $\log |\mathcal{Y}|$, so

$$I(X;Y) = \log |\mathcal{Y}| - H(\mathbf{r}).$$

For a symmetric channel, when the input distribution is uniform, $P_X(x) = 1/|\mathcal{X}|$, then the output distribution is also uniform, so $H(Y) = \log |\mathcal{Y}|$. Thus

$$C = \max_{p(x)} I(X;Y) = \log |\mathcal{Y}| - H(\mathbf{r}). \tag{1.41}$$

For the BSC with crossover probability p, the capacity is

$$\boxed{C = 1 - H_2(p),}$$

which is achieved when the input symbols are chosen with $P(X = 0) = P(X = 1) = \frac{1}{2}$. For the BEC with erasure probability α, the capacity is

$$\boxed{C = 1 - \alpha,}$$

which is achieved when the input symbols are chosen with $P(X = 0) = P(X = 1) = \frac{1}{2}$.

1.12.5 Information Theoretic Definitions for Continuous Random Variables

Let Y be a continuous random variable taking on values in an (uncountable) set $\mathcal{Y}$, with pdf $p_Y(y)$. The differential entropy is defined as

$$H(Y) = -E[\log_2 p_Y(y)] = -\int_{\mathcal{Y}} p_Y(y) \log_2 p_Y(y)\, dy.$$

Whereas entropy for discrete random variables is always nonnegative, differential entropy (for a continuous random variable) can be positive or negative.

Example 1.25 Let $Y \sim \mathcal{N}(0, \sigma^2)$. Then

$$\begin{aligned} H(Y) &= -E\left[\log_2 \frac{1}{\sqrt{2\pi}\sigma} e^{-Y^2/2\sigma^2}\right] = -E\left[\log_2 \frac{1}{\sqrt{2\pi}\sigma} + \log_2(e)\left(-\frac{1}{2\sigma^2}\right)(Y^2)\right] \\ &= \log_2(e)\frac{1}{2\sigma^2}E[Y^2] + \frac{1}{2}\log_2 2\pi\sigma^2 \\ &= \frac{1}{2}\log_2(e) + \frac{1}{2}\log_2 2\pi\sigma^2 = \frac{1}{2}\log_2 2\pi e\sigma^2 (\text{bits}). \end{aligned}$$

□

It can be shown that, of all continuous random variables having mean 0 and variance σ^2, the Gaussian $\mathcal{N}(0, \sigma^2)$ has the largest differential entropy.

Let X be a discrete-valued random variable taking on values in the alphabet $\mathcal{X}$ with probability $\Pr(X = x) = P_X(x), x \in \mathcal{X}$ and let X be passed through a channel which produces a continuous-valued output Y for $Y \in \mathcal{Y}$. A typical example of this is the AWGNC , where

$$Y = X + N,$$

and $N \sim \mathcal{N}(0, \sigma^2)$. Let

$$p_{XY}(x, y) = p_{Y|X}(y|x)P_X(x)$$

denote the joint distribution of X and Y and let

$$p_Y(y) = \sum_{x\in\mathcal{X}} p_{XY}(x, y) = \sum_{x\in\mathcal{X}} p_{Y|X}(y|x)P_X(x)$$

denote the pdf of Y. Then the mutual information $I(X; Y)$ is computed as

$$\begin{aligned} I(X; Y) &= D(p_{XY}(X, Y)||P_X(X)p_Y(Y)) = \int_{\mathcal{Y}} \sum_{\mathcal{X}} p_{XY}(x, y) \log_2 \frac{p_{XY}(x, y)}{p_Y(y)P_X(x)}\, dy \\ &= \sum_{x\in\mathcal{X}} \int_{y\in\mathcal{Y}} p_{Y|X}(y|x)P_X(x) \log_2 \frac{p_{Y|X}(y|x)}{\sum_{x'\in\mathcal{X}} p_{Y|X}(y|x')P_X(x')}\, dy. \end{aligned}$$

Example 1.26 Suppose $\mathcal{X} = \{a, -a\}$ (e.g., BPSK modulation with amplitude a) with probabilities $P(X = a) = P(X = -a) = \frac{1}{2}$. Let $N \sim \mathcal{N}(0, \sigma^2)$ and let

$$Y = X + N.$$

Because the channel has only binary inputs, this is referred to as the *binary* additive white Gaussian noise channel (BAWGNC). Then

$$\begin{aligned} I(X;Y) &= \frac{1}{2}\left[\int_{-\infty}^{\infty} p(y|a)\log_2\frac{p(y|a)}{\frac{1}{2}(p(y|a)+p(y|-a))}\right. \\ &\qquad \left. + p(y|-a)\log_2\frac{p(y|-a)}{\frac{1}{2}(p(y|a)+p(y|-a))}\,dy\right] \\ &= \frac{1}{2}\int_{-\infty}^{\infty} p(y|a)\log_2 p(y|a) + p(y|-a)\log_2 p(y|-a) \qquad (1.42) \\ &\quad - (p(y|a)+p(y|-a))\log_2\left[\frac{1}{2}(p(y|a)+p(y|-a))\right]\,dy \\ &= \frac{1}{2}[-H(Y) - H(Y) \\ &\quad - \int_{-\infty}^{\infty}(p(y|a)+p(y|-a))\log_2\left[\frac{1}{2}(p(y|a)+p(y|-a))\right]\,dy\Bigg] \\ &= \boxed{-\int_{-\infty}^{\infty}\phi(y,a,\sigma^2)\log_2\phi(y,a,\sigma^2)\,dy - \frac{1}{2}\log_2 2\pi e\sigma^2 \text{ (bits)},} \qquad (1.43) \end{aligned}$$

where we define the function

$$\phi(y,a,\sigma^2) = \frac{1}{\sqrt{8\pi\sigma^2}}[e^{-(y-a)^2/2\sigma^2} + e^{-(y+a)^2/2\sigma^2}].$$

□

When both the channel input X and the output Y are continuous random variables, then the mutual information is

$$I(X;Y) = D(p_{XY}(X,Y)||P_X(x)p_Y(Y)) = \int_{\mathcal{X}}\int_{\mathcal{Y}} p_{XY}(x,y)\log_2\frac{p_{XY}(x,y)}{p_Y(y)P_X(x)}\,dy\,dx.$$

Example 1.27 Let $X \sim \mathcal{N}(0,\sigma_x^2)$ and $N \sim \mathcal{N}(0,\sigma_n^2)$, independent of X. Let $Y = X + N$. Then $Y \sim \mathcal{N}(0,\sigma_x^2+\sigma_n^2)$.

$$\begin{aligned} I(X;Y) &= H(Y) - H(Y|X) = H(Y) - H(X+N|X) = H(Y) - H(N|X) \\ &= H(Y) - H(N) = \frac{1}{2}\log_2 2\pi e\sigma_y^2 - \frac{1}{2}\log_2 2\pi e\sigma_n^2 \\ &= \boxed{\frac{1}{2}\log_2\left(1+\frac{\sigma_x^2}{\sigma_n^2}\right)\text{ (bits).}} \qquad (1.44) \end{aligned}$$

The quantity σ_x^2 represents the average power in the transmitted signal X and σ_n^2 represents the average power in the noise signal N. This channel is called the AWGNC. □

As for the discrete channel, the channel capacity C of a channel with input X and output Y is the maximum mutual information between X and Y, where the maximum is over all input distributions. In Example 1.26, the maximizing distribution is, in fact, the uniform distribution, $P(X=a) = P(X=-a) = \frac{1}{2}$, so (1.43) is the capacity for the BAWGNC. In Example 1.27, the maximizing distribution is, in fact, the Gaussian distribution (since this maximizes the entropy of the output), so (1.44) is the capacity for the AWGNC.

1.12.6 The Channel Coding Theorem

The channel capacity has been *defined* as the maximum mutual information between the input and the output. But Shannon's channel coding theorem, tells us what the capacity *means*. Recall that an error

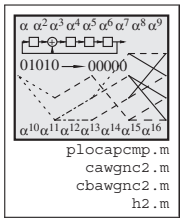
correction code has a rate $R = k/n$, where k is the number of input symbols and n is the number of output symbols, the length of the code. The channel coding theorem says this:

> Provided that the coded rate of transmission R is less than the channel capacity, for any given probability of error ϵ specified, there is an error correction code of length n_0 such that there exist codes of length n exceeding n_0 for which the decoded probability of error is less than ϵ.

That is, provided that we transmit at a rate less than capacity, arbitrarily low probabilities of error can be obtained, if a sufficiently long error correction code is employed. The capacity is the amount of information that can be transmitted *reliably* through the channel *per channel use*.

A converse to the channel coding theorem states that for a channel with capacity C, if $R > C$, then the probability of error is bounded away from zero: reliable transmission is not possible.

The channel coding theorem is an *existence* theorem; it tells us that codes exist that can be used for reliable transmission, but not how to find practical codes. Shannon's remarkable proof used random codes. But as the code gets long, the decoding complexity of a truly random (unstructured) code goes up exponentially with the length of the code. Since Shannon's proof, engineers and mathematicians have been looking for ways of constructing codes that are both good (meaning they can correct a lot of errors) and practical, meaning that they have some kind of structure that makes decoding of sufficiently low complexity that decoders can be practically constructed.



plocapcmp.m
cawgnc2.m
cbawgnc2.m
h2.m

Figure 1.24 shows a comparison of the capacity of the AWGNC and the BAWGNC channels as a function of E_c/σ^2 (an SNR measure). In this figure, we observe that the capacity of the AWGNC increases with SNR beyond 1 bit per channel use, while the BAWGNC asymptotes to a maximum of 1 bit per channel use — if only binary data is put into the channel, only 1 bit of useful information can be obtained. It is always the case that

$$C_{\text{AWGNC}} > C_{\text{BAWGNC}}.$$

Over all possible input distributions, the Gaussian distribution is information maximizing, so C_{AWGNC} is an upper bound on capacity for any modulation or coding that might be employed. However, at very low SNRs, C_{AWGNC} and C_{BAWGNC} are very nearly equal.

Figure 1.24 also shows the capacity of the equivalent BSC, with crossover probability $p = Q(\sqrt{E_c/\sigma^2})$ and capacity $C_{\text{BSC}} = 1 - H_2(p)$. This corresponds to hard-input decoding. Clearly, there is some loss of potential rate due to hard-input decoding, although the loss diminishes as the SNR increases.

1.12.7 "Proof" of the Channel Coding Theorem

In this section, we present a "proof" of the channel coding theorem. While mathematically accurate, it is not complete. The arguments can be considerably tightened, but are sufficient to show the main ideas of coding. Also, the proof is only presented for the discrete input/discrete channel case. The intuition, however, generally carries over to the Gaussian channel.

An important preliminary concept is the "asymptotic equipartition property" (AEP). Let X be a random variable taking values in a set $\mathcal{X}$. Let $\mathbf{X} = (X_1, X_2, \ldots, X_n)$ be an i.i.d. (independent, identically distributed) random vector and let $\mathbf{x}$ denote an outcome of $\mathbf{X}$.

Theorem 1.28 ***(AEP)*** *Let $\boldsymbol{X} = (X_1, X_2, \ldots, X_n)$, where X_i are i.i.d., and let $\boldsymbol{x} = (x_1, x_2, \ldots, x_n)$ be a particular draw. As $n \to \infty$*

$$P(\boldsymbol{X} = \boldsymbol{x}) \approx 2^{-nH(X)}, \quad \boldsymbol{x} \in \mathcal{T}, \tag{1.45}$$

The interpretation of this theorem is as follows. Note that the probability $2^{-nH(X)}$ is a function of the distribution of X, not the particular outcome $\mathbf{x}$. By the AEP, most of the probability of a draw of $\mathbf{X}$ is "concentrated" in a set that occurs with probability $2^{-nH(X)}$. the typical set. That is, a "typical" outcome

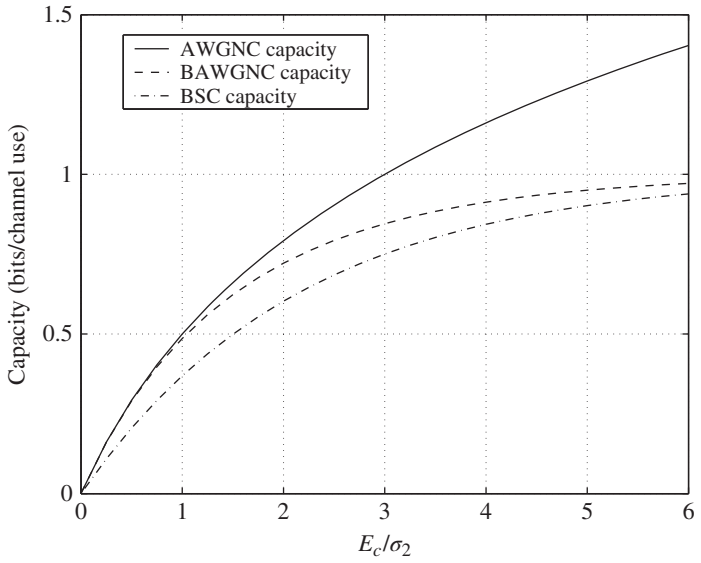


Figure 1.24: Capacities of AWGNC, BAWGNC, and BSC.

is likely to occur, while an outcome which is not "typical" is not likely to occur. Since the "typical" outcomes all have approximately the same probability, given by (1.45), there must be approximately $2^{nH(X)}$ outcomes in the typical set $\mathcal{T}$.

Proof We sketch the main idea of the proof. Let the outcome space for a random variable Y be $\mathcal{Y} = \{b_1, b_2, \ldots, b_K\}$, occurring with probabilities $P_i = P(Y = b_i)$. Out of n samples of the i.i.d. variable Y, let n_i be the number of outcomes that are equal to b_i. By the law of large numbers,[7] when n is large,

$$\frac{n_i}{n} \approx P_i. \tag{1.46}$$

The product of n observations can be written as

$$\begin{aligned} y_1 y_2 \cdots y_n &= b_1^{n_1} b_2^{n_2} \cdots b_K^{n_K} = [b_1^{(n_1/n)} b_2^{(n_2/n)} \cdots b_K^{(n_K/n)}]^n \\ &= \left[2^{\frac{n_1}{n}\log_2 b_1} 2^{\frac{n_2}{n}\log_2 b_2} \cdots 2^{\frac{n_K}{n}\log_2 b_K}\right]^n = \left[2^{\sum_{i=1}^K \frac{n_i}{n}\log_2 b_i}\right]^n \\ &\approx \left[2^{\sum_{i=1}^K P_i \log_2 b_i}\right]^n \quad \text{(by (1.46))} \\ &= [2^{E[\log_2 Y]}]^n. \end{aligned} \tag{1.47}$$

Now suppose that Y is, in fact, a function of a random variable X, $Y = f(X)$, where, specifically, this function is the probability mass function of X: $f(x) = P_X(x) = P(X = x)$. Then

$$y_1 y_2 \cdots y_n = f(x_1) f(x_2) \cdots f(x_n) = \prod_{i=1}^n P_X(x_i) = P(\mathbf{X} = \mathbf{x}).$$

Also, by (1.47),

$$y_1 y_2 \cdots y_n = f(x_1) f(x_2) \cdots f(x_n) = \prod_{i=1}^n p_X(x_i) \approx [2^{E[\log_2 p_X(X)]}]^n = 2^{-nH(X)}.$$

This establishes (1.45). □

[7] Thorough proof of the AEP merely requires putting all of the discussion in the formal language of the weak law of large numbers.

Let X be a binary source with entropy $H(X)$ and let each X be transmitted through a memoryless channel to produce the output Y. Consider transmitting the sequence of i.i.d. outcomes $x_1, x_2, \ldots, x_n$. While the number of possible sequences is $M = 2^n$, the typical set has only about $2^{nH(X)}$ sequences in it. Let the total possible number of output sequences $\mathbf{y} = y_1, y_2, \ldots, y_n$ be N. There are about $2^{nH(Y)} \leq N$ typical output sequences. For each typical output sequence $\mathbf{y}$ there are approximately $2^{nH(X|Y)}$ input sequences that could have caused it. Furthermore, each input sequence $\mathbf{x}$ typically could produce $2^{nH(Y|X)}$ output sequences. This is summarized in Figure 1.25(a).

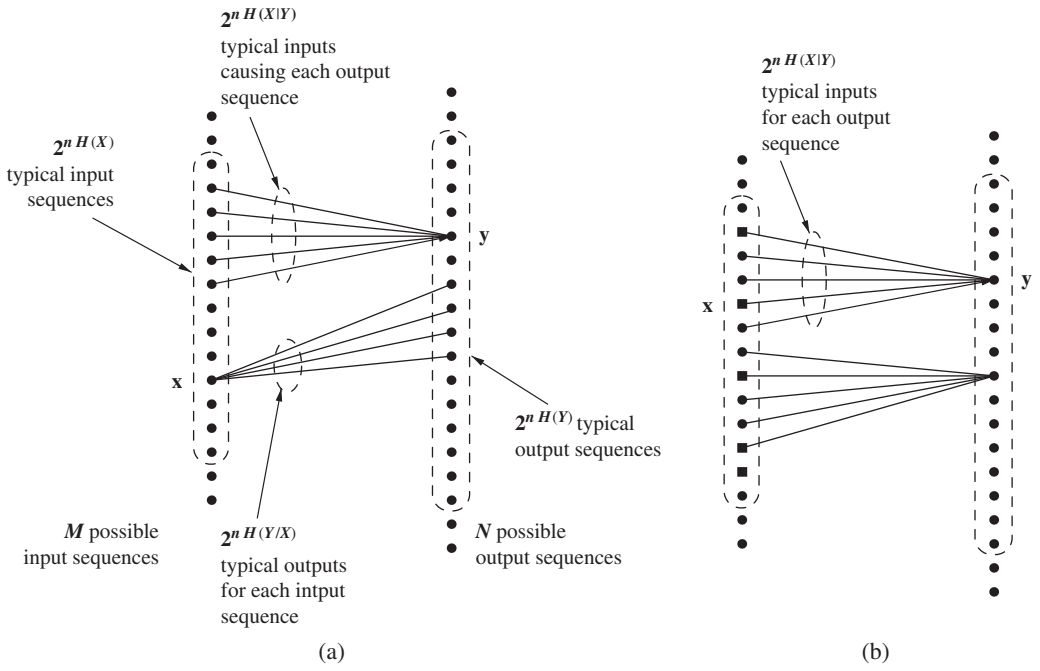


Figure 1.25: Relationship between input and output entropies for a channel. Each • or ■ represents a sequence. (a) Basic input output relationship; (b) ■ represents codewords.

Now let X be coded by a rate-R code to produce a coded sequence which selects, out of the 2^n possible input sequences, only 2^{nR} of these. In Figure 1.25(b), these coded sequences are denoted with filled squares, ■. The mapping which selects the 2^{nR} points is the *code*. Rather than select any particular code, we contemplate using all possible codes *at random* (using, however, only the typical sequences). Under the random code, a sequence selected at random is a codeword with probability

$$\frac{2^{nR}}{2^{nH(X)}} = 2^{n(R-H(X))}.$$

Now consider the problem of correct decoding. A sequence $\mathbf{y}$ is observed. It can be decoded correctly if there is only one code vector $\mathbf{x}$ that could have caused it. From Figure 1.25(b), the probability that none of the points in the "fan" leading to $\mathbf{y}$ other than the original code point is a codeword is

$$\begin{aligned} P &= (\text{probability a point } \mathbf{x} \text{ is not a codeword})^{(\text{typical number of inputs for this } \mathbf{y})} \\ &= (1 - 2^{n(R-H(X))})^{2^{nH(X|Y)}}. \end{aligned}$$

If we now choose $R < \max_{P_X(x)} H(X) - H(X|Y)$, that is, choose $R <$ the capacity C, then

$$R - H(X) + H(X|Y) < 0$$

for any input distribution $P_X(x)$. In this case,

$$R - H(X) = -H(X|Y) - \eta$$

for some $\eta > 0$. Then

$$P = (1 - 2^{n(-H(X|Y)-\eta)})^{2^{nH(X|Y)}}.$$

Expanding out using the binomial expansion,

$$P = 1 - \binom{2^{nH(X|Y)}}{1} 2^{n(-H(X|Y)-\eta)} + \text{higher order terms},$$

so as $n \to \infty$,

$$P \to 1 - 2^{-n\eta} \to 1.$$

Thus, the probability that none of the points except the original code point leading to $\mathbf{y}$ is a codeword approaches 1, so that the probability of decoding error — due to multiple codewords mapping to a single received vector — approaches 0.

We observe that if the *average* of an ensemble approaches zero, then there are elements in the ensemble that must approach 0. Thus, among all the possible codes in the ensemble, there are codes for which the probability of error approaches zero as $n \to \infty$.

There are two other ways of viewing the coding rate requirement. The $2^{nH(Y|X)}$ typical sequences resulting from transmitting a vector $\mathbf{x}$ must partition the $2^{nH(Y)}$ typical output sequences, so that each observed output sequence can be attributed to a unique input sequence. The number of subsets in this partition is

$$\frac{2^{nH(Y)}}{2^{nH(Y|X)}} = 2^{n(H(Y)-H(Y|X))},$$

so the condition $R < H(Y) - H(Y|X)$ must be enforced. Alternatively, the $2^{nH(X)}$ typical input sequences must be partitioned so that the $2^{nH(X|Y)}$ typical input sequences associated with an observation $\mathbf{y}$ are disjoint. There must be $2^{n(H(X)-H(X|Y))}$ distinct subsets, so again the condition $R < H(X) - H(X|Y)$ must be enforced.

Let us summarize what we learn from the proof of the channel coding theorem:

- As long as $R < C$, arbitrarily reliable transmission is possible.
- The code lengths, however, may have to be long to achieve the desired reliability. The closer R is to C, the larger we would expect n to need to be in order to obtain some specified level of performance.
- Since the theorem was based on ensembles of random codes, it does not specify what the best code should be. We don't know how to "design" the best codes, we only know that they exist.
- However, random codes have a high probability of being good. So we are likely to get a good code simply by picking one at random!

So what, then, is the issue? Why the need for decades of research in coding theory, if a code can simply be selected at random? The answer has to do with the complexity of representing and decoding the code. To represent a random code of length n, there must be memory to store all the codewords, which requires $n2^{Rn}$ bits. Furthermore, to decode a received word $\mathbf{y}$, ML decoding for a random code requires that a received vector $\mathbf{y}$ must be compared with all 2^{Rn} possible codewords. For a $R = 1/2$ code with $n = 1000$ (a relatively modest code length and a low-rate code), 2^{500} comparisons must be made for each received vector. This is prohibitively expensive, beyond practical feasibility for even massively parallel computing systems, let alone a portable communication device.

Ideally, we would like to explore the space of codes parameterized by rate, probability of decoding error, block length (which governs latency), and encoding and decoding complexity, identifying thereby

all achievable tuples of $(R, P, n, \chi_E, \chi_D)$, where P is the probability of error and χ_E and χ_D are the encoding and decoding complexities. This is an overwhelmingly complex task. The essence of coding research has taken the pragmatic stance of identifying families of codes which have some kind of algebraic or graphical *structure* that will enable representation and decoding with manageable complexity. In some cases what is sought are codes in which the encoding and decoding can be accomplished readily using algebraic methods — essentially so that decoding can be accomplished by solving sets of equations. In other cases, codes employ constraints on certain graphs to reduce the encoding and decoding complexity. Most recently, families of codes have been found for which very long block lengths can be effectively obtained with low complexity using very sparse representations, which keep the decoding complexity in check. Describing these codes and their decoding algorithms is the purpose of this book.

The end result of the decades of research in coding is that the designer has a rich palette of code options, with varying degrees of rate and encode and decode complexity. This book presents many of the major themes that have emerged from this research.

1.12.8 Capacity for the Continuous-Time AWGN Channel

Let X_i be a zero-mean random variable with $E[X_i^2] = \sigma_x^2$ which is input to a discrete AWGN channel, so that

$$R_i = X_i + N_i,$$

where the N_i are i.i.d. $N_i \sim \mathcal{N}(0, \sigma_n^2)$. The capacity of this channel is

$$C = \frac{1}{2}\log_2\left(1 + \frac{\sigma_x^2}{\sigma_n^2}\right) \text{ bits/channel use.}$$

Now consider sending a continuous-time signal $x(t)$ according to

$$x(t) = \sum_{i=1}^{n} X_i \varphi_i(t),$$

where the $\varphi_i(t)$ functions are orthonormal over $[0, T]$. Let us suppose that the transmitter power available is P watts, so that the energy dissipated in T seconds is $E = PT$. This energy is also expressed as

$$E = \int_0^T x^2(t)\, dt = \sum_{i=1}^{n} X_i^2.$$

We must therefore have

$$\sum_{i=1}^{n} X_i^2 = PT$$

or $nE[X_i^2] = PT$, so that $\sigma_x^2 = PT/n$.

Now consider transmitting a signal $x(t)$ through a continuous-time channel with bandwidth W. By the sampling theorem (frequently attributed to Nyquist, but in this context it is frequently called *Shannon's* sampling theorem), a signal of bandwidth W can be exactly characterized by $2W$ samples/second — any more samples than this cannot convey any more information about this bandlimited signal. So we can get $2W$ independent channel uses per second over this bandlimited channel. There are $n = 2WT$ symbols transmitted over T seconds.

If the received signal is

$$R(t) = x(t) + N(t),$$

where $N(t)$ is a white Gaussian noise random process with two-sided power spectral density $N_0/2$, then in the discrete-time sample

$$R_i = x_i + N_i,$$

where $R_i = \int_0^T R(t)\varphi_i(t)\,dt$, the variance of N_i is $\sigma^2 = N_0/2$. The capacity for this bandlimited channel is

$$C = \left(\frac{1}{2}\log_2\left(1 + \frac{PT/n}{N_0/2}\right) \text{ bits/channel use}\right)(2W \text{ channel uses/second})$$
$$= W\log_2\left(1 + \frac{2PT}{nN_0}\right) \text{ bits/second.}$$

Now, using $n = 2WT$, we obtain

$$C = W\log_2(1 + P/N_0W) \text{ bits/second.} \tag{1.48}$$

Since P is the average transmitted power, in terms of its units, we have

$$P = \frac{\text{energy}}{\text{second}} = (\text{energy/bit})(\text{bits/second}).$$

Since E_b is the energy/bit and the capacity is the rate of transmission in bits per second, we have $P = E_bR_b = E_bC$ (when transmitting at capacity), giving

$$C = W\log_2\left(1 + \frac{C}{W}\frac{E_b}{N_0}\right). \tag{1.49}$$

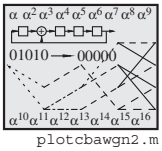
Let $\eta = C/W$ be the spectral efficiency in bits/second/Hz; this is the data rate available for each Hertz of channel bandwidth. From (1.49),

$$\eta = \log_2\left(1 + \eta\frac{E_b}{N_0}\right)$$

or

$$E_b/N_0 = \frac{2^\eta - 1}{\eta}. \tag{1.50}$$

For BPSK the spectral efficiency is $\eta = 1$ bit/second/Hz, so (1.50) indicates that it is theoretically possible to transmit arbitrarily reliably at $E_b/N_0 = 1$, which is 0 dB. In principle, then, it should be possible to devise a coding scheme which could transmit BPSK-modulated signals arbitrarily reliably at an SNR of 0 dB. By contrast, for uncoded transmission when $E_b/N_0 = 9.6$ dB, the BPSK performance shown in Figure 1.10 has $P_b = 10^{-5}$. There is at least 9.6 dB of coding gain possible. The approximately 0.44 dB of gain provided by the (7,4) Hamming code of Section 1.9 falls over 9 dB short of what is theoretically possible!



plotcbawgn2.m
cbawgnc.m
cawgnc.m
philog.m
phifun.m

1.12.9 Transmission at Capacity with Errors

By the channel coding theorem, zero probability of error is (asymptotically) attainable provided that the transmission rate is less than the capacity. What if we allow a nonvanishing probability of error. What is the maximum rate of transmission? Or, equivalently, for a given rate, which is the minimum SNR that will allow transmission at that rate, with a specified probability of error?

The theoretical tools we need to address these questions are the separation theorem and rate–distortion theory. The separation theorem says that we can separately consider and optimally (at least, asymptotically) data compression and error correction. Suppose that the source has a rate of r bits/second. First, compress the information so that the bits of the compressed signal match the bits of the source signal with probability p. From rate-distortion theory, this produces a source at rate $1 - H_2(p)$ per source bit (see (1.2)). These compressed bits, at a rate $r(1 - H_2(p))$ are then transmitted over the channel with vanishingly small probability of error. We must therefore have $r(1 - H_2(p)) < C$. The maximum rate achievable with average distortion (i.e., probability of bit error) p, which we denote

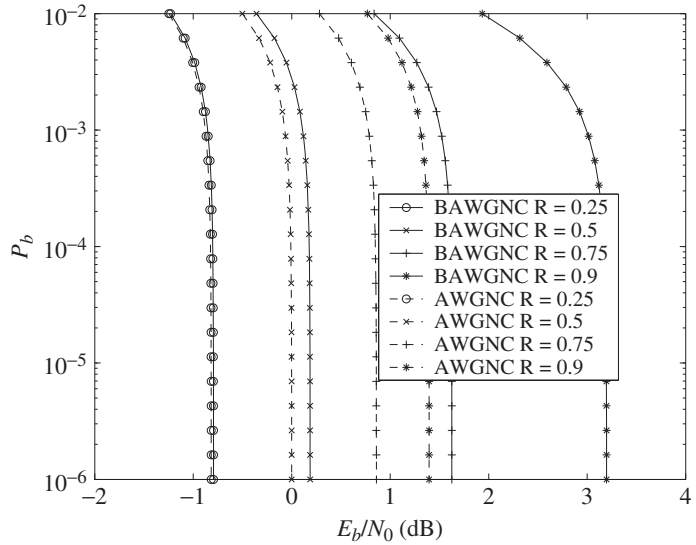


Figure 1.26: Capacity lower bounds on P_b as a function of SNR.

as $C^{(p)}$, is therefore

$$C^{(p)} = \frac{C}{1 - H_2(p)}.$$

Figure 1.26 shows the required SNR E_b/N_0 for transmission at various rates for both the BAWGNC and the AWGNC. For any given line in the plot, the region to the right of the plot is achievable — it should theoretically be possible to transmit at that probability of error at that SNR. Curves such as these therefore represent a goal to be achieved by a particular code: we say that we are transmitting at capacity if the performance falls on the curve.

We note the following from the plot:

- At very low SNR, the binary channel and the AWGN channel have very similar performance. This was also observed in conjunction with Figure 1.24.
- The higher the rate, the higher the required SNR.
- The vertical asymptote (as $P_b \to 0$) is the capacity C for that channel.

1.12.10 The Implication of the Channel Coding Theorem

The implication of the channel coding theorem, fundamentally, is that for a block code of length n and rate $R = k/n$, the probability of a block decoding error can be bounded as

$$P(E) \leq 2^{-nE_b(R)}, \tag{1.51}$$

where $E_b(R)$ is a positive function of R for $R < C$. Work on a class of codes known as convolutional codes — to be introduced in Chapter 12 has shown (see, e.g., [471]) that

$$P(E) \leq 2^{-(m+1)nE_c(R)}, \tag{1.52}$$

where m is the memory of the code and $E_c(R)$ is positive for $R < C$. The problem, as we shall see (and what makes coding such a fascinating topic) is that, in the absence of some kind of structure, as either

n or m grow, the complexity can grow exponentially. How should this structure be introduced, to make the encoders and decoders tractable, while producing good error correction capability?

1.12.11 Non-Asymptotic Information Theory

The channel coding theorem described above is an *asymptotic* theory, asserting the existence of codes producing low probability of error, *as the code length n goes to infinity.* This asymptotic theory serves best when the code lengths are long. Recent technological developments, however, are pushing toward short codes. For example, in a cybercontrol system, near real-time response cannot wait around for a long codeword to be formed, so short codewords protecting brief data packets are needed to move data around in a responsive way. Thus, it becomes increasingly of interest to consider what the information-theoretic performance limits are for short codes. These issues are addressed by finite-block length, non-asymptotic information theory, briefly developed in this section.

A code (not necessarily linear) may be described by the parameters (n, M, ϵ), where:

- n is the length of the code;
- M is the number of codewords in the code;
- ϵ is the probability of error that the code achieves.

Since M messages can be indexed by $\log_2 M$ bits, the rate of the code (the number of bits to select a codeword divided by the codeword length) is

$$R = \frac{\log_2 M}{n}.$$

Expressed in this language, Shannon's channel coding theorem can be expressed as: there exists sequences of codes described by (n, M_n, ϵ_n) such that

$$R = \lim_{n\to\infty} \frac{\log_2 M_n}{n} > 0.$$

that is, there is a positive asymptotic rate, and

$$\epsilon_n \to 0.$$

Here, M_n is the maximum number of codewords associated with a code of length n, and ϵ_n is the probability of error associated with the code of length n.

Let

$$M^*(n, \epsilon) = \max\{M : \exists(n, M, \epsilon) - \text{code}\}$$

be the maximum number of codewords of length n such that it is possible to recover the original message with probability at least $1 - \epsilon$. $M^*(n, \epsilon)$ is basic to non-asymptotic information theory.

Shannon's asymptotic result can be expressed as

$$\lim_{\epsilon\to 0} \lim_{n\to\infty} \frac{1}{n} M^*(n, \epsilon) = C,$$

where C is the capacity. For any $0 < \epsilon < 1$, it is also true that

$$\lim_{n\to\infty} \frac{1}{n} M^*(n, \epsilon) \leq C.$$

Thus,

$$\log_2 M^*(n, \epsilon) = nC + o(n).$$

We define the *information density* as

$$i_{X;Y}(x;y) = \log \frac{P_{XY}(x,y)}{P_X(x)P_Y(y)}$$

The expectation of the information density is the mutual information,

$$E[i_{X;Y}(x;y)] = I(X;Y).$$

Under asymptotic information theory,

$$\lim_{n\to\infty} \frac{1}{n}\log_2 M^*(n,\epsilon) = E[i_{X;Y}(x,y)] = C.$$

Asymptotic information theory is thus seen to be a first-order theory, accounting for only the mean value of the information density.

1.12.11.1 Discrete Channels

Under non-asymptotic information theory, it can be shown [343, Section 3.2.2] that for a discrete channel (such as a BSC with crossover probability δ) the *Gaussian approximation* holds:

$$\frac{1}{n}\log_2 M^*(n,\epsilon) = C - \sqrt{\frac{V}{n}}Q^{-1}(\epsilon) + \frac{1}{2n}\log_2 n + \frac{1}{n}O(1). \tag{1.53}$$

Here, Q^{-1} is the inverse of the Q function defined by

$$Q(x) = \int_x^\infty \frac{1}{\sqrt{2\pi}} e^{-y^2/2}\, dy.$$

Also, V is called the *channel dispersion*, and is computed as the *variance* of the information density:

$$V = \mathrm{var}[i_{X;Y}(x,y)].$$

For the BSC, it can be shown by computation that

$$V = \delta(1-\delta)\Big(\log\frac{1-\delta}{\delta}\Big)^2,$$

so that

$$\frac{1}{n}\log_2 M^*(n,\epsilon) \approx C - \sqrt{\frac{\delta(1-\delta)}{n}}Q^{-1}(\epsilon)\log_2\frac{1-\delta}{\delta} + \frac{1}{2n}\log_2 n. \tag{1.54}$$

There are two other bounds that are interesting to compare with the Gaussian approximation. An upper bound is expressed as follows.

Theorem 1.29 *[343, Theorem 40, p. 51] For a BSC with crossover probability δ, the size M of an (n, M, ϵ) code is bounded by*

$$M \le \frac{1}{\beta^n_{1-\epsilon}}, \tag{1.55}$$

where $\beta^n_{1-\epsilon}$ is computed as follows.

- *Let $\alpha = 1 - \epsilon$.*
- *Determine the integer L and the number $\lambda \in [0, 1)$ such that*

$$\alpha = (1-\lambda)\alpha_L + \lambda\alpha_{L+1},$$

where

$$\alpha_\ell = \sum_{k=0}^{\ell-1} \binom{n}{k} (1-\delta)^{n-k}\delta^k.$$

(Since α_ℓ is an increasing function of ℓ, select L as the smallest value such that $\alpha_\ell > \alpha$, then λ will be within its bounds.)

- *Compute*

$$\beta^n_{1-\epsilon} = (1-\lambda)\beta_L + \lambda\beta_{L+1},$$

where

$$\beta_t = 2^{-n}\sum_{k=0}^{t}\binom{n}{k}.$$

The proof is provided in [343]. This theorem describes a non-asymptotic converse to the channel coding theorem: in order to achieve a given probability of error ϵ, the number of codewords in the code cannot exceed $1/\beta^n_{1-\epsilon}$. This provides an upper bound on the rate of the code.

Another bound is provided by the following.

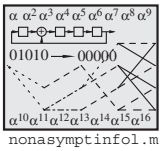
Theorem 1.30 *[343, Corollary 39, p. 49] For the BSC with crossover probability δ, there exists an (n, M, ϵ) code (where ϵ represents the average probability of error) such that*

$$\epsilon \le \sum_{t=0}^{n}\binom{n}{t}\delta^t(1-\delta)^{n-t}\min\left(1, (M-1)2^{-n}\sum_{s=0}^{t}\binom{n}{s}\right). \tag{1.56}$$

This is an existence theorem. For a given probability of error, codes with a small number of codewords M (low rate) could be used. But this theorem promises that there *exist* codes bounding the probability of error ϵ. This provides a lower bound on what values of M are possible.

To employ this as a bound, a value of M which achieves the bound (1.56) is found. In the plot below, this was done by a binary search on the value $\log_2 M/n$, starting with a bracketing solution of $M = 2$ and a value of M large enough that the right-hand side of (1.56) exceeds ϵ.

Figure 1.27 shows these bounds for a BSC with crossover probability $\delta = 0.11$ (giving a capacity $C = 1 - H(\delta) = 0.5$) and maximal block error $\epsilon = 10^{-3}$, compared with the capacity $C = 1 - H(\delta) = 0.5$. While (1.55) and (1.56) are actual bounds and not approximations, the plot shows that the Gaussian approximation (1.54) closely tracks these bounds, and has the benefit of being much easier to compute than the bounds. As the figure shows, there is a substantial gap between the asymptotic rate (Capacity $C = 0.5$) and the non-asymptotic rate — at lengths of $n = 2000$ bits, the rate achieving $\epsilon = 10^{-3}$ is only about 0.43, so only 88% of the asymptotic capacity is achievable even at $n = 2000$.



nonasymptinfol.m
qfinv.m
logncho osek.m

For the BEC channel, the Gaussian approximation is [343, Section 3.3.2]

$$\frac{1}{n}\log\ M^*(n,\epsilon) = (1-\delta)\log 2 - \sqrt{\frac{\delta(1-\delta)}{n}}Q^{-1}(\epsilon)\log 2 + \frac{1}{n}O(1). \tag{1.57}$$

1.12.11.2 The AWGN Channel

For the Gaussian channel (a continuous channel), with codewords $(c_1, c_2, \ldots, c_n)$ satisfying the power constraint

$$\sum_{i=1}^{n} c_i^2 = nP$$

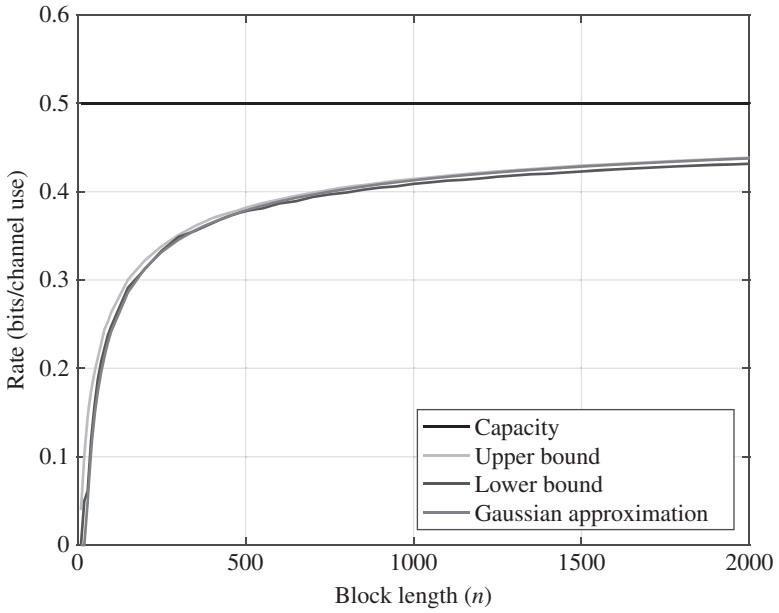


Figure 1.27: Rate-block length tradeoff for the BSC with $\delta = 0.11$ and maximal block error rate $\epsilon = 10^{-3}$.

having unit-covariance noise, let $M^*(n, \epsilon, P)$ denote the number of codewords in a code of length satisfying the power constraint. Shannon's (asymptotic) capacity is

$$\lim_{\epsilon \to 0} \lim_{n \to \infty} \frac{1}{n} \log\, M^*(n, \epsilon, P) = \frac{1}{2} \log(1 + P) \triangleq C(P).$$

For the non-asymptotic theory,

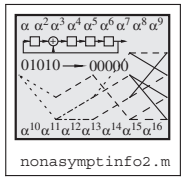
$$\frac{1}{n} M^*(n, \epsilon, P) \leq C(P) - \sqrt{\frac{V(P)}{n}} Q^{-1}(\epsilon) + \frac{1}{n} O(1).$$

Here, $V(P)$ is the channel dispersion

$$V(P) = \frac{P}{2} \frac{P+2}{(P+1)^2} (\log\ e)^2.$$

It can be shown [343] that this approximation closely tracks other bounds.

Figure 1.28 shows the asymptotic rate (capacity) compared with the Gaussian approximation. Again, the finite length effect is clear. Even with a codelength of $n = 2000$, only 88% of the channel capacity is achievable.



1.12.11.3 Comparison of Codes

Polyanskiy [343] suggests the use of M^* as a means of comparing different codes. For a code with M codewords, he defines (for the AWGN channel) the ratio

$$R_{\text{norm}}(\epsilon) = \frac{\log\, M}{\log\, M^*(n, \epsilon, \gamma_{\min}(\epsilon))},$$

where $\gamma_{\min}(\epsilon)$ is the smallest SNR at which the code will admit decoding with probability of error less than ϵ. Practically, $M^*(n, \epsilon, \gamma_{\min}(\epsilon))$ is approximated using (1.57) with negligible loss in accuracy.

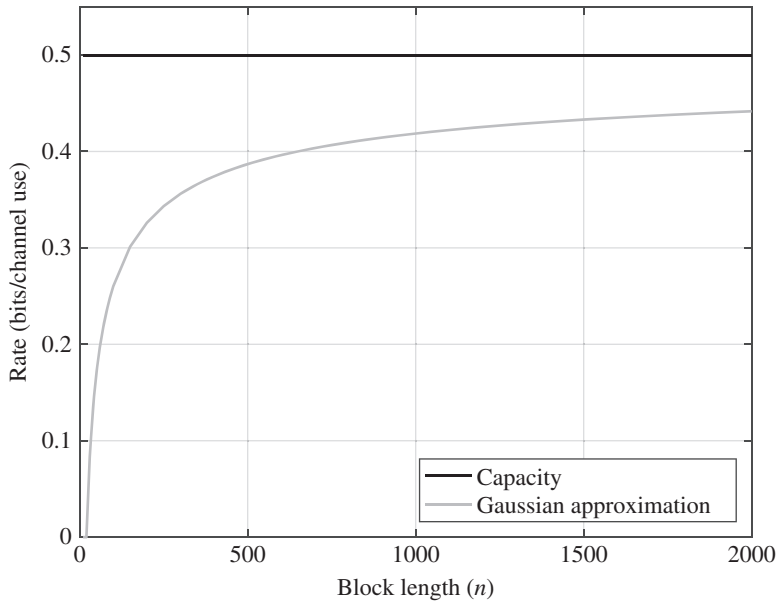


Figure 1.28: Rate-block length tradeoff for the Gaussian channel with $P = 1$ and maximal block error rate $\epsilon = 10^{-3}$.

"Better" codes should have $R_{\text{norm}}(\epsilon)$ which is near 1. A comparison of the sorts of codes discussed in this book is provided in [343].

Programming Laboratory 1: Simulating a Communications Channel

Objective

In this lab, you will simulate a BPSK communication system and a coded system with a Hamming code employing hard-input decoding rules.

Background

Reading: Sections 1.5, 1.7, 1.9.

In the case of BPSK, an exact expression for the probability of error is available, (1.25). However, in many more interesting communication systems, a closed-form expression for the probability of error is not available or is difficult to compute. Results must be therefore obtained by simulation of the system.

One of the great strengths of the signal-space viewpoint is that probability of error simulations can be made based only on points in the signal space. In other words, it suffices to simulate random variables as in the matched filter output (1.12), rather than creating the continuous-time functions as in (1.10). (However, for other kinds of questions, a simulation of the continuous-time function might be necessary. For example, if you are simulating the effect of synchronization, timing jitter, delay, or fading, simulating the time signal is probably necessary.)

A framework for simulating a communication system from the signal space point of view for the purpose of computing the probability of error is as follows:

Algorithm 1.2 Outline for Simulating Digital Communications

1 Initialization: Store the points in the signal constellation. Fix E_b (typically $E_b = 1$).
2 FOR each signal-to-noise ratio $\gamma = E_b/N_0$:
3 Compute $N_0 = E_b/\gamma$ and $\sigma^2 = N_0/2$.
4 DO:
5 Generate some random bit(s) (the ''transmitted'' bits). according to the bit probabilities
6 Map the bit(s) into the signal constellation (e.g., BPSK or 8-PSK) to create signal **s**.
7 Generate a Gaussian random vector **n** (noise) with variance $\sigma^2 = N_0/2$ in each signal direction.

8 Add the noise to the signal to create the matched filter output signal $\mathbf{r} = \mathbf{s} + \mathbf{n}$.
9 Perform a detection on the symbol. (e.g., find closest point in signal constellation to $\mathbf{r}$).
10 From the detected symbol, determine the detected bits.
11 Compare detected bits with the transmitted bits.
12 Accumulate the number of bits in error.
13 UNTIL at least N bit errors have been counted.
14 The estimated probability of error at this SNR is

$$P_e \approx \frac{\text{number of errors counted}}{\text{number of bits generated}}$$

15 End FOR

As a general rule, the more errors N you count, the smaller will be the variance of your estimate of the probability of error. However, the bigger N is, the longer the simulation will take to run. For example, if the probability of error is near 10^{-6} at some particular value of SNR, around 1 million bits must be generated before you can expect an error. If you choose $N = 100$, then 100 million bits must be generated to estimate the probability of error, for just that *one* point on the plot!

Use of Coding in Conjunction with the BSC

For an (n, k) code having rate $R = k/n$ transmitted with energy per bit equal to E_b, the energy per coded bit is $E_c = E_b R$. It is convenient to fix the coded energy per bit in the simulation. To simulate the BSC channel with coding, the following outline can be used.

Algorithm 1.3 Outline for Simulating (n, k)-coded Digital Communications

1 Initialization: Store the points in the signal constellation. Fix E_c (typically $E_c = 1$). Compute R.
2 FOR each signal-to-noise ratio $\gamma = E_b/N_0$:
3 Compute $N_0 = E_c/(R\gamma)$ and $\sigma^2 = N_0/2$.
4 Compute the BSC crossover probability $p = Q(\sqrt{2E_c/N_0})$.
5 DO:
6 Generate a block of k ''transmitted'' input bits and accumulate the number of bits generated
7 Encode the input bits to n codeword bits
8 Pass the n bits through the BSC (flip each bit with probability p)
9 Run the n bits through the decoder to produce k output bits
10 Compare the decoded output bits with the input bits
11 Accumulate the number of bits in error
12 UNTIL at least N bit errors have been counted.
13 The estimated probability of error is

$$P_e \approx \frac{\text{number of errors counted}}{\text{number of bits generated}}$$

14 End FOR

The encoding and decoding operations depend on the kind of code used. In this lab, you will use codes which are among the simplest possible, the Hamming codes.

Since for linear codes the codeword is irrelevant, the simulation can be somewhat simplified by assuming that the input bits are all zero so that the codeword is also all zero. For the Hamming code, the simulation can be arranged as follows:

Algorithm 1.4 Outline for Simulating (n, k) Hamming-coded Digital Communications

1 Fix E_c (typically $E_c = 1$). Compute R.
2 FOR each signal-to-noise ratio $\gamma = E_b/N_0$:
3 Compute $N_0 = E_c/(R\gamma)$ and $\sigma^2 = N_0/2$.
4 Compute the BSC crossover probability $p = Q(\sqrt{2E_c/N_0})$.
5 DO:
6 Generate $\mathbf{r}$ as a vector of n random bits which are 1 with probability p
7 Increment the number of bits generated by k.
8 Compute the syndrome $\mathbf{s} = \mathbf{r}H^T$.
9 If $\mathbf{s} \neq 0$, determine the error location based on the column of H which is equal to $\mathbf{s}$ and complement that bit of $\mathbf{r}$
10 Count the number of decoded bits (out of k) in $\mathbf{r}$ which match the all-zero message bits
11 Accumulate the number of bits in error.
12 UNTIL at least N bit errors have been counted.
13 Compute the probability of error.
14 End FOR

The **coding gain** for a coded system is the difference in the SNR required between uncoded and coded systems achieving the same probability of error. Usually the coding gain is expressed in dB.

Assignment

Preliminary Exercises

Show that if X is a random variable with mean 0 and variance 1 then

$$Y = aX + b$$

is a random variable with mean b and variance a^2.

Programming Part

BPSK Simulation

1. Write a program that will simulate a BPSK communication system with unequal prior bit probabilities. Using your program, create data from which to plot the probability of bit error obtained from your simulation for SNRs in the range from 0 to 10 dB, for the three cases that $P_0 = 0.5$ (in which case your

plot should look much like Figure 1.10), $P_0 = 0.25$, and $P_0 = 0.1$. Decide on an appropriate value of N.

2. Prepare data from which to plot the theoretical probability of error (1.24) for the same three values of P_0. (You may want to combine these first two programs into a single program.)
3. Plot the simulated probability of error on the same axes as the theoretical probability of error. The plots should have E_b/N_0 in dB as the horizontal axis and the probability as the vertical axis, plotted on a logarithmic scale (e.g., `semilogy` in MATLAB).
4. Compare the theoretical and simulated results. Comment on the accuracy of the simulation and the amount of time it took to run the simulation. Comment on the importance of theoretical models (where it is possible to obtain them).
5. Plot the probability of error for $P_0 = 0.1$, $P_0 = 0.25$ and $P_0 = 0.5$ on the same axes. Compare them and comment.

8-PSK Simulation

1. Write a program that will simulate an 8-PSK communication system with equal prior bit probabilities. Use a signal constellation in which the points are numbered in Gray code order. Make your program so that you can estimate both the symbol error probability and the bit error probability. Decide on an appropriate value of N.
2. Prepare data from which to plot the bound on the probability of symbol error P_s using (1.26) and probability of bit error P_b using (1.28).
3. Plot the simulated probability of symbol error and bit error on the same axes as the bounds on the probabilities of error.
4. Compare the theoretical and simulated results. Comment on the accuracy of the bound compared to the simulation and the amount of time it took to run the simulation.

Coded BPSK Simulation

1. Write a program that will simulate performance of the $(7, 4)$ Hamming code over a BSC channel with channel crossover probability $p = Q(\sqrt{2E_b/N_0})$ and plot the probability of error as a function of E_b/N_0 in dB. On the same plot, plot the theoretical probability of error for uncoded BPSK transmission. Identify what the coding gain is for a probability of error $P_b = 10^{-5}$.
2. Repeat this for a $(15, 11)$ Hamming code. (See page 112 and Equations (3.6) and (3.4).)

Resources and Implementation Suggestions

- A unit Gaussian random variable has mean 0 and variance 1. Given a unit Gaussian random variable, using the preliminary exercise, it is straightforward to generate a Gaussian random variable with any desired variance.

  The function `gran` provides a unit Gaussian random variable, generated using the Box–Muller transformation of two uniform random variables. The function `gran2` returns two unit Gaussian random variables. This is useful for simulations in two-dimensional signal constellations.
- There is nothing in this lab that makes the use of C++ imperative, as opposed to C. However, you may find it useful to use C++ in the following ways:
  (a) Create an AWGN class to represent a 1-D or 2-D channel.
  (b) Create a BSC class.
  (c) Create a Hamming code class to take care of encoding and decoding (as you learn more about coding algorithms, you may want to change how this is done).
  (d) In the literature, points in two-dimensional signal constellations are frequently represented as points in the complex plane. You may find it convenient to do similarly, using the complex number capabilities that are present in C++.
- Since the horizontal axis of the probability of error plot is expressed as a ratio E_b/N_0, there is some flexibility in how to proceed. Given a value of E_b/N_0, you can either fix N_0 and determine E_b, or you can fix E_b and determine N_0. An example of how this can be done is in `testrepcode.cc`.
- The function `uran` generates a uniform random number between 0 and 1. This can be used to generate a bit which is 1 with probability p.
- The Q function, used to compute the theoretical probability of error, is implemented in the function `qf`.
- There are two basic approaches to generating the sequence of bits in the simulation. One way is to generate and store a large array of bits (or their resulting signals) then processing them all together. This is effective in a language such as MATLAB, where vectorized operations are faster than using

`for` loops. The other way, and the way recommended here, is to generate each signal separately and to process it separately. This is recommended because it is not necessarily known in advance how many bits should be generated. The number of bits to be generated could be extremely large — in the millions or even billions when the probability of error is small enough.

- For the Hamming encoding and decoding operation, vector/matrix multiply operations over GF(2) are required, such as $\mathbf{c} = \mathbf{m}G$. ($GF(2)$ is addition/subtraction/multiplication/division modulo 2.) These could be done in the conventional way using nested `for` loops. However, for short binary codes, a computational simplification is possible. Write G in terms of its columns as

$$G = \left[\mathbf{g}_1 \ \mathbf{g}_2 \ \cdots \ \mathbf{g}_n\right].$$

Then the encoding process can be written as a series of vector/vector products (inner products)

$$\begin{aligned}\mathbf{c} &= [c_1, c_2, \ldots, c_n] \\ &= \left[\mathbf{mg}_1 \ \mathbf{mg}_2 \ \cdots \ \mathbf{mg}_n\right].\end{aligned}$$

Let us consider the inner product operation: it consists of element-by-element multiplication, followed by a sum.

Let m be an integer variable, whose bits represent the elements of the message vector $\mathbf{m}$. Also, let $g[i]$ be an integer variable in C whose bits represent the elements of the column $\mathbf{g}_i$. Then the element-by-element multiplication involved in the product $\mathbf{mg}_i$ can be written simply using the bitwise-and operator `&` in C. How, then, to sum up the elements of the resulting vector? One way, of course, is to use a `for` loop, such as:

```
// Compute c=m*G, where m is a
// bit-vector, and G is represented by
// g[i]
c = 0;   // set vector of bits to 0
for(i = 0; i < n; i++) {
   mg = m & g[i];
   // mod-2 multiplication
   // of all elements
   bitsum=0;
   for(j = 0, mask=1; j < n; j++) {
   // mask selects a single bit
      if(mg & mask) {
         bitsum++;
         // accumulate if the bit != 0
      }
      mask «= 1;
      // shift mask over by 1 bit
   }
   bitsum = bitsum % 2; // mod-2 sum
   c = c | bitsum*(1«i);
   // assign to vector of bits ...
}
```

However, for sufficiently small codes (such as in this assignment) the inner `for` loop can be eliminated by *precomputing* the sums. Consider the table below. For a given number m, the last column provides the sum of all the bits in m, modulo 2.

m	m (binary)	$\sum m$	$s[m] = \sum m \pmod 2$
0	0000	0	0
1	0001	1	1
2	0010	1	1
3	0011	2	0
4	0100	1	1
5	0101	2	0
6	0110	2	0
7	0111	3	1
8	1000	1	1
9	1001	2	0
10	1010	2	0
11	1011	3	1
12	1100	2	0
13	1101	3	1
14	1110	3	1
15	1111	4	0

To use this in a program, precompute the table of bit sums, then use this to look up the result. An outline follows:

```
// Compute the table s, having all
// the bit sums modulo 2
// ...

// Compute c=m*G, where
// m is a bit-vector, and
// G is represented by g[i]
c = 0;
for(i = 0; i < n; i++) {
   c = c | s[m & g[i]]*(1«i);
   // assign to vector of bits
}
```

1.13 Exercises

1.1 Weighted codes. Let $s_1, s_2, \ldots, s_n$ be a sequence of digits, each in the range $0 \leq s_i < p$, where p is a prime number. The weighted sum is

$$W = ns_1 + (n-1)s_2 + (n-2)s_3 + \cdots + 2s_{n-1} + s_n.$$

The final digit s_n is selected so that W modulo p is equal to 0. That is, $W \equiv 0 \pmod{p}$. W is called the checksum.

(a) Show that the weighted sum W can be computed by computing the cumulative sum sequence $t_1, t_2, \ldots, t_n$ by

$$t_1 = s_1, t_2 = s_1 + s_2, \ldots, t_n = s_1 + s_2 + \cdots + s_n$$

then computing the cumulative sum sequence

$$w_1 = t_1, w_2 = t_1 + t_2, \ldots, w_n = t_1 + t_2 + \cdots + t_n,$$

with $W = w_n$.

(b) Suppose that the digits s_k and s_{k+1} are interchanged, with $s_k \neq s_{k+1}$, and then a new checksum W' is computed. Show that if the original sequence satisfies $W \equiv 0 \pmod{p}$, then the modified sequence cannot satisfy $W' \equiv 0 \pmod{p}$. Thus, interchanged digits can be detected.

(c) For a sequence of digits of length $< p$, suppose that digit s_k is altered to some $s'_k \neq s_k$, and a new checksum W' is computed. Show that if the original sequence satisfies $W \equiv 0 \pmod{p}$, then the modified sequence cannot satisfy $W' \equiv 0 \pmod{p}$. Thus, a single modified digit can be detected. Why do we need the added restriction on the length of the sequence?

(d) See if the ISBN 0-13-139072-4 is valid.

(e) See if the ISBN 0-13-193072-4 is valid.

1.2 See if the UPCs 0 59280 00020 0 and 0 41700 00037 9 are valid.

1.3 A coin having $P(\text{head}) = 0.001$ is tossed 10,000 times, each toss independent. What is the lower limit on the number of bits it would take to accurately describe the outcomes? Suppose it were possible to send only 100 bits of information to describe all 10,000 outcomes. What is the minimum average distortion per bit that must be accrued sending the information in this case?

1.4 Show that the entropy of a source X with M outcomes described by (1.1) is maximized when all the outcomes are equally probable: $p_1 = p_2 = \cdots = p_M$.

1.5 Show that (1.7) follows from (1.5) using (1.4).

1.6 Show that (1.12) is true and that the mean and variance of N_{1i} and N_{2i} are as in (1.13) and (1.14).

1.7 Show that the decision rule and threshold in (1.19) and (1.20) are correct.

1.8 Show that (1.24) is correct.

1.9 Show that if X is a random variable with mean 0 and variance 1 then $Y = aX + b$ is a random variable with mean b and variance a^2.

1.10 Show that the detection rule for 8-PSK

$$\hat{\mathbf{s}} = \arg\max_{\mathbf{s} \in \mathcal{S}} \mathbf{r}^T \mathbf{s}$$

follows from (1.18) when all points are equally likely.

1.11 Consider a series of M BSCs, each with transition probability p, where the outputs of each BSC is connected to the inputs of the next in the series. Show that the resulting overall channel is a BSC and determine the crossover probability as a function of M. What happens as $M \to \infty$? *Hint*: To simplify, consider the difference of $(x + y)^n$ and $(x - y)^n$.

1.12 [319] **Bounds and approximations to the Q function**. For many analyses it is useful to have analytical bounds and approximations to the Q function. This exercise introduces some of the most important of these.

(a) Show that

$$\sqrt{2\pi}Q(x) = \frac{1}{x}e^{-x^2/2} - \int_x^\infty \frac{1}{y^2}e^{-y^2/2}dy \quad x > 0.$$

Hint: integrate by parts.

(b) Show that

$$0 < \int_x^\infty \frac{1}{y^2}e^{-y^2/2}\,dy < \frac{1}{x^3}e^{-x^2/2}.$$

(c) Hence conclude that

$$\frac{1}{\sqrt{2\pi}x}e^{-x^2/2}(1 - 1/x^2) < Q(x) < \frac{1}{\sqrt{2\pi}x}e^{-x^2/2} \quad x > 0.$$

(d) Plot these lower and upper bounds on a plot with $Q(x)$ (use a log scale).

(e) Another useful bound is $Q(x) \leq \frac{1}{2}e^{-x^2/2}$. Derive this bound. *Hint*: Identify $[Q(\alpha)]^2$ as the probability that the zero-mean unit-Gaussian random variables lie in the shaded region shown on the left in Figure 1.29, (the region $[\alpha, \infty) \times [\alpha, \infty)$). This probability is exceeded by the probability that (x, y) lies in the shaded region shown on the right (extended out to ∞). Evaluate this probability.

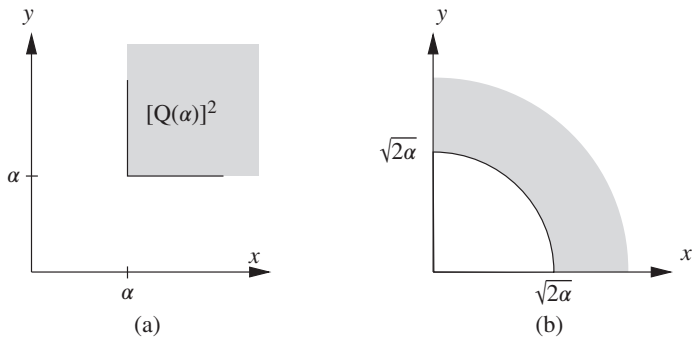


Figure 1.29: Regions for bounding the Q function.

1.13 Let $V_2(n, t)$ be the number of points in a Hamming sphere of "radius" t around a binary codeword of length n. That is, it is the number of points within a Hamming distance t of a binary vector. Determine a formula for $V_2(n, t)$.

1.14 Show that the Hamming distance satisfies the triangle inequality. That is, for three binary vectors $\mathbf{x}$, $\mathbf{y}$, and $\mathbf{z}$ of length n, show that

$$d_H(\mathbf{x}, \mathbf{z}) \leq d_H(\mathbf{x}, \mathbf{y}) + d_H(\mathbf{y}, \mathbf{z}).$$

1.15 Show that for BPSK modulation with amplitudes $\pm\sqrt{E_c}$, the Hamming distance d_H and the Euclidean distance d_E between a pair of codewords are related by $d_E = 2\sqrt{E_c d_H}$.

1.16 In this problem, we will demonstrate that the probability of error for a repetition code decreases exponentially with the code length. Several other useful facts will also be introduced by this problem.

(a) Show that

$$2^{-nH_2(p)} = (1-p)^n \left(\frac{p}{1-p}\right)^{np}.$$

(b) For the fact

$$\text{If } 0 \le p \le \frac{1}{2} \text{ then } \sum_{0 \le i \le pn} \binom{n}{i} \le 2^{nH_2(p)},$$

justify the following steps of its proof

$$\begin{aligned} 1 = (p + (1-p))^n &\ge \sum_{0 \le i \le pn} \binom{n}{i} p^i (1-p)^{n-i} \\ &\ge \sum_{0 \le i \le pn} \binom{n}{i} (1-p)^n \left(\frac{p}{1-p}\right)^{pn} \\ &= 2^{-nH_2(p)} \sum_{0 \le i \le pn} \binom{n}{i}. \end{aligned}$$

(c) Show that the probability of error for a repetition code can be written as

$$P_e^n = \sum_{j=0}^{n-(t+1)} \binom{n}{j} (1-p)^j p^{(n-j)},$$

where $t = \lfloor (n-1)/2 \rfloor$.

(d) Show that

$$P_e^n \le [2\sqrt{p(1-p)}]^n$$

1.17 [292, p. 14] Identities on $\binom{n}{k}$. We can define

$$\binom{x}{m} = \begin{cases} \frac{x(x-1)(x-2)\cdots(x-m+1)}{m!} & \text{if } m \text{ is a positive integer} \\ 1 & \text{if } m = 0 \\ 0 & \text{otherwise.} \end{cases}$$

Show that

(a) $\binom{n}{k} = \frac{n!}{k!(n-k)!}$ if k is a nonnegative integer.

(b) $\binom{n}{k} = 0$ if n is an integer and $k > n$ is a nonzero integer.

(c) $\binom{n}{k} + \binom{n}{k-1} = \binom{n+1}{k}$.

(d) $(-1)^k \binom{-n}{k} = \binom{n+k-1}{k}$.

(e) $\sum_{k=0}^{n} \binom{n}{k} = 2^n$.

(f) $\sum_{k \text{ even}} \binom{n}{k} = \sum_{k \text{ odd}} \binom{n}{k} = 2^{n-1}$ if $n \ge 1$.

(g) $\sum_{k=0}^{n} (-1)^k \binom{n}{k} = 0$ if $n \ge 1$.

1.18 Show that for soft-decision decoding on the $(n, 1)$ repetition code, (1.33) is correct.

1.19 For the $(n, 1)$ code used over a BSC with crossover probability p, what is the probability that an error event occurs which is not detected?

1.20 Hamming code decoding.

(a) For G in (1.34) and H in (1.37), verify that $GH^T = \mathbf{0}$. (Recall that operations are computed modulo 2.)

(b) Let $\mathbf{m} = [1, 1, 0, 0]$. Determine the transmitted Hamming codeword when the generator of (1.34) is used.

(c) Let $\mathbf{r} = [1, 1, 1, 1, 1, 0, 0]$. Using Algorithm 1.1, determine the transmitted codeword $\mathbf{c}$. Also determine the transmitted message $\mathbf{m}$.

(d) The message $\mathbf{m} = [1, 0, 0, 1]$ is encoded to form the codeword $\mathbf{c} = [1, 1, 0, 0, 1, 0, 1]$. The vector $\mathbf{r} = [1, 0, 1, 0, 1, 0, 0]$ is received. Decode $\mathbf{r}$ to obtain $\hat{\mathbf{c}}$. Is the codeword $\hat{\mathbf{c}}$ found the same as the original $\mathbf{c}$? Why or why not?

1.21 For the $(7, 4)$ Hamming code generator polynomial $g(x) = 1 + x + x^3$, generate all possible code polynomials $c(x)$. Verify that they correspond to the codewords in (1.35). Take a nonzero codeword $c(x)$ and compute $c(x)h(x)$ modulo $x^7 + 1$. Do this also for two other nonzero codewords. What is the check condition for this code?

1.22 Is it possible that the polynomial $g(x) = x^4 + x^3 + x^2 + 1$ is a generator polynomial for a cyclic code of length $n = 7$?

1.23 For the parity check matrix

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

draw the Wolf trellis and the Tanner graph.

1.24 Let X be a random variable taking on the values $\mathcal{A}_x = \{a, b, c, d\}$ with probabilities

$$P(X = a) = \frac{1}{2} \quad P(X = b) = \frac{1}{4} \quad P(X = c) = \frac{1}{8} \quad P(X = d) = \frac{1}{8}.$$

Determine $H(X)$. Suppose that 100 measurements of independent draws of X are made per second. Determine what the entropy rate of this source is. Determine how to encode the X data to achieve this rate.

1.25 Show that the information inequality $\log x \leq x - 1$ is true.

1.26 Show that for a discrete random variable X, $H(X) \geq 0$.

1.27 Show that $I(X; Y) \geq 0$ and that $I(X; Y) = 0$ only if X and Y are independent. *Hint*: Use the information inequality.

1.28 Show that the formulas $I(X; Y) = H(X) - H(X|Y)$ and $I(X; Y) = H(Y) - H(Y|X)$ follow from the Definition (1.38).

1.29 Show that $H(X) \geq H(X|Y)$. *Hint*: Use the previous two problems.

1.30 Show that the mutual information $I(X; Y)$ can be written as

$$I(X; Y) = \sum_{x \in \mathcal{A}_x} P_X(x) \sum_{y \in \mathcal{A}_y} P_{Y|X}(y|x) \log_2 \frac{P_{Y|X}(y|x)}{\sum_{x' \in \mathcal{A}_x} P_X(x') P_{Y|X}(y|x')}$$

1.31 For a BSC with crossover probability p having input X and output Y, let the probability of the inputs be $P(X = 0) = q$ and $P(X = 1) = 1 - q$.

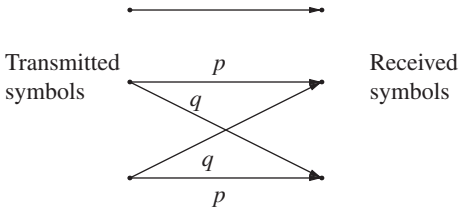
(a) Show that the mutual information is

$$I(X; Y) = H(Y) + p \log_2 p + (1 - p) \log_2 (1 - p).$$

(b) By maximizing over q show that the channel capacity per channel use is

$$C = 1 - H_2(p) \text{ (bits)}.$$

1.32 Consider the channel model shown here, which accepts three different symbols.



The first symbol is not affected by noise, while the second and third symbols have a probability p of not being corrupted, and a probability q of being changed into the other of the pair. Let $\alpha = -p \log p - q \log q$, and let P be the probability that the first symbol is chosen and let Q be the probability that either of the other two is chosen, so that $P + 2Q = 1$.

(a) Show that $H(X) = -P \log P - 2Q \log Q$.

(b) Show that $H(X|Y) = 2Q\alpha$.

(c) Choose the input distribution (i.e., choose P and Q) in such a way to maximize $I(X;Y) = H(X) - H(X|Y)$) subject to $P + 2Q = 1$. What is the capacity for this channel?

1.33 Let $X \sim \mathcal{U}(-a, a)$ (that is, X is uniformly distributed on $[-a, a]$). Compute $H(X)$. Compare $H(X)$ with the entropy of a Gaussian distribution having the same variance.

1.34 Let $g(x)$ denote the pdf of a random variable X with variance σ^2. Show that

$$H(X) \leq \frac{1}{2} \log_2 2\pi e\sigma^2.$$

with equality if and only if X is Gaussian. *Hint*: Let $p(x)$ denote the pdf of a Gaussian r.v. with variance σ^2 and consider $D(g||p)$. Also, note that $\log p(x)$ is quadratic in x.

1.35 Show that $H(X + N|X) = H(N)$.

1.14 References

The information age was heralded with Shannon's work [404]. Thorough coverage of information theory appears in [82], [147] and [498]. The books [300] and [471] place coding theory in its information theoretic context. Our discussion of the AEP follows [26], while our "proof" of the channel coding theorem closely follows Shannon's original [404]. More analytical proofs appear in the textbooks cited above. See also [458]. Discussion about tradeoffs with complexity are in [376], as is the discussion in Section 1.12.9. Non-asymptotic information theory is developed in the masterful dissertation [343].

The detection theory and signal space background is available in most books on digital communication. See, for example, [26, 319, 344, 354].

Hamming codes were presented in [181]. The trellis representation was presented first in [488]; a thorough treatment of the concept appears in [273]. The Tanner graph representation appears in [432]; see also [148]. Exercise 1.16b comes from [458, p. 21].

The discussion relating to simulating communication systems points out that such simulations can be very slow. Faster results can in some cases be obtained using *importance sampling*. Some references on importance sampling are [123, 279, 403].