Chapter 1

Introduction

The advent of cheap high-speed global communications ranks as one of the most important developments of human civilization in the second half of the twentieth century.

In 1950, an international telephone call was a remarkable event, and black-and-white television was just beginning to become widely available. By 2000, in contrast, an intercontinental phone call could often cost less than a postcard, and downloading large files instantaneously from anywhere in the world had become routine. The effects of this revolution are felt in daily life from Boston to Berlin to Bangalore.

Underlying this development has been the replacement of analog by digital communications.

Before 1948, digital communications had hardly been imagined. Indeed, Shannon's 1948 paper [7] may have been the first to use the word "bit."[1]

Even as late as 1988, the authors of an important text on digital communications [5] could write in their first paragraph:

> Why would [voice and images] be transmitted digitally? Doesn't digital transmission squander bandwidth? Doesn't it require more expensive hardware? After all, a voice-band data modem (for digital transmission over a telephone channel) costs ten times as much as a telephone and (in today's technology) *is incapable of transmitting voice signals* with quality comparable to an ordinary telephone [authors' emphasis]. This sounds like a serious indictment of digital transmission for analog signals, but for most applications, the advantages outweigh the disadvantages ...

But by their second edition in 1994 [6], they were obliged to revise this passage as follows:

> Not so long ago, digital transmission of voice and video was considered wasteful of bandwidth, and the cost ... was of concern. [More recently, there has been] a complete turnabout in thinking ... In fact, today virtually all communication is either already digital, in the process of being converted to digital, or under consideration for conversion.

[1] Shannon explains that "bit" is a contraction of "binary digit," and credits the neologism to J. W. Tukey.

The most important factor in the digital communications revolution has undoubtedly been the staggering technological progress of microelectronics and optical fiber technology. For wireline and wireless radio transmission (but not optical), another essential factor has been progress in channel coding, data compression and signal processing algorithms. For instance, data compression algorithms that can encode telephone-quality speech at 8–16 kbps and voiceband modem algorithms that can transmit 40–56 kbps over ordinary telephone lines have become commodities that require a negligible fraction of the capacity of today's personal-computer microprocessors.

This book attempts to tell the channel coding part of this story. In particular, it focusses on coding for the point-to-point additive white Gaussian noise (AWGN) channel. This choice is made in part for pedagogical reasons, but also because in fact almost all of the advances in practical channel coding have taken place in this arena. Moreover, performance on the AWGN channel is the standard benchmark for comparison of different coding schemes.

1.1 Shannon's grand challenge

The field of information theory and coding has a unique history, in that many of its ultimate limits were determined at the very beginning, in Shannon's founding paper [7].

Shannon's most celebrated result is his channel capacity theorem, which we will review in Chapter 3. This theorem states that for many common classes of channels there exists a channel capacity C such that there exist codes at any rate $R < C$ that can achieve arbitrarily reliable transmission, whereas no such codes exist for rates $R > C$. For a band-limited AWGN channel, the capacity C in bits per second (b/s) depends on only two parameters, the channel bandwidth W in Hz and the signal-to-noise ratio SNR, as follows:

$$C = W \log_2(1 + \text{SNR}) \quad \text{b/s}.$$

Shannon's theorem has posed a magnificent challenge to succeeding generations of researchers. Its proof is based on randomly chosen codes and optimal (maximum likelihood) decoding. In practice, it has proved to be remarkably difficult to find classes of constructive codes that can be decoded by feasible decoding algorithms at rates which come at all close to the Shannon limit. Indeed, for a long time this problem was regarded as practically insoluble. Each significant advance toward this goal has been awarded the highest accolades the coding community has to offer, and most such advances have been immediately incorporated into practical systems.

In the next two sections we give a brief history of these advances for two different practical channels: the deep-space channel and the telephone channel. The deep-space channel is an unlimited-bandwidth, power-limited AWGN channel, whereas the telephone channel is very much bandwidth-limited. (We realize that many of the terms used here may be unfamiliar to the reader at this point, but we hope that these surveys will give at least an impressionistic picture. After reading later chapters, the reader may wish to return to reread these sections.)

Within the past decade there have been remarkable breakthroughs, principally the invention of turbo codes [1] and the rediscovery of low-density parity check (LDPC) codes [4], which have allowed the capacity of AWGN and similar channels to be approached in a practical sense. For example, Figure 1 (from [2]) shows that an optimized rate-1/2 LDPC code on an AWGN channel can approach the relevant Shannon limit within 0.0045 decibels (dB) in theory, and within 0.04 dB with an arguably practical code of block length 10^7 bits. Practical systems using block lengths of the order of 10^4–10^5 bits now approach the Shannon limit within tenths of a dB.

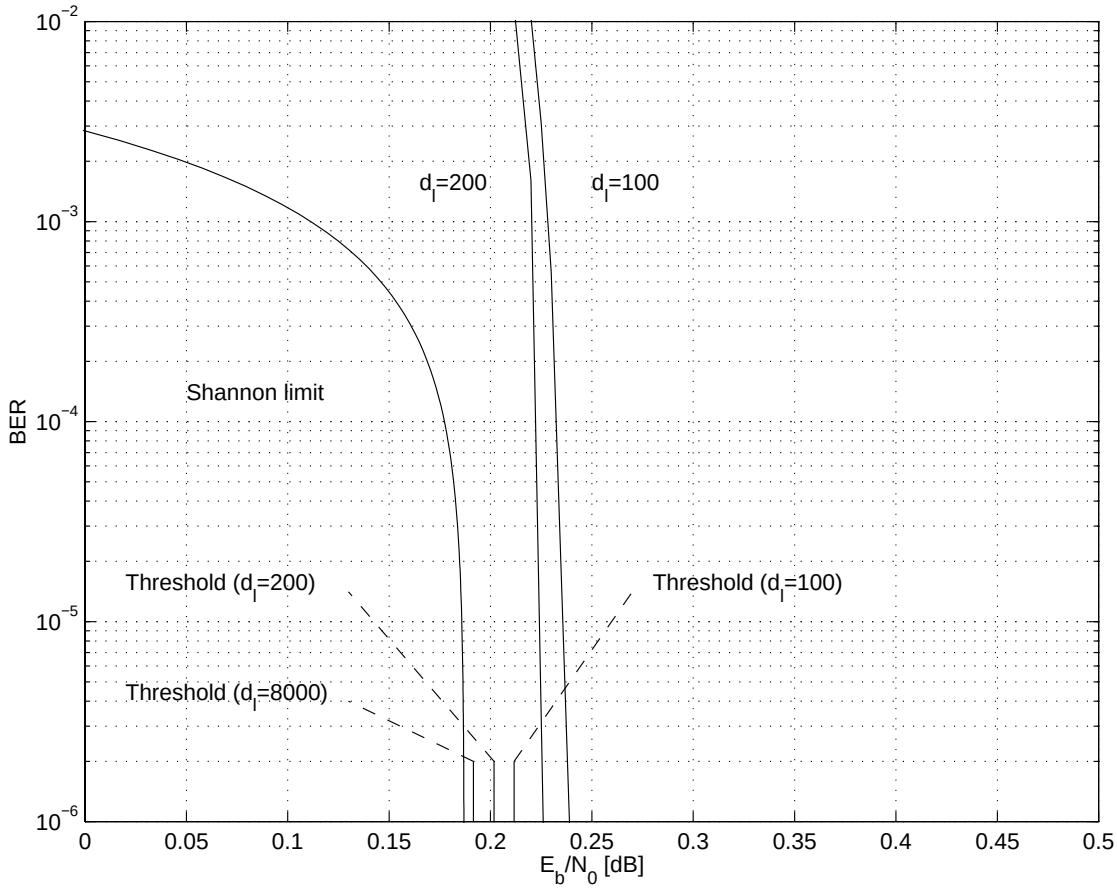


Figure 1. Bit error rate *vs.* E_b/N_0 in dB for optimized irregular rate-1/2 binary LDPC codes with maximum left degree d_l. Threshold: theoretical limit as block length $\to \infty$. Solid curves: simulation results for block length $= 10^7$. Shannon limit: binary codes, $R = 1/2$. (From [2].)

Here we will tell the story of how Shannon's challenge has been met for the AWGN channel, first for power-limited channels, where binary codes are appropriate, and then for bandwidth-limited channels, where multilevel modulation must be used. We start with the simplest schemes and work up to capacity-approaching codes, which for the most part follows the historical sequence.

1.2 Brief history of codes for deep-space missions

The deep-space communications application has been the arena in which most of the most powerful coding schemes for the power-limited AWGN channel have been first deployed, because:

- The only noise is AWGN in the receiver front end;
- Bandwidth is effectively unlimited;
- Fractions of a dB have huge scientific and economic value;
- Receiver (decoding) complexity is effectively unlimited.

For power-limited AWGN channels, we will see that there is no penalty to using binary codes with binary modulation rather than more general modulation schemes.

The first coded scheme to be designed was a simple $(32, 6, 16)$ biorthogonal code for the Mariner missions (1969), decoded by efficient maximum-likelihood decoding (the fast Hadamard transform, or "Green machine;" see Exercise 2, below). We will see that such a scheme can achieve a nominal coding gain of 3 (4.8 dB). At a target error probability per bit of $P_b(E) \approx 5 \cdot 10^{-3}$, the actual coding gain achieved was only about 2.2 dB.

The first coded scheme actually to be launched was a rate-1/2 convolutional code with constraint length $\nu = 20$ for the Pioneer 1968 mission. The receiver used 3-bit soft decisions and sequential decoding implemented on a general-purpose 16-bit minicomputer with a 1 MHz clock rate. At 512 b/s, the actual coding gain achieved at $P_b(E) \approx 5 \cdot 10^{-3}$ was about 3.3 dB.

During the 1970's, the NASA standard became a concatenated coding scheme based on a $\nu = 6$, rate-1/3 inner convolutional code and a $(255, 223, 33)$ Reed-Solomon outer code over $\mathbb{F}_{256}$. Such a system can achieve a real coding gain of about 8.3 dB at $P_b(E) \approx 10^{-6}$.

When the primary antenna failed to deploy on the Galileo mission (*circa* 1992), an elaborate concatenated coding scheme using a $\nu = 14$ rate-1/4 inner code with a Big Viterbi Decoder (BVD) and a set of variable-strength RS outer codes was reprogrammed into the spacecraft computers. This scheme was able to operate at $E_b/N_0 \approx 0.8$ dB at $P_b(E) \approx 2 \cdot 10^{-7}$, for a real coding gain of about 10.2 dB.

Turbo coding systems for deep-space communications have been developed by NASA's Jet Propulsion Laboratory (JPL) and others to get within 1 dB of the Shannon limit, and have now been standardized.

For a more comprehensive history of coding for deep-space channels, see [3].

1.3 Brief history of telephone-line modems

For several decades the telephone channel was the arena in which the most powerful coding and modulation schemes for the bandwidth-limited AWGN channel were first developed and deployed, because:

- The telephone channel is fairly well modeled as a band-limited AWGN channel;
- One dB has a significant commercial value;
- Data rates are low enough that a considerable amount of processing can be done per bit.

To approach the capacity of bandwidth-limited AWGN channels, multilevel modulation must be used. Moreover, it is important to use as much of the available bandwidth as possible.

The earliest modems developed in the 1950s and 1960s (Bell 103 and 202, and international standards V.21 and V.23) used simple binary frequency-shift keying (FSK) to achieve data rates of 300 and 1200 b/s, respectively. Implementation was entirely analog.

The first synchronous "high-speed" modem was the Bell 201 (later V.24), a 2400 b/s modem which was introduced about 1962. This modem used four-phase (4-PSK) modulation at 1200 symbols/s, so the nominal (Nyquist) bandwidth was 1200 Hz. However, because the modulation pulse had 100% rolloff, the actual bandwidth used was closer to 2400 Hz.

The first successful 4800 b/s modem was the Milgo 4400/48 (later V.27), which was introduced about 1967. This modem used eight-phase (8-PSK) modulation at 1600 symbols/s, so the nominal (Nyquist) bandwidth was 1600 Hz. "Narrow-band" filters with 50% rolloff kept the actual bandwidth used to 2400 Hz.

The first successful 9600 b/s modem was the Codex 9600C (later V.29), which was introduced in 1971. This modem used quadrature amplitude modulation (QAM) at 2400 symbols/s with an unconventional 16-point signal constellation (see Exercise 3, below) to combat combined "phase jitter" and AWGN. More importantly, it used digital adaptive linear equalization to keep the actual bandwidth needed to not much more than the Nyquist bandwidth of 2400 Hz.

All of these modems were designed for private point-to-point conditioned voice-grade lines, which use four-wire circuits (independent transmission in each direction) whose quality is higher and more consistent than that of the typical telephone connection in the two-wire (simultaneous transmission in both directions) public switched telephone network (PSTN).

The first international standard to use coding was the V.32 standard (1986) for 9600 b/s transmission over the PSTN (later raised to 14.4 kb/s in V.32*bis*). This modem used an 8-state, two-dimensional (2D) rotationally invariant Wei trellis code to achieve a coding gain of about 3.5 dB with a 32-QAM (later 128-QAM) constellation at 2400 symbols/s, again with an adaptive linear equalizer. Digital echo cancellation was also introduced to combat echoes on two-wire channels.

The "ultimate modem standard" was V.34 (1994) for transmission at up to 28.8 kb/s over the PSTN (later raised to 33.6 kb/s in V.34*bis*). This modem used a 16-state, 4D rotationally invariant Wei trellis code to achieve a coding gain of about 4.0 dB with a variable-sized QAM constellation with up to 1664 points. An optional 32-state, 4D trellis code with an additional coding gain of 0.3 dB and four times (4x) the decoding complexity and a 64-state, 4D code with a further 0.15 dB coding gain and a further 4x increase in complexity were also provided. A 16D "shell mapping" constellation shaping scheme provided an additional shaping gain of about 0.8 dB (see Exercise 4, below). A variable symbol rate of up to 3429 symbols/s was used, with symbol rate and data rate selection determined by "line probing" of individual channels. Nonlinear transmitter precoding combined with adaptive linear equalization in the receiver was used for equalization, again with echo cancellation. In short, this modem used almost every tool in the AWGN channel toolbox.

However, this standard was shortly superseded by V.90 (1998). V.90 is based on a completely different, non-AWGN model for the telephone channel: namely, it recognizes that within today's PSTN, analog signals are bandlimited, sampled and quantized to one of 256 amplitude levels at 8 kHz, transmitted digitally at 64 kb/s, and then eventually reconstructed by pulse amplitude modulation (PAM). By gaining direct access to the 64 kb/s digital data stream at a central site, and by using a well-spaced subset of the pre-existing nonlinear 256-PAM constellation, data can easily be transmitted at 40–56 kb/s (see Exercise 5, below). In V.90, such a scheme is used for downstream transmission only, with V.34 modulation upstream. In V.92 (2000) this scheme has been extended to the more difficult upstream direction.

Neither V.90 nor V.92 uses coding, nor the other sophisticated techniques of V.34. In this sense, the end of the telephone-line modem story is a bit of a fizzle. However, techniques similar to those of V.34 are now used in higher-speed wireline modems, such as digital subscriber line (DSL) modems, as well as on wireless channels such as digital cellular. In other words, the story continues in other settings.

1.4 Exercises

In this section we offer a few warm-up exercises to give the reader some preliminary feeling for data communication on the AWGN channel.

In these exercises the underlying channel model is assumed to be a discrete-time AWGN channel whose output sequence is given by $\mathbf{Y} = \mathbf{X} + \mathbf{N}$, where $\mathbf{X}$ is a real input data sequence and $\mathbf{N}$ is a sequence of real independent, identically distributed (iid) zero-mean Gaussian noise variables. This model will be derived from a continuous-time model in Chapter 2.

We will also give the reader some practice in the use of decibels (dB). In general, a dB representation is useful wherever logarithms are useful; *i.e.,* wherever a real number is a multiplicative factor of some other number, and particularly for computing products of many factors. The dB scale is simply the logarithmic mapping

$$\text{ratio or multiplicative factor of } \alpha \leftrightarrow 10 \log_{10} \alpha \text{ dB},$$

where the scaling is chosen so that the decade interval 1–10 maps to the interval 0–10. (In other words, the value of α in dB is $\log_\beta \alpha$, where $\beta = 10^{0.1} = 1.2589....$) This scale is convenient for human memory and calculation. It is often useful to have the little log table below committed to memory, even in everyday life (see Exercise 1, below).

α	dB (round numbers)	dB (two decimal places)
1	0	0.00
1.25	1	0.97
2	3	3.01
2.5	4	3.98
e	4.3	4.34
3	4.8	4.77
π	5	4.97
4	6	6.02
5	7	6.99
8	9	9.03
10	10	10.00

Exercise 1. (Compound interest and dB) How long does it take to double your money at an interest rate of P%? The bankers' "Rule of 72" estimates that it takes about $72/P$ years; *e.g.,* at a 5% interest rate compounded annually, it takes about 14.4 years to double your money.

(a) An engineer decides to interpolate the dB table above linearly for $1 \le 1 + p \le 1.25$; *i.e.,*

$$\text{ratio or multiplicative factor of } 1 + p \leftrightarrow 4p \text{ dB}.$$

Show that this corresponds to a "Rule of 75;" *e.g.,* at a 5% interest rate compounded annually, it takes 15 years to double your money.

(b) A mathematician linearly approximates the dB table for $p \approx 0$ by noting that as $p \to 0$, $\ln(1+p) \to p$, and translates this into a "Rule of N" for some real number N. What is N? Using this rule, how many years will it take to double your money at a 5% interest rate, compounded annually? What happens if interest is compounded continuously?

(c) How many years will it actually take to double your money at a 5% interest rate, compounded annually? [Hint: $10 \log_{10} 7 = 8.45$ dB.] Whose rule best predicts the correct result?

Exercise 2. (Biorthogonal codes) A $2^m \times 2^m$ $\{\pm 1\}$-valued Hadamard matrix H_{2^m} may be constructed recursively as the m-fold tensor product of the 2×2 matrix

$$H_2 = \begin{bmatrix} +1 & +1 \\ +1 & -1 \end{bmatrix},$$

as follows:

$$H_{2^m} = \begin{bmatrix} +H_{2^{m-1}} & +H_{2^{m-1}} \\ +H_{2^{m-1}} & -H_{2^{m-1}} \end{bmatrix}.$$

(a) Show by induction that:

(i) $(H_{2^m})^T = H_{2^m}$, where ^{T} denotes the transpose; *i.e.,* H_{2^m} is symmetric;

(ii) The rows or columns of H_{2^m} form a set of mutually orthogonal vectors of length 2^m;

(iii) The first row and the first column of H_{2^m} consist of all $+1$s;

(iv) There are an equal number of $+1$s and -1s in all other rows and columns of H_{2^m};

(v) $H_{2^m} H_{2^m} = 2^m I_{2^m}$; *i.e.,* $(H_{2^m})^{-1} = 2^{-m} H_{2^m}$, where $^{-1}$ denotes the inverse.

(b) A biorthogonal signal set is a set of real equal-energy orthogonal vectors and their negatives. Show how to construct a biorthogonal signal set of size 64 as a set of $\{\pm 1\}$-valued sequences of length 32.

(c) A simplex signal set S is a set of real equal-energy vectors that are equidistant and that have zero mean $\mathbf{m}(S)$ under an equiprobable distribution. Show how to construct a simplex signal set of size 32 as a set of 32 $\{\pm 1\}$-valued sequences of length 31. [Hint: The fluctuation $O - \mathbf{m}(O)$ of a set O of orthogonal real vectors is a simplex signal set.]

(d) Let $\mathbf{Y} = \mathbf{X} + \mathbf{N}$ be the received sequence on a discrete-time AWGN channel, where the input sequence $\mathbf{X}$ is chosen equiprobably from a biorthogonal signal set B of size 2^{m+1} constructed as in part (b). Show that the following algorithm implements a minimum-distance decoder for B (*i.e.,* given a real 2^m-vector $\mathbf{y}$, it finds the closest $\mathbf{x} \in B$ to $\mathbf{y}$):

(i) Compute $\mathbf{z} = H_{2^m}\mathbf{y}$, where $\mathbf{y}$ is regarded as a column vector;

(ii) Find the component z_j of $\mathbf{z}$ with largest magnitude $|z_j|$;

(iii) Decode to $\mathrm{sgn}(z_j)\mathbf{x}_j$, where $\mathrm{sgn}(z_j)$ is the sign of the largest-magnitude component z_j and $\mathbf{x}_j$ is the corresponding column of H_{2^m}.

(e) Show that a circuit similar to that shown below for $m = 2$ can implement the $2^m \times 2^m$ matrix multiplication $\mathbf{z} = H_{2^m}\mathbf{y}$ with a total of only $m \times 2^m$ addition and subtraction operations. (This is called the "fast Hadamard transform," or "Walsh transform," or "Green machine.")

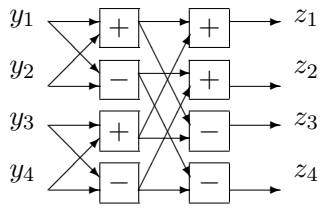


Figure 2. Fast $2^m \times 2^m$ Hadamard transform for $m = 2$.

Exercise 3. (16-QAM signal sets) Three 16-point 2-dimensional quadrature amplitude modulation (16-QAM) signal sets are shown in Figure 3, below. The first is a standard 4×4 signal set; the second is the V.29 signal set; the third is based on a hexagonal grid and is the most power-efficient 16-QAM signal set known. The first two have 90° symmetry; the last, only 180°. All have a minimum squared distance between signal points of $d^2_{\min} = 4$.

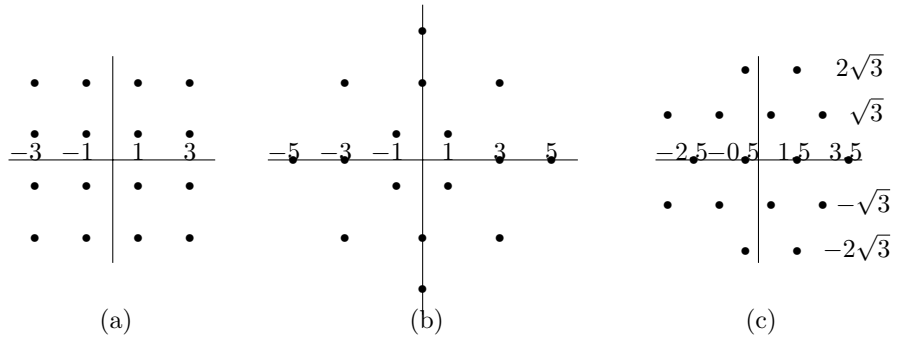


Figure 3. 16-QAM signal sets. (a) (4×4)-QAM; (b) V.29; (c) hexagonal.

(a) Compute the average energy (squared norm) of each signal set if all points are equiprobable. Compare the power efficiencies of the three signal sets in dB.

(b) Sketch the decision regions of a minimum-distance detector for each signal set.

(c) Show that with a phase rotation of $\pm 10°$ the minimum distance from any rotated signal point to any decision region boundary is substantially greatest for the V.29 signal set.

Exercise 4. (Shaping gain of spherical signal sets) In this exercise we compare the power efficiency of n-cube and n-sphere signal sets for large n.

An n-cube signal set is the set of all odd-integer sequences of length n within an n-cube of side $2M$ centered on the origin. For example, the signal set of Figure 3(a) is a 2-cube signal set with $M = 4$.

An n-sphere signal set is the set of all odd-integer sequences of length n within an n-sphere of squared radius r^2 centered on the origin. For example, the signal set of Figure 3(a) is also a 2-sphere signal set for any squared radius r^2 in the range $18 \le r^2 < 25$. In particular, it is a 2-sphere signal set for $r^2 = 64/\pi = 20.37$, where the area πr^2 of the 2-sphere (circle) equals the area $(2M)^2 = 64$ of the 2-cube (square) of the previous paragraph.

Both n-cube and n-sphere signal sets therefore have minimum squared distance between signal points $d^2_{\min} = 4$ (if they are nontrivial), and n-cube decision regions of side 2 and thus volume 2^n associated with each signal point. The point of the following exercise is to compare their average energy using the following large-signal-set approximations:

- The number of signal points is approximately equal to the volume $V(\mathcal{R})$ of the bounding n-cube or n-sphere region $\mathcal{R}$ divided by 2^n, the volume of the decision region associated with each signal point (an n-cube of side 2).
- The average energy of the signal points under an equiprobable distribution is approximately equal to the average energy $E(\mathcal{R})$ of the bounding n-cube or n-sphere region $\mathcal{R}$ under a uniform continuous distribution.

(a) Show that if $\mathcal{R}$ is an n-cube of side $2M$ for some integer M, then under the two above approximations the approximate number of signal points is M^n and the approximate average energy is $nM^2/3$. Show that the first of these two approximations is exact.

(b) For n even, if $\mathcal{R}$ is an n-sphere of radius r, compute the approximate number of signal points and the approximate average energy of an n-sphere signal set, using the following known expressions for the volume $V_{\otimes}(n, r)$ and the average energy $E_{\otimes}(n, r)$ of an n-sphere of radius r:

$$\begin{aligned} V_{\otimes}(n, r) &= \frac{(\pi r^2)^{n/2}}{(n/2)!}; \\ E_{\otimes}(n, r) &= \frac{nr^2}{n+2}. \end{aligned}$$

(c) For $n = 2$, show that a large 2-sphere signal set has about 0.2 dB smaller average energy than a 2-cube signal set with the same number of signal points.

(d) For $n = 16$, show that a large 16-sphere signal set has about 1 dB smaller average energy than a 16-cube signal set with the same number of signal points. [Hint: $8! = 40320$ (46.06 dB).]

(e) Show that as $n \to \infty$ a large n-sphere signal set has a factor of $\pi e/6$ (1.53 dB) smaller average energy than an n-cube signal set with the same number of signal points. [Hint: Use Stirling's approximation, $m! \to (m/e)^m$ as $m \to \infty$.]

Exercise 5. (56 kb/s PCM modems)

This problem has to do with the design of "56 kb/s PCM modems" such as V.90 and V.92.

In the North American telephone network, voice is commonly digitized by low-pass filtering to about 3.8 KHz, sampling at 8000 samples per second, and quantizing each sample into an 8-bit byte according to the so-called "μ law." The μ law specifies 255 distinct signal levels, which are a quantized, piecewise-linear approximation to a logarithmic function, as follows:

- 1 level at 0;
- 15 positive levels evenly spaced with $d = 2$ between 2 and 30 (*i.e.,* 2, 4, 6, 8, ..., 30);
- 16 positive levels evenly spaced with $d = 4$ between 33 and 93;
- 16 positive levels evenly spaced with $d = 8$ between 99 and 219;
- 16 positive levels evenly spaced with $d = 16$ between 231 and 471;
- 16 positive levels evenly spaced with $d = 32$ between 495 and 975;
- 16 positive levels evenly spaced with $d = 64$ between 1023 and 1983;
- 16 positive levels evenly spaced with $d = 128$ between 2079 and 3999;
- 16 positive levels evenly spaced with $d = 256$ between 4191 and 8031;
- plus 127 symmetric negative levels.

The resulting 64 kb/s digitized voice sequence is transmitted through the network and ultimately reconstructed at a remote central office by pulse amplitude modulation (PAM) using a μ-law digital/analog converter and a 4 KHz low-pass filter.

For a V.90 modem, one end of the link is assumed to have a direct 64 kb/s digital connection and to be able to send any sequence of 8000 8-bit bytes per second. The corresponding levels are reconstructed at the remote central office. For the purposes of this exercise, assume that the reconstruction is exactly according to the μ-law table above, and that the reconstructed pulses are then sent through an ideal 4 KHz additive AWGN channel to the user.

(a) Determine the maximum number M of levels that can be chosen from the 255-point μ-law constellation above such that the minimum separation between levels is $d = 2$, 4, 8, 16, 64, 128, 256, 512, or 1024, respectively.

(b) These uncoded M-PAM subconstellations may be used to send up to $r = \log_2 M$ bits per symbol. What level separation can be obtained while sending 40 kb/s? 48 kb/s? 56 kb/s?

(c) How much more SNR in dB is required to transmit reliably at 48 kb/s compared to 40 kb/s? At 56 kb/s compared to 48 kb/s?

References

[1] C. Berrou, A. Glavieux and P. Thitimajshima, "Near Shannon limit error-correcting coding and decoding: Turbo codes," *Proc. 1993 Int. Conf. Commun.* (Geneva), pp. 1064–1070, May 1993.

[2] S.-Y. Chung, G. D. Forney, Jr., T. J. Richardson and R. Urbanke, "On the design of low-density parity-check codes within 0.0045 dB from the Shannon limit," *IEEE Commun. Letters*, vol. 5, pp. 58–60, Feb. 2001.

[3] D. J. Costello, Jr., J. Hagenauer, H. Imai and S. B. Wicker, "Applications of error-control coding," *IEEE Trans. Inform. Theory*, vol. 44, pp. 2531–2560, Oct. 1998.

[4] R. G. Gallager, *Low-Density Parity-Check Codes*. Cambridge, MA: MIT Press, 1962.

[5] E. A. Lee and D. G. Messerschmitt, *Digital Communication* (first edition). Boston: Kluwer, 1988.

[6] E. A. Lee and D. G. Messerschmitt, *Digital Communication* (second edition). Boston: Kluwer, 1994.

[7] C. E. Shannon, "A mathematical theory of communication," *Bell Syst. Tech. J.*, vol. 27, pp. 379–423 and 623–656, 1948.

Chapter 2

Discrete-time and continuous-time AWGN channels

In this chapter we begin our technical discussion of coding for the AWGN channel. Our purpose is to show how the continuous-time AWGN channel model $Y(t) = X(t) + N(t)$ may be reduced to an equivalent discrete-time AWGN channel model $\mathbf{Y} = \mathbf{X} + \mathbf{N}$, without loss of generality or optimality. This development relies on the sampling theorem and the theorem of irrelevance. More practical methods of obtaining such a discrete-time model are orthonormal PAM or QAM modulation, which use an arbitrarily small amount of excess bandwidth. Important parameters of the continuous-time channel such as SNR, spectral efficiency and capacity carry over to discrete time, provided that the bandwidth is taken to be the nominal (Nyquist) bandwidth. Readers who are prepared to take these assertions on faith may skip this chapter.

2.1 Continuous-time AWGN channel model

The continuous-time AWGN channel is a random channel whose output is a real random process

$$Y(t) = X(t) + N(t),$$

where $X(t)$ is the input waveform, regarded as a real random process, and $N(t)$ is a real white Gaussian noise process with single-sided noise power density N_0 which is independent of $X(t)$.

Moreover, the input $X(t)$ is assumed to be both power-limited and band-limited. The average input power of the input waveform $X(t)$ is limited to some constant P. The channel band B is a positive-frequency interval with *bandwidth* W Hz. The channel is said to be baseband if $B = [0, W]$, and passband otherwise. The (positive-frequency) support of the Fourier transform of any sample function $x(t)$ of the input process $X(t)$ is limited to B.

The *signal-to-noise ratio* SNR of this channel model is then

$$\text{SNR} = \frac{P}{N_0 W},$$

where $N_0 W$ is the total noise power in the band B. The parameter N_0 is defined by convention to make this relationship true; *i.e.,* N_0 is the noise power per positive-frequency Hz. Therefore the double-sided power spectral density of $N(t)$ must be $S_{nn}(f) = N_0/2$, at least over the bands $\pm B$.

The two parameters W and SNR turn out to characterize the channel completely for digital communications purposes; the absolute scale of P and N_0 and the location of the band B do not affect the model in any essential way. In particular, as we will show in Chapter 3, the capacity of any such channel in bits per second is

$$C_{[\text{b/s}]} = W \log_2(1 + \text{SNR}) \quad \text{b/s}.$$

If a particular digital communication scheme transmits a continuous bit stream over such a channel at rate R b/s, then the *spectral efficiency* of the scheme is said to be $\rho = R/W$ (b/s)/Hz (read as "bits per second per Hertz"). The Shannon limit on spectral efficiency is therefore

$$C_{[(\text{b/s})/\text{Hz}]} = \log_2(1 + \text{SNR}) \quad (\text{b/s})/\text{Hz};$$

i.e., reliable transmission is possible when $\rho < C_{[(\text{b/s})/\text{Hz}]}$, but not when $\rho > C_{[(\text{b/s})/\text{Hz}]}$.

2.2 Signal spaces

In the next few sections we will briefly review how this continuous-time model may be reduced to an equivalent discrete-time model via the sampling theorem and the theorem of irrelevance. We assume that the reader has seen such a derivation previously, so our review will be rather succinct.

The set of all real finite-energy signals $x(t)$, denoted by $\mathcal{L}_2$, is a real vector space; *i.e.,* it is closed under addition and under multiplication by real scalars. The inner product of two signals $x(t), y(t) \in \mathcal{L}_2$ is defined by

$$\langle x(t), y(t) \rangle = \int x(t) y(t) \, dt.$$

The squared Euclidean norm (energy) of $x(t) \in \mathcal{L}_2$ is defined as $||x(t)||^2 = \langle x(t), x(t) \rangle < \infty$, and the squared Euclidean distance between $x(t), y(t) \in \mathcal{L}_2$ is $d^2(x(t), y(t)) = ||x(t) - y(t)||^2$. Two signals in $\mathcal{L}_2$ are regarded as the same ($\mathcal{L}_2$-equivalent) if their distance is 0. This allows the following strict positivity property to hold, as it must for a proper distance metric:

$$||x(t)||^2 \geq 0, \quad \text{with strict inequality unless } x(t) \text{ is } \mathcal{L}_2\text{-equivalent to } 0.$$

Every signal $x(t) \in \mathcal{L}_2$ has an $\mathcal{L}_2$ Fourier transform

$$\hat{x}(f) = \int x(t) e^{-2\pi i f t} dt,$$

such that, up to $\mathcal{L}_2$-equivalence, $x(t)$ can be recovered by the inverse Fourier transform:

$$x(t) = \int \hat{x}(f) e^{2\pi i f t} df.$$

We write $\hat{x}(f) = \mathcal{F}(x(t)), x(t) = \mathcal{F}^{-1}(\hat{x}(f))$, and $x(t) \leftrightarrow \hat{x}(f)$.

It can be shown that an $\mathcal{L}_2$ signal $x(t)$ is $\mathcal{L}_2$-equivalent to a signal which is continuous except at a discrete set of points of discontinuity ("almost everywhere"); therefore so is $\hat{x}(f)$. The values of an $\mathcal{L}_2$ signal or its transform at points of discontinuity are immaterial.

By Parseval's theorem, the Fourier transform preserves inner products:

$$\langle x(t), y(t)\rangle = \langle \hat{x}(f), \hat{y}(f)\rangle = \int \hat{x}(f)\hat{y}^*(f)\, df.$$

In particular, $||x(t)||^2 = ||\hat{x}(f)||^2$.

A signal space is any subspace $\mathcal{S} \subseteq \mathcal{L}_2$. For example, the set of $\mathcal{L}_2$ signals that are time-limited to an interval $[0, T]$ ("have support $[0, T]$") is a signal space, as is the set of $\mathcal{L}_2$ signals whose Fourier transforms are nonzero only in $\pm B$ ("have frequency support $\pm B$").

Every signal space $\mathcal{S} \subseteq \mathcal{L}_2$ has an orthogonal basis $\{\phi_k(t), k \in \mathcal{I}\}$, where $\mathcal{I}$ is some discrete index set, such that every $x(t) \in \mathcal{S}$ may be expressed as

$$x(t) = \sum_{k\in\mathcal{I}} \frac{\langle x(t), \phi_k(t)\rangle}{||\phi_k(t)||^2}\phi_k(t),$$

up to $\mathcal{L}_2$ equivalence. This is called an orthogonal expansion of $x(t)$.

Of course this expression becomes particularly simple if $\{\phi_k(t)\}$ is an orthonormal basis with $||\phi_k(t)||^2 = 1$ for all $k \in \mathcal{I}$. Then we have the orthonormal expansion

$$x(t) = \sum_{k\in\mathcal{I}} x_k\phi_k(t),$$

where $\mathbf{x} = \{x_k = \langle x(t), \phi_k(t)\rangle, k \in \mathcal{I}\}$ is the corresponding set of orthonormal coefficients. From this expression, we see that inner products are preserved in an orthonormal expansion; *i.e.,*

$$\langle x(t), y(t)\rangle = \langle \mathbf{x}, \mathbf{y}\rangle = \sum_{k\in\mathcal{I}} x_k y_k.$$

In particular, $||x(t)||^2 = ||\mathbf{x}||^2$.

2.3 The sampling theorem

The sampling theorem allows us to convert a continuous signal $x(t)$ with frequency support $[-W, W]$ (*i.e.,* a baseband signal with bandwidth W) to a discrete-time sequence of samples $\{x(kT), k \in \mathbb{Z}\}$ at a rate of $2W$ samples per second, with no loss of information.

The sampling theorem is basically an orthogonal expansion for the space $\mathcal{L}_2[0, W]$ of signals that have frequency support $[-W, W]$. If $T = 1/2W$, then the complex exponentials $\{\exp(2\pi i f kT), k \in \mathbb{Z}\}$ form an orthogonal basis for the space of Fourier transforms with support $[-W, W]$. Therefore their scaled inverse Fourier transforms $\{\phi_k(t) = \mathrm{sinc}_T(t - kT), k \in \mathbb{Z}\}$ form an orthogonal basis for $\mathcal{L}_2[0, W]$, where $\mathrm{sinc}_T(t) = (\sin \pi t/T)/(\pi t/T)$. Since $||\mathrm{sinc}_T(t)||^2 = T$, every $x(t) \in \mathcal{L}_2[0, W]$ may therefore be expressed up to $\mathcal{L}_2$ equivalence as

$$x(t) = \frac{1}{T}\sum_{k\in\mathbb{Z}} \langle x(t), \mathrm{sinc}_T(t - kT)\rangle \mathrm{sinc}_T(t - kT).$$

Moreover, evaluating this equation at $t = jT$ gives $x(jT) = \frac{1}{T}\langle x(t), \mathrm{sinc}_T(t - jT)\rangle$ for all $j \in \mathbb{Z}$ (provided that $x(t)$ is continuous at $t = jT$), since $\mathrm{sinc}_T((j - k)T) = 1$ for $k = j$ and $\mathrm{sinc}_T((j - k)T) = 0$ for $k \neq j$. Thus if $x(t) \in \mathcal{L}_2[0, W]$ is continuous, then

$$x(t) = \sum_{k\in\mathbb{Z}} x(kT)\mathrm{sinc}_T(t - kT).$$

This is called the sampling theorem.

Since inner products are preserved in an orthonormal expansion, and here the orthonormal coefficients are $x_k = \frac{1}{\sqrt{T}}\langle x(t), \mathrm{sinc}_T(t-kT)\rangle = \sqrt{T}x(kT)$, we have

$$\langle x(t), y(t)\rangle = \langle \mathbf{x}, \mathbf{y}\rangle = T\sum_{k\in\mathbb{Z}} x(kT)y(kT).$$

The following exercise shows similarly how to convert a continuous passband signal $x(t)$ with bandwidth W (*i.e.,* with frequency support $\pm[f_c - W/2, f_c + W/2]$ for some center frequency $f_c > W/2$) to a discrete-time sequence of sample pairs $\{(x_{c,k}, x_{s,k}), k \in \mathbb{Z}\}$ at a rate of W pairs per second, with no loss of information.

Exercise 2.1 (Orthogonal bases for passband signal spaces)

(a) Show that if $\{\phi_k(t)\}$ is an orthogonal set of signals in $\mathcal{L}_2[0, W]$, then $\{\phi_k(t)\cos 2\pi f_c t, \phi_k(t)\sin 2\pi f_c t\}$ is an orthogonal set of signals in $\mathcal{L}_2[f_c - W, f_c + W]$, the set of signals in $\mathcal{L}_2$ that have frequency support $\pm[f_c - W, f_c + W]$, provided that $f_c \geq W$.

[Hint: use the facts that $\mathcal{F}(\phi_k(t)\cos 2\pi f_c t) = (\hat{\phi}_k(f - f_c) + \hat{\phi}_k(f + f_c))/2$ and $\mathcal{F}(\phi_k(t)\sin 2\pi f_c t) = (\hat{\phi}_k(f - f_c) - \hat{\phi}_k(f + f_c))/2i$, plus Parseval's theorem.]

(b) Show that if the set $\{\phi_k(t)\}$ is an orthogonal basis for $\mathcal{L}_2[0, W]$, then the set $\{\phi_k(t)\cos 2\pi f_c t, \phi_k(t)\sin 2\pi f_c t\}$ is an orthogonal basis for $\mathcal{L}_2[f_c - W, f_c + W]$, provided that $f_c \geq W$.

[Hint: show that every $x(t) \in \mathcal{L}_2[f_c - W, f_c + W]$ may be written as $x(t) = x_c(t)\cos 2\pi f_c t + x_s(t)\sin 2\pi f_c t$ for some $x_c(t), x_s(t) \in \mathcal{L}_2[0, W]$.]

(c) Conclude that every $x(t) \in \mathcal{L}_2[f_c - W, f_c + W]$ may be expressed up to $\mathcal{L}_2$ equivalence as

$$x(t) = \sum_{k\in\mathbb{Z}} (x_{c,k}\cos 2\pi f_c t + x_{s,k}\sin 2\pi f_c t)\,\mathrm{sinc}_T(t - kT), \quad T = \frac{1}{2W},$$

for some sequence of pairs $\{(x_{c,k}, x_{s,k}), k \in \mathbb{Z}\}$, and give expressions for $x_{c,k}$ and $x_{s,k}$. □

2.4 White Gaussian noise

The question of how to define a white Gaussian noise (WGN) process $N(t)$ in general terms is plagued with mathematical difficulties. However, when we are given a signal space $\mathcal{S} \subseteq \mathcal{L}_2$ with an orthonormal basis as here, then defining WGN with respect to $\mathcal{S}$ is not so problematic. The following definition captures the essential properties that hold in this case:

Definition 2.1 (White Gaussian noise with respect to a signal space $\mathcal{S}$) *Let $\mathcal{S} \subseteq \mathcal{L}_2$ be a signal space with an orthonormal basis $\{\phi_k(t), k \in \mathcal{I}\}$. A Gaussian process $N(t)$ is defined as white Gaussian noise with respect to $\mathcal{S}$ with single-sided power spectral density N_0 if*

(a) The sequence $\{N_k = \langle N(t), \phi_k(t)\rangle, k \in \mathcal{I}\}$ is a sequence of iid Gaussian noise variables with mean zero and variance $N_0/2$;

(b) Define the "in-band noise" as the projection $N_{|\mathcal{S}}(t) = \sum_{k\in\mathcal{I}} N_k\phi_k(t)$ of $N(t)$ onto the signal space $\mathcal{S}$, and the "out-of-band noise" as $N_{|\mathcal{S}^\perp}(t) = N(t) - N_{|\mathcal{S}}(t)$. Then $N_{|\mathcal{S}^\perp}(t)$ is a process which is jointly Gaussian with $N_{|\mathcal{S}}(t)$, has sample functions which are orthogonal to $\mathcal{S}$, is uncorrelated with $N_{|\mathcal{S}}(t)$, and thus is statistically independent of $N_{|\mathcal{S}}(t)$.

For example, any stationary Gaussian process whose single-sided power spectral density is equal to N_0 within a band B and arbitrary elsewhere is white with respect to the signal space $\mathcal{L}_2(B)$ of signals with frequency support $\pm B$.

Exercise 2.2 (Preservation of inner products) Show that a Gaussian process $N(t)$ is white with respect to a signal space $\mathcal{S} \subseteq \mathcal{L}_2$ with psd N_0 if and only if for any signals $x(t), y(t) \in \mathcal{S}$,

$$\mathsf{E}[\langle N(t), x(t)\rangle\langle N(t), y(t)\rangle] = \frac{N_0}{2}\langle x(t), y(t)\rangle. \qquad \square$$

Here we are concerned with the detection of signals that lie in some signal space $\mathcal{S}$ in the presence of additive white Gaussian noise. In this situation the following theorem is fundamental:

Theorem 2.1 (Theorem of irrelevance) *Let $X(t)$ be a random signal process whose sample functions $x(t)$ lie in some signal space $\mathcal{S} \subseteq \mathcal{L}_2$ with an orthonormal basis $\{\phi_k(t), k \in \mathcal{I}\}$, let $N(t)$ be a Gaussian noise process which is independent of $X(t)$ and white with respect to $\mathcal{S}$, and let $Y(t) = X(t) + N(t)$. Then the set of samples*

$$Y_k = \langle Y(t), \phi_k(t)\rangle, \quad k \in \mathcal{I},$$

is a set of sufficient statistics for detection of $X(t)$ from $Y(t)$.

Sketch of proof. We may write

$$Y(t) = Y_{|\mathcal{S}}(t) + Y_{|\mathcal{S}^\perp}(t),$$

where $Y_{|\mathcal{S}}(t) = \sum_k Y_k\phi_k(t)$ and $Y_{|\mathcal{S}^\perp}(t) = Y(t) - Y_{|\mathcal{S}}(t)$. Since $Y(t) = X(t) + N(t)$ and

$$X(t) = \sum_k \langle X(t), \phi_k(t)\rangle\phi_k(t),$$

since all sample functions of $X(t)$ lie in $\mathcal{S}$, we have

$$Y(t) = \sum_k Y_k\phi_k(t) + N_{|\mathcal{S}^\perp}(t),$$

where $N_{|\mathcal{S}^\perp}(t) = N(t) - \sum_k\langle N(t), \phi_k(t)\rangle\phi_k(t)$. By Definition 2.1, $N_{|\mathcal{S}^\perp}(t)$ is independent of $N_{|\mathcal{S}}(t) = \sum_k\langle N(t), \phi_k(t)\rangle\phi_k(t)$, and by hypothesis it is independent of $X(t)$. Thus the probability distribution of $X(t)$ given $Y_{|\mathcal{S}}(t) = \sum_k Y_k\phi_k(t)$ and $Y_{|\mathcal{S}^\perp}(t) = N_{|\mathcal{S}^\perp}(t)$ depends only on $Y_{|\mathcal{S}}(t)$, so without loss of optimality in detection of $X(t)$ from $Y(t)$ we can disregard $Y_{|\mathcal{S}^\perp}(t)$; *i.e.,* $Y_{|\mathcal{S}}(t)$ is a sufficient statistic. Moreover, since $Y_{|\mathcal{S}}(t)$ is specified by the samples $\{Y_k\}$, these samples equally form a set of sufficient statistics for detection of $X(t)$ from $Y(t)$. $\square$

The sufficient statistic $Y_{|\mathcal{S}}(t)$ may alternatively be generated by filtering out the out-of-band noise $N_{|\mathcal{S}^\perp}(t)$. For example, for the signal space $\mathcal{L}_2(B)$ of signals with frequency support $\pm B$, we may obtain $Y_{|\mathcal{S}}(t)$ by passing $Y(t)$ through a brick-wall filter which passes all frequency components in B and rejects all components not in B.[1]

[1]Theorem 2.1 may be extended to any model $Y(t) = X(t) + N(t)$ in which the out-of-band noise $N_{|\mathcal{S}^\perp}(t) = N(t) - N_{|\mathcal{S}}(t)$ is independent of both the signal $X(t)$ and the in-band noise $N_{|\mathcal{S}}(t) = \sum_k N_k\phi_k(t)$; *e.g.,* to models in which the out-of-band noise contains signals from other independent users. In the Gaussian case, independence of the out-of-band noise is automatic; in more general cases, independence is an additional assumption.

Combining Definition 2.1 and Theorem 2.1, we conclude that for any AWGN channel in which the signals are confined to a sample space $\mathcal{S}$ with orthonormal basis $\{\phi_k(t), k \in \mathcal{I}\}$, we may without loss of optimality reduce the output $Y(t)$ to the set of samples

$$Y_k = \langle Y(t), \phi_k(t)\rangle = \langle X(t), \phi_k(t)\rangle + \langle N(t), \phi_k(t)\rangle = X_k + N_k, \quad k \in \mathcal{I},$$

where $\{N_k, k \in \mathcal{I}\}$ is a set of iid Gaussian variables with mean zero and variance $N_0/2$. Moreover, if $x_1(t), x_2(t) \in \mathcal{S}$ are two sample functions of $X(t)$, then this orthonormal expansion preserves their inner product:

$$\langle x_1(t), x_2(t)\rangle = \langle \mathbf{x}_1, \mathbf{x}_2\rangle,$$

where $\mathbf{x}_1$ and $\mathbf{x}_2$ are the orthonormal coefficient sequences of $x_1(t)$ and $x_2(t)$, respectively.

2.5 Continuous time to discrete time

We now specialize these results to our original AWGN channel model $Y(t) = X(t) + N(t)$, where the average power of $X(t)$ is limited to P and the sample functions of $X(t)$ are required to have positive frequency support in a band B of width W. For the time being we consider the baseband case in which $B = [0, W]$.

The signal space is then the set $\mathcal{S} = \mathcal{L}_2[0, W]$ of all finite-energy signals $x(t)$ whose Fourier transform has support $\pm B$. The sampling theorem shows that $\{\phi_k(t) = \frac{1}{\sqrt{T}}\text{sinc}_T(t - kT), k \in \mathbb{Z}\}$ is an orthonormal basis for this signal space, where $T = 1/2W$, and that therefore without loss of generality we may write any $x(t) \in \mathcal{S}$ as

$$x(t) = \sum_{k\in\mathbb{Z}} x_k\phi_k(t),$$

where x_k is the orthonormal coefficient $x_k = \langle x(t), \phi_k(t)\rangle$, and equality is in the sense of $\mathcal{L}_2$ equivalence.

Consequently, if $X(t)$ is a random process whose sample functions $x(t)$ are all in $\mathcal{S}$, then we can write

$$X(t) = \sum_{k\in\mathbb{Z}} X_k\phi_k(t),$$

where $X_k = \langle X(t), \phi_k(t)\rangle = \int X(t)\phi_k(t)\ dt$, a random variable that is a linear functional of $X(t)$. In this way we can identify any random band-limited process $X(t)$ of bandwidth W with a discrete-time random sequence $\mathbf{X} = \{X_k\}$ at a rate of $2W$ real variables per second. Hereafter the input will be regarded as the sequence $\mathbf{X}$ rather than $X(t)$.

Thus $X(t)$ may be regarded as a sum of amplitude-modulated orthonormal pulses $X_k\phi_k(t)$. By the Pythagorean theorem,

$$||X(t)||^2 = \sum_{k\in\mathbb{Z}} ||X_k\phi_k(t)||^2 = \sum_{k\in\mathbb{Z}} X_k^2,$$

where we use the orthonormality of the $\phi_k(t)$. Therefore the requirement that the average power (energy per second) of $X(t)$ be less than P translates to a requirement that the average energy of the sequence $\mathbf{X}$ be less than P per $2W$ symbols, or equivalently less than $P/2W$ per symbol.[2]

[2]The requirement that the sample functions of $X(t)$ must be in $\mathcal{L}_2$ translates to the requirement that the sample sequences $\mathbf{x}$ of $\mathbf{X}$ must have finite energy. This requirement can be met by requiring that only finitely many elements of $\mathbf{x}$ be nonzero. However, we do not pursue such finiteness issues.

Similarly, the random Gaussian noise process $N(t)$ may be written as

$$N(t) = \sum_{k\in\mathbb{Z}} N_k\phi_k(t) + N_{|\mathcal{S}^\perp}(t)$$

where $\mathbf{N} = \{N_k = \langle N(t), \phi_k(t)\rangle\}$ is the sequence of orthonormal coefficients of $N(t)$ in $\mathcal{S}$, and $N_{|\mathcal{S}^\perp}(t) = N(t) - \sum_k N_k\phi_k(t)$ is out-of-band noise. The theorem of irrelevance shows that $N_{|\mathcal{S}^\perp}(t)$ may be disregarded without loss of optimality, and therefore that the sequence $\mathbf{Y} = \mathbf{X} + \mathbf{N}$ is a set of sufficient statistics for detection of $X(t)$ from $Y(t)$.

In summary, we conclude that the characteristics of the discrete-time model $\mathbf{Y} = \mathbf{X}+\mathbf{N}$ mirror those of the continuous-time model $Y(t) = X(t) + N(t)$ from which it was derived:

- The symbol interval is $T = 1/2W$; equivalently, the symbol rate is $2W$ symbols/s;
- The average signal energy per symbol is limited to $P/2W$;
- The noise sequence $\mathbf{N}$ is iid zero-mean (white) Gaussian, with variance $N_0/2$ per symbol;
- The signal-to-noise ratio is thus SNR $= (P/2W)/(N_0/2) = P/N_0W$, the same as for the continuous-time model;
- A data rate of ρ bits per two dimensions (b/2D) translates to a data rate of $R = W\rho$ b/s, or equivalently to a spectral efficiency of ρ (b/s)/Hz.

This important conclusion is the fundamental result of this chapter.

2.5.1 Passband case

Suppose now that the channel is instead a passband channel with positive-frequency support band $B = [f_c - W/2, f_c + W/2]$ for some center frequency $f_c > W/2$.

The signal space is then the set $\mathcal{S} = \mathcal{L}_2[f_c - W/2, f_c + W/2]$ of all finite-energy signals $x(t)$ whose Fourier transform has support $\pm B$.

In this case Exercise 2.1 shows that an orthogonal basis for the signal space is a set of signals of the form $\phi_{k,c}(t) = \mathrm{sinc}_T(t - kT)\cos 2\pi f_c t$ and $\phi_{k,s}(t) = \mathrm{sinc}_T(t - kT)\sin 2\pi f_c t$, where the symbol interval is now $T = 1/W$. Since the support of the Fourier transform of $\mathrm{sinc}_T(t - kT)$ is $[-W/2, W/2]$, the support of the transform of each of these signals is $\pm B$.

The derivation of a discrete-time model then goes as in the baseband case. The result is that the sequence of real pairs

$$(Y_{k,c}, Y_{k,s}) = (X_{k,c}, X_{k,s}) + (N_{k,c}, N_{k,s})$$

is a set of sufficient statistics for detection of $X(t)$ from $Y(t)$. If we compute scale factors correctly, we find that the characteristics of this discrete-time model are as follows:

- The symbol interval is $T = 1/W$, or the symbol rate is W symbols/s. In each symbol interval a pair of two real symbols is sent and received. We may therefore say that the rate is $2W = 2/T$ real dimensions per second, the same as in the baseband model.
- The average signal energy per dimension is limited to $P/2W$;

- The noise sequences $\mathbf{N}_c$ and $\mathbf{N}_s$ are independent real iid zero-mean (white) Gaussian sequences, with variance $N_0/2$ per dimension;
- The signal-to-noise ratio is again SNR $= (P/2W)/(N_0/2) = P/N_0W$;
- A data rate of ρ b/2D again translates to a spectral efficiency of ρ (b/s)/Hz.

Thus the passband discrete-time model is effectively the same as the baseband model.

In the passband case, it is often convenient to identify real pairs with single complex variables via the standard correspondence between $\mathbb{R}^2$ and $\mathbb{C}$ given by $(x, y) \leftrightarrow x + iy$, where $i = \sqrt{-1}$. This is possible because a complex iid zero-mean Gaussian sequence $\mathbf{N}$ with variance N_0 per complex dimension may be defined as $\mathbf{N} = \mathbf{N}_c + i\mathbf{N}_s$, where $\mathbf{N}_c$ and $\mathbf{N}_s$ are independent real iid zero-mean Gaussian sequences with variance $N_0/2$ per real dimension. Then we obtain a complex discrete-time model $\mathbf{Y} = \mathbf{X} + \mathbf{N}$ with the following characteristics:

- The symbol interval is $T = 1/W$, or the rate is W complex dimensions/s.
- The average signal energy per complex dimension is limited to P/W;
- The noise sequence $\mathbf{N}$ is a complex iid zero-mean Gaussian sequence, with variance N_0 per complex dimension;
- The signal-to-noise ratio is again SNR $= (P/W)/N_0 = P/N_0W$;
- A data rate of ρ bits per complex dimension translates to a spectral efficiency of ρ (b/s)/Hz.

This is still the same as before, if we regard one complex dimension as two real dimensions.

Note that even the baseband real discrete-time model may be converted to a complex discrete-time model simply by taking real variables two at a time and using the same map $\mathbb{R}^2 \to \mathbb{C}$.

The reader is cautioned that the correspondence between $\mathbb{R}^2$ and $\mathbb{C}$ given by $(x, y) \leftrightarrow x + iy$ preserves some algebraic, geometric and probabilistic properties, but not all.

Exercise 2.3 (Properties of the correspondence $\mathbb{R}^2 \leftrightarrow \mathbb{C}$) Verify the following assertions:

(a) Under the correspondence $\mathbb{R}^2 \leftrightarrow \mathbb{C}$, addition is preserved.

(b) However, multiplication is not preserved. (Indeed, the product of two elements of $\mathbb{R}^2$ is not even defined.)

(c) Inner products are not preserved. Indeed, two orthogonal elements of $\mathbb{R}^2$ can map to two collinear elements of $\mathbb{C}$.

(d) However, (squared) Euclidean norms and Euclidean distances are preserved.

(e) In general, if $\mathbf{N}_c$ and $\mathbf{N}_s$ are real jointly Gaussian sequences, then $\mathbf{N}_c + i\mathbf{N}_s$ is not a proper complex Gaussian sequence, even if $\mathbf{N}_c$ and $\mathbf{N}_s$ are independent iid sequences.

(f) However, if $\mathbf{N}_c$ and $\mathbf{N}_s$ are independent real iid zero-mean Gaussian sequences with variance $N_0/2$ per real dimension, then $\mathbf{N}_c + i\mathbf{N}_s$ is a complex zero-mean Gaussian sequence with variance N_0 per complex dimension. □

2.6 Orthonormal PAM and QAM modulation

More generally, suppose that $X(t) = \sum_k X_k \phi_k(t)$, where $\mathbf{X} = \{X_k\}$ is a random sequence and $\{\phi_k(t) = p(t - kT)\}$ is an orthonormal sequence of time shifts $p(t - kT)$ of a basic modulation pulse $p(t) \in \mathcal{L}_2$ by integer multiples of a symbol interval T. This is called *orthonormal pulse-amplitude modulation (PAM).*

The signal space $\mathcal{S}$ is then the subspace of $\mathcal{L}_2$ spanned by the orthonormal sequence $\{p(t-kT)\}$; *i.e.,* $\mathcal{S}$ consists of all signals in $\mathcal{L}_2$ that can be written as linear combinations $\sum_k x_k p(t - kT)$.

Again, the average power of $X(t) = \sum_k X_k p(t-kT)$ will be limited to P if the average energy of the sequence $\mathbf{X}$ is limited to PT per symbol, since the symbol rate is $1/T$ symbol/s.

The theorem of irrelevance again shows that the set of inner products

$$Y_k = \langle Y(t), \phi_k(t) \rangle = \langle X(t), \phi_k(t) \rangle + \langle N(t), \phi_k(t) \rangle = X_k + N_k$$

is a set of sufficient statistics for detection of $X(t)$ from $Y(t)$. These inner products may be obtained by filtering $Y(t)$ with a *matched filter* with impulse response $p(-t)$ and sampling at integer multiples of T as shown in Figure 1 to obtain

$$Z(kT) = \int Y(\tau) p(\tau - kT)\, d\tau = Y_k,$$

Thus again we obtain a discrete-time model $\mathbf{Y} = \mathbf{X} + \mathbf{N}$, where by the orthonormality of the $p(t - kT)$ the noise sequence $\mathbf{N}$ is iid zero-mean Gaussian with variance $N_0/2$ per symbol.

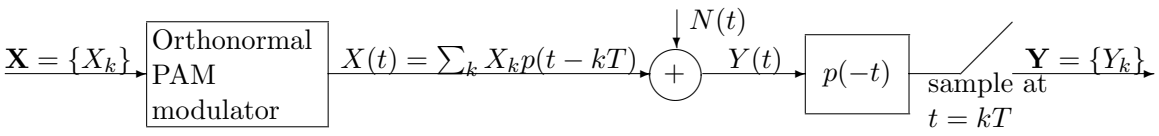


Figure 1. Orthonormal PAM system.

The conditions that ensure that the time shifts $\{p(t-kT)\}$ are orthonormal are determined by Nyquist theory as follows. Define the composite response in Figure 1 as $g(t) = p(t) * p(-t)$, with Fourier transform $\hat{g}(f) = |\hat{p}(f)|^2$. (The composite response $g(t)$ is also called the autocorrelation function of $p(t)$, and $\hat{g}(f)$ is also called its power spectrum.) Then:

Theorem 2.2 (Orthonormality conditions) *For a signal $p(t) \in \mathcal{L}_2$ and a time interval T, the following are equivalent:*

(a) The time shifts $\{p(t - kT), k \in \mathbb{Z}\}$ are orthonormal;

(b) The composite response $g(t) = p(t) * p(-t)$ satisfies $g(0) = 1$ and $g(kT) = 0$ for $k \neq 0$;

(c) The Fourier transform $\hat{g}(f) = |\hat{p}(f)|^2$ satisfies the Nyquist criterion for zero intersymbol interference, namely

$$\frac{1}{T} \sum_{m \in \mathbb{Z}} \hat{g}(f - m/T) = 1 \quad \text{for all } f.$$

Sketch of proof. The fact that (a) $\Leftrightarrow$ (b) follows from $\langle p(t - kT), p(t - k'T) \rangle = g((k - k')T)$. The fact that (b) $\Leftrightarrow$ (c) follows from the aliasing theorem, which says that the discrete-time Fourier transform of the sample sequence $\{g(kT)\}$ is the aliased response $\frac{1}{T} \sum_m \hat{g}(f - m/T)$. □

It is clear from the Nyquist criterion (c) that if $p(t)$ is a baseband signal of bandwidth W, then

(i) The bandwidth W cannot be less than $1/2T$;

(ii) If $W = 1/2T$, then $\hat{g}(f) = T, -W \leq f \leq W$, else $\hat{g}(f) = 0$; *i.e.,* $g(t) = \mathrm{sinc}_T(t)$;

(iii) If $1/2T < W \leq 1/T$, then any real non-negative power spectrum $\hat{g}(f)$ that satisfies $\hat{g}(1/2T + f) + \hat{g}(1/2T - f) = T$ for $0 \leq f \leq 1/2T$ will satisfy (c).

For this reason $W = 1/2T$ is called the *nominal* or *Nyquist bandwidth* of a PAM system with symbol interval T. No orthonormal PAM system can have bandwidth less than the Nyquist bandwidth, and only a system in which the modulation pulse has autocorrelation function $g(t) = p(t)*p(-t) = \mathrm{sinc}_T(t)$ can have exactly the Nyquist bandwidth. However, by (iii), which is called the *Nyquist band-edge symmetry condition*, the Fourier transform $|\hat{p}(f)|^2$ may be designed to roll off arbitrarily rapidly for $f > W$, while being continuous and having a continuous derivative.

Figure 2 illustrates a raised-cosine frequency response that satisfies the Nyquist band-edge symmetry condition while being continuous and having a continuous derivative. Nowadays it is no great feat to implement such responses with excess bandwidths of 5–10% or less.

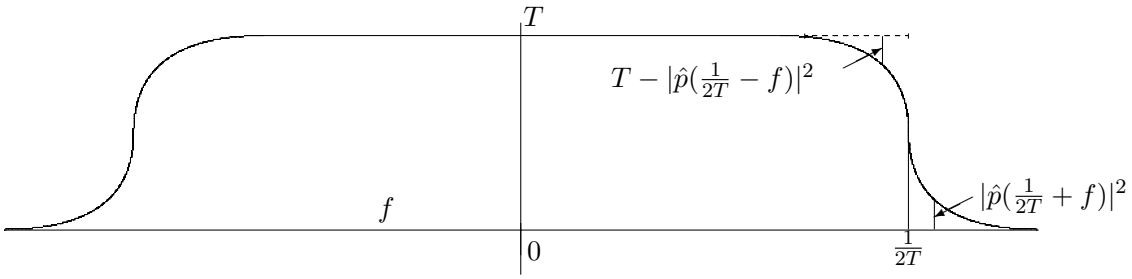


Figure 2. Raised-cosine spectrum $\hat{g}(f) = |\hat{p}(f)|^2$ with Nyquist band-edge symmetry.

We conclude that an orthonormal PAM system may use arbitrarily small excess bandwidth beyond the Nyquist bandwidth $W = 1/2T$, or alternatively that the power in the out-of-band frequency components may be made to be arbitrarily small, without violating the practical constraint that the Fourier transform $\hat{p}(f)$ of the modulation pulse $p(t)$ should be continuous and have a continuous derivative.

In summary, if we let W denote the Nyquist bandwidth $1/2T$ rather than the actual bandwidth, then we again obtain a discrete-time channel model $\mathbf{Y} = \mathbf{X} + \mathbf{N}$ for any orthonormal PAM system, not just a system with the modulation pulse $p(t) = \frac{1}{\sqrt{T}}\mathrm{sinc}_T(t)$, in which:

- The symbol interval is $T = 1/2W$; equivalently, the symbol rate is $2W$ symbols/s;
- The average signal energy per symbol is limited to $P/2W$;
- The noise sequence $\mathbf{N}$ is iid zero-mean (white) Gaussian, with variance $N_0/2$ per symbol;
- The signal-to-noise ratio is $\mathrm{SNR} = (P/2W)/(N_0/2) = P/N_0W$;
- A data rate of ρ bits per two dimensions (b/2D) translates to a data rate of $R = \rho/W$ b/s, or equivalently to a spectral efficiency of ρ (b/s)/Hz.

Exercise 2.4 (Orthonormal QAM modulation)

Figure 3 illustrates an orthonormal quadrature amplitude modulation (QAM) system with symbol interval T in which the input and output variables X_k and Y_k are complex, $p(t)$ is a complex finite-energy modulation pulse whose time shifts $\{p(t-kT)\}$ are orthonormal (the inner product of two complex signals is $\langle x(t), y(t)\rangle = \int x(t)y^*(t)\ dt$), the matched filter response is $p^*(-t)$, and $f_c > 1/2T$ is a carrier frequency. The box marked $2\Re\{\cdot\}$ takes twice the real part of its input— *i.e.,* it maps a complex signal $f(t)$ to $f(t) + f^*(t)$— and the Hilbert filter is a complex filter whose frequency response is 1 for $f > 0$ and 0 for $f < 0$.

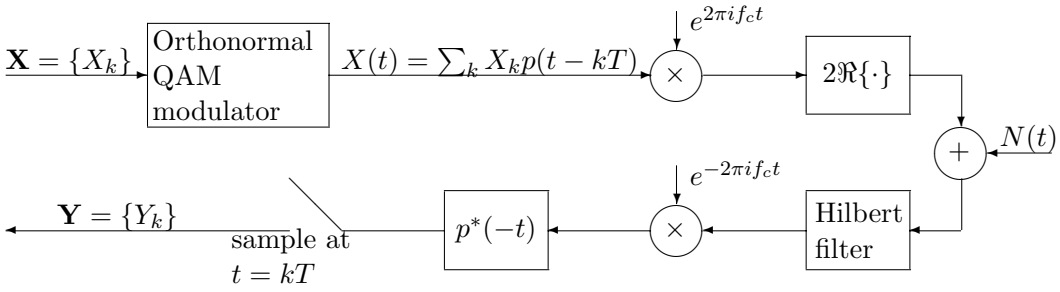


Figure 3. Orthonormal QAM system.

(a) Assume that $\hat{p}(f) = 0$ for $|f| \geq f_c$. Show that the Hilbert filter is superfluous.

(b) Show that Theorem 2.2 holds for a complex response $p(t)$ if we define the composite response (autocorrelation function) as $g(t) = p(t) * p^*(-t)$. Conclude that the bandwidth of an orthonormal QAM system is lowerbounded by its Nyquist bandwidth $W = 1/T$.

(c) Show that $\mathbf{Y} = \mathbf{X} + \mathbf{N}$, where $\mathbf{N}$ is an iid complex Gaussian noise sequence. Show that the signal-to-noise ratio in this complex discrete-time model is equal to the channel signal-to-noise ratio $\text{SNR} = P/N_0W$, if we define $W = 1/T$. [Hint: use Exercise 2.1.]

(d) Show that a mismatch in the receive filter— *i.e.,* an impulse response $h(t)$ other than $p^*(-t)$— results in linear intersymbol interference— *i.e.,* in the absence of noise $Y_k = \sum_j X_j h_{k-j}$ for some discrete-time response $\{h_k\}$ other than the ideal response δ_{k0} (Kronecker delta).

(e) Show that a phase error of θ in the receive carrier— *i.e.,* demodulation by $e^{-2\pi i f_c t + i\theta}$ rather than by $e^{-2\pi i f_c t}$— results (in the absence of noise) in a phase rotation by θ of all outputs Y_k.

(f) Show that a sample timing error of δ— *i.e.,* sampling at times $t = kT + \delta$— results in linear intersymbol interference. □

2.7 Summary

To summarize, the key parameters of a band-limited continuous-time AWGN channel are its bandwidth W in Hz and its signal-to-noise ratio SNR, regardless of other details like where the bandwidth is located (in particular whether it is at baseband or passband), the scaling of the signal, etc. The key parameters of a discrete-time AWGN channel are its symbol rate W in two-dimensional real or one-dimensional complex symbols per second and its SNR, regardless of other details like whether it is real or complex, the scaling of the symbols, etc. With orthonormal PAM or QAM, these key parameters are preserved, regardless of whether PAM or QAM is used, the precise modulation pulse, etc. The (nominal) spectral efficiency ρ (in (b/s)/Hz or in b/2D) is also preserved, and (as we will see in the next chapter) so is the channel capacity (in b/s).

Chapter 3

Capacity of AWGN channels

In this chapter we prove that the capacity of an AWGN channel with bandwidth W and signal-to-noise ratio SNR is $W \log_2(1+\text{SNR})$ bits per second (b/s). The proof that reliable transmission is possible at any rate less than capacity is based on Shannon's random code ensemble, typical-set decoding, the Chernoff-bound law of large numbers, and a fundamental result of large-deviation theory. We also sketch a geometric proof of the converse. Readers who are prepared to accept the channel capacity formula without proof may skip this chapter.

3.1 Outline of proof of the capacity theorem

The first step in proving the channel capacity theorem or its converse is to use the results of Chapter 2 to replace a continuous-time AWGN channel model $Y(t) = X(t) + N(t)$ with bandwidth W and signal-to-noise ratio SNR by an equivalent discrete-time channel model $\mathbf{Y} = \mathbf{X} + \mathbf{N}$ with a symbol rate of $2W$ real symbol/s and the same SNR, without loss of generality or optimality.

We then wish to prove that arbitrarily reliable transmission can be achieved on the discrete-time channel at any rate (nominal spectral efficiency)

$$\rho < C_{[\text{b/2D}]} = \log_2(1 + \text{SNR}) \quad \text{b/2D}.$$

This will prove that reliable transmission can be achieved on the continuous-time channel at any data rate

$$R < C_{[\text{b/s}]} = WC_{[\text{b/2D}]} = W \log_2(1 + \text{SNR}) \quad \text{b/s}.$$

We will prove this result by use of Shannon's random code ensemble and a suboptimal decoding technique called typical-set decoding.

Shannon's random code ensemble may be defined as follows. Let $S_x = P/2W$ be the allowable average signal energy per symbol (dimension), let ρ be the data rate in b/2D, and let N be the code block length in symbols. A block code $\mathcal{C}$ of length N, rate ρ, and average energy S_x per dimension is then a set of $M = 2^{\rho N/2}$ real sequences (codewords) $\mathbf{c}$ of length N such that the expected value of $||\mathbf{c}||^2$ under an equiprobable distribution over $\mathcal{C}$ is NS_x.

For example, the three 16-QAM signal sets shown in Figure 3 of Chapter 1 may be regarded as three block codes of length 2 and rate 4 b/2D with average energies per dimension of $S_x = 5, 6.75$ and 4.375, respectively.

In Shannon's random code ensemble, every symbol c_k of every codeword $\mathbf{c} \in \mathcal{C}$ is chosen independently at random from a Gaussian ensemble with mean 0 and variance S_x. Thus the average energy per dimension over the ensemble of codes is S_x, and by the law of large numbers the average energy per dimension of any particular code in the ensemble is highly likely to be close to S_x.

We consider the probability of error under the following scenario. A code $\mathcal{C}$ is selected randomly from the ensemble as above, and then a particular codeword $\mathbf{c}_0$ is selected for transmission. The channel adds a noise sequence $\mathbf{n}$ from a Gaussian ensemble with mean 0 and variance $S_n = N_0/2$ per symbol. At the receiver, given $\mathbf{y} = \mathbf{c}_0 + \mathbf{n}$ and the code $\mathcal{C}$, a typical-set decoder implements the following decision rule (where ε is some small positive number):

- If there is one and only one codeword $\mathbf{c} \in \mathcal{C}$ within squared distance $N(S_n \pm \varepsilon)$ of the received sequence $\mathbf{y}$, then decide on $\mathbf{c}$;
- Otherwise, give up.

A decision error can occur only if one of the following two events occurs:

- The squared distance $||\mathbf{y} - \mathbf{c}_0||^2$ between $\mathbf{y}$ and the transmitted codeword $\mathbf{c}_0$ is not in the range $N(S_n \pm \varepsilon)$;
- The squared distance $||\mathbf{y} - \mathbf{c}_i||^2$ between $\mathbf{y}$ and some other codeword $\mathbf{c}_i \neq \mathbf{c}_0$ is in the range $N(S_n \pm \varepsilon)$.

Since $\mathbf{y} - \mathbf{c}_0 = \mathbf{n}$, the probability of the first of these events is the probability that $||\mathbf{n}||^2$ is not in the range $N(S_n - \varepsilon) \leq ||\mathbf{n}||^2 \leq N(S_n + \varepsilon)$. Since $\mathbf{N} = \{N_k\}$ is an iid zero-mean Gaussian sequence with variance S_n per symbol and $||\mathbf{N}||^2 = \sum_k N_k^2$, this probability goes to zero as $N \to \infty$ for any $\varepsilon > 0$ by the weak law of large numbers. In fact, by the Chernoff bound of the next section, this probability goes to zero exponentially with N.

For any particular other codeword $\mathbf{c}_i \in \mathcal{C}$, the probability of the second event is the probability that a code sequence drawn according to an iid Gaussian pdf $p_X(\mathbf{x})$ with symbol variance S_x and a received sequence drawn *independently* according to an iid Gaussian pdf $p_Y(\mathbf{y})$ with symbol variance $S_y = S_x + S_n$ are "typical" of the joint pdf $p_{XY}(\mathbf{x}, \mathbf{y}) = p_X(\mathbf{x})p_N(\mathbf{y} - \mathbf{x})$, where here we define "typical" by the distance $||\mathbf{x} - \mathbf{y}||^2$ being in the range $N(S_n \pm \varepsilon)$. According to a fundamental result of large-deviation theory, this probability goes to zero as e^{-NE}, where, up to terms of the order of ε, the exponent E is given by the relative entropy (Kullback-Leibler divergence)

$$D(p_{XY}||p_X p_Y) = \int dx\ dy\ p_{XY}(x,y) \log \frac{p_{XY}(x,y)}{p_X(x)p_Y(y)}.$$

If the logarithm is binary, then this is the mutual information $I(X;Y)$ between the random variables X and Y in bits per dimension (b/D).

In the Gaussian case considered here, the mutual information is easily evaluated as

$$I(X;Y) = \mathsf{E}_{XY}\left[-\frac{1}{2}\log_2 2\pi S_n - \frac{(y-x)^2 \log_2 e}{2S_n} + \frac{1}{2}\log_2 2\pi S_y + \frac{y^2 \log_2 e}{2S_y}\right] = \frac{1}{2}\log_2 \frac{S_y}{S_n} \quad \text{b/D}.$$

Since $S_y = S_x + S_n$ and $\text{SNR} = S_x/S_n$, this expression is equal to the claimed capacity in b/D.

Thus we can say that the probability that any incorrect codeword $\mathbf{c}_i \in \mathcal{C}$ is "typical" with respect to $\mathbf{y}$ goes to zero as $2^{-N(I(X;Y)-\delta(\varepsilon))}$, where $\delta(\varepsilon)$ goes to zero as $\varepsilon \to 0$. By the union bound, the probability that any of the $M-1 < 2^{\rho N/2}$ incorrect codewords is "typical" with respect to $\mathbf{y}$ is upperbounded by

$$\Pr\{\text{any incorrect codeword "typical"}\} < 2^{\rho N/2} 2^{-N(I(X;Y)-\delta(\varepsilon))},$$

which goes to zero exponentially with N provided that $\rho < 2I(X;Y)$ b/2D and ε is small enough.

In summary, the probabilities of both types of error go to zero exponentially with N provided that

$$\rho < 2I(X;Y) = \log_2(1+\text{SNR}) = C_{[\text{b/2D}]} \quad \text{b/2D}$$

and ε is small enough. This proves that an arbitrarily small probability of error can be achieved using Shannon's random code ensemble and typical-set decoding.

To show that there is a particular code of rate $\rho < C_{[\text{b/2D}]}$ that achieves an arbitrarily small error probability, we need merely observe that the probability of error over the random code ensemble is the average probability of error over all codes in the ensemble, so there must be at least one code in the ensemble that achieves this performance. More pointedly, if the average error probability is $\Pr(E)$, then no more than a fraction of $1/K$ of the codes can achieve error probability worse than $K\Pr(E)$ for any constant $K > 0$; *e.g.,* at least 99% of the codes achieve performance no worse than $100\Pr(E)$. So we can conclude that almost all codes in the random code ensemble achieve very small error probabilities. Briefly, "almost all codes are good" (when decoded by typical-set or maximum-likelihood decoding).

3.2 Laws of large numbers

The channel capacity theorem is essentially an application of various laws of large numbers.

3.2.1 The Chernoff bound

The weak law of large numbers states that the probability that the sample average of a sequence of N iid random variables differs from the mean by more than $\varepsilon > 0$ goes to zero as $N \to \infty$, no matter how small ε is. The Chernoff bound shows that this probability goes to zero exponentially with N, for arbitrarily small ε.

Theorem 3.1 (Chernoff bound) *Let S_N be the sum of N iid real random variables X_k, each with the same probability distribution $p_X(x)$ and mean $\overline{X} = \mathsf{E}_X[X]$. For $\tau > \overline{X}$, the probability that $S_N \geq N\tau$ is upperbounded by*

$$\Pr\{S_N \geq N\tau\} \leq e^{-NE_c(\tau)},$$

where the Chernoff exponent $E_c(\tau)$ is given by

$$E_c(\tau) = \max_{s \geq 0} s\tau - \mu(s),$$

where $\mu(s)$ denotes the semi-invariant moment-generating function, $\mu(s) = \log \mathsf{E}_X[e^{sX}]$.

Proof. The indicator function $\Phi(S_N \geq N\tau)$ of the event $\{S_N \geq N\tau\}$ is bounded by

$$\Phi(S_N \geq N\tau) \leq e^{s(S_N - N\tau)}$$

for any $s \geq 0$. Therefore

$$\Pr\{S_N \geq N\tau\} = \overline{\Phi(S_N \geq N\tau)} \leq \overline{e^{s(S_N - N\tau)}}, \quad s \geq 0,$$

where the overbar denotes expectation. Using the facts that $S_N = \sum_k X_k$ and that the X_k are independent, we have

$$\overline{e^{s(S_N - N\tau)}} = \prod_k \overline{e^{s(X_k - \tau)}} = e^{-N(s\tau - \mu(s))},$$

where $\mu(s) = \log \overline{e^{sX}}$. Optimizing the exponent over $s \geq 0$, we obtain the Chernoff exponent

$$E_c(\tau) = \max_{s \geq 0} s\tau - \mu(s). \qquad \square$$

We next show that the Chernoff exponent is positive:

Theorem 3.2 (Positivity of Chernoff exponent) *The Chernoff exponent $E_c(\tau)$ is positive when $\tau > \overline{X}$, provided that the random variable X is nondeterministic.*

Proof. Define $X(s)$ as a random variable with the same alphabet as X, but with the tilted probability density function $q(x, s) = p(x)e^{sx - \mu(s)}$. This is a valid pdf because $q(x, s) \geq 0$ and

$$\int q(x, s)\ dx = e^{-\mu(s)} \int e^{sx} p(x)\ dx = e^{-\mu(s)} e^{\mu(s)} = 1.$$

Evidently $\mu(0) = \log \mathsf{E}_X[1] = 0$, so $q(x, 0) = p(x)$ and $X(0) = X$.

Define the moment-generating (partition) function

$$Z(s) = e^{\mu(s)} = \mathsf{E}_X[e^{sX}] = \int e^{sx} p(x)\ dx.$$

Now it is easy to see that

$$Z'(s) = \int x e^{sx} p(x)\ dx = e^{\mu(s)} \int x e^{sx} q(x, s)\ dx = Z(s)\overline{X(s)}.$$

Similarly,

$$Z''(s) = \int x^2 e^{sx} p(x)\ dx = Z(s)\overline{X^2(s)}.$$

Consequently, from $\mu(s) = \log Z(s)$, we have

$$\begin{aligned} \mu'(s) &= \frac{Z'(s)}{Z(s)} = \overline{X(s)}; \\ \mu''(s) &= \frac{Z''(s)}{Z(s)} - \left(\frac{Z'(s)}{Z(s)}\right)^2 = \overline{X^2(s)} - \overline{X(s)}^2. \end{aligned}$$

Thus the second derivative $\mu''(s)$ is the variance of $X(s)$, which must be strictly positive unless $X(s)$ and thus X is deterministic.

We conclude that if X is a nondeterministic random variable with mean $\overline{X}$, then $\mu(s)$ is a strictly convex function of s that equals 0 at $s = 0$ and whose derivative at $s = 0$ is $\overline{X}$. It follows that the function $s\tau - \mu(s)$ is a strictly concave function of s that equals 0 at $s = 0$ and whose derivative at $s = 0$ is $\tau - \overline{X}$. Thus if $\tau > \overline{X}$, then the function $s\tau - \mu(s)$ has a unique maximum which is strictly positive. □

Exercise 1. Show that if X is a deterministic random variable— *i.e.,* the probability that X equals its mean $\overline{X}$ is 1— and $\tau > \overline{X}$, then $\Pr\{S_N \geq N\tau\} = 0$. □

The proof of this theorem shows that the general form of the function $f(s) = s\tau - \mu(s)$ when X is nondeterministic is as shown in Figure 1. The second derivative $f''(s)$ is negative everywhere, so the function $f(s)$ is strictly concave and has a unique maximum $E_c(\tau)$. The slope $f'(s) = \tau - \overline{X(s)}$ therefore decreases continually from its value $f'(0) = \tau - \overline{X} > 0$ at $s = 0$. The slope becomes equal to 0 at the value of s for which $\tau = \overline{X(s)}$; in other words, to find the maximum of $f(s)$, keep increasing the "tilt" until the tilted mean $\overline{X(s)}$ is equal to τ. If we denote this value of s by $s^*(\tau)$, then we obtain the following parametric equations for the Chernoff exponent:

$$E_c(\tau) = s^*(\tau)\tau - \mu(s^*(\tau)); \quad \tau = \overline{X(s^*(\tau))}.$$

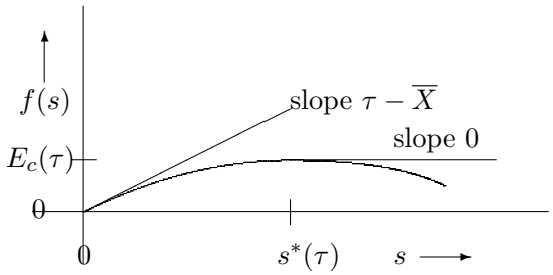


Figure 1. General form of function $f(s) = s\tau - \mu(s)$ when $\tau > \overline{X}$.

We will show below that the Chernoff exponent $E_c(\tau)$ is the correct exponent, in the sense that

$$\lim_{N\to\infty} \frac{\log \Pr\{S_N \geq N\tau\}}{N} = E_c(\tau).$$

The proof will be based on a fundamental theorem of large-deviation theory

We see that finding the Chernoff exponent is an exercise in convex optimization. In convex optimization theory, $E_c(\tau)$ and $\mu(s)$ are called conjugate functions. It is easy to show from the properties of $\mu(s)$ that $E_c(\tau)$ is a continuous, strictly convex function of τ that equals 0 at $\tau = \overline{X}$ and whose derivative at $\tau = \overline{X}$ is 0.

3.2.2 Chernoff bounds for functions of rvs

If $g : \mathcal{X} \to \mathbb{R}$ is any real-valued function defined on the alphabet $\mathcal{X}$ of a random variable X, then $g(X)$ is a real random variable. If $\{X_k\}$ is a sequence of iid random variables X_k with the same distribution as X, then $\{g(X_k)\}$ is a sequence of iid random variables $g(X_k)$ with the same distribution as $g(X)$. The Chernoff bound thus applies to the sequence $\{g(X_k)\}$, and shows that the probability that the sample mean $\frac{1}{N}\sum_k g(X_k)$ exceeds τ goes to zero exponentially with N as $N \to \infty$ whenever $\tau > \overline{g(X)}$.

Let us consider any finite set $\{g_j\}$ of such functions $g_j : \mathcal{X} \to \mathbb{R}$. Because the Chernoff bound decreases exponentially with N, we can conclude that the probability that *any* of the sample means $\frac{1}{N}\sum_k g_j(X_k)$ exceeds its corresponding expectation $\overline{g_j(X)}$ by a given fixed $\varepsilon > 0$ goes to zero exponentially with N as $N \to \infty$.

We may define a sequence $\{X_k\}$ to be ε-typical with respect to a function $g_j : \mathcal{X} \to \mathbb{R}$ if $\frac{1}{N}\sum_k g_j(X_k) < \overline{g_j(X)} + \varepsilon$. We can thus conclude that the probability that $\{X_k\}$ is not ε-typical with respect to any finite set $\{g_j\}$ of functions g_j goes to zero exponentially with N as $N \to \infty$.

A simple application of this result is that the probability that the sample mean $\frac{1}{N}\sum_k g_j(X_k)$ is not in the range $\overline{g_j(X)} \pm \varepsilon$ goes to zero exponentially with N as $N \to \infty$ for any $\varepsilon > 0$, because this probability is the sum of the two probabilities $\Pr\{\sum_k g_j(X_k) \geq N(\overline{g_j(X)} + \varepsilon)\}$ and $\Pr\{\sum_k -g_j(X_k) \geq N(-\overline{g_j(X)} + \varepsilon)\}$.

More generally, if the alphabet $\mathcal{X}$ is finite, then by considering the indicator functions of each possible value of X we can conclude that the probability that all observed relative frequencies in a sequence are not within ε of the corresponding probabilities goes to zero exponentially with N as $N \to \infty$. Similarly, for any alphabet $\mathcal{X}$, we can conclude that the probability of any finite number of sample moments $\frac{1}{N}\sum_k X_k^m$ are not within ε of the corresponding expected moments $\overline{X^m}$ goes to zero exponentially with N as $N \to \infty$.

In summary, the Chernoff bound law of large numbers allows us to say that as $N \to \infty$ we will almost surely observe a sample sequence $\mathbf{x}$ which is typical in every (finite) way that we might specify.

3.2.3 Asymptotic equipartition principle

One consequence of any law of large numbers is the asymptotic equipartition principle (AEP): as $N \to \infty$, the observed sample sequence $\mathbf{x}$ of an iid sequence whose elements are chosen according to a random variable X will almost surely be such that $p_X(\mathbf{x}) \approx 2^{-N\mathcal{H}(X)}$, where $\mathcal{H}(X) = \mathsf{E}_X[-\log_2 p(x)]$. If X is discrete, then $p_X(x)$ is its probability mass function (pmf) and $\mathcal{H}(X)$ is its entropy; if X is continuous, then $p_X(x)$ is its probability density function (pdf) and $\mathcal{H}(X)$ is its differential entropy.

The AEP is proved by observing that $-\log_2 p_X(\mathbf{x})$ is a sum of iid random variables $-\log_2 p_X(x_k)$, so the probability that $-\log_2 p_X(\mathbf{x})$ differs from its mean $N\mathcal{H}(X)$ by more than $\varepsilon > 0$ goes to zero as $N \to \infty$. The Chernoff bound shows that this probability in fact goes to zero exponentially with N.

A consequence of the AEP is that the set T_ε of all sequences $\mathbf{x}$ that are ε-typical with respect to the function $-\log_2 p_X(x)$ has a total probability that approaches 1 as $N \to \infty$. Since for all sequences $\mathbf{x} \in T_\varepsilon$ we have $p_X(\mathbf{x}) \approx 2^{-N\mathcal{H}(X)}$— *i.e.,* the probability distribution $p_X(\mathbf{x})$ is approximately uniform over T_ε— this implies that the "size" $|T_\varepsilon|$ of T_ε is approximately $2^{N\mathcal{H}(X)}$. In the discrete case, the "size" $|T_\varepsilon|$ is the number of sequences in T_ε, whereas in the continuous case $|T_\varepsilon|$ is the volume of T_ε.

In summary, the AEP implies that as $N \to \infty$ the observed sample sequence $\mathbf{x}$ will almost surely lie in an ε-typical set T_ε of size $\approx 2^{N\mathcal{H}(X)}$, and within that set the probability distribution $p_X(\mathbf{x})$ will be approximately uniform.

3.2.4 Fundamental theorem of large-deviation theory

As another application of the law of large numbers, we prove a fundamental theorem of large-deviation theory. A rough statement of this result is as follows: if an iid sequence $\mathbf{X}$ is chosen according to a probability distribution $q(x)$, then the probability that the sequence will be typical of a second probability distribution $p(x)$ is approximately

$$\Pr\{\mathbf{x} \text{ typical for } p \mid q\} \approx e^{-ND(p||q)},$$

where the exponent $D(p||q)$ denotes the relative entropy (Kullback-Leibler divergence)

$$D(p||q) = \mathsf{E}_p\left[\log \frac{p(x)}{q(x)}\right] = \int_{\mathcal{X}} dx\ p(x) \log \frac{p(x)}{q(x)}.$$

Again, $p(x)$ and $q(x)$ denote pmfs in the discrete case and pdfs in the continuous case; we use notation that is appropriate for the continuous case.

Exercise 2 (Gibbs' inequality).

(a) Prove that for $x > 0$, $\log x \leq x - 1$, with equality if and only if $x = 1$.

(b) Prove that for any pdfs $p(x)$ and $q(x)$ over $\mathcal{X}$, $D(p||q) \geq 0$, with equality if and only if $p(x) = q(x)$. □

Given $p(x)$ and $q(x)$, we will now define a sequence $\mathbf{x}$ to be ε-typical with regard to $\log p(x)/q(x)$ if the log likelihood ratio $\lambda(\mathbf{x}) = \log p(\mathbf{x})/q(\mathbf{x})$ is in the range $N(D(p||q) \pm \varepsilon)$, where $D(p||q) = \mathsf{E}_p[\lambda(x)]$ is the mean of $\lambda(x) = \log p(x)/q(x)$ under $p(x)$. Thus an iid sequence $\mathbf{X}$ chosen according to $p(x)$ will almost surely be ε-typical by this definition.

The desired result can then be stated as follows:

Theorem 3.3 (Fundamental theorem of large-deviation theory) *Given two probability distributions $p(x)$ and $q(x)$ on a common alphabet $\mathcal{X}$, for any $\varepsilon > 0$, the probability that an iid random sequence $\mathbf{X}$ drawn according to $q(x)$ is ε-typical for $p(x)$, in the sense that $\log p(\mathbf{x})/q(\mathbf{x})$ is in the range $N(D(p||q) \pm \varepsilon)$, is bounded by*

$$(1 - \delta(N))e^{-N(D(p||q)+\varepsilon)} \leq \Pr\{\mathbf{x}\ \varepsilon\text{−typical for } p \mid q\} \leq e^{-N(D(p||q)-\varepsilon)},$$

where $\delta(N) \to 0$ as $N \to \infty$.

Proof. Define the ε-typical region

$$T_\varepsilon = \{\mathbf{x} \mid N(D(p||q) - \varepsilon) \leq \log \frac{p(\mathbf{x})}{q(\mathbf{x})} \leq N(D(p||q) + \varepsilon)\}.$$

By any law of large numbers, the probability that $\mathbf{X}$ will fall in T_ε goes to 1 as $N \to \infty$; *i.e.,*

$$1 - \delta(N) \leq \int_{T_\varepsilon} d\mathbf{x}\ p(\mathbf{x}) \leq 1,$$

where $\delta(N) \to 0$ as $N \to \infty$. It follows that

$$\begin{aligned}
\int_{T_\varepsilon} d\mathbf{x}\ q(\mathbf{x}) &\leq \int_{T_\varepsilon} d\mathbf{x}\ p(\mathbf{x}) e^{-N(D(p||q)-\varepsilon)} \leq e^{-N(D(p||q)-\varepsilon)}; \\
\int_{T_\varepsilon} d\mathbf{x}\ q(\mathbf{x}) &\geq \int_{T_\varepsilon} d\mathbf{x}\ p(\mathbf{x}) e^{-N(D(p||q)+\varepsilon)} \geq (1 - \delta(N)) e^{-N(D(p||q)+\varepsilon)}.
\end{aligned}$$

□

Since we can choose an arbitrarily small $\varepsilon > 0$ and $\delta(N) > 0$, it follows the exponent $D(p||q)$ is the correct exponent for this probability, in the sense that

$$\lim_{N\to\infty} \frac{\log \Pr\{\mathbf{x} \ \varepsilon\text{−typical for } p \mid q\}}{N} = D(p||q).$$

Exercise 3 (Generalization of Theorem 3.3).

(a) Generalize Theorem 3.3 to the case in which $q(x)$ is a general function over $\mathcal{X}$. State any necessary restrictions on $q(x)$.

(b) Using $q(x) = 1$ in (a), state and prove a form of the Asymptotic Equipartition Principle. □

As an application of Theorem 3.3, we can now prove:

Theorem 3.4 (Correctness of Chernoff exponent) *The Chernoff exponent $E_c(\tau)$ is the correct exponent for $\Pr\{S_N \geq N\tau\}$, in the sense that*

$$\lim_{N\to\infty} \frac{\log \Pr\{S_N \geq N\tau\}}{N} = E_c(\tau),$$

where $S_N = \sum_k x_k$ is the sum of N iid nondeterministic random variables drawn according to some distribution $p(x)$ with mean $\overline{X} < \tau$, and $E_c(\tau) = \max_{s\geq 0} s\tau - \mu(s)$ where $\mu(s) = \log \overline{e^{sX}}$.

Proof. Let s^* be the s that maximizes $s\tau - \mu(s)$ over $s \geq 0$. As we have seen above, for $s = s^*$ the tilted random variable $X(s^*)$ with tilted distribution $q(x, s^*) = p(x)e^{s^*x-\mu(s^*)}$ has mean $\overline{X(s^*)} = \tau$, whereas for $s = 0$ the untilted random variable $X(0)$ with untilted distribution $q(x, 0) = p(x)$ has mean $\overline{X(0)} = \overline{X}$.

Let $q(0)$ denote the untilted distribution $q(x, 0) = p(x)$ with mean $\overline{X(0)} = \overline{X}$, and let $q(s^*)$ denote the optimally tilted distribution $q(x, s^*) = p(x)e^{s^*x-\mu(s^*)}$ with mean $\overline{X(s^*)} = \tau$. Then $\log q(x, s^*)/q(x, 0) = s^*x - \mu(s^*)$, so

$$D(q(s^*)||q(0)) = s^*\tau - \mu(s^*) = E_c(\tau).$$

Moreover, the event that $\mathbf{X}$ is ε-typical with respect to the variable $\log q(x, s^*)/q(x, 0) = s^*x - \mu(s^*)$ under $q(x, 0) = p(x)$ is the event that $s^*S_N - N\mu(s^*)$ is in the range $N(s^*\tau - \mu(s^*) \pm \varepsilon)$, since τ is the mean of X under $q(x, s^*)$. This event is equivalent to S_N being in the range $N(\tau \pm \varepsilon/s^*)$. Since ε may be arbitrarily small, it is clear that the correct exponent of the event $\Pr\{S_N \approx N\tau\}$ is $E_c(\tau)$. This event evidently dominates the probability $\Pr\{S_N \geq N\tau\}$, which we have already shown to be upperbounded by $e^{-NE_c(\tau)}$. □

Exercise 4 (Chernoff bound ⇒ divergence upper bound.)

Using the Chernoff bound, prove that for any two distributions $p(x)$ and $q(x)$ over $\mathcal{X}$,

$$\Pr\{\log \frac{p(\mathbf{x})}{q(\mathbf{x})} \geq ND(p||q) \mid q\} \leq e^{-N(D(p||q))}.$$

[Hint: show that the s that maximizes $s\tau - \mu(s)$ is $s = 1$.] □

3.2.5 Proof of the forward part of the capacity theorem

We now prove that with Shannon's random Gaussian code ensemble and with a slightly different definition of typical-set decoding, we can achieve reliable communication at any rate $\rho < C_{[\mathrm{b/2D}]} = \log_2(1 + \mathrm{SNR})$ b/2D.

We recall that under this scenario the joint pdf of the channel input X and output Y is

$$p_{XY}(x, y) = p_X(x)p_N(y - x) = \frac{1}{\sqrt{2\pi S_x}} e^{-x^2/2S_x} \frac{1}{\sqrt{2\pi S_n}} e^{-(y-x)^2/2S_n}.$$

Since $Y = X + N$, the marginal probability of Y is

$$p_Y(y) = \frac{1}{\sqrt{2\pi S_y}} e^{-y^2/2S_y},$$

where $S_y = S_x + S_n$. On the other hand, since incorrect codewords are independent of the correct codeword and of the output, the joint pdf of an incorrect codeword symbol X' and of Y is

$$q_{XY}(x', y) = p_X(x')p_Y(y) = \frac{1}{\sqrt{2\pi S_x}} e^{-(x')^2/2S_x} \frac{1}{\sqrt{2\pi S_y}} e^{-y^2/2S_y}.$$

We now redefine typical-set decoding as follows. An output sequence $\mathbf{y}$ will be said to be ε-typical for a code sequence $\mathbf{x}$ if

$$\lambda(\mathbf{x}, \mathbf{y}) = \log \frac{p_{XY}(\mathbf{x}, \mathbf{y})}{p_X(\mathbf{x})p_Y(\mathbf{y})} \geq N(D(p_{XY}||p_X p_Y) - \varepsilon).$$

Substituting for the pdfs and recalling that $D(p_{XY}||p_X p_Y) = \frac{1}{2} \log S_y/S_n$, we find that this is equivalent to

$$\frac{||\mathbf{y} - \mathbf{x}||^2}{S_n} \leq \frac{||\mathbf{y}||^2}{S_y} + 2N\varepsilon.$$

Since $||\mathbf{y}||^2/N$ is almost surely very close to its mean S_y, this amounts to asking that $||\mathbf{y}-\mathbf{x}||^2/N$ be very close to its mean S_n under the hypothesis that $\mathbf{x}$ and $\mathbf{y}$ are drawn according to the joint pdf $p_{XY}(x, y)$. The correct codeword will therefore almost surely meet this test.

According to Exercise 4, the probability that any particular incorrect codeword meets the test

$$\lambda(\mathbf{x}, \mathbf{y}) = \log \frac{p_{XY}(\mathbf{x}, \mathbf{y})}{p_X(\mathbf{x})p_Y(\mathbf{y})} \geq ND(p_{XY}||p_X p_Y)$$

is upperbounded by $e^{-ND(p_{XY}||p_X p_Y)} = 2^{-NI(X;Y)}$. If we relax this test by an arbitrarily small number $\varepsilon > 0$, then by the continuity of the Chernoff exponent, the exponent will decrease by an amount $\delta(\varepsilon)$ which can be made arbitrarily small. Therefore we can assert that the probability that a random output sequence $\mathbf{Y}$ will be ε-typical for a random incorrect sequence $\mathbf{X}$ is upperbounded by

$$\Pr\{\mathbf{Y} \ \varepsilon\text{-typical for } \mathbf{X}\} \leq 2^{-N(I(X;Y)-\delta(\varepsilon))},$$

where $\delta(\varepsilon) \to 0$ as $\varepsilon \to 0$.

Now if the random codes have rate $\rho < 2I(X;Y)$ b/2D, then there are $M = 2^{\rho N/2}$ codewords, so by the union bound the total probability of any incorrect codeword being ε-typical is upperbounded by

$$\Pr\{\mathbf{Y} \text{ } \varepsilon\text{-typical for any incorrect } \mathbf{X}\} \leq (M-1)2^{-N(I(X;Y)-\delta(\varepsilon))} < 2^{-N(I(X;Y)-\rho/2-\delta(\varepsilon))}.$$

If $\rho < 2I(X;Y)$ and ε is small enough, then the exponent will be positive and this probability will go to zero as $N \to \infty$.

Thus we have proved the forward part of the capacity theorem: the probability of any kind of error with Shannon's random code ensemble and this variant of typical-set decoding goes to zero as $N \to \infty$, in fact exponentially with N.

3.3 Geometric interpretation and converse

For AWGN channels, the channel capacity theorem has a nice geometric interpretation in terms of the geometry of spheres in real Euclidean N-space $\mathbb{R}^N$.

By any law of large numbers, the probability that the squared Euclidean norm $||\mathbf{X}||^2$ of a random sequence $\mathbf{X}$ of iid Gaussian variables of mean zero and variance S_x per symbol falls in the range $N(S_x \pm \varepsilon)$ goes to 1 as $N \to \infty$, for any $\varepsilon > 0$. Geometrically, the typical region

$$T_\varepsilon = \{\mathbf{x} \in \mathbb{R}^N \mid N(S_x - \varepsilon) \leq ||\mathbf{x}||^2 \leq N(S_x + \varepsilon)\}$$

is a spherical shell with outer squared radius $N(S_x + \varepsilon)$ and inner squared radius $N(S_x - \varepsilon)$. Thus the random N-vector $\mathbf{X}$ will almost surely lie in the spherical shell T_ε as $N \to \infty$. This phenomenon is known as "sphere hardening."

Moreover, the pdf $p_X(\mathbf{x})$ within the spherical shell T_ε is approximately uniform, as we expect from the asymptotic equipartition principle (AEP). Since $p_X(\mathbf{x}) = (2\pi S_x)^{-N/2} \exp -||\mathbf{x}||^2/2S_x$, within T_ε we have

$$(2\pi e S_x)^{-N/2} e^{-(N/2)(\varepsilon/S_x)} \leq p_X(\mathbf{x}) \leq (2\pi e S_x)^{-N/2} e^{(N/2)(\varepsilon/S_x)}.$$

Moreover, the fact that $p_X(\mathbf{x}) \approx (2\pi e S_x)^{-N/2}$ implies that the volume of T_ε is approximately $|T_\varepsilon| \approx (2\pi e S_x)^{N/2}$. More precisely, we have

$$1 - \delta(N) \leq \int_{T_\varepsilon} p_X(\mathbf{x}) \, d\mathbf{x} \leq 1,$$

where $\delta(N) \to 0$ as $N \to \infty$. Since $|T_\varepsilon| = \int_{T_\varepsilon} d\mathbf{x}$, we have

$$\begin{aligned} 1 \geq (2\pi e S_x)^{-N/2} e^{-(N/2)(\varepsilon/S_x)} |T_\varepsilon| &\Rightarrow |T_\varepsilon| \leq (2\pi e S_x)^{N/2} e^{(N/2)(\varepsilon/S_x)}; \\ 1 - \delta(N) \leq (2\pi e S_x)^{-N/2} e^{(N/2)(\varepsilon/S_x)} |T_\varepsilon| &\Rightarrow |T_\varepsilon| \geq (1-\delta(N))(2\pi e S_x)^{N/2} e^{-(N/2)(\varepsilon/S_x)}. \end{aligned}$$

Since these bounds hold for any $\varepsilon > 0$, this implies that

$$\lim_{N\to\infty} \frac{\log |T_\varepsilon|}{N} = \frac{1}{2} \log 2\pi e S_x = \mathcal{H}(X),$$

where $\mathcal{H}(X) = \frac{1}{2} \log 2\pi e S_x$ denotes the differential entropy of a Gaussian random variable with mean zero and variance S_x.

We should note at this point that practically all of the volume of an N-sphere of squared radius $N(S_x+\varepsilon)$ lies within the spherical shell $|T_\varepsilon|$ as $N\to\infty$, for any $\varepsilon>0$. By dimensional analysis, the volume of an N-sphere of radius r must be given by $A_N r^N$ for some constant A_N that does not depend on r. Thus the ratio of the volume of an N-sphere of squared radius $N(S_x-\varepsilon)$ to that of an N-sphere of squared radius $N(S_x+\varepsilon)$ must satisfy

$$\frac{A_N(N(S_x-\varepsilon))^{N/2}}{A_N(N(S_x+\varepsilon))^{N/2}}=\left(\frac{S_x-\varepsilon}{S_x+\varepsilon}\right)^{N/2}\to 0 \text{ as } N\to\infty, \text{ for any } \varepsilon>0.$$

It follows that the volume of an N-sphere of squared radius NS_x is also approximated by $e^{N\mathcal{H}(X)}=(2\pi eS_x)^{N/2}$ as $N\to\infty$.

Exercise 5. In Exercise 4 of Chapter 1, the volume of an N-sphere of radius r was given as

$$V_\otimes(N,r)=\frac{(\pi r^2)^{N/2}}{(N/2)!},$$

for N even. In other words, $A_N=\pi^{N/2}/((N/2)!)$. Using Stirling's approximation, $m!\to(m/e)^m$ as $m\to\infty$, show that this exact expression leads to the same asymptotic approximation for $V_\otimes(N,r)$ as was obtained above by use of the asymptotic equipartition principle. □

The sphere-hardening phenomenon may seem somewhat bizarre, but even more unexpected phenomena occur when we code for the AWGN channel using Shannon's random code ensemble.

In this case, each randomly chosen transmitted N-vector $\mathbf{X}$ will almost surely lie in a spherical shell T_X of squared radius $\approx NS_x$, and the random received N-vector $\mathbf{Y}$ will almost surely lie in a spherical shell T_Y of squared radius $\approx NS_y$, where $S_y=S_x+S_n$.

Moreover, given the correct transmitted codeword $\mathbf{c}_0$, the random received vector $\mathbf{Y}$ will almost surely lie in a spherical shell $T_\varepsilon(\mathbf{c}_0)$ of squared radius $\approx NS_n$ centered on $\mathbf{c}_0$. A further consequence of the AEP is that almost all of the volume of this nonzero-mean shell, whose center $\mathbf{c}_0$ has squared Euclidean norm $||\mathbf{c}_0||^2\approx NS_x$, lies in the zero-mean shell T_Y whose squared radius is $\approx NS_y$, since the expected squared Euclidean norm of $\mathbf{Y}=\mathbf{c}_0+\mathbf{N}$ is

$$\mathsf{E}_N[||\mathbf{Y}||^2]=||\mathbf{c}_0||^2+NS_n\approx NS_y.$$

"Curiouser and curiouser," said Alice.

We thus obtain the following geometrical picture. We choose $M=2^{\rho N/2}$ code vectors at random according to a zero-mean Gaussian distribution with variance S_x, which almost surely puts them within the shell T_X of squared radius $\approx NS_x$. Considering the probable effects of a random noise sequence $\mathbf{N}$ distributed according to a zero-mean Gaussian distribution with variance S_n, we can define for each code vector $\mathbf{c}_i$ a typical region $T_\varepsilon(\mathbf{c}_i)$ of volume $|T_\varepsilon(\mathbf{c}_i)|\approx(2\pi eS_n)^{N/2}$, which falls almost entirely within the shell T_Y of volume $|T_Y|\approx(2\pi eS_y)^{N/2}$.

Now if a particular code vector $\mathbf{c}_0$ is sent, then the probability that the received vector $\mathbf{y}$ will fall in the typical region $T_\varepsilon(\mathbf{c}_0)$ is nearly 1. On the other hand, the probability that $\mathbf{y}$ will fall in the typical region $T_\varepsilon(\mathbf{c}_i)$ of some other independently-chosen code vector $\mathbf{c}_i$ is approximately equal to the ratio $|T_\varepsilon(\mathbf{c}_i)|/|T_Y|$ of the volume of $T_\varepsilon(\mathbf{c}_i)$ to that of the entire shell, since if $\mathbf{y}$ is generated according to $p_y(\mathbf{y})$ independently of $\mathbf{c}_i$, then it will be approximately uniformly distributed over T_Y. Thus this probability is approximately

$$\Pr\{\mathbf{Y}\text{ typical for }\mathbf{c}_i\}\approx\frac{|T_\varepsilon(\mathbf{c}_i)|}{|T_Y|}\approx\frac{(2\pi eS_n)^{N/2}}{(2\pi eS_y)^{N/2}}=\left(\frac{S_n}{S_y}\right)^{N/2}.$$

As we have seen in earlier sections, this argument may be made precise.

It follows then that if $\rho < \log_2(1 + S_x/S_n)$ b/2D, or equivalently $M = 2^{\rho N/2} < (S_y/S_n)^{N/2}$, then the probability that $\mathbf{Y}$ is typical with respect to any of the $M - 1$ incorrect codewords is very small, which proves the forward part of the channel capacity theorem.

On the other hand, it is clear from this geometric argument that if $\rho > \log_2(1 + S_x/S_n)$ b/2D, or equivalently $M = 2^{\rho N/2} > (S_y/S_n)^{N/2}$, then the probability of decoding error must be large. For the error probability to be small, the decision region for each code vector $\mathbf{c}_i$ must include almost all of its typical region $T_\varepsilon(\mathbf{c}_i)$. If the volume of the $M = 2^{\rho N/2}$ typical regions exceeds the volume of T_Y, then this is impossible. Thus in order to have small error probability we must have

$$2^{\rho N/2}(2\pi e S_n)^{N/2} \leq (2\pi e S_y)^{N/2} \quad \Rightarrow \quad \rho \leq \log_2 \frac{S_y}{S_n} = \log_2(1 + \frac{S_x}{S_n}) \quad \text{b/2D}.$$

This argument may also be made precise, and is the converse to the channel capacity theorem.

In conclusion, we obtain the following picture of a capacity-achieving code. Let T_Y be the N-shell of squared radius $\approx NS_y$, which is almost the same thing as the N-sphere of squared radius NS_y. A capacity-achieving code consists of the centers $\mathbf{c}_i$ of M typical regions $T_\varepsilon(\mathbf{c}_i)$, where $||\mathbf{c}_i||^2 \approx NS_x$ and each region $T_\varepsilon(\mathbf{c}_i)$ consists of an N-shell of squared radius $\approx NS_n$ centered on $\mathbf{c}_i$, which is almost the same thing as an N-sphere of squared radius NS_x. As $\rho \to C_{[\text{b/2D}]} = \log_2(1 + \frac{S_x}{S_n})$ b/2D, these regions $T_\varepsilon(\mathbf{c}_i)$ form an almost disjoint partition of T_Y. This picture is illustrated in Figure 2.

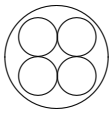


Figure 2. Packing $\approx (S_y/S_n)^{N/2}$ typical regions $T_\varepsilon(\mathbf{c}_i)$ of squared radius $\approx NS_n$ into a large typical region T_Y of squared radius $\approx NS_y$.

3.3.1 Discussion

It is natural in view of the above picture to frame the problem of coding for the AWGN channel as a sphere-packing problem. In other words, we might expect that a capacity-achieving code basically induces a disjoint partition of an N-sphere of squared radius NS_y into about $(S_y/S_n)^{N/2}$ disjoint decision regions, such that each decision region includes the sphere of squared radius NS_n about its center.

However, it can be shown by geometric arguments that such a disjoint partition is impossible as the code rate approaches capacity. What then is wrong with the sphere-packing approach? The subtle distinction that makes all the difference is that Shannon's probabilistic approach does not require decision regions to be disjoint, but merely probabilistically almost disjoint. So the solution to Shannon's coding problem involves what might be called "soft sphere-packing."

We will see that hard sphere-packing— *i.e.,* maximizing the minimum distance between code vectors subject to a constraint on average energy— is a reasonable approach for moderate-size codes at rates not too near to capacity. However, to obtain reliable transmission at rates near capacity, we will need to consider probabilistic codes and decoding algorithms that follow more closely the spirit of Shannon's original work.

Chapter 4

The gap between uncoded performance and the Shannon limit

The channel capacity theorem gives a sharp upper limit $C_{[\text{b/2D}]} = \log_2(1 + \text{SNR})$ b/2D on the rate (nominal spectral efficiency) ρ b/2D of any reliable transmission scheme. However, it does not give constructive coding schemes that can approach this limit. Finding such schemes has been the main problem of coding theory and practice for the past half century, and will be our main theme in this book.

We will distinguish sharply between the power-limited regime, where the nominal spectral efficiency ρ is small, and the bandwidth-limited regime, where ρ is large. In the power-limited regime, we will take 2-PAM as our baseline uncoded scheme, whereas in the bandwidth-limited regime, we will take M-PAM (or equivalently $(M \times M)$-QAM) as our baseline.

By evaluating the performance of these simplest possible uncoded modulation schemes and comparing baseline performance to the Shannon limit, we will establish how much "coding gain" is possible.

4.1 Discrete-time AWGN channel model

We have seen that with orthonormal PAM or QAM, the channel model reduces to an analogous real or complex discrete-time AWGN channel model

$$\mathbf{Y} = \mathbf{X} + \mathbf{N},$$

where $\mathbf{X}$ is the random input signal point sequence, and $\mathbf{N}$ is an independent iid Gaussian noise sequence with mean zero and variance $\sigma^2 = N_0/2$ per real dimension. We have also seen that there is no essential difference between the real and complex versions of this model, so from now on we will consider only the real model.

We recapitulate the connections between the parameters of this model and the corresponding continuous-time parameters. If the symbol rate is $1/T$ real symbols/s (real dimensions per second), the bit rate per two dimensions is ρ b/2D, and the average signal energy per two dimensions is E_s, then:

- The nominal bandwidth is $W = 1/2T$ Hz;
- The data rate is $R = \rho W$ b/s, and the nominal spectral efficiency is ρ (b/s)/Hz;
- The signal power (average energy per second) is $P = E_s W$;
- The signal-to-noise ratio is $\text{SNR} = E_s/N_0 = P/N_0 W$;
- The channel capacity in b/s is $C_{[\text{b/s}]} = W C_{[\text{b/2D}]} = W \log_2(1 + \text{SNR})$ b/s.

4.2 Normalized SNR and E_b/N_0

In this section we introduce two normalized measures of SNR that are suggested by the capacity bound $\rho < C_{[\text{b/2D}]} = \log_2(1 + \text{SNR})$ b/2D, which we will now call the *Shannon limit*.

An equivalent statement of the Shannon limit is that for a coding scheme with rate ρ b/2D, if the error probability is to be small, then the SNR must satisfy

$$\text{SNR} > 2^\rho - 1.$$

This motivates the definition of the *normalized signal-to-noise ratio* SNR_{norm} as

$$\text{SNR}_{\text{norm}} = \frac{\text{SNR}}{2^\rho - 1}. \tag{4.1}$$

SNR_{norm} is commonly expressed in dB. Then the Shannon limit may be expressed as

$$\text{SNR}_{\text{norm}} > 1 \text{ (0 dB)}.$$

Moreover, the value of SNR_{norm} in dB measures how far a given coding scheme is operating from the Shannon limit, in dB (the "gap to capacity").

Another commonly used normalized measure of signal-to-noise ratio is E_b/N_0, where E_b is the average signal energy per information bit and N_0 is the noise variance per two dimensions. Note that since $E_b = E_s/\rho$, where E_s is the average signal energy per two dimensions, we have

$$E_b/N_0 = E_s/\rho N_0 = \text{SNR}/\rho.$$

The quantity E_b/N_0 is sometimes called the "signal-to-noise ratio per information bit," but it is not really a signal-to-noise ratio, because its numerator and denominator do not have the same units. It is probably best just to call it "E_b/N_0" (pronounced "eebee over enzero" or "ebno"). E_b/N_0 is commonly expressed in dB.

Since $\text{SNR} > 2^\rho - 1$, the Shannon limit on E_b/N_0 may be expressed as

$$E_b/N_0 > \frac{2^\rho - 1}{\rho}. \tag{4.2}$$

Notice that the Shannon limit on E_b/N_0 is a monotonic function of ρ. For $\rho = 2$, it is equal to 3/2 (1.76 dB); for $\rho = 1$, it is equal to 1 (0 dB); and as $\rho \to 0$, it approaches $\ln 2 \approx 0.69$ (-1.59 dB), which is called the *ultimate Shannon limit* on E_b/N_0.

4.3 Power-limited and bandwidth-limited channels

Ideal band-limited AWGN channels may be classified as bandwidth-limited or power-limited according to whether they permit transmission at high spectral efficiencies or not. There is no sharp dividing line, but we will take $\rho = 2$ b/2D or (b/s)/Hz as the boundary, corresponding to the highest spectral efficiency that can be achieved with binary transmission.

We note that the behavior of the Shannon limit formulas is very different in the two regimes. If SNR is small (the *power-limited regime*), then we have

$$\begin{aligned} \rho < \log_2(1+\mathrm{SNR}) &\approx \mathrm{SNR}\log_2 e; \\ \mathrm{SNR}_{\mathrm{norm}} &\approx \frac{\mathrm{SNR}}{\rho \ln 2} = (E_b/N_0)\log_2 e. \end{aligned}$$

In words, in the power-limited regime, the capacity (achievable spectral efficiency) increases linearly with SNR, and as $\rho \to 0$, $\mathrm{SNR}_{\mathrm{norm}}$ becomes equivalent to E_b/N_0, up to a scale factor of $\log_2 e = 1/\ln 2$. Thus as $\rho \to 0$ the Shannon limit $\mathrm{SNR}_{\mathrm{norm}} > 1$ translates to the ultimate Shannon limit on E_b/N_0, namely $E_b/N_0 > \ln 2$.

On the other hand, if SNR is large (the *bandwidth-limited regime*), then we have

$$\begin{aligned} \rho < \log_2(1+\mathrm{SNR}) &\approx \log_2 \mathrm{SNR}; \\ \mathrm{SNR}_{\mathrm{norm}} &\approx \frac{\mathrm{SNR}}{2^\rho}. \end{aligned}$$

Thus in the bandwidth-limited regime, the capacity (achievable spectral efficiency) increases logarithmically with SNR, which is dramatically different from the linear behavior in the power-limited regime. In the power-limited regime, every doubling of SNR doubles the achievable rate, whereas in the bandwidth-limited regime, every additional 3 dB in SNR yields an increase in achievable spectral efficiency of only 1 b/2D or 1 (b/s)/Hz.

Example 1. A standard voice-grade telephone channel may be crudely modeled as an ideal band-limited AWGN channel with $W \approx 3500$ Hz and $\mathrm{SNR} \approx 37$ dB. The Shannon limit on spectral efficiency and bit rate of such a channel are roughly $\rho < 37/3 \approx 12.3$ (b/s)/Hz and $R <$ 43,000 b/s. Increasing the SNR by 3 dB would increase the achievable spectral efficiency ρ by only 1 (b/s)/Hz, or the bit rate R by only 3500 b/s. □

Example 2. In contrast, there are no bandwidth restrictions on a deep-space communication channel. Therefore it makes sense to use as much bandwidth as possible, and operate deep in the power-limited region. In this case the bit rate is limited by the ultimate Shannon limit on E_b/N_0, namely $E_b/N_0 > \ln 2$ (-1.59 dB). Since $E_b/N_0 = P/RN_0$, the Shannon limit becomes $R < (P/N_0)/(\ln 2)$. Increasing P/N_0 by 3 dB will now double the achievable rate R in b/s. □

We will find that the power-limited and bandwidth-limited regimes differ in almost every way. In the power-limited regime, we will be able to use binary coding and modulation, whereas in the bandwidth-limited regime we must use nonbinary ("multilevel") modulation. In the power-limited regime, it is appropriate to normalize everything "per information bit," and E_b/N_0 is a reasonable normalized measure of signal-to-noise ratio. In the bandwidth-limited regime, on the other hand, we will see that it is much better to normalize everything "per two dimensions," and $\mathrm{SNR}_{\mathrm{norm}}$ will become a much more appropriate measure than E_b/N_0. Thus the first thing to do in a communications design problem is to determine which regime you are in, and then proceed accordingly.

4.4 Performance of M-PAM and $(M \times M)$-QAM

We now evaluate the performance of the simplest possible uncoded systems, namely M-PAM and $(M \times M)$-QAM. This will give us a baseline. The difference between the performance achieved by baseline systems and the Shannon limit determines the maximum possible gain that can be achieved by the most sophisticated coding systems. In effect, it defines our playing field.

4.4.1 Uncoded 2-PAM

We first consider the important special case of a binary 2-PAM constellation

$$\mathcal{A} = \{-\alpha, +\alpha\},$$

where $\alpha > 0$ is a scale factor chosen such that the average signal energy per bit, $E_b = \alpha^2$, satisfies the average signal energy constraint.

For this constellation, the bit rate (nominal spectral efficiency) is $\rho = 2$ b/2D, and the average signal energy per bit is $E_b = \alpha^2$.

The usual symbol-by-symbol detector (which is easily seen to be optimum) makes an independent decision on each received symbol y_k according to whether the sign of y_k is positive or negative. The probability of error per bit is evidently the same regardless of which of the two signal values is transmitted, and is equal to the probability that a Gaussian noise variable of variance $\sigma^2 = N_0/2$ exceeds α, namely

$$P_b(E) = Q\left(\alpha/\sigma\right),$$

where the Gaussian probability of error $Q(\cdot)$ function is defined by

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

Substituting the energy per bit $E_b = \alpha^2$ and the noise variance $\sigma^2 = N_0/2$, the probability of error per bit is

$$P_b(E) = Q\left(\sqrt{2E_b/N_0}\right). \tag{4.3}$$

This gives the performance curve of $P_b(E)$ *vs.* E_b/N_0 for uncoded 2-PAM that is shown in Figure 1, below.

4.4.2 Power-limited baseline *vs.* the Shannon limit

In the power-limited regime, we will take binary pulse amplitude modulation (2-PAM) as our baseline uncoded system, since it has $\rho = 2$. By comparing the performance of the uncoded baseline system to the Shannon limit, we will be able to determine the maximum possible gains that can be achieved by the most sophisticated coding systems.

In the power-limited regime, we will primarily use E_b/N_0 as our normalized signal-to-noise ratio, although we could equally well use $\mathrm{SNR}_{\mathrm{norm}}$. Note that when $\rho = 2$, since $E_b/N_0 = \mathrm{SNR}/\rho$ and $\mathrm{SNR}_{\mathrm{norm}} = \mathrm{SNR}/(2^\rho - 1)$, we have $2E_b/N_0 = 3\mathrm{SNR}_{\mathrm{norm}}$. The baseline performance curve can therefore be written in two equivalent ways:

$$P_b(E) = Q\left(\sqrt{2E_b/N_0}\right) = Q\left(\sqrt{3\mathrm{SNR}_{\mathrm{norm}}}\right).$$

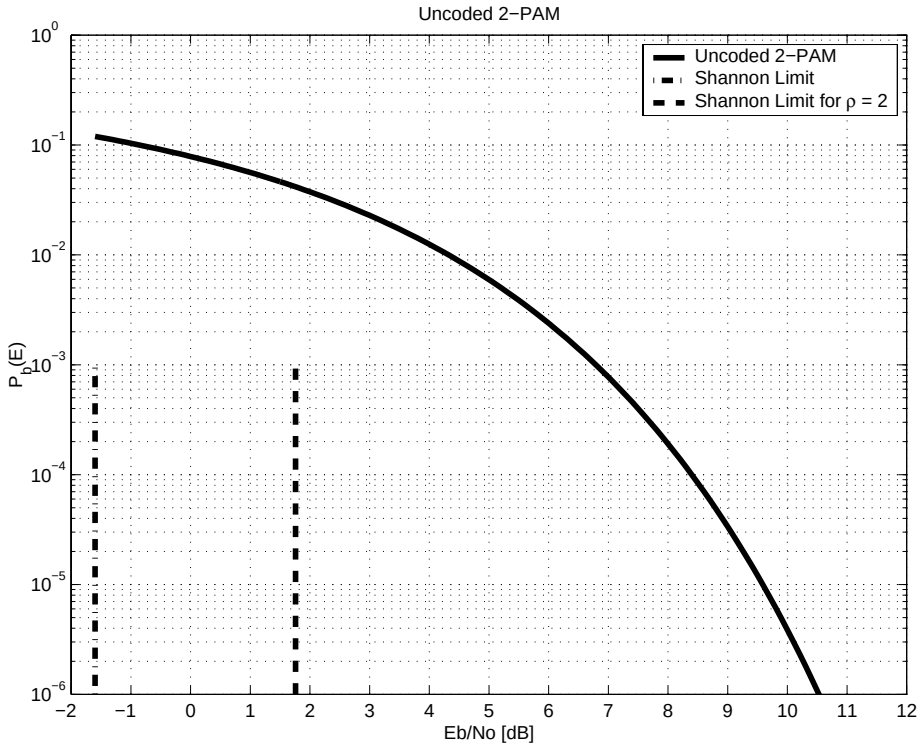


Figure 1. $P_b(E)$ *vs.* E_b/N_0 for uncoded binary PAM.

Figure 1 gives us a universal design tool. For example, if we want to achieve $P_b(E) \approx 10^{-5}$ with uncoded 2-PAM, then we know that we will need to achieve $E_b/N_0 \approx 9.6$ dB.

We may also compare the performance shown in Figure 1 to the Shannon limit. The rate of 2-PAM is $\rho = 2$ b/2D. The Shannon limit on E_b/N_0 at $\rho = 2$ b/2D is $E_b/N_0 > (2^\rho - 1)/\rho = 3/2$ (1.76 dB). Thus if our target error rate is $P_b(E) \approx 10^{-5}$, then we can achieve a coding gain of up to about 8 dB with powerful codes, at the same rate of $\rho = 2$ b/2D.

However, if there is no limit on bandwidth and therefore no lower limit on spectral efficiency, then it makes sense to let $\rho \to 0$. In this case the ultimate Shannon limit on E_b/N_0 is $E_b/N_0 > \ln 2$ (-1.59 dB). Thus if our target error rate is $P_b(E) \approx 10^{-5}$, then Shannon says that we can achieve a coding gain of over 11 dB with powerful codes, by letting the spectral efficiency approach zero.

4.4.3 Uncoded M-PAM and $(M \times M)$-QAM

We next consider the more general case of an M-PAM constellation

$$\mathcal{A} = \alpha\{\pm 1, \pm 3, \ldots, \pm(M-1)\},$$

where $\alpha > 0$ is again a scale factor chosen to satisfy the average signal energy constraint. The bit rate (nominal spectral efficiency) is then $\rho = 2 \log_2 M$ b/2D.

The average energy per M-PAM symbol is

$$E(\mathcal{A}) = \frac{\alpha^2(M^2-1)}{3}.$$

An elegant way of making this calculation is to consider a random variable $Z = X + U$, where X is equiprobable over $\mathcal{A}$ and U is an independent continuous uniform random variable over the interval $(-\alpha, \alpha]$. Then Z is a continuous random variable over the interval $(-M\alpha, M\alpha]$, and[1]

$$\overline{X^2} = \overline{Z^2} - \overline{U^2} = \frac{(\alpha M)^2}{3} - \frac{\alpha^2}{3}.$$

The average energy per two dimensions is then $E_s = 2E(\mathcal{A}) = 2\alpha^2(M^2 - 1)/3$.

Again, an optimal symbol-by-symbol detector makes an independent decision on each received symbol y_k. In this case the decision region associated with an input value αz_k (where z_k is an odd integer) is the interval $\alpha[z_k - 1, z_k + 1]$ (up to tie-breaking at the boundaries, which is immaterial), except for the two outermost signal values $\pm\alpha(M-1)$, which have decision regions $\pm\alpha[M-2, \infty)$. The probability of error for any of the $M-2$ inner signals is thus equal to twice the probability that a Gaussian noise variable N_k of variance $\sigma^2 = N_0/2$ exceeds α, namely $2Q(\alpha/\sigma)$, whereas for the two outer signals it is just $Q(\alpha/\sigma)$. The average probability of error with equiprobable signals per M-PAM symbol is thus

$$\Pr(E) = \frac{M-2}{M} 2Q(\alpha/\sigma) + \frac{2}{M} Q(\alpha/\sigma) = \frac{2(M-1)}{M} Q(\alpha/\sigma).$$

For $M = 2$, this reduces to the usual expression for 2-PAM. For $M \geq 4$, the "error coefficient" $2(M-1)/M$ quickly approaches 2, so $\Pr(E) \approx 2Q(\alpha/\sigma)$.

Since an $(M \times M)$-QAM signal set $\mathcal{A}' = \mathcal{A}^2$ is equivalent to two independent M-PAM transmissions, we can easily extend this calculation to $(M \times M)$-QAM. The bit rate (nominal spectral efficiency) is again $\rho = 2\log_2 M$ b/2D, and the average signal energy per two dimensions (per QAM symbol) is again $E_s = 2\alpha^2(M^2-1)/3$. The same dimension-by-dimension decision method and calculation of probability of error per dimension $\Pr(E)$ hold. The probability of error per $(M \times M)$-QAM symbol, or per two dimensions, is given by

$$P_s(E) = 1 - (1 - \Pr(E))^2 = 2\Pr(E) - (\Pr(E))^2 \approx 2\Pr(E).$$

Therefore for $(M \times M)$-QAM we obtain a probability of error per two dimensions of

$$P_s(E) \approx 2\Pr(E) \approx 4Q(\alpha/\sigma). \tag{4.4}$$

4.4.4 Bandwidth-limited baseline *vs.* the Shannon limit

In the bandwidth-limited regime, we will take $(M \times M)$-QAM with $M \geq 4$ as our baseline uncoded system, we will normalize everything per two dimensions, and we will use $\mathrm{SNR}_{\mathrm{norm}}$ as our normalized signal-to-noise ratio.

For $M \geq 4$, the probability of error per two dimensions is given by (4.4):

$$P_s(E) \approx 4Q(\alpha/\sigma).$$

[1] This calculation is actually somewhat fundamental, since it is based on a perfect one-dimensional sphere-packing and on the fact that the difference between the average energy of a continuous random variable and the average energy of an optimally quantized discrete version thereof is the average energy of the quantization error. As the same principle is used in the calculation of channel capacity, in the relation $S_y = S_x + S_n$, we can even say that the "1" that appears in the capacity formula is the same "1" as appears in the formula for $E(\mathcal{A})$. Therefore the cancellation of this term below is not quite as miraculous as it may at first seem.

Substituting the average energy $E_s = 2\alpha^2(M^2-1)/3$ per two dimensions, the noise variance $\sigma^2 = N_0/2$, and the normalized signal-to-noise ratio

$$\mathrm{SNR}_{\mathrm{norm}} = \frac{\mathrm{SNR}}{2^\rho - 1} = \frac{E_s/N_0}{M^2-1},$$

we find that the factors of M^2-1 cancel (*cf.* footnote 1) and we obtain the performance curve

$$P_s(E) \approx 4Q\left(\sqrt{3\mathrm{SNR}_{\mathrm{norm}}}\right). \tag{4.5}$$

Note that this curve does not depend on M, which shows that $\mathrm{SNR}_{\mathrm{norm}}$ is correctly normalized for the bandwidth-limited regime.

The bandwidth-limited baseline performance curve (4.5) of $P_s(E)$ *vs.* $\mathrm{SNR}_{\mathrm{norm}}$ for uncoded $(M \times M)$-QAM is plotted in Figure 2.

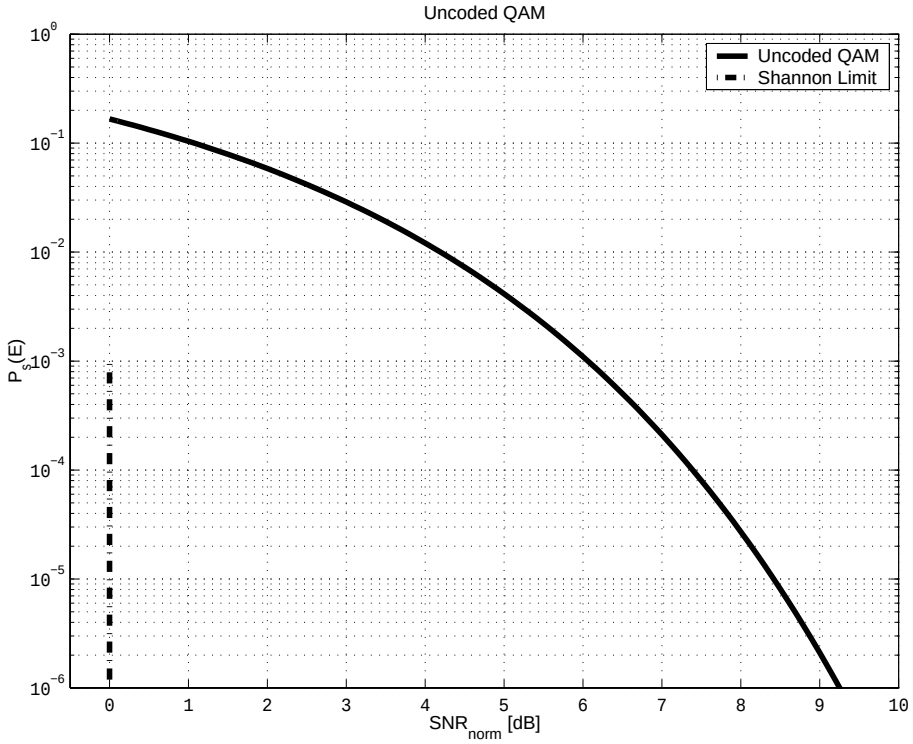


Figure 2. $P_s(E)$ *vs.* $\mathrm{SNR}_{\mathrm{norm}}$ for uncoded $(M \times M)$-QAM.

Figure 2 gives us another universal design tool. For example, if we want to achieve $P_s(E) \approx 10^{-5}$ with uncoded $(M \times M)$-QAM (or M-PAM), then we know that we will need to achieve $\mathrm{SNR}_{\mathrm{norm}} \approx 8.4$ dB. (Notice that $\mathrm{SNR}_{\mathrm{norm}}$, unlike E_b/N_0, is already normalized for spectral efficiency.)

The Shannon limit on $\mathrm{SNR}_{\mathrm{norm}}$ for any spectral efficiency is $\mathrm{SNR}_{\mathrm{norm}} > 1$ (0 dB). Thus if our target error rate is $P_s(E) \approx 10^{-5}$, then Shannon says that we can achieve a coding gain of up to about 8.4 dB with powerful codes, at any spectral efficiency. (This result holds approximately even for $M = 2$, as we have already seen in the previous subsection.)

Chapter 5

Performance of small signal sets

In this chapter, we show how to estimate the performance of small-to-moderate-sized signal constellations on the discrete-time AWGN channel.

With equiprobable signal points in iid Gaussian noise, the optimum decision rule is a minimum-distance rule, so the optimum decision regions are minimum-distance (Voronoi) regions.

We develop useful performance estimates for the error probability based on the union bound. These are based on exact expressions for pairwise error probabilities, which involve the Gaussian probability of error $Q(\cdot)$ function. An appendix develops the main properties of this function.

Finally, we use the union bound estimate to find the "coding gain" of small-to-moderate-sized signal constellations in the power-limited and bandwidth-limited regimes, compared to the 2-PAM or $(M \times M)$-QAM baselines, respectively.

5.1 Signal constellations for the AWGN channel

In general, a coding scheme for the discrete-time AWGN channel model $\mathbf{Y} = \mathbf{X} + \mathbf{N}$ is a method of mapping an input bit sequence into a transmitted real symbol sequence $\mathbf{x}$, which is called encoding, and a method for mapping a received real symbol sequence $\mathbf{y}$ into an estimated transmitted signal sequence $\hat{\mathbf{x}}$, which is called decoding.

Initially we will consider coding schemes of the type considered by Shannon, namely block codes with a fixed block length N. With such codes, the transmitted sequence $\mathbf{x}$ consists of a sequence $(\ldots, \mathbf{x}_k, \mathbf{x}_{k+1}, \ldots)$ of N-tuples $\mathbf{x}_k \in \mathbb{R}^N$ that are chosen *independently* from some block code of length N with M codewords. Block codes are not the only possible kinds of coding schemes, as we will see when we study convolutional and trellis codes.

Usually the number M of codewords is chosen to be a power of 2, and codewords are chosen by some encoding map from blocks of $\log_2 M$ bits in the input bit sequence. If the input bit sequence is assumed to be an iid random sequence of equiprobable bits, then the transmitted sequence will be an iid random sequence $\mathbf{X} = (\ldots, \mathbf{X}_k, \mathbf{X}_{k+1}, \ldots)$ of equiprobable random codewords $\mathbf{X}_k$. We almost always assume equiprobability, because this is a worst-case (minimax) assumption. Also, the bit sequence produced by an efficient source coder must statistically resemble an iid equiprobable bit sequence.

In digital communications, we usually focus entirely on the code, and do not care what encoding map is used from bits to codewords. In other contexts the encoding map is also important; *e.g.,* in the "Morse code" of telegraphy.

If the block length N and the number of codewords M are relatively small, then a block code for the AWGN channel may alternatively be called a signal set, signal constellation, or signal alphabet. A scheme in which the block length N is 1 or 2, corresponding to a single signaling interval of PAM or QAM, may be regarded as an "uncoded" scheme.

Figure 1 illustrates some 1-dimensional and 2-dimensional signal constellations.

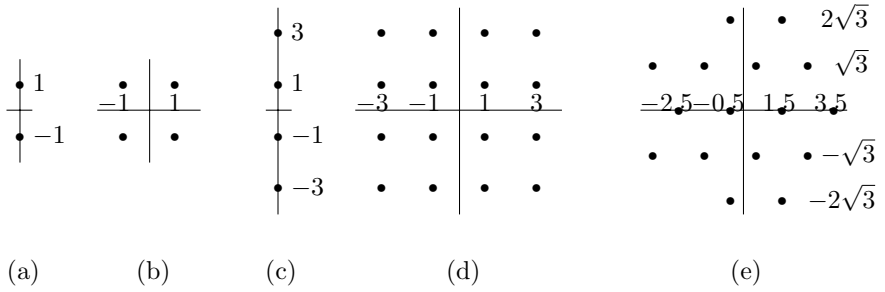


Figure 1. Uncoded signal constellations: (a) 2-PAM; (b) (2×2)-QAM; (c) 4-PAM; (d) (4×4)-QAM; (e) hexagonal 16-QAM.

An N-dimensional *signal constellation* (set, alphabet) will be denoted by

$$\mathcal{A} = \{\mathbf{a}_j, 1 \le j \le M\}.$$

Its M elements $\mathbf{a}_j \in \mathbb{R}^N$ will be called *signal points* (vectors, N-tuples).

The basic parameters of a signal constellation $\mathcal{A} = \{\mathbf{a}_j, 1 \le j \le M\}$ are its *dimension* N; its *size* M (number of signal points); its *average energy* $E(\mathcal{A}) = \frac{1}{M}\sum_j ||\mathbf{a}_j||^2$; and its *minimum squared distance* $d^2_{\min}(\mathcal{A})$, which is an elementary measure of its noise resistance. A secondary parameter is the average number $K_{\min}(\mathcal{A})$ of nearest neighbors (points at distance $d_{\min}(\mathcal{A})$).

From these basic parameters we can derive such parameters as:

- The bit rate (nominal spectral efficiency) $\rho = (2/N)\log_2 M$ b/2D;
- The average energy per two dimensions $E_s = (2/N)E(\mathcal{A})$,
  or the average energy per bit $E_b = E(\mathcal{A})/(\log_2 M) = E_s/\rho$;
- Energy-normalized figures of merit such as $d^2_{\min}(\mathcal{A})/E(\mathcal{A}), d^2_{\min}(\mathcal{A})/E_s$ or $d^2_{\min}(\mathcal{A})/E_b$, which are independent of scale.

For example, in Figure 1, the bit rate (nominal spectral efficiency) of the 2-PAM and (2×2)-QAM constellations is $\rho = 2$ b/2D, whereas for the other three constellations it is $\rho = 4$ b/2D. The average energy per two dimensions of the 2-PAM and (2×2)-QAM constellations is $E_s = 2$, whereas for the 4-PAM and (4×4)-QAM constellations it is $E_s = 10$, and for the hexagonal 16-QAM constellation it is $E_s = 8.75$. For all constellations, $d^2_{\min} = 4$. The average numbers of nearest neighbors are $K_{\min} = 1, 2, 1.5, 3$, and 4.125, respectively.

5.1.1 Cartesian-product constellations

Some of these relations are explained by the fact that an $(M \times M)$-QAM constellation is the Cartesian product of two M-PAM constellations. In general, a *Cartesian-product constellation* $\mathcal{A}^K$ is the set of all sequences of K points from an elementary constellation $\mathcal{A}$; *i.e.,*

$$\mathcal{A}^K = \{(\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_K) \mid \mathbf{x}_k \in \mathcal{A}\}.$$

If the dimension and size of $\mathcal{A}$ are N and M, respectively, then the dimension of $\mathcal{A}' = \mathcal{A}^K$ is $N' = KN$ and its size is $M' = M^K$.

Exercise 1 (Cartesian-product constellations). (a) Show that if $\mathcal{A}' = \mathcal{A}^K$, then the parameters $N, \log_2 M, E(\mathcal{A}')$ and $K_{\min}(\mathcal{A}')$ of $\mathcal{A}'$ are K times as large as the corresponding parameters of $\mathcal{A}$, whereas the normalized parameters ρ, E_s, E_b and $d^2_{\min}(\mathcal{A})$ are the same as those of $\mathcal{A}$. Verify that these relations hold for the $(M \times M)$-QAM constellations of Figure 1. □

Notice that there is no difference between a random input sequence $\mathbf{X}$ with elements from $\mathcal{A}$ and a sequence $\mathbf{X}$ with elements from a Cartesian-product constellation $\mathcal{A}^K$. For example, there is no difference between a random M-PAM sequence and a random $(M \times M)$-QAM sequence. Thus Cartesian-product constellations capture in a non-statistical way the idea of independent transmissions. We thus may regard a Cartesian-product constellation $\mathcal{A}^K$ as equivalent to (or a "version" of) the elementary constellation $\mathcal{A}$. In particular, it has the same ρ, E_s, E_b and $d^2_{\min}$.

We may further define a "code over $\mathcal{A}$" as a subset $\mathcal{C} \subset \mathcal{A}^K$ of a Cartesian-product constellation $\mathcal{A}^K$. In general, a code $\mathcal{C}$ over $\mathcal{A}$ will have a lower bit rate (nominal spectral efficiency) ρ than $\mathcal{A}$, but a higher minimum squared distance $d^2_{\min}$. Via this tradeoff, we hope to achieve a "coding gain." Practically all of the codes that we will consider in later chapters will be of this type.

5.1.2 Minimum-distance decoding

Again, a decoding scheme is a method for mapping the received sequence into an estimate of the transmitted signal sequence. (Sometimes the decoder does more than this, but this definition will do for a start.)

If the encoding scheme is a block scheme, then it is plausible that the receiver should decode block-by-block as well. That there is no loss of optimality in block-by-block decoding can be shown from the theorem of irrelevance, or alternatively by an extension of the exercise involving Cartesian-product constellations at the end of this subsection.

We will now recapitulate how for block-by-block decoding, with equiprobable signals and iid Gaussian noise, the optimum decision rule is a minimum-distance (MD) rule.

For block-by-block decoding, the channel model is $\mathbf{Y} = \mathbf{X} + \mathbf{N}$, where all sequences are N-tuples. The transmitted sequence $\mathbf{X}$ is chosen equiprobably from the M N-tuples $\mathbf{a}_j$ in a signal constellation $\mathcal{A}$. The noise pdf is

$$p_N(\mathbf{n}) = \frac{1}{(2\pi\sigma^2)^{N/2}} e^{-||\mathbf{n}||^2/2\sigma^2},$$

where the symbol variance is $\sigma^2 = N_0/2$.

In digital communications, we are usually interested in the *minimum-probability-of-error* (MPE) decision rule: given a received vector $\mathbf{y}$, choose the signal point $\hat{\mathbf{a}} \in \mathcal{A}$ to minimize the probability of decision error $\Pr(E)$.

Since the probability that a decision $\hat{\mathbf{a}}$ is correct is simply the *a posteriori* probability $p(\hat{\mathbf{a}} \mid \mathbf{y})$, the MPE rule is equivalent to the *maximum-a-posteriori-probability* (MAP) rule: choose the $\hat{\mathbf{a}} \in \mathcal{A}$ such that $p(\hat{\mathbf{a}} \mid \mathbf{y})$ is maximum among all $p(\mathbf{a}_j \mid \mathbf{y}), \mathbf{a}_j \in \mathcal{A}$.

By Bayes' law,

$$p(\mathbf{a}_j \mid \mathbf{y}) = \frac{p(\mathbf{y} \mid \mathbf{a}_j)p(\mathbf{a}_j)}{p(\mathbf{y})}.$$

If the signals $\mathbf{a}_j$ are equiprobable, so $p(\mathbf{a}_j) = 1/M$ for all j, then the MAP rule is equivalent to the *maximum-likelihood* (ML) rule: choose the $\hat{\mathbf{a}} \in \mathcal{A}$ such that $p(\mathbf{y} \mid \hat{\mathbf{a}})$ is maximum among all $p(\mathbf{y} \mid \mathbf{a}_j), \mathbf{a}_j \in \mathcal{A}$.

Using the noise pdf, we can write

$$p(\mathbf{y} \mid \mathbf{a}_j) = p_N(\mathbf{y} - \mathbf{a}_j) = \frac{1}{(2\pi\sigma^2)^{N/2}} e^{-||\mathbf{y}-\mathbf{a}_j||^2/2\sigma^2}.$$

Therefore the ML rule is equivalent to the *minimum-distance* (MD) rule: choose the $\hat{\mathbf{a}} \in \mathcal{A}$ such that $||\mathbf{y} - \hat{\mathbf{a}}||^2$ is minimum among all $||\mathbf{y} - \mathbf{a}_j||^2, \mathbf{a}_j \in \mathcal{A}$.

In summary, under the assumption of equiprobable inputs and iid Gaussian noise, the MPE rule is the minimum-distance rule. Therefore from this point forward we consider only MD detection, which is easy to understand from a geometrical point of view.

Exercise 1 (Cartesian-product constellations, cont.).

(b) Show that if the signal constellation is a Cartesian product $\mathcal{A}^K$, then MD detection can be performed by performing independent MD detection on each of the K components of the received KN-tuple $\mathbf{y} = (\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_K)$. Using this result, sketch the decision regions of the (4×4)-QAM signal set of Figure 1(d).

(c) Show that if $\Pr(E)$ is the probability of error for MD detection of $\mathcal{A}$, then the probability of error for MD detection of $\mathcal{A}'$ is

$$\Pr(E)' = 1 - (1 - \Pr(E))^K,$$

Show that $\Pr(E)' \approx K \Pr(E)$ if $\Pr(E)$ is small. □

Example 1. The K-fold Cartesian product $\mathcal{A}' = \mathcal{A}^K$ of a 2-PAM signal set $\mathcal{A} = \{\pm\alpha\}$ corresponds to independent transmission of K bits using 2-PAM. Geometrically, $\mathcal{A}'$ is the vertex set of a K-cube of side 2α. For example, for $K = 2$, $\mathcal{A}'$ is the (2×2)-QAM constellation of Figure 1(b).

From Exercise 1(a), the K-cube constellation $\mathcal{A}' = \mathcal{A}^K$ has dimension $N' = K$, size $M' = 2^K$, bit rate (nominal spectral efficiency) $\rho = 2$ b/2D, average energy $E(\mathcal{A}') = K\alpha^2$, average energy per bit $E_b = \alpha^2$, minimum squared distance $d^2_{\min}(\mathcal{A}') = 4\alpha^2$, and average number of nearest neighbors $K'_{\min}(\mathcal{A}') = K$. From Exercise 1(c), its probability of error is approximately K times the single-bit error probability:

$$\Pr(E)' \approx KQ\left(\sqrt{2E_b/N_0}\right).$$

Consequently, if we define the probability of error per bit as $P_b(E) = \Pr(E)'/K$, then we obtain the curve of (4.3) for all K-cube constellations:

$$P_b(E) \approx Q\left(\sqrt{2E_b/N_0}\right),$$

including the (2×2)-QAM constellation of Figure 1(b).

A code over the 2-PAM signal set $\mathcal{A}$ is thus simply a subset of the vertices of a K-cube. □

5.1.3 Decision regions

Under a minimum-distance (MD) decision rule, real N-space $\mathbb{R}^N$ is partitioned into M *decision regions* $\mathcal{R}_j, 1 \le j \le M$, where $\mathcal{R}_j$ consists of the received vectors $\mathbf{y} \in \mathbb{R}^N$ that are at least as close to $\mathbf{a}_j$ as to any other point in $\mathcal{A}$:

$$\mathcal{R}_j = \{\mathbf{y} \in \mathbb{R}^N : ||\mathbf{y} - \mathbf{a}_j||^2 \le ||\mathbf{y} - \mathbf{a}_{j'}||^2 \text{ for all } j' \ne j\}. \tag{5.1}$$

The minimum-distance regions $\mathcal{R}_j$ are also called *Voronoi regions*. Under the MD rule, given a received sequence $\mathbf{y}$, the decision is $\mathbf{a}_j$ only if $\mathbf{y} \in \mathcal{R}_j$. The decision regions $\mathcal{R}_j$ cover all of N-space $\mathbb{R}^N$, and are disjoint except on their boundaries.

Since the noise vector $\mathbf{N}$ is a continuous random vector, the probability that $\mathbf{y}$ will actually fall precisely on the boundary of $\mathcal{R}_j$ is zero, so in that case it does not matter which decision is made.

The decision region $\mathcal{R}_j$ is the intersection of the $M-1$ pairwise decision regions $\mathcal{R}_{jj'}$ defined by

$$\mathcal{R}_{jj'} = \{\mathbf{y} \in \mathbb{R}^N : ||\mathbf{y} - \mathbf{a}_j||^2 \le ||\mathbf{y} - \mathbf{a}_{j'}||^2\}.$$

Geometrically, it is obvious that $\mathcal{R}_{jj'}$ is the half-space containing $\mathbf{a}_j$ that is bounded by the perpendicular bisector hyperplane $\mathcal{H}_{jj'}$ between $\mathbf{a}_j$ and $\mathbf{a}_{j'}$, as shown in Figure 2.

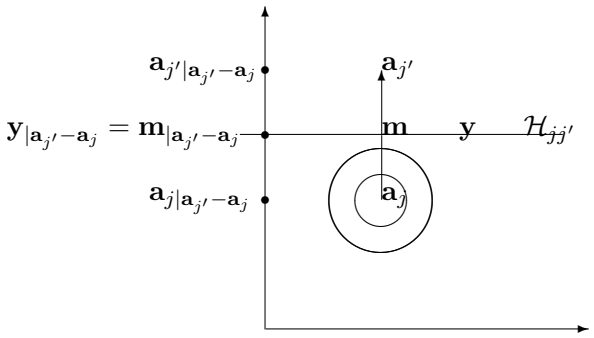


Figure 2. The boundary hyperplane $\mathcal{H}_{jj'}$ is the perpendicular bisector between $\mathbf{a}_j$ and $\mathbf{a}_{j'}$.

Algebraically, since $\mathcal{H}_{jj'}$ is the set of points in $\mathbb{R}^N$ that are equidistant from $\mathbf{a}_j$ and $\mathbf{a}_{j'}$, it is characterized by the following equivalent equations:

$$\begin{aligned} ||\mathbf{y} - \mathbf{a}_j||^2 &= ||\mathbf{y} - \mathbf{a}_{j'}||^2; \\ -2\langle \mathbf{y}, \mathbf{a}_j \rangle + ||\mathbf{a}_j||^2 &= -2\langle \mathbf{y}, \mathbf{a}_{j'} \rangle + ||\mathbf{a}_{j'}||^2; \\ \langle \mathbf{y}, \mathbf{a}_{j'} - \mathbf{a}_j \rangle &= \langle \frac{\mathbf{a}_j + \mathbf{a}_{j'}}{2}, \mathbf{a}_{j'} - \mathbf{a}_j \rangle = \langle \mathbf{m}, \mathbf{a}_{j'} - \mathbf{a}_j \rangle. \end{aligned} \tag{5.2}$$

where $\mathbf{m}$ denotes the midvector $\mathbf{m} = (\mathbf{a}_j + \mathbf{a}_{j'})/2$. If the difference vector between $\mathbf{a}_{j'}$ and $\mathbf{a}_j$ is $\mathbf{a}_{j'} - \mathbf{a}_j$ and

$$\boldsymbol{\phi}_{j \to j'} = \frac{\mathbf{a}_{j'} - \mathbf{a}_j}{||\mathbf{a}_{j'} - \mathbf{a}_j||}$$

is the normalized difference vector, so that $||\boldsymbol{\phi}_{j \to j'}||^2 = 1$, then the projection of any vector $\mathbf{x}$ onto the difference vector $\mathbf{a}_{j'} - \mathbf{a}_j$ is

$$\mathbf{x}_{|\mathbf{a}_{j'} - \mathbf{a}_j} = \langle \mathbf{x}, \boldsymbol{\phi}_{j \to j'} \rangle \boldsymbol{\phi}_{j \to j'} = \frac{\langle \mathbf{x}, \mathbf{a}_{j'} - \mathbf{a}_j \rangle}{||\mathbf{a}_{j'} - \mathbf{a}_j||^2} (\mathbf{a}_{j'} - \mathbf{a}_j).$$

The geometric meaning of Equation (5.2) is thus that $\mathbf{y} \in \mathcal{H}_{jj'}$ if and only if the projection $\mathbf{y}_{|\mathbf{a}_{j'}-\mathbf{a}_j}$ of $\mathbf{y}$ onto the difference vector $\mathbf{a}_{j'}-\mathbf{a}_j$ is equal to the projection $\mathbf{m}_{|\mathbf{a}_{j'}-\mathbf{a}_j}$ of the midvector $\mathbf{m} = (\mathbf{a}_j + \mathbf{a}_{j'})/2$ onto the difference vector $\mathbf{a}_j - \mathbf{a}_{j'}$, as illustrated in Figure 2.

The decision region $\mathcal{R}_j$ is the intersection of these $M-1$ half-spaces:

$$\mathcal{R}_j = \bigcap_{j' \neq j} \mathcal{R}_{jj'}.$$

(Equivalently, the complementary region $\overline{\mathcal{R}_j}$ is the union of the complementary half-spaces $\overline{\mathcal{R}_{jj'}}$.) A decision region $\mathcal{R}_j$ is therefore a convex polytope bounded by portions of a subset $\{\mathcal{H}_{jj'}, \mathbf{a}_{j'} \in \mathcal{N}(\mathbf{a}_j)\}$ of the boundary hyperplanes $\mathcal{H}_{jj'}$, where the subset $\mathcal{N}(\mathbf{a}_j) \subseteq \mathcal{A}$ of neighbors of $\mathbf{a}_j$ that contribute boundary faces to this polytope is called the *relevant subset*. It is easy to see that the relevant subset must always include the nearest neighbors to $\mathbf{a}_j$.

5.2 Probability of decision error

The probability of decision error given that $\mathbf{a}_j$ is transmitted is the probability that $\mathbf{Y} = \mathbf{a}_j + \mathbf{N}$ falls outside the decision region $\mathcal{R}_j$, whose "center" is $\mathbf{a}_j$. Equivalently, it is the probability that the noise variable $\mathbf{N}$ falls outside the translated region $\mathcal{R}_j - \mathbf{a}_j$, whose "center" is $\mathbf{0}$:

$$\Pr(E \mid \mathbf{a}_j) = 1 - \int_{\mathcal{R}_j} p_Y(\mathbf{y} \mid \mathbf{a}_j)\, d\mathbf{y} = 1 - \int_{\mathcal{R}_j} p_N(\mathbf{y} - \mathbf{a}_j)\, d\mathbf{y} = 1 - \int_{\mathcal{R}_j - \mathbf{a}_j} p_N(\mathbf{n})\, d\mathbf{n}.$$

Exercise 2 (error probability invariance). (a) Show that the probabilities of error $\Pr(E \mid \mathbf{a}_j)$ are unchanged if $\mathcal{A}$ is translated by any vector $\mathbf{v}$; *i.e.,* the constellation $\mathcal{A}' = \mathcal{A} + \mathbf{v}$ has the same error probability $\Pr(E)$ as $\mathcal{A}$.

(b) Show that $\Pr(E)$ is invariant under orthogonal transformations; *i.e.,* the constellation $\mathcal{A}' = U\mathcal{A}$ has the same $\Pr(E)$ as $\mathcal{A}$ when U is any orthogonal $N \times N$ matrix (*i.e.,* $U^{-1} = U^T$).

(c) Show that $\Pr(E)$ is unchanged if both the constellation $\mathcal{A}$ and the noise $\mathbf{N}$ are scaled by the same scale factor $\alpha > 0$. □

Exercise 3 (optimality of zero-mean constellations). Consider an arbitrary signal set $\mathcal{A} = \{\mathbf{a}_j, 1 \leq j \leq M\}$. Assume that all signals are equiprobable. Let $\mathbf{m}(\mathcal{A}) = \frac{1}{M}\sum_j \mathbf{a}_j$ be the average signal, and let $\mathcal{A}'$ be $\mathcal{A}$ translated by $\mathbf{m}(\mathcal{A})$ so that the mean of $\mathcal{A}'$ is zero:

$$\mathcal{A}' = \mathcal{A} - \mathbf{m}(\mathcal{A}) = \{\mathbf{a}_j - \mathbf{m}(\mathcal{A}), 1 \leq j \leq M\}.$$

Let $E(\mathcal{A})$ and $E(\mathcal{A}')$ denote the average energies of $\mathcal{A}$ and $\mathcal{A}'$, respectively.

(a) Show that the error probability of an optimum detector is the same for $\mathcal{A}'$ as it is for $\mathcal{A}$.

(b) Show that $E(\mathcal{A}') = E(\mathcal{A}) - ||\mathbf{m}(\mathcal{A})||^2$. Conclude that removing the mean $\mathbf{m}(\mathcal{A})$ is always a good idea.

(c) Show that a binary antipodal signal set $\mathcal{A} = \{\pm\mathbf{a}\}$ is always optimal for $M = 2$. □

In general, there is no closed-form expression for the Gaussian integral $\Pr(E \mid \mathbf{a}_j)$. However, we can obtain an upper bound in terms of pairwise error probabilities, called the union bound, which is usually quite sharp. The first term of the union bound, called the union bound estimate, is usually an excellent approximation, and will be the basis for our analysis of coding gains of small-to-moderate-sized constellations. A lower bound with the same exponential behavior may be obtained by considering only the worst-case pairwise error probability.

5.2.1 Pairwise error probabilities

We now show that each pairwise probability has a simple closed-form expression that depends only on the squared distance $d^2(\mathbf{a}_j, \mathbf{a}_{j'}) = ||\mathbf{a}_j - \mathbf{a}_{j'}||^2$ and the noise variance $\sigma^2 = N_0/2$.

From Figure 2, it is clear that whether $\mathbf{y} = \mathbf{a}_j + \mathbf{n}$ is closer to $\mathbf{a}_{j'}$ than to $\mathbf{a}_j$ depends only on the projection $\mathbf{y}_{|\mathbf{a}_{j'} - \mathbf{a}_j}$ of $\mathbf{y}$ onto the difference vector $\mathbf{a}_{j'} - \mathbf{a}_j$. In fact, from (5.2), an error can occur if and only if

$$|\mathbf{n}_{|\mathbf{a}_{j'} - \mathbf{a}_j}| = |\langle \mathbf{n}, \boldsymbol{\phi}_{j \to j'} \rangle| = \frac{|\langle \mathbf{n}, \mathbf{a}_{j'} - \mathbf{a}_j \rangle|}{||\mathbf{a}_{j'} - \mathbf{a}_j||} \quad \geq \quad \frac{\langle \mathbf{a}_{j'} - \mathbf{a}_j, \mathbf{a}_{j'} - \mathbf{a}_j \rangle}{2||\mathbf{a}_{j'} - \mathbf{a}_j||} = \frac{||\mathbf{a}_{j'} - \mathbf{a}_j||}{2}.$$

In other words, an error can occur if and only if the magnitude of the one-dimensional noise component $n_1 = \mathbf{n}_{|\mathbf{a}_{j'} - \mathbf{a}_j}$, the projection of $\mathbf{n}$ onto the difference vector $\mathbf{a}_{j'} - \mathbf{a}_j$, exceeds half the distance $d(\mathbf{a}_{j'}, \mathbf{a}_j) = ||\mathbf{a}_{j'} - \mathbf{a}_j||$ between $\mathbf{a}_{j'}$ and $\mathbf{a}_j$.

We now use the fact that the distribution $p_N(\mathbf{n})$ of the iid Gaussian noise vector $\mathbf{N}$ is spherically symmetric, so the pdf of any one-dimensional projection such as n_1 is

$$p_N(n_1) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-n_1^2/2\sigma^2}.$$

In other words, $\mathbf{N}$ is an iid zero-mean Gaussian vector with variance σ^2 in any coordinate system, including a coordinate system in which the first coordinate axis is aligned with the vector $\mathbf{a}_{j'} - \mathbf{a}_j$.

Consequently, the pairwise error probability $\Pr\{\mathbf{a}_j \to \mathbf{a}_{j'}\}$ that if $\mathbf{a}_j$ is transmitted, the received vector $\mathbf{y} = \mathbf{a}_j + \mathbf{n}$ will be at least as close to $\mathbf{a}_{j'}$ as to $\mathbf{a}_j$ is given simply by

$$\Pr\{\mathbf{a}_j \to \mathbf{a}_{j'}\} = \frac{1}{\sqrt{2\pi\sigma^2}} \int_{d(\mathbf{a}_{j'}, \mathbf{a}_j)/2}^{\infty} e^{-x^2/2\sigma^2} dx = Q\left(\frac{d(\mathbf{a}_{j'}, \mathbf{a}_j)}{2\sigma}\right), \tag{5.3}$$

where $Q(\cdot)$ is again the Gaussian probability of error function.

As we have seen, the probability of error for a 2-PAM signal set $\{\pm\alpha\}$ is $Q(\alpha/\sigma)$. Since the distance between the two signals is $d = 2\alpha$, this is just a special case of this general formula.

In summary, the spherical symmetry of iid Gaussian noise leads to the remarkable result that the pairwise error probability from $\mathbf{a}_j$ to $\mathbf{a}_{j'}$ depends only on the squared distance $d^2(\mathbf{a}_{j'}, \mathbf{a}_j) = ||\mathbf{a}_{j'} - \mathbf{a}_j||^2$ between $\mathbf{a}_j$ and $\mathbf{a}_{j'}$ and the noise variance σ^2.

Exercise 4 (non-equiprobable signals).

Let $\mathbf{a}_j$ and $\mathbf{a}_{j'}$ be two signals that are not equiprobable. Find the optimum (MPE) pairwise decision rule and pairwise error probability $\Pr\{\mathbf{a}_j \to \mathbf{a}_{j'}\}$. □

5.2.2 The union bound and the UBE

The *union bound* on error probability is based on the elementary union bound of probability theory: if A and B are any two events, then $\Pr(A \cup B) \leq \Pr(A) + \Pr(B)$. Thus the probability of detection error $\Pr(E \mid \mathbf{a}_j)$ with minimum-distance detection if $\mathbf{a}_j$ is sent— *i.e.,* the probability that $\mathbf{y}$ will be closer to some other $\mathbf{a}_{j'} \in \mathcal{A}$ than to $\mathbf{a}_j$– is upperbounded by the sum of the pairwise error probabilities to all other signals $\mathbf{a}_{j'} \neq \mathbf{a}_j \in \mathcal{A}$:

$$\Pr(E \mid \mathbf{a}_j) \leq \sum_{\mathbf{a}_j \neq \mathbf{a}_{j'} \in \mathcal{A}} \Pr\{\mathbf{a}_j \to \mathbf{a}_{j'}\} = \sum_{\mathbf{a}_j \neq \mathbf{a}_{j'} \in \mathcal{A}} Q\left(\frac{d(\mathbf{a}_j, \mathbf{a}_{j'})}{2\sigma}\right).$$

Let D denote the set of distances between signal points in $\mathcal{A}$; then we can write the union bound as

$$\Pr(E \mid \mathbf{a}_j) \leq \sum_{d \in D} K_d(\mathbf{a}_j) Q\left(\frac{d}{2\sigma}\right), \tag{5.4}$$

where $K_d(\mathbf{a}_j)$ is the number of signals $\mathbf{a}_{j'} \neq \mathbf{a}_j \in \mathcal{A}$ at distance d from $\mathbf{a}_j$. Because $Q(x)$ decreases exponentially as $e^{-x^2/2}$ (see Appendix), the factor $Q(d/2\sigma)$ will be largest for the minimum Euclidean distance

$$d_{\min}(\mathcal{A}) = \min_{\mathbf{a}_j \neq \mathbf{a}_{j'} \in \mathcal{A}} ||\mathbf{a}_{j'} - \mathbf{a}_j||,$$

and will decrease rapidly for larger distances.

The *union bound estimate* (UBE) of $\Pr(E \mid \mathbf{a}_j)$ is based on the idea that the nearest neighbors to $\mathbf{a}_j$ at distance $d_{\min}(\mathcal{A})$ (if there are any) will dominate this sum. If there are $K_{\min}(\mathbf{a}_j)$ neighbors at distance $d_{\min}(\mathcal{A})$ from $\mathbf{a}_j$, then

$$\Pr(E \mid \mathbf{a}_j) \approx K_{\min}(\mathbf{a}_j)\ Q\left(\frac{d_{\min}(\mathcal{A})}{2\sigma}\right). \tag{5.5}$$

Of course this estimate is valid only if the next nearest neighbors are at a significantly greater distance and there are not too many of them; if these assumptions are violated, then further terms should be used in the estimate.

The union bound may be somewhat sharpened by considering only signals in the relevant subset $\mathcal{N}(\mathbf{a}_j)$ that determine faces of the decision region $\mathcal{R}_j$. However, since $\mathcal{N}(\mathbf{a}_j)$ includes all nearest neighbors at distance $d_{\min}(\mathcal{A})$, this will not affect the UBE.

Finally, if there is at least one neighbor $\mathbf{a}_{j'}$ at distance $d_{\min}(\mathcal{A})$ from $\mathbf{a}_j$, then we have the *pairwise lower bound*

$$\Pr(E \mid \mathbf{a}_j) \geq \Pr\{\mathbf{a}_j \to \mathbf{a}_{j'}\} = Q\left(\frac{d_{\min}(\mathcal{A})}{2\sigma}\right), \tag{5.6}$$

since there must be a detection error if $\mathbf{y}$ is closer to $\mathbf{a}_{j'}$ than to $\mathbf{a}_j$. Thus we are usually able to obtain upper and lower bounds on $\Pr(E \mid \mathbf{a}_j)$ that have the same "exponent" (argument of the $Q(\cdot)$ function) and that differ only by a small factor of the order of $K_{\min}(\mathbf{a}_j)$.

We can obtain similar upper and lower bounds and estimates for the total error probability

$$\Pr(E) = \overline{\Pr(E \mid \mathbf{a}_j)},$$

where the overbar denotes the expectation over the equiprobable ensemble of signals in $\mathcal{A}$. For example, if $K_{\min}(\mathcal{A}) = \overline{K_{\min}(\mathbf{a}_j)}$ is the average number of nearest neighbors at distance $d_{\min}(\mathcal{A})$, then the union bound estimate of $\Pr(E)$ is

$$\Pr(E) \approx K_{\min}(\mathcal{A}) Q\left(\frac{d_{\min}(\mathcal{A})}{2\sigma}\right). \tag{5.7}$$

Exercise 5 (UBE for M-PAM constellations). For an M-PAM constellation $\mathcal{A}$, show that $K_{\min}(\mathcal{A}) = 2(M-1)/M$. Conclude that the union bound estimate of $\Pr(E)$ is

$$\Pr(E) \approx 2\left(\frac{M-1}{M}\right) Q\left(\frac{d}{2\sigma}\right).$$

Show that in this case the union bound estimate is exact. Explain why. □

5.3 Performance analysis in the power-limited regime

Recall that the power-limited regime is defined as the domain in which the nominal spectral efficiency ρ is not greater than 2 b/2D. In this regime we normalize all quantities "per bit," and generally use E_b/N_0 as our normalized measure of signal-to-noise ratio.

The baseline uncoded signal set in this regime is the one-dimensional 2-PAM signal set $\mathcal{A} = \{\pm\alpha\}$, or equivalently a K-cube constellation $\mathcal{A}^K$. Such a constellation has bit rate (nominal spectral efficiency) $\rho = 2$ b/2D, average energy ber bit $E_b = \alpha^2$, minimum squared distance $d^2_{\min}(\mathcal{A}) = 4\alpha^2$, and average number of nearest neighbors per bit $K_b(\mathcal{A}) = 1$. By the UBE (5.7), its error probability per bit is given by

$$P_b(E) \approx Q^{\surd}(2E_b/N_0), \tag{5.8}$$

where we now use the "Q-of-the-square-root-of" function $Q^{\surd}$, defined by $Q^{\surd}(x) = Q(\sqrt{x})$ (see Appendix). This baseline curve of $P_b(E)$ *vs.* E_b/N_0 is plotted in Chapter 4, Figure 1.

The *effective coding gain* $\gamma_{\rm eff}(\mathcal{A})$ of a signal set $\mathcal{A}$ at a given target error probability per bit $P_b(E)$ will be defined as the difference in dB between the E_b/N_0 required to achieve the target $P_b(E)$ with $\mathcal{A}$ and the E_b/N_0 required to achieve the target $P_b(E)$ with 2-PAM (*i.e.,* no coding).

For example, we have seen that the maximum possible effective coding gain at $P_b(E) \approx 10^{-5}$ is approximately 11.2 dB. For lower $P_b(E)$, the maximum possible gain is higher, and for higher $P_b(E)$, the maximum possible gain is lower.

In this definition, the effective coding gain includes any gains that result from using a lower nominal spectral efficiency $\rho < 2$ b/2D, which as we have seen can range up to 3.35 dB. If ρ is held constant at $\rho = 2$ b/2D, then the maximum possible effective coding gain is lower; *e.g.,* at $P_b(E) \approx 10^{-5}$ it is approximately 8 dB. If there is a constraint on ρ (bandwidth), then it is better to plot $P_b(E)$ *vs.* $\mathrm{SNR}_{\rm norm}$, especially to measure how far $\mathcal{A}$ is from achieving capacity.

The UBE allows us to estimate the effective coding gain as follows. The probability of error per bit (not in general the same as the bit error probability!) is

$$P_b(E) = \frac{\Pr(E)}{\log_2|\mathcal{A}|} \approx \frac{K_{\min}(\mathcal{A})}{\log_2|\mathcal{A}|} Q^{\surd}\left(\frac{d^2_{\min}(\mathcal{A})}{2N_0}\right),$$

since $Q^{\surd}\left(d^2_{\min}(\mathcal{A})/2N_0\right) = Q(d_{\min}(\mathcal{A})/2\sigma)$. In the power-limited regime, we define the *nominal coding gain* $\gamma_{\rm c}(\mathcal{A})$ as

$$\gamma_{\rm c}(\mathcal{A}) = \frac{d^2_{\min}(\mathcal{A})}{4E_b}. \tag{5.9}$$

This definition is normalized so that for 2-PAM, $\gamma_{\rm c}(\mathcal{A}) = 1$. Because nominal coding gain is a multiplicative factor in the argument of the $Q^{\surd}(\cdot)$ function, it is often measured in dB. The UBE then becomes

$$P_b(E) \approx K_b(\mathcal{A}) Q^{\surd}(2\gamma_{\rm c}(\mathcal{A})E_b/N_0), \tag{5.10}$$

where $K_b(\mathcal{A}) = K_{\min}(\mathcal{A})/\log_2|\mathcal{A}|$ is the average number of nearest neighbors per transmitted bit. Note that for 2-PAM, this expression is exact.

Given $\gamma_{\rm c}(\mathcal{A})$ and $K_b(\mathcal{A})$, we may obtain a plot of the UBE (5.10) simply by moving the baseline curve (Figure 1 of Chapter 4) to the left by $\gamma_{\rm c}(\mathcal{A})$ (in dB), and then up by a factor of $K_b(\mathcal{A})$, since $P_b(E)$ is plotted on a log scale. (This is an excellent reason why error probability curves are always plotted on a log-log scale, with SNR measured in dB.)

Thus if $K_b(\mathcal{A}) = 1$, then the effective coding gain $\gamma_{\rm eff}(\mathcal{A})$ is equal to the nominal coding gain $\gamma_{\rm c}(\mathcal{A})$ for all $P_b(E)$, to the accuracy of the UBE. However, if $K_b(\mathcal{A}) > 1$, then the effective coding gain is less than the nominal coding gain by an amount which depends on the steepness of the $P_b(E)$ *vs.* E_b/N_0 curve at the target $P_b(E)$. At $P_b(E) \approx 10^{-5}$, a rule of thumb which is fairly accurate if $K_b(\mathcal{A})$ is not too large is that an increase of a factor of two in $K_b(\mathcal{A})$ costs about 0.2 dB in effective coding gain; *i.e.,*

$$\gamma_{\rm eff}(\mathcal{A}) \approx \gamma_{\rm c}(\mathcal{A}) - (0.2)(\log_2 K_b(\mathcal{A})) \quad \text{(in dB)}. \tag{5.11}$$

A more accurate estimate may be obtained by a plot of the union bound estimate (5.10).

Exercise 6 (invariance of coding gain). Show that the nominal coding gain $\gamma_{\rm c}(\mathcal{A})$ of (5.9), the UBE (5.10) of $P_b(E)$, and the effective coding gain $\gamma_{\rm eff}(\mathcal{A})$ are invariant to scaling, orthogonal transformations and Cartesian products. □

5.4 Orthogonal and related signal sets

Orthogonal, simplex and biorthogonal signal sets are concrete examples of large signal sets that are suitable for the power-limited regime when bandwidth is truly unconstrained. Orthogonal signal sets are the easiest to describe and analyze. Simplex signal sets are believed to be optimal for a given constellation size M when there is no constraint on dimension. Biorthogonal signal sets are slightly more bandwidth-efficient. For large M, all become essentially equivalent.

The following exercises develop the parameters of these signal sets, and show that they can achieve reliable transmission for E_b/N_0 within 3 dB from the ultimate Shannon limit.[1] The drawback of these signal sets is that the number of dimensions (bandwidth) becomes very large and the spectral efficiency ρ very small as $M \to \infty$. Also, even with the "fast" Walsh-Hadamard transform (see Chapter 1, Problem 2), decoding complexity is of the order of $M \log_2 M$, which increases exponentially with the number of bits transmitted, $\log_2 M$, and thus is actually "slow."

Exercise 7 (Orthogonal signal sets). An *orthogonal signal set* is a set $\mathcal{A} = \{\mathbf{a}_j, 1 \le j \le M\}$ of M orthogonal vectors in $\mathbb{R}^M$ with equal energy $E(\mathcal{A})$; *i.e.,* $\langle \mathbf{a}_j, \mathbf{a}_{j'} \rangle = E(\mathcal{A})\delta_{jj'}$ (Kronecker delta).

(a) Compute the nominal spectral efficiency ρ of $\mathcal{A}$ in bits per two dimensions. Compute the average energy E_b per information bit.

(b) Compute the minimum squared distance $d^2_{\rm min}(\mathcal{A})$. Show that every signal has $K_{\rm min}(\mathcal{A}) = M - 1$ nearest neighbors.

(c) Let the noise variance be $\sigma^2 = N_0/2$ per dimension. Show that the probability of error of an optimum detector is bounded by the UBE

$$\Pr(E) \le (M-1)Q^{\surd}(E(\mathcal{A})/N_0).$$

(d) Let $M \to \infty$ with E_b held constant. Using an asymptotically accurate upper bound for the $Q^{\surd}(\cdot)$ function (see Appendix), show that $\Pr(E) \to 0$ provided that $E_b/N_0 > 2\ln 2$ (1.42 dB). How close is this to the ultimate Shannon limit on E_b/N_0? What is the nominal spectral efficiency ρ in the limit? □

[1] Actually, it can be shown that with optimum detection orthogonal signal sets can approach the ultimate Shannon limit on E_b/N_0 as $M \to \infty$; however, the union bound is too weak to prove this.

Exercise 8 (Simplex signal sets). Let $\mathcal{A}$ be an orthogonal signal set as above.

(a) Denote the mean of $\mathcal{A}$ by $\mathbf{m}(\mathcal{A})$. Show that $\mathbf{m}(\mathcal{A}) \neq \mathbf{0}$, and compute $||\mathbf{m}(\mathcal{A})||^2$.

The zero-mean set $\mathcal{A}' = \mathcal{A} - \mathbf{m}(\mathcal{A})$ (as in Exercise 2) is called a *simplex signal set*. It is universally believed to be the optimum set of M signals in AWGN in the absence of bandwidth constraints, except at ridiculously low SNRs.

(b) For $M = 2, 3, 4$, sketch $\mathcal{A}$ and $\mathcal{A}'$.

(c) Show that all signals in $\mathcal{A}'$ have the same energy $E(\mathcal{A}')$. Compute $E(\mathcal{A}')$. Compute the inner products $\langle \mathbf{a}_j, \mathbf{a}_{j'} \rangle$ for all $\mathbf{a}_j, \mathbf{a}_{j'} \in \mathcal{A}'$.

(d) [Optional]. Show that for ridiculously low SNRs, a signal set consisting of $M - 2$ zero signals and two antipodal signals $\{\pm\mathbf{a}\}$ has a lower $\Pr(E)$ than a simplex signal set. [Hint: see M. Steiner, "The strong simplex conjecture is false," IEEE TRANSACTIONS ON INFORMATION THEORY, pp. 721-731, May 1994.] □

Exercise 9 (Biorthogonal signal sets). The set $\mathcal{A}'' = \pm\mathcal{A}$ of size $2M$ consisting of the M signals in an orthogonal signal set $\mathcal{A}$ with symbol energy $E(\mathcal{A})$ and their negatives is called a *biorthogonal signal set*.

(a) Show that the mean of $\mathcal{A}''$ is $\mathbf{m}(\mathcal{A}'') = \mathbf{0}$, and that the average energy per symbol is $E(\mathcal{A})$.

(b) How much greater is the nominal spectral efficiency ρ of $\mathcal{A}''$ than that of $\mathcal{A}$, in bits per two dimensions?

(c) Show that the probability of error of $\mathcal{A}''$ is approximately the same as that of an orthogonal signal set with the same size and average energy, for M large.

(d) Let the number of signals be a power of 2: $2M = 2^k$. Show that the nominal spectral efficiency is $\rho(\mathcal{A}'') = 4k2^{-k}$ b/2D, and that the nominal coding gain is $\gamma_c(\mathcal{A}'') = k/2$. Show that the number of nearest neighbors is $K_{\min}(\mathcal{A}'') = 2^k - 2$. □

Example 2 (Biorthogonal signal sets). Using Exercise 9, we can estimate the effective coding gain of a biorthogonal signal set using our rule of thumb (5.11), and check its accuracy against a plot of the UBE (5.10).

The $2^k = 16$ biorthogonal signal set $\mathcal{A}$ has dimension $N = 2^{k-1} = 8$, rate $k = 4$ b/sym, and nominal spectral efficiency $\rho(\mathcal{A}) = 1$ b/2D. With energy $E(\mathcal{A})$ per symbol, it has $E_b = E(\mathcal{A})/4$ and $d^2_{\min}(\mathcal{A}) = 2E(\mathcal{A})$, so its nominal coding gain is

$$\gamma_c(\mathcal{A}) = d^2_{\min}(\mathcal{A})/4E_b = 2 \text{ (3.01 dB)},$$

The number of nearest neighbors is $K_{\min}(\mathcal{A}) = 2^k - 2 = 14$, so $K_b(\mathcal{A}) = 14/4 = 3.5$, and the estimate of its effective coding gain at $P_b(E) \approx 10^{-5}$ by our rule of thumb (5.11) is thus

$$\gamma_{\text{eff}}(\mathcal{A}) \approx 3 - 2(0.2) = 2.6 \text{ dB}.$$

A more accurate plot of the UBE (5.10) may be obtained by shifting the baseline curve (Figure 1 of Chapter 4) left by 3 dB and up by half a vertical unit (since $3.5 \approx \sqrt{10}$), as shown in Figure 3. This plot shows that the rough estimate $\gamma_{\text{eff}}(\mathcal{A}) \approx 2.6$ dB is quite accurate at $P_b(E) \approx 10^{-5}$.

Similarly, the 64-biorthogonal signal set $\mathcal{A}'$ has nominal coding gain $\gamma_c(\mathcal{A}') = 3$ (4.77 dB), $K_b(\mathcal{A}') = 62/6 \approx 10$, and effective coding gain $\gamma_{\text{eff}}(\mathcal{A}') \approx 4.8$ - $3.5(0.2) = 4.1$ dB by our rule of thumb. The 256-biorthogonal signal set $\mathcal{A}''$ has nominal coding gain $\gamma_c(\mathcal{A}'') = 4$ (6.02 dB), $K_b(\mathcal{A}'') = 254/8 \approx 32$, and effective coding gain $\gamma_{\text{eff}}(\mathcal{A}'') \approx 6$ - $5(0.2) = 5.0$ dB by our rule of thumb. Figure 3 also shows plots of the UBE (5.10) for these two signal constellations, which show that our rule of thumb continues to be fairly accurate.

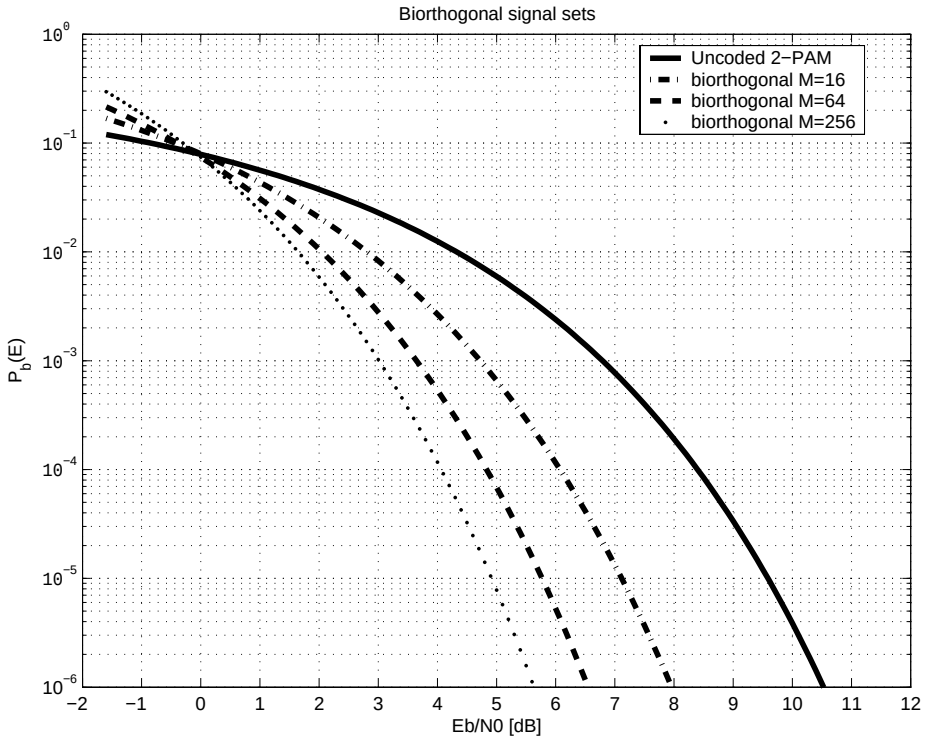


Figure 3. $P_b(E)$ *vs.* E_b/N_0 for biorthogonal signal sets with $2^k = 16, 64$ and 256.

5.5 Performance in the bandwidth-limited regime

Recall that the bandwidth-limited regime is defined as the domain in which the nominal spectral efficiency ρ is greater than 2 b/2D; *i.e.,* the domain of nonbinary signaling. In this regime we normalize all quantities "per two dimensions," and use $\mathrm{SNR}_{\mathrm{norm}}$ as our normalized measure of signal-to-noise ratio.

The baseline uncoded signal set in this regime is the M-PAM signal set $\mathcal{A} = \alpha\{\pm 1, \pm 3, \ldots, \pm(M-1)\}$, or equivalently the $(M \times M)$-QAM constellation $\mathcal{A}^2$. Typically M is a power of 2. Such a constellation has bit rate (nominal spectral efficiency) $\rho = 2\log_2 M$ b/2D and minimum squared distance $d^2_{\min}(\mathcal{A}^2) = 4\alpha^2$. As shown in Chapter 4, its average energy per two dimensions is

$$E_s = \frac{2\alpha^2(M^2-1)}{3} = \frac{d^2_{\min}(\mathcal{A})(2^\rho - 1)}{6}. \tag{5.12}$$

The average number of nearest neighbors per two dimensions is twice that of M-PAM, namely $K_s(\mathcal{A}) = 4(M-1)/M$, which rapidly approaches $K_s(\mathcal{A}) \approx 4$ as M becomes large. By the UBE (5.7), the error probability per two dimensions is given by

$$P_s(E) \approx 4Q\sqrt{(3\mathrm{SNR}_{\mathrm{norm}})}. \tag{5.13}$$

This baseline curve of $P_s(E)$ *vs.* $\mathrm{SNR}_{\mathrm{norm}}$ was plotted in Figure 2 of Chapter 4.

In the bandwidth-limited regime, the *effective coding gain* $\gamma_{\rm eff}(\mathcal{A})$ of a signal set $\mathcal{A}$ at a given target error rate $P_s(E)$ will be defined as the difference in dB between the $\mathrm{SNR}_{\rm norm}$ required to achieve the target $P_s(E)$ with $\mathcal{A}$ and the $\mathrm{SNR}_{\rm norm}$ required to achieve the target $P_s(E)$ with M-PAM or $(M \times M)$-QAM (no coding).

For example, we saw from Figure 2 of Chapter 4 that the maximum possible effective coding gain at $P_s(E) \approx 10^{-5}$ is approximately 8.4 dB, which is about 3 dB less than in the power-limited regime (due solely to the fact that the bandwidth is fixed).

The effective coding gain is again estimated by the UBE, as follows. The probability of error per two dimensions is

$$P_s(E) = \frac{2\,\Pr(E)}{N} \approx \frac{2K_{\min}(\mathcal{A})}{N} Q\sqrt{\left(\frac{d^2_{\min}(\mathcal{A})}{2N_0}\right)}.$$

In the bandwidth-limited regime, we define the *nominal coding gain* $\gamma_{\rm c}(\mathcal{A})$ as

$$\gamma_{\rm c}(\mathcal{A}) = \frac{(2^\rho - 1)d^2_{\min}(\mathcal{A})}{6E_s}. \tag{5.14}$$

This definition is normalized so that for M-PAM or $(M \times M)$-QAM, $\gamma_{\rm c}(\mathcal{A}) = 1$. Again, $\gamma_{\rm c}(\mathcal{A})$ is often measured in dB. The UBE (5.10) then becomes

$$P_s(E) \approx K_s(\mathcal{A}) Q\sqrt{(3\gamma_{\rm c}(\mathcal{A})\mathrm{SNR}_{\rm norm})}, \tag{5.15}$$

where $K_s(\mathcal{A}) = 2K_{\min}(\mathcal{A})/N$ is the average number of nearest neighbors per two dimensions. Note that for M-PAM or $(M \times M)$-QAM, this expression reduces to (5.13).

Given $\gamma_{\rm c}(\mathcal{A})$ and $K_s(\mathcal{A})$, we may obtain a plot of (5.15) by moving the baseline curve (Figure 2 of Chapter 4) to the left by $\gamma_{\rm c}(\mathcal{A})$ (in dB), and up by a factor of $K_s(\mathcal{A})/4$. The rule of thumb that an increase of a factor of two in $K_s(\mathcal{A})$ over the baseline $K_s(\mathcal{A}) = 4$ costs about 0.2 dB in effective coding gain at $P_s(E) \approx 10^{-5}$ may still be used if $K_s(\mathcal{A})$ is not too large.

Exercise 6 (invariance of coding gain, cont.) Show that in the bandwidth-limited regime the nominal coding gain $\gamma_{\rm c}(\mathcal{A})$ of (5.14), the UBE (5.15) of $P_s(E)$, and the effective coding gain $\gamma_{\rm eff}(\mathcal{A})$ are invariant to scaling, orthogonal transformations and Cartesian products. □

5.6 Design of small signal constellations

The reader may now like to try to find the best constellations of small size M in N dimensions, using coding gain $\gamma_{\rm c}(\mathcal{A})$ as the primary figure of merit, and $K_{\min}(\mathcal{A})$ as a secondary criterion.

Exercise 10 (small nonbinary constellations).

(a) For $M = 4$, the (2×2)-QAM signal set is known to be optimal in $N = 2$ dimensions. Show however that there exists at least one other inequivalent two-dimensional signal set $\mathcal{A}'$ with the same coding gain. Which signal set has the lower "error coefficient" $K_{\min}(\mathcal{A})$?

(b) Show that the coding gain of (a) can be improved in $N = 3$ dimensions. [Hint: consider the signal set $\mathcal{A}'' = \{(1,1,1),(1,-1,-1),(-1,1,-1),(-1,-1,1)\}$.] Sketch $\mathcal{A}''$. What is the geometric name of the polytope whose vertex set is $\mathcal{A}''$?

(c) For $M = 8$ and $N = 2$, propose at least two good signal sets, and determine which one is better. [Open research problem: Find the optimal such signal set, and prove that it is optimal.]

(d) [Open research problem.] For $M = 16$ and $N = 2$, the hexagonal signal set of Figure 1(e), Chapter 4, is thought to be near-optimal. Prove that it is optimal, or find a better one. □

5.7 Summary: Performance analysis and coding gain

The results of this chapter may be summarized very simply.

In the power-limited regime, the nominal coding gain is $\gamma_c(\mathcal{A}) = d^2_{\min}(\mathcal{A})/4E_b$. To the accuracy of the UBE, $P_b(E) \approx K_b(\mathcal{A})Q^{\surd}(2\gamma_c(\mathcal{A})E_b/N_0)$. This curve may be plotted by moving the power-limited baseline curve $P_b(E) \approx Q^{\surd}(2E_b/N_0)$ to the left by $\gamma_c(\mathcal{A})$ in dB and up by a factor of $K_b(\mathcal{A})$. An estimate of the effective coding gain at $P_b(E) \approx 10^{-5}$ is $\gamma_{\text{eff}}(\mathcal{A}) \approx \gamma_c(\mathcal{A}) - (0.2)(\log_2 K_b(\mathcal{A}))$ dB.

In the bandwidth-limited regime, the nominal coding gain is $\gamma_c(\mathcal{A}) = (2^\rho - 1)d^2_{\min}(\mathcal{A})/6E_s$. To the accuracy of the UBE, $P_s(E) \approx K_s(\mathcal{A})Q^{\surd}(3\gamma_c(\mathcal{A})\text{SNR}_{\text{norm}})$. This curve may be plotted by moving the bandwidth-limited baseline curve $P_s(E) \approx 4Q^{\surd}(3\text{SNR}_{\text{norm}})$ to the left by $\gamma_c(\mathcal{A})$ in dB and up by a factor of $K_s(\mathcal{A})/4$. An estimate of the effective coding gain at $P_s(E) \approx 10^{-5}$ is $\gamma_{\text{eff}}(\mathcal{A}) \approx \gamma_c(\mathcal{A}) - (0.2)(\log_2 K_s(\mathcal{A})/4)$ dB.

Appendix: The Q function

The Gaussian probability of error (or Q) function, defined by

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

arises frequently in error probability calculations on Gaussian channels. In this appendix we discuss some of its properties.

As we have seen, there is very often a square root in the argument of the Q function. This suggests that it might have been more useful to define a "Q-of-the-square-root-of" function $Q^{\surd}(x)$ such that $Q^{\surd}(x^2) = Q(x)$; *i.e.,*

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

From now on we will use this $Q^{\surd}$ function instead of the Q function. For example, our baseline curves for 2-PAM and $(M \times M)$-QAM will be

$$\begin{aligned} P_b(E) &= Q^{\surd}(2E_b/N_0); \\ P_s(E) &\approx 4Q^{\surd}(3\text{SNR}_{\text{norm}}). \end{aligned}$$

The Q or $Q^{\surd}$ functions do not have a closed-form expression, but must be looked up in tables. Non-communications texts usually tabulate the complementary error function, namely

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

Evidently $Q(x) = \text{erfc}(x/\sqrt{2})$, and $Q^{\surd}(x) = \text{erfc}(\sqrt{x/2})$.

The main property of the Q or $Q^{\surd}$ function is that it decays exponentially with x^2 according to

$$Q^{\surd}(x^2) = Q(x) \approx e^{-x^2/2}.$$

The following exercise gives several ways to prove this, including upper bounds, a lower bound, and an estimate.

Exercise A (Bounds on the $Q^{\surd}$ function).

(a) As discussed in Chapter 3, the Chernoff bound on the probability that a real random variable Z exceeds b is given by

$$\Pr\{Z \geq b\} \leq \overline{e^{s(Z-b)}}, \quad s \geq 0$$

(since $e^{s(z-b)} \geq 1$ when $z \geq b$, and $e^{s(z-b)} \geq 0$ otherwise). When optimized over $s \geq 0$, the Chernoff exponent is asymptotically correct.

Use the Chernoff bound to show that

$$Q^{\surd}(x^2) \leq e^{-x^2/2}. \tag{5.16}$$

(b) Integrate by parts to derive the upper and lower bounds

$$Q^{\surd}(x^2) < \frac{1}{\sqrt{2\pi x^2}} e^{-x^2/2}; \tag{5.17}$$

$$Q^{\surd}(x^2) > \left(1 - \frac{1}{x^2}\right) \frac{1}{\sqrt{2\pi x^2}} e^{-x^2/2}. \tag{5.18}$$

(c) Here is another way to establish these tight upper and lower bounds. By using a simple change of variables, show that

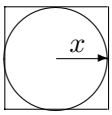
$$Q^{\surd}(x^2) = \frac{1}{\sqrt{2\pi}} e^{\frac{-x^2}{2}} \int_0^\infty \exp\left(\frac{-y^2}{2} - xy\right)\, dy.$$

Then show that

$$1 - \frac{y^2}{2} \leq \exp\left(\frac{-y^2}{2}\right) \leq 1.$$

Putting these together, derive the bounds of part (b).

For (d)-(f), consider a circle of radius x inscribed in a square of side $2x$ as shown below.



(d) Show that the probability that a two-dimensional iid real Gaussian random variable $\mathbf{X}$ with variance $\sigma^2 = 1$ per dimension falls inside the square is equal to $(1 - 2Q^{\surd}(x^2))^2$.

(e) Show that the probability that $\mathbf{X}$ falls inside the circle is $1 - e^{-x^2/2}$. [Hint: write $p_X(\mathbf{x})$ in polar coordinates: *i.e.*, $p_{R\Theta}(r, \theta) = \frac{1}{2\pi} r e^{-r^2/2}$. You can then compute the integral $\int_0^{2\pi} d\theta \int_0^x dr\, p_{R\Theta}(r, \theta)$ in closed form.]

(f) Show that (d) and (e) imply that when x is large,

$$Q^{\surd}(x^2) \leq \frac{1}{4} e^{-x^2/2}. \qquad \square$$

Chapter 6

Introduction to binary block codes

In this chapter we begin to study binary signal constellations, which are the Euclidean-space images of binary block codes. Such constellations have bit rate (nominal spectral efficiency) $\rho \leq 2$ b/2D, and are thus suitable only for the power-limited regime.

We will focus mainly on binary linear block codes, which have a certain useful algebraic structure. Specifically, they are vector spaces over the binary field $\mathbb{F}_2$. A useful infinite family of such codes is the set of Reed-Muller codes.

We discuss the penalty incurred by making hard decisions and then performing classical error-correction, and show how the penalty may be partially mitigated by using erasures, or rather completely by using generalized minimum distance (GMD) decoding.

6.1 Binary signal constellations

In this chapter we will consider constellations that are the Euclidean-space images of binary codes via a coordinatewise 2-PAM map. Such constellations will be called *binary signal constellations*.

A *binary block code* of length n is any subset $\mathcal{C} \subseteq \{0,1\}^n$ of the set of all binary n-tuples of length n. We will usually identify the binary alphabet $\{0,1\}$ with the finite field $\mathbb{F}_2$ with two elements, whose arithmetic rules are those of mod-2 arithmetic. Moreover, we will usually impose the requirement that $\mathcal{C}$ be *linear*; *i.e.,* that $\mathcal{C}$ be a subspace of the n-dimensional vector space $(\mathbb{F}_2)^n$ of all binary n-tuples. We will shortly begin to discuss such algebraic properties.

Each component $x_k \in \{0,1\}$ of a codeword $\mathbf{x} \in \mathcal{C}$ will be mapped to one of the two points $\pm\alpha$ of a 2-PAM signal set $\mathcal{A} = \{\pm\alpha\} \subset \mathbb{R}$ according to a 2-PAM map $s\colon \{0,1\} \to \mathcal{A}$. Explicitly, two standard ways of specifying such a 2-PAM map are

$$\begin{aligned} s(x) &= \alpha(-1)^x; \\ s(x) &= \alpha(1-2x). \end{aligned}$$

The first map is more algebraic in that, ignoring scaling, it is an isomorphism from the additive binary group $\mathbb{Z}_2 = \{0,1\}$ to the multiplicative binary group $\{\pm 1\}$, since $s(x)\cdot s(x') = (-1)^{x+x'} = s(x+x')$. The second map is more geometric, in that it is the composition of a map from $\{0,1\} \in \mathbb{F}_2$ to $\{0,1\} \in \mathbb{R}$, followed by a linear transformation and a translation. However, ultimately both formulas specify the same map:

$$\{s(0) = \alpha, s(1) = -\alpha\}.$$

Under the 2-PAM map, the set $(\mathbb{F}_2)^n$ of all binary n-tuples maps to the set of all real n-tuples of the form $(\pm\alpha, \pm\alpha, \ldots, \pm\alpha)$. Geometrically, this is the set of all 2^n vertices of an n-cube of side 2α centered on the origin. It follows that a binary signal constellation $\mathcal{A}' = s(\mathcal{C})$ based on a binary code $\mathcal{C} \subseteq (\mathbb{F}_2)^n$ maps to a subset of the vertices of this n-cube.

The size of an N-dimensional binary constellation $\mathcal{A}'$ is thus bounded by $|\mathcal{A}'| \leq 2^n$, and its bit rate $\rho = (2/n)\log_2|\mathcal{A}'|$ is bounded by $\rho \leq 2$ b/2D. Thus binary constellations can be used only in the power-limited regime.

Since the n-cube constellation $\mathcal{A}^n = s((\mathbb{F}_2)^n) = (s(\mathbb{F}_2))^n$ is simply the n-fold Cartesian product $\mathcal{A}^n$ of the 2-PAM constellation $\mathcal{A} = s(\mathbb{F}_2) = \{\pm\alpha\}$, its normalized parameters are the same as those of 2-PAM, and it achieves no coding gain. Our hope is that by restricting to a subset $\mathcal{A}' \subset \mathcal{A}^n$, a distance gain can be achieved that will more than offset the rate loss, thus yielding a coding gain.

Example 1. Consider the binary code $\mathcal{C} = \{000, 011, 110, 101\}$, whose four codewords are binary 3-tuples. The bit rate of $\mathcal{C}$ is thus $\rho = 4/3$ b/2D. Its Euclidean-space image $s(\mathcal{C})$ is a set of four vertices of a 3-cube that form a regular tetrahedron, as shown in Figure 1. The minimum squared Euclidean distance of $s(\mathcal{C})$ is $d^2_{\min}(s(\mathcal{C})) = 8\alpha^2$, and every signal point in $s(\mathcal{C})$ has 3 nearest neighbors. The average energy of $s(\mathcal{C})$ is $E(s(\mathcal{C})) = 3\alpha^2$, so its average energy per bit is $E_b = (3/2)\alpha^2$, and its nominal coding gain is

$$\gamma_c(s(\mathcal{C})) = \frac{d^2_{\min}(s(\mathcal{C}))}{4E_b} = \frac{4}{3} \quad (1.25 \text{ dB}).$$

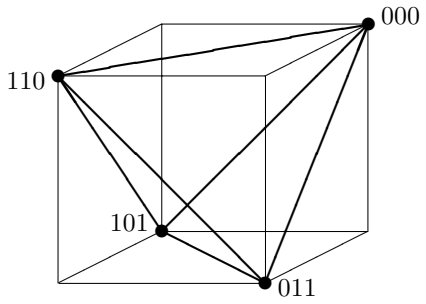


Figure 1. The Euclidean image of the binary code $\mathcal{C} = \{000, 011, 110, 101\}$ is a regular tetrahedron in $\mathbb{R}^3$.

It might at first appear that the restriction of constellation points to vertices of an n-cube might force binary signal constellations to be seriously suboptimal. However, it turns out that when ρ is small, this apparently drastic restriction does not hurt potential performance very much. A capacity calculation using a random code ensemble with binary alphabet $\mathcal{A} = \{\pm\alpha\}$ rather than $\mathbb{R}$ shows that the Shannon limit on E_b/N_0 at $\rho = 1$ b/2D is 0.2 dB rather than 0 dB; *i.e.,* the loss is only 0.2 dB. As $\rho \to 0$, the loss becomes negligible. Therefore at spectral efficiencies $\rho \leq 1$ b/2D, binary signal constellations are good enough.

6.2 Binary linear block codes as binary vector spaces

Practically all of the binary block codes that we consider will be linear. A *binary linear block code* is a set of n-tuples of elements of the binary finite field $\mathbb{F}_2 = \{0, 1\}$ that form a vector space over the field $\mathbb{F}_2$. As we will see in a moment, this means simply that $\mathcal{C}$ must have the group property under n-tuple addition.

We therefore begin by studying the algebraic structure of the binary finite field $\mathbb{F}_2 = \{0, 1\}$ and of vector spaces over $\mathbb{F}_2$. In later chapters we will study codes over general finite fields.

In general, a field $\mathbb{F}$ is a set of elements with two operations, addition and multiplication, which satisfy the usual rules of ordinary arithmetic (*i.e.,* commutativity, associativity, distributivity). A field contains an additive identity 0 such that $a + 0 = a$ for all field elements $a \in \mathbb{F}$, and every field element a has an additive inverse $-a$ such that $a + (-a) = 0$. A field contains a multiplicative identity 1 such that $a \cdot 1 = a$ for all field elements $a \in \mathbb{F}$, and every nonzero field element a has a multiplicative inverse a^{-1} such that $a \cdot a^{-1} = 1$.

The binary field $\mathbb{F}_2$ (sometimes called a Galois field, and denoted by GF(2)) is the finite field with only two elements, namely 0 and 1, which may be thought of as representatives of the even and odd integers, modulo 2. Its addition and multiplication tables are given by the rules of mod 2 (even/odd) arithmetic, with 0 acting as the additive identity and 1 as the multiplicative identity:

$$\begin{array}{ll} 0+0=0 & 0\cdot 0=0 \\ 0+1=1 & 0\cdot 1=0 \\ 1+0=1 & 1\cdot 0=0 \\ 1+1=0 & 1\cdot 1=1 \end{array}$$

In fact these rules are determined by the general properties of 0 and 1 in any field. Notice that the additive inverse of 1 is 1, so $-a = a$ for both field elements.

In general, a vector space V over a field $\mathbb{F}$ is a set of vectors v including 0 such that addition of vectors and multiplication by scalars in $\mathbb{F}$ is well defined, and such that various other vector space axioms are satisfied.

For a vector space over $\mathbb{F}_2$, multiplication by scalars is trivially well defined, since $0v = 0$ and $1v = v$ are automatically in V. Therefore all that really needs to be checked is additive closure, or the *group property* of V under vector addition; *i.e.,* for all $v, v' \in V$, $v + v'$ is in V. Finally, every vector is its own additive inverse, $-v = v$, since

$$v + v = 1v + 1v = (1 + 1)v = 0v = 0.$$

In summary, over a binary field, *subtraction is the same as addition.*

A vector space over $\mathbb{F}_2$ is called a *binary vector space.* The set $(\mathbb{F}_2)^n$ of all binary n-tuples $\mathbf{v} = (v_1, \ldots, v_n)$ under componentwise binary addition is an elementary example of a binary vector space. Here we consider only binary vector spaces which are subspaces $\mathcal{C} \subseteq (\mathbb{F}_2)^n$, which are called *binary linear block codes* of length n.

If $G = \{\mathbf{g}_1, \ldots, \mathbf{g}_k\}$ is a set of vectors in a binary vector space V, then the set $C(G)$ of all binary linear combinations

$$C(G) = \{\sum_j a_j \mathbf{g}_j, a_j \in \mathbb{F}_2, 1 \leq j \leq k\}$$

is a subspace of V, since $C(G)$ evidently has the group property. The set G is called linearly independent if these 2^k binary linear combinations are all distinct, so that the size of $C(G)$ is $|C(G)| = 2^k$. A set G of linearly independent vectors such that $C(G) = V$ is called a *basis* for V, and the elements $\{\mathbf{g}_j, 1 \le j \le k\}$ of the basis are called *generators*. The set $G = \{\mathbf{g}_1, \ldots, \mathbf{g}_k\}$ may be arranged as a $k \times n$ matrix over $\mathbb{F}_2$, called a *generator matrix* for $C(G)$.

The dimension of a binary vector space V is the number k of generators in any basis for V. As with any vector space, the dimension k and a basis G for V may be found by the following greedy algorithm:

Initialization:	set $k = 0$ and $G = \emptyset$ (the empty set);
Do loop:	if $C(G) = V$ we are done, and $\dim V = k$;
	otherwise, increase k by 1 and take any $\mathbf{v} \in V - C(G)$ as $\mathbf{g}_k$.

Thus the size of V is always $|V| = 2^k$ for some integer $k = \dim V$; conversely, $\dim V = \log_2 |V|$.

An *(n, k) binary linear code* $\mathcal{C}$ is any subspace of the vector space $(\mathbb{F}_2)^n$ with dimension k, or equivalently size 2^k. In other words, an (n, k) binary linear code is any set of 2^k binary n-tuples including $\mathbf{0}$ that has the group property under componentwise binary addition.

Example 2 (simple binary linear codes). The (n, n) binary linear code is the set $(\mathbb{F}_2)^n$ of all binary n-tuples, sometimes called the *universe code* of length n. The $(n, 0)$ binary linear code is $\{\mathbf{0}\}$, the set containing only the all-zero n-tuple, sometimes called the *trivial code* of length n. The code consisting of $\mathbf{0}$ and the all-one n-tuple $\mathbf{1}$ is an $(n, 1)$ binary linear code, called the *repetition code* of length n. The code consisting of all n-tuples with an even number of ones is an $(n, n-1)$ binary linear code, called the even-weight or *single-parity-check* (SPC) code of length n. □

6.2.1 The Hamming metric

The geometry of $(\mathbb{F}_2)^n$ is defined by the *Hamming metric*:

$$w_H(\mathbf{x}) = \text{number of ones in } \mathbf{x}.$$

The Hamming metric satisfies the axioms of a metric:

(a) Strict positivity: $w_H(\mathbf{x}) \ge 0$, with equality if and only if $\mathbf{x} = \mathbf{0}$;

(b) Symmetry: $w_H(-\mathbf{x}) = w_H(\mathbf{x})$ (since $-\mathbf{x} = \mathbf{x}$);

(c) Triangle inequality: $w_H(\mathbf{x} + \mathbf{y}) \le w_H(\mathbf{x}) + w_H(\mathbf{y})$.

Therefore the *Hamming distance*,

$$d_H(\mathbf{x}, \mathbf{y}) = w_H(\mathbf{x} - \mathbf{y}) = w_H(\mathbf{x} + \mathbf{y}),$$

may be used to define $(\mathbb{F}_2)^n$ as a metric space, called a *Hamming space*.

We now show that the group property of a binary linear block code $\mathcal{C}$ leads to a remarkable symmetry in the distance distributions from each of the codewords of $\mathcal{C}$ to all other codewords.

Let $\mathbf{x} \in \mathcal{C}$ be a given codeword of $\mathcal{C}$, and consider the set $\{\mathbf{x} + \mathbf{y} \mid \mathbf{y} \in \mathcal{C}\} = \mathbf{x} + \mathcal{C}$ as $\mathbf{y}$ runs through the codewords in $\mathcal{C}$. By the group property of $\mathcal{C}$, $\mathbf{x} + \mathbf{y}$ must be a codeword in $\mathcal{C}$.

Moreover, since $\mathbf{x}+\mathbf{y}=\mathbf{x}+\mathbf{y}'$ if and only if $\mathbf{y}=\mathbf{y}'$, all of these codewords must be distinct. But since the size of the set $\mathbf{x}+\mathcal{C}$ is $|\mathcal{C}|$, this implies that $\mathbf{x}+\mathcal{C}=\mathcal{C}$; *i.e.,* $\mathbf{x}+\mathbf{y}$ runs through all codewords in $\mathcal{C}$ as $\mathbf{y}$ runs through $\mathcal{C}$. Since $d_H(\mathbf{x},\mathbf{y})=w_H(\mathbf{x}+\mathbf{y})$, this implies the following symmetry:

Theorem 6.1 (Distance invariance) *The set of Hamming distances $d_H(\mathbf{x},\mathbf{y})$ from any codeword $\mathbf{x}\in\mathcal{C}$ to all codewords $\mathbf{y}\in\mathcal{C}$ is independent of $\mathbf{x}$, and is equal to the set of distances from $\mathbf{0}\in\mathcal{C}$, namely the set of Hamming weights $w_H(\mathbf{y})$ of all codewords $\mathbf{y}\in\mathcal{C}$.* □

An (n,k) binary linear block code $\mathcal{C}$ is said to have *minimum Hamming distance* d, and is denoted as an (n,k,d) code, if

$$d=\min_{\mathbf{x}\neq\mathbf{y}\in\mathcal{C}} d_H(\mathbf{x},\mathbf{y}).$$

Theorem 6.1 then has the immediate corollary:

Corollary 6.2 (Minimum distance = minimum nonzero weight) *The minimum Hamming distance of $\mathcal{C}$ is equal to the minimum Hamming weight of any nonzero codeword of $\mathcal{C}$. More generally, the number of codewords $\mathbf{y}\in\mathcal{C}$ at distance d from any codeword $\mathbf{x}\in\mathcal{C}$ is equal to the number N_d of weight-d codewords in $\mathcal{C}$, independent of $\mathbf{x}$.* □

Example 2 (cont.) The (n,n) universe code has minimum Hamming distance $d=1$, and the number of words at distance 1 from any codeword is $N_1=n$. The $(n,n-1)$ SPC code has minimum weight and distance $d=2$, and $N_2=n(n-1)/2$. The $(n,1)$ repetition code has $d=n$ and $N_n=1$. By convention, the trivial $(n,0)$ code $\{\mathbf{0}\}$ is said to have $d=\infty$. □

6.2.2 Inner products and orthogonality

A symmetric, bilinear *inner product* on the vector space $(\mathbb{F}_2)^n$ is defined by the $\mathbb{F}_2$-valued dot product

$$\langle\mathbf{x},\mathbf{y}\rangle=\mathbf{x}\cdot\mathbf{y}=\mathbf{x}\mathbf{y}^T=\sum_i x_i y_i,$$

where n-tuples are regarded as row vectors and "T" denotes "transpose." Two vectors are said to be *orthogonal* if $\langle\mathbf{x},\mathbf{y}\rangle=0$.

However, this $\mathbb{F}_2$ inner product does not have a property analogous to strict positivity: $\langle\mathbf{x},\mathbf{x}\rangle=0$ does not imply that $\mathbf{x}=\mathbf{0}$, but only that $\mathbf{x}$ has an even number of ones. Thus it is perfectly possible for a nonzero vector to be orthogonal to itself. Hence $\langle\mathbf{x},\mathbf{x}\rangle$ does not have a key property of the Euclidean squared norm and cannot be used to define a metric space analogous to Euclidean space. The Hamming geometry of $(\mathbb{F}_2)^n$ is very different from Euclidean geometry.

In particular, the projection theorem does not hold, and it is therefore not possible in general to find an orthogonal basis G for a binary vector space $\mathcal{C}$.

Example 3. The $(3,2)$ SPC code consists of the four 3-tuples $\mathcal{C}=\{000,011,101,110\}$. Any two nonzero codewords form a basis for $\mathcal{C}$, but no two such codewords are orthogonal. □

The orthogonal code (*dual code*) $\mathcal{C}^\perp$ to an (n,k) code $\mathcal{C}$ is defined as the set of all n-tuples that are orthogonal to all elements of $\mathcal{C}$:

$$\mathcal{C}^\perp=\{\mathbf{y}\in(\mathbb{F}_2)^n \mid \langle\mathbf{x},\mathbf{y}\rangle=0 \text{ for all } \mathbf{x}\in\mathcal{C}\}.$$

Here are some elementary facts about $\mathcal{C}^\perp$:

(a) $\mathcal{C}^\perp$ is an $(n, n-k)$ binary linear code, and thus has a basis H of size $n-k$.[1]

(b) If G is a basis for $\mathcal{C}$, then a set H of $n-k$ linearly independent n-tuples in $\mathcal{C}^\perp$ is a basis for $\mathcal{C}^\perp$ if and only if every vector in H is orthogonal to every vector in G.

(c) $(\mathcal{C}^\perp)^\perp = \mathcal{C}$.

A basis G for $\mathcal{C}$ consists of k linearly independent n-tuples in $\mathcal{C}$, and is usually written as a $k \times n$ *generator matrix* G of rank k. The code $\mathcal{C}$ then consists of all binary linear combinations $\mathcal{C} = \{\mathbf{a}G, \mathbf{a} \in (\mathbb{F}_2)^k\}$. A basis H for $\mathcal{C}^\perp$ consists of $n-k$ linearly independent n-tuples in $\mathcal{C}^\perp$, and is usually written as an $(n-k) \times n$ matrix H; then $\mathcal{C}^\perp = \{\mathbf{b}H, \mathbf{b} \in (\mathbb{F}_2)^{n-k}\}$. According to property (b) above, $\mathcal{C}$ and $\mathcal{C}^\perp$ are dual codes if and only if their generator matrices satisfy $GH^T = 0$. The transpose H^T of a generator matrix H for $\mathcal{C}^\perp$ is called a *parity-check matrix* for $\mathcal{C}$; it has the property that a vector $\mathbf{x} \in (\mathbb{F}_2)^n$ is in $\mathcal{C}$ if and only if $\mathbf{x}H^T = \mathbf{0}$, since $\mathbf{x}$ is in the dual code to $\mathcal{C}^\perp$ if and only if it is orthogonal to all generators of $\mathcal{C}^\perp$.

Example 2 (cont.; duals of simple codes). In general, the (n, n) universe code and the $(n, 0)$ trivial code are dual codes. The $(n, 1)$ repetition code and the $(n, n-1)$ SPC code are dual codes. Note that the $(2, 1)$ code $\{00, 11\}$ is both a repetition code and an SPC code, and is its own dual; such a code is called *self-dual*. (Self-duality cannot occur in real or complex vector spaces.) □

6.3 Euclidean-space images of binary linear block codes

In this section we derive the principal parameters of a binary signal constellation $s(\mathcal{C})$ from the parameters of the binary linear block code $\mathcal{C}$ on which it is based, namely the parameters (n, k, d) and the number N_d of weight-d codewords in $\mathcal{C}$.

The dimension of $s(\mathcal{C})$ is $N = n$, and its size is $M = 2^k$. It thus supports k bits per block. The bit rate (nominal spectral efficiency) is $\rho = 2k/n$ b/2D. Since $k \leq n$, $\rho \leq 2$ b/2D, and we are in the power-limited regime.

Every point in $s(\mathcal{C})$ is of the form $(\pm\alpha, \pm\alpha, \ldots, \pm\alpha)$, and therefore every point has energy $n\alpha^2$; *i.e.,* the signal points all lie on an n-sphere of squared radius $n\alpha^2$. The average energy per block is thus $E(s(C)) = n\alpha^2$, and the average energy per bit is $E_b = n\alpha^2/k$.

If two codewords $\mathbf{x}, \mathbf{y} \in \mathcal{C}$ have Hamming distance $d_H(\mathbf{x}, \mathbf{y})$, then their Euclidean images $s(\mathbf{x}), s(\mathbf{y})$ will be the same in $n - d_H(\mathbf{x}, \mathbf{y})$ places, and will differ by 2α in $d_H(\mathbf{x}, \mathbf{y})$ places, so

[1] The standard proof of this fact involves finding a *systematic* generator matrix $G = [I_k \mid P]$ for $\mathcal{C}$, where I_k is the $k \times k$ identity matrix and P is a $k \times (n-k)$ check matrix. Then $\mathcal{C} = \{(\mathbf{u}, \mathbf{u}P), \mathbf{u} \in (\mathbb{F}_2)^k\}$, where $\mathbf{u}$ is a free information k-tuple and $\mathbf{u}P$ is a check $(n-k)$-tuple. The dual code $\mathcal{C}^\perp$ is then evidently the code generated by $H = [-P^T \mid I_{n-k}]$, where P^T is the transpose of P; *i.e.,* $\mathcal{C}^\perp = \{(-\mathbf{v}P^T, \mathbf{v}), \mathbf{v} \in (\mathbb{F}_2)^{n-k}\}$, whose dimension is $n-k$.

A more elegant proof based on the fundamental theorem of group homomorphisms (which the reader is not expected to know at this point) is as follows. Let M be the $|\mathcal{C}^\perp| \times n$ matrix whose rows are the codewords of $\mathcal{C}^\perp$. Consider the homomorphism $M^T : (\mathbb{F}_2)^n \to (\mathbb{F}_2)^{|\mathcal{C}^\perp|}$ defined by $\mathbf{y} \mapsto \mathbf{y}M^T$; *i.e.,* $\mathbf{y}M^T$ is the set of inner products of an n-tuple $\mathbf{y} \in (\mathbb{F}_2)^n$ with all codewords $\mathbf{x} \in \mathcal{C}^\perp$. The kernel of this homomorphism is evidently $\mathcal{C}$. By the fundamental theorem of homomorphisms, the image of M^T (the row space of M^T) is isomorphic to the quotient space $(\mathbb{F}_2)^n/\mathcal{C}$, which is isomorphic to $(\mathbb{F}_2)^{n-k}$. Thus the column rank of M is $n-k$. But the column rank is equal to the row rank, which is the dimension of the row space $\mathcal{C}^\perp$ of M. □

their squared Euclidean distance will be[2]

$$\|s(\mathbf{x}) - s(\mathbf{y})\|^2 = 4\alpha^2 d_H(\mathbf{x}, \mathbf{y}).$$

Therefore

$$d^2_{\min}(s(\mathcal{C})) = 4\alpha^2 d_H(\mathcal{C}) = 4\alpha^2 d,$$

where $d = d_H(\mathcal{C})$ is the minimum Hamming distance of $\mathcal{C}$.

It follows that the nominal coding gain of $s(\mathcal{C})$ is

$$\gamma_c(s(C)) = \frac{d^2_{\min}(s(C))}{4E_b} = \frac{kd}{n}. \tag{6.1}$$

Thus the parameters (n, k, d) directly determine $\gamma_c(s(C))$ in this very simple way. (This gives another reason to prefer E_b/N_0 to $\mathrm{SNR}_{\mathrm{norm}}$ in the power-limited regime.)

Moreover, every vector $s(\mathbf{x}) \in s(\mathcal{C})$ has the same number of nearest neighbors $K_{\min}(s(\mathbf{x}))$, namely the number N_d of nearest neighbors to $\mathbf{x} \in \mathcal{C}$. Thus $K_{\min}(s(C)) = N_d$, and $K_b(s(C)) = N_d/k$.

Consequently the union bound estimate of $P_b(E)$ is

$$\begin{aligned} P_b(E) &\approx K_b(s(\mathcal{C}))Q^{\surd}(\gamma_c(s(\mathcal{C}))(2E_b/N_0)) \\ &= \frac{N_d}{k} Q^{\surd}\left(\frac{dk}{n} 2E_b/N_0\right). \end{aligned} \tag{6.2}$$

In summary, the parameters and performance of the binary signal constellation $s(\mathcal{C})$ may be simply determined from the parameters (n, k, d) and N_d of $\mathcal{C}$.

Exercise 1. Let $\mathcal{C}$ be an (n, k, d) binary linear code with d odd. Show that if we append an overall parity check $p = \sum_i x_i$ to each codeword $\mathbf{x}$, then we obtain an $(n + 1, k, d + 1)$ binary linear code $\mathcal{C}'$ with d even. Show that the nominal coding gain $\gamma_c(\mathcal{C}')$ is always greater than $\gamma_c(\mathcal{C})$ if $k > 1$. Conclude that we can focus primarily on linear codes with d even. □

Exercise 2. Show that if $\mathcal{C}$ is a binary linear block code, then in every coordinate position either all codeword components are 0 or half are 0 and half are 1. Show that a coordinate in which all codeword components are 0 may be deleted ("punctured") without any loss in performance, but with savings in energy and in dimension. Show that if $\mathcal{C}$ has no such all-zero coordinates, then $s(\mathcal{C})$ has zero mean: $\mathbf{m}(s(\mathcal{C})) = \mathbf{0}$. □

6.4 Reed-Muller codes

The Reed-Muller (RM) codes are an infinite family of binary linear codes that were among the first to be discovered (1954). For block lengths $n \leq 32$, they are the best codes known with minimum distances d equal to powers of 2. For greater block lengths, they are not in general the best codes known, but in terms of performance *vs.* decoding complexity they are still quite good, since they admit relatively simple ML decoding algorithms.

[2]Moreover, the Eudlidean-space inner product of $s(\mathbf{x})$ and $s(\mathbf{y})$ is

$$\langle s(\mathbf{x}), s(\mathbf{y})\rangle = (n - d_H(\mathbf{x}, \mathbf{y}))\alpha^2 + d_H(\mathbf{x}, \mathbf{y})(-\alpha^2) = (n - 2d_H(\mathbf{x}, \mathbf{y}))\alpha^2.$$

Therefore $s(\mathbf{x})$ and $s(\mathbf{y})$ are orthogonal if and only if $d_H(\mathbf{x}, \mathbf{y}) = n/2$. Also, $s(\mathbf{x})$ and $s(\mathbf{y})$ are antipodal ($s(\mathbf{x}) = -s(\mathbf{y})$) if and only if $d_H(\mathbf{x}, \mathbf{y}) = n$.

For any integers $m \geq 0$ and $0 \leq r \leq m$, there exists an RM code, denoted by $\mathrm{RM}(r, m)$, that has length $n = 2^m$ and minimum Hamming distance $d = 2^{m-r}$, $0 \leq r \leq m$.

For $r = m$, $\mathrm{RM}(m, m)$ is defined as the universe $(2^m, 2^m, 1)$ code. It is helpful also to define RM codes for $r = -1$ by $\mathrm{RM}(-1, m) = (2^m, 0, \infty)$, the trivial code of length 2^m. Thus for $m = 0$, the two RM codes of length 1 are the $(1, 1, 1)$ universe code $\mathrm{RM}(0, 0)$ and the $(1, 0, \infty)$ trivial code $\mathrm{RM}(-1, 0)$.

The remaining RM codes for $m \geq 1$ and $0 \leq r < m$ may be constructed from these elementary codes by the following length-doubling construction, called the $|u|u+v|$ construction (originally due to Plotkin). $\mathrm{RM}(r, m)$ is constructed from $\mathrm{RM}(r-1, m-1)$ and $\mathrm{RM}(r, m-1)$ as

$$\mathrm{RM}(r, m) = \{(\mathbf{u}, \mathbf{u} + \mathbf{v}) \mid \mathbf{u} \in \mathrm{RM}(r, m-1), \mathbf{v} \in \mathrm{RM}(r-1, m-1)\}. \tag{6.3}$$

From this construction, it is easy to prove the following facts by recursion:

(a) $\mathrm{RM}(r, m)$ is a binary linear block code with length $n = 2^m$ and dimension

$$k(r, m) = k(r, m-1) + k(r-1, m-1).$$

(b) The codes are nested, in the sense that $\mathrm{RM}(r-1, m) \subseteq \mathrm{RM}(r, m)$.

(c) The minimum distance of $\mathrm{RM}(r, m)$ is $d = 2^{m-r}$ if $r \geq 0$ (if $r = -1$, then $d = \infty$).

We verify that these assertions hold for $\mathrm{RM}(0, 0)$ and $\mathrm{RM}(-1, 0)$.

For $m \geq 1$, the linearity and length of $\mathrm{RM}(r, m)$ are obvious from the construction. The dimension (size) follows from the fact that $(\mathbf{u}, \mathbf{u} + \mathbf{v}) = \mathbf{0}$ if and only if $\mathbf{u} = \mathbf{v} = \mathbf{0}$.

Exercise 5 below shows that the recursion for $k(r, m)$ leads to the explicit formula

$$k(r, m) = \sum_{0 \leq j \leq r} \binom{m}{j}, \tag{6.4}$$

where $\binom{m}{j}$ denotes the combinatorial coefficient $\frac{m!}{j!(m-j)!}$.

The nesting property for m follows from the nesting property for $m - 1$.

Finally, we verify that the minimum nonzero weight of $\mathrm{RM}(r, m)$ is 2^{m-r} as follows:

(a) if $\mathbf{u} = \mathbf{0}$, then $w_H((\mathbf{0}, \mathbf{v})) = w_H(\mathbf{v}) \geq 2^{m-r}$ if $\mathbf{v} \neq \mathbf{0}$, since $\mathbf{v} \in \mathrm{RM}(r-1, m-1)$.

(b) if $\mathbf{u} + \mathbf{v} = \mathbf{0}$, then $\mathbf{u} = \mathbf{v} \in \mathrm{RM}(r-1, m-1)$ and $w_H((\mathbf{v}, \mathbf{0})) \geq 2^{m-r}$ if $\mathbf{v} \neq \mathbf{0}$.

(c) if $\mathbf{u} \neq \mathbf{0}$ and $\mathbf{u} + \mathbf{v} \neq \mathbf{0}$, then both $\mathbf{u}$ and $\mathbf{u} + \mathbf{v}$ are in $\mathrm{RM}(r, m-1)$ (since $\mathrm{RM}(r-1, m-1)$ is a subcode of $\mathrm{RM}(r, m-1)$), so

$$w_H((\mathbf{u}, \mathbf{u} + \mathbf{v})) = w_H(\mathbf{u}) + w_H(\mathbf{u} + \mathbf{v}) \geq 2 \cdot 2^{m-r-1} = 2^{m-r}.$$

Equality clearly holds for $(\mathbf{0}, \mathbf{v})$, $(\mathbf{v}, \mathbf{0})$ or $(\mathbf{u}, \mathbf{u})$ if we choose $\mathbf{v}$ or $\mathbf{u}$ as a minimum-weight codeword from their respective codes.

The $|u|u+v|$ construction suggests the following tableau of RM codes:

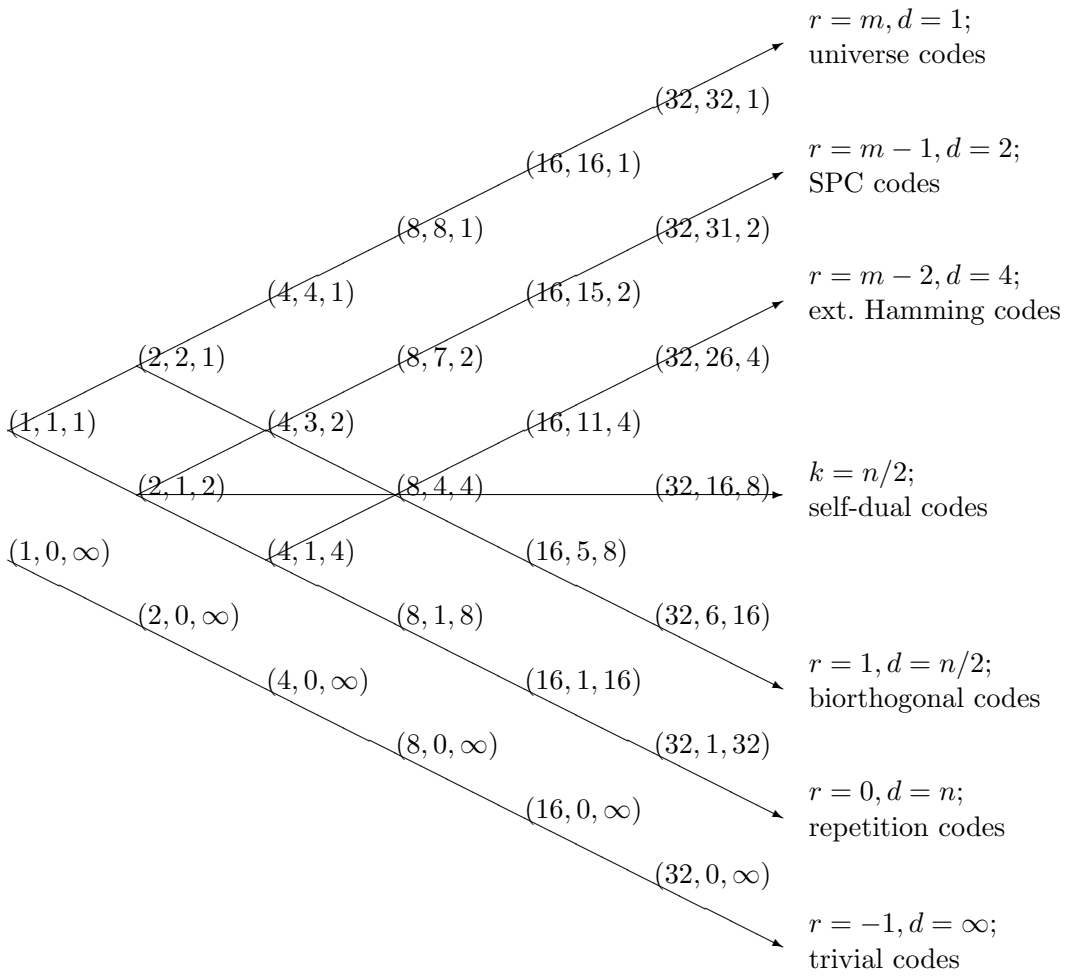


Figure 2. Tableau of Reed-Muller codes.

In this tableau each RM code lies halfway between the two codes of half the length that are used to construct it in the $|u|u+v|$ construction, from which we can immediately deduce its dimension k.

Exercise 3. Compute the parameters (k, d) of the RM codes of lengths $n = 64$ and 128. □

There is a known closed-form formula for the number N_d of codewords of minimum weight $d = 2^{m-r}$ in $\mathrm{RM}(r, m)$:

$$N_d = 2^r \prod_{0 \le i \le m-r-1} \frac{2^{m-i} - 1}{2^{m-r-i} - 1}. \tag{6.5}$$

Example 4. The number of weight-8 words in the $(32, 16, 8)$ code $\mathrm{RM}(2, 5)$ is

$$N_8 = 4\frac{31 \cdot 15 \cdot 7}{7 \cdot 3 \cdot 1} = 620.$$

The nominal coding gain of $\mathrm{RM}(2, 5)$ is $\gamma_c(\mathcal{C}) = 4$ (6.02 dB); however, since $K_b = N_8/k = 38.75$, the effective coding gain by our rule of thumb is only about $\gamma_{\mathrm{eff}}(\mathcal{C}) \approx 5.0$ dB. □

The codes with $r = m - 1$ are *single-parity-check (SPC) codes* with $d = 2$. These codes have nominal coding gain $2(k/n)$, which goes to 2 (3.01 dB) as $n \to \infty$; however, since $N_d = 2^m(2^m - 1)/2$, we have $K_b = 2^{m-1} \to \infty$, which ultimately limits the effective coding gain.

The codes with $r = m - 2$ are *extended Hamming (EH) codes* with $d = 4$. These codes have nominal coding gain $4(k/n)$, which goes to 4 (6.02 dB) as $n \to \infty$; however, since $N_d = 2^m(2^m - 1)(2^m - 2)/24$, we again have $K_b \to \infty$.

Exercise 4 (optimizing SPC and EH codes). Using the rule of thumb that a factor of two increase in K_b costs 0.2 dB in effective coding gain, find the value of n for which an $(n, n-1, 2)$ SPC code has maximum effective coding gain, and compute this maximum in dB. Similarly, find m such that a $(2^m, 2^m - m - 1, 4)$ extended Hamming code has maximum effective coding gain, using $N_d = 2^m(2^m - 1)(2^m - 2)/24$, and compute this maximum in dB. □

The codes with $r = 1$ (*first-order Reed-Muller codes*) are interesting, because as shown in Exercise 5 they generate biorthogonal signal sets of dimension $n = 2^m$ and size 2^{m+1}, with nominal coding gain $(m + 1)/2 \to \infty$. It is known that as $n \to \infty$ this sequence of codes can achieve arbitrarily small $\Pr(E)$ for any E_b/N_0 greater than the ultimate Shannon limit, namely $E_b/N_0 > \ln 2$ (-1.59 dB).

Exercise 5 (biorthogonal codes). We have shown that the first-order Reed-Muller codes $\mathrm{RM}(1, m)$ have parameters $(2^m, m+1, 2^{m-1})$, and that the $(2^m, 1, 2^m)$ repetition code $\mathrm{RM}(0, m)$ is a subcode.

(a) Show that $\mathrm{RM}(1, m)$ has one word of weight 0, one word of weight 2^m, and $2^{m+1} - 2$ words of weight 2^{m-1}. [Hint: first show that the $\mathrm{RM}(1, m)$ code consists of 2^m complementary codeword pairs $\{\mathbf{x}, \mathbf{x} + \mathbf{1}\}$.]

(b) Show that the Euclidean image of an $\mathrm{RM}(1, m)$ code is an $M = 2^{m+1}$ biorthogonal signal set. [Hint: compute all inner products between code vectors.]

(c) Show that the code $\mathcal{C}'$ consisting of all words in $\mathrm{RM}(1, m)$ with a 0 in any given coordinate position is a $(2^m, m, 2^{m-1})$ binary linear code, and that its Euclidean image is an $M = 2^m$ orthogonal signal set. [Same hint as in part (a).]

(d) Show that the code $\mathcal{C}''$ consisting of the code words of $\mathcal{C}'$ with the given coordinate deleted ("punctured") is a binary linear $(2^m - 1, m, 2^{m-1})$ code, and that its Euclidean image is an $M = 2^m$ simplex signal set. [Hint: use Exercise 7 of Chapter 5.] □

In Exercise 2 of Chapter 1, it was shown how a 2^m-orthogonal signal set $\mathcal{A}$ can be constructed as the image of a $2^m \times 2^m$ binary Hadamard matrix. The corresponding 2^{m+1}-biorthogonal signal set $\pm\mathcal{A}$ is identical to that constructed above from the $(2^m, m + 1, 2^{m-1})$ first-order RM code.

The code dual to $\mathrm{RM}(r, m)$ is $\mathrm{RM}(m - r - 1, m)$; this can be shown by recursion from the facts that the $(1, 1)$ and $(1, 0)$ codes are duals and that by bilinearity

$$\langle(\mathbf{u}, \mathbf{u} + \mathbf{v}), (\mathbf{u}', \mathbf{u}' + \mathbf{v}')\rangle = \langle \mathbf{u}, \mathbf{u}'\rangle + \langle \mathbf{u} + \mathbf{v}, \mathbf{u}' + \mathbf{v}'\rangle = \langle \mathbf{u}, \mathbf{v}'\rangle + \langle \mathbf{v}, \mathbf{u}'\rangle + \langle \mathbf{v}, \mathbf{v}'\rangle,$$

since $\langle \mathbf{u}, \mathbf{u}'\rangle + \langle \mathbf{u}, \mathbf{u}'\rangle = 0$. In particular, this confirms that the repetition and SPC codes are duals, and shows that the biorthogonal and extended Hamming codes are duals.

This also shows that RM codes with $k/n = 1/2$ are self-dual. The nominal coding gain of a rate-1/2 RM code of length 2^m (m odd) is $2^{(m-1)/2}$, which goes to infinity as $m \to \infty$. It seems likely that as $n \to \infty$ this sequence of codes can achieve arbitrarily small $\Pr(E)$ for any E_b/N_0 greater than the Shannon limit for $\rho = 1$ b/2D, namely $E_b/N_0 > 1$ (0 dB).

Exercise 6 (generator matrices for RM codes). Let square $2^m \times 2^m$ matrices U_m, $m \geq 1$, be specified recursively as follows. The matrix U_1 is the 2×2 matrix

$$U_1 = \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix}.$$

The matrix U_m is the $2^m \times 2^m$ matrix

$$U_m = \begin{bmatrix} U_{m-1} & 0 \\ U_{m-1} & U_{m-1} \end{bmatrix}.$$

(In other words, U_m is the m-fold tensor product of U_1 with itself.)

(a) Show that RM(r, m) is generated by the rows of U_m of Hamming weight 2^{m-r} or greater. [Hint: observe that this holds for $m = 1$, and prove by recursion using the $|u|u+v|$ construction.] For example, give a generator matrix for the $(8, 4, 4)$ RM code.

(b) Show that the number of rows of U_m of weight 2^{m-r} is $\binom{m}{r}$. [Hint: use the fact that $\binom{m}{r}$ is the coefficient of z^{m-r} in the integer polynomial $(1 + z)^m$.]

(c) Conclude that the dimension of RM(r, m) is $k(r, m) = \sum_{0 \leq j \leq r} \binom{m}{j}$. □

6.4.1 Effective coding gains of RM codes

We provide below a table of the nominal spectral efficiency ρ, nominal coding gain γ_c, number of nearest neighbors N_d, error coefficient per bit K_b, and estimated effective coding gain γ_{eff} at $P_b(E) \approx 10^{-5}$ for various Reed-Muller codes, so that the student can consider these codes as components in system design exercises.

In later lectures, we will consider trellis representations and trellis decoding of RM codes. We give here two complexity parameters of the minimal trellises for these codes: the state complexity s (the binary logarithm of the maximum number of states in a minimal trellis), and the branch complexity t (the binary logarithm of the maximum number of branches per section in a minimal trellis). The latter parameter gives a more accurate estimate of decoding complexity.

code	ρ	γ_c	(dB)	N_d	K_b	γ_{eff} (dB)	s	t
(8,7,2)	1.75	7/4	2.43	28	4	2.0	1	2
(8,4,4)	1.00	2	3.01	14	4	2.6	2	3
(16,15,2)	1.88	15/8	2.73	120	8	2.1	1	2
(16,11,4)	1.38	11/4	4.39	140	13	3.7	3	5
(16, 5,8)	0.63	5/2	3.98	30	6	3.5	3	4
(32,31, 2)	1.94	31/16	2.87	496	16	2.1	1	2
(32,26, 4)	1.63	13/4	5.12	1240	48	4.0	4	7
(32,16, 8)	1.00	4	6.02	620	39	4.9	6	9
(32, 6,16)	0.37	3	4.77	62	10	4.2	4	5
(64,63, 2)	1.97	63/32	2.94	2016	32	1.9	1	2
(64,57, 4)	1.78	57/16	5.52	10416	183	4.0	5	9
(64,42, 8)	1.31	21/4	7.20	11160	266	5.6	10	16
(64,22,16)	0.69	11/2	7.40	2604	118	6.0	10	14
(64, 7,32)	0.22	7/2	5.44	126	18	4.6	5	6

Table 1. Parameters of RM codes with lengths $n \leq 64$.

6.5 Decoding of binary block codes

In this section we will first show that with binary codes MD decoding reduces to "maximum-reliability decoding." We will then discuss the penalty incurred by making hard decisions and then performing classical error-correction. We show how the penalty may be partially mitigated by using erasures, or rather completely by using generalized minimum distance (GMD) decoding.

6.5.1 Maximum-reliability decoding

All of our performance estimates assume minimum-distance (MD) decoding. In other words, given a received sequence $\mathbf{r} \in \mathbb{R}^n$, the receiver must find the signal $s(\mathbf{x})$ for $\mathbf{x} \in \mathcal{C}$ such that the squared distance $\|\mathbf{r} - s(\mathbf{x})\|^2$ is minimum. We will show that in the case of binary codes, MD decoding reduces to maximum-reliability (MR) decoding.

Since $\|s(\mathbf{x})\|^2 = n\alpha^2$ is independent of $\mathbf{x}$ with binary constellations $s(\mathcal{C})$, MD decoding is equivalent to *maximum-inner-product decoding*: find the signal $s(\mathbf{x})$ for $\mathbf{x} \in \mathcal{C}$ such that the inner product

$$\langle \mathbf{r}, s(\mathbf{x}) \rangle = \sum_k r_k s(x_k)$$

is maximum. Since $s(x_k) = (-1)^{x_k}\alpha$, the inner product may be expressed as

$$\langle \mathbf{r}, s(\mathbf{x}) \rangle = \alpha \sum_k r_k (-1)^{x_k} = \alpha \sum_k |r_k| \ \mathrm{sgn}(r_k)(-1)^{x_k}$$

The sign $\mathrm{sgn}(r_k) \in \{\pm 1\}$ is often regarded as a "hard decision" based on r_k, indicating which of the two possible signals $\{\pm\alpha\}$ is more likely in that coordinate without taking into account the remaining coordinates. The magnitude $|r_k|$ may be viewed as the reliability of the hard decision. This rule may thus be expressed as: find the codeword $\mathbf{x} \in \mathcal{C}$ that maximizes the reliability

$$r(\mathbf{x} \mid \mathbf{r}) = \sum_k |r_k| (-1)^{e(x_k, r_k)},$$

where the "error" $e(x_k, r_k)$ is 0 if the signs of $s(x_k)$ and r_k agree, or 1 if they disagree. We call this rule *maximum-reliability decoding*.

Any of these optimum decision rules is easy to implement for small constellations $s(\mathcal{C})$. However, without special tricks they require at least one computation for every codeword $\mathbf{x} \in \mathcal{C}$, and therefore become impractical when the number 2^k of codewords becomes large. Finding simpler decoding algorithms that give a good tradeoff of performance *vs.* complexity, perhaps only for special classes of codes, has therefore been the major theme of practical coding research.

For example, the Wagner decoding rule, the earliest "soft-decision" decoding algorithm (*circa* 1955), is an optimum decoding rule for the special class of $(n, n-1, 2)$ SPC codes that requires many fewer than 2^{n-1} computations.

Exercise 7 ("Wagner decoding"). Let $\mathcal{C}$ be an $(n, n-1, 2)$ SPC code. The Wagner decoding rule is as follows. Make hard decisions on every symbol r_k, and check whether the resulting binary word is in $\mathcal{C}$. If so, accept it. If not, change the hard decision in the symbol r_k for which the reliability metric $|r_k|$ is minimum. Show that the Wagner decoding rule is an optimum decoding rule for SPC codes. [Hint: show that the Wagner rule finds the codeword $\mathbf{x} \in \mathcal{C}$ that maximizes $r(\mathbf{x} \mid \mathbf{r})$.] □

6.5.2 Hard decisions and error-correction

Early work on decoding of binary block codes assumed hard decisions on every symbol, yielding a hard-decision n-tuple $\mathbf{y} \in (\mathbb{F}_2)^n$. The main decoding step is then to find the codeword $\mathbf{x} \in \mathcal{C}$ that is closest to $\mathbf{y}$ in Hamming space. This is called *error-correction*.

If $\mathcal{C}$ is a linear (n, k, d) code, then, since the Hamming metric is a true metric, no error can occur when a codeword $\mathbf{x}$ is sent unless the number of hard decision errors $t = d_H(\mathbf{x}, \mathbf{y})$ is at least as great as half the minimum Hamming distance, $t \geq d/2$. For many classes of binary block codes, efficient algebraic error-correction algorithms exist that are guaranteed to decode correctly provided that $2t < d$. This is called *bounded-distance error-correction*.

Example 5 (Hamming codes). The first binary error-correction codes were the Hamming codes (mentioned in Shannon's original paper). A Hamming code $\mathcal{C}$ is a $(2^m - 1, 2^m - m - 1, 3)$ code that may be found by puncturing a $(2^m, 2^m - m - 1, 4)$ extended Hamming RM$(m - 2, m)$ code in any coordinate. Its dual $\mathcal{C}^\perp$ is a $(2^m - 1, m, 2^{m-1})$ code whose Euclidean image is a 2^m-simplex constellation. For example, the simplest Hamming code is the $(3, 1, 3)$ repetition code; its dual is the $(3, 2, 2)$ SPC code, whose image is the 4-simplex constellation of Figure 1.

The generator matrix of $\mathcal{C}^\perp$ is an $m \times (2^m - 1)$ matrix H whose $2^m - 1$ columns must run through the set of all nonzero binary m-tuples in some order (else $\mathcal{C}$ would not be guaranteed to correct any single error; see next paragraph).

Since $d = 3$, a Hamming code should be able to correct any single error. A simple method for doing so is to compute the "syndrome"

$$\mathbf{y}H^T = (\mathbf{x} + \mathbf{e})H^T = \mathbf{e}H^T,$$

where $\mathbf{e} = \mathbf{x} + \mathbf{y}$. If $\mathbf{y}H^T = \mathbf{0}$, then $\mathbf{y} \in \mathcal{C}$ and $\mathbf{y}$ is assumed to be correct. If $\mathbf{y}H^T \neq \mathbf{0}$, then the syndrome $\mathbf{y}H^T$ is equal to one of the rows in H^T, and a single error is assumed to have occurred in the corresponding position. Thus it is always possible to change any $\mathbf{y} \in (\mathbb{F}_2)^n$ into a codeword by changing at most one bit.

This implies that the 2^{n-m} "Hamming spheres" of radius 1 and size 2^m centered on the 2^{n-m} codewords $\mathbf{x}$, which consist of $\mathbf{x}$ and the $n = 2^m - 1$ n-tuples $\mathbf{y}$ within Hamming distance 1 of $\mathbf{x}$, form an exhaustive partition of the set of 2^n n-tuples that comprise Hamming n-space $(\mathbb{F}_2)^n$.

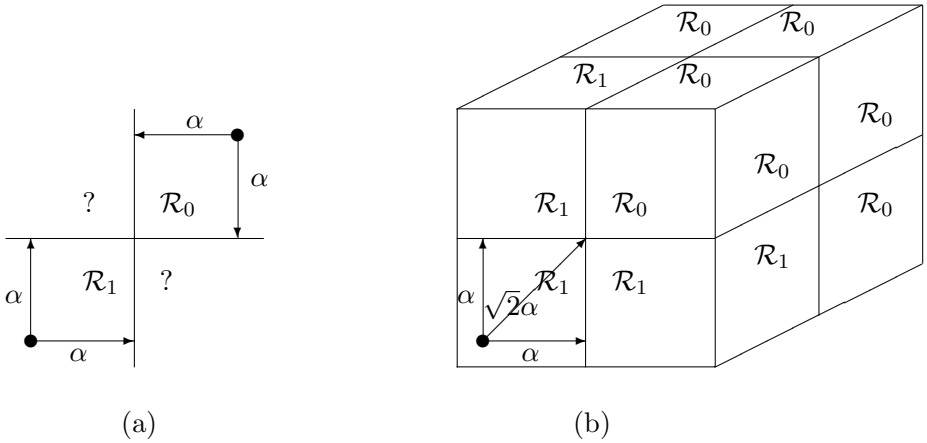
In summary, Hamming codes form a "perfect" Hamming sphere-packing of $(\mathbb{F}_2)^n$, and have a simple single-error-correction algorithm. □

We now show that even if an error-correcting decoder does optimal MD decoding in Hamming space, there is a loss in coding gain of the order of 3 dB relative to MD Euclidean-space decoding.

Assume an (n, k, d) binary linear code $\mathcal{C}$ with d odd (the situation is worse when d is even). Let $\mathbf{x}$ be the transmitted codeword; then there is at least one codeword at Hamming distance d from $\mathbf{x}$, and thus at least one real n-tuple in $s(\mathcal{C})$ at Euclidean distance $4\alpha^2 d$ from $s(\mathbf{x})$. For any $\varepsilon > 0$, a hard-decision decoding error will occur if the noise exceeds $\alpha + \varepsilon$ in any $(d + 1)/2$ of the places in which that word differs from $\mathbf{x}$. Thus with hard decisions the minimum squared distance to the decision boundary in Euclidean space is $\alpha^2(d + 1)/2$. (For d even, it is $\alpha^2 d/2$.)

On the other hand, with "soft decisions" (reliability weights) and MD decoding, the minimum squared distance to any decision boundary in Euclidean space is $\alpha^2 d$. To the accuracy of the union bound estimate, the argument of the $Q\sqrt{}$ function thus decreases with hard-decision decoding by a factor of $(d + 1)/2d$, or approximately $1/2$ (-3 dB) when d is large. (When d is even, this factor is exactly $1/2$.)

Example 6 (Hard and soft decoding of antipodal codes). Let $\mathcal{C}$ be the $(2, 1, 2)$ binary code; then the two signal points in $s(\mathcal{C})$ are antipodal, as shown in Figure 3(a) below. With hard decisions, real 2-space $\mathbb{R}^2$ is partitioned into four quadrants, which must then be assigned to one or the other of the two signal points. Of course, two of the quadrants are assigned to the signal points that they contain. However, no matter how the other two quadrants are assigned, there will be at least one decision boundary at squared distance α^2 from a signal point, whereas with MD decoding the decision boundary is at distance $2\alpha^2$ from both signal points. The loss in the error exponent of $P_b(E)$ is therefore a factor of 2 (3 dB).



(a) (b)

Figure 3. Decision regions in $\mathbb{R}^n$ with hard decisions. (a) $(2, 1, 2)$ code; (b) $(3, 1, 3)$ code.

Similarly, if $\mathcal{C}$ is the $(3, 1, 3)$ code, then $\mathbb{R}^3$ is partitioned by hard decisions into 8 octants, as shown in Figure 3(b). In this case (the simplest example of a Hamming code), it is clear how best to assign four octants to each signal point. The squared distance from each signal point to the nearest decision boundary is now $2\alpha^2$, compared to $3\alpha^2$ with "soft decisions" and MD decoding in Euclidean space, for a loss of $2/3$ (1.76 dB) in the error exponent. □

6.5.3 Erasure-and-error-correction

A decoding method halfway between hard-decision and "soft-decision" (reliability-based) techniques involves the use of "erasures." With this method, the first step of the receiver is to map each received signal r_k into one of three values, say $\{0, 1, ?\}$, where for some threshold T,

$$\begin{aligned} r_k \to \ & 0 \quad \text{if} \quad r_k > T; \\ r_k \to \ & 1 \quad \text{if} \quad r_k < -T; \\ r_k \to \ & ? \quad \text{if} \quad -T \le r_k \le T. \end{aligned}$$

The decoder subsequently tries to map the ternary-valued n-tuple into the closest codeword $\mathbf{x} \in \mathcal{C}$ in Hamming space, where the erased positions are ignored in measuring Hamming distance.

If there are s erased positions, then the minimum distance between codewords is at least $d - s$ in the unerased positions, so correct decoding is guaranteed if the number t of errors in the unerased positions satisfies $t < (d-s)/2$, or equivalently if $2t + s < d$. For many classes of binary block codes, efficient algebraic erasure-and-error-correcting algorithms exist that are guaranteed to decode correctly if $2t + s < d$. This is called *bounded-distance erasure-and-error-correction.*

Erasure-and-error-correction may be viewed as a form of MR decoding in which all reliabilities $|r_k|$ are made equal in the unerased positions, and are set to 0 in the erased positions.

The ternary-valued output allows a closer approximation to the optimum decision regions in Euclidean space than with hard decisions, and therefore reduces the loss. With an optimized threshold T, the loss is typically only about half as much (in dB).

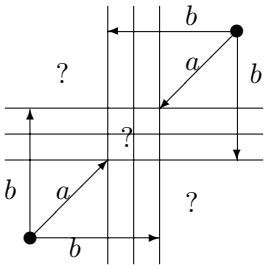


Figure 4. Decision regions with hard decisions and erasures for the $(2, 1, 2)$ code.

Example 6 (cont.). Figure 4 shows the 9 decision regions for the $(2, 1, 2)$ code that result from hard decisions and/or erasures on each symbol. Three of the resulting regions are ambiguous. The minimum squared distances to these regions are

$$\begin{aligned} a^2 &= 2(\alpha - T)^2 \\ b^2 &= (\alpha + T)^2. \end{aligned}$$

To maximize the minimum of a^2 and b^2, we make $a^2 = b^2$ by choosing $T = \frac{\sqrt{2}-1}{\sqrt{2}+1}\alpha$, which yields

$$a^2 = b^2 \frac{8}{(\sqrt{2}+1)^2}\alpha^2 = 1.372\alpha^2.$$

This is about 1.38 dB better than the squared Euclidean distance α^2 achieved with hard decisions only, but is still 1.63 dB worse than the $2\alpha^2$ achieved with MD decoding. □

Exercise 8 (Optimum threshold T). Let $\mathcal{C}$ be a binary code with minimum distance d, and let received symbols be mapped into hard decisions or erasures as above. Show that:

(a) For any integers t and s such that $2t + s \geq d$ and for any decoding rule, there exists some pattern of t errors and s erasures that will cause a decoding error;

(b) The minimum squared distance from any signal point to its decoding decision boundary is equal to at least $\min_{2t+s \geq d} \{s(\alpha - T)^2 + t(\alpha + T)^2\}$;

(c) The value of T that maximizes this minimum squared distance is $T = \frac{\sqrt{2}-1}{\sqrt{2}+1}\alpha$, in which case the minimum squared distance is equal to $\frac{4}{(\sqrt{2}+1)^2}\alpha^2 d = 0.686\ \alpha^2 d$. Again, this is a loss of 1.63 dB relative to the squared distance $\alpha^2 d$ that is achieved with MD decoding. □

6.5.4 Generalized minimum distance decoding

A further step in this direction that achieves almost the same performance as MD decoding, to the accuracy of the union bound estimate, yet still permits algebraic decoding algorithms, is generalized minimum distance (GMD) decoding.

In GMD decoding, the decoder keeps both the hard decision $\mathrm{sgn}(r_k)$ and the reliability $|r_k|$ of each received symbol, and orders them in order of their reliability.

The GMD decoder then performs a series of erasure-and-error decoding trials in which the $s = d-1, d-3, \ldots$ least reliable symbols are erased. (The intermediate trials are not necessary because if $d-s$ is even and $2t < d-s$, then also $2t < d-s-1$, so the trial with one additional erasure will find the same codeword.) The number of such trials is $d/2$ if d is even, or $(d+1)/2$ if d is odd; *i.e.,* the number of trials needed is $\lceil d/2 \rceil$.

Each trial may produce a candidate codeword. The set of $\lceil d/2 \rceil$ trials may thus produce up to $\lceil d/2 \rceil$ distinct candidate codewords. These words may finally be compared according to their reliability $\mathbf{r}(\mathbf{x} \mid \mathbf{r})$ (or any equivalent optimum metric), and the best candidate chosen.

Example 7. For an $(n, n-1, 2)$ SPC code, GMD decoding performs just one trial with the least reliable symbol erased; the resulting candidate codeword is the unique codeword that agrees with all unerased symbols. Therefore in this case the GMD decoding rule is equivalent to the Wagner decoding rule (Exercise 7), which implies that it is optimum. □

It can be shown that no error can occur with a GMD decoder provided that the squared norm $||\mathbf{n}||^2$ of the noise vector is less than $\alpha^2 d$; *i.e.,* the squared distance from any signal point to its decision boundary is $\alpha^2 d$, just as for MD decoding. Thus there is no loss in coding gain or error exponent compared to MD decoding.

It has been shown that for the most important classes of algebraic block codes, GMD decoding can be performed with little more complexity than ordinary hard-decision or erasures-and-errors decoding. Furthermore, it has been shown that not only is the error exponent of GMD decoding equal to that of optimum MD decoding, but also the error coefficient and thus the union bound estimate are the same, provided that GMD decoding is augmented to include a d-erasure-correction trial (a purely algebraic solution of the $n-k$ linear parity-check equations for the d unknown erased symbols).

However, GMD decoding is a bounded-distance decoding algorithm, so its decision regions are like spheres of squared radius $\alpha^2 d$ that lie within the MD decision regions $\mathcal{R}_j$. For this reason GMD decoding is inferior to MD decoding, typically improving over erasure-and-error-correction by 1 dB or less. GMD decoding has rarely been used in practice.

6.5.5 Summary

In conclusion, hard decisions allow the use of efficient algebraic decoding algorithms, but incur a significant SNR penalty, of the order of 3 dB. By using erasures, about half of this penalty can be avoided. With GMD decoding, efficient algebraic decoding algorithms can in principle be used with no loss in performance, at least as estimated by the the union bound estimate.

Chapter 7

Introduction to finite fields

This chapter provides an introduction to several kinds of abstract algebraic structures, particularly groups, fields, and polynomials. Our primary interest is in finite fields, *i.e.,* fields with a finite number of elements (also called Galois fields). In the next chapter, finite fields will be used to develop Reed-Solomon (RS) codes, the most useful class of algebraic codes. Groups and polynomials provide the requisite background to understand finite fields.

A field is more than just a set of elements: it is a set of elements under two operations, called addition and multiplication, along with a set of properties governing these operations. The addition and multiplication operations also imply inverse operations called subtraction and division. The reader is presumably familiar with several examples of fields, such as the real field $\mathbb{R}$, the complex field $\mathbb{C}$, the field of rational numbers $\mathbb{Q}$, and the binary field $\mathbb{F}_2$.

7.1 Summary

In this section we briefly summarize the results of this chapter. The main body of the chapter will be devoted to defining and explaining these concepts, and to proofs of these results.

For each prime p and positive integer $m \geq 1$, there exists a finite field $\mathbb{F}_{p^m}$ with p^m elements, and there exists no finite field with q elements if q is not a prime power. Any two fields with p^m elements are isomorphic.

The integers modulo p form a prime field $\mathbb{F}_p$ under mod-p addition and multiplication. The polynomials $\mathbb{F}_p[x]$ over $\mathbb{F}_p$ modulo an irreducible polynomial $g(x) \in \mathbb{F}_p[x]$ of degree m form a finite field with p^m elements under mod-$g(x)$ addition and multiplication. For every prime p, there exists at least one irreducible polynomial $g(x) \in \mathbb{F}_p[x]$ of each positive degree $m \geq 1$, so all finite fields may be constructed in this way.

Under addition, $\mathbb{F}_{p^m}$ is isomorphic to the vector space $(\mathbb{F}_p)^m$. Under multiplication, the nonzero elements of $\mathbb{F}_{p^m}$ form a cyclic group $\{1, \alpha, \ldots, \alpha^{p^m-2}\}$ generated by a primitive element $\alpha \in \mathbb{F}_{p^m}$.

The elements of $\mathbb{F}_{p^m}$ are the p^m roots of the polynomial $x^{p^m} - x \in \mathbb{F}_p[x]$. The polynomial $x^{p^m} - x$ is the product of all monic irreducible polynomials $g(x) \in \mathbb{F}_p[x]$ such that $\deg g(x)$ divides m. The roots of a monic irreducible polynomial $g(x) \in \mathbb{F}_p[x]$ form a cyclotomic coset of $\deg g(x)$ elements of $\mathbb{F}_{p^m}$ which is closed under the operation of raising to the pth power.

For every n that divides m, $\mathbb{F}_{p^m}$ contains a subfield with p^n elements.

For further reading on this beautiful subject, see [E. R. Berlekamp, *Algebraic Coding Theory*, Aegean Press, 1984], [R. Lidl and H. Niederreiter, *Introduction to Finite Fields and their Applications*, Cambridge University Press, 1986] or [R. J. McEliece, *Finite Fields for Computer Scientists and Engineers*, Kluwer, 1987], [M. R. Schroeder, *Number Theory in Science and Communication*, Springer, 1986], or indeed any book on finite fields or algebraic coding theory.

7.2 The integers

We begin with a brief review of the familiar factorization properties of the set $\mathbb{Z}$ of integers. We will use these properties immediately in our discussion of cyclic groups and their subgroups and of prime fields. Moreover, we will model our later discussion of the factorization properties of polynomials on the discussion here.

7.2.1 Definitions

An integer n is said to be a *divisor* of an integer i if i is an integer multiple of n; *i.e.,* $i = qn$ for some integer q. Thus all integers are trivially divisors of 0.

The integers that have integer inverses, namely ± 1, are called the *units* of $\mathbb{Z}$. If u is a unit and n is a divisor of i, then un is a divisor of i and n is a divisor of ui. Thus the factorization of an integer can only be unique up to a unit u, and ui has the same divisors as i. We therefore consider only factorizations of positive integers into products of positive integers.

Every nonzero integer i is divisible by 1 and i; these divisors are called trivial. An integer n is said to be a *factor* of an integer i if n is positive and a nontrivial divisor of i. For example, 1 has no nontrivial divisors and thus no factors.

A positive integer that has no nontrivial divisors is called a *prime integer*.

7.2.2 Mod-n arithmetic

Given a positive integer n, every integer i may be uniquely expressed as $i = qn + r$ for some integer remainder r in the interval $0 \le r \le n - 1$ and some integer quotient q. This may be proved by the Euclidean division algorithm, which if $i \ge n$ just subtracts n from i repeatedly until the remainder lies in the desired interval.

The remainder r, denoted by $r = i \bmod n$, is the more important part of this expression. The set of possible mod-n remainders is the set of n integers $R_n = \{0, 1, \ldots, n-1\}$. Evidently n is a divisor of i if and only if $i \bmod n = 0$.

Remainder arithmetic using the mod-n remainder set R_n is called "mod-n arithmetic." The rules for mod-n arithmetic follow from the rules for integer arithmetic as follows. Let $r = i \bmod n$ and $s = j \bmod n$; then, as integers, $r = i - qn$ and $s = j - tn$ for some quotients q and t. Then

$$\begin{aligned} r + s &= i + j - (q + t)n; \\ rs &= ij - (qj + ti)n + qtn^2. \end{aligned}$$

Hence $(r + s) \bmod n = (i + j) \bmod n$ and $rs \bmod n = ij \bmod n$; *i.e.,* the mod-n remainder of the sum or product of two integers is equal to the mod-n remainder of the sum or product of their mod-n remainders, as integers.

The mod-n addition and multiplication rules are therefore defined as follows:

$$\begin{aligned} r \oplus s &= (r+s) \bmod n; \\ r * s &= (rs) \bmod n, \end{aligned}$$

where "r" and "s" denote elements of the remainder set R_n on the left and the corresponding ordinary integers on the right. This makes mod-n arithmetic consistent with ordinary integer arithmetic in the sense expressed in the previous paragraph.

7.2.3 Unique factorization

Given a positive integer i, we may factor i into a unique product of prime factors by simply factoring out primes no greater than i until we arrive at the quotient 1, as the reader has known since grade school. For the time being, we will take this unique factorization property as given. A proof will be given as an exercise after we prove the corresponding property for polynomials.

7.3 Groups

We now introduce groups.

Definition 7.1 *A group is a set of elements $G = \{a, b, c, \ldots\}$ and an operation $\oplus$ for which the following axioms hold:*

- *Closure: for any $a \in G$, $b \in G$, the element $a \oplus b$ is in G.*
- *Associative law: for any $a, b, c \in G$, $(a \oplus b) \oplus c = a \oplus (b \oplus c)$.*
- *Identity: There is an identity element 0 in G for which $a \oplus 0 = 0 \oplus a = a$ for all $a \in G$.*
- *Inverse: For each $a \in G$, there is an inverse $(-a)$ such that $a \oplus (-a) = 0$.*

In general it is not necessary that $a \oplus b = b \oplus a$. A group G for which $a \oplus b = b \oplus a$ for all $a, b \in G$ is called *abelian* or *commutative*. In these notes all groups will be abelian.

In view of the associative law, we may write $(a \oplus b) \oplus c$ as $a \oplus b \oplus c$ without ambiguity. Moreover, in an abelian group the elements a, b, c may be written in any order.

Frequently, the operation in a group is called multiplication, usually represented either by $*$ or juxtaposition. The identity is then denoted by 1 (or e) and the inverse of a by a^{-1}. Additive notation is generally used only for abelian groups, whereas multiplicative notation is used for both abelian and nonabelian groups. Since we consider only abelian groups, we will use additive notation when the nature of the group is unspecified.

As an example, the set of integers $\mathbb{Z}$ with the usual addition operation $+$ forms an abelian group. Also, the real field $\mathbb{R}$ forms an additive abelian group under ordinary addition in which the identity is 0 and the inverse of a is $-a$. More interestingly, as the reader should verify, the nonzero elements of $\mathbb{R}$ form a multiplicative abelian group under ordinary multiplication, in which the identity is 1 and the inverse of a is $a^{-1} = 1/a$. We will see that every field has similar additive and multiplicative group properties.

This example illustrates that the group structure (*i.e.,* the properties stemming from the group operation $\oplus$) may reflect only part of the structure of the given set of elements; *e.g.,* the additive group structure of $\mathbb{R}$ takes no account of the fact that real numbers may also be multiplied, and the multiplicative group structure of $\mathbb{R} - \{0\}$ takes no account of the fact that real numbers may also be added.

We abbreviate $b \oplus (-a)$ for any $a, b \in G$ by $b - a$ and regard "$-$" as an additional operation implicitly defined by the axioms. In an additive group, "$-$" is called subtraction; in a multiplicative group, "$-$" is called division and denoted by / or $\div$.

Because of the inverse operation, cancellation is always permissible; *i.e.,* if $x \oplus a = y \oplus a$, we can add $-a$ to both sides, showing that $x = y$. Similarly, one can move terms from one side of an equation to the other; *i.e.,* $x \oplus a = y$ implies $x = y - a$.

Exercise 1 (Inverses and cancellation)

(a) Verify the following set of implications for arbitrary elements a, b of a group G which is not necessarily abelian:

$$b \oplus a = 0 \;\Rightarrow\; b = -a \;\Rightarrow\; a \oplus b = 0 \;\Rightarrow\; a = -b \;\Rightarrow\; b \oplus a = 0.$$

(b) Use this result to show that the inverse is unique, *i.e.,* that $a \oplus b = 0 \;\Rightarrow\; b = -a$, and that the inverse also works on the left, *i.e.,* $b \oplus a = 0 \;\Rightarrow\; b = -a$. Note that this shows that cancellation is permitted on either the right or the left.

(c) Show that the identity element is unique, *i.e.,* that for $a, b \in G$, $a \oplus b = a \;\Rightarrow\; b = 0$ and $b \oplus a = a \;\Rightarrow\; b = 0$. □

If G has a finite number of elements, $G = \{a_1, a_2, \ldots, a_n\}$, then G is said to be *finite* and $|G| = n$ is said to be the *order* of G. The group operation $\oplus$ may then be specified by an $n \times n$ "addition table" whose entry at row i, column j is $a_i \oplus a_j$. The cancellation property implies that if $a_j \neq a_k$, then $a_i \oplus a_j \neq a_i \oplus a_k$. This means that all elements in any row i of the addition table are distinct; *i.e.,* each row contains each element of G exactly once. Similarly, each column contains each element of G exactly once. Thus the group axioms restrict the group operation $\oplus$ more than might be immediately evident.

7.3.1 Alternative group axioms

The property that a "row of the addition table," namely $a \oplus G = \{a \oplus b \mid b \in G\}$ is just the set of elements of G in a different order (*i.e.,* a *permutation* of G) is a fundamental property of any group G. We will now show that this permutation property may be taken as one of the group axioms. Subsequently we will use this property to prove that certain sets are groups.

Theorem 7.1 (Alternative group axioms) *Let $G = \{a, b, c, \ldots\}$ be a set of elements on which an operation $\oplus$ is defined. Then G is a group under the operation $\oplus$ if and only if the following axioms hold:*

- *Associative law: for any $a, b, c \in G$, $(a \oplus b) \oplus c = a \oplus (b \oplus c)$.*
- *Identity: There is an identity element 0 in G for which $a \oplus 0 = 0 \oplus a = a$ for all $a \in G$.*
- *Permutation property: For each $a \in G$, $a \oplus G = \{a \oplus b \mid b \in G\}$ is a permutation of G.*

Proof. ($\Rightarrow$) If G is a group under $\oplus$, then by the closure property every element $a \oplus b$ is in G. Moreover, the fact that $a \in G$ has an inverse $-a \in G$ implies that every element $b \in G$ may be written as $a \oplus (-a \oplus b) \in a \oplus G$, so every element of G is in $a \oplus G$. Finally, from the cancellation property, $a \oplus b = a \oplus c$ implies $b = c$. Thus the correspondence between G and $a \oplus G$ defined by $b \leftrightarrow a \oplus b$ is one-to-one; *i.e.,* a permutation.

($\Leftarrow$) Conversely, if $a \oplus G$ is a permutation of G for every $a \in G$, then (a) the closure property holds; *i.e.,* $a \oplus b \in G$ for all $a, b \in G$; (b) since $0 \in a \oplus G$, there must exist a unique $b \in G$ such that $a \oplus b = 0$, so a has a unique inverse $-a = b$ under $\oplus$. Thus G is a group under $\oplus$. □

The properties of "rows" $a \oplus G$ hold equally for "columns" $G \oplus a$, even when G is nonabelian.

For example, the set $\mathbb{R}^*$ of nonzero elements of the real field $\mathbb{R}$ form an abelian group under real multiplication, because real multiplication is associative and commutative with identity 1, and $\alpha\mathbb{R}^*$ is a permutation of $\mathbb{R}^*$ for any $\alpha \in \mathbb{R}^*$.

Exercise 2 (Invertible subsets).

(a) Let H be a set of elements on which an associative operation $\oplus$ is defined with identity 0, and let G be the subset of elements $h \in H$ which have unique inverses $-h$ such that $h \oplus -h = 0$. Show that G is a group under $\oplus$.

(b) Show that the nonzero elements of the complex field form a group under complex multiplication.

(c) Show that the set of invertible $n \times n$ real matrices forms a (nonabelian) group under real matrix multiplication.

(d) What are the invertible elements of $\mathbb{Z}$ under multiplication? Do they form a group?

7.3.2 Cyclic groups

An important example of a finite abelian group is the set of remainders $R_n = \{0, 1, \ldots, n-1\}$ under mod-n addition, where n is a given positive integer. This group is called "the integers mod n" and is denoted by $\mathbb{Z}_n$. Note that $\mathbb{Z}_1$ is the trivial group $\{0\}$.

A *finite cyclic group* is a finite group G with a particular element $g \in G$, called the *generator*, such that each element of G can be expressed as the sum, $g \oplus \cdots \oplus g$, of some number of repetitions of g.[1] Thus each element of G appears in the sequence of elements $\{g, g \oplus g, g \oplus g \oplus g, \ldots\}$. We denote such an i-fold sum by ig, where i is a positive integer and g is a group element; *i.e.,*

$$1g = g, 2g = g \oplus g, \ldots, ig = \underbrace{g \oplus \cdots \oplus g}_{i \text{ terms}}, \ldots$$

Since g generates G and G includes the identity element 0, we must have $ig = 0$ for some positive integer i. Let n be the smallest such integer; thus $ng = 0$ and $ig \neq 0$ for $1 \leq i \leq n-1$. Adding the sum of j g's for any $j > 0$ to each side of $ig \neq 0$ results in $(i+j)g \neq jg$. Thus the elements $\{1g, 2g, \ldots, ng = 0\}$ must all be different.

[1] Mathematicians say also that an infinite group $G = \{\ldots, -1g, 0g, 1g, 2g, \ldots\}$ generated by a single element g is cyclic; *e.g.,* the group of integers $\mathbb{Z}$ is an infinite cyclic group with generator 1. Although such infinite cyclic groups have the single-generator property of finite cyclic groups, they do not "cycle." Hereafter, "cyclic group" will mean "finite cyclic group."

We can also add jg to both sides of the equality $ng = 0$, yielding $(j+n)g = jg$ for any $j > 0$. Thus for each $i > n$, ig is equal to some earlier element in the sequence, namely $(i-n)g$. The elements $\{1g, 2g, \ldots, ng = 0\}$ therefore constitute all of the distinct elements in G, and the order of G is $|G| = n$. If we define $0g$ to be the identity 0, then the elements of G may be conveniently represented as $G = \{0g = 0, 1g, \ldots, (n-1)g\}$.

Figure 1 illustrates the cyclic structure of G that arises from the relation $(j+n)g = jg$.

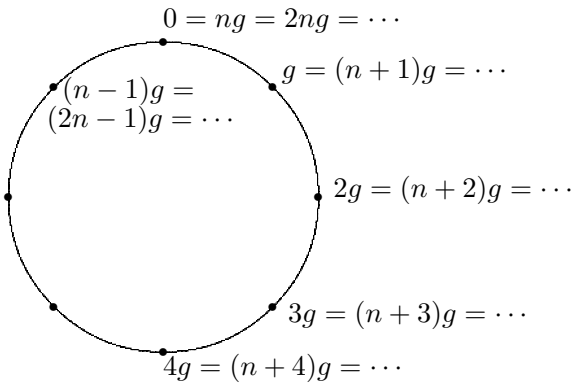


Figure 1. The cyclic structure of a cyclic group: the sequence $\{1g, 2g, \ldots\}$ goes from the group element g up to $ng = 0$, then returns to g and continues to cycle.

Addition in a cyclic group of order n can be understood in terms of mod-n addition. In particular, since $ng = 0$, we also have $2ng = 0, 3ng = 0$, etc. Since any integer i may be uniquely written as $i = qn + r$ where the remainder $r = i \bmod n$ is in the set $R_n = \{0, 1, \ldots, n-1\}$, we have $ig = (qn)g + rg = rg$, where $rg = (i \bmod n)g$ is one of the elements of G. The addition rule of G is thus as follows: for each $0 \le i, j < n$,

$$ig \oplus jg = (i + j \bmod n)g.$$

Evidently $0g$ is the identity, and the inverse of a nonzero element ig is $(n-i)g$.

We thus see that any cyclic group G of order n is essentially identical to $\mathbb{Z}_n$. More precisely, the correspondence $ig \in G \leftrightarrow i \in \mathbb{Z}_n$ is preserved under addition; *i.e.,* $ig \oplus jg \leftrightarrow i \oplus j$ for each $i, j \in \mathbb{Z}_n$. This type of correspondence is called an *isomorphism.* Specifically, two finite groups G and H are *isomorphic* if there exists an invertible[2] function $h : G \to H$ mapping each $\alpha \in G$ into a $\beta = h(\alpha) \in H$ such that $h(\alpha \oplus \alpha') = h(\alpha) \oplus h(\alpha')$, where $\oplus$ denotes the group operation of G on the left and that of H on the right. In summary:

Theorem 7.2 (Cyclic groups) *The elements of a cyclic group G of order n with generator g are $\{0g, 1g, 2g, \ldots, (n-1)g\}$. The addition rule is $ig \oplus jg = (i + j \bmod n)g$, the identity is $0g$, and the inverse of $ig \neq 0g$ is $(n-i)g$. Finally, G is isomorphic to $\mathbb{Z}_n$ under the one-to-one correspondence $ig \leftrightarrow i$.*

Since $\mathbb{Z}_n$ is abelian, it follows that all cyclic groups are abelian.

In multiplicative notation, the elements of a cyclic group G of order n with generator g are denoted by $\{g^0 = 1, g^1, g^2, \ldots, g^{n-1}\}$, the multiplication rule is $g^i * g^j = g^{(i+j \bmod n)}$, the identity is $g^0 = 1$, and the inverse of $g^i \neq 1$ is g^{n-i}. For example, if $\omega = e^{2\pi i/n}$, the set $\{1, \omega, \omega^2, \ldots, \omega^{n-1}\}$ of complex nth roots of unity is a cyclic group under complex multiplication, isomorphic to $\mathbb{Z}_n$.

[2] A function $h : G \to H$ is called invertible if for each $\beta \in H$ there is a unique $\alpha \in G$ such that $\beta = h(\alpha)$. An invertible function is also called a one-to-one correspondence, denoted by $G \leftrightarrow H$.

7.3.3 Subgroups

A subgroup S of a group G is a subset of the elements of the group such that if $a, b \in S$, then $a \oplus b \in S$ and $-a \in S$. A subgroup S therefore includes the identity element of G and the inverse of each element in S. The associative law holds for S since it holds for G. Therefore a subgroup $S \subseteq G$ is itself a group under the group operation of G.

For example, the set of integers $\mathbb{Z}$ is a subgroup of the additive group of $\mathbb{R}$.

If G is abelian, then S must be abelian; however, S may be abelian even if G is nonabelian.

For any $g \in G$, we define the coset (translate) $S \oplus g = \{s \oplus g \mid s \in S\}$. The zero coset $S \oplus 0$ is thus equal to S itself; moreover, by Theorem 7.1, $S \oplus g = S$ whenever $g \in S$.

The following theorem states a more general result:

Lemma 7.3 (Cosets) *Two cosets $S \oplus g$ and $S \oplus h$ are the same if $g - h \in S$, but are disjoint if $g - h \notin S$.*

Proof. If $g - h \in S$, then the elements of $S \oplus h$ include $(g - h) \oplus h = g$ and therefore all elements of $S \oplus g$, so $S \oplus g \subseteq S \oplus h$; similarly $S \oplus h \subseteq S \oplus g$.

On the other hand, if $S \oplus g$ and $S \oplus h$ have any element in common, say $s \oplus g = s' \oplus h$, then $g - h = s' - s \in S$; thus, $g - h \notin S$ implies that $S \oplus g$ and $S \oplus h$ are disjoint. □

It follows that the distinct cosets $S \oplus g$ of a subgroup $S \subseteq G$ form a disjoint partition of G, since every element $g \in G$ lies in some coset, namely $S \oplus g$.

The elements $s \oplus g$ of a coset $S \oplus g$ are all distinct, since $s \oplus g = s' \oplus g$ implies $s = s'$. Therefore if S is finite, then all cosets of S have the same size, namely the size $|S|$ of $S = S \oplus 0$. If G is finite, G is therefore the disjoint union of a finite number $|C|$ of cosets of $S \subseteq G$, each of size $|S|$, so $|G| = |C||S|$. This proves Lagrange's theorem:

Theorem 7.4 (Lagrange) *If S is a subgroup of a finite group G, then $|S|$ divides $|G|$.*

7.3.4 Cyclic subgroups

Given any finite group G and any element $g \in G$, the set of elements generated by g, namely $S(g) = \{g, g \oplus g, \ldots\}$, is a cyclic subgroup of G. The *order* of g is defined as the order $|S(g)|$ of $S(g)$. By Lagrange's theorem, $|S(g)|$ divides $|G|$, and by the cyclic groups theorem, $S(g)$ is isomorphic to $\mathbb{Z}_{|S(g)|}$. (If $g = 0$, then $S(g) = \{0\}$ and $|S(g)| = 1$. We will assume $g \neq 0$.)

As a fundamental example, let G be the cyclic group $\mathbb{Z}_n = \{0, 1, \ldots, n-1\}$, and let $S(m)$ be the cyclic subgroup $\{m, 2m, \ldots\}$ generated by $m \in \mathbb{Z}_n$. Here $im = m \oplus \cdots \oplus m$ is simply the sum of m with itself i times; *i.e.,* $im \in G$ is the ordinary product im mod n. The order $|S(m)|$ of $S(m)$ is the least positive integer k such that $km = 0 \bmod n$; *i.e.,* such that the integer product km is divisible by n. Thus km is the least common multiple of m and n, denoted $\mathrm{lcm}(m, n)$, and $|S(m)| = k = \mathrm{lcm}(m, n)/m$. By elementary number theory, $\mathrm{lcm}(m, n) = mn/\gcd(m, n)$ for any positive integers m, n, so we may alternatively write $|S(m)| = n/\gcd(m, n)$, where $\gcd(m, n)$ denotes the greatest common divisor of m and n. This shows explicitly that $|S(m)|$ divides n.

For example, suppose $n = 10$ and $m = 4$. Then $S(4) = \{4, 8, 2, 6, 0\}$. Thus $|S(4)| = 5$, consistent with $|S(4)| = \mathrm{lcm}(4, 10)/4 = 20/4$ or $|S(4)| = 10/\gcd(4, 10)/4 = 10/2$.

Now when does $S(m) = \mathbb{Z}_n$? This occurs if and only if $\gcd(m, n) = 1$; *i.e.,* if and only if m is relatively prime to n. In short, m generates $\mathbb{Z}_n$ and has order $|S(m)| = n$ if and only if m and n are relatively prime. The number of integers in the set $\{0, 1, \ldots, n-1\}$ that have order n is called the *Euler number* $\phi(n)$.

For example, in $\mathbb{Z}_{10}$ the integers that are relatively prime to 10 are $\{1, 3, 7, 9\}$, so $\phi(10) = 4$. The order of the other elements of $\mathbb{Z}_{10}$ are as follows:

- 0 is the only element of order 1, and $S(0) = \{0\}$.
- 5 is the only element of order 2, and $S(5) = \{0, 5\}$.
- $\{2, 4, 6, 8\}$ have order 5, and $S(2) = S(4) = S(6) = S(8) = \{0, 2, 4, 6, 8\}$.

In general, $\mathbb{Z}_n$ has a cyclic subgroup S_d of order d for each positive integer d that divides n, including 1 and n. S_d consists of $\{0, n/d, 2n/d, \ldots, (d-1)n/d\}$, and is isomorphic to $\mathbb{Z}_d$. S_d thus contains $\phi(d)$ elements that are relatively prime to d, each of which has order d and generates S_d. The remaining elements of S_d belong also to smaller cyclic subgroups.

For example, $\mathbb{Z}_{10}$ has a subgroup $S_5 = \{0, 2, 4, 6, 8\}$ with 5 elements. Four of these elements, namely $\{2, 4, 6, 8\}$, are relatively prime to 5 and generate S_5. The remaining element of S_5, namely 0, has order 1.

Since every element of $\mathbb{Z}_n$ has some definite order d that divides n, we have

$$n = \sum_{d:\, d|n} \phi(d). \tag{7.1}$$

The notation $d : d|n$ means the set of positive integers d, including 1 and n, that divide n. All Euler numbers may be determined recursively from this expression. For example, $\phi(1) = 1, \phi(2) = 2 - \phi(1) = 1, \phi(3) = 3 - \phi(1) = 2, \phi(4) = 4 - \phi(1) - \phi(2) = 2, \ldots$.

Exercise 3. Show that $\phi(n) \geq 1$ for all $n \geq 1$. [Hint: Find the order of 1 in $\mathbb{Z}_n$.] □

Since every cyclic group G of size n is isomorphic to $\mathbb{Z}_n$, these results apply to every cyclic group. In particular, every cyclic group G of size n has $\phi(n)$ generators that generate G, which are called the *primitive elements* of G. G also contains one cyclic subgroup of size d for each d that divides n.

Exercise 4. Show that every subgroup of $\mathbb{Z}_n$ is cyclic. [Hint: Let s be the smallest nonzero element in a subgroup $S \subseteq \mathbb{Z}_n$, and compare S to the subgroup generated by s.] □

7.4 Fields

Definition 7.2 *A field is a set $\mathbb{F}$ of at least two elements, with two operations $\oplus$ and $*$, for which the following axioms are satisfied:*

- *The set $\mathbb{F}$ forms an abelian group (whose identity is called 0) under the operation $\oplus$.*
- *The set $\mathbb{F}^* = \mathbb{F} - \{0\} = \{a \in \mathbb{F}, a \neq 0\}$ forms an abelian group (whose identity is called 1) under the operation $*$.*
- *Distributive law: For all $a, b, c \in \mathbb{F}$, $(a \oplus b) * c = (a * c) \oplus (b * c)$.*

The operation $\oplus$ is called addition (and often denoted by $+$), and the operation $*$ is called multiplication (and often denoted by juxtaposition). As in ordinary arithmetic, we often omit the parentheses around a product of elements, using the convention "multiplication before addition;" *e.g.,* we interpret $a \oplus b * c$ as $a \oplus (b * c)$.

The reader may verify that $\mathbb{R}$, $\mathbb{C}$, $\mathbb{Q}$ and $\mathbb{F}_2$ each form a field according to this definition under conventional addition and multiplication.

Exercise 5. Show that for any element $a \in \mathbb{F}$, $a * 0 = 0$. □

7.4.1 Prime fields

A fundamental example of a finite (Galois) field is the set $\mathbb{F}_p$ of mod-p remainders, where p is a given prime number. Here, as in $\mathbb{Z}_p$, the set of elements is $R_p = \{0, 1, \cdots, p-1\}$, and the operation $\oplus$ is mod-p addition. The multiplicative operation $*$ is mod-p multiplication; *i.e.,* multiply integers as usual and then take the remainder after division by p.

Theorem 7.5 (Prime fields) *For every prime p, the set $R_p = \{0, 1, \cdots, p-1\}$ forms a field (denoted by $\mathbb{F}_p$) under mod-p addition and multiplication.*

Proof. We have already seen that the elements of $\mathbb{F}_p$ form an abelian group under addition modulo p, namely the cyclic group $\mathbb{Z}_p$.

The associative and commutative properties of multiplication mod p follow from the corresponding properties of ordinary multiplication; the distributive law follows from the corresponding property for ordinary addition and multiplication. The multiplicative identity is 1.

To see that the nonzero elements $\mathbb{F}_p^* = \mathbb{F}_p - \{0\}$ form a group under multiplication, we use Theorem 7.1. By unique factorization, the product of two nonzero integers $a, b < p$ cannot equal 0 mod p. Therefore the nonzero elements $\mathbb{F}_p^*$ are closed under multiplication mod p. Also, for $a, b, c \in \mathbb{F}_p^*$ and $b \neq c$ we have $a(b-c) \bmod p \neq 0$. Thus $ab \neq ac \bmod p$, which implies $a * b \neq a * c$. Consequently there are no zeroes or repetitions in the set of $p-1$ elements $\{a*1, a*2, \ldots, a*(p-1)\}$, which means they must be a permutation of $\mathbb{F}_p^*$. □

We next show that $\mathbb{F}_p$ is essentially the only field with p elements. More precisely, we show that all fields with p elements are isomorphic. Two fields $\mathbb{F}$ and $\mathbb{G}$ are isomorphic if there is an invertible function $h : \mathbb{F} \to \mathbb{G}$ mapping each $\alpha \in \mathbb{F}$ into a $\beta = h(\alpha) \in \mathbb{G}$ such that $h(\alpha \oplus \alpha') = h(\alpha) \oplus h(\alpha')$ and $h(\alpha * \alpha') = h(\alpha) * h(\alpha')$. Less formally, $\mathbb{F}$ and $\mathbb{G}$ are isomorphic if there is a one-to-one correspondence $\mathbb{F} \leftrightarrow \mathbb{G}$ that translates the addition and multiplication tables of $\mathbb{F}$ to those of $\mathbb{G}$ and *vice versa.*

Let $\mathbb{F}$ be any field with a prime p number of elements. By the field axioms, $\mathbb{F}$ has an additive identity 0 and multiplicative identity 1. Consider the additive cyclic subgroup generated by 1, namely $S(1) = \{1, 1 \oplus 1, \ldots\}$. By Lagrange's theorem, the order of $S(1)$ divides $|\mathbb{F}| = p$, and therefore must be equal to 1 or p. But $1 \oplus 1 \neq 1$, else $1 = 0$, so 1 must have order p. In other words, $S(1) = \mathbb{F}$, and the additive group of $\mathbb{F}$ is isomorphic to that of $\mathbb{Z}_p$. We may therefore denote the elements of $\mathbb{F}$ by $\{0, 1, 2, \ldots, p-1\}$, and use mod-p addition as the addition rule.

The only remaining question is whether this correspondence $\mathbb{F} \leftrightarrow \mathbb{Z}_p$ under addition extends to multiplication. The distributive law shows that it does: $j * i$ is the sum of j terms each equal to i, so $j * i = (ji \bmod p)$. Therefore, in summary:

Theorem 7.6 (Prime field uniqueness) *Every field $\mathbb{F}$ with a prime number p of elements is isomorphic to $\mathbb{F}_p$ via the correspondence* $\underbrace{1 \oplus \cdots \oplus 1}_{i \text{ terms}} \in \mathbb{F} \leftrightarrow i \in \mathbb{F}_p$.

In view of this elementary isomorphism, we will denote any field with a prime number p of elements by $\mathbb{F}_p$.

It is important to note that the set $\mathbb{Z}_n$ of integers mod n does not form a field if n is not prime. The reason is that $n = ab$ for some positive integers $a, b < n \in \mathbb{Z}_n$; thus $ab = 0 \bmod n$, so the set of nonzero elements of $\mathbb{Z}_n$ is not closed under multiplication mod n.

However, we will see shortly that there do exist finite fields with non-prime numbers of elements that use other rules for addition and multiplication.

7.4.2 The prime subfield of a finite field

A subfield $\mathbb{G}$ of a field $\mathbb{F}$ is a subset of the field that is itself a field under the operations of $\mathbb{F}$. For example, the real field $\mathbb{R}$ is a subfield of the complex field $\mathbb{C}$. We now show that every finite field $\mathbb{F}_q$ has a subfield that is isomorphic to a prime field $\mathbb{F}_p$.

Let $\mathbb{F}_q$ be a finite field with q elements. By the field axioms, $\mathbb{F}_q$ has an additive identity 0 and a multiplicative identity 1.

Consider the cyclic subgroup of the additive group of $\mathbb{F}_q$ that is generated by 1, namely $S(1) = \{1, 1 \oplus 1, \ldots\}$. Let $n = |S(1)|$. By the cyclic group theorem, $S(1)$ is isomorphic to $\mathbb{Z}_n$, and its elements may be denoted by $\{0, 1, 2, \ldots, n-1\}$, with mod-n addition.

By the distributive law in $\mathbb{F}_q$, the product $i * j$ (in $\mathbb{F}_q$) of two nonzero elements in $S(1)$ is simply the sum of ij ones, which is an element of $S(1)$, namely $ij \bmod n$. Since this is a product of nonzero elements of $\mathbb{F}_q$, by the field axioms $ij \bmod n$ must be nonzero for all nonzero i, j. This will be true if and only if n is a prime number p.

Thus $S(1)$ forms a subfield of $\mathbb{F}_q$ with a prime number p of elements. By the prime field theorem of the previous subsection, $S(1)$ is isomorphic to $\mathbb{F}_p$. Thus the elements of $S(1)$, which are called the *integers* of $\mathbb{F}_q$, may be denoted by $\mathbb{F}_p = \{0, 1, \ldots, p-1\}$, and the addition and multiplication rules of $\mathbb{F}_q$ reduce to mod-p addition and multiplication in $\mathbb{F}_p$.

The prime p is called the *characteristic* of $\mathbb{F}_q$. Since the p-fold sum of the identity 1 with itself is 0, the p-fold sum of every field element $\beta \in \mathbb{F}_q$ with itself is 0: $p\beta = 0$.

In summary:

Theorem 7.7 (Prime subfields) *The integers $\{1, 1 \oplus 1, \ldots\}$ of any finite field $\mathbb{F}_q$ form a subfield $\mathbb{F}_p \subseteq \mathbb{F}_q$ with a prime number p of elements, where p is the characteristic of $\mathbb{F}_q$.*

7.5 Polynomials

We now consider polynomials over $\mathbb{F}_p$, namely polynomials whose coefficients lie in $\mathbb{F}_p$ and for which polynomial addition and multiplication is performed in $\mathbb{F}_p$. We will see that the factorization properties of polynomials are similar to those of the integers, and that the analogue to mod-n arithmetic is arithmetic modulo a polynomial $f(x)$.

A nonzero polynomial $f(x)$ of degree m over a field $\mathbb{F}$ is an expression of the form

$$f(x) = f_0 + f_1 x + f_2 x^2 + \cdots + f_m x^m,$$

where $f_i \in \mathbb{F}$, $0 \le i \le m$, and $f_m \neq 0$. We say that $\deg f(x) = m$. The symbol x represents an indeterminate (or "placeholder"), *not* an element of $\mathbb{F}$; *i.e.,* two polynomials are different if and only if their coefficients are different[3]. The nonzero polynomials of degree 0 are simply the nonzero field elements $f_0 \in \mathbb{F}$. There is also a special *zero polynomial* $f(x) = 0$ whose degree is defined by convention as $\deg 0 = -\infty$; we will explain the reason for this convention shortly. The set of all polynomials over $\mathbb{F}$ in an indeterminate x is denoted by $\mathbb{F}[x]$.

The rules for adding, subtracting or multiplying polynomials are the same over a general field $\mathbb{F}$ as over the real field $\mathbb{R}$, except that coefficient operations are in $\mathbb{F}$. In particular, addition and subtraction are performed componentwise. For multiplication, the coefficients of a polynomial product $f(x) = h(x)g(x)$ are determined by convolution:

$$f_i = \sum_{j=0}^{i} h_j g_{i-j}.$$

If two nonzero polynomials are multiplied, then their degrees add; *i.e.,* $\deg(h(x)g(x)) = \deg h(x) + \deg g(x)$. The convention $\deg 0 = -\infty$ ensures that this formula continues to hold when $h(x)$ or $g(x)$ is the zero polynomial.

The set $\mathbb{F}[x]$ has many of the properties of a field. It is evidently an abelian group under addition whose identity is the zero polynomial $0 \in \mathbb{F}[x]$. It is closed under multiplication, which is both associative and commutative and which distributes over addition. It has a multiplicative identity $1 \in \mathbb{F}[x]$, and the cancellation law holds.

However, in general we cannot divide evenly by a nonzero polynomial, since a polynomial $f(x)$ with $\deg f(x) > 0$ has no multiplicative inverse. Therefore $\mathbb{F}[x]$ is a *ring,*[4] not a field, like the ring of integers $\mathbb{Z}$. We now develop a series of properties of $\mathbb{F}[x]$ that resemble those of $\mathbb{Z}$.

[3] Over the real field $\mathbb{R}$, a polynomial $f(x)$ is sometimes regarded as a function $f : \mathbb{R} \to \mathbb{R}$. This alternative viewpoint makes little difference in the real case, since two polynomials over $\mathbb{R}$ are different if and only if the corresponding polynomial functions are different. However, over finite fields it is important to maintain the distinction. For example, over $\mathbb{F}_2$ the polynomial functions x and x^2 both map $0 \to 0, 1 \to 1$, yet the polynomials x and x^2 are different.

[4] The axioms of a ring are similar to those for a field, except that there is no multiplicative inverse. For example, $\mathbb{Z}$ and $\mathbb{Z}_n$ (for n not a prime) are rings. In fact, $\mathbb{Z}$ and $\mathbb{F}[x]$ are integer domains, which are the nicest kind of rings. An integer domain is a ring with commutative multiplication and a multiplicative identity 1 such that the nonzero elements are closed under multiplication.

Exercise 6. Show that an integer domain with a finite number of elements must be a finite field. [Hint: consider its cyclic multiplicative subgroups.] □

7.5.1 Definitions

A polynomial $g(x)$ is said to be a *divisor* of an polynomial $f(x)$ if $f(x)$ is a polynomial multiple of $g(x)$; *i.e.,* $f(x) = q(x)g(x)$ for some polynomial $q(x)$. Thus all polynomials are trivially divisors of the zero polynomial 0.

The polynomials that have polynomial inverses are the nonzero degree-0 polynomials $\beta \in \mathbb{F}^* = \mathbb{F} - \{0\}$. These are called the *units* of $\mathbb{F}[x]$. If $u(x)$ is a unit polynomial and $g(x)$ is a divisor of $f(x)$, then $u(x)g(x)$ is a divisor of $f(x)$ and $g(x)$ is a divisor of $u(x)f(x)$. Thus the factorization of a polynomial can be unique only up to a unit polynomial $u(x)$, and $u(x)f(x)$ has the same divisors as $f(x)$.

A monic polynomial is a nonzero polynomial $f(x)$ of degree m with high-order coefficient f_m equal to 1; *i.e.,* $f(x) = f_0 + f_1x + f_2x^2 + \cdots + x^m$. Every nonzero polynomial $g(x)$ may be written as the product $g(x) = g_m f(x)$ of a monic polynomial $f(x)$ of the same degree with a unit polynomial $u(x) = g_m$, and the product of two monic polynomials is monic. We may therefore consider only factorizations of monic polynomials into products of monic polynomials.

Every nonzero polynomial $f(x)$ is divisible by 1 and $f(x)$; these divisors are called trivial. A polynomial $g(x)$ is said to be a *factor* of a polynomial $f(x)$ if $g(x)$ is monic and a nontrivial divisor of $f(x)$. Thus the degree of any factor $g(x)$ of $f(x)$ satisfies $1 \leq \deg g(x) < \deg f(x)$.

A polynomial $g(x)$ of degree 1 or more that has no factors is called an *irreducible polynomial*, and a monic irreducible polynomial is called a *prime polynomial*. Our goal now is to show that every monic polynomial has a unique factorization into prime polynomial factors.

7.5.2 Mod-$g(x)$ arithmetic

Given a monic polynomial $g(x)$ of degree m, every polynomial $f(x)$ may be expressed as $f(x) = q(x)g(x)+r(x)$ for some polynomial remainder $r(x)$ such that $\deg r(x) < m$ and some polynomial quotient $q(x)$. This may be proved by the Euclidean long division algorithm of high school, with component operations in $\mathbb{F}$; *i.e.,* divide $g(x)$ into $f(x)$ by long division, high-degree terms first, stopping when the degree of the remainder is less than that of $g(x)$. The following exercise shows that the resulting quotient $q(x)$ and remainder $r(x)$ are unique.

Exercise 7 (Euclidean division algorithm).

(a) For the set $\mathbb{F}[x]$ of polynomials over any field $\mathbb{F}$, show that the distributive law holds: $(f_1(x) + f_2(x))h(x) = f_1(x)h(x) + f_2(x)h(x)$.

(b) Use the distributive law to show that for any given $f(x)$ and $g(x)$ in $\mathbb{F}[x]$, there is a unique $q(x)$ and $r(x)$ with $\deg r(x) < \deg g(x)$ such that $f(x) = q(x)g(x) + r(x)$. □

The remainder polynomial $r(x)$, denoted by $r(x) = f(x) \bmod g(x)$, is the more important part of this decomposition. The set of all possible remainder polynomials is the set $R_{\mathbb{F},m} = \{r_0 + r_1x + \cdots + r_{m-1}x^{m-1} \mid r_j \in \mathbb{F}, 0 \leq j \leq m-1\}$, whose size is $|R_{\mathbb{F},m}| = |\mathbb{F}|^m$. Evidently $g(x)$ is a divisor of $f(x)$ if and only if $f(x) \bmod g(x) = 0$.

Remainder arithmetic using the remainder set $R_{\mathbb{F},m}$ is called "mod-$g(x)$ arithmetic." The rules for mod-$g(x)$ arithmetic follow from the rules for polynomial arithmetic as follows. Let $r(x) = f(x) \bmod g(x)$ and $s(x) = h(x) \bmod g(x)$; then, as polynomials, $r(x) = f(x) - q(x)g(x)$ and $s(x) = h(x) - t(x)g(x)$ for some quotient polynomials $q(x)$ and $t(x)$. Then

$$\begin{aligned} f(x) + h(x) &= r(x) + s(x) - (q(x) + t(x))g(x); \\ f(x)h(x) &= r(x)s(x) - (q(x)s(x) + t(x)r(x))g(x) + q(x)t(x)g^2(x). \end{aligned}$$

Hence $(f(x) + h(x)) \bmod g(x) = (r(x) + s(x)) \bmod g(x)$ and $f(x)h(x) \bmod g(x) = r(x)s(x) \bmod g(x)$. In other words, the mod-$g(x)$ remainder of the sum or product of two polynomials is equal to the mod-$g(x)$ remainder of the sum or product of their mod-$g(x)$ remainders.

The mod-$g(x)$ addition and multiplication rules are therefore defined as follows:

$$\begin{aligned} r(x) \oplus s(x) &= (r(x) + s(x)) \bmod g(x); \\ r(x) * s(x) &= (r(x)s(x)) \bmod g(x), \end{aligned}$$

where "$r(x)$" and "$s(x)$" denote elements of the remainder set $R_{\mathbb{F},m}$ on the left and the corresponding ordinary polynomials on the right. This makes mod-$g(x)$ arithmetic consistent with ordinary polynomial arithmetic in the sense of the previous paragraph.

Note that the mod-$g(x)$ addition rule is just componentwise addition of coefficients in $\mathbb{F}$. In this sense the additive groups of $R_{\mathbb{F},m}$ and of the vector space $\mathbb{F}^m$ of m-tuples over $\mathbb{F}$ are isomorphic.

7.5.3 Unique factorization

By definition, every monic polynomial $f(x)$ is either irreducible or can be factored into a product of monic polynomial factors, each of lower degree. In turn, if a factor is not irreducible, it can be factored further. Since factor degrees are decreasing but bounded below by 1, we must eventually arrive at a product of monic irreducible (prime) polynomials. The following theorem shows that there is only one such set of prime polynomial factors, regardless of the order in which the polynomial is factored.

Theorem 7.8 (Unique factorization of polynomials) *Over any field $\mathbb{F}$, every monic polynomial $f(x) \in \mathbb{F}[x]$ of degree $m \geq 1$ may be written in the form*

$$f(x) = \prod_{i=1}^{k} a_i(x),$$

where each $a_i(x), 1 \leq i \leq k$, is a prime polynomial in $\mathbb{F}[x]$. This factorization is unique, up to the order of the factors.

Proof. We have already shown that $f(x)$ may be factored in this way, so we need only prove uniqueness. Thus assume hypothetically that the theorem is false and let m be the smallest degree such that there exists a degree-m monic polynomial $f(x)$ with more than one such factorization,

$$f(x) = a_1(x) \cdots a_k(x) = b_1(x) \cdots b_j(x); \qquad j, k \geq 1, \tag{7.2}$$

where $a_1(x), \ldots, a_k(x)$ and $b_1(x), \ldots, b_j(x)$ are prime polynomials. We will show that this implies a polynomial $f'(x)$ with degree less than m with non-unique factorization, and this contradiction will prove the theorem. Now $a_1(x)$ cannot appear on the right side of (7.2), else it could be factored out for an immediate contradiction. Similarly, $b_1(x)$ cannot appear on the left. Without loss of generality, assume $\deg b_1(x) \leq \deg a_1(x)$. By the Euclidean division algorithm, $a_1(x) = q(x)b_1(x) + r(x)$. Since $a_1(x)$ is irreducible, $r(x) \neq 0$ and $0 \leq \deg r(x) < \deg b_1(x) \leq \deg a_1(x)$.

Thus $r(x)$ has a prime factorization $r(x) = \beta r_1(x) \cdots r_n(x)$, where β is the high-order coefficient of $r(x)$, and $b_1(x)$ is not a divisor of any of the $r_i(x)$, since it has greater degree. Substituting into (7.2), we have

$$(q(x)b_1(x) + \beta r_1(x) \cdots r_n(x))a_2(x) \cdots a_k(x) = b_1(x) \cdots b_j(x),$$

or, defining $f'(x) = r_1(x) \cdots r_n(x)a_2(x) \cdots a_k(x)$ and rearranging terms,

$$f'(x) = r_1(x) \cdots r_n(x)a_2(x) \cdots a_k(x) = \beta^{-1}b_1(x)(b_2(x) \cdots b_j(x) - q(x)a_2(x) \cdots a_k(x)).$$

Now $f'(x)$ is monic, because it is a product of monic polynomials; it has degree less than $f(x)$, since $\deg r(x) < \deg a_1(x)$; and it has two different factorizations, with $b_1(x)$ a factor in one but not a divisor of any of the factors in the other; contradiction. □

Exercise 8. Following this proof, prove unique factorization for the integers $\mathbb{Z}$. □

7.5.4 Enumerating prime polynomials

The prime polynomials in $\mathbb{F}[x]$ are analogous to the prime numbers in $\mathbb{Z}$. One way to enumerate the prime polynomials is to use an analogue of the sieve of Eratosthenes. For integers, this method goes as follows: Start with a list of all integers greater than 1. The first integer on the list is 2, which is prime. Erase all multiples of 2 (even integers). The next remaining integer is 3, which must be the next prime. Erase all multiples of 3. The next remaining integer is 5, which must be the next prime. Erase all multiples of 5. And so forth.

Similarly, to find the prime polynomials in $\mathbb{F}_2[x]$, for example, first list all polynomials of degree 1 or more in $\mathbb{F}_2[x]$ in order of degree. (Note that all nonzero polynomials in $\mathbb{F}_2[x]$ are monic.) No degree-1 polynomial can have a factor, so the two degree-1 polynomials, x and $x + 1$, are both prime. Next, erase all degree-2 multiples of x and $x + 1$, namely

$$\begin{aligned} x^2 &= x * x; \\ x^2 + x &= x * (x + 1); \\ x^2 + 1 &= (x + 1) * (x + 1) \end{aligned}$$

from the list of four degree-2 polynomials. This leaves one prime degree-2 polynomial, namely $x^2 + x + 1$. Next, erase all degree-3 multiples of x, $x + 1$, and $x^2 + x + 1$ from the list of eight degree-3 polynomials, namely the six polynomials

$$\begin{aligned} x^3 &= x * x * x; \\ x^3 + x^2 &= (x + 1) * x * x; \\ x^3 + x &= (x + 1) * (x + 1) * x; \\ x^3 + x^2 + x &= x * (x^2 + x + 1); \\ x^3 + 1 &= (x + 1) * (x^2 + x + 1); \\ x^3 + x^2 + x + 1 &= (x + 1) * (x + 1) * (x + 1). \end{aligned}$$

The remaining two polynomials, namely $x^3 + x^2 + 1$ and $x^3 + x + 1$, must therefore be prime.

Exercise 9. Find all prime polynomials in $\mathbb{F}_2[x]$ of degrees 4 and 5. [Hint: There are three prime polynomials in $\mathbb{F}_2[x]$ of degree 4 and six of degree 5.] □

Continuing in this way, we may list all prime polynomials in $\mathbb{F}_2[x]$ up to any desired degree.

It turns out that the number $N(m)$ of prime polynomials of $\mathbb{F}_2[x]$ of degree m is $N(m) = 2, 1, 2, 3, 6, 9, 18, 30, 56, 99, \ldots$ for $m = 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, \ldots$. (In Section 7.9 we will give a simpler method to compute $N(m)$, and will show that $N(m) > 0$ for all m.)

A similar sieve algorithm may be used to find the prime polynomials in $\mathbb{F}[x]$ over any finite field $\mathbb{F}$. The algorithm starts with a listing of the monic polynomials ordered by degree, and successively erases the multiples of lower-degree prime polynomials.

7.6 A construction of a field with p^m elements

We now show how to construct a field with p^m elements for any prime integer p and positive integer $m \geq 1$. Its elements will be the set $R_{\mathbb{F},m}$ of remainder polynomials of degree less than m, and multiplication will be defined modulo an irreducible polynomial $g(x)$ of degree m. We will subsequently show that that every finite field is isomorphic to a finite field that is constructed in this way.

The construction assumes the existence of a prime polynomial $g(x) \in \mathbb{F}_p[x]$ of degree m. The proof that such a polynomial exists for all prime p and $m \geq 1$ will be deferred until later. The field that we construct will be denoted by $\mathbb{F}_{g(x)}$.

The set of elements of $\mathbb{F}_{g(x)}$ will be taken to be the mod-$g(x)$ remainder set $R_{\mathbb{F}_p,m} = \{r_0 + r_1 x + \cdots + r_{m-1}x^{m-1} \mid r_j \in \mathbb{F}_p, 0 \leq j \leq m-1\}$, whose size is $|R_{\mathbb{F}_p,m}| = p^m$.

The addition and multiplication rules will be taken to be those of mod-$g(x)$ arithmetic. We must show that the axioms of a field are satisfied with these definitions.

The associative, commutative and distributive laws for mod-$g(x)$ arithmetic follow from the corresponding laws for ordinary polynomial arithmetic.

Mod-$g(x)$ addition of two remainder polynomials in $\mathbb{F}_{g(x)}$ yields a remainder polynomial of degree $< m$ in $\mathbb{F}_{g(x)}$. $\mathbb{F}_{g(x)}$ evidently forms an abelian group under mod-$g(x)$ addition. (As already mentioned, this group is isomorphic to the additive group of $(\mathbb{F}_p)^m$.)

Mod-$g(x)$ multiplication of two remainder polynomials $r(x), s(x)$ yields the remainder polynomial $t(x) = r(x)s(x) \bmod g(x)$. The following exercise shows that the nonzero elements of $\mathbb{F}_{g(x)}$ form an abelian group under mod-$g(x)$ multiplication:

Exercise 10. Let $g(x)$ be a prime polynomial of degree m, and let $r(x), s(x), t(x)$ be polynomials in $\mathbb{F}_{g(x)}$.

(a) Prove the distributive law, *i.e.*, $(r(x)+s(x))*t(x) = r(x)*t(x)+s(x)*t(x)$. [Hint: Express each product as a remainder using the Euclidean division algorithm.]

(b) For $r(x) \neq 0$, show that $r(x) * s(x) \neq r(x) * t(x)$ if $s(x) \neq t(x)$.

(c) For $r(x) \neq 0$, show that as $s(x)$ runs through all nonzero polynomials in $\mathbb{F}_{g(x)}$, the product $r(x) * s(x)$ also runs through all nonzero polynomials in $\mathbb{F}_{g(x)}$.

(d) Using part (c) and Theorem 7.1, show that the nonzero elements of $\mathbb{F}_{g(x)}$ form an abelian group under mod-$g(x)$ multiplication. □

Since we have verified the three field axioms, we have proved:

Theorem 7.9 (Construction of $\mathbb{F}_{g(x)}$) *If $g(x)$ is an prime polynomial of degree m over a prime field $\mathbb{F}_p$, then the set of remainder polynomials $R_{\mathbb{F}_p,m}$ with mod-$g(x)$ arithmetic forms a finite field $\mathbb{F}_{g(x)}$ with p^m elements.*

Example 1. Let us construct a finite field with $2^2 = 4$ elements using the prime degree-2 polynomial $g(x) = x^2 + x + 1 \in \mathbb{F}_2[x]$.

There are four remainder polynomials mod $x^2 + x + 1$, namely $\{0, 1, x, x + 1\}$. Addition is componentwise mod 2. For multiplication, note that $x*x = x+1$ since x^2 mod $(x^2+x+1) = x+1$. Also $x * x * x = x * (x + 1) = 1$ since x^3 mod $(x^2 + x + 1) = 1$. The three nonzero elements $\{1, x, x + 1\}$ thus form a cyclic group under mod-$g(x)$ multiplication, which verifies the second field axiom for this example.

The complete mod-$g(x)$ addition and multiplication tables are as follows:

$\oplus$	0	1	x	$x+1$
0	0	1	x	$x+1$
1	1	0	$x+1$	x
x	x	$x+1$	0	1
$x+1$	$x+1$	x	1	0

$*$	0	1	x	$x+1$
0	0	0	0	0
1	0	1	x	$x+1$
x	0	x	$x+1$	1
$1+x$	0	$x+1$	1	x

7.7 The multiplicative group of $\mathbb{F}_q^*$ is cyclic

In this section we consider an arbitrary finite field $\mathbb{F}_q$ with q elements. By the second field axiom, the set $\mathbb{F}_q^*$ of all $q - 1$ nonzero elements must form a finite abelian group under multiplication. In this section we will show that this group is actually cyclic.

We start by showing that every element of $\mathbb{F}_q^*$ is a root of the polynomial $x^{q-1} - 1 \in \mathbb{F}_q[x]$. Thus we first need to discuss roots of polynomials over arbitrary fields.

7.7.1 Roots of polynomials

Let $\mathbb{F}[x]$ be the set of polynomials over an arbitrary field $\mathbb{F}$. If $f(x) \in \mathbb{F}[x]$ has a degree-1 factor $x - \alpha$ for some $\alpha \in \mathbb{F}$, then α is called a *root* of $f(x)$.

Since any $f(x)$ may be uniquely expressed as $f(x) = q(x)(x-\alpha)+\beta$ for some quotient $q(x)$ and some $\beta \in \mathbb{F}$ (*i.e.,* for some remainder $r(x) = \beta$ of degree less than 1), it follows that $f(\alpha) = \beta$. Therefore α is a root of $f(x)$ if and only if $f(\alpha) = 0$— *i.e.,* if and only if α is a root of the polynomial equation $f(x) = 0$.

By degree additivity, the degree of a polynomial $f(x)$ is equal to the sum of the degrees of its prime factors, which are unique by unique factorization. Therefore a polynomial of degree m can have at most m degree-1 factors. This yields what is sometimes called the fundamental theorem of algebra:

Theorem 7.10 (Fundamental theorem of algebra) *Over any field $\mathbb{F}$, a monic polynomial $f(x) \in \mathbb{F}[x]$ of degree m can have no more than m roots in $\mathbb{F}$. If it does have m roots $\{\beta_1, \ldots, \beta_m\}$, then the unique factorization of $f(x)$ is $f(x) = (x - \beta_1) \cdots (x - \beta_m)$.* □

Since the polynomial $x^n - 1$ can have at most n roots in $\mathbb{F}$, we have an important corollary:

Theorem 7.11 (Cyclic multiplicative subgroups) *In any field $\mathbb{F}$, the multiplicative group $\mathbb{F}^*$ of nonzero elements has at most one cyclic subgroup of any given order n. If such a subgroup exists, then its elements $\{1, \beta, \ldots, \beta^{n-1}\}$ satisfy*

$$x^n - 1 = (x - 1)(x - \beta) \cdots (x - \beta^{n-1}). \qquad \square$$

For example, the complex multiplicative group $\mathbb{C}^*$ has precisely one cyclic subgroup of each finite size n, consisting of the n complex nth roots of unity. The real multiplicative group $\mathbb{R}^*$ has cyclic subgroups of size 1 ($\{1\}$) and 2 ($\{\pm 1\}$), but none of any larger size.

Exercise 11. For $1 \le j \le n$, the jth *elementary symmetric function* $\sigma_j(S)$ of a set S of n elements of a field $\mathbb{F}$ is the sum of all $\binom{n}{j}$ products of j distinct elements of S. In particular, $\sigma_1(S)$ is the sum of all elements of S, and $\sigma_n(S)$ is the product of all elements of S.

(a) Show that if $S = \{1, \beta, \ldots, \beta^{n-1}\}$ is a cyclic subgroup of $\mathbb{F}^*$, then $\sigma_j(S) = 0$ for $1 \le j \le n-1$ and $\sigma_n(S) = (-1)^{n+1}$. In particular,

$$\sum_{j=0}^{n-1} \beta^j = 0, \quad \text{if } n > 1; \qquad \prod_{j=0}^{n-1} \beta^j = (-1)^{n+1}.$$

Verify for $S = \{\pm 1, \pm i\}$ (the four complex 4th roots of unity).

(b) Prove that for any odd prime integer p,

$$(p-1)! = 1 \cdot 2 \cdot 3 \cdots (p-1) = -1 \bmod p.$$

Verify for $p = 3, 5$ and 7. □

7.7.2 Factoring $x^q - x$ over $\mathbb{F}_q$

For any $\beta \in \mathbb{F}_q^*$, consider the cyclic subgroup $S(\beta) = \{1, \beta, \beta^2, \beta^3, \ldots\}$ of $\mathbb{F}_q^*$ generated by β. The size $|S(\beta)|$ of this subgroup is called the multiplicative order of β.

By the cyclic group theorem, $\beta^{|S(\beta)|} = 1$, and by Lagrange's theorem, $|S(\beta)|$ must divide $|\mathbb{F}_q^*| = q - 1$. It follows that $\beta^{q-1} = 1$ for all $\beta \in \mathbb{F}_q^*$.

In other words, every $\beta \in \mathbb{F}_q^*$ is a root of the polynomial equation $x^{q-1} = 1$, or equivalently of the polynomial $x^{q-1} - 1 \in \mathbb{F}_q[x]$. By the polynomial roots theorem, $x^{q-1} - 1$ can have at most $q - 1$ roots in $\mathbb{F}_q$, so these are all the roots of $x^{q-1} - 1$. Thus $x^{q-1} - 1$ factors into the product of the degree-1 polynomials $x - \beta$ for all $\beta \in \mathbb{F}_q^*$. Moreover, since $0 \in \mathbb{F}_q$ is a root of the polynomial x and $x(x^{q-1} - 1) = x^q - x$, the polynomial $x^q - x$ factors into the product of the degree-1 polynomials $x - \beta$ for all $\beta \in \mathbb{F}_q$.

To summarize:

Theorem 7.12 *In a finite field $\mathbb{F}_q$ with q elements, every nonzero field element $\beta \in \mathbb{F}_q$ satisfies $\beta^{q-1} = 1$ and has a multiplicative order $|S(\beta)|$ that divides $q - 1$. The nonzero elements of $\mathbb{F}_q$ are the $q - 1$ distinct roots of the polynomial $x^{q-1} - 1 \in \mathbb{F}_q[x]$; i.e.,*

$$x^{q-1} - 1 = \prod_{\beta \in \mathbb{F}_q^*} (x - \beta). \tag{7.3}$$

The elements of $\mathbb{F}_q$ are the q distinct roots of the polynomial $x^q - x \in \mathbb{F}_q[x]$; i.e.,

$$x^q - x = \prod_{\beta \in \mathbb{F}_q} (x - \beta). \tag{7.4}$$

Exercise 12.

(a) Verify (7.3) for the prime field $\mathbb{F}_5$.

(b) Verify (7.3) for the field $\mathbb{F}_4$ that was constructed in Example 1. [Hint: use a symbol other than x for the indeterminate in (7.3).] □

7.7.3 Every finite field has a primitive element

A *primitive element* of a finite field $\mathbb{F}_q$ is an element α whose multiplicative order $|S(\alpha)|$ equals $q-1$. If α is a primitive element, then the cyclic group $\{1, \alpha, \alpha^2, \ldots, \alpha^{q-2}\}$ is a set of $q-1$ distinct nonzero elements of $\mathbb{F}_q$, which therefore must be all the nonzero elements. Thus if we can show that $\mathbb{F}_q$ has at least one primitive element, we will have shown that its nonzero elements $\mathbb{F}_q^*$ form a cyclic group under multiplication of size $q-1$.

By Lagrange's theorem, the multiplicative order $|S(\beta)|$ of each nonzero element $\beta \in \mathbb{F}_q^*$ divides $q-1$. Therefore the size d of each cyclic subgroup of $\mathbb{F}_q^*$ divides $q-1$. As we have seen, the number of elements in a cyclic group or subgroup of size d that have order d is the Euler number $\phi(d)$. Since by the cyclic subgroups theorem $\mathbb{F}_q^*$ has at most one cyclic subgroup of each size d, the number of elements in $\mathbb{F}_q^*$ with order less than $q-1$ is at most

$$\sum_{d:\, d|(q-1),\ d \neq q-1} \phi(d).$$

But since the Euler numbers satisfy the relationship (7.1), which in this case is

$$q-1 = \sum_{d:\, d|(q-1)} \phi(d),$$

we conclude that there must be at least $\phi(q-1)$ elements of $\mathbb{F}_q^*$ with order $q-1$. Indeed, since $\mathbb{F}_q^*$ has at most $\phi(q-1)$ elements of order $q-1$, all inequalities must be satisfied with equality; *i.e.,* $\mathbb{F}_q^*$ has precisely $\phi(d)$ elements of order d for each divisor d of $q-1$.

We saw in Exercise 3 that $\phi(q-1) \geq 1$, so a primitive element α of order $q-1$ exists. Thus $\mathbb{F}_q^*$ is cyclic and has one cyclic subgroup of each order d that divides $q-1$. This proves the following theorem:

Theorem 7.13 (Primitive elements) *Given any field $\mathbb{F}_q$ with q elements, the nonzero elements of $\mathbb{F}_q$ form a multiplicative cyclic group $\mathbb{F}_q^* = \{1, \alpha, \alpha^2, \ldots, \alpha^{q-2}\}$. Consequently $\mathbb{F}_q^*$ has $\phi(d) \geq 1$ elements of multiplicative order d for every d that divides $q-1$, and no elements of any other order. In particular, $\mathbb{F}_q^*$ has $\phi(q-1) \geq 1$ primitive elements.*

Henceforth we will usually write the elements of a finite field $\mathbb{F}_q$ as $\{0, 1, \alpha, \alpha^2, \ldots, \alpha^{q-2}\}$, where α denotes a primitive element. For $\mathbb{F}_{g(x)}$, denoting a field element β as a power of α rather than as a remainder polynomial helps to avoid confusion when we consider polynomials in β.

Example 2. The prime field $\mathbb{F}_5$ has $\phi(1) = 1$ element of order 1 (the element 1), $\phi(2) = 1$ element of order 2 (namely 4 = -1), and $\phi(4) = 2$ primitive elements of order 4 (namely, 2 and 3). We can therefore write $\mathbb{F}_5 = \{0, 1, 2, 2^2, 2^3\}$, since $2^2 = 4$ and $2^3 = 3 \bmod 5$. □

Example 3. A field $\mathbb{F}_{16} = \{0, 1, \alpha, \ldots, \alpha^{14}\}$ with 16 elements has

- $\phi(1) = 1$ element of order 1 (the element 1);
- $\phi(3) = 2$ elements of order 3 (α^5 and α^{10});
- $\phi(5) = 4$ elements of order 5 ($\alpha^3, \alpha^6, \alpha^9, \alpha^{12}$), and
- $\phi(15) = 8$ primitive elements of order 15 ($\alpha, \alpha^2, \alpha^4, \alpha^7, \alpha^8, \alpha^{11}, \alpha^{13}, \alpha^{14}$). □

The "logarithmic" representation of the nonzero elements of $\mathbb{F}_q$ as distinct powers of a primitive element α is obviously highly convenient for multiplication and division. Multiplication in $\mathbb{F}_q$ is often carried out by using such a "log table" to convert a polynomial $f(x) \in \mathbb{F}_q$ to the exponent i such that $f(x) = \alpha^i$, and then using an inverse "antilog table" to convert back after adding or subtracting exponents. (Note that the zero element can be included in this scheme if we define $0 = \alpha^{-\infty}$.)

7.8 Every finite field is isomorphic to a field $\mathbb{F}_{g(x)}$

We now wish to show that every finite field $\mathbb{F}_q$ is isomorphic to a field $\mathbb{F}_{g(x)}$ of the type that we have previously constructed. In particular, this will show that the number of elements of a finite field must be $q = p^m$, a prime power.

The development relies on the properties of minimal polynomials, which are the factors that appear in the unique factorization of $x^q - x$ over the prime subfield $\mathbb{F}_p$ of $\mathbb{F}_q$.

7.8.1 Factoring $x^q - x$ into minimal polynomials over $\mathbb{F}_p$

Again, consider any field $\mathbb{F}_q$ with q elements. We have seen in Theorem 7.12 that the polynomial $x^q - x \in \mathbb{F}_q[x]$ factors completely into q deqree-1 factors $x - \beta \in \mathbb{F}_q[x], \beta \in \mathbb{F}_q$.

We have also seen that if $\mathbb{F}_q$ has characteristic p, then $\mathbb{F}_q$ has a prime subfield $\mathbb{F}_p$ with p elements. The prime subfield $\mathbb{F}_p$ contains the integers of $\mathbb{F}_q$, which include $\{0, \pm 1\}$. Therefore we may regard $x^q - x$ alternatively as a polynomial in $\mathbb{F}_p[x]$.

By unique factorization, $x^q - x$ factors over $\mathbb{F}_p$ into a unique product of prime polynomials $g_i(x) \in \mathbb{F}_p[x]$:

$$x^q - x = \prod_i g_i(x). \tag{7.5}$$

Since each coefficient of $g_i(x)$ is an element of $\mathbb{F}_p \subseteq \mathbb{F}_q$, it is also an element of $\mathbb{F}_q$, so $g_i(x)$ is also a monic polynomial in $\mathbb{F}_q[x]$. We therefore have the following two factorizations of $x^q - x$ in $\mathbb{F}_q[x]$:

$$x^q - x = \prod_{\beta \in \mathbb{F}_q} (x - \beta) = \prod_i g_i(x). \tag{7.6}$$

Since the first factorization is the unique prime factorization, it follows that each monic polynomial $g_i(x)$ of degree greater than 1 must be reducible over $\mathbb{F}_q$, and must factor into a product of degree-1 monic polynomials; *i.e.,*

$$g_i(x) = \prod_{j=1}^{\deg g_i(x)} (x - \beta_{ij}). \tag{7.7}$$

The prime polynomials $g_i(x)$ are called the *minimal polynomials* of $\mathbb{F}_q$. Since each $\beta \in \mathbb{F}_q$ appears exactly once on the left side of (7.6), it also appears as a factor in exactly one minimal polynomial in (7.7). Thus the elements of $\mathbb{F}_q$ are partitioned into disjoint sets $\{\beta_{i1}, \ldots, \beta_{ik}\}$ where $k = \deg g_i(x)$, and each $\beta \in \mathbb{F}_q$ is a root of exactly one minimal polynomial of $\mathbb{F}_q$, called the minimal polynomial of β.

The key property of the minimal polynomial of β is the following:

Lemma 7.14 *Let $g(x)$ be the minimal polynomial of any given $\beta \in \mathbb{F}_q$. Then $g(x)$ is the monic polynomial of least degree in $\mathbb{F}_p[x]$ such that $g(\beta) = 0$. Moreover, for any $f(x) \in \mathbb{F}_p[x]$, $f(\beta) = 0$ if and only if $g(x)$ divides $f(x)$.*

Proof: Let $h(x) \in \mathbb{F}_p[x]$ be a monic polynomial of least degree such that $h(\beta) = 0$. Using the Euclidean division algorithm, $g(x) = q(x)h(x) + r(x)$ where $\deg r(x) < \deg h(x)$. Since $h(\beta) = g(\beta) = 0$, we must have $r(\beta) = 0$. By the smallest degree property of $h(x)$, this implies that $r(x) = 0$, so $h(x)$ divides $g(x)$. But since $g(x)$ is irreducible, $h(x)$ cannot have degree less than $g(x)$; *i.e.,* $\deg h(x) = \deg g(x)$. Moreover, since both $h(x)$ and $g(x)$ are monic, this implies that $h(x) = g(x)$. Thus $g(x)$ is the monic polynomial of least degree in $\mathbb{F}_p[x]$ such that $g(\beta) = 0$.

Now let $f(x)$ be any polynomial in $\mathbb{F}_p[x]$ that satisfies $f(\beta) = 0$. By Euclidean division, $f(x) = q(x)g(x) + r(x)$ with $\deg r(x) < \deg g(x)$. Thus $r(\beta) = f(\beta) = 0$. Since $\deg r(x) < \deg g(x)$, $r(\beta) = 0$ if and only if $r(x) = 0$; *i.e.,* if and only if $g(x)$ divides $f(x)$. □

Example 1 (cont.). Again consider the field $\mathbb{F}_4$ of Example 1, whose elements we now write as $\{0, 1, \alpha, \alpha^2\}$, where α may be taken as x or $x + 1$. This field has characteristic 2. The prime factorization of the binary polynomial $x^4 - x = x^4 + x \in \mathbb{F}_2[x]$ is

$$x^4 + x = x(x + 1)(x^2 + x + 1),$$

so the minimal polynomials of $\mathbb{F}_4$ are x, $x + 1$ and $x^2 + x + 1$. The elements 0 and $1 \in \mathbb{F}_4$ are the roots of x and $x + 1$, respectively. From (7.7), the other two elements of $\mathbb{F}_4$, namely α and α^2, must be roots of $x^2 + x + 1 \in \mathbb{F}_2[x]$. We verify that

$$x^2 + x + 1 = (x + \alpha)(x + \alpha^2)$$

since $\alpha + \alpha^2 = 1$ and $\alpha * \alpha^2 = \alpha^3 = 1$. □

7.8.2 Valuation maps, minimal polynomials and subfields

Given a field $\mathbb{F}_q$ with prime subfield $\mathbb{F}_p$, we now consider evaluating a nonzero polynomial $f(x) = \sum_i f_i x^i \in \mathbb{F}_p[x]$ at an element $\beta \in \mathbb{F}_q$ to give a value

$$f(\beta) = \sum_{i=0}^{\deg f(x)} f_i \beta^i$$

in $\mathbb{F}_q$, where f_i is taken as an element of $\mathbb{F}_q$ for the purposes of this evaluation. The value of the zero polynomial at any β is 0.

The value $f(\beta)$ depends on both the polynomial $f(x)$ and the field element $\beta \in \mathbb{F}_q$. Rather than regarding $f(\beta)$ as a function of β, as the notation suggests, we will regard $f(\beta)$ as a function of the polynomial $f(x) \in \mathbb{F}_p[x]$ for a fixed β. In other words, we consider the map $m_\beta : \mathbb{F}_p[x] \to \mathbb{F}_q$ that is defined by $m_\beta(f(x)) = f(\beta)$.

The set of values $m_\beta(\mathbb{F}_p[x])$ of this map as $f(x)$ ranges over polynomials in $\mathbb{F}_p[x]$ is by definition the subset of elements $\mathbb{G}_\beta \subseteq \mathbb{F}_q$ that can be expressed as linear combinations over $\mathbb{F}_p$ of powers of β. We will show that $\mathbb{G}_\beta$ forms a subfield of $\mathbb{F}_q$ that is isomorphic to the polynomial remainder field $\mathbb{F}_{g(x)}$, where $g(x)$ is the minimal polynomial of β, namely the monic polynomial of least degree such that $g(\beta) = 0$.

We observe that the map $m_\beta : \mathbb{F}_p[x] \to \mathbb{F}_q$ preserves addition and multiplication; *i.e.*, $m_\beta(f_1(x) + f_2(x)) = m_\beta(f_1(x)) + m_\beta(f_2(x))$ since both sides equal $f_1(\beta) + f_2(\beta)$, and $m_\beta(f_1(x)f_2(x)) = m_\beta(f_1(x))m_\beta(f_2(x))$ since both sides equal $f_1(\beta)f_2(\beta)$.

We can now prove the desired isomorphism between the fields $\mathbb{F}_{g(x)}$ and $\mathbb{G}_\beta$:

Theorem 7.15 (Subfields generated by $\beta \in \mathbb{F}_q$) *For any $\beta \in \mathbb{F}_q$, let $g(x)$ be the minimal polynomial of β. Then the set of all linear combinations $\mathbb{G}_\beta = \{f(\beta) = \sum_i f_i\beta^i, f(x) \in \mathbb{F}_p[x]\}$ over $\mathbb{F}_p$ of powers of β is equal to the set $\{r(\beta), r(x) \in R_{\mathbb{F}_p,m}\}$ of values of remainder polynomials $r(x) \in R_{\mathbb{F}_p,m}$, and $\mathbb{G}_\beta$ is a field which is isomorphic to the field $\mathbb{F}_{g(x)}$ under the correspondence $r(\beta) \in \mathbb{G}_\beta \leftrightarrow r(x) \in R_{\mathbb{F}_p,m}$.*

Proof. We first verify that the correspondence $m_\beta : R_{\mathbb{F}_p,m} \to \mathbb{G}_\beta$ is one-to-one (invertible). First, if $f(\beta)$ is any element of $\mathbb{G}_\beta$, then by Euclidean division we can write $f(x) = q(x)g(x)+r(x)$ where $r(x) \in R_{\mathbb{F}_p,m}$, and then $f(\beta) = q(\beta)g(\beta)+r(\beta) = r(\beta)$, so $f(\beta) = r(\beta)$ for some remainder polynomial $r(x)$. Thus $m_\beta(R_{\mathbb{F}_p,m}) = m_\beta(\mathbb{F}_p[x]) = \mathbb{G}_\beta$. On the other hand, no two remainder polynomials $r(x), s(x)$ with degrees less than m can evaluate to the same element of $\mathbb{G}_\beta$, because if $r(\beta) = s(\beta)$, then $r(x) - s(x)$ is a nonzero polynomial of degree less than $g(x)$ that evaluates to 0, contradiction.

Now, as we have already seen, $m_\beta(r(x) + s(x)) = m_\beta(r(x)) + m_\beta(s(x))$ and $m_\beta(r(x)s(x)) = m_\beta(r(x))m_\beta(s(x))$, which verifies that this correspondence is an isomorphism. □

We remark that $\mathbb{G}_\beta$ may be viewed as the smallest subfield of $\mathbb{F}_q$ containing the element β, because any subfield containing β must also contain all powers of β and all linear combinations of powers over $\mathbb{F}_p$.

7.8.3 Isomorphism theorems

We have shown that every finite field $\mathbb{F}_q$ contains a primitive element α. In this case, the subfield $\mathbb{G}_\alpha$ consisting of all linear combinations over $\mathbb{F}_p$ of powers of α must evidently be the whole field $\mathbb{F}_q$. Thus we obtain our main theorem:

Theorem 7.16 (Every finite field is isomorphic to a field $\mathbb{F}_{g(x)}$) *Every finite field $\mathbb{F}_q$ of characteristic p with q elements is isomorphic to a polynomial remainder field $\mathbb{F}_{g(x)}$, where $g(x)$ is a prime polynomial in $\mathbb{F}_p[x]$ of degree m. Hence $q = p^m$ for some positive integer m.*

Exercise 13. For which integers $q, 1 \le q \le 12$, does a finite field $\mathbb{F}_q$ exist? □

Finally, we wish to show that all fields with p^m elements are isomorphic. The following lemma shows that every prime polynomial $g(x)$ of degree m (we are still assuming that there exists at least one) is a minimal polynomial of every field with p^m elements:

Lemma 7.17 *Every prime polynomial $g(x) \in \mathbb{F}_p[x]$ of degree m divides $x^{p^m} - x$.*

Proof. If $g(x)$ is a prime polynomial in $\mathbb{F}_p[x]$ of degree m, then the set $R_{\mathbb{F}_p,m}$ with mod-$g(x)$ arithmetic forms a field $\mathbb{F}_{g(x)}$ with p^m elements. The remainder polynomial $x \in R_{\mathbb{F}_p,m}$ is a field element $\beta \in \mathbb{F}_{g(x)}$. Evidently $g(\beta) = 0$, but $r(\beta) \neq 0$ if $\deg r(x) < m$; therefore $g(x)$ is the minimal polynomial of β. Since $\beta^{p^m-1} = 1$, β is a root of $x^{p^m-1} - 1$. This implies that $g(x)$ divides $x^{p^m-1} - 1$, and thus also $x^{p^m} - x$. □

Consequently every field of size p^m includes m elements whose minimal polynomial is $g(x)$. Therefore by the same construction as above, we can prove:

Theorem 7.18 (All finite fields of the same size are isomorphic) *For any prime polynomial $g(x) \in \mathbb{F}_p[x]$ of degree m, every field of p^m elements is isomorphic to the polynomial remainder field $\mathbb{F}_{g(x)}$.*

7.8.4 More on the factorization of $x^{p^m} - x$

We can now obtain further information on the factorization of $x^q - x$. In view of Theorem 7.16, we now set $q = p^m$.

We first show that the set of roots of a minimal polynomial $g_i(x) \in \mathbb{F}_p[x]$ is closed under the operation of taking the pth power. This follows from the curious but important fact that over a field $\mathbb{F}$ of characteristic p, taking the pth power is a linear operation. For example, when $p = 2$, squaring is linear because

$$(\alpha + \beta)^2 = \alpha^2 + \alpha\beta + \alpha\beta + \beta^2 = \alpha^2 + \beta^2.$$

More generally, over any field $\mathbb{F}$,

$$(\alpha + \beta)^p = \sum_{j=0}^{p} \binom{p}{j} \alpha^j \beta^{p-j},$$

where $\binom{p}{j}\alpha^j\beta^{p-j}$ denotes the sum of $\binom{p}{j}$ terms equal to $\alpha^j\beta^{p-j}$. If $\mathbb{F}$ has characteristic p, then the integer $\binom{p}{j} = p!/(j!)((n-j)!)$ may be reduced mod p. Now $p!$ contains a factor of p, but for $1 \le j \le p-1$, $j!$ and $(n-j)!$ do not contain a factor of p. Therefore $\binom{p}{j} = 0 \bmod p$ for $1 \le j \le p-1$, and

$$(\alpha + \beta)^p = \alpha^p + \beta^p.$$

By taking the pth power n times, we may extend this result as follows:

Lemma 7.19 (Linearity of taking the p^nth power) *Over any field $\mathbb{F}$ of characteristic p, for any $n \ge 1$, taking the p^nth power is linear; i.e.,*

$$(\alpha + \beta)^{p^n} = \alpha^{p^n} + \beta^{p^n}.$$

Note that if $\mathbb{F}$ has $q = p^m$ elements, then $\beta^{p^m} = \beta$ for all $\beta \in \mathbb{F}$, so this lemma becomes repetitive for $n \ge m$.

Exercise 14. Using this lemma, prove that if $f(x) = \sum_{i=0}^{m} f_i x^i$, then

$$f^{p^n}(x) = (f_0 + f_1 x + f_2 x^2 + \cdots + f_m x^m)^{p^n} = f_0^{p^n} + f_1^{p^n} x^{p^n} + f_2^{p^n} x^{2p^n} + \cdots + f_m^{p^n} x^{mp^n}.$$

□

This result yields a useful test for whether a polynomial $f(x) \in \mathbb{F}[x]$ is in $\mathbb{F}_p[x]$ or not, and a useful formula in case it is:

Lemma 7.20 (Prime subfield polynomials) *For any field $\mathbb{F}$ of characteristic p and any $f(x) \in \mathbb{F}[x]$, $f^p(x) = f(x^p)$ if and only if $f(x) \in \mathbb{F}_p[x]$; i.e., if and only if all coefficients f_i are in the prime subfield $\mathbb{F}_p \subseteq \mathbb{F}$.*

Proof. By Exercise 14, we have

$$f^p(x) = (f_0 + f_1 x + f_2 x^2 + \cdots + f_n x^n)^p = f_0^p + f_1^p x^p + f_2^p x^{2p} + \cdots + f_n^p x^{np}.$$

Now the elements of $\mathbb{F}$ that are in $\mathbb{F}_p$ are precisely the p roots of the polynomial $x^p - x$; thus $\beta^p = \beta$ if and only if $\beta \in \mathbb{F}_p$. Thus the right side of this equation simplifies to $f(x^p)$ if and only if $f_i \in \mathbb{F}_p$ for all i. □

Exercise 15. Prove that a positive integer n is prime if and only if $(x - a)^n = x^n - a \bmod n$ for every integer a that is relatively prime to n.[5] □

Using Lemma 7.20, we now show that the roots of a minimal polynomial are a *cyclotomic coset* of the form $\{\beta, \beta^p, \beta^{p^2}, \ldots\}$:

Theorem 7.21 (Roots of minimal polynomials) *Let $g(x)$ be a minimal polynomial of a finite field $\mathbb{F}$ with p^m elements. Then the roots of $g(x)$ are a set of the form $\{\beta, \beta^p, \beta^{p^2}, \ldots, \beta^{p^{n-1}}\}$, where n is a divisor of m. Moreover, $g(x)$ divides $x^{p^n} - x$.*

Proof. Let β be any root of $g(x)$. Since $g(x) \in \mathbb{F}_p[x]$, Lemma 7.20 shows that $g(x^p) = g^p(x)$. Therefore $g(\beta^p) = g^p(\beta) = 0$. Thus β^p is also a root of $g(x)$. Iterating, $\beta^{p^2}, \beta^{p^3}, \ldots, \beta^{p^i}, \ldots$ are all roots of $g(x)$. Because $\mathbb{F}$ is finite, these roots cannot all be distinct. Therefore let n be the smallest integer such that $\beta^{p^n} = \beta$. Thus $\beta^{p^j} \neq \beta$ for $1 \leq j < n$. This implies that $\beta^{p^j} \neq \beta^{p^{j+k}}$ for $0 \leq j < n$, $1 \leq k < n$; *i.e.,* all elements of the set $\{\beta, \beta^p, \beta^{p^2}, \ldots, \beta^{p^{n-1}}\}$ are distinct. Thus $\beta, \beta^p, \beta^{p^2}, \ldots$ is a cyclic sequence and $\beta^{p^j} = \beta$ if and only if n is a divisor of j. Since $\beta^{p^m} = \beta$, we see that n must divide m.

Finally, we show that these roots are all of the roots of $g(x)$; *i.e.,* $\deg g(x) = n$ and

$$g(x) = \prod_{i=0}^{n-1} (x - \beta^{p^i}).$$

The right side of this equation is a monic polynomial $h(x) \in \mathbb{F}[x]$ of degree n. Since the roots of $h(x)$ are roots of $g(x)$, $h(x)$ must divide $g(x)$ in $\mathbb{F}[x]$. Now, using Lemma 7.20, we can prove that $h(x)$ is actually a polynomial in $\mathbb{F}_p[x]$, because

$$h^p(x) = \prod_{i=0}^{n-1} (x - \beta^{p^i})^p = \prod_{i=0}^{n-1} (x^p - \beta^{p^{i+1}}) = \prod_{i=0}^{n-1} (x^p - \beta^{p^i}) = h(x^p),$$

where we use the linearity of taking the pth power and the fact that $\beta^{p^n} = \beta$. Therefore, since $g(x)$ has no factors in $\mathbb{F}_p[x]$, $g(x)$ must actually be equal to $h(x)$.

Finally, since the roots of $g(x)$ all satisfy $\beta^{p^n} = \beta$, they are all roots of the polynomial $x^{p^n} - x$, which implies that $g(x)$ divides $x^{p^n} - x$. □

[5] This is the basis of the polynomial-time primality test of [Agrawal, Kayal and Saxena, 2002].

This theorem has some important implications. First, the degree n of a minimal polynomial $g(x)$ of a finite field $\mathbb{F}$ with p^m elements must be a divisor of m. Second, the subfield $\mathbb{G}_\beta$ of $\mathbb{F}$ generated by a root β of $g(x)$ must have p^n elements. Third, $x^{p^n} - x$ divides $x^{p^m} - x$, since the elements of $\mathbb{G}_\beta$ are all the roots of $x^{p^n} - x$ and are also roots of $x^{p^m} - x$.

Conversely, let $g(x)$ be any prime polynomial in $\mathbb{F}_p[x]$ of degree n. Then there is a finite field generated by $g(x)$ with p^n elements. This proves that $g(x)$ divides $x^{p^n} - x$, and thus $g(x)$ divides $x^{p^m} - x$ for every multiple m of n. Thus the divisors of $x^{p^m} - x$ include every prime polynomial in $\mathbb{F}_p[x]$ whose degree n divides m.

Moreover, $x^{p^m} - x$ has no repeated factors. We proved this earlier assuming the existence of a field $\mathbb{F}$ with p^m elements; however, we desire a proof that does not make this assumption. The following exercise yields such a proof.

Exercise 16 ($x^{p^m} - x$ has no repeated factors). The formal derivative of a degree-n polynomial $f(x) \in \mathbb{F}_p[x]$ is defined as

$$f'(x) = \sum_{j=1}^{n} (j \bmod p) f_j x^{j-1}$$

(a) Show that if $f(x) = g(x)h(x)$, then $f'(x) = g'(x)h(x) + g(x)h'(x)$.

(b) Show that an prime polynomial $g(x)$ is a repeated divisor of $f(x)$ if and only if $g(x)$ is a divisor of both $f(x)$ and $f'(x)$.

(c) Show that $x^{p^m} - x$ has no repeated prime factors over $\mathbb{F}_p$. □

Now we can conclude our discussion of the factorization of $x^{p^m} - x$ as follows:

Theorem 7.22 (Factors of $x^{p^m} - x$) *The polynomial $x^{p^m} - x$ factors over $\mathbb{F}_p$ into the product of the prime polynomials in $\mathbb{F}_p[x]$ whose degrees divide m, with no repetitions.* □

For example, over $\mathbb{F}_2$, we have

$$\begin{aligned}
x^2 + x &= x(x+1); \\
x^4 + x &= x(x+1)(x^2+x+1); \\
x^8 + x &= x(x+1)(x^3+x^2+1)(x^3+x+1); \\
x^{16} + x &= x(x+1)(x^2+x+1)(x^4+x^3+1)(x^4+x^3+x^2+x+1)(x^4+x+1).
\end{aligned}$$

Exercise 17. Find all prime polynomials $g(x) \in \mathbb{F}_3[x]$ of degree 1 and 2 over the ternary field $\mathbb{F}_3$. Show that the product of these polynomials is $x^9 - x = x^9 + 2x$. Explain, with reference to $\mathbb{F}_9$. □

7.9 Finite fields $\mathbb{F}_{p^m}$ exist for all prime p and $m \geq 1$

At last we can prove that for every prime p and positive integer m there exists a prime polynomial $g(x) \in \mathbb{F}_p[x]$ of degree m. This will prove the existence of a finite field $\mathbb{F}_{g(x)}$ with p^m elements.

Using the factorization of Theorem 7.22, we will show that there do not exist enough prime polynomials of degree less than m that their product could have degree p^m.

Let $N(n)$ denote the number of prime polynomials over $\mathbb{F}_p$ of degree n. The product of these polynomials has degree $nN(n)$. Since $x^{p^m} - x$ is the product of these polynomials for all divisors n of m, and there are no repeated factors, its degree p^m is equal to

$$p^m = \sum_{n:\, n|m} nN(n) \tag{7.8}$$

This formula may be solved recursively for each $N(m)$, starting with $N(1) = p$.

Exercise 18. Calculate $N(m)$ for $p = 2$ for $m = 1$ to 10. Check your results against those stated in Section 7.5.4. □

Now we are in a position to prove the desired theorem:

Theorem 7.23 (Existence of prime polynomials) *Let $N(m)$ be the number of prime polynomials in $\mathbb{F}_p[x]$ of degree m, which is given recursively by (7.8). For every prime p and positive integer m, $N(m) > 0$.*

Proof. Note first that $nN(n) \leq p^n$. Thus

$$p^m \;\leq\; mN(m) + \sum_{n<m:\, n|m} p^n \;\leq\; mN(m) + (m/2)p^{m/2},$$

where we have upperbounded the number of terms in the sum by $m/2$ and upperbounded each term by $p^{m/2}$, since the largest divisor of m other than m is at most $m/2$. Thus

$$mN(m) \geq p^m - (m/2)p^{m/2} = p^{m/2}(p^{m/2} - m/2).$$

The quantity $p^{m/2} - m/2$ is positive for $p = 2, m = 2$, and is increasing in both p and m. Thus $mN(m)$ is positive for all prime p and all $m \geq 2$. Moreover $N(1) = p$. □

Since a finite field $\mathbb{F}_{g(x)}$ with p^m elements can be constructed from any prime polynomial $g(x) \in \mathbb{F}_p[x]$ of degree m, this implies:

Theorem 7.24 (Existence of finite fields) *For every prime p and positive integer m, there exists a finite field with p^m elements.*

Moreover, for each n that divides m, there exists a unique subfield $\mathbb{G}$ with p^n elements, namely the roots of the polynomial $x^{p^n} - x$:

Theorem 7.25 (Existence of finite subfields) *Every finite field with p^m elements has a subfield with p^n elements for each positive integer n that divides m.*

In summary, the factorization of $x^{p^m} - x$ into minimal polynomials partitions the elements of $\mathbb{F}_{p^m}$ into cyclotomic cosets whose properties are determined by their minimal polynomials. The roots of $g(x)$ have multiplicative order k if $g(x)$ divides $x^k - 1$ and does not divide $x^j - 1$ for $j < k$. Moreover, the roots of $g(x)$ are elements of the subfield with p^n elements if and only if $g(x)$ divides $x^{p^n} - x$, or equivalently if their order k divides $p^n - 1$.

Example 3 (cont.) Over $\mathbb{F}_2$, the polynomial $x^{16} + x$ factors as follows:

$$x^{16} + x = x(x+1)(x^2+x+1)(x^4+x^3+1)(x^4+x^3+x^2+x+1)(x^4+x+1).$$

Moreover, $x^3+1 = (x+1)(x^2+x+1)$ and $x^5+1 = (x+1)(x^4+x^3+x^2+x+1)$. The primitive elements are thus the roots of x^4+x+1 and x^4+x^3+1. If we choose a root of x^4+x+1 as α, then $\mathbb{F}_{16} = \{0, 1, \alpha, \ldots, \alpha^{14}\}$ partitions into cyclotomic cosets as follows:

- One zero element (0), minimal polynomial x;
- One element of order 1 (1), minimal polynomial $x+1$;
- Two elements of order 3 (α^5, α^{10}), minimal polynomial x^2+x+1;
- Four elements of order 5 ($\alpha^3, \alpha^6, \alpha^9, \alpha^{12}$), minimal polynomial $x^4+x^3+x^2+x+1$;
- Four elements of order 15 ($\alpha, \alpha^2, \alpha^4, \alpha^8$), minimal polynomial x^4+x+1;
- Four elements of order 15 ($\alpha^7, \alpha^{14}, \alpha^{13}, \alpha^{11}$), minimal polynomial x^4+x^3+1.

$\mathbb{F}_{16}$ has a prime subfield $\mathbb{F}_2$ consisting of the elements whose minimal polynomials divide x^2+x, namely 0 and 1. It also has a subfield $\mathbb{F}_4$ consisting of the elements whose minimal polynomials divide x^4+x, namely $\{0, 1, \alpha^5, \alpha^{10}\}$. Alternatively, $\mathbb{F}_4^*$ consists of the three elements of $\mathbb{F}_{16}^*$ whose multiplicative orders divide 3. □

Exercise 19 (construction of $\mathbb{F}_{32}$).

(a) Find the prime polynomials in $\mathbb{F}_2[x]$ of degree 5, and determine which have primitive roots.

(b) For some minimal polynomial $g(x)$ with a primitive root α, construct a field $\mathbb{F}_{g(x)}$ with 32 elements. Give a table with the elements partitioned into cyclotomic cosets as above. Specify the minimal polynomial and the multiplicative order of each nonzero element. Identify the subfields of $\mathbb{F}_{g(x)}$.

(c) Show how to do multiplication and division in $\mathbb{F}_{g(x)}$ using this "log table." Discuss the rules for multiplication and division in $\mathbb{F}_{g(x)}$ when one of the field elements involved is the zero element $0 \in \mathbb{F}_{g(x)}$.

(d) [Optional] If you know something about maximum-length shift-register (MLSR) sequences, show that there exists a correspondence between the "log table" given above and a certain MLSR sequence of length 31. □

Chapter 8

Reed-Solomon codes

In the previous chapter we discussed the properties of finite fields, and showed that there exists an essentially unique finite field $\mathbb{F}_q$ with $q = p^m$ elements for any prime p and integer $m \geq 1$.

We now consider (n, k, d) linear codes over a finite field $\mathbb{F}_q$. The most useful such codes are the Reed-Solomon (RS) codes (1960), whose length is limited to $n \leq q$, but which have arbitrary dimension $k, 1 \leq k \leq n$, and which have a minimum distance of $d = n - k + 1$. This meets a certain fundamental bound called the Singleton bound. RS codes are easily decoded, and have a wide range of uses.

We also briefly mention binary Bose-Chaudhuri-Hocquenghem (BCH) codes (1959), which are binary derivatives of RS codes. The parameters (n, k, d) of BCH codes are sometimes slightly better than those of Reed-Muller (RM) codes.

8.1 Linear codes over finite fields

An (n, k) linear code $\mathcal{C}$ over a finite field $\mathbb{F}_q$ is a k-dimensional subspace of the vector space $\mathbb{F}_q^n$ of all n-tuples over $\mathbb{F}_q$. For $q = 2$, this reduces to our previous definition of binary linear codes.

A subset $\mathcal{C} \subseteq \mathbb{F}_q^n$ is a subspace if it is closed under componentwise addition (has the group property) as well as under multiplication by scalars (elements of $\mathbb{F}_q$). A linear code $\mathcal{C}$ thus necessarily contains the all-zero n-tuple $\mathbf{0}$.

Any k linearly independent codewords $(\mathbf{g}_1, \ldots, \mathbf{g}_k)$ generate $\mathcal{C}$, in the sense that

$$\mathcal{C} = \{\sum_{j=1}^{k} a_j \mathbf{g}_j, a_j \in \mathbb{F}_q, 1 \leq j \leq k\}.$$

Thus $\mathcal{C}$ has q^k distinct codewords.

Example 1. The following nine 4-tuples over $\mathbb{F}_3$ form a $(4, 2, 3)$ linear code over the ternary field $\mathbb{F}_3 = \{0, 1, 2\}$, with generators $\mathbf{g}_1 = (1110)$ and $\mathbf{g}_2 = (0121)$:

$$\mathcal{C} = \{0000, 1110, 2220, 0121, 1201, 2011, 0212, 1022, 2102\}.$$

By the group property, $\mathcal{C} = \mathcal{C} - \mathbf{c}$ for any codeword $\mathbf{c} \in \mathcal{C}$, so the set of Hamming distances between any codeword $\mathbf{c} \in \mathcal{C}$ and all other codewords is independent of $\mathbf{c}$. The minimum Hamming distance d between codewords in a linear code $\mathcal{C}$ is thus equal to the minimum Hamming

weight of any nonzero codeword (the minimum distance between **0** and any other codeword). An (n, k) linear code with minimum Hamming distance d is called an (n, k, d) linear code.

More generally, the group property shows that the number of codewords at Hamming distance w from any codeword $\mathbf{c} \in \mathcal{C}$ is the number N_w of codewords of Hamming weight w in $\mathcal{C}$.

Much of classical algebraic coding theory has been devoted to optimizing the parameters (n, k, d); *i.e.,* maximizing the size q^k of the code for a given length n, minimum distance d and field $\mathbb{F}_q$, or maximizing d for a given n, k and q. The practical motivation for this research has been to maximize the guaranteed error-correction power of the code. Because Hamming distance is a metric satisfying the triangle inequality, a code with Hamming distance d is guaranteed to correct t symbol errors whenever $2t < d$, or in fact to correct t errors and s erasures whenever $2t + s < d$. This elementary metric is not the whole story with regard to performance on an AWGN channel, nor does it take into account decoding complexity; however, it is a good first measure of code power.

8.2 The Singleton bound and MDS codes

The Singleton bound is the most fundamental bound on the parameters (n, k, d) over any field: $k + d \leq n + 1$. A code that meets this bound with equality is called "maximum distance separable" (MDS).

The only binary codes that are MDS are the trivial $(n, n, 1)$ universe codes, the $(n, n-1, 2)$ single-parity-check (SPC) codes, and the $(n, 1, n)$ repetition codes. However, over nonbinary fields, there exist much less trivial MDS codes, notably the Reed-Solomon codes and their close relations.

Let $\mathcal{C} \subseteq \mathcal{A}^n$ be any set of $|\mathcal{C}|$ n-tuples over any symbol alphabet $\mathcal{A}$ of size $q = |\mathcal{A}|$, and let the minimum Hamming distance between codewords in $\mathcal{C}$ be d. Then $\mathcal{C}$ should be able to correct any set of $s = d - 1$ erasures; *i.e.,* after puncturing $s = d - 1$ of the coordinates, the codewords must still all differ in the remaining $n - s = n - d + 1$ coordinates. Therefore there can be at most q^{n-d+1} codewords in $\mathcal{C}$:

Theorem 8.1 (Singleton bound) *A code $\mathcal{C}$ of length n with minimum Hamming distance d over an alphabet of size $q = |\mathcal{A}|$ can have at most $|\mathcal{C}| \leq q^{n-d+1}$ codewords. Equality holds if and only if the codewords run through all q^k possible k-tuples in every set of $k = n-d+1$ coordinates.*

Note that the Singleton bound holds also for nonlinear codes. Any code that meets the Singleton bound with equality is called an MDS code.

Given a (possibly nonlinear) code $\mathcal{C}$ with q^k codewords over an alphabet of size q, a set of k coordinates is called an *information set* of $\mathcal{C}$ if the codewords run through all q^k possible k-tuples in that set of coordinates; *i.e.,* if there is a unique codeword associated with every possible set of symbol values in that set of coordinates. In other words, we may freely specify the symbols in these coordinates, and then the remainder of the codeword is uniquely specified. The case of equality may therefore be restated as follows:

Corollary 8.2 (Information sets of MDS codes) *$\mathcal{C}$ is an MDS code if and only if every subset of $k = n - d + 1$ coordinates is an information set of $\mathcal{C}$.*

If $\mathcal{C}$ is a linear (n, k, d) code over a finite field $\mathbb{F}_q$, then its size is $|\mathcal{C}| = q^k$. Therefore:

Corollary 8.3 (Singleton bound for linear codes) *A linear (n, k, d) code $\mathcal{C}$ over any finite field $\mathbb{F}_q$ has $k \leq n-d+1$. Equality holds if and only if every set of k coordinates is an information set of $\mathcal{C}$.*

Equivalently, $d + k \leq n + 1$, or $d \leq n - k + 1$.

The conditions governing MDS codes are so stringent that the weight distribution of a linear (n, k, d) MDS code over $\mathbb{F}_q$ is completely determined by the parameters n, k, d and q. We will show how to derive N_d and N_{d+1}; the number of codewords N_w of higher weights w may be derived similarly.

Theorem 8.4 (N_d for MDS codes) *An (n, k, d) MDS code over $\mathbb{F}_q$ has $q - 1$ codewords that have weight d in every subset of d coordinates. The total number of codewords of weight d is thus*

$$N_d = \binom{n}{d}(q-1).$$

Proof. Given a subset of $d = n - k + 1$ coordinates, consider the information set consisting of the complementary $n - d = k - 1$ coordinates plus any one of the given coordinates. Fix the symbol values in the $k - 1$ complementary coordinates to 0, and let the symbol values in the remaining information coordinate run through all q symbols in $\mathbb{F}_q$. This must yield q different codewords with all zeroes in the $n-d$ complementary coordinates. One of these codewords must be the all-zero codeword $\mathbf{0}$. The remaining $q - 1$ codewords have weight at most d; but since the minimum nonzero weight is d, all must have weight d. The formula for N_d then follows from the fact that there are $\binom{n}{d}$ different subsets of d coordinates. □

Exercise 1. Show that the number of codewords of weight $d + 1$ in an (n, k, d) linear MDS code is

$$N_{d+1} = \binom{n}{d+1}\left((q^2-1) - \binom{d+1}{d}(q-1)\right),$$

where the first term in parentheses represents the number of codewords with weight $\geq d$ in any subset of $d+1$ coordinates, and the second term represents the number of codewords with weight equal to d. □

Exercise 2. Compute the number of codewords of weights 2 and 3 in an $(n, n-1, 2)$ SPC code over $\mathbb{F}_2$. Compute the number of codewords of weights 2 and 3 in an $(n, n-1, 2)$ linear code over $\mathbb{F}_3$. Compute the number of codewords of weights 3 and 4 in a $(4, 2, 3)$ linear code over $\mathbb{F}_3$. □

8.3 Reed-Solomon (RS) Codes

For any code parameters $n, k, d = n - k + 1$ $(1 \leq k \leq n)$ and finite field $\mathbb{F}_q$, there exists a linear (n, k, d) RS code over $\mathbb{F}_q$, as long as $n \leq q + 1$. Thus RS codes form a very large class of MDS codes. In fact, for any n, $1 < k < n - 1$, and q for which an MDS code is known to exist, there also exists an RS code (or doubly, or triply, extended RS code) with the same parameters.

Binary RS codes have length at most $n \leq q + 1 = 3$, and therefore are not very interesting. When we speak of RS codes, we are usually thinking of nonbinary RS codes over $\mathbb{F}_q$, where q may be relatively large.

8.3.1 RS codes as evaluation codes

The most natural definition of RS codes is for length $n = q$. The definition is in terms of a certain evaluation map from the set $(\mathbb{F}_q)^k$ of all k-tuples over $\mathbb{F}_q$ to the set $(\mathbb{F}_q)^n$ of n-tuples over $\mathbb{F}_q$.

Let the k information symbols be denoted by $(f_0, f_1, \ldots, f_{k-1})$, where $f_j \in \mathbb{F}_q, 0 \le j \le k-1$. Let $f(z) = f_0 + f_1 z + \cdots + f_{k-1} z^{k-1} \in \mathbb{F}_q[z]$ be the corresponding polynomial in the indeterminate z. (We use z for the indeterminate here to avoid confusion with the indeterminate x used in the polynomials representing the elements of $\mathbb{F}_q$.) Thus $f(z)$ is one of the q^k polynomials over $\mathbb{F}_q$ of degree less than k.

Let $\beta_1, \beta_2, \ldots \beta_q$ be the q different elements of $\mathbb{F}_q$ arranged in some arbitrary order. The most convenient arrangement is $\beta_1 = 0, \beta_2 = 1, \beta_3 = \alpha, \ldots, \beta_j = \alpha^{j-2}, \ldots, \beta_q = \alpha^{q-2} = \alpha^{-1}$, where α is a primitive element of $\mathbb{F}_q$.

The information polynomial $f(z)$ is then mapped into the n-tuple $(f(\beta_1), f(\beta_2), \ldots, f(\beta_q))$ over $\mathbb{F}_q$, whose components $f(\beta_i)$ are equal to the evaluations of the polynomial $f(z)$ at each field element $\beta_i \in \mathbb{F}_q$:

$$f(\beta_i) = \sum_{j=0}^{k-1} f_j \beta_i^j \in \mathbb{F}_q, \quad 1 \le i \le q,$$

where we use the convention $0^0 = 1$. Note that it is the polynomial $f(z)$ that varies from codeword to codeword; the "symbol locators" $0, 1, \beta_3 = \alpha, \ldots, \beta_q = \alpha^{-1}$ are fixed. The code generators may thus be taken as the polynomials $g_j(z) = z^j, 0 \le j \le k-1$, or as the n-tuples

$$\begin{aligned} \mathbf{g}_0 &= (1, 1, 1, \ldots, 1) \\ \mathbf{g}_1 &= (0, 1, \alpha, \ldots, \alpha^{-1}) \\ \mathbf{g}_2 &= (0, 1, \alpha^2, \ldots, \alpha^{-2}) \\ &\cdots \\ \mathbf{g}_{k-1} &= (0, 1, \alpha^{k-1}, \ldots, \alpha^{-(k-1)}) \end{aligned}$$

Theorem 8.5 (RS codes) *The q^k q-tuples generated by the mapping $f(z) \mapsto \{f(\beta_i), \beta_i \in \mathbb{F}_q\}$ as the polynomial $f(z)$ ranges over all q^k polynomials over $\mathbb{F}_q$ of degree less than k form a linear $(n = q, k, d = n - k + 1)$ MDS code over $\mathbb{F}_q$.*

Proof. The code is linear because the sum of the codewords corresponding to two polynomials $f_1(z)$ and $f_2(z)$ is the codeword corresponding to the polynomial $f_1(z) + f_2(z)$, and the multiple of the codeword corresponding to $f(z)$ by $\beta \in \mathbb{F}_q$ is the codeword corresponding to the polynomial $\beta f(z)$.

A codeword has a zero symbol in the coordinate corresponding to β_i if and only if $f(\beta_i) = 0$; *i.e.,* if and only if β_i is a root of the equation $f(z) = 0$. By the fundamental theorem of algebra (Theorem 7.9), if $f(z) \neq 0$, then since $\deg f(z) \le k-1$, this equation can have at most $k-1$ roots in $\mathbb{F}_q$. Therefore a nonzero codeword can have at most $k-1$ symbols equal to zero, so its weight is at least $n - k + 1$. Since the code is linear, this implies that its minimum distance is at least $d \ge n - k + 1$. But by the Singleton bound, $d \le n - k + 1$; thus $d = n - k + 1$. □

Example 2. Consider the quaternary field $\mathbb{F}_4$ as constructed in Chapter 7, using the irreducible polynomial $g(x) = x^2 + x + 1$ over $\mathbb{F}_2$. Let $\beta_1 = 0, \beta_2 = 1, \beta_3 = x, \beta_4 = x^2 = x + 1$. (This is the same ordering of the symbols as in the tables of Chapter 7.) The RS code with $n = q = 4$ and $k = 2$ is then given by the mapping

$$f(z) = f_0 + f_1 z \longrightarrow (f(\beta_1), f(\beta_2), f(\beta_3), f(\beta_4)).$$

More explicitly, the mapping is

$$(f_0, f_1) \longrightarrow ((f_0 + f_1\beta_1), (f_0 + f_1\beta_2), (f_0 + f_1\beta_3), (f_0 + f_1\beta_4)).$$

Since f_0 and f_1 can each take on arbitrary values from $\mathbb{F}_4$, the code has 16 codewords. The two generators of the code correspond to the two polynomials $g_0(z) = 1$ and $g_1(z) = z$, or equivalently the 4-tuples $(1, 1, 1, 1)$ and $(0, 1, x, x^2)$, respectively. The full coding map is

0,	0	$\longrightarrow$	0,	0,	0,	0		x,	0	$\longrightarrow$	x,	x,	x,	x
0,	1	$\longrightarrow$	0,	1,	x,	x^2		x,	1	$\longrightarrow$	x,	x^2,	0,	1
0,	x	$\longrightarrow$	0,	x,	x^2,	1		x,	x	$\longrightarrow$	x,	0,	1,	x^2
0,	x^2	$\longrightarrow$	0,	x^2,	1,	x		x,	x^2	$\longrightarrow$	x,	1,	x^2,	0
1,	0	$\longrightarrow$	1,	1,	1,	1		x^2,	0	$\longrightarrow$	x^2,	x^2,	x^2,	x^2
1,	1	$\longrightarrow$	1,	0,	x^2,	x		x^2,	1	$\longrightarrow$	x^2,	x,	1,	0
1,	x	$\longrightarrow$	1,	x^2,	x,	0		x^2,	x	$\longrightarrow$	x^2,	1,	0,	x
1,	x^2	$\longrightarrow$	1,	x,	0,	x^2		x^2,	x^2	$\longrightarrow$	x^2,	0,	x,	1

We see by inspection that the minimum nonzero weight of this $(4, 2)$ linear code is $d = 3$. Moreover, there are $N_3 = 12$ codewords of weight 3 and $N_4 = 3$ codewords of weight 4, in accord with the weight formulas of the previous section. □

The RS codes are naturally nested— *i.e.,* the $(n = q, k, d)$ code is a subcode of the $(n = q, k + 1, d - 1)$ code— since the polynomials of degree less than k are a subset of the polynomials of degree less than $k + 1$. The $(n = q, n, 1)$ RS code is the universe code consisting of all q^q q-tuples over $\mathbb{F}_q$, and the $(n = q, 1, n)$ code is the repetition code over $\mathbb{F}_q$, consisting of all q codewords of the form $(\beta, \beta, \ldots, \beta)$ for $\beta \in \mathbb{F}_q$.

An $(n = q, k, d = n - k + 1)$ RS code may be punctured in any $s \leq n - k = d - 1$ places to yield an $(n = q - s, k, d = n - k + 1 - s)$ MDS code. The minimum distance can be reduced by at most s by such puncturing, so $d \geq n - k + 1 - s$, and the code still has q^k distinct codewords; but then by the Singleton bound, $d \leq n - s - k - 1$. Thus by puncturing an RS code, we can create an $(n, k, d = n - k + 1)$ MDS code for any $n \leq q$ and $k, 1 \leq k \leq n$.

For historical reasons (see following subsections), an RS code of length $n = q$ is called an "extended RS code." Here is how to construct a "doubly extended RS code."

Exercise 3 (doubly extended RS codes). Consider the following map from $(\mathbb{F}_q)^k$ to $(\mathbb{F}_q)^{q+1}$. Let $(f_0, f_1, \ldots, f_{k-1})$ be any k-tuple over $\mathbb{F}_q$, and define the polynomial $f(z) = f_0 + f_1 z + \cdots + f_{k_1} z^{k-1}$ of degree less than k. Map $(f_0, f_1, \ldots, f_{k-1})$ to the $(q + 1)$-tuple $(\{f(\beta_j), \beta_j \in \mathbb{F}_q\}, f_{k-1})$— *i.e.,* to the RS codeword corresponding to $f(z)$, plus an additional component equal to f_{k-1}. Show that the q^k $(q+1)$-tuples generated by this mapping as the polynomial $f(z)$ ranges over all q^k polynomials over $\mathbb{F}_q$ of degree less than k form a linear $(n = q + 1, k, d = n - k + 1)$ MDS code over $\mathbb{F}_q$. [Hint: $f(z)$ has degree less than $k - 1$ if and only if $f_{k-1} = 0$.]

Define a $(4, 2, 3)$ linear code over $\mathbb{F}_3$. Verify that all nonzero words have weight 3. □

8.3.2 Punctured RS codes as transform codes

The $(n = q, k, d = n - k + 1)$ RS code defined above has generators $\mathbf{g}_j$ corresponding to the polynomials $f_j(z) = z^j, 0 \le j \le k - 1$, evaluated at each of the elements of $\mathbb{F}_q$.

In this section we will consider the punctured $(n = q - 1, k, d = n - k + 1)$ RS code obtained by puncturing the above code in the first symbol position, whose locator is $\beta_0 = 0$. We will see that such a code may be regarded as being generated by a transform similar to a Fourier transform. We will also see that it can be characterized as the set of all polynomial multiples of a certain generator polynomial $g(z)$, and that it has a cyclic property.

A handy general lemma in any field $\mathbb{F}$ is as follows:

Lemma 8.6 *Let $\mathbb{F}$ be a field with multiplicative identity 1, and let ω be any nth root of 1 in $\mathbb{F}$ other than 1; i.e., such that $\omega^n = 1, \omega \neq 1$. Then*

$$\sum_{j=0}^{n-1} \omega^j = 0.$$

Proof. Since $\omega^n = 1 = \omega^0$, we have

$$\omega \sum_{j=0}^{n-1} \omega^j = \sum_{j=1}^{n} \omega^j = \sum_{j=0}^{n-1} \omega^j;$$

i.e., the sum $S = \sum_{j=0}^{n-1} \omega^j$ satisfies $\omega S = S$. If $\omega \neq 1$, then this implies $S = 0$. □

If ω is an nth root of unity, then so is ω^i for any integer i, since $(\omega^i)^n = \omega^{in} = (\omega^n)^i = 1$. Therefore we have the immediate corollary:

Corollary 8.7 *Let $\mathbb{F}$ be a field with multiplicative identity 1, let ω be a primitive nth root of 1 in $\mathbb{F}$ (i.e., $\omega^n = 1$ and $\omega^i \neq 1$ for $1 \le i < n$), and let i be any integer. Then*

$$\sum_{j=0}^{n-1} \omega^{ij} = \begin{cases} n, & \text{if } i = 0 \bmod n; \\ 0, & \text{otherwise.} \end{cases}$$

Proof. If $i = 0 \bmod n$, then $\omega^i = 1$ and the sum is simply $1 + 1 + \cdots + 1$ (n times), an element of $\mathbb{F}$ which we denote by n. If $i \neq 0 \bmod n$ and ω is a primitive nth root of 1, then ω^i is an nth root of 1 other than 1, so Lemma 8.6 applies and the sum is 0. □

Note that if $\mathbb{F}$ is a finite field $\mathbb{F}_q$ and $n = q - 1$, then $n = -1$ in $\mathbb{F}_q$, since p divides q and thus $q = 0$ in $\mathbb{F}_q$.

Using this corollary, we can then define a "finite Fourier transform" (FFT) and "inverse finite Fourier transform" (IFFT) over $\mathbb{F}$ as follows. Let $\mathbf{f} = (f_0, f_1, \ldots, f_{n-1}) \in \mathbb{F}^n$ be any n-tuple of elements of $\mathbb{F}$. Then the FFT of $\mathbf{f}$ is defined as the n-tuple $\mathbf{F} = (F_0, F_1, \ldots, F_{n-1}) \in \mathbb{F}^n$, where

$$F_i = \sum_{j=0}^{n-1} f_j \omega^{ij}.$$

The IFFT of $\mathbf{F}$ is defined as $\mathbf{f} = (f_0, f_1, \ldots, f_{n-1}) \in \mathbb{F}^n$, where

$$f_j = n^{-1} \sum_{i=0}^{n-1} F_i \omega^{-ij}.$$

Here n^{-1} denotes the multiplicative inverse of n in $\mathbb{F}_q$, which we assume exists; *i.e.*, $n \neq 0 \in \mathbb{F}_q$. Using Corollary 8.7, we then verify that the IFFT of the FFT of $\mathbf{f}$ is $\mathbf{f}$:

$$n^{-1} \sum_{i=0}^{n-1} \sum_{j'=0}^{n-1} f_{j'} \omega^{ij'} \omega^{-ij} = n^{-1} \sum_{j'=0}^{n-1} f_{j'} \sum_{i=0}^{n-1} \omega^{i(j'-j)} = \sum_{j'=0}^{n-1} f_{j'} \delta_{jj'} = f_j,$$

since Corollary 8.7 shows that $\sum_{i=0}^{n-1} \omega^{i(j'-j)}$ is equal to n when $j = j' \pmod n$ and 0 otherwise. Similarly the FFT of the IFFT of $\mathbf{F}$ is $\mathbf{F}$. In other words, $\mathbf{f} \leftrightarrow \mathbf{F}$ is a transform pair over $\mathbb{F}_q$ analogous to a finite Fourier transform pair over the complex field $\mathbb{C}$.

Now a codeword $\mathbf{F} = (f(1), f(\alpha), \ldots, f(\alpha^{-1}))$ of a punctured $(n = q-1, k, d = n-k+1)$ RS code over $\mathbb{F}_q$ is obtained from an information k-tuple $(f_0, f_1, \ldots, f_{k-1})$ as follows:

$$\mathbf{F} = \sum_{j=0}^{k-1} f_j \alpha^{ij},$$

where α is a primitive nth root of unity in $\mathbb{F}_q$. Thus $\mathbf{F}$ may be regarded as the FFT of the n-tuple $\mathbf{f} = (f_0, f_1, \ldots, f_{k-1}, 0, \ldots, 0)$ whose last $n-k$ elements are zeroes, where the FFT is defined using a primitive nth root of unity $\alpha \in \mathbb{F}_q$.

Using the IFFT and the fact that $n^{-1} = (-1)^{-1} = -1$, the information vector $\mathbf{f}$ is therefore easily recovered from a codeword $\mathbf{F}$ as follows:

$$f_j = -\sum_{i=0}^{n-1} F_i \alpha^{-ij}.$$

Note that the IFFT must equal 0 for the last $n-k$ symbols $f_j, k \leq j \leq n-1$:

$$\sum_{i=0}^{n-1} F_i \alpha^{-ij} = 0, k \leq j \leq n-1.$$

8.3.3 RS codes as sets of polynomials

Let us define a polynomial $F(z) = F_0 + F_1 z + \cdots + F_{n-1} z^{n-1}$ of degree $n-1$ or less corresponding to a codeword $\mathbf{F}$. This last equation then implies that that $F(\alpha^{-j}) = 0$ for $k \leq j \leq n-1$. In other words, $\alpha^j \in \mathbb{F}_q$ is a root of $F(z)$ for $1 \leq j \leq n-k$.

It follows that the polynomial $F(z)$ has $n-k$ factors of the form $z - \alpha^j, 1 \leq j \leq n-k$. In other words, $F(z)$ is divisible by the *generator polynomial*

$$g(z) = \prod_{j=1}^{n-k} \left(z - \alpha^j\right).$$

Evidently $\deg g(z) = n-k$.

In fact, the RS code may now be characterized as the set of all polynomials $F(z)$ of degree less than n that are multiples of the degree-$(n-k)$ generator polynomial $g(z)$. We have already observed that $g(z)$ divides every codeword polynomial $F(z)$. But since $F(z) = g(z)h(z)$ has degree less than n if and only if $h(z)$ has degree less than k, there are precisely q^k distinct multiples $g(z)h(z)$ of $g(z)$ such that $\deg F(z) < n$. This accounts for all codewords. Thus the RS code may be characterized as follows:

Theorem 8.8 (RS codes as sets of polynomials) *The $(n = q-1, k, d = n-k+1)$ punctured RS code over $\mathbb{F}_q$ corresponds to the set of q^k polynomials*

$$\{F(z) = g(z)h(z), \deg h(z) < k\},$$

where $g(z) = \prod_{j=1}^{n-k} \left(z - \alpha^j\right)$.

Example 2 (cont.) For the $(4, 2, 3)$ RS code over $\mathbb{F}_4$ of Example 2, if we puncture the first symbol, then we get a $(3, 2, 2)$ RS code with generators $\mathbf{g}_0 = (1, 1, 1)$ and $\mathbf{g}_1 = (1, \alpha, \alpha^2)$ (where α is a primitive field element). This code corresponds to the set of 16 polynomials $\{F(z) = g(z)h(z), \deg h(z) < 2\}$ of degree less than 3 that are multiples of the degree-1 generator polynomial $g(z) = z + \alpha$, as can be verified by inspection. □

8.3.4 Cyclic property of punctured RS codes

Finally, we show that punctured RS codes are cyclic; *i.e.,* the end-around cyclic rotation $\mathbf{F}' = (F_{n-1}, F_0, F_1, \ldots, F_{n-2})$ of any codeword $\mathbf{F} = (F_0, F_1, \ldots, F_{n-1})$ is a codeword. In polynomial terms, this is equivalent to the statement that $F'(z) = zF(z) - F_{n-1}(z^n - 1)$ is a codeword.

We have seen that the polynomial $z^n - 1$ factors completely as follows: $z^n - 1 = \prod_{j=1}^{n} \left(z - \alpha^j\right)$. Consequently $g(z) = \prod_{j=1}^{n-k} \left(z - \alpha^j\right)$ divides $z^n - 1$. Since $g(z)$ also divides $F(z)$, it follows that $g(z)$ divides $F'(z) = zF(z) - F_{n-1}(z^n - 1)$, so $F'(z)$ is a codeword.

Example 2 (cont.) The punctured $(3, 2, 2)$ RS code described above has the cyclic property, as can be verified by inspection. □

Historically, RS codes were introduced by Reed and Solomon (1960) as valuation codes. In the 1960s and 1970s, RS and BCH codes were primarily studied as cyclic codes. The transform approach was popularized by Blahut in the early 1980s. In the past decade, RS codes have again come to be regarded as valuation codes, due in part to the work of Sudan *et al.* on decoding as interpolation. This will be our next topic.

8.4 Introduction to RS decoding

An important property of RS codes is that in practice they can be decoded rather easily, using finite-field arithmetic. For example, decoding of the NASA-standard $(255, 233, 33)$ 16-error-correcting code over $\mathbb{F}_{256}$ is today routinely accomplished at rates of Gb/s.

A series of decoding algorithms bearing names such as Peterson, Berlekamp-Massey, Euclid, and Welch-Berlekamp have been developed over the years for error-correction and erasure-and-error correction. More recently, Sudan and others have devised list and soft-decision error-correction algorithms that can decode significantly beyond the guaranteed error-correction capability of the code. All of these algorithms are polynomial-time, with the most efficient requiring a number of finite-field operations of the order of n^2.

We will present briefly an error-correction algorithm based on viewing the decoding problem as an interpolation problem. Recent decoding algorithms, including the Sudan-type algorithms, involve extensions of the ideas in this algorithm.

We start by introducing a new notation for n-tuples over $\mathbb{F}_q$ for $n \leq q$. As before, we index each of the $n \leq q$ coordinates by a unique field element $\beta_j \in \mathbb{F}_q, 1 \leq j \leq n$. An n-tuple $(y_1, y_2, \ldots, y_n)$ can then be represented by a set of pairs $\{(y_j, \beta_j), 1 \leq j \leq n\}$. Each pair $(y_j, \beta_j) \in (\mathbb{F}_q)^2$ is regarded as a *point* in the two-dimensional space $(\mathbb{F}_q)^2$. An n-tuple is thus specified by a set of n points in $(\mathbb{F}_q)^2$.

A codeword in an $(n = q, k, d = n - k + 1)$ RS code is then an n-tuple specified by a set of $n = q$ points (y_j, β_j) satisfying the equation $y_j = f(\beta_j)$ for a fixed polynomial $f(z) \in \mathbb{F}_q[z]$ of degree less than k. In other words, the n points of the codeword are the n roots $(y, z) \in (\mathbb{F}_q)^2$ of the algebraic equation $y = f(z)$, or $y - f(z) = 0$.

Curves specified by such algebraic equations are called *algebraic curves*. The n points of the curve are said to *lie on* the bivariate polynomial $p(y, z) = y - f(z)$; or, the bivariate polynomial $p(y, z) = y - f(z)$ is said to *pass through* the point $(\alpha, \beta) \in (\mathbb{F}_q)^2$ if $p(\alpha, \beta) = 0$.

The decoding problem can then be expressed as an interpolation problem: given n received points, find the bivariate polynomial of the form $p(y, z) = y - f(z)$ with $\deg f(z) < k$ that passes through as many of the received points as possible.

To make further progress, we express the received n-tuple $\mathbf{y}$ as the sum (over $\mathbb{F}_q$) of a codeword $\mathbf{c}$ and an unknown error n-tuple $\mathbf{e}$. The closest codeword $\mathbf{c}$ is the one for which the corresponding error n-tuple $\mathbf{e} = \mathbf{y} - \mathbf{c}$ has least Hamming weight. If the minimum distance of the code is d and $\mathbf{y}$ is within distance $t < d/2$ of a codeword $\mathbf{c}$, then the distance to any other codeword is at least $d - t > d/2$, so the closest codeword is unique.

An *error-locator polynomial* associated with an error n-tuple $\mathbf{e}$ is any polynomial $\Lambda(z)$ such that $\Lambda(\beta_j) = 0$ whenever $e_j \neq 0$. If the error vector has weight t, then clearly the degree-t polynomial

$$\Lambda_0(z) = \prod_{j:e_j \neq 0} (z - \beta_j)$$

whose t roots are the error locators $\{\beta_j : e_j \neq 0\}$ is an error-locator polynomial. Moreover, every error-locator polynomial $\Lambda(z)$ must be a polynomial multiple of $\Lambda_0(z)$.

If $\Lambda(z)$ is an error-locator polynomial for $\mathbf{e}$ and $f(z)$ is the polynomial that maps to the codeword $\mathbf{c}$, then the bivariate polynomial

$$q(y, z) = \Lambda(z)(y - f(z))$$

evaluates to 0 for every received point (y_j, β_j); *i.e.*, $q(y, z)$ passes through all n received points. The decoding interpolation problem can therefore be expressed as follows: given n received points, find the bivariate polynomial of the form $q(y, z) = \Lambda(z)(y - f(z))$ with $\deg f(z) < k$ and $\deg \Lambda(z) = t$ as small as possible that passes through all received points.

Define $g(z) = \Lambda(z)f(z)$; then

$$q(y, z) = y\Lambda(z) - g(z).$$

If $\deg \Lambda(z) = t$, then $\deg g(z) < k + t$. Thus, given t, the coefficients of $\Lambda(z)$ and $g(z)$ are a set of $t + 1 + k + t = k + 2t + 1$ unknowns.

Assuming that the number of errors is not greater than t', and substituting the n received points (y_j, β_j) in the equation $y\Lambda(z) - g(z) = 0$, we obtain a system of n homogeneous linear equations in $k + 2t' + 1$ unknowns. In general, these equations may have no solutions other than the all-zero solution, or may have a linear vector space of solutions. If $k + 2t' + 1 > n$, then there must be a space of solutions of dimension at least $k + 2t' + 1 - n$. In particular, if $2t' + 1 = d = n - k + 1$, then there is at least a one-dimensional space of solutions.

The set of all such solutions may be found by standard techniques for solving systems of linear equations, which in general have complexity of the order of n^3. Because of the special structure of these equations, "fast" techniques with complexity of the order of n^2 may be used. There is active current research on such fast solution techniques.

Any such solution specifies two candidate polynomials $\Lambda(z)$ and $g(z)$. If these are the correct polynomials, then since $g(z) = \Lambda(z)f(z)$, the polynomial $f(z)$ may be found simply by dividing $g(z)$ by $\Lambda(z)$. (Note that the roots of $\Lambda(z)$ need not be found explicitly.) If the candidate polynomials are incorrect, then $g(z)$ may not be divisible by $\Lambda(z)$, in which case this candidate pair may be discarded. If $g(z)$ is divisible by $\Lambda(z)$ and the result $f(z) = g(z)/\Lambda(z)$ is a polynomial of degree less than k such that the codeword corresponding to $f(z)$ is within distance t' of the received n-tuple, then a candidate for the closest codeword has been found.

For standard errors-only error-correction, the distance d is chosen to be odd and t' is chosen such that $2t' + 1 = d$. We then obtain n homogeneous linear equations in $n + 1$ unknowns, ensuring that there exists at least a one-dimensional space of solutions. If the actual number of errors is $t \leq t'$, then there is guaranteed to be one and only one valid candidate solution, so as soon as any valid solution is found, it may be declared to be the closest codeword.

For erasure-and-error-correction, or for decoding of a punctured code, a certain number s of the received symbols may be erased or punctured— *i.e.,* unknown. The above decoding method is easily adapted to this case. We then have a system of $n - s$ homogeneous linear equations in $k + 2t' + 1$ unknowns. If $d - s$ is odd and t' is chosen so that $2t' + s + 1 = d$, then we obtain $n - s$ equations in $k + d - s = n - s + 1$ unknowns, ensuring at least a one-dimensional space of solutions. Moreover, if the actual number of errors is $t \leq t'$, then there is guaranteed to be one and only one valid candidate solution.

Recent work by Sudan *et al.* has explored decoding beyond the guaranteed error-correction capability of the code. The main idea is to replace $\Lambda(z)$ by $\Lambda(y, z)$. Then the polynomial $q(y, z)$ can have more then one factor of type $y - f(z)$, and the list of all such factors is produced at the decoder output. The number of such factors (the list size in Sudan-type decoding) is obviously bounded by $\deg_y q(y, z) = \deg_y \Lambda(y, z) + 1$. It can be shown that under certain conditions, the codeword $\mathbf{c}$ will be on the list, even if the distance between $\mathbf{y}$ and $\mathbf{c}$ exceeds $d/2$. In practice, the probability that the list will contain more than one codeword is usually very small.

Finally, in many cases it may be possible to assign a reliability to each received symbol, where the reliability is an estimate of the log likelihood of the symbol being correct. Or, the symbol receiver may itself put out a list of more than one candidate symbol, each with a different reliability. In such cases the decoder may develop a list of candidate codewords, from which the one with greatest reliability (log likelihood) may then be chosen. Such reliability-based decoding schemes can make up much of the performance difference between hard-decision and maximum-likelihood decoding.

8.5 Applications of RS codes

RS codes are useful for a variety of applications:

(a) For channels that naturally transmit packets or binary m-tuples, RS codes over $\mathbb{F}_{2^m}$ are used for *m-tuple (byte) error correction.*

(b) Over memoryless channels such as the AWGN channel, powerful codes may be constructed by *concatenation* of an "inner code" consisting of $q = 2^m$ codewords or signal points together with an "outer code" over $\mathbb{F}_q$, where $\mathbb{F}_q$ is identified with the inner codewords. Such a concatenated code may be efficiently decoded by maximum-likelihood detection of the inner code followed by algebraic decoding of the RS outer code, preferably using reliability information from the inner decoder in the RS decoding.

(c) RS codes are also useful for *burst error correction*; for this purpose they are often combined with m-tuple (byte) interleavers.

8.5.1 Concatenation using RS Codes

A basic concatenated code involves two codes, called the inner code and outer code, as shown in Figure 1. We see immediately that a concatenation scheme is another example of a layered architecture.

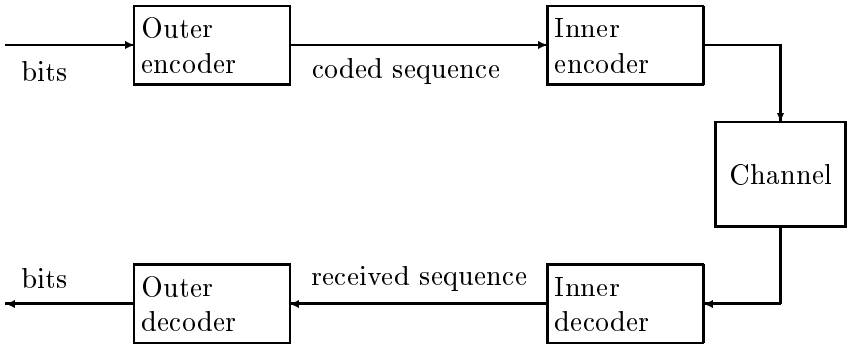


Figure 1. Concatenation of an inner code and an outer code.

The inner code is typically a binary code, either block or convolutional, used over a binary channel, such as a discrete-time AWGN channel with a 2-PAM signal set. The inner decoder is typically a maximum-likelihood decoder, such as a Viterbi decoder, which uses soft (reliability-weighted) outputs from the channel.

The outer code is typically a RS code over a finite field $\mathbb{F}_{2^m}$ of characteristic 2. Each field element may thus be represented by a binary m-tuple. The field $\mathbb{F}_{256}$ is commonly used, in which case field elements are represented by 8-bit bytes. An RS codeword of length n symbols may therefore be converted into an RS-coded sequence of nm bits, which is the input sequence to the inner encoder.

The input sequence to the inner encoder is often permuted by an interleaver to ensure that symbols in a given RS codeword are transmitted and decoded independently. A simple row-

column block interleaver works as follows, assuming that the inner code is a binary linear block code of dimension $k = mN$. Each RS codeword, consisting of n m-bit symbols, forms a horizontal row in a rectangular $N \times n$ array of m-bit symbols, as shown below. Each column of the array, consisting of N symbols, is taken as the input symbol sequence of length $k = mN$ bits for one inner code block.

symbol 1 in RS cw 1	symbol 2 in RS cw 1	...	symbol n in RS cw 1
symbol 1 in RS cw 2	symbol 2 in RS cw 2	...	symbol n in RS cw 2
...	...		...
symbol 1 in RS cw N	symbol 2 in RS cw N	...	symbol n in RS cw N

The input sequence is read out column by column, with the column length N chosen large enough so that symbols in the same row are effectively independent. Such an interleaver also protects against error bursts of length mN bits or less.

The output of the inner decoder is correspondingly viewed as a sequence of m-bit symbols. Typically the inner decoder puts out a sequence of hard decisions on bits, which are converted to hard decisions on symbols. As we have shown, the inner decoder output may preferably also include reliability information indicating how confident it is in its hard decisions. However, for simplicity we will omit this refinement. The performance of the inner code may then be characterized by the probability $P_s(E)$ that an m-bit output symbol is in error.

The outer RS decoder takes this sequence of symbol hard decisions and performs error-correction, using an algebraic decoding algorithm. If the RS code has minimum distance d, then the decoder can decode any pattern of t errors correctly provided that $2t < d$; *i.e.,* provided that $t \leq t_{\max} = \lfloor (d-1)/2 \rfloor$. Most RS error-correctors give up if there is no codeword within Hamming distance $t_{\max}$ of the received sequence. With such a bounded-distance decoder, therefore, the probability per codeword of not decoding correctly is

$$\Pr(\text{ndc}) = \Pr\{t_{\max} + 1 \text{ or more symbol errors}\}.$$

It is difficult to obtain good analytical estimates of the probability of not decoding correctly when $p = P_s(E)$ is relatively large, which is the operating region for concatenated codes (but see Exercise 4 below). Empirically, one observes that $\Pr(\text{ndc}) \approx 1$ when $p \approx t_{\max}/n$. As p decreases, there is a threshold region, typically $p \approx 10^{-2}$–10^{-3}, where $\Pr(\text{ndc})$ suddenly decreases to very low values, *e.g.,* $\Pr(\text{ndc}) \approx 10^{-12}$.

The objective of the inner code is therefore to achieve a very moderate symbol probability of error such as $P_s(E) \approx 10^{-2}$–10^{-3} at as low an E_b/N_0 as possible (on an AWGN channel). For this purpose maximum-likelihood decoding— specifically, the VA— is a good choice. The code should have a trellis with as many states as the VA can reasonably decode, and should be the best code with that number of states, which almost always implies a convolutional code. A 64-state convolutional code was standardized in the 1970's, and around 1990 NASA built a 2^{14}-state Viterbi decoder.

The objective of the outer code is then to drive $\Pr(\text{ndc})$ down to the target error rate with as little redundancy as possible. RS codes and decoders are ideal for this purpose. A $(255, 223, 33)$ RS code over $\mathbb{F}_{256}$ was standardized in the 1970's, and is still in common use, nowadays at rates up to Gb/s. Concatenated codes like this can attain low error rates within about 2–3 dB of the Shannon limit on a power-limited AWGN channel.

Exercise 4. Show that if symbol errors occur independently with probability $p = P_s(E)$, then the probability of the RS decoder not decoding correctly is

$$\Pr(\text{ndc}) = \sum_{t=t_{\max}+1}^{n} \binom{n}{t} p^t (1-p)^{n-t}.$$

Using the Chernoff bound, prove the exponential upper bound

$$\Pr(\text{ndc}) \leq e^{-nD(\tau||p)},$$

where $\tau = (t_{\max}+1)/n$ and $D(\tau||p)$ is the divergence (relative entropy)

$$D(\tau||p) = \tau \ln \frac{\tau}{p} + (1-\tau) \ln \frac{1-\tau}{1-p}$$

between two Bernoulli distributions with parameters τ and p, respectively. □

8.6 Binary BCH codes

In a finite field $\mathbb{F}_{2^m}$ of characteristic 2, there always exists a prime subfield $\mathbb{F}_2$ consisting of the two elements $\mathbb{F}_2 = \{0, 1\}$. Correspondingly, in an $(n, k, d = n-k+1)$ RS code over $\mathbb{F}_{2^m}$, there always exists a subset of codewords whose components are entirely in the subfield $\mathbb{F}_2$. This "subfield subcode" is called a binary Bose-Chaudhuri-Hocquenghem (BCH) code. (Historically, binary BCH codes were developed independently of RS codes, but at about the same time, namely 1959-60.)

Example 2 (cont.). The binary BCH code derived from the $(4, 2, 3)$ RS code over $\mathbb{F}_4$ of Example 2 is the binary code consisting of the two codewords 0000 and 1111; *i.e.,* the $(4, 1, 4)$ repetition code over $\mathbb{F}_2$.

A binary BCH code is obviously linear, since the sum of any two binary RS codewords is another binary RS codeword. Its parameters (n', k', d') are related to the parameters (n, k, d) of the RS code from which it is derived as follows. Its length n' is evidently the same as the length n of the RS code. Its minimum Hamming distance d' must be at least as great as the minimum distance d of the RS code, because any two words must differ in at least d places. Usually $d' = d$. Finally, by the Singleton bound, $k' \leq n' - d' + 1 \leq n - d + 1 = k$. Usually k' is considerably less than k, and the binary BCH code falls considerably short of being MDS.

Determining k' for a binary BCH code derived from a given (n, k, d) RS code over F_{2^m} is an exercise in cyclotomics. Let us associate a polynomial $F(z) \in \mathbb{F}_{2^m}[z]$ with each codeword $\mathbf{F} = (F_0, F_1, \ldots, F_{n-1})$ of the RS code as in Section 8.3.3. We showed in Lemma 7.19 that $F(z)$ is actually a binary polynomial with all coefficients in the binary subfield $\{0, 1\}$ if and only if the roots of $F(z)$ are cyclotomic cosets; *i.e.,* if $\beta \in \mathbb{F}_{2^m}$ is a root of $F(z)$, then so are $\beta^2, \beta^4, \ldots$.

It was further shown in Section 8.3.3 that the punctured $(n = q-1, k, d = n-k+1)$ RS code over $\mathbb{F}_q$ may be characterized as the set of q^k polynomials $F(z)$ of degree less than n that are multiples of the generator polynomial $g(z) = \prod_{j=1}^{n-k}(z - \alpha^j)$, whose roots are the first $n-k$ powers $\{\alpha, \alpha^2, \ldots, \alpha^{n-k}\}$ of a primitive element $\alpha \in \mathbb{F}_q$. Since $n-k = d-1$, the subset of binary polynomials in this set can thus be characterized as follows:

Theorem 8.9 (BCH codes) *Given a field $\mathbb{F}_q$ of characteristic 2, the $(n = q-1, k', d)$ BCH code over $\mathbb{F}_2$ corresponds to the set of $2^{k'}$ binary polynomials*

$$\{F(z) = g(z)h(z), \deg h(z) < k'\},$$

where $g(z)$ is the product of the distinct polynomials in the set of cyclotomic polynomials of the elements $\{\alpha, \alpha^2, \ldots, \alpha^{d-1}\}$ of $\mathbb{F}_q$, and $k' = n - \deg g(z)$.

Example 3. Let us find the parameters (k', d) of the BCH codes of length $n = 15$. The cyclotomic polynomials of the elements of $\mathbb{F}_{16}$ are the binary irreducible polynomials whose degrees divide 4, namely (taking α to be a root of $x^4 + x + 1$):

polynomial	roots
x	0
$x+1$	1
x^2+x+1	α^5, α^{10}
x^4+x+1	$\alpha, \alpha^2, \alpha^4, \alpha^8$
$x^4+x^3+x^2+x+1$	$\alpha^3, \alpha^6, \alpha^{12}, \alpha^9$
x^4+x^3+1	$\alpha^7, \alpha^{14}, \alpha^{13}, \alpha^{11}$

For the first BCH code, we take $g(x) = x^4 + x + 1$, which has α and α^2 as roots; therefore this code is the subfield subcode of the $(15, 13, 3)$ RS code over $\mathbb{F}_{16}$. It has $d = 3$ and $k' = n - \deg g(x) = 15 - 4 = 11$; *i.e.,* it is a $(15, 11, 3)$ code.

For the second BCH code, we take $g(x) = (x^4 + x + 1)(x^4 + x^3 + x^2 + x + 1)$, which has $\{\alpha, \alpha^2, \alpha^3, \alpha^4\}$ as roots; therefore this code is the subfield subcode of the $(15, 11, 5)$ RS code over $\mathbb{F}_{16}$. It has $d = 5$ and $k' = n - \deg g(x) = 7$; *i.e.,* it is a $(15, 7, 5)$ code.

For the third BCH code, we take $g(x) = (x^4 + x + 1)(x^4 + x^3 + x^2 + x + 1)(x^2 + x + 1)$, which has $\{\alpha, \alpha^2, \alpha^3, \alpha^4, \alpha^5, \alpha^6\}$ as roots; therefore this code is the subfield subcode of the $(15, 9, 7)$ RS code over $\mathbb{F}_{16}$. It has $d = 7$ and $k' = n - \deg g(x) = 5$; *i.e.,* $(n, k, d) = (15, 5, 7)$.

Finally, we find a $(15, 1, 15)$ BCH code with a degree-14 generator polynomial $g(x)$ which has $\{\alpha, \alpha^2, \ldots, \alpha^{14}\}$ as roots; *i.e.,* the binary repetition code of length 15. □

Table I below gives the parameters $(n = q, k', d+1)$ for all binary BCH codes of lengths $n = 2^m$ with even minimum distances $d+1$. By puncturing one symbol, one may derive $(n = q-1, k', d)$ binary BCH codes with odd minimum distances as above; these are the binary BCH codes that are more usually tabulated. The former codes may be obtained from the latter by adding an overall parity check as in Chapter 6, Exercise 1.

$n = q$	k'	$d+1$	$n = q$	k'	$d+1$	$n = q$	k'	$d+1$
2	1	2	16	15	2	64	63	2
			16	11	4	64	57	4
			16	7	6	64	51	6
4	3	2	16	5	8	64	45	8
4	1	4	16	1	16	64	39	10
						64	36	12
			32	31	2	64	30	14
8	7	2	32	26	4	64	24	16
8	4	4	32	21	6	64	18	22
8	1	8	32	16	8	64	16	24
			32	11	12	64	10	28
			32	6	16	64	7	32
			32	1	32	64	1	64

Table I. Binary BCH codes of lengths $n = 2^m$ for $m \leq 6$.

We see that binary BCH codes with $n = q$ include codes equivalent to SPC codes, extended Hamming codes, biorthogonal codes, and repetition codes.

In comparison to binary RM codes, there exist more binary BCH codes; in particular, codes with distances other than 2^{m-r}. Also, for $n \geq 64$, we begin to see codes with slightly better k for a given n and d; *e.g.,* the $(64, 45, 8)$ binary BCH code has three more information bits than the corresponding RM code, and the $(64, 24, 16)$ code has two more information bits.

In principle, a binary BCH code may be decoded by any decoding algorithm for the RS code from which it is derived. Candidate decoded RS codewords that are not binary may be discarded. In practice, certain simplifications are possible in the binary case.

For these reasons, particularly the availability of algebraic decoding algorithms, binary BCH codes have historically received more attention than binary RM codes.

However, for trellis-based maximum-likelihood (Viterbi) decoding, which we will discuss shortly, RM codes usually have a better performance-complexity tradeoff. For example, the $(64, 45, 8)$ binary BCH code has eight times the trellis complexity of the $(64, 42, 8)$ RM code, and the $(64, 24, 16)$ binary BCH code has four times the trellis complexity of the $(64, 22, 16)$ RM code. The corresponding increase in complexity of trellis-based decoding algorithms will usually not justify the slight increase in number of information bits.

Chapter 9

Introduction to convolutional codes

We now introduce binary linear convolutional codes, which like binary linear block codes are useful in the power-limited (low-SNR, low-ρ) regime. In this chapter we will concentrate on rate-$1/n$ binary linear time-invariant convolutional codes, which are the simplest to understand and also the most useful in the power-limited regime. Here is a canonical example:

Example 1. Figure 1 shows a simple rate-1/2 binary linear convolutional encoder. At each time k, one input bit u_k comes in, and two output bits (y_{1k}, y_{2k}) go out. The input bits enter a 2-bit shift register, which has 4 possible states (u_{k-1}, u_{k-2}). The output bits are binary linear combinations of the input bit and the stored bits.

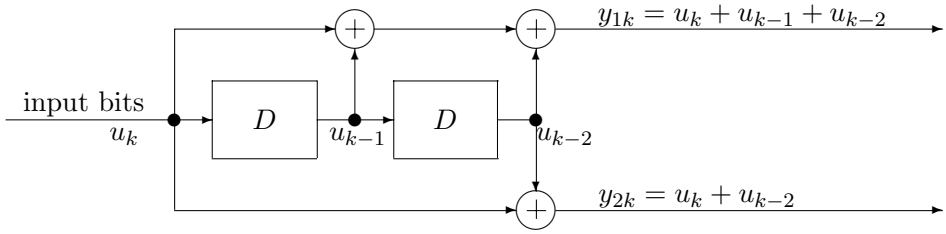


Figure 1. Four-state rate-1/2 binary linear convolutional encoder.

The code $\mathcal{C}$ generated by this encoder is the set of all output sequences that can be produced in response to an input sequence $\mathbf{u} = (\ldots, u_k, u_{k+1}, \ldots)$. The code is linear because if $(\mathbf{y}_1, \mathbf{y}_2)$ and $(\mathbf{y}'_1, \mathbf{y}'_2)$ are the code sequences produced by $\mathbf{u}$ and $\mathbf{u}'$, respectively, then $(\mathbf{y}_1 + \mathbf{y}'_1, \mathbf{y}_2 + \mathbf{y}'_2)$ is the code sequence produced by $\mathbf{u} + \mathbf{u}'$.

As in the case of linear block codes, this implies that the minimum Hamming distance d_{free} between code sequences is the minimum Hamming weight of any nonzero codeword. By inspection, we can see that the Hamming weight of the output sequence when the input sequence is $(\ldots, 0, 0, 1, 0, 0, \ldots)$ is 5, and that this is the only weight-5 sequence starting at a given time k. Of course, by time-invariance, there is such a weight-5 sequence starting at each time k.

We will see that maximum-likelihood sequence decoding of a convolutional code on an AWGN channel can be performed efficiently by the Viterbi algorithm (VA), with complexity proportional to the number of states (in this case 4). As with block codes, the probability of error per bit $P_b(E)$ may be estimated by the union bound estimate (UBE) as

$$P_b(E) \approx K_b(\mathcal{C}) Q\surd\left(\gamma_c(\mathcal{C})(2E_b/N_0)\right),$$

where the nominal coding gain is $\gamma_c(\mathcal{C}) = Rd_{\text{free}}$, R is the code rate in input bits per output bit, and $K_b(\mathcal{C})$ is the number of minimum-weight code sequences per input bit. For this code, $d_{\text{free}} = 5$, $R = 1/2$, and $K_b(\mathcal{C}) = 1$, which means that the nominal coding gain is $\gamma_c(\mathcal{C}) = 5/2$ (4 dB), and the effective coding gain is also 4 dB. There is no block code that can achieve an effective coding gain of 4 dB with so little decoding complexity. □

As Example 1 shows, convolutional codes have two different kinds of structure: algebraic structure, which arises from convolutional encoders being linear systems, and dynamical structure, which arises from convolutional encoders being finite-state systems. We will first study their linear system structure. Then we will consider their finite-state structure, which is the key to ML decoding via the VA. Finally, we will show how to estimate performance using the UBE, and will give tables of the complexity and performance of the best known rate-$1/n$ codes.

9.1 Linear time-invariant systems over finite fields

We start with a little linear system theory, namely the theory of linear time-invariant (LTI) systems over finite fields. The reader is probably familiar with the theory of discrete-time LTI systems over the real or the complex field, which are sometimes called discrete-time real or complex filters. The theory of discrete-time LTI systems over an arbitrary field $\mathbb{F}$ is similar, except that over a finite field there is no notion of convergence of an infinite sum.

9.1.1 The input/output map of an LTI system

In general, a discrete-time system is characterized by an input alphabet $\mathcal{U}$, an output alphabet $\mathcal{Y}$, and an input/output map from bi-infinite discrete-time input sequences $\mathbf{u} = (\ldots, u_k, u_{k+1}, \ldots)$ to output sequences $\mathbf{y} = (\ldots, y_k, y_{k+1}, \ldots)$. Here we will take the input and output alphabets to be a common finite field, $\mathcal{U} = \mathcal{Y} = \mathbb{F}_q$. The indices of the input and output sequences range over all integers in $\mathbb{Z}$ and are regarded as time indices, so that for example we may speak of u_k as the value of the input at time k.

Such a system is *linear* if whenever $\mathbf{u}$ maps to $\mathbf{y}$ and $\mathbf{u}'$ maps to $\mathbf{y}'$, then $\mathbf{u} + \mathbf{u}'$ maps to $\mathbf{y} + \mathbf{y}'$ and $\alpha\mathbf{u}$ maps to $\alpha\mathbf{y}$ for any $\alpha \in \mathbb{F}_q$. It is *time-invariant* if whenever $\mathbf{u}$ maps to $\mathbf{y}$, then $D\mathbf{u}$ maps to $D\mathbf{y}$, where D represents the *delay operator*, namely the operator whose effect is to delay every element in a sequence by one time unit; *i.e.,* $\mathbf{u}' = D\mathbf{u}$ means $\mathbf{u}'_k = \mathbf{u}_{k-1}$ for all k.

It is well known that the input/output map of an LTI system is completely characterized by its *impulse response* $\mathbf{g} = (\ldots, 0, 0, \ldots, 0, g_0, g_1, g_2, \ldots)$ to an input sequence $\mathbf{e}_0$ which is equal to 1 at time zero and 0 otherwise. The expression for $\mathbf{g}$ assumes that the LTI system is *causal*, which implies that the impulse response $\mathbf{g}$ must be equal to 0 before time zero.

The proof of this result is as follows. If the input is a sequence $\mathbf{e}_k$ which is equal to 1 at time k and zero otherwise, then since $\mathbf{e}_k = D^k\mathbf{e}_0$, by time invariance the output must be $D^k\mathbf{g}$. Then since an arbitrary input sequence $\mathbf{u}$ can be written as $\mathbf{u} = \sum_k u_k\mathbf{e}_k$, by linearity the output must be the linear combination

$$\mathbf{y} = \sum_k u_k D^k \mathbf{g}.$$

The output y_k at time k is thus given by the convolution

$$y_k = \sum_{k' \le k} u_{k'} g_{k-k'}, \tag{9.1}$$

where we use the fact that by causality $g_{k-k'} = 0$ if $k < k'$. In other words, $\mathbf{y}$ is the convolution of the input sequence $\mathbf{u}$ and the impulse response $\mathbf{g}$:

$$\mathbf{y} = \mathbf{u} * \mathbf{g}.$$

It is important to note that the sum (**??**) that defines y_k is well defined if and only if it is a sum of only a finite number of nonzero terms, since over finite fields there is no notion of convergence of an infinite sum. The sum (**??**) is finite if and only if one of the following two conditions holds:

(a) There are only a finite number of nonzero elements g_k in the impulse response $\mathbf{g}$;

(b) For every k, there are only a finite number of nonzero elements $u_{k'}$ with $k' \leq k$ in the input sequence $\mathbf{u}$. This occurs if and only if there are only a finite number of nonzero elements u_k with negative time indices k. Such a sequence is called a *Laurent sequence*.

Since we do not in general want to restrict $\mathbf{g}$ to have a finite number of nonzero terms, we will henceforth impose the condition that the input sequence $\mathbf{u}$ must be Laurent, in order to guarantee that the sum (**??**) is well defined. With this condition, we have our desired result:

Theorem 9.1 (An LTI system is characterized by its impulse response) *If an LTI system over $\mathbb{F}_q$ has impulse response $\mathbf{g}$, then the output sequence in response to an arbitrary Laurent input sequence $\mathbf{u}$ is the convolution $\mathbf{y} = \mathbf{u} * \mathbf{g}$.*

9.1.2 The field of Laurent sequences

A nonzero Laurent sequence $\mathbf{u}$ has a definite starting time or *delay*, namely the time index of the first nonzero element: $\operatorname{del} \mathbf{u} = \min\{k : u_k \neq 0\}$. The zero sequence $\mathbf{0}$ is Laurent, but has no definite starting time; by convention we define $\operatorname{del} \mathbf{0} = \infty$. A *causal sequence* is a Laurent sequence with non-negative delay.

The (componentwise) sum of two Laurent sequences is Laurent, with delay not less than the minimum delay of the two sequences. The Laurent sequences form an abelian group under sequence addition, whose identity is the zero sequence $\mathbf{0}$. The additive inverse of a Laurent sequence $\mathbf{x}$ is $-\mathbf{x}$.

The convolution of two Laurent sequences is a well-defined Laurent sequence, whose delay is equal to the sum of the delays of the two sequences. The nonzero Laurent sequences form an abelian group under convolution, whose identity is the unit impulse $\mathbf{e}_0$. The inverse under convolution of a nonzero Laurent sequence $\mathbf{x}$ is a Laurent sequence $\mathbf{x}^{-1}$ which may be determined by long division, and which has delay equal to $\operatorname{del} \mathbf{x}^{-1} = -\operatorname{del} \mathbf{x}$.

Thus the set of all Laurent sequences forms a field under sequence addition and convolution.

9.1.3 D-transforms

The fact that the input/output map in any LTI system may be written as a convolution $\mathbf{y} = \mathbf{u} * \mathbf{g}$ suggests the use of polynomial-like notation, under which convolution becomes multiplication.

Therefore let us define the formal power series $u(D) = \sum_k u_k D^k$, $g(D) = \sum_k g_k D^k$, and $y(D) = \sum_k y_k D^k$. These are called "D-transforms," although the term "transform" may be misleading because these are still time-domain representations of the corresponding sequences.

In these expressions D is algebraically just an indeterminate (place-holder). However, D may also be regarded as representing the delay operator, because if the D-transform of $\mathbf{g}$ is $g(D)$, then the D-transform of $D\mathbf{g}$ is $Dg(D)$.

These expressions appear to be completely analogous to the "z-transforms" used in the theory of discrete-time real or complex LTI systems, with the substitution of D for z^{-1}. The subtle difference is that in the real or complex case z is often regarded as a complex number in the "frequency domain" and an expression such as $g(z^{-1})$ as a set of values as z ranges over $\mathbb{C}$; *i.e.,* as a true frequency-domain "transform."

It is easy to see that the convolution $\mathbf{y} = \mathbf{u} * \mathbf{g}$ then translates to

$$y(D) = u(D)g(D),$$

if for multiplication of D-transforms we use the usual rule of polynomial multiplication,

$$y_k = \sum_{k'} u_{k'} g_{k-k'},$$

since this expression is identical to (**??**). Briefly, convolution of sequences corresponds to multiplication of D-transforms.

In general, if $x(D)$ and $y(D)$ are D-transforms, then the product $x(D)y(D)$ is well defined when either $x(D)$ or $y(D)$ is finite, or when both $x(D)$ and $y(D)$ are Laurent. Since a causal impulse response $g(D)$ is Laurent, the product $u(D)g(D)$ is well defined whenever $u(D)$ is Laurent.

Addition of sequences is defined by componentwise addition of their components, as with vector addition. Correspondingly, addition of D-transforms is defined by componentwise addition. In other words, D-transform addition and multiplication are defined in the same way as polynomial addition and multiplication, and are consistent with the addition and convolution operations for the corresponding sequences.

It follows that the set of all Laurent D-transforms $x(D)$, which are called the Laurent power series in D over $\mathbb{F}_q$ and denoted by $\mathbb{F}_q((D))$, form a field under D-transform addition and multiplication, with additive identity 0 and multiplicative identity 1.

9.1.4 Categories of D-transforms

We pause to give a brief systematic exposition of various other categories of D-transforms.

Let $f(D) = \sum_{k\in\mathbb{Z}} f_k D^k$ be the D-transform of a sequence $\mathbf{f}$. We say that $\mathbf{f}$ or $f(D)$ is "zero on the past" (resp. "finite on the past") if it has zero (resp. a finite number) of nonzero f_k with negative time indices k, and "finite on the future" if it has a finite number of nonzero f_k with non-negative time indices k. $f(D)$ is *finite* if it is finite on both past and future.

(a) (Polynomials $\mathbb{F}_q[D]$.) If $f(D)$ is zero on the past and finite on the future, then $f(D)$ is a *polynomial* in D over $\mathbb{F}_q$. The set of all polynomials in D over $\mathbb{F}_q$ is denoted by $\mathbb{F}_q[D]$. D-transform addition and multiplication of polynomials is the same as polynomial addition and multiplication. Under these operations, $\mathbb{F}_q[D]$ is a ring, and in fact an integral domain (see Chapter 7).

(b) (Formal power series $\mathbb{F}_q[[D]]$.) If $f(D)$ is zero on the past and unrestricted on the future, then $f(D)$ is a *formal power series* in D over $\mathbb{F}_q$. The set of all formal power series in D over $\mathbb{F}_q$ is denoted by $\mathbb{F}_q[[D]]$. Under D-transform addition and multiplication, $\mathbb{F}_q[[D]]$ is a ring, and in fact an integral domain. However, $\mathbb{F}_q[[D]]$ is not a field, because D has no inverse in $\mathbb{F}_q[[D]]$. A formal power series corresponds to a causal sequence.

(c) (Laurent polynomials $\mathbb{F}_q[D, D^{-1}]$.) If $f(D)$ is finite, then $f(D)$ is a *Laurent polynomial* in D over $\mathbb{F}_q$. The set of all Laurent polynomials in D over $\mathbb{F}_q$ is denoted by $\mathbb{F}_q[D, D^{-1}]$. Under D-transform addition and multiplication, $\mathbb{F}_q[D, D^{-1}]$ is a ring, and in fact an integral domain. However, $\mathbb{F}_q[D, D^{-1}]$ is not a field, because $1 + D$ has no inverse in $\mathbb{F}_q[D, D^{-1}]$. A Laurent polynomial corresponds to a finite sequence.

(d) (Laurent power series $\mathbb{F}_q((D))$.) If $f(D)$ is finite on the past and unrestricted on the future, then $f(D)$ is a *Laurent power series* in D over $\mathbb{F}_q$. The set of all Laurent power series in D over $\mathbb{F}_q$ is denoted by $\mathbb{F}_q((D))$. As we have already seen, under D-transform addition and multiplication, $\mathbb{F}_q((D))$ is a field; *i.e.,* every nonzero $f(D) \in \mathbb{F}_q((D))$ has an inverse $f^{-1}(D) \in \mathbb{F}_q((D))$ such that $f(D)f^{-1}(D) = 1$, which can be found by long division of D-transforms; *e.g.,* over any field, the inverse of D is D^{-1}, and the inverse of $1 + D$ is $1 - D + D^2 - D^3 + \cdots$.

(e) (Bi-infinite power series $\mathbb{F}_q[[D, D^{-1}]]$.) If $f(D)$ is unrestricted on the past and future, then $f(D)$ is a *bi-infinite power series* in D over $\mathbb{F}_q$. The set of all bi-infinite power series in D over $\mathbb{F}_q$ is denoted by $\mathbb{F}_q[[D, D^{-1}]]$. As we have seen, D-transform multiplication is not well defined for all $f(D), g(D) \in \mathbb{F}_q[[D, D^{-1}]]$, so $\mathbb{F}_q[[D, D^{-1}]]$ is merely a group under D-transform addition.

(f) (Rational functions $\mathbb{F}_q(D)$.) A Laurent power series $f(D)$ (or the corresponding sequence $\mathbf{f}$) is called *rational* if it can be written as $f(D) = n(D)/d(D)$, where $n(D)$ and $d(D) \neq 0$ are polynomials in $\mathbb{F}_q[D]$, and $n(D)/d(D)$ denotes the product of $n(D)$ with $d^{-1}(D)$. The set of all rational power series in D over $\mathbb{F}_q$ is denoted by $\mathbb{F}_q(D)$. It is easy to verify that that $\mathbb{F}_q(D)$ is closed under D-transform addition and multiplication. Moreover, the multiplicative inverse of a nonzero rational D-transform $f(D) = n(D)/d(D)$ is $f^{-1}(D) = d(D)/n(D)$, which is evidently rational. It follows that $\mathbb{F}_q(D)$ is a field.

It is easy to see that $\mathbb{F}_q[D] \subset \mathbb{F}_q[D, D^{-1}] \subset \mathbb{F}_q(D) \subset \mathbb{F}_q((D))$, since every polynomial is a Laurent polynomial, and every Laurent polynomial is rational (for example, $D^{-1} + 1 = (1 + D)/D$). The rational functions and the Laurent power series have the nicest algebraic properties, since both are fields. Indeed, the rational functions form a subfield of the Laurent power series, as the rational numbers $\mathbb{Q}$ form a subfield of the real numbers $\mathbb{R}$. The polynomials form a subring of the rational functions, as the integers $\mathbb{Z}$ form a subring of $\mathbb{Q}$.

The following exercise shows that an infinite Laurent D-transform $f(D)$ is rational if and only if the corresponding sequence $\mathbf{f}$ eventually becomes periodic. (This should remind the reader of the fact that a real number is rational if and only if its decimal expansion is eventually periodic.)

Exercise 1 (rational = eventually periodic). Show that a Laurent D-transform $f(D)$ is rational if and only if the corresponding sequence $\mathbf{f}$ is finite or eventually becomes periodic. [Hints: (a) show that a sequence $\mathbf{f}$ is eventually periodic with period P if and only if its D-transform $f(D)$ can be written as $f(D) = g(D)/(1 - D^P)$, where $g(D)$ is a Laurent polynomial; (b) using the results of Chapter 7, show that every nonzero polynomial $d(D) \in \mathbb{F}_q[D]$ divides $1 - D^P$ for some integer P.] □

9.1.5 Realizations of LTI systems

So far we have characterized an LTI system over $\mathbb{F}_q$ by its input/output map, which we have shown is entirely determined by its impulse response $\mathbf{g}$ or the corresponding D-transform $g(D)$. The only restriction that we have placed on the impulse response is that it be causal; *i.e.,* that $g(D)$ be a formal power series in $\mathbb{F}_q[[D]]$. In order that the input/output map be well-defined, we have further required that the input $u(D)$ be Laurent, $u(D) \in \mathbb{F}_q(D)$.

In this subsection we consider realizations of such an LTI system. A realization is a block diagram whose blocks represent $\mathbb{F}_q$-adders, $\mathbb{F}_q$-multipliers, and $\mathbb{F}_q$-delay (memory) elements, which we take as our elementary LTI systems. An $\mathbb{F}_q$-adder may have any number of inputs in $\mathbb{F}_q$, and its output is their (instantaneous) sum. An $\mathbb{F}_q$-multiplier may have any number of inputs in $\mathbb{F}_q$, and its output is their (instantaneous) product. An $\mathbb{F}_q$-delay element has a single input in $\mathbb{F}_q$, and a single output which is equal to the input one time unit earlier.

For example, if an LTI system has impulse response $g(D) = 1 + \alpha D + \beta D^3$, then it can be realized by the realization shown in Figure 2. In this realization there are three delay (memory) elements, arranged as a shift register of length 3. It is easy to check that the input/output map is given by $y(D) = u(D) + \alpha D u(D) + \beta D^3 u(D) = u(D)g(D)$.

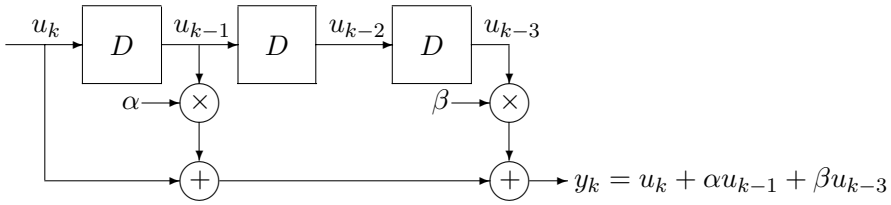


Figure 2. Realization of an LTI system with impulse response $g(D) = 1 + \alpha D + \beta D^3$.

More generally, it is easy to see that if $g(D)$ is finite (polynomial) with degree $\deg g(D) = \nu$, then an LTI system with impulse response $g(D)$ can be realized similarly, using a shift register of length ν.

The *state* of a realization at time k is the set of contents of its memory elements. For example, in Figure 2 the state at time k is the 3-tuple $(u_{k-1}, u_{k-2}, u_{k-3})$. The *state space* is the set of all possible states. If a realization over $\mathbb{F}_q$ has only a finite number ν of memory elements, then the state space has size q^ν, which is finite. For example, the state space size in Figure 2 is q^3.

Now suppose that we consider only realizations with a finite number of blocks; in particular, with a finite number of memory elements, and thus a finite state space size. What is the most general impulse response that we can realize? If the input is the impulse $e_0(D) = 1$, then the input is zero after time zero, so the impulse response is determined by the autonomous (zero-input) behavior after time zero. Since the state space size is finite, it is clear that the autonomous behavior must be periodic after time zero, and therefore the impulse response must be eventually periodic— *i.e.,* rational. We conclude that finite realizations can realize only rational impulse responses.

Conversely, given a rational impulse response $g(D) = n(D)/d(D)$, where $n(D)$ and $d(D) \neq 0$ are polynomial, it is straightforward to show that $g(D)$ can be realized with a finite number of memory elements, and in fact with $\nu = \max\{\deg n(D), \deg d(D)\}$ memory elements. For example, Figure 3 shows a realization of an LTI system with the impulse response $g(D) =$

$(1 + \alpha D + \beta D^3)/(1 - D^2 - D^3)$ using $\nu = 3$ memory elements. Because the impulse response is infinite, the realization necessarily involves feedback, in contrast to a feedbackfree realization of a finite impulse response as in Figure 2.

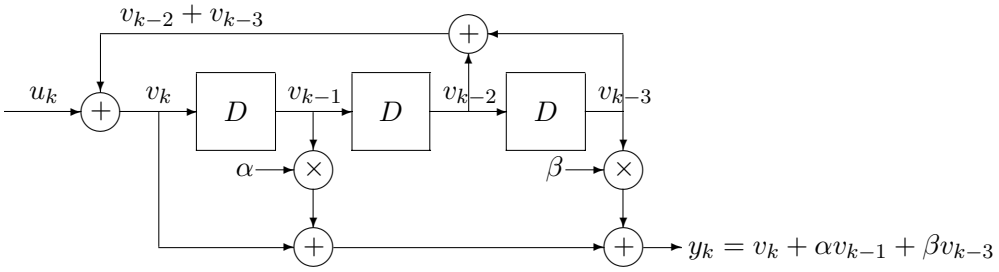


Figure 3. Realization of an LTI system with impulse response $g(D) = (1 + \alpha D + \beta D^3)/(1 - D^2 - D^3)$.

Exercise 2 (rational realizations). Generalize Figure 2 to realize any rational impulse response $g(D) = n(D)/d(D)$ with $\nu = \max\{\deg n(D), \deg d(D)\}$ memory elements. □

In summary:

Theorem 9.2 (finitely realizable = rational) *An LTI system with causal (thus Laurent) impulse response $g(D)$ has a finite realization if and only if $g(D)$ is rational. If $g(D) = n(D)/d(D)$, then there exists a realization with state space size q^ν, where $\nu = \max\{\deg n(D), \deg d(D)\}$. The realization can be feedbackfree if and only if $g(D)$ is polynomial.*

9.2 Rate-$1/n$ binary linear convolutional codes

A *rate-1/n binary linear convolutional encoder* is a single-input, n-output LTI system over the binary field $\mathbb{F}_2$. Such a system is characterized by the n impulse responses $\{g_j(D), 1 \le j \le n\}$, which can be written as an n-tuple $\mathbf{g}(D) = (g_1(D), \ldots, g_n(D))$.

If the input sequence $u(D)$ is Laurent, then the n output sequences $\{y_j(D),\ 1 \le j \le n\}$ are well defined and are given by $y_j(D) = u(D)g_j(D)$ More briefly, the output n-tuple $\mathbf{y}(D) = (y_1(D), \ldots, y_n(D))$ is given by $\mathbf{y}(D) = u(D)\mathbf{g}(D)$.

The encoder is *polynomial* (or "feedforward") if all impulse responses $g_j(D)$ are polynomial. In that case there is a shift-register realization of $\mathbf{g}(D)$ as in Figure 1 involving a single shift register of length $\nu = \deg \mathbf{g}(D) = \max_j \deg g_j(D)$. The encoder state space size is then 2^ν.

Example 1 (Rate-1/2 convolutional encoder). A rate-1/2 polynomial binary convolutional encoder is defined by a polynomial 2-tuple $\mathbf{g}(D) = (g_1(D), g_2(D))$. A shift-register realization of $\mathbf{g}(D)$ involves a single shift register of length $\nu = \max\{\deg g_1(D), \deg g_2(D)\}$ and has a state space of size 2^ν. For example, the impulse response 2-tuple $\mathbf{g}(D) = (1 + D^2, 1 + D + D^2)$ is realized by the 4-state rate-1/2 encoder illustrated in Figure 1. □

More generally, the encoder is *realizable* if all impulse responses $g_j(D)$ are causal and rational, since each response must be (finitely) realizable by itself, and we can obtain a finite realization of all of them by simply realizing each one separately.

A more efficient (and in fact minimal, although we will not show this yet) realization may be obtained as follows. If each $g_j(D)$ is causal and rational, then $g_j(D) = n_j(D)/d_j(D)$ for polynomials $n_j(D)$ and $d_j(D) \neq 0$, where by reducing to lowest terms, we may assume that $n_j(D)$ and $d_j(D)$ have no common factors. Then we can write

$$\mathbf{g}(D) = \frac{(n'_1(D), n'_2(D), \ldots, n'_n(D))}{d(D)} = \frac{\mathbf{n}'(D)}{d(D)},$$

where the common denominator polynomial $d(D)$ is the least common multiple of the denominator polynomials $d_j(D)$, and $\mathbf{n}'(D)$ and $d(D)$ have no common factors. In order that $\mathbf{g}(D)$ be causal, $d(D)$ cannot be divisible by D; *i.e.,* $d_0 = 1$.

Now, as the reader may verify by extending Exercise 2, this set of n impulse responses may be realized by a single shift register of length $\nu = \max\{\deg \mathbf{n}'(D), \deg d(D)\}$ memory elements, with feedback coefficients determined by the common denominator polynomial $d(D)$ as in Figure 3, and with the n outputs formed as n different linear combinations of the shift register contents, as in Figure 1. To summarize:

Theorem 9.3 (Rate-$1/n$ convolutional encoders) *If $\mathbf{g}(D) = (g_1(D), g_2(D), \ldots, g_n(D))$ is the set of n impulse responses of a rate-$1/n$ binary linear convolutional encoder, then there exists a unique denominator polynomial $d(D)$ with $d_0 = 1$ such that we can write each $g_j(D)$ as $g_j(D) = n_j(D)/d(D)$, where the numerator polynomials $n_j(D)$ and $d(D)$ have no common factor. There exists a (minimal) realization of $\mathbf{g}(D)$ with $\nu = \max\{\deg \mathbf{n}'(D), \deg d(D)\}$ memory elements and thus 2^ν states, which is feedbackfree if and only if $d(D) = 1$.*

9.2.1 Finite-state representations of convolutional codes

Because a convolutional encoder is a finite-state machine, it may be characterized by a finite *state-transition diagram*. For example, the encoder of Example 1 has the 4-state state-transition diagram shown in Figure 4(a). Each state is labelled by two bits representing the contents of the shift register, and each state transition is labelled by the two output bits associated with that transition.

Alternatively, a finite-state encoder may be characterized by a *trellis diagram*, which is simply a state-transition diagram with the states s_k at each time k shown separately. Thus there are transitions only from states s_k at time k to states s_{k+1} at time $k+1$. Figure 4(b) shows a segment of a trellis diagram for the encoder of Example 1, with states labelled as in Figure 4(a).

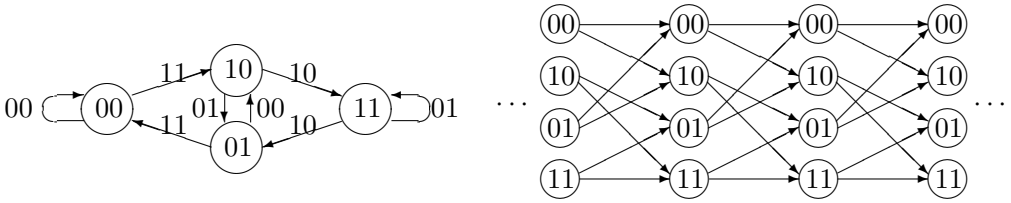


Figure 4. (a) Four-state state transition diagram; (b) Corresponding trellis diagram.

9.2.2 Rate-$1/n$ binary linear convolutional codes

A rate-$1/n$ binary linear convolutional code $\mathcal{C}$ is defined as the set of "all" output n-tuples $\mathbf{y}(D)$ that can be generated by a rate-$1/n$ binary linear convolutional encoder. If the encoder is characterized by an impulse response n-tuple $\mathbf{g}(D)$, then $\mathcal{C}$ is the set of output n-tuples $\mathbf{y}(D) = u(D)\mathbf{g}(D)$ as the input sequence $u(D)$ ranges over "all" possible input sequences.

How to define the set of "all" input sequences is actually a subtle question. We have seen that if $\mathbf{g}(D)$ is not polynomial, then we must restrict the input sequences $u(D)$ to be Laurent in order that the output $\mathbf{y}(D)$ be well-defined. On the other hand, $u(D)$ should be permitted to be infinite, because it can happen that some finite code sequences are generated by infinite input sequences (this phenomenon is called catastrophicity; see below).

The following definitions of the set of "all" input sequences meet both these criteria and are therefore OK; all have been used in the literature.

- The set $\mathbb{F}_2((D))$ of all formal Laurent series;
- The set $\mathbb{F}_2(D)$ of all rational functions;
- The set $\mathbb{F}_2[[D]]$ of all formal power series.

Here we will use the set $\mathbb{F}_2((D))$ of formal Laurent series. As we have seen, $\mathbb{F}_2((D))$ is a field under D-transform addition and multiplication, and includes the field $\mathbb{F}_2(D)$ of rational functions as a proper subfield. Since $\mathbf{g}(D)$ is rational, the set $\mathbb{F}_2(D)$ would suffice to generate all finite code sequences; however, we prefer $\mathbb{F}_2((D))$ because there seems no reason to constrain input sequences to be rational. We prefer either of these to $\mathbb{F}_2[[D]]$ because $\mathbb{F}_2((D))$ and $\mathbb{F}_2(D)$ are time-invariant, whereas $\mathbb{F}_2[[D]]$ is not; moreover, $\mathbb{F}_2[[D]]$ is not a field, but only a ring.

The *convolutional code* generated by $\mathbf{g}(D) \in (\mathbb{F}_2(D))^n$ will therefore be defined as

$$\mathcal{C} = \{\mathbf{y}(D) = u(D)\mathbf{g}(D), u(D) \in \mathbb{F}_2((D))\}.$$

The *rational subcode* of $\mathcal{C}$ is the set of all its rational sequences,

$$\mathcal{C}_r = \{\mathbf{y}(D) \in \mathcal{C} : \mathbf{y}(D) \in (\mathbb{F}_2(D))^n\},$$

and the *finite subcode* of $\mathcal{C}$ is the set of all its Laurent polynomial sequences,

$$\mathcal{C}_f = \{\mathbf{y}(D) \in \mathcal{C} : \mathbf{y}(D) \in (\mathbb{F}_2[D, D^{-1}])^n\}.$$

Since $\mathbb{F}_2[D, D^{-1}] \subset \mathbb{F}_2(D) \subset \mathbb{F}_2((D))$, we have $\mathcal{C}_f \subset \mathcal{C}_r \subset \mathcal{C}$.

Exercise 3 (input/output properties)

(a) Show that $\mathbf{y}(D)$ is an n-tuple of formal Laurent series, $\mathbf{y}(D) \in (\mathbb{F}_2((D)))^n$.

(b) Show that $\mathbf{y}(D)$ is rational if and only if $u(D)$ is rational; *i.e.,*

$$\mathcal{C}_r = \{\mathbf{y}(D) = u(D)\mathbf{g}(D), u(D) \in \mathbb{F}_2(D)\}.$$

(c) Show that $\mathbf{y}(D)$ is finite if and only if $u(D) = a(D)\text{lcm}\{d_j(D)\}/\gcd\{n_j(D)\}$, where $a(D)$ is finite, $\text{lcm}\{d_j(D)\}$ is the least common multiple of the denominators $d_j(D)$ of the $g_j(D)$, and $\gcd\{n_j(D)\}$ is the greatest common divisor of their numerators. □

A convolutional code $\mathcal{C}$ has the group property under D-transform addition, since if $\mathbf{y}(D) = u(D)\mathbf{g}(D)$ and $\mathbf{y}'(D) = u'(D)\mathbf{g}(D)$ are any two convolutional code n-tuples generated by two input sequences and $u(D)$ and $u'(D)$, respectively, then the n-tuple $\mathbf{y}(D) + \mathbf{y}'(D)$ is generated by the input sequence $u(D) + u'(D)$. It follows that $\mathcal{C}$ is a vector space over the binary field $\mathbb{F}_2$, an infinite-dimensional subspace of the infinite-dimensional vector space $(\mathbb{F}_2((D)))^n$ of all Laurent n-tuples.

At a higher level, a rate-$1/n$ convolutional code $\mathcal{C}$ is a one-dimensional subspace with *generator* $\mathbf{g}(D)$ of the n-dimensional vector space $(\mathbb{F}_2((D)))^n$ of all Laurent n-tuples over the Laurent field $\mathbb{F}_2((D))$. Similarly, the rational subcode $\mathcal{C}_r$ is a one-dimensional subspace with generator $\mathbf{g}(D)$ of the n-dimensional vector space $(\mathbb{F}_2(D))^n$ of all rational n-tuples over the rational field $\mathbb{F}_2(D)$. In these respects rate-$1/n$ convolutional codes are like $(n, 1)$ linear block codes.

9.2.3 Encoder equivalence

Two generator n-tuples $\mathbf{g}(D)$ and $\mathbf{g}'(D)$ will now be defined to be *equivalent* if they generate the same code, $\mathcal{C} = \mathcal{C}'$. We will shortly seek the best encoder to generate any given code $\mathcal{C}$.

Theorem 9.4 (Rate-$1/n$ encoder equivalence) *Two generator n-tuples $\mathbf{g}(D), \mathbf{g}'(D)$ are equivalent if and only if $\mathbf{g}(D) = u(D)\mathbf{g}'(D)$ for some nonzero rational function $u(D) \in \mathbb{F}_2(D)$.*

Proof. If the two encoders generate the same code, then $\mathbf{g}(D)$ must be a sequence in the code generated by $\mathbf{g}'(D)$, so we have $\mathbf{g}(D) = u(D)\mathbf{g}'(D)$ for some nonzero $u(D) \in \mathbb{F}_2((D))$. Moreover $\mathbf{g}(D)$ is rational, so from Exercise 3(b) $u(D)$ must be rational. Conversely, if $\mathbf{g}(D) = u(D)\mathbf{g}'(D)$ and $\mathbf{y}(D) \in \mathcal{C}$, then $\mathbf{y}(D) = v(D)\mathbf{g}(D)$ for some $v(D) \in F((D))$; thus $\mathbf{y}(D) = v(D)u(D)\mathbf{g}'(D)$ and $\mathbf{y}(D) \in \mathcal{C}'$, so $\mathcal{C} \subseteq \mathcal{C}'$. Since $\mathbf{g}'(D) = \mathbf{g}'(D)/u(D)$, a similar argument can be used in the other direction to show $\mathcal{C}' \subseteq \mathcal{C}$, so we can conclude $\mathcal{C} = \mathcal{C}'$. □

Thus let $\mathbf{g}(D) = (g_1(D), g_2(D), \ldots, g_n(D))$ be an arbitrary rational generator n-tuple. We can obtain an equivalent polynomial generator n-tuple by multiplying $\mathbf{g}(D)$ by any polynomial which is a multiple of all denominator polynomials $d_j(D)$, and in particular by their least common multiple $\text{lcm}\{d_j(D)\}$. Thus we have:

Corollary 9.5 *Every generator n-tuple $\mathbf{g}(D)$ is equivalent to a polynomial n-tuple $\mathbf{g}'(D)$.*

A generator n-tuple $\mathbf{g}(D)$ is called *delay-free* if at least one generator has a nonzero term at time index 0; *i.e.,* if $\mathbf{g} \neq \mathbf{0}$. If $\mathbf{g}(D)$ is not delay-free, then we can eliminate the delay by using instead the equivalent generator n-tuple $\mathbf{g}'(D) = \mathbf{g}(D)/D^{\text{del } \mathbf{g}(D)}$, where del $\mathbf{g}(D)$ is the smallest time index k such that $\mathbf{g}_k \neq \mathbf{0}$. Thus:

Corollary 9.6 *Every generator n-tuple $\mathbf{g}(D)$ is equivalent to a delay-free n-tuple $\mathbf{g}'(D)$, namely $\mathbf{g}'(D) = \mathbf{g}(D)/D^{\text{del } \mathbf{g}(D)}$.*

A generator n-tuple $\mathbf{g}(D)$ is called *catastrophic* if there exists an infinite input sequence $u(D)$ that generates a finite output n-tuple $\mathbf{y}(D) = u(D)\mathbf{g}(D)$. Any realization of $\mathbf{g}(D)$ must thus have a cycle in its state-transition diagram other than the zero-state self-loop such that the outputs are all zero during the cycle.

There is a one-to-one correspondence between the set of all paths through the trellis diagram of a convolutional encoder and the set $\{u(D)\}$ of all input sequences. However, the correspondence to the set of all output sequences $\{u(D)\mathbf{g}(D)\}$— *i.e.,* to the convolutional code $\mathcal{C}$ generated by the encoder— could be many-to-one. For example, if the encoder is catastrophic, then there are at least two paths corresponding to the all-zero output sequence, and from the group property of the encoder there will be at least two paths corresponding to all output sequences. In fact, the correspondence between convolutional code sequences and trellis paths is one-to-one if and only if $\mathbf{g}(D)$ is noncatastrophic.

By Exercise 3(c), $\mathbf{y}(D)$ is finite if and only if $u(D) = a(D)\text{lcm}\{d_j(D)\}/\gcd\{n_j(D)\}$ for some finite $a(D)$. Since $\text{lcm}\{d_j(D)\}$ is also finite, $u(D)$ can be infinite if and only if $\gcd\{n_j(D)\}$ has an infinite inverse. This is true if and only $\gcd\{n_j(D)\} \neq D^d$ for some integer d. In summary:

Theorem 9.7 (Catastrophicity) *A generator n-tuple $\mathbf{g}(D)$ is catastrophic if and only if all numerators $n_j(D)$ have a common factor other than D.*

Example 2. The rate-1/2 encoder $\mathbf{g}(D) = (1 + D^2, 1 + D + D^2)$ is noncatastrophic, since the polynomials $1 + D^2$ and $1 + D + D^2$ are relatively prime. However, the rate-1/2 encoder $\mathbf{g}(D) = (1 + D^2, 1 + D)$ is catastrophic, since $1 + D^2$ and $1 + D$ have the common divisor $1 + D$; thus the infinite input $1/(1 + D) = 1 + D + D^2 + \ldots$ leads to the finite output $\mathbf{y}(D) = (1 + D, 1)$ (due to a self-loop from the 11 state to itself with output 00). □

If $\mathbf{g}(D)$ is catastrophic with greatest common factor $\gcd\{n_j(D)\}$, then we can easily find an equivalent noncatastrophic generator n-tuple by dividing out the common factor: $\mathbf{g}'(D) = \mathbf{g}(D)/\gcd\{n_j(D)\}$. For instance, in Example 2, we should replace $(1 + D^2, 1 + D)$ by $(1 + D, 1)$.

In summary, we can and should eliminate denominators and common numerator factors:

Corollary 9.8 *Every generator n-tuple $\mathbf{g}(D)$ is equivalent to a unique (up to a scalar multiple) noncatastrophic delay-free polynomial n-tuple $\mathbf{g}'(D)$, namely*

$$\mathbf{g}'(D) = \frac{\text{lcm}\{d_j(D)\}}{\gcd\{n_j(D)\}}\mathbf{g}(D).$$

Thus every rate-$1/n$ binary convolutional cade $\mathcal{C}$ has a unique noncatastrophic delay-free polynomial generator n-tuple $\mathbf{g}(D)$, which is called *canonical*. Canonical generators have many nice properties, including:

- The finite subcode $\mathcal{C}_f$ is the subcode generated by the finite input sequences.
- The feedbackfree shift-register realization of $\mathbf{g}(D)$ is minimal over all equivalent encoders.

In general, it has been shown that a convolutional encoder is *minimal*— that is, has the minimal number of states over all encoders for the same code— if and only if it is noncatastrophic and moreover there are no state transitions to or from the zero state with all-zero outputs, other than from the zero state to itself. For a rate-$1/n$ encoder with generator $\mathbf{g}(D) = (n_1(D), \ldots, n_n(D))/d(D)$, this means that the numerator polynomials $n_j(D)$ must have no common factor, and that the degree of the denominator polynomial $d(D)$ must be no greater than the maximum degree of the numerators.

A *systematic* encoder for a rate-$1/n$ convolutional code is one in which the input sequence appears unchanged in one of the n output sequences. A systematic encoder for a code generated by polynomial encoder $\mathbf{g}(D) = (g_1(D), \ldots, g_n(D))$ may be obtained by multiplying all generators by $1/g_1(D)$ (provided that $g_{10} = 1$). For example, $\mathbf{g}(D) = (1, (1 + D + D^2)/(1 + D^2))$ is a systematic encoder for the code generated by $(1 + D^2, 1 + D + D^2)$.

A systematic encoder is always delay-free, noncatastrophic and minimal. Sometimes systematic encoders are taken as an alternative class of nonpolynomial canonical encoders.

9.2.4 Algebraic theory of rate-k/n convolutional codes

There is a more general theory of rate-k/n convolutional codes, the main result of which is as follows. Let $\mathcal{C}$ be any time-invariant subspace of the vector space of all $\mathbf{y}(D) \in (\mathbb{F}_q((D)))^n$. A *canonical polynomial encoder* for $\mathcal{C}$ may be constructed by the following greedy algorithm:

Initialization:	set $i = 0$ and $\mathcal{C}_0 = \{\mathbf{0}\}$;
Do loop:	if $\mathcal{C}_i = \mathcal{C}$, we are done, and $k = i$;
	otherwise, increase i by 1 and take any polynomial $\mathbf{y}(D)$ of
	least degree in $\mathcal{C} \setminus \mathcal{C}_i$ as $\mathbf{g}_i(D)$.

The shift-register realization of the resulting $k \times n$ generator matrix $G(D) = \{\mathbf{g}_i(D), 1 \le i \le k\}$ is then a minimal encoder for $\mathcal{C}$, in the sense that no other encoder for $\mathcal{C}$ has fewer memory elements. The degrees $\nu_i = \deg \mathbf{g}_i(D) = \max_{1 \le j \le n} \deg g_{ij}(D)$, $1 \le i \le k$, are uniquely determined by this construction, and are called the "constraint lengths" (or controllability indices) of $\mathcal{C}$; their maximum $\nu_{\max}$ is the (controller) memory of $\mathcal{C}$, and their sum $\nu = \sum_i \nu_i$ (the "overall constraint length") determines the size 2^ν of the state space of any minimal encoder for $\mathcal{C}$. Indeed, this encoder turns out to have every property one could desire, except for systematicity.

9.3 Terminated convolutional codes

A rate-$1/n$ *terminated convolutional code* $\mathcal{C}_\mu$ is the code generated by a polynomial convolutional encoder $\mathbf{g}(D)$ (possibly catastrophic) when the input sequence $u(D)$ is constrained to be a polynomial of degree less than some nonnegative integer μ:

$$\mathcal{C}_\mu = \{u(D)\mathbf{g}(D) \mid \deg u_i(D) < \mu\}.$$

In this case the total number of possibly nonzero input bits is $k = \mu$, and the total number of possibly nonzero output bits is $n' = n(\mu + \nu)$, where $\nu = \max \deg g_j(D)$ is the shift-register length in a shift-register realization of $\mathbf{g}(D)$. Thus a terminated convolutional code may be regarded as an $(n', k) = (n(\mu + \nu), \mu)$ binary linear block code.

The trellis diagram of a terminated convolutional code is finite. For example, Figure 5 shows the trellis diagram of the 4-state ($\nu = 2$) rate-1/2 encoder of Example 1 when $\mu = 5$, which yields a $(14, 5, 5)$ binary linear block code.

Exercise 4. Show that if the (catastrophic) rate-1/1 binary linear convolutional code generated by $g(D) = 1 + D$ is terminated with $\deg u(D) < \mu$, then the resulting code is a $(\mu + 1, \mu, 2)$ SPC code. Conclude that any binary linear SPC code may be represented by a 2-state trellis diagram. □

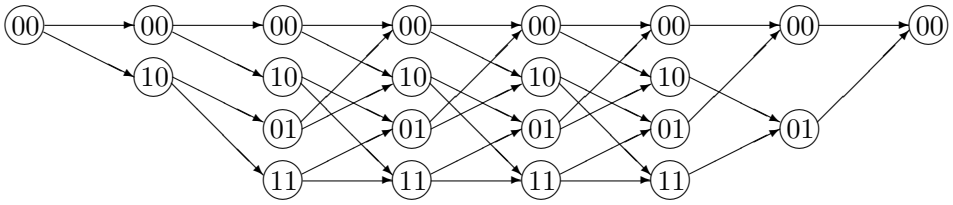


Figure 5. Trellis diagram of a terminated rate-1/2 convolutional encoder.

Exercise 5. Show that if the (catastrophic) rate-1/1 binary linear convolutional code generated by $g(D) = 1 + D + D^3$ is terminated with $\mu = 4$, then the resulting code is a $(7, 4, 3)$ Hamming code. □

9.4 Maximum likelihood sequence detection: The VA

A convolutional code sequence $\mathbf{y}(D) = u(D)\mathbf{g}(D)$ may be transmitted through an AWGN channel in the usual fashion, by mapping each bit $y_{jk} \in \{0, 1\}$ to $s(y_{jk}) \in \{\pm\alpha\}$ via the usual 2-PAM map. We denote the resulting sequence by $s(\mathbf{y}(D))$. At the output of the channel, the received sequence is

$$\mathbf{r}(D) = s(\mathbf{y}(D)) + \mathbf{n}(D),$$

where $\mathbf{n}(D)$ is an iid Gaussian noise sequence with variance $N_0/2$ per dimension.

As usual, maximum-likelihood sequence detection is equivalent to minimum-distance (MD) detection; *i.e.,* find the code sequence $\mathbf{y}(D)$ such that $\|\mathbf{r}(D) - s(\mathbf{y}(D))\|^2$ is minimum. In turn, since $s(\mathbf{y}(D))$ is binary, this is equivalent to maximum-inner-product (MIP) detection: find the code sequence $\mathbf{y}(D)$ that maximizes

$$\langle \mathbf{r}(D), s(\mathbf{y}(D)) \rangle = \sum_k \langle \mathbf{r}_k, s(\mathbf{y}_k) \rangle = \sum_k \sum_j r_{kj} s(y_{kj}).$$

It may not be immediately clear that this is a well-posed problem, since all sequences are bi-infinite. However, we will now develop a recursive algorithm that solves this problem for terminated convolutional codes; it will then be clear that the algorithm continues to specify a well-defined maximum-likelihood sequence detector as the length of the terminated convolutional code goes to infinity.

The algorithm is the celebrated *Viterbi algorithm* (VA), which some readers may recognize simply as "dynamic programming" applied to a trellis diagram.

The first observation is that if we assign to each branch in the trellis a "metric" equal to $\|\mathbf{r}_k - s(\mathbf{y}_k)\|^2$, or equivalently $-\langle \mathbf{r}_k, s(\mathbf{y}_k) \rangle$, where $\mathbf{y}_k$ is the output n-tuple associated with that branch, then since each path through the trellis corresponds to a unique code sequence, MD or MIP detection becomes equivalent to finding the least-metric ("shortest") path through the trellis diagram.

The second observation is that the initial segment of the shortest path, say from time 0 through time k, must be the shortest path to whatever state s_k it passes through at time k, since if there were any shorter path to s_k, it could be substituted for that initial segment to create a shorter path overall, contradiction. Therefore it suffices at time k to determine and retain for each state

s_k at time k only the shortest path from the unique state at time 0 to that state, called the "survivor."

The last observation is that the time-$(k+1)$ survivors may be determined from the time-k survivors by the following recursive "add-compare-select" operation:

(a) For each branch from a state at time k to a state at time $k+1$, add the metric of that branch to the metric of the time-k survivor to get a candidate path metric at time $k+1$;

(b) For each state at time $k+1$, compare the candidate path metrics arriving at that state and select the path corresponding to the smallest as the survivor. Store the survivor path history from time 0 and its metric.

At the final time $\mu+\nu$, there is a unique state, whose survivor must be the (or at least a) shortest path through the trellis diagram of the terminated code.

The Viterbi algorithm has a regular recursive structure that is attractive for software or hardware implementation. Its complexity is clearly proportional to the number of trellis branches per unit of time ("branch complexity"), which in the case of a binary rate-k/n 2^ν-state convolutional code is $2^{k+\nu}$.

For infinite trellises, there are a few additional issues that must be addressed, but none lead to problems in practice.

First, there is the issue of how to get started. In practice, if the algorithm is simply initiated with arbitrary accumulated path metrics at each state, say all zero, then it will automatically synchronize in a few constraint lengths.

Second, path histories need to be truncated and decisions put out after some finite delay. In practice, it has been found empirically that with high probability all survivors will have the same history prior to about five constraint lengths in the past, so that a delay of this magnitude usually suffices to make the additional probability of error due to premature decisions negligible, for any reasonable method of making final decisions.

In some applications, it is important that the decoded path be a true path in the trellis; in this case, when decisions are made it is important to purge all paths that are inconsistent with those decisions.

Finally, the path metrics must be renormalized from time to time so that their magnitude does not become too great; this can be done by subtracting the minimum metric from all of them.

In summary, the VA works perfectly well on unterminated convolutional codes. Maximum-likelihood sequence detection (MLSD) of unterminated codes may be operationally defined by the VA.

More generally, the VA is a general method for maximum-likelihood detection of the state sequence of any finite-state Markov process observed in memoryless noise. For example, it can be used for maximum-likelihood sequence detection of digital sequences in the presence of intersymbol interference.

9.5 Performance analysis of convolutional codes

The performance analysis of convolutional codes is based on the notion of an "error event."

Suppose that the transmitted code sequence is $\mathbf{y}(D)$ and the detected code sequence is $\mathbf{y}'(D)$. Each of these sequences specifies a unique path through a minimal code trellis. Typically these paths will agree for long periods of time, but will disagree over certain finite intervals. An error event corresponds to one of these finite intervals. It begins when the path $\mathbf{y}'(D)$ first diverges from the path $\mathbf{y}(D)$, and ends when these two paths merge again. The error sequence is the difference $\mathbf{e}(D) = \mathbf{y}'(D) - \mathbf{y}(D)$ over this interval.

By the group property of a convolutional code $\mathcal{C}$, such an error sequence $\mathbf{e}(D)$ is a finite code sequence in $\mathcal{C}$. If the encoder $\mathbf{g}(D)$ is noncatastrophic (or if the code is a terminated code), then $\mathbf{e}(D) = u_e(D)\mathbf{g}(D)$ for some finite input sequence $u_e(D)$.

The probability of any such finite error event in AWGN is given as usual by

$$\Pr(\mathbf{y}'(D) \mid \mathbf{y}(D)) = Q^{\surd}(\|s(\mathbf{y}'(D)) - s(\mathbf{y}(D))\|^2/2N_0).$$

As in the block code case, we have

$$\|s(\mathbf{y}'(D)) - s(\mathbf{y}(D))\|^2 = 4\alpha^2 d_H(\mathbf{y}', \mathbf{y}).$$

The minimum error event probability is therefore governed by the minimum Hamming distance between code sequences $\mathbf{y}(D)$. For convolutional codes, this is called the *free distance* $d_{\rm free}$.

By the group property of a convolutional code, $d_{\rm free}$ is equal to the minimum Hamming weight of any finite code sequence $\mathbf{y}(D)$; *i.e.,* a code sequence that starts and ends with semi-infinite all-zero sequences. In a minimal trellis, such a code sequence must start and end with semi-infinite all-zero state sequences. Thus $d_{\rm free}$ is simply the minimum weight of a trellis path that starts and ends in the zero state in a minimal trellis, which can easily be found by a search through the trellis using a version of the VA.

Example 1 (cont.) From the state-transition diagram of Figure 4(a) or the trellis diagram of Figure 4(b), it is clear that the the free distance of the example 4-state rate-1/2 code is $d_{\rm free} = 5$, and that the only code sequences with this weight are the generator sequence $\mathbf{g}(D) = (1 + D^2, 1 + D + D^2)$ and its time shifts $D^k\mathbf{g}(D)$. □

For a rate-k/n binary convolutional code, it makes sense to normalize the error probability per unit time of the code— *i.e.,* per k input or n output bits. The probability of an error event starting at a given time, assuming that no error event is already in progress at that time, may be estimated by the union bound estimate as

$$\begin{aligned} P_c(E) &\approx K_{\min}(\mathcal{C})Q^{\surd}(2\alpha^2 d_{\rm free}/N_0) && (9.2)\\ &= K_{\min}(\mathcal{C})Q^{\surd}(\gamma_{\rm c}(\mathcal{C})(2E_b/N_0)), && (9.3)\end{aligned}$$

where $E_b = n\alpha^2/k$ is the average energy per bit, and

$$\gamma_{\rm c}(\mathcal{C}) = d_{\rm free}(k/n) \tag{9.4}$$

is the *nominal coding gain* of the convolutional code $\mathcal{C}$, while $K_{\min}(\mathcal{C})$ is the number of error events of weight $d_{\rm free}$ in $\mathcal{C}$ per unit time. Note that the nominal coding gain $\gamma_{\rm c}(\mathcal{C}) = d_{\rm free}(k/n)$

is defined analogously to the block code case. Note also that for a time-invariant code $\mathcal{C}$, the error event probability $P_c(E)$ is independent of time.

For direct comparison to block codes, the error probability per bit may be estimated as:

$$P_b(E) = P_c(E)/k \approx K_b(\mathcal{C}) Q^{\surd}(\gamma_c(\mathcal{C})(2E_b/N_0)), \tag{9.5}$$

where the error coefficient is $K_b(\mathcal{C}) = K_{\min}(\mathcal{C})/k$.

Example 1 (cont.) The nominal coding gain of the Example 1 code is $\gamma_c(\mathcal{C}) = 5/2$ (4 dB) and its error coefficient is $K_b(\mathcal{C}) = K_c(\mathcal{C}) = 1$, so the UBE is

$$P_b(E) \approx Q^{\surd}(5E_b/N_0).$$

Compare the $(8, 4, 4)$ RM block code, which has the same rate (and the same trellis complexity), but has nominal coding gain $\gamma_c(\mathcal{C}) = 2$ (3 dB) and error coefficient $K_b(\mathcal{C}) = 14/4 = 3.5$. □

In general, for the same rate and "complexity" (measured by minimal trellis complexity), convolutional codes have better coding gains than block codes, and much lower error coefficients. They therefore usually yield a better performance *vs.* complexity tradeoff than block codes, when trellis-based ML decoding algorithms such as the VA are used for each. Convolutional codes are also more naturally suited to continuous sequential data transmission than block codes. Even when the application calls for block transmission, terminated convolutional codes usually outperform binary block codes. On the other hand, if there is a decoding delay constraint, then block codes will usually slightly outperform convolutional codes.

Tables 1-3 give the parameters of the best known convolutional codes of short to moderate constraint lengths for rates 1/2, 1/3 and 1/4, which are well suited to the power-limited regime.

We see that it is possible to achieve nominal coding gains of 6 dB with as few as 16 states with rate-1/3 or rate-1/4 codes, and effective coding gains of 6 dB with as few as 32 states, to the accuracy of the union bound estimate (which becomes somewhat optimistic as these codes become more complex). (There also exists a *time-varying* rate-1/2 16-state convolutional code with $d_{\text{free}} = 8$ and thus $\gamma_c = 4$ (6 dB).) With 128 or 256 states, one can achieve effective coding gains of the order of 7 dB. ML sequence detection using the Viterbi algorithm is not a big thing for any of these codes.

Table 9.1: Rate-1/2 binary linear convolutional codes

ν	d_{free}	γ_{c}	dB	K_b	γ_{eff}(dB)
1	3	1.5	1.8	1	1.8
2	5	2.5	4.0	1	4.0
3	6	3	4.8	2	4.6
4	7	3.5	5.2	4	4.8
5	8	4	6.0	5	5.6
6	10	5	7.0	46	5.9
6	9	4.5	6.5	4	6.1
7	10	5	7.0	6	6.7
8	12	6	7.8	10	7.1

Table 9.2: Rate-1/3 binary linear convolutional codes

ν	d_{free}	γ_{c}	dB	K_b	γ_{eff}(dB)
1	5	1.67	2.2	1	2.2
2	8	2.67	4.3	3	4.0
3	10	3.33	5.2	6	4.7
4	12	4	6.0	12	5.3
5	13	4.33	6.4	1	6.4
6	15	5	7.0	11	6.3
7	16	5.33	7.3	1	7.3
8	18	6	7.8	5	7.4

Table 9.3: Rate-1/4 binary linear convolutional codes

ν	d_{free}	γ_{c}	dB	K_b	γ_{eff}(dB)
1	7	1.75	2.4	1	2.4
2	10	2.5	4.0	2	3.8
3	13	3.25	5.1	4	4.7
4	16	4	6.0	8	5.6
5	18	4.5	6.5	6	6.0
6	20	5	7.0	37	6.0
7	22	5.5	7.4	2	7.2
8	24	6	7.8	2	7.6

9.6 More powerful codes and decoding algorithms

We conclude that by use of the optimal VA with moderate-complexity convolutional codes like those in Tables 1-3, we can close about 7 dB of the 12 dB gap between the Shannon limit and uncoded performance at error rates of the order of $P_b(E) \approx 10^{-6}$.

To get closer to the Shannon limit, one uses much longer and more powerful codes with suboptimal decoding algorithms. For instance:

1. *Concatenation* of an inner code with ML decoding with an outer Reed-Solomon (RS) code with algebraic decoding is a powerful approach. For the inner code, it is generally better to use a convolutional code with VA decoding than a block code.

2. A convolutional code with an arbitrarily long constraint length may be decoded by a recursive tree-search technique called *sequential decoding*. There exist various sequential decoding algorithms, of which the fastest is probably the *Fano algorithm*. In general, sequential algorithms follow the best code path through the code trellis (which becomes a tree for long constraint lengths) as long as the path metric exceeds its expected value for the correct path. When a wrong branch is taken, the path begins to look bad; the algorithm then backtracks and tries alternative paths until it again finds a good one.

The amount of decoding computation is a random variable with a Pareto (power-law) distribution; *i.e.,* Pr(number of computations $\geq N$) $\simeq N^{-\alpha}$. A Pareto distribution has a finite mean when the Pareto exponent α is greater than 1. The rate at which $\alpha = 1$ is called the computational cut-off rate R_0. On the AWGN channel, at low code rates, the computational cut-off rate occurs when E_b/N_0 is about 3 dB away from the Shannon limit; thus an effective coding gain of the order of 9 dB can be obtained at error rates of the order of $P_b(E) \approx 10^{-6}$.

3. In the past decade, these techniques have been superseded by capacity-approaching codes such as *turbo codes* and *low-density parity-check codes* with iterative decoding. Such powerful coding schemes will be the principal topic of the latter part of this course.

Chapter 10

Trellis representations of binary linear block codes

We now return to binary linear block codes and discuss trellis representations, particularly minimal trellis representations. The three main reasons for doing so are:

(a) Trellis-based (Viterbi algorithm) decoding is one of the most efficient methods known for maximum-likelihood (ML) decoding of general binary linear block codes;

(b) The complexity of a minimal trellis gives a good measure of the complexity of a code, whereas the parameters (n, k, d) do not;

(c) Trellis representations are the simplest class of graphical representations of codes, which will be a central concept in our later discussion of capacity-approaching codes.

The topic of trellis complexity of block codes was an active research area in the 1990s. We will summarize its main results. For an excellent general review, see [A. Vardy, "Trellis structure of codes," in HANDBOOK OF CODING THEORY, Elsevier, 1998.]

10.1 Definition

We saw in the previous chapter that certain binary linear block codes could be represented as terminated convolutional codes, and therefore have trellis representations.

Example 1. (SPC codes) Any $(n, n-1, 2)$ single-parity-check (SPC) code has a two-state trellis representation like that shown in Figure 1 (see Exercise 9.4).

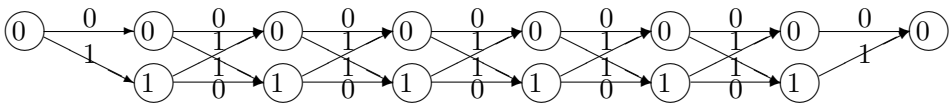


Figure 1. Two-state trellis for a binary $(n, n-1, 2)$ single-parity-check code $(n = 7)$.

In this chapter, we will show that all binary linear block codes have trellis (finite-state) representations. Indeed, we will show how to find minimal trellis representations.

In general, a trellis representation of a block code is a directed graph like that of Figure 1 in which there is one starting state (root) at time 0, one ending state (goal, "toor") at time N, and state spaces $\mathcal{S}_k$ of size greater than one at all $N-1$ intermediate times k, $1 \leq k < N$. All edges (branches, state transitions) go from a state at some time k to a state at the next time $k+1$. Each edge is labelled by an n-tuple of output symbols. The set of all possible codewords is in one-to-one correspondence with the set of all possible paths through the trellis. The codeword associated with any particular path is the sequence of corresponding n-tuple labels.

Example 2. ((8, 4, 4) RM code.) Figure 2 shows a trellis representation of an $(8, 4, 4)$ binary Reed-Muller code with generators $\{11110000, 10101010, 11001100, 11111111\}$. Here two output symbols (bits) are associated with each branch. It can be seen that each of the 16 codewords in this code corresponds to a unique path through this trellis.

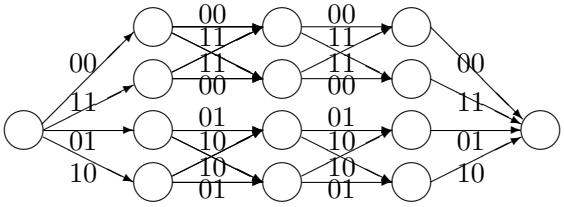


Figure 2. Four-state trellis for $(8, 4, 4)$ Reed-Muller code.

Given a trellis representation of a block code and a sequence of received symbol log likelihoods, the Viterbi algorithm (VA) may be used to find the trellis path with greatest log likelihood— *i.e.,* to perform ML decoding of the code.

There are various measures of the complexity of a trellis, which are all related to VA decoding complexity. The most common is perhaps the *state complexity profile*, which is just the sequence of state space sizes. For example, the state complexity profile of the trellis of Figure 1 is $\{1, 2, 2, 2, 2, 2, 2, 1\}$, and that of the trellis of Figure 2 is $\{1, 4, 4, 4, 1\}$. The *state complexity* of a trellis is often defined as the maximum state space size; *e.g.,* 2 or 4, respectively, for these two trellises. The *branch complexity profile* is the sequence of numbers of branches in each section— *e.g.,* $\{2, 4, 4, 4, 4, 4, 2\}$ and $\{4, 8, 8, 4\}$ for the trellises of Figures 1 and 2, respectively— and the *branch complexity* is the maximum number of branches in any section— *e.g.,* 4 or 8, respectively. The complexity of VA decoding is more precisely measured by the branch complexity profile, but any of these measures gives a good idea of VA decoding complexity.[1]

10.2 Minimal trellises and the state space theorem

A *minimal trellis* for a given linear block code will be defined as a trellis that has the minimum possible state space size at each time. It is not immediately clear that there exists a single trellis that minimizes every state space size, but for linear block codes, we will see that there exists a canonical minimal trellis with this property. It turns out that the canonical minimal trellis also minimizes every other trellis complexity measure, including VA decoding complexity, so that we need not worry whether we are minimizing the right quantity.

[1]The most precise measure of VA decoding complexity is $2|E| - |V| + 1$, where $|E|$ and $|V|$ denote the numbers of edges and vertices in the graph of the trellis, respectively.

10.2.1 The state space theorem for linear block codes

The state space theorem is an easy but fundamental theorem that sets a lower bound on the state space size of a trellis for a linear block code at any time, and also indicates how to construct a canonical minimal trellis. For simplicity, we will prove it here only for binary linear block codes, but it should be clear how it generalizes to arbitrary linear codes. (In fact, it generalizes to arbitrary codes over groups.)

The reader should think of a trellis as defining a state-space realization of a time-varying finite-state discrete-time linear system, where these terms are used as in system theory. The time axis $\mathcal{I}$ of the system is some subinterval of the integers $\mathbb{Z}$; *e.g.,* a finite subinterval $\mathcal{I} = [0, N] \subseteq \mathbb{Z}$.

In a state-space realization, a state space $\mathcal{S}_k$ is defined at each time $k \in \mathcal{I}$. The defining property of a state space is the *Markov property*:

Markov property. The state space $\mathcal{S}_k$ of a system at time k has the Markov property if, given that the system is in a certain state $s_k \in \mathcal{S}_k$ at time k, its possible future trajectories depend only on s_k and not otherwise on the previous history of the system. In other words, the state $s_k \in \mathcal{S}_k$ is a sufficient statistic for the past with respect to prediction of possible futures.

Let the **past** at time k be the set $\mathcal{P} = (-\infty, k) \cap \mathcal{I}$ of time indices in $\mathcal{I}$ prior to time k, and the **future** at time k be the set $\mathcal{F} = [k, \infty) \cap \mathcal{I}$ of time indices at time k or later. Given a linear code $\mathcal{C}$, the state space theorem may be expressed in terms of certain linear codes defined on the past $\mathcal{P}$ and future $\mathcal{F}$, as follows.

Given any subset $\mathcal{J} \subseteq \mathcal{I}$, the **subcode** $\mathcal{C}_\mathcal{J}$ is defined as the set of codewords whose components are equal to 0 on the complement of $\mathcal{J}$ in $\mathcal{I}$. It is possible to think of $\mathcal{C}_\mathcal{J}$ either as a code of length $|\mathcal{J}|$ or as a code of length $|\mathcal{I}|$ in which all symbols not in $\mathcal{J}$ equal 0. It should be clear from the context which of these two viewpoints is used.

The subcode $\mathcal{C}_\mathcal{J}$ is evidently a subset of the codewords in $\mathcal{C}$ that has the group property, and therefore is a linear code. In coding theory, $\mathcal{C}_\mathcal{J}$ is called a *shortened code* of $\mathcal{C}$. Because $\mathcal{C}_\mathcal{J} \subseteq \mathcal{C}$, the minimum Hamming distance of $\mathcal{C}_\mathcal{J}$ is at least as great as that of $\mathcal{C}$.

Similarly, given $\mathcal{J} \subseteq \mathcal{I}$, the **projection** $\mathcal{C}_{|\mathcal{J}}$ is defined as the set of all projections of codewords of $\mathcal{C}$ onto $\mathcal{J}$. By projection, we mean either zeroing of all coordinates whose indices are not in $\mathcal{J}$, or throwing away (puncturing) all such coordinates. Correspondingly, it is possible to think of $\mathcal{C}_{|\mathcal{J}}$ either as a code of length $|\mathcal{J}|$ or as a code of length $|\mathcal{I}|$ in which all symbols not in $\mathcal{J}$ equal 0. In coding theory, $\mathcal{C}_{|\mathcal{J}}$ is called a *punctured code* of $\mathcal{C}$.

The projection $\mathcal{C}_{|\mathcal{J}}$ evidently inherits the group property from $\mathcal{C}$, and therefore is also a linear code defined on $\mathcal{J}$. Moreover, $\mathcal{C}_\mathcal{J}$ is evidently a subcode of $\mathcal{C}_{|\mathcal{J}}$.

Example 2 (cont.) For the $(8, 4, 4)$ code illustrated in Figure 2, regarding the "past" as the first two time units or first four bits, the subcode $\mathcal{C}_\mathcal{P}$ consists of the two codewords $\mathcal{C}_\mathcal{P} = \{00000000, 11110000\}$. This code may be regarded as effectively a $(4, 1, 4)$ binary repetition code defined on the past subinterval $\mathcal{P} = [0, 1, 2, 3]$. The projection on this subinterval is the set $\mathcal{C}_{|\mathcal{P}} = \{0000, 0011, 1100, 1111, 0101, 0110, 1001, 1010\}$, which is a $(4, 3, 2)$ binary linear SPC code that has the $(4, 1, 4)$ code as a subcode. □

Now for any time $k \in \mathcal{I}$, let $\mathcal{P}$ and $\mathcal{F}$ denote the past and future subintervals with respect to k, and let $\mathcal{C}_\mathcal{P}, \mathcal{C}_{|\mathcal{P}}, \mathcal{C}_\mathcal{F}$ and $\mathcal{C}_{|\mathcal{F}}$ be the past and future subcodes and projections, respectively.

Then $\mathcal{C}$ must have a generator matrix of the following form:

$$\begin{bmatrix} G(\mathcal{C}_\mathcal{P}) & 0 \\ 0 & G(\mathcal{C}_\mathcal{F}) \\ G(\mathcal{C}_{|\mathcal{P}}/\mathcal{C}_\mathcal{P}) & G(\mathcal{C}_{|\mathcal{F}}/\mathcal{C}_\mathcal{F}) \end{bmatrix},$$

where $[G(\mathcal{C}_\mathcal{P}), 0]$ is a generator matrix for the past subcode $\mathcal{C}_\mathcal{P}$, $[0, G(\mathcal{C}_\mathcal{F})]$ is a generator matrix for the future subcode $\mathcal{C}_\mathcal{F}$, and $[G(\mathcal{C}_{|\mathcal{P}}/\mathcal{C}_\mathcal{P}), G(\mathcal{C}_{|\mathcal{F}}/\mathcal{C}_\mathcal{F})]$ is an additional set of linearly independent generators that together with $[G(\mathcal{C}_\mathcal{P}), 0]$ and $[0, G(\mathcal{C}_\mathcal{F})]$ generate $\mathcal{C}$. Moreover, it is clear from the form of this generator matrix that $G(\mathcal{C}_\mathcal{P})$ and $G(\mathcal{C}_{|\mathcal{P}}/\mathcal{C}_\mathcal{P})$ together generate the past projection $\mathcal{C}_{|\mathcal{P}}$, and that $G(\mathcal{C}_\mathcal{F})$ and $G(\mathcal{C}_{|\mathcal{F}}/\mathcal{C}_\mathcal{F})$ generate the future projection $\mathcal{C}_{|\mathcal{F}}$.

The **state code** $\mathcal{S}$ will be defined as the linear code generated by this last set of generators, $[G(\mathcal{C}_{|\mathcal{P}}/\mathcal{C}_\mathcal{P}), G(\mathcal{C}_{|\mathcal{F}}/\mathcal{C}_\mathcal{F})]$. The dimension of the state code is evidently

$$\dim \mathcal{S} = \dim \mathcal{C} - \dim \mathcal{C}_\mathcal{P} - \dim \mathcal{C}_\mathcal{F}.$$

Moreover, by the definitions of $\mathcal{C}_\mathcal{P}$ and $\mathcal{C}_\mathcal{F}$, the state code cannot contain any codewords that are all-zero on the past or on the future. This implies that the projections $\mathcal{S}_{|\mathcal{P}}$ and $\mathcal{S}_{|\mathcal{F}}$ on the past and future are linear codes with the same dimension as $\mathcal{S}$, so we also have

$$\begin{aligned} \dim \mathcal{S} &= \dim \mathcal{S}_{|\mathcal{P}} = \dim \mathcal{C}_{|\mathcal{P}} - \dim \mathcal{C}_\mathcal{P}; \\ \dim \mathcal{S} &= \dim \mathcal{S}_{|\mathcal{F}} = \dim \mathcal{C}_{|\mathcal{F}} - \dim \mathcal{C}_\mathcal{F}. \end{aligned}$$

Example 2 (cont.) Taking $\mathcal{P}$ as the first four symbols and $\mathcal{F}$ as the last four, the $(8, 4, 4)$ code has a generator matrix

$$\begin{bmatrix} 1111 & 0000 \\ 0000 & 1111 \\ 1010 & 1010 \\ 1100 & 1100 \end{bmatrix}.$$

Here the state code $\mathcal{S}$ is generated by the last two generators and has dimension 2. The past and future projections of the state code evidently also have dimension 2. □

In view of the generator matrix above, any codeword $\mathbf{c} \in \mathcal{C}$ may be expressed uniquely as the sum of a past codeword $\mathbf{c}_\mathcal{P} \in \mathcal{C}_\mathcal{P}$, a future codeword $\mathbf{c}_\mathcal{F} \in \mathcal{C}_\mathcal{F}$, and a state codeword $\mathbf{s} \in \mathcal{S}$. We then say that the codeword $\mathbf{c}$ is associated with the state codeword $\mathbf{s}$. This allows us to conclude that the state codewords have the Markov property, and thus may be taken as states:

Lemma 10.1 (Markov property) *For all codewords $\mathbf{c} \in \mathcal{C}$ that are associated with a given state codeword $\mathbf{s} \in \mathcal{S}$, the past projection $\mathbf{c}_{|\mathcal{P}}$ has the same set of possible future trajectories. On the other hand, if two codewords $\mathbf{c}$ and $\mathbf{c}'$ are associated with different state codewords, then the sets of possible future trajectories of $\mathbf{c}_{|\mathcal{P}}$ and $\mathbf{c}'_{|\mathcal{P}}$ are disjoint.*

Proof. If $\mathbf{c}$ is associated with $\mathbf{s}$, then $\mathbf{c} = \mathbf{c}_\mathcal{P} + \mathbf{c}_\mathcal{F} + \mathbf{s}$ for some $\mathbf{c}_\mathcal{P} \in \mathcal{C}_\mathcal{P}$ and $\mathbf{c}_\mathcal{F} \in \mathcal{C}_\mathcal{F}$. Hence $\mathbf{c}_{|\mathcal{P}} = (\mathbf{c}_\mathcal{P})_{|\mathcal{P}} + \mathbf{s}_{|\mathcal{P}}$ and $\mathbf{c}_{|\mathcal{F}} = (\mathbf{c}_\mathcal{F})_{|\mathcal{F}} + \mathbf{s}_{|\mathcal{F}}$. Thus, for every such $\mathbf{c}_{|\mathcal{P}}$, the set of possible future trajectories is the same, namely the coset $\mathcal{C}_\mathcal{F} + \mathbf{s}_{|\mathcal{F}} = \{(\mathbf{c}_\mathcal{F})_{|\mathcal{F}} + \mathbf{s}_{|\mathcal{F}} \mid \mathbf{c}_\mathcal{F} \in \mathcal{C}_\mathcal{F}\}$.

If $\mathbf{c}$ and $\mathbf{c}'$ are associated with different state codewords $\mathbf{s}$ and $\mathbf{s}'$, then the sets of possible future trajectories are $\mathcal{C}_\mathcal{F} + \mathbf{s}_{|\mathcal{F}}$ and $\mathcal{C}_\mathcal{F} + \mathbf{s}'_{|\mathcal{F}}$, respectively; these cosets are disjoint because the difference $\mathbf{s}_{|\mathcal{F}} - \mathbf{s}'_{|\mathcal{F}}$ is not a codeword in $\mathcal{C}_\mathcal{F}$. □

In other words, a codeword $\mathbf{c} \in \mathcal{C}$ is in the coset $\mathcal{C}_\mathcal{P} + \mathcal{C}_\mathcal{F} + \mathbf{s}$ if and only if $\mathbf{c}_{|\mathcal{P}} \in \mathcal{C}_{|\mathcal{P}}$ is in the coset $\mathcal{C}_\mathcal{P} + \mathbf{s}_{|\mathcal{P}}$ and $\mathbf{c}_{|\mathcal{F}} \in \mathcal{C}_{|\mathcal{F}}$ is in the coset $\mathcal{C}_\mathcal{F} + \mathbf{s}_{|\mathcal{F}}$.

This yields the following picture. The set of past projections associated with a given state codeword $\mathbf{s}$ is $\mathcal{C}_\mathcal{P} + \mathbf{s}_{|\mathcal{P}} = \{(\mathbf{c}_\mathcal{P})_{|\mathcal{P}} + \mathbf{s}_{|\mathcal{P}} \mid \mathbf{c}_\mathcal{P} \in \mathcal{C}_\mathcal{P}\}$. For any past projection in this set, we have the same set of possible future trajectories, namely $\mathcal{C}_\mathcal{F} + \mathbf{s}_{|\mathcal{F}}$. Moreover, these past and future subsets are disjoint. Therefore the entire code may be written as a disjoint union of Cartesian products of past and future subsets:

$$\mathcal{C} = \bigcup_{\mathbf{s}\in\mathcal{S}} \left(\mathcal{C}_\mathcal{P} + \mathbf{s}_{|\mathcal{P}}\right) \times \left(\mathcal{C}_\mathcal{F} + \mathbf{s}_{|\mathcal{F}}\right).$$

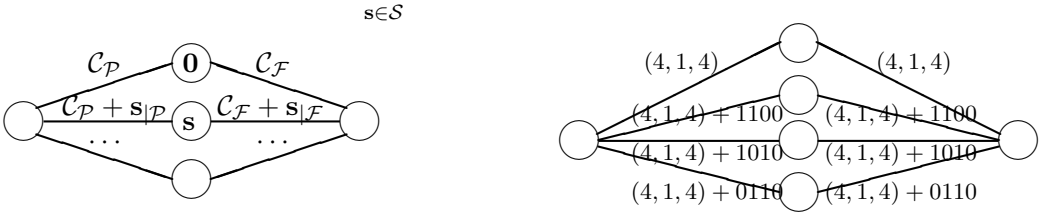


Figure 3. (a) Two-section trellis for generic code; (b) two-section trellis for $(8, 4, 4)$ code.

Figure 3(a) is a two-section trellis that illustrates this Cartesian-product decomposition for a general code. The states are labelled by the state codewords $\mathbf{s} \in \mathcal{S}$ to which they correspond. Each edge represents an entire set of parallel projections, namely $\mathcal{C}_\mathcal{P}+\mathbf{s}_{|\mathcal{P}}$ for past edges and $\mathcal{C}_\mathcal{F}+\mathbf{s}_{|\mathcal{F}}$ for future edges, and is therefore drawn as a thick line. The particular path corresponding to $\mathbf{s} = \mathbf{0}$ is shown, representing the subcode $\mathcal{C}_\mathcal{P} + \mathcal{C}_\mathcal{F}$, as well as a generic path corresponding to a general state $\mathbf{s}$, representing $\mathcal{C}_\mathcal{P} + \mathcal{C}_\mathcal{F} + \mathbf{s}$.

Example 2 (cont.) Figure 3(b) is a similar illustration for our running example $(8, 4, 4)$ code. Here $\mathcal{C}_\mathcal{P} = \mathcal{C}_\mathcal{F} = (4, 1, 4) = \{0000, 1111\}$, so each edge represents two paths that are binary complements of one another. The reader should compare Figure 3(b) to Figure 2 and verify that these two trellises represent the same code. □

It should now be clear that the code $\mathcal{C}$ has a trellis representation with a state space $\mathcal{S}_k$ at time k that is in one-to-one correspondence with the state code $\mathcal{S}$, and it has no trellis representation with fewer states at time k. Indeed, Figure 3(a) exhibits a two-section trellis with a state space $\mathcal{S}_k$ such that $|\mathcal{S}_k| = |\mathcal{S}|$. The converse is proved by noting that no two past projections $\mathbf{c}_{|\mathcal{P}}, \mathbf{c}'_{|\mathcal{P}}$ that do not go to the same state in Figure 3(a) can go to the same state in any trellis for $\mathcal{C}$, because by Lemma 10.1 they must have different sets of future continuations.

We summarize this development in the following theorem:

Theorem 10.2 (State space theorem for binary linear block codes) *Let $\mathcal{C}$ be a binary linear block code defined on a time axis $\mathcal{I} = [0, N]$, let $k \in \mathcal{I}$, and let $\mathcal{C}_\mathcal{P}, \mathcal{C}_{|\mathcal{P}}, \mathcal{C}_\mathcal{F}$ and $\mathcal{C}_{|\mathcal{F}}$ be the past and future subcodes and projections relative to time k, respectively. Then there exists a trellis representation for $\mathcal{C}$ with $|\mathcal{S}_k| = 2^{(\dim \mathcal{S})}$ states at time k, and no trellis representation with fewer states, where $\dim \mathcal{S}$ is given by any of the following three expressions:*

$$\begin{aligned} \dim \mathcal{S} &= \dim \mathcal{C} - \dim \mathcal{C}_\mathcal{P} - \dim \mathcal{C}_\mathcal{F}; \\ &= \dim \mathcal{C}_{|\mathcal{P}} - \dim \mathcal{C}_\mathcal{P}; \\ &= \dim \mathcal{C}_{|\mathcal{F}} - \dim \mathcal{C}_\mathcal{F}. \end{aligned}$$ □

We note that if we subtract the first expression for $\dim \mathcal{S}$ from the sum of the other two, then we obtain yet another expression:

$$\dim \mathcal{S} = \dim \mathcal{C}_{|\mathcal{P}} + \dim \mathcal{C}_{|\mathcal{F}} - \dim \mathcal{C}.$$

Exercise 1. Recall the $|u|u+v|$ construction of a Reed-Muller code $\mathrm{RM}(r,m)$ with length $n=2^m$ and minimum distance $d=2^{m-r}$:

$$\mathrm{RM}(r,m)=\{(\mathbf{u},\mathbf{u}+\mathbf{v}) \mid \mathbf{u}\in \mathrm{RM}(r,m-1), \mathbf{v}\in \mathrm{RM}(r-1,m-1)\}.$$

Show that if the past $\mathcal{P}$ is taken as the first half of the time axis and the future $\mathcal{F}$ as the second half, then the subcodes $\mathcal{C}_\mathcal{P}$ and $\mathcal{C}_\mathcal{F}$ are both effectively equal to $\mathrm{RM}(r-1,m-1)$ (which has the same minimum distance $d=2^{m-r}$ as $\mathrm{RM}(r,m)$), while the projections $\mathcal{C}_{|\mathcal{P}}$ and $\mathcal{C}_{|\mathcal{F}}$ are both equal to $\mathrm{RM}(r,m-1)$. Conclude that the dimension of the minimal central state space of $\mathrm{RM}(r,m)$ is

$$\dim \mathcal{S}=\dim \mathrm{RM}(r,m-1)-\dim \mathrm{RM}(r-1,m-1).$$

Evaluate $\dim \mathcal{S}$ for all RM codes with length $n\le 32$.

Similarly, show that if the past $\mathcal{P}$ is taken as the first quarter of the time axis and the future $\mathcal{F}$ as the remaining three quarters, then the subcode $\mathcal{C}_\mathcal{P}$ is effectively equal to $\mathrm{RM}(r-2,m-2)$, while the projection $\mathcal{C}_{|\mathcal{P}}$ is equal to $\mathrm{RM}(r,m-2)$. Conclude that the dimension of the corresponding minimal state space of $\mathrm{RM}(r,m)$ is

$$\dim \mathcal{S}=\dim \mathrm{RM}(r,m-2)-\dim \mathrm{RM}(r-2,m-2).$$

Using the relation $\dim \mathrm{RM}(r,m)=\dim \mathrm{RM}(r,m-1)+\dim \mathrm{RM}(r-1,m-1)$, show that $\dim \mathrm{RM}(r,m-2)-\dim \mathrm{RM}(r-2,m-2)=\dim \mathrm{RM}(r,m-1)-\dim \mathrm{RM}(r-1,m-1)$. □

Exercise 2. Recall that the dual code to a binary linear (n,k,d) block code $\mathcal{C}$ is defined as the orthogonal subspace $\mathcal{C}^\perp$, namely the set of of all n-tuples that are orthogonal to all codewords in $\mathcal{C}$, and that $\mathcal{C}^\perp$ is a binary linear block code whose dimension is $\dim \mathcal{C}^\perp=n-k$.

Show that for any partition of the time axis $\mathcal{I}$ of $\mathcal{C}$ into past $\mathcal{P}$ and future $\mathcal{F}$, the subcode $(\mathcal{C}^\perp)_\mathcal{P}$ is equal to the dual $(\mathcal{C}_{|\mathcal{P}})^\perp$ of the projection $\mathcal{C}_{|\mathcal{P}}$, and *vice versa*. [Hint: notice that $(\mathbf{a},\mathbf{0})$ is orthogonal to $(\mathbf{b},\mathbf{c})$ if and only if $\mathbf{a}$ is orthogonal to $\mathbf{b}$.]

Conclude that the minimal state spaces of $\mathcal{C}$ and $\mathcal{C}^\perp$ at any time k have the same size. □

10.2.2 Canonical minimal trellis representation

We now show that we can construct a single trellis for $\mathcal{C}$ such that the state space is minimal according to the state space theorem at each time k.

The construction is straightforwardly based on the development of the previous subsection. We define a state space $\mathcal{S}_k$ at each time $k\in\mathcal{I}=[0,N]$ corresponding to the state code $\mathcal{S}$ at time k. Note that $\mathcal{S}_0$ is trivial, because $\mathcal{C}_{|\mathcal{F}}=\mathcal{C}_\mathcal{F}=\mathcal{C}$, so $\dim \mathcal{S}_0=0$; similarly $\dim \mathcal{S}_N=0$.

Every codeword $\mathbf{c}\in\mathcal{C}$ passes through a definite state in $\mathcal{S}_k$ corresponding to the state codeword $\mathbf{s}\in\mathcal{S}$ in the unique decomposition $\mathbf{c}=\mathbf{c}_\mathcal{P}+\mathbf{c}_\mathcal{F}+\mathbf{s}$. It thus goes through a well-defined sequence of states $(s_0,s_1,\ldots,s_N)$, which defines a certain path through the trellis. Each edge in each such path is added to the trellis. The result must be a trellis representation of $\mathcal{C}$.

Example 1 (cont.) We can now see that the two-state trellis of Figure 1 is a canonical minimal trellis. For each time $k, 0<k<N$, $\mathcal{C}_\mathcal{P}$ is the set of all even-weight codewords that are all-zero on $\mathcal{F}$, which is effectively a $(k,k-1,2)$ SPC code, and $\mathcal{C}_{|\mathcal{P}}$ is the universe $(k,k,1)$ code consisting of all binary k-tuples. There are thus two states at each time $k, 0<k<N$, one reached by all even-weight pasts, and the other by all odd-weight pasts. A given codeword passes through a well-defined sequence of states, and makes a transition from the zero (even) to the nonzero (odd) state or *vice versa* whenever it has a symbol equal to 1. □

Example 2 (cont.) Similarly, the four-state trellis of Figure 2 is a canonical minimal trellis. For $k = 1$, $\mathcal{C}_\mathcal{P}$ is the trivial $(2, 0, \infty)$ code, so each of the four past projections at time $k = 1$ leads to a different state. The same is true at time $k = 3$ for future projections. Each of the 16 codewords goes through a well-defined state sequence $(s_0, s_1, s_2, s_3, s_4)$, and the set of all these sequences defines the trellis.

10.2.3 Trellis-oriented generator matrices

It is very convenient to have a generator matrix for $\mathcal{C}$ from which a minimal trellis for $\mathcal{C}$ and its parameters can be read directly. In this subsection we define such a generator matrix, show how to find it, and give some of its properties.

The *span* of a codeword will be defined as the interval from its first to last nonzero symbols. Its *effective length* will be defined as the length of this span.

A *trellis-oriented* (or *minimum-span*) generator matrix will be defined as a set of $k = \dim \mathcal{C}$ linearly independent generators whose effective lengths are as short as possible.

Concretely, a trellis-oriented generator matrix may be found by first finding all codewords with effective length 1, then all codewords of effective length 2 that are not linearly dependent on codewords of effective length 1, ..., all codewords of effective length i that are not linearly dependent on codewords of lower effective length, ..., until we have k independent generators.

The following theorem shows how to check whether a given generator matrix is trellis-oriented, and also suggests how to reduce any given generator matrix to one that is trellis-oriented.

Theorem 10.3 (Trellis-oriented generator matrices) *A set of k linearly independent generators is a trellis-oriented generator matrix if and only if the starting times of all spans are distinct and the ending times of all spans are distinct.*

Proof. If all starting times and ending times are distinct, then given a linear combination (with nonzero coefficients) of a certain subset of generators, the starting time and ending time of the combination are the least and greatest starting and ending times of the given subset of generators. It follows that the generators that combine to form any non-generator codeword have effective lengths no greater than the effective length of that codeword, so the given generators are indeed a set of generators whose effective lengths are as short as possible.

Conversely, if two starting or ending times are not distinct, then the sum of the two corresponding generators is a codeword whose effective length is shorter than that of at least one of the two generators. If this generator is replaced by this codeword, then we obtain a set of linearly independent generators of which one has a shorter effective length, so the original set was not trellis-oriented. □

The second part of the proof suggests a simple greedy algorithm for finding a trellis-oriented generator matrix from a given generator matrix. If the starting or ending times of two generators in the given matrix are not distinct, then replace the generator with greater effective length by the sum of the two generators, which must necessarily have a shorter effective length. This algorithm reduces the aggregate effective length in each step, and therefore must terminate after a finite number of steps in a trellis-oriented generator matrix.

Example 2 (cont.) The standard generator matrix for the $(8, 4, 4)$ RM code is as follows:

$$\begin{bmatrix} 1111 & 0000 \\ 1010 & 1010 \\ 1100 & 1100 \\ 1111 & 1111 \end{bmatrix}.$$

The ending times are all distinct, but the starting times are all the same. Adding the first generator to all others results in

$$\begin{bmatrix} 1111 & 0000 \\ 0101 & 1010 \\ 0011 & 1100 \\ 0000 & 1111 \end{bmatrix}.$$

All starting and ending times are now distinct, so this generator matrix is trellis-oriented. □

Exercise 3. Consider the following generator matrix for the $(16, 5, 8)$ RM code, which follows directly from the $|u|u + v|$ construction:

$$\begin{bmatrix} 1111111100000000 \\ 1111000011110000 \\ 1100110011001100 \\ 1010101010101010 \\ 1111111111111111 \end{bmatrix}.$$

Convert this generator matrix to a trellis-oriented generator matrix. □

Exercise 4 (minimum-span generators for convolutional codes).

(a) Let $\mathcal{C}$ be a rate-$1/n$ binary linear convolutional code generated by a rational n-tuple $\mathbf{g}(D)$, and let $\mathbf{g}'(D)$ be the canonical polynomial n-tuple that generates $\mathcal{C}$. Show that the generators $\{D^k\mathbf{g}'(D), k \in \mathbb{Z}\}$ are a set of minimum-span generators for $\mathcal{C}$.

(b) Show that the greedy algorithm of Section 9.2.4 chooses a set of minimum-span generators for a rate-k/n binary linear convolutional code. □

The key property of a trellis-oriented generator matrix, used in the first part of the proof of Theorem 10.3, is that the starting and ending times of a linear combination (with nonzero coefficients) of a subset of generators are the earliest and latest starting times of the component generators, respectively. We state this important observation as a lemma:

Lemma 10.4 (Generators for subcodes) *Given a trellis-oriented generator matrix for a linear code $\mathcal{C}$, if $[k, k'] \subseteq \mathcal{I}$ is any subinterval of the time axis $\mathcal{I}$, then the subcode $\mathcal{C}_{[k,k']}$ is the set of all linear combinations of generators whose spans are contained in $[k, k']$.*

Thus the dimensions of each past and future subcode may be read directly from a trellis-oriented generator matrix. Moreover, for any partition into past and future, the state subcode $\mathcal{S}$ is generated by those generators which lie neither wholly in the past nor wholly in the future. The dimension of the minimal state space is the number of such *active* generators.

Example 1 (cont.) A trellis-oriented generator matrix for the (7, 6, 2) SPC code of Figure 1 is

$$\begin{bmatrix} 1100000 \\ 0110000 \\ 0011000 \\ 0001100 \\ 0000110 \\ 0000011 \end{bmatrix}.$$

At each cut time k, only one generator is active, so each state space $\mathcal{S}_k$ has dimension 1. □

Example 2 (cont.) For the $(8, 4, 4)$ code, we constructed the following trellis-oriented generator matrix:

$$\begin{bmatrix} 1111 & 0000 \\ 0101 & 1010 \\ 0011 & 1100 \\ 0000 & 1111 \end{bmatrix}.$$

There are two active generators at each of the three cut times corresponding to a nontrivial state space in the trellis of Figure 2, so the four-state trellis of Figure 2 is minimal.

Notice from this matrix that the complete state complexity profile of a minimal 8-section trellis for this code is as follows: $\{1, 2, 4, 8, 4, 8, 4, 2, 1\}$. The maximum state complexity is 8, so a 4-section trellis somewhat masks the state complexity of a full minimal trellis. □

Exercise 3 (cont.). For the $(16, 5, 8)$ code given earlier, determine the state complexity profile of a minimal trellis. □

Exercise 5. (Trellis complexity of MDS codes, and the Wolf bound)

Let $\mathcal{C}$ be a linear $(n, k, d = n - k + 1)$ MDS code over a finite field $\mathbb{F}_q$. Using the property that in an MDS code there exist $q - 1$ weight-d codewords with support $\mathcal{J}$ for every subset $\mathcal{J} \subseteq \mathcal{I}$ of size $|\mathcal{J}| = d$, show that a trellis-oriented generator matrix for $\mathcal{C}$ must have the following form:

$$\begin{bmatrix} xxxx0000 \\ 0xxxx000 \\ 00xxxx00 \\ 000xxxx0 \\ 0000xxxx \end{bmatrix},$$

where $xxxx$ denotes a span of length $d = n - k + 1$, which shifts right by one position for each of the k generators (*i.e.,* from the interval $[1, n - k + 1]$ to $[k, n]$).

For example, show that binary linear $(n, n-1, 2)$ and $(n, 1, n)$ block codes have trellis-oriented generator matrices of this form.

Conclude that the state complexity profile of any $(n, k, d = n - k + 1)$ MDS code is

$$\{1, q, q^2, \ldots, |\mathcal{S}|_{\max}, |\mathcal{S}|_{\max}, \ldots, q^2, q, 1\},$$

where $|\mathcal{S}|_{\max} = q^{\min(k,\ n-k)}$.

Using the state space theorem and Exercise 2, show that this is the worst possible state complexity profile for a (n, k) linear code over $\mathbb{F}_q$. This is called the Wolf bound. □

10.2.4 Branch complexity

Most of the work on trellis complexity has focussed on state complexity. However, branch complexity is in some respects more fundamental. It is a better measure of Viterbi algorithm decoding complexity. Also, as we shall see, it cannot be reduced by sectionalization.

The time axis for branches is not the same as the time axis $\mathcal{I} = [0, N]$ for states. Branches occur *at* symbol times, whereas states occur *between* symbol times. Thus there are only N branch times, say $[0, N)$, whereas there are $N + 1$ state times.

A branch at time k may be identified by a triple (s_k, c_k, s_{k+1}), where (s_k, s_{k+1}) is a valid state transition, and c_k is a valid code symbol that may be generated during that transition. Thus there may be more than one branch (*parallel transition*) associated with a given state transition, if there is more than one output possible during that transition. The branch space at time k is the set of all possible branches, $\mathcal{B}_k = \{(s_k, c_k, s_{k+1})\}$.

Theorem 10.5 (Branch space theorem) *Let $\mathcal{C}$ be a binary linear block code defined on a time axis $\mathcal{I} = [0, N]$. Then in any minimal trellis for $\mathcal{C}$, for any $k \in \mathcal{I}$, the branch space $\mathcal{B}_k = \{(s_k, c_k, s_{k+1})\}$ is a linear vector space with dimension*

$$\dim \mathcal{B}_k = \dim \mathcal{C} - \dim \mathcal{C}_{[0,k)} - \dim \mathcal{C}_{[k+1,N)}.$$

where $\mathcal{C}_{|[0,k)}$ and $\mathcal{C}_{|[k+1,N)}$ are the subcodes defined on $[0, k)$ and $[k + 1, N)$, respectively.

Proof In a minimal trellis, the state spaces $\mathcal{S}_k$ and $\mathcal{S}_{k+1}$ are linear vector spaces of minimum dimension. The set of all codewords that pass through a given branch (s_k, c_k, s_{k+1}) is the set that have a past projection $\mathbf{c}_{|\mathcal{P}} = (\mathbf{c}_{\mathcal{P}})_{|\mathcal{P}} + \mathbf{s}_{|\mathcal{P}}$ consistent with the state s_k associated with $\mathbf{s}_{|\mathcal{P}}$, a projection $\mathbf{c}_{|\{k\}}$ at time k equal to c_k, and a future projection (with respect to time $k + 1$) $\mathbf{c}_{|\mathcal{F}} = (\mathbf{c}_{\mathcal{F}})_{|\mathcal{F}} + \mathbf{s}_{|\mathcal{F}}$ consistent with the state s_{k+1} associated with $\mathbf{s}_{|\mathcal{P}}$. Thus two codewords go through the same branch at time k if and only if they differ only by an element of the past subcode $\mathcal{C}_{[0,k)}$ and/or an element of the future subcode $\mathcal{C}_{[k+1,N)}$. It follows that the branch space is a linear vector space with dimension $\dim \mathcal{B}_k = \dim \mathcal{C} - \dim \mathcal{C}_{[0,k)} - \dim \mathcal{C}_{[k+1,N)}$. □

Since $\dim \mathcal{C}_{[0,k)}$ is equal to the number of generators in a trellis-oriented generator matrix whose span lies in $[0, k)$, and $\dim \mathcal{C}_{[k+1,N)}$ is the number that lie in $[k + 1, N)$, we can read the branch complexity profile by inspection from a trellis-oriented generator matrix. In other words, $\dim \mathcal{B}_k$ is equal to the number of trellis-oriented generators that are active at symbol time k.

Example 2 (cont.) For the $(8, 4, 4)$ code, the trellis-oriented generator matrix given above shows that the branch complexity profile of a minimal 8-section trellis is as follows: $\{2, 4, 8, 8, 8, 8, 4, 2\}$. The maximum branch complexity is 8, which here equals the maximum state complexity. □

Exercise 3 (cont.) Find the branch complexity profile of a minimal trellis for the $(16, 5, 8)$ code.

10.2.5 Average dimension bounds, and asymptotics

Each generator in a trellis-oriented generator matrix contributes one dimension to the state and branch spaces during the time that it is active; *i.e.,* between its start and its end. More precisely, if its span has length L, then it contributes to $L - 1$ state spaces and L branch spaces.

The one-dimensional code generated by each generator may in fact be realized by a small time-varying state machine that has a state space of size 1 when it is inactive and of size 2 when it is active, illustrated for the generator 11110000 in Figure 4.

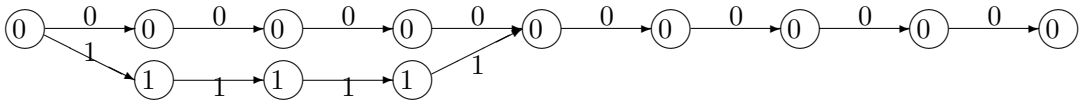


Figure 4. Two-state trellis for the one-dimensional code generated by the generator 11110000.

A minimal trellis is in effect the "product" of component trellises of this type, with the minimal state space at each time being the Cartesian product of the state spaces of each of the component trellises, and the branch space the product of the component branch spaces.

It follows that the sum of the dimensions of all state spaces is equal to the sum of the effective lengths of all generators, and the sum of the dimensions of all branch spaces is equal to the sum of the span lengths. Since each of the k generators must have span length at least d and effective length at least $d-1$, the sum of the branch space dimensions must be at least kd, and the sum of the state space dimensions must be at least $k(d-1)$.

Since there are n branch spaces and $n-1$ nontrivial state spaces, the average branch space dimension must be at least kd/n, and the average nontrivial state space dimension must be at least $k(d-1)/(n-1)$. Since the maximum dimension must be at least as large as the average dimension, this implies the following **average dimension bounds**:

$$|\mathcal{B}|_{\max} \geq 2^{kd/n}; \qquad |\mathcal{S}|_{\max} \geq 2^{k(d-1)/(n-1)}.$$

Note that we may write the branch complexity bound as $|\mathcal{B}|_{\max} \geq 2^{\gamma_c}$, where $\gamma_c = kd/n$ is the nominal coding gain of a binary (n, k, d) code on an AWGN channel. Although rarely tight, this bound is reasonably indicative for small codes; *e.g.,* a branch complexity of at least 4 is required to get a nominal coding gain of $\gamma_c = 2$ (3 dB), 16 to get $\gamma_c = 4$ (6 dB), and 256 to get $\gamma_c = 8$ (9 dB). This bound applies also to convolutional codes. The convolutional code tables show that it gives a good idea of how many states are needed to get how much nominal coding gain.

Asymptotically, this implies that, given any "good" sequence of codes such that $n \to \infty$ with d/n and k/n both bounded above zero, both the branch and state complexity must increase exponentially with n. Thus if codes are "good" in this sense, then their "complexity" (in the sense of minimal VA decoding complexity) must increase exponentially with n.

However, note that to approach the Shannon limit, the nominal coding gain $\gamma_c = kd/n$ needs only to be "good enough," not necessarily to become infinite with increasing n. In more classical coding terms, we need the minimum distance d to be "large enough," not arbitrarily large. Therefore capacity-approaching codes need not be "good" in this sense. Indeed, turbo codes have poor minimum distances and are not "good" in this sense.

10.3 The permutation and sectionalization problems

The astute reader will have observed that the minimal trellis for a linear code $\mathcal{C}$ that was found in the previous section assumes a given coordinate ordering for $\mathcal{C}$. On a memoryless channel, the performance of a code $\mathcal{C}$ is independent of the coordinate ordering, so two codes that differ only by a coordinate permutation are often taken to be equivalent. This raises a new question: what is the minimal trellis for $\mathcal{C}$ over all permutations of its coordinates?

Another point that has not been explicitly addressed previously is whether to take code symbols (bits) one at a time, two at a time, or in some other manner; *i.e.,* how to divide the code trellis into sections. This will affect the trellis appearance and (slightly) how the VA operates.

In this section we address these two problems. Permutations can make a big difference in trellis complexity, but finding the optimum permutation is intractable (NP-hard). Nonetheless, a few results are known. Sectionalization typically makes little difference in trellis complexity, but optimum sectionalization is fairly easy.

10.3.1 The permutation problem

Finding the optimum coordinate permutation from the point of view of trellis complexity is the only substantive outstanding issue in the field of trellis complexity of linear codes. Since little is known theoretically about this problem, it has been called "the art of trellis decoding" [Massey].

To illustrate that coordinate permutations do make a difference, consider the $(8, 4, 4)$ code that we have used as a running example. As an RM code with a standard coordinate ordering, we have seen that this code has state complexity profile $\{1, 2, 4, 8, 4, 8, 4, 2, 1\}$. On the other hand, consider the equivalent $(8, 4, 4)$ code generated by the following trellis-oriented generator matrix:

$$\begin{bmatrix} 11101000 \\ 01110100 \\ 00111010 \\ 00010111 \end{bmatrix}.$$

We see that the state complexity profile of this code is $\{1, 2, 4, 8, 16, 8, 4, 2, 1\}$, so its maximum state space size is 16.

In general, generator matrices that have a "cyclic" structure have poor state complexity profiles. For example, Exercise 5 shows that a trellis-oriented generator matrix for an MDS code always has such a structure, so an MDS code has the worst possible state complexity profile.

Finding the coordinate permutation that minimizes trellis complexity has been shown to be an NP-hard problem. On the other hand, various fragmentary results are known, such as:

- The Muder bound (see next subsection) applies to any coordinate permutation.
- The optimum coordinate ordering for RM codes is the standard coordinate ordering that results from the $|u|u + v|$ construction.
- A standard coordinate ordering for the $(24, 12, 8)$ binary Golay code achieves the Muder bound on the state complexity profile (see Exercise 6, below) everywhere.

10.3.2 The Muder bound

The Muder bound is a simple lower bound which shows that certain trellises have the smallest possible state space sizes. We show how this bound works by example.

Example 2 (cont.) Consider the $(8, 4, 4)$ code, with the first and last 4-tuples regarded as the past and future. The past subcode $\mathcal{C}_\mathcal{P}$ is then effectively a binary linear block code with length 4 and minimum distance at least 4. An upper bound on the largest possible dimension for such a code is $\dim \mathcal{C}_\mathcal{P} \leq 1$, achieved by the $(4, 1, 4)$ repetition code. A similar argument holds for the future subcode $\mathcal{C}_\mathcal{F}$. Thus the dimension of the state code $\mathcal{S}$ is lowerbounded by

$$\dim \mathcal{S} = \dim \mathcal{C} - \dim \mathcal{C}_\mathcal{P} - \dim \mathcal{C}_\mathcal{F} \geq 4 - 1 - 1 = 2.$$

Thus no $(8, 4, 4)$ code can have a central state space with fewer than 4 states. □

Example 3 Consider any $(32, 16, 8)$ binary linear block code. If we partition the time axis into two halves of length 16, then the past subcode $\mathcal{C}_\mathcal{P}$ is effectively a binary linear block code with length 16 and minimum distance at least 8. An upper bound on the largest possible dimension for such a code is $\dim \mathcal{C}_\mathcal{P} \leq 5$, achieved by the $(16, 5, 8)$ biorthogonal code. A similar argument holds for the future subcode $\mathcal{C}_\mathcal{F}$. Therefore $\dim \mathcal{S}$ is lowerbounded by

$$\dim \mathcal{S} = \dim \mathcal{C} - \dim \mathcal{C}_\mathcal{P} - \dim \mathcal{C}_\mathcal{F} \geq 16 - 5 - 5 = 6.$$

Thus no $(32, 16, 8)$ code can have a central state space with fewer than 64 states. Exercise 1 showed that the $(32, 16, 8)$ RM code has a trellis whose central state space has 64 states. □

In general, define $k_{\max}(n, d)$ as the largest possible dimension of code of length n and minimum distance d. There exist tables of $k_{\max}(n, d)$ for large ranges of (n, d). The **Muder bound** is then (with k denoting the time index k and $\dim \mathcal{C}$ denoting the dimension of $\mathcal{C}$):

$$\dim \mathcal{S}_k \geq \dim \mathcal{C} - k_{\max}(k, d) - k_{\max}(n - k, d).$$

Similarly, for branch complexity, we have the Muder bound

$$\dim \mathcal{B}_k \geq \dim \mathcal{C} - k_{\max}(k - 1, d) - k_{\max}(n - k, d).$$

Exercise 6. The maximum possible dimension of a binary linear $(n, k, d \geq 8)$ block code is

$$k_{\max} = \{0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 2, 2, 3, 4, 5, 5, 6, 7, 8, 9, 10, 11, 12\}$$

for $n = \{1, 2, \ldots, 24\}$, respectively. [These bounds are achieved by $(8, 1, 8), (12, 2, 8)$, $(16, 5, 8)$ and $(24, 12, 8)$ codes and shortened codes thereof.] Show that the best possible state complexity profile of any $(24, 12, 8)$ code (known as a binary Golay code) is

$$\{1, 2, 4, 8, 16, 32, 64, 128, 64, 128, 256, 512, 256, 512, 256, 128, 64, 128, 64, 32, 16, 8, 4, 2, 1\}.$$

Show that the best possible branch complexity profile is

$$\{2, 4, 8, 16, 32, 64, 128, 128, 128, 256, 512, 512, 512, 512, 256, 128, 128, 128, 64, 32, 16, 8, 4, 2\}.$$

A standard coordinate ordering that achieves both these bounds exists. □

10.3.3 The sectionalization problem

The sectionalization problem is the problem of how many symbols to take at a time in the construction of a trellis. For example, we have seen that if we take one symbol at a time with our example $(8, 4, 4)$ code, then we obtain a state complexity profile of $\{1, 2, 4, 8, 4, 8, 4, 2, 1\}$ and a branch complexity profile of $\{2, 4, 8, 8, 8, 8, 4, 2\}$, whereas if we take two symbols at a time we obtain a state complexity profile of $\{1, 4, 4, 4, 1\}$ and a branch complexity profile of $\{4, 8, 8, 4\}$. The latter trellis has less apparent state complexity and a nicer-looking trellis, but its branch complexity is not significantly less.

Sectionalization affects to some degree the order of operations in VA decoding. For example, taking symbols two at a time means that the VA computes metrics of pairs of symbols before making comparisons. For the two trellises that we have considered for the $(8, 4, 4)$ code, it is apparent that this makes no material difference in VA decoding complexity.

Sectionalization may reduce the apparent state complexity, as we have seen. In fact, if the whole trellis is clustered into one section, then the state complexity profile becomes $\{1, 1\}$.

However, clustering cannot reduce and may increase branch complexity. For example, with a one-section trellis, there are 2^k parallel transitions from the starting to the ending state.

To prove this, we generalize the branch space theorem from intervals $\{k\} = [k, k+1)$ of length 1 to general clustered intervals $[k, k')$. A branch is then identified by a triple $(s_k, \mathbf{c}_{|[k,k')}, s_{k'})$, where $s_k \in \mathcal{S}_k$, $s_{k'} \in \mathcal{S}_{k'}$, and $\mathbf{c}_{|[k,k')}$ is the projection of a codeword onto the interval $[k, k')$. The branch space $\mathcal{B}_{[k,k')}$ is the set of all such branches such that there exists a codeword $\mathbf{c}$ that passes through states $(s_k, s_{k'})$ and has the projection $\mathbf{c}_{|[k,k')}$.

By a similar argument to that in the proof of Theorem 10.5, we can conclude that in a minimal trellis two codewords go through the same branch in $\mathcal{B}_{[k,k')}$ if and only if they differ only by an element of the past subcode $\mathcal{C}_{[0,k)}$ and/or an element of the future subcode $\mathcal{C}_{[k',N)}$. This implies that the branch space $\mathcal{B}_{[k,k')}$ is a linear vector space with dimension

$$\dim \mathcal{B}_{[k,k')} = \dim \mathcal{C} - \dim \mathcal{C}_{[0,k)} - \dim \mathcal{C}_{[k',N)}.$$

Since $\dim \mathcal{C}_{[0,k)}$ is nonincreasing with decreasing k, and $\dim \mathcal{C}_{[k',N)}$ is nonincreasing with increasing k', this shows that $\dim \mathcal{B}_{[k,k')} \geq \dim \mathcal{B}_{[\kappa,\kappa')}$ for $[\kappa, \kappa') \subseteq [k, k')$; in other words,

Theorem 10.6 *Clustering cannot decrease branch complexity.*

A good elementary rule for sectionalization is therefore to cluster as much as possible without increasing branch complexity. A heuristic rule for clustering as much as possible without increasing branch complexity is therefore as follows:

HEURISTIC CLUSTERING RULE: Extend sections toward the past as long as $\dim \mathcal{C}_{[0,k)}$ does not decrease, and toward the future as long as $\dim \mathcal{C}_{(k')^+}$ does not decrease; *i.e.,* to the past up to the next trellis-oriented generator stop time, and to the future up to the next trellis-oriented generator start time.

Example 2 (cont.) For our example $(8, 4, 4)$ code, the time-3 section may be extended back to the beginning and the time-4 section to the end without violating our rule, so the optimum sectionalization has just one boundary, at the center of the trellis— *i.e.,* we take symbols four at a time, as in Figure 3(b). The state complexity profile with this sectionalization is $\{1, 4, 1\}$, and the branch complexity profile is $\{8, 8\}$, so this sectionalization simplifies the state complexity profile as much as possible without increasing the maximum branch complexity. □

Exercise 6 (cont.). Assuming that an optimum coordinate ordering for the $(24, 12, 8)$ binary Golay code exists such that the Muder bounds are met everywhere (it does), show that application of our heuristic clustering rule results in section boundaries at $k = \{0, 8, 12, 16, 24\}$, state complexity profile $\{0, 64, 256, 64, 0\}$ and branch complexity profile $\{128, 512, 512, 128\}$. □

Lafourcade and Vardy have shown that, for any reasonable definition of optimality, there exists a polynomial-time algorithm for optimal sectionalization. They further observe that the following simple rule appears always to yield the optimal sectionalization:

LV RULE: Between any time when branches merge and the next subsequent time when branches diverge in an unclustered trellis, insert one and only one section boundary.

The LV rule and our rule give the same sectionalizations for the $(8, 4, 4)$ and $(24, 12, 8)$ codes.

Chapter 11

Codes on graphs

In this chapter we will introduce the subject of codes on graphs. This subject forms an intellectual foundation for all known classes of capacity-approaching codes, including turbo codes and low-density parity-check (LDPC) codes.

There are many styles of graphical realizations of codes; *e.g.,* parity-check realizations (Tanner graphs), generator realizations, and trellis (state) realizations. More generally, we consider "behavioral" realizations, in which the time axis of a trellis realization is replaced by a general unordered graph.

We will first consider elementary linear behavioral realizations, in which the constraints are given by sets of linear equations. We then generalize to cases in which the constraints are given by linear codes. We show how behavioral realizations may be naturally represented by graphical models of various types: in particular, Tanner graphs and normal (Forney) graphs. More general classes of graphical models (*e.g.,* factor graphs, Maarkov graphs, block diagrams, and Bayesian networks) are discussed briefly in an Appendix.

Finally, we develop some elementary but important properties of graphical realizations, particularly the cut-set bound, which builds on the state space theorem.

11.1 Elementary realizations of linear block codes

We continue to restrict attention to linear (n, k) block codes over a finite field $\mathbb{F}_q$. So far we have seen several general methods of characterizing such codes:

- By a set of k generators $\{\mathbf{g}_j, 1 \leq j \leq k\}$. The code $\mathcal{C}$ is then the set of all linear combinations $\sum_i u_i \mathbf{g}_i$ of the generators over $\mathbb{F}_q$.
- By a set of $n-k$ generators $\{\mathbf{h}_j, 1 \leq j \leq n-k\}$ for the dual code $\mathcal{C}^\perp$. The code $\mathcal{C}$ is then the set of all n-tuples $\mathbf{y} \in (\mathbb{F}_q)^n$ such that $\langle \mathbf{y}, \mathbf{h}_j \rangle = 0$ for all j.
- By a trellis (state-space) realization. The code is then the set of all n-tuples corresponding to paths through the trellis.

We will see that these realizations are all special cases of a general class of realizations called *behavioral realizations* (from the behavioral approach to system theory pioneered by Willems). In general, a behavioral realization defines a code by a set of constraints that the code symbols and other auxiliary state variables must satisfy.

For linear codes, we need consider only linear behavioral realizations, where the variables are over a field and the constraints are linear. In the simplest case, the variables are field elements and the constraints are linear equations involving the variables. In the general case, the variables can be vector spaces over the field, and the constraints are expressed in terms of linear codes.

11.1.1 Elementary linear behavioral realizations

The elements of an elementary linear behavioral realization of a linear (n, k, d) block code over $\mathbb{F}_q$ are as follows:

- The n code symbols $\mathbf{y} = \{y_i \in \mathbb{F}_q, i \in \mathcal{I}\}$, where $\mathcal{I}$ denotes the symbol index set.
- An additional set of s auxiliary variables $\mathbf{s} = \{s_j \in \mathbb{F}_q, j \in \mathcal{J}\}$, often called state (hidden, latent, unobserved) variables, where the state variable index set $\mathcal{J}$ may be unrelated to $\mathcal{I}$.
- A set of e linear homogeneous equations over $\mathbb{F}_q$ involving the components of the symbol and state variables, called the constraint equations.

The *full behavior* $\mathfrak{B}$ generated by the realization is the set of all combinations $(\mathbf{y}, \mathbf{s})$ (called *trajectories*) of symbol and state variables that satisfy all constraint equations. The code $\mathcal{C}$ generated by the realization is the set of all symbol n-tuples $\mathbf{y}$ that appear in any trajectory $(\mathbf{y}, \mathbf{s}) \in \mathfrak{B}$; *i.e.,* such that there exists some set $\mathbf{s}$ of state variables such that $(\mathbf{y}, \mathbf{s}) \in \mathfrak{B}$.

In general, the e linear homogeneous constraint equations may be written in matrix form as

$$\mathbf{y}A + \mathbf{s}B = \mathbf{0},$$

where $\mathbf{y}$ is a row n-tuple of symbols, $\mathbf{s}$ is a row s-tuple of state variable components, and A and B are $n \times e$ and $s \times e$ $\mathbb{F}_q$-matrices, respectively. The set $\mathfrak{B}$ of all solutions $(\mathbf{y}, \mathbf{s})$ to such a set of equations is a subspace of the vector space $(\mathbb{F}_q)^{n+s}$ of dimension $\dim \mathfrak{B} \geq n + s - e$, with equality if and only if all equations are linearly independent.

The code $\mathcal{C}$ is the projection of $\mathfrak{B}$ onto its first n components. The dimension of $\mathcal{C}$ is equal to the dimension of $\mathfrak{B}$ if and only if codewords corresponding to distinct trajectories are distinct.

We now show that generator matrices and parity-check matrices yield elementary behavioral realizations of this kind.

Example 1 (generator realizations). Let G be a $k \times n$ generator matrix for $\mathcal{C}$, whose k rows form a set of linearly independent generators for $\mathcal{C}$. Then $\mathcal{C}$ is the set of all n-tuples of the form $\mathbf{y} = \mathbf{u}G$ for some information k-tuple $\mathbf{u} \in (\mathbb{F}_q)^k$. Thus $\mathcal{C}$ has an elementary linear behavioral realization with a state k-tuple $\mathbf{u}$ and n constraint equations, namely

$$\mathbf{y} - \mathbf{u}G = \mathbf{0}.$$

For example, in the previous chapter we found the following trellis-oriented generator matrix for the $(8, 4, 4)$ RM code:

$$\begin{bmatrix} 11110000 \\ 01011010 \\ 00111100 \\ 00001111 \end{bmatrix}. \tag{11.1}$$

This yields a linear behavioral realization with 4 state variables and 8 constraint equations, namely the following linear homogeneous equations over $\mathbb{F}_2$:

$$\begin{aligned} y_0 &= u_1; \\ y_1 &= u_1 + u_2; \\ y_2 &= u_1 + u_3; \\ y_3 &= u_1 + u_2 + u_3; \\ y_4 &= u_2 + u_3 + u_4; \\ y_5 &= u_3 + u_4; \\ y_6 &= u_2 + u_4; \\ y_7 &= u_4. \end{aligned} \tag{11.2}$$

□

Example 2 (parity-check realizations). Let H be an $(n-k) \times n$ generator matrix for $\mathcal{C}^\perp$. Then $\mathcal{C}$ is the set of all n-tuples that satisfy the $n-k$ constraint equations

$$\mathbf{y}H^T = \mathbf{0}.$$

This corresponds to an elementary linear behavioral realization with no state variables.

For example, since the $(8, 4, 4)$ code $\mathcal{C}$ is self-dual, the generator matrix (11.1) is also a generator matrix for $\mathcal{C}^\perp$. This yields an elementary linear behavioral realization with no state variables and 4 constraint equations, namely the following linear homogeneous equations over $\mathbb{F}_2$:

$$\begin{aligned} y_0 + y_1 + y_2 + y_3 &= 0; \\ y_1 + y_3 + y_4 + y_6 &= 0; \\ y_2 + y_3 + y_4 + y_5 &= 0; \\ y_4 + y_5 + y_6 + y_7 &= 0. \end{aligned} \tag{11.3}$$

This is evidently a more compact realization than the generator realization of Example 1— in fact, it can be found by eliminating the state variables from Eqs. (11.2)— and, because it has no state variables, it is better suited for checking whether a given 8-tuple $\mathbf{y}$ is in $\mathcal{C}$. On the other hand, in the generator realization the state 4-tuple $\mathbf{u}$ may be freely chosen and determines the codeword— *i.e.,* the generator realization is an input-output realization— so it is better for generating codewords, *e.g.,* in encoding or in simulation. □

11.1.2 Graphs of elementary linear behavioral realizations

We may draw a graph of an elementary linear behavioral realization as follows. In coding theory, such a graph is called a *Tanner graph*.

The graph has two types of vertices, namely $n + s$ vertices corresponding to the n symbol and s state variables, and e vertices corresponding to the e constraint equations. An edge is drawn between a variable vertex and a constraint vertex if the corresponding variable is involved in the corresponding constraint. Thus the graph is *bipartite*; *i.e.,* the vertices are partitioned into two sets such that every edge connects a vertex of one type to one of the other type.

A generic Tanner graph therefore has the form of Figure 1(a). Here symbol variables are represented by filled circles, state variables by open circles, and constraints by squares containing a "+" sign, since all constraint equations are zero-sum (parity-check) constraints.

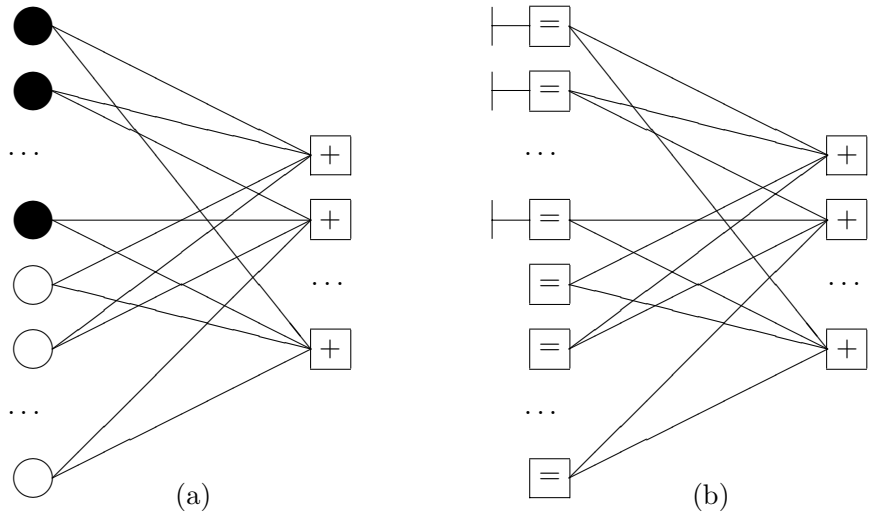


Figure 1. (a) Generic bipartite Tanner graph, with symbol variables (filled circles), state variables (open circles), and zero-sum constraints (squares with "+"). (b) Equivalent normal graph, with equality constraints replacing variables, and observed variables indicated by "dongles."

Figure 1(b) shows an equivalent normal graph (also called a Forney graph). Here the variables are replaced by equality constraints, so that all graph vertices represent constraints. Variables are represented by edges; an equality constraint ensures that all of its incident edges represent a common variable (as the edges in Tanner graphs do implicitly). Finally, a symbol variable is indicated by a special "half-edge" ("dongle") symbol incident on the corresponding equality constraint. The dongle may be regarded as an input/output terminal that can connect the corresponding symbol variable with the outside world, whereas state variables are hidden, internal variables that do not interact with the outside world.

The *degree* of a variable or equation will be defined as the degree (number of incident edges) of the corresponding graph vertex— *i.e.,* the degree of a variable is the number of equations that it is involved in, and the degree of an equation is the number of variables that it involves. In a Tanner graph, the sum of the variable degrees is equal to the sum of the constraint degrees, since both are equal to the number of edges in the graph. In a normal graph, if we choose not to count half-edges in vertex degrees, then the vertex degrees are the same.

Example 1 (generator realizations) (cont.) The Tanner graph corresponding to the generator realization of the $(8, 4, 4)$ code defined by Eqs. (11.2) is shown in Figure 2(a). Since each symbol variable has degree 1 in this realization, the corresponding symbol vertex is located adjacent to the unique constraint vertex with which it is associated. The equivalent normal graph is shown in Figure 2(b); here symbol variables may simply be represented by dongles.

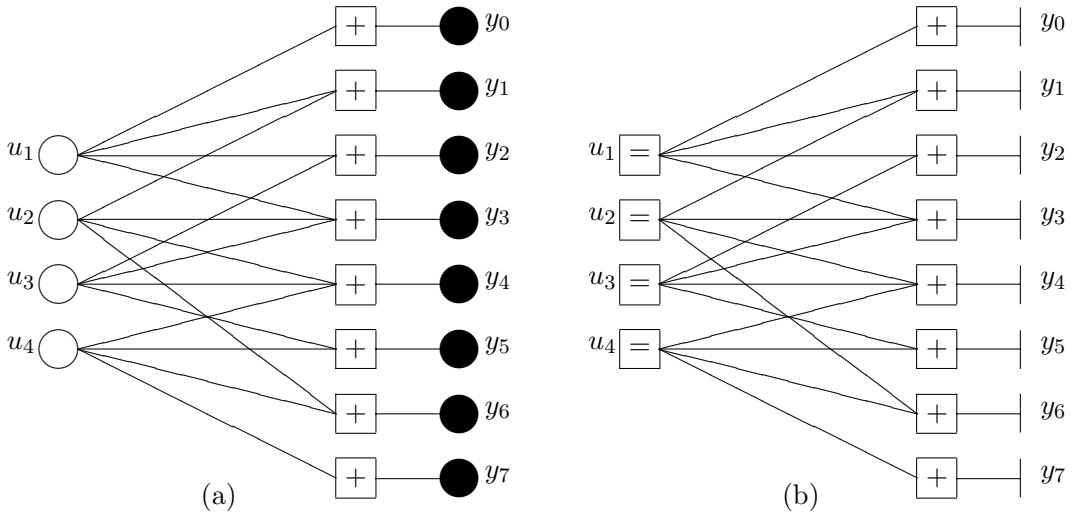


Figure 2. Generator realizations for $(8, 4, 4)$ code. (a) Tanner graph. (b) Normal graph.

Example 2 (parity-check realizations) (cont.) The Tanner graph and normal graph of the parity-check realization of the $(8, 4, 4)$ code defined by Eqs. (11.3) are shown in Figure 3.

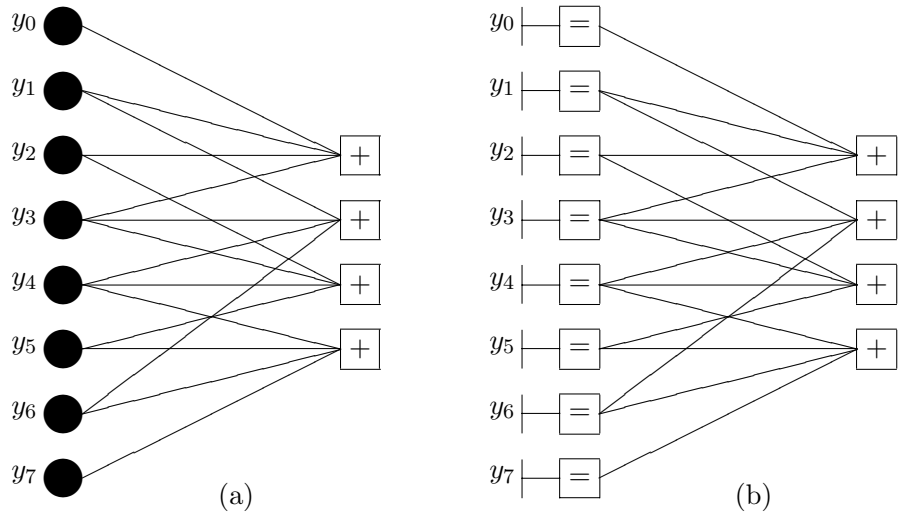


Figure 3. Parity-check realizations for $(8, 4, 4)$ code. (a) Tanner graph. (b) Normal graph.

The normal graphs of Figures 2(b) and 3(b) are duals, in the sense that one is obtained from the other by replacing equality constraints by zero-sum constraints and *vice versa*. In general, the dual of a generator realization for $\mathcal{C}$ will be a parity-check realization for $\mathcal{C}^\perp$, and *vice versa*; here we have $\mathcal{C} = \mathcal{C}^\perp$, since the $(8, 4, 4)$ code is self-dual. This illustrates an important general duality property of normal graphs, which we will not prove here: the dual of a normal graph realization of a code is a realization of the dual code. □

11.2 General linear behavioral realizations

We now generalize the elementary realizations above by letting symbol and state variables be vector spaces of dimension m over $\mathbb{F}_q$, or more particularly m-tuples over $\mathbb{F}_q$, where the dimension m may be different for each variable. Furthermore, we generalize to constraints that certain subsets of variables must lie in certain small linear block codes over $\mathbb{F}_q$.

The elements of a general linear behavioral realization of a linear (n, k) block code over $\mathbb{F}_q$ are therefore as follows:

- A set of symbol m_i-tuples $\{y_i \in (\mathbb{F}_q)^{m_i}, i \in \mathcal{I}\}$, where $\mathcal{I}$ denotes the symbol variable index set. We define $n = \sum_{\mathcal{I}} m_i$.
- A state index set $\mathcal{J}$, and a set of state spaces $\Sigma_j, j \in \mathcal{J}$, where Σ_j is a vector space over $\mathbb{F}_q$ of dimension μ_j. Such a state space Σ_j may always be represented by a vector space of μ_j-tuples, $\{\Sigma_j = (\mathbb{F}_q)^{\mu_j}, j \in \mathcal{J}\}$. We define $s = \sum_{\mathcal{J}} \mu_j$.
- A set of linear constraint codes $\{\mathcal{C}_k, k \in \mathcal{K}\}$ over $\mathbb{F}_q$, where each code $\mathcal{C}_k$ involves a certain subset of the symbol and state variables, and $\mathcal{K}$ denotes the constraint index set.

Again, the *full behavior* $\mathfrak{B}$ generated by the realization is the set of all trajectories $(\mathbf{y}, \mathbf{s})$ such that all constraints are satisfied— *i.e.,* such that for each k the values taken by the subset of variables involved in the constraint code $\mathcal{C}_k$ forms a codeword in $\mathcal{C}_k$— and the code $\mathcal{C}$ generated by the realization is the set of all symbol sequences $\mathbf{y}$ that appear in any trajectory $(\mathbf{y}, \mathbf{s}) \in \mathfrak{B}$.

Notice that the constraint imposed by a zero-sum constraint constrains the d variables incident on a zero-sum vertex of degree d to lie in the $(d, d-1, 2)$ zero-sum (SPC) code over $\mathbb{F}_q$. Similarly, an equality constraint of degree d constrains the d incident variables to lie in the $(d, 1, d)$ repetition code over $\mathbb{F}_q$. Thus allowing each constraint code $\mathcal{C}_k$ to be an arbitrary linear code generalizes the elementary linear behavioral realizations discussed earlier.

The generalization to variables of dimension m allows us to consider state spaces of dimension larger than 1, which we need for general trellis (state-space) realizations. It also allows us to consider the clustered symbol variables that arise in sectionalized trellis realizations.

We now show how trellis (state-space) realizations may be expressed as general linear behavioral realizations.

Example 3 (trellis realizations). Let us consider an unsectionalized minimal trellis realization of an (n, k) binary linear block code $\mathcal{C}$ on the time axis $\mathcal{I} = [0, n)$.

As we saw in the previous chapter, a minimal trellis realization of $\mathcal{C}$ may be defined by a trellis-oriented generator matrix G for $\mathcal{C}$ comprising k minimal-span generators $\{\mathbf{g}_j, 1 \le j \le k\}$. We thus have a one-to-one correspondence $\mathcal{C} \leftrightarrow (\mathbb{F}_2)^k$ defined by $\mathbf{u}G \leftrightarrow \mathbf{u}$.

We need to define an index set $\mathcal{J}$ for the state spaces $\Sigma_j, j \in \mathcal{J}$. When the symbol time index set is $\mathcal{I} = [0, n)$, we define the state index set as $\mathcal{J} = [0, n]$, with the understanding that the kth symbol comes after the kth state and before the $(k+1)$st state. The initial and final state spaces Σ_0 and Σ_n have dimension 0; *i.e.,* they are trivial.

We further need to define an explicit realization for the state spaces Σ_k. Let $\mathcal{J}(k)$ denote the set of indices of the trellis-oriented generators $\mathbf{g}_j$ that are active at state time k. The state code $\mathcal{S}_k$ at state time k is then generated by the submatrix $G_k = \{\mathbf{g}_j, j \in \mathcal{J}(k)\}$. We thus have a one-to-one correspondence $\mathcal{S}_k \leftrightarrow (\mathbb{F}_2)^{|\mathcal{J}(k)|}$ defined by $\mathbf{u}_{|\mathcal{J}(k)} G_k \leftrightarrow \mathbf{u}_{|\mathcal{J}(k)}$, where $\mathbf{u}_{|\mathcal{J}(k)} = \{u_j, j \in \mathcal{J}(k)\}$.

Thus if we define a state space Σ_k whose alphabet is the set $(\mathbb{F}_2)^{|\mathcal{J}(k)|}$ of $|\mathcal{J}(k)|$-tuples $\mathbf{u}_{|\mathcal{J}(k)}$, then we obtain a state space Σ_k of minimal dimension $|\mathcal{J}(k)|$ such that any codeword associated with a state codeword $\mathbf{u}_{|\mathcal{J}(k)}G_k \in \mathcal{S}_k$ passes through the state $\mathbf{u}_{|\mathcal{J}(k)} \in \Sigma_k$, as desired.

The branch space $\mathcal{B}_k$ at symbol time $k \in [0, n)$ is then the set of all $(|\mathcal{J}(k)|+|\mathcal{J}(k+1)|+1)$-tuples $(\sigma_k, y_k, \sigma_{k+1})$ that can actually occur. If $\mathcal{K}(k)$ denotes the subset of trellis-oriented generators that are active at symbol time k, then by a similar development it can be seen that $\mathcal{B}_k$ is a one-to-one linear function of $\mathbf{u}_{|\mathcal{K}(k)}$, so $\dim \mathcal{B}_k = |\mathcal{K}(k)|$. We may thus view $\mathcal{B}_k$ as a linear constraint code of length $|\mathcal{J}(k)| + |\mathcal{J}(k+1)| + 1$ and dimension $|\mathcal{K}(k)|$.

In summary, the elements of an unsectionalized minimal trellis realization of an (n, k) binary linear block code $\mathcal{C}$ are therefore as follows:

- A set of binary symbol variables $\{Y_k, k \in [0, n)\}$;
- A set of state spaces $\{\Sigma_k, k \in [0, n]\}$ of dimension $|\mathcal{J}(k)|$, represented by binary $|\mathcal{J}(k)|$-tuples, where $\{\mathbf{g}_j, j \in \mathcal{J}(k)\}$ is the subset of trellis-oriented generators that are active at state time $k \in [0, n]$;
- A set of binary linear constraint codes $\{\mathcal{B}_k, k \in [0, n)\}$, where $\mathcal{B}_k \subseteq \Sigma_k \times Y_k \times \Sigma_{k+1}$ and $\dim \mathcal{B}_k = |\mathcal{K}(k)|$, where $\{\mathbf{g}_j, j \in \mathcal{J}(k)\}$ is the subset of trellis-oriented generators that are active at symbol time $k \in [0, n)$.

The full behavior $\mathfrak{B}$ of the trellis realization is then the set of all state/symbol sequences $(\mathbf{s}, \mathbf{y})$ such that $(s_k, y_k, s_{k+1}) \in \mathcal{B}_k$ for $k \in [0, n)$. For each state/symbol sequence $(\mathbf{s}, \mathbf{y})$ in $\mathfrak{B}$, the state sequence $\mathbf{s}$ represents a valid path through the code trellis, and the symbol sequence $\mathbf{y}$ represents the corresponding codeword. If the trellis is minimal, then each path $(\mathbf{s}, \mathbf{y})$ corresponds to a distinct codeword $\mathbf{y}$, so $|\mathfrak{B}| = |\mathcal{C}|$.

Continuing with our $(8, 4, 4)$ example, its trellis-oriented generators (11.1) are active during $[0, 3), [1, 6), [2, 5)$ and $[4, 7)$, respectively. Therefore the state space dimension profile is $|\mathcal{J}(k)| = \{0, 1, 2, 3, 2, 3, 2, 1, 0\}$, and the branch space dimension profile is $|\mathcal{K}(k)| = \{1, 2, 3, 3, 3, 3, 2, 1\}$.

Figure 4(a) shows a Tanner graph of this minimal trellis realization, and Figure 4(b) is an equivalent normal graph. Each state space Σ_k is labelled by its dimension. The state spaces Σ_0 and Σ_8 do not need to be shown, because they are trivial and not actually involved in any constraints. Each constraint code (branch space) $\mathcal{B}_k$ is labelled by its length and dimension. Since the symbol variables Y_k have degree 1, we use the special "dongle" symbol for them. Since the state spaces Σ_k have degree 2, they are naturally represented by edges in a normal graph.

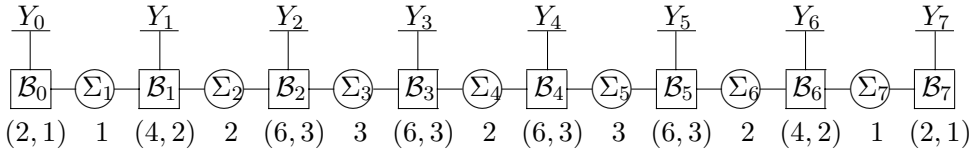


Figure 4(a). Tanner graph for minimal trellis realization of $(8, 4, 4)$ code.

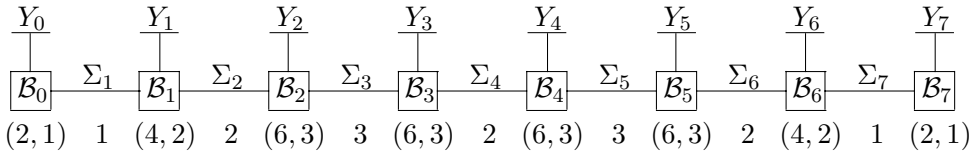


Figure 4(b). Equivalent normal graph for minimal trellis realization of $(8, 4, 4)$ code.

Every trellis (state-space) realization has a similar chain graph, with constraint codes $\mathcal{B}_k$ constraining one symbol variable Y_k and two state variables, Σ_k and Σ_{k+1}. More generally, the symbol variables Y_k may have arbitrary dimension; *i.e.,* symbols may be clustered. Note that in any trellis realization all symbol variables have degree 1, and all nontrivial state variables have degree 2. Note also that a trellis graph has no cycles (loops). □

11.3 Graph-theoretic properties of graphical realizations

A graph illustrates dependency relationships. We now develop some elementary but important connections between graph properties and dependencies.

11.3.1 Connectedness and independence

Suppose that a code $\mathcal{C}$ is the Cartesian product of two codes, $\mathcal{C} = \mathcal{C}_1 \times \mathcal{C}_2$. In other words, $\mathcal{C}$ consists of the pairs of codewords $(\mathbf{c}_1, \mathbf{c}_2)$ such that $\mathbf{c}_1 \in \mathcal{C}_1, \mathbf{c}_2 \in \mathcal{C}_2$. Then a realization of $\mathcal{C}$ may be constructed from independent realizations of $\mathcal{C}_1$ and $\mathcal{C}_2$. A graph of such a realization is a disconnected graph, with two component subgraphs representing $\mathcal{C}_1$ and $\mathcal{C}_2$, respectively.

Conversely, if a graph of a realization of $\mathcal{C}$ is disconnected, then $\mathcal{C}$ is evidently the Cartesian product of the codes realized by the component subgraphs. In short, a code $\mathcal{C}$ has a realization whose graph is disconnected if and only if $\mathcal{C}$ is the Cartesian product of shorter codes. Thus disconnectedness is a graph-theoretic expression of independence.

11.3.2 Cut sets and conditional independence

A *cut set* of a connected graph is a minimal set of edges such that removal of the set partitions the graph into two disconnected subgraphs.

Notice that a connected graph is *cycle-free* if and only if every edge is by itself a cut set.

In a normal graph, a cut set consists of a set of ordinary (state) edges, and may be specified by the corresponding subset $\chi \subseteq \mathcal{J}$ of the state index set $\mathcal{J}$. If the cut set consists of a single edge, then the cut set may be identified with the corresponding state variable Σ_j. If the cut set consists of several edges, then it may be identified with the set of all corresponding state spaces $\Sigma_j, j \in \chi$. We will regard the Cartesian product of all these state spaces as a superstate variable $\Sigma_\chi = \prod_{j\in\chi} \Sigma_j$. Note that the size of the alphabet of Σ_χ is $|\Sigma_\chi| = \prod_{j\in\chi} |\Sigma_j|$, the product of the sizes of its component state spaces.

Figure 5 gives a high-level view of a realization with a cut set χ. Since removal of a cut set partitions a graph into two disconnected subgraphs, it follows that the symbol variables, the constraint codes, and the states not in χ are partitioned by the cut set into two disjoint subsets connected only by the states in χ. We label these two components arbitrarily as the "past" $\mathcal{P}$ and the "future" $\mathcal{F}$ relative to the cut set χ. The two subsets of symbol variables associated with the past and future components are denoted by $Y_{|\mathcal{P}}$ and $Y_{|\mathcal{F}}$, respectively. The states in the cut set are regarded as a single superstate variable $\Sigma_\chi = \{\Sigma_j, j \in \chi\}$, with values $\mathbf{s}_\chi = \{s_j, j \in \chi\}$. The constraints and internal variables in the past and future components are agglomerated into aggregate constraints $\mathcal{C}_\mathcal{P}$ and $\mathcal{C}_\mathcal{F}$ that jointly constrain the aggregate superstate variable Σ_χ and the aggregate symbol variables $Y_{|\mathcal{P}}$ and $Y_{|\mathcal{F}}$, respectively.

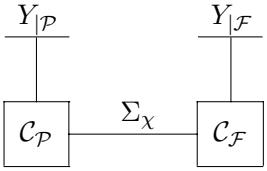


Figure 5. High-level view of a realization with a cut set χ.

Figure 5 makes it clear that a cut set in the graph corresponds to a certain conditional independence (Markov) property: given that the state variables in the cut set have a certain set of values $\mathbf{s}_\chi = \{s_j, j \in \chi\}$, the possible values of $Y_{|\mathcal{F}}$ depend only on $\mathbf{s}_\chi$ and not otherwise on the "past" $\mathcal{P}$, and *vice versa*. In other words, the superstate variable value $\mathbf{s}_\chi$ is a sufficient statistic for the past with respect to the set of possible futures, and *vice versa*.

More concretely, let $Y_{|\mathcal{P}}(\mathbf{s}_\chi)$ and $Y_{|\mathcal{F}}(\mathbf{s}_\chi)$ denote the sets of possible past and future symbol values that are consistent with a given superstate value $\mathbf{s}_\chi$, in view of the constraints $\mathcal{C}_\mathcal{P}$ and $\mathcal{C}_\mathcal{F}$. Then the set of possible codewords consistent with $\mathbf{s}_\chi$ is the Cartesian product $Y_{|\mathcal{P}}(\mathbf{s}_\chi) \times Y_{|\mathcal{F}}(\mathbf{s}_\chi)$. (In effect, fixing the value of the superstate removes the corresponding edge and disconnects the graph.) The set $\mathcal{C}$ of all possible codewords is then the union of such Cartesian products over all superstates:

$$\mathcal{C} = \bigcup_{\mathbf{s}_\chi \in \Sigma_\chi} Y_{|\mathcal{P}}(\mathbf{s}_\chi) \times Y_{|\mathcal{F}}(\mathbf{s}_\chi). \tag{11.4}$$

11.3.3 The cut-set bound

Observe that (11.4) is a generic expression for a code $\mathcal{C}$ generated by a two-section trellis with central state space Σ_χ, and that Figure 5 is a generic normal graph for such a two-section trellis. This observation leads directly to a lower bound on the size of Σ_χ:

Theorem 11.1 (Cut-set bound) *Given a graphical realization of a code $\mathcal{C}$ and a cut set χ, the size $|\Sigma_\chi| = \prod_{j\in\chi} |\Sigma_j|$ of the alphabet of the superstate variable $\Sigma_\chi = \{\Sigma_j, j \in \chi\}$ is lowerbounded by the minimal state space size in a conventional trellis in which the symbol variables are divided into "past" and "future" in the same way.*

If $\mathcal{C}$ is linear, then minimal state space size is given by the state space theorem for linear codes.

For example, by the cut-set bound and the Muder bound, given any graphical realization of the $(8, 4, 4)$ binary code and any cut set that partitions the code symbols into two subsets of size 4, the size of the alphabet of the superstate variable Σ_χ must be at least 4.

We can draw some important general conclusions from the cut-set bound.

First, consider cycle-free graphical realizations. In a cycle-free realization, every edge (state variable) is a cut set, and therefore the size of every state space is lowerbounded by the minimal size of a state space in some conventional trellis in which the symbol variables are partitioned into "past" and "future" in the same way. Therefore we cannot expect any great reduction in state space sizes from using general cycle-free graphs rather than conventional trellis realizations.

On the other hand, significant reductions in state space size are possible if we use graphs with cycles. Then cut sets will generally correspond to multiple state variables, and the complexity mandated by the cut-set lower bound may be spread out across these multiple state spaces.

We now illustrate these general conclusions by considering two particular styles of realizations: tail-biting trellis realizations, and Hadamard-transform-based realizations of Reed-Muller codes.

11.3.4 Tail-biting trellis realizations

A tail-biting trellis, illustrated in Figure 6, is a trellis in which the topology of the time axis is that of a circle rather than an interval. In other words, the ending state variable is also the starting state variable, and its alphabet may be of any size.

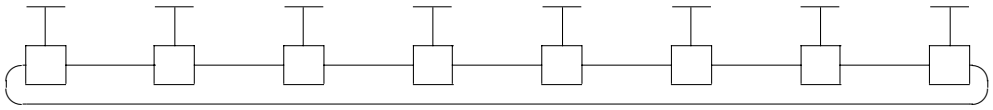


Figure 6. Normal graph of a tail-biting trellis realization.

In a tail-biting trellis realizations, all cut sets involve two state variables. Therefore the minimum complexity mandated by the cut-set bound may be spread out over two state spaces, each of which may be as small as the square root of the cut-set lower bound.

For a simple but not very impressive example, consider the $(8, 4, 4)$ code. Figure 7(a) shows a two-section, four-state conventional trellis realization which clusters four symbol bits at a time, which as we saw in Chapter 10 is an optimal sectionalization. Figure 7(b) shows a two-section tail-biting trellis realization with two 2-state state spaces, which in this case may be obtained merely by splitting the central state space of Figure 7(a). Thus the square root lower bound is achieved. Note however that while Figure 7(b) has smaller state spaces, it is no longer cycle-free.

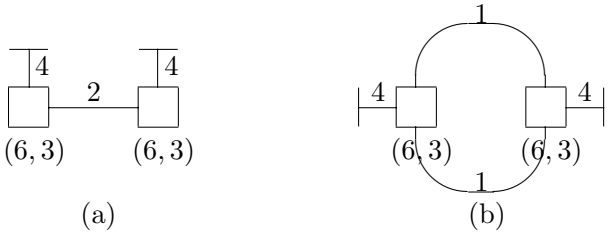


Figure 7. (a) Two-section, four-state trellis for $(8, 4, 4)$ code;
(b) Two-section, two-state tail-biting trellis realization.

More impressively, it has been shown that the state complexity of a 12-section realization of the $(24, 12, 8)$ Golay code may similarly be reduced from 256 (the minimum permitted by the Muder bound; see Exercise 10.6) to 16 by using a tail-biting realization. A "tail-biting trellis-oriented" generator matrix that yields this 16-state tail-biting realization is as follows:

11	01	11	01	11	00	00	00	00	00	00	00
00	11	11	10	01	11	00	00	00	00	00	00
00	00	11	01	10	11	11	00	00	00	00	00
00	00	00	11	01	11	01	11	00	00	00	00
00	00	00	00	11	01	11	01	11	00	00	00
00	00	00	00	00	11	11	10	01	11	00	00
00	00	00	00	00	00	11	01	10	11	11	00
00	00	00	00	00	00	00	11	01	11	01	11
11	00	00	00	00	00	00	00	11	01	11	01
01	11	00	00	00	00	00	00	00	11	11	10
10	11	11	00	00	00	00	00	00	00	11	01
01	11	01	11	00	00	00	00	00	00	00	11

Note that there are 4 "active" generators at each of the 12 state times, if we take the time axis to be circular ("end-around"). On the other hand, if we were to assume a conventional time axis, then at least 8 generators would have to be active at the central state time.

Note also that if we "unwrap" these generators onto an infinite conventional time axis, then we get generators for a rate-1/2 16-state period-4 time-varying (or rate-4/8 16-state time-invariant) binary linear convolutional code, as follows:

$$
\begin{array}{cccccccccccc}
 & & & & & \ldots & & & & & & \\
\ldots & 00 & 11 & 01 & 11 & 01 & 11 & 00 & 00 & 00 & 00 & \ldots \\
\ldots & 00 & 00 & 11 & 11 & 10 & 01 & 11 & 00 & 00 & 00 & \ldots \\
\ldots & 00 & 00 & 00 & 11 & 01 & 10 & 11 & 11 & 00 & 00 & \ldots \\
\ldots & 00 & 00 & 00 & 00 & 11 & 01 & 11 & 01 & 11 & 00 & \ldots \\
 & & & & & \ldots & & & & & &
\end{array}
$$

This "Golay convolutional code" has a minimum Hamming distance of 8 and an average of $K_b = 12.25$ weight-8 codewords per information bit, so its nominal coding gain is $\gamma_c = 4$ (6 dB) and its effective coding gain is $\gamma_{\text{eff}} = 5.3$ dB, which are remarkable for a 16-state rate-1/2 code.

In summary, by considering a state realization with a single cycle rather than a conventional trellis realization, we may be able to obtain a state complexity as small as the square root of the minimum state complexity of a conventional trellis.

11.3.5 Hadamard-transform-based realizations of RM codes

In Exercise 6 of Chapter 6, it was shown that all Reed-Muller codes $\text{RM}(r, m)$ of length 2^m could be generated by a single "universal" $2^m \times 2^m$ generator matrix $U_m = (U_1)^{\otimes m}$, the m-fold tensor product of the 2×2 matrix $U_1 = \left[\begin{smallmatrix} 1 & 0 \\ 1 & 1 \end{smallmatrix}\right]$ with itself. The matrix U_m is called the *Hadamard transform* matrix over $\mathbb{F}_2$. For any binary 2^m-tuple $\mathbf{u} \in (\mathbb{F}_2)^{2^m}$, the binary 2^m-tuple $\mathbf{y} = \mathbf{u}U_m$ is called the *Hadamard transform* of $\mathbf{u}$. Since $(U_m)^2 = I_{2^m}$, the identity matrix, it follows that the Hadamard transform of $\mathbf{y}$ is $\mathbf{u}$; *i.e.,* $\mathbf{u} = \mathbf{y}U_m$

More particularly, $\text{RM}(r, m) = \{\mathbf{y} = \mathbf{u}U_m\}$, where the coordinates of the binary 2^m-tuple $\mathbf{u}$ are free in the $k(r, m)$ positions corresponding to the $k(r, m)$ rows of U_m of weight 2^{m-r} or greater, and fixed to 0 in the remaining coordinates. In other words, $\text{RM}(r, m)$ is the set of Hadamard transforms of all $2^{k(r,m)}$ binary 2^m-tuples that are all-zero in a certain $2^m - k(r, m)$ coordinates. (Compare the Fourier transform characterization of Reed-Solomon codes in Chapter 8.)

We can construct a graphical realization of a Hadamard transform as follows. The 2×2 Hadamard transform $\mathbf{y} = \mathbf{u}U_1$ is explicitly given by the two equations

$$
\begin{aligned}
y_0 &= u_0 + u_1; \\
y_1 &= u_1,
\end{aligned}
$$

which are realized by the normal graph of Figure 8. (This is sometimes called a controlled-NOT gate, where $y_1 = u_1$ is regarded as a control variable.)

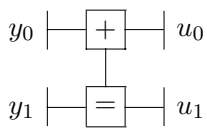


Figure 8. Normal graph of a 2×2 Hadamard transform.

Note that there are no arrows (directed edges) in this behavioral realization. Either $\mathbf{u}$ or $\mathbf{y}$ may be taken as input, and correspondingly $\mathbf{y}$ or $\mathbf{u}$ as output; *i.e.,* the graph is a realization of either the Hadamard transform $\mathbf{y} = \mathbf{u}U_1$ or the inverse Hadamard transform $\mathbf{u} = \mathbf{y}U_1$.

A $2^m \times 2^m$ Hadamard transform $\mathbf{y} = \mathbf{u}U_m$ may then be realized by connecting these 2×2 transforms in tensor product fashion. For example, the 8×8 Hadamard transform is given explicitly by the eight equations

$$\begin{aligned}
y_0 &= u_0 + u_1 + u_2 + u_3 + u_4 + u_5 + u_6 + u_7; \\
y_1 &= u_1 + u_3 + u_5 + u_7; \\
y_2 &= u_2 + u_3 + u_6 + u_7; \\
y_3 &= u_3 + u_7; \\
y_4 &= u_4 + u_5 + u_6 + u_7; \\
y_5 &= u_5 + u_7; \\
y_6 &= u_6 + u_7; \\
y_7 &= u_7.
\end{aligned}$$

These equations are realized by the tensor product graph of Figure 9. (Compare the "butterflies" in the graph of an 8×8 fast Fourier transform.)

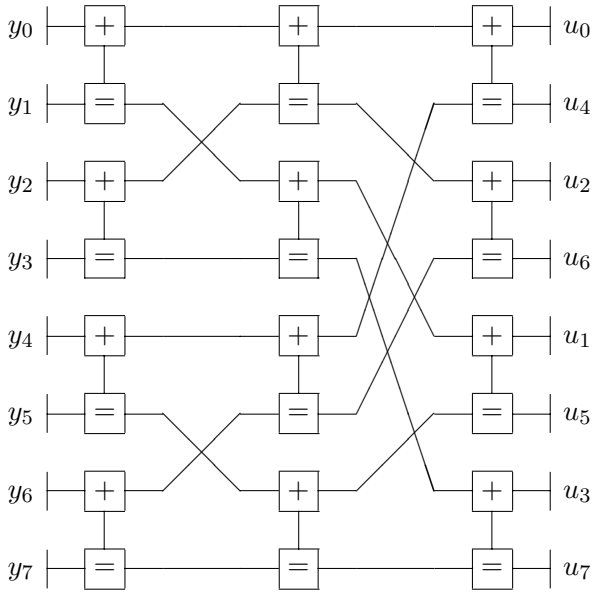


Figure 9. Normal graph of an 8×8 Hadamard transform.

A Reed-Muller code of length 8 may then be realized by fixing certain of the u_k to zero while letting the others range freely. For example, the $(8, 4, 4)$ code is obtained by fixing $u_0 = u_1 = u_2 = u_4 = 0$, which yields the equations

$$\begin{aligned}
y_0 &= u_3 + u_5 + u_6 + u_7; \\
y_1 &= u_3 + u_5 + u_7; \\
y_2 &= u_3 + u_6 + u_7; \\
y_3 &= u_3 + u_7;
\end{aligned}$$

$$\begin{aligned} y_4 &= u_5 + u_6 + u_7; \\ y_5 &= u_5 + u_7; \\ y_6 &= u_6 + u_7; \\ y_7 &= u_7. \end{aligned}$$

These equations are realized by the graph of Figure 10(a), which may be simplified to that of Figure 10(b). Here we regard the "inputs" u_j as internal variables, and the "outputs" y_k as external variables.

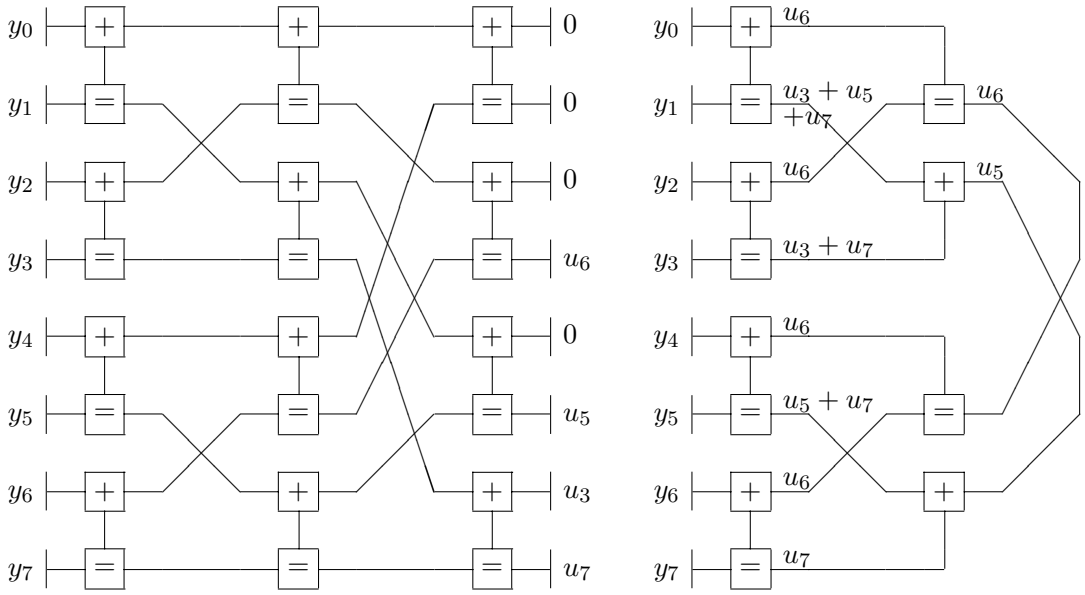


Figure 10. (a) Normal graph of $(8, 4, 4)$ RM code. (b) Equivalent realization.

In Figure 10(b), all state variables are binary and all constraint codes are simple $(3, 2, 2)$ parity-check constraints or $(3, 1, 3)$ repetition constraints. It is believed (but not proved) that this realization is the most efficient possible realization for the $(8, 4, 4)$ code in this sense. However, Figure 10(b) has cycles.

It is easy to see how the cycle-free graph of Figures 7(a) (as well as 7(b), or a minimal four-section, four-state trellis) may be obtained by agglomerating subgraphs of Figure 10(b). Such a graph is depicted in Figure 11. The code symbols are partitioned into four 2-tuples. A state space of dimension 2 connects the two halves of a codeword (meeting the cut-set bound). Two constraint codes of length 6 and dimension 3 determine the possible combinations of symbol 4-tuples and state 2-tuples in each half of the code.

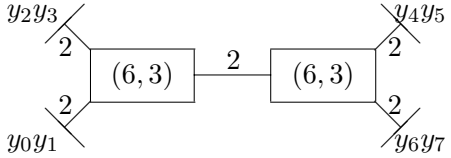


Figure 11. Tree-structured realization of $(8, 4, 4)$ RM code.

Similarly, we may realize any Reed-Muller code $\mathrm{RM}(r,m)$ in any of these styles. By starting with a Hadamard transform realization as in Figure 10(a) and reducing it as in Figure 10(b), we can obtain a realization in which all state variables are binary and all constraint codes are simple $(3,2,2)$ parity-check constraints or $(3,1,3)$ repetition constraints; however, such a realization will generally have cycles. By agglomerating variables, we can obtain a tree-structured, cycle-free realization as in Figure 11 which reflects the $|u|u+v|$ iterative RM code construction.

Exercise 1. (Realizations of repetition and SPC codes)

Show that a reduced Hadamard transform realization of a repetition code $\mathrm{RM}(0,m)$ or a single-parity-check code $\mathrm{RM}(m-1,m)$ is a cycle-free tree-structured realization with a minimum number of $(3,1,3)$ repetition constraints or $(3,2,2)$ parity-check constraints, respectively, and furthermore with minimum diameter (distance between any two code symbols in the tree). Show that these two realizations are duals; *i.e.,* one is obtained from the other via interchange of $(3,2,2)$ constraints and $(3,1,3)$ constraints.

Exercise 2. (Dual realizations of RM codes)

Show that in general a Hadamard transform (HT) realization of any Reed-Muller code $\mathrm{RM}(r,m)$ is the dual of the HT realization of the dual code $\mathrm{RM}(m-r-1,m)$; *i.e.,* one is obtained from the other via interchange of $(3,2,2)$ constraints and $(3,1,3)$ constraints.

Exercise 3. (General tree-structured realizations of RM codes)

Show that there exists a tree-structured realization of $\mathrm{RM}(r,m)$ of the following form:

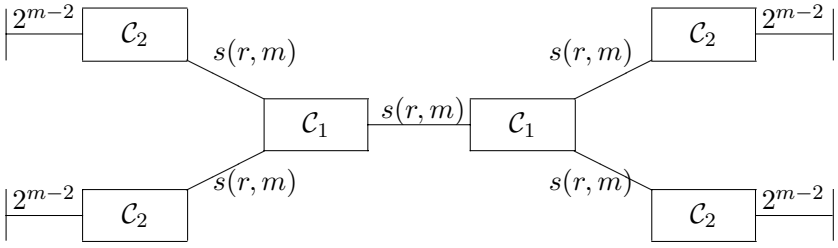


Figure 12. Tree-structured realization of $\mathrm{RM}(r,m)$.

Show that $s(r,m) = \dim \mathrm{RM}(r,m-1) - \dim \mathrm{RM}(r-1,m-1)$ (see Exercise 1 of Chapter 10). Show that the cut-set bound is met everywhere. Finally, show that

$$\begin{aligned} \dim \mathcal{C}_2 &= \dim \mathrm{RM}(r,m-2); \\ \dim \mathcal{C}_1 &= \dim \mathrm{RM}(r,m-1) - 2 \dim \mathrm{RM}(r-2,m-2) = t(r,m), \end{aligned}$$

where $t(r,m)$ is the branch complexity of $\mathrm{RM}(r,m)$ (compare Table 1 of Chapter 6). For example, there exists a tree-structured realization of the $(32,16,8)$ RM code as follows:

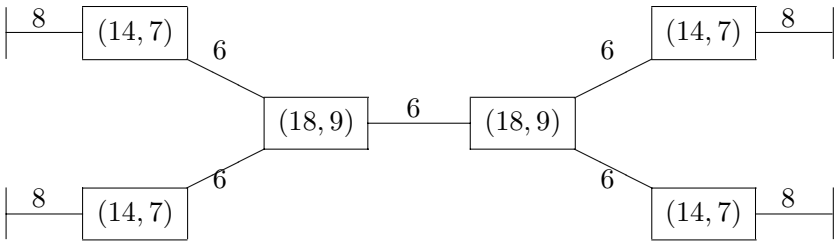


Figure 13. Tree-structured realization of $(32,16,8)$ RM ocde.

11.4 Appendix. Classes of graphical realizations

There are various classes of graphical realizations that can be used for general linear behavioral realizations. Here we will briefly discuss factor graphs, Markov graphs,, and block diagrams.

11.4.1 Factor graphs

A factor graph represents a global function of a set of variables (both internal and external) that factors into a product of local functions defined on subsets of the variables.

The indicator function $\Phi_{\mathfrak{B}}(\mathbf{y}, \mathbf{s})$ of a behavior $\mathfrak{B}$ is a $\{0, 1\}$-valued function of external variables $\mathbf{y}$ and internal variables $\mathbf{s}$ that equals 1 for valid trajectories $(\mathbf{y}, \mathbf{s})$ and equals 0 otherwise. If a trajectory $(\mathbf{y}, \mathbf{s})$ is valid whenever its components lie in a set of local constraint codes $\{\mathcal{C}_k, k \in \mathcal{K}\}$, then the global indicator function $\Phi_{\mathfrak{B}}$ is the product of local indicator functions $\{\Phi_{\mathcal{C}_k}, k \in \mathcal{K}\}$. Thus a behavioral realization may be represented by a factor graph.

A Tanner-type factor graph is an undirected bipartite graph in which variables are represented by one type of vertex (with internal and external variables denoted differently), and functions are represented by a different type of vertex. A Tanner graph of a behavioral realization may be interpreted as a Tanner-type factor graph simply by regarding the constraint vertices as representatives of constraint indicator functions. Similarly, a normal (Forney-type) factor graph is an undirected graph in which internal variables are represented by edges, external variables are represented by dongles, and functions are represented by vertices; in the same way a normal graph of a behavioral realization may be interpreted as a normal factor graph.

In the following chapters, we will be interested in global probability functions that factor into a product of local probability functions; then factor graphs become very useful.

11.4.2 Markov graphs

Markov graphs are often used in statistical physics and statistical inference to represent global probability distributions that factor into a product of local distributions.

A *Markov graph* (Markov random field) is an undirected graph in which variables are represented by vertices, and a constraint or function is represented by an edge (if it has degree 2), or by a hyperedge (if it has degree greater than 2). Moreover, a hyperedge is usually represented by a *clique*, *i.e.,* a set of ordinary edges between every pair of variables incident on the hyperedge. (This style of graph representation sometimes generates inadvertent cliques.)

Markov graphs are particularly nice when the degrees of all constraints are 2 or less. Such a representation is called a *pairwise* Markov graph. We may then represent constraints by ordinary edges. Pairwise constraints often arise naturally in physical models.

Figure 14 shows how any Tanner graph (or Tanner-type factor graph) may be transformed into a pairwise Markov realization by a simple conversion. Here each constraint code has been replaced by a state "supervariable" whose alphabet is the set of all codewords in the constraint code. Each edge then represents the constraint that the associated ordinary variable must be equal to the corresponding component of the supervariable.

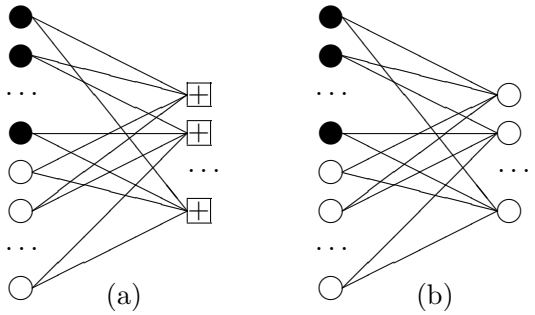


Figure 14. (a) Tanner graph. (b) Equivalent pairwise Markov graph.

For example, suppose the constraint code has degree 3 and constrains three incident variables (y_1, y_2, y_3) to satisfy the parity check $y_1 + y_2 + y_3 = 0$; *i.e.,* the constraint code is a $(3, 2, 2)$ code with four codewords, namely $\{000, 110, 101, 011\}$. We then define a supervariable y_{123} to have these codewords as its alphabet, and constrain y_1 to equal the first component of y_{123}, etc.

11.4.3 Block diagrams and directed normal graphs

Conventional block diagrams may often be regarded as normal graphs, with the vertices ("blocks") representing constraints, and the edges labeled by internal or external variables.

However, one difference is that the blocks usually represent input-output (causal) relationships, so a block diagram is usually a directed graph in which the edges are also labelled with arrows, indicating the direction of causality. In this respect block diagrams resemble Bayesian networks, which are directed acyclic graphs representing probabilistic cause-and-effect models.

This style of graphical model can sometimes be superimposed on a normal graph, as follows. If a constraint code is a linear (n, k) code and has an information set of size k, then the corresponding k symbols may be regarded as "inputs" to the constraint, and the remaining $n - k$ symbols as "outputs" determined by the inputs. Arrows may be drawn on the edges to represent such input-output relationships. If arrows can be drawn consistently on all edges in this way, then a normal graph may be converted to a directed normal graph (block diagram).

For example, Figure 15 shows how a parity-check realization for the $(8, 4, 4)$ code (Figure 3(b)) may be converted to directed normal graph form. This could be useful if, for example, we wanted to use such a graph to implement an encoder. However, this example is a bit misleading, as parity-check realizations cannot always be converted to encoders in this way.

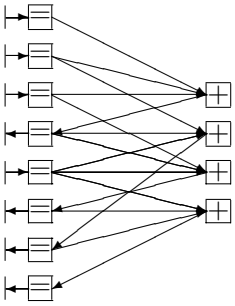


Figure 15. Conversion of parity-check realization of $(8, 4, 4)$ code to directed normal graph representing an encoder.

Chapter 12

The sum-product algorithm

The sum-product algorithm is the basic "decoding" algorithm for codes on graphs. For finite cycle-free graphs, it is finite and exact. However, because all its operations are local, it may also be applied to graphs with cycles; then it becomes iterative and approximate, but in coding applications it often works very well. It has become the standard decoding algorithm for capacity-approaching codes (*e.g.,* turbo codes, LDPC codes).

There are many variants and applications of the sum-product algorithm. The most straightforward application is to *a posteriori* probability (APP) decoding. When applied to a trellis, it becomes the celebrated BCJR decoding algorithm. In the field of statistical inference, it becomes the even more widely known "belief propagation" (BP) algorithm. For Gaussian state-space models, it becomes the Kalman smoother.

There is also a "min-sum" or maximum-likelihood sequence detection (MLSD) version of the sum-product algorithm. When applied to a trellis, the min-sum algorithm gives the same result as the Viterbi algorithm.

12.1 The sum-product algorithm on cycle-free graphs

We will develop the sum-product algorithm as an APP decoding algorithm for a code $\mathcal{C}$ that has a cycle-free normal graph realization. We then discuss generalizations.

The code $\mathcal{C}$ is therefore described by a realization involving a certain set of symbol variables $\{Y_i, i \in \mathcal{I}\}$ represented by half-edges (dongles), a certain set of state variables $\{\Sigma_j, j \in \mathcal{J}\}$ represented by edges, and a certain set of constraint codes $\{\mathcal{C}_k, k \in \mathcal{K}\}$ of arbitrary degree, such that the graph of the realization is cycle-free; *i.e.,* every edge (and obviously every half-edge) is by itself a cut set.

APP decoding is defined in general as follows. We assume that a set of independent observations are made on all symbol variables $\{Y_i, i \in I\}$, resulting in a set of observations $\mathbf{r} = \{r_i, i \in \mathcal{I}\}$ and likelihood vectors $\{\{p(r_i \mid y_i), y_i \in \mathcal{Y}_i\}, i \in \mathcal{I}\}$, where $\mathcal{Y}_i$ is the alphabet of Y_i. The likelihood of a codeword $\mathbf{y} = \{y_i, i \in \mathcal{I}\} \in \mathcal{C}$ is then defined as the componentwise product $p(\mathbf{r} \mid \mathbf{y}) = \prod_{i \in \mathcal{I}} p(r_i \mid y_i)$.

Assuming equiprobable codewords, the *a posteriori* probabilities $\{p(\mathbf{y} \mid \mathbf{r}), \mathbf{y} \in \mathcal{C}\}$ (APPs) are proportional to the likelihoods $\{p(\mathbf{r} \mid \mathbf{y}), \mathbf{y} \in \mathcal{C}\}$, since by Bayes' law,

$$p(\mathbf{y} \mid \mathbf{r}) = \frac{p(\mathbf{r} \mid \mathbf{y})p(\mathbf{y})}{p(\mathbf{r})} \propto p(\mathbf{r} \mid \mathbf{y}), \quad \mathbf{y} \in \mathcal{C}.$$

Let $\mathcal{C}_i(y_i)$ denote the subset of codewords in which the symbol variable Y_i has the value $y_i \in \mathcal{Y}_i$. Then, up to a scale factor, the symbol APP vector $\{p(Y_i = y_i \mid \mathbf{r}), y_i \in \mathcal{Y}_i\}$ is given by

$$p(Y_i = y_i \mid \mathbf{r}) = \sum_{\mathbf{y} \in \mathcal{C}_i(y_i)} p(\mathbf{y} \mid \mathbf{r}) \propto \sum_{\mathbf{y} \in \mathcal{C}_i(y_i)} p(\mathbf{r} \mid \mathbf{y}) = \sum_{\mathbf{y} \in \mathcal{C}_i(y_i)} \prod_{i' \in \mathcal{I}} p(r_{i'} \mid y_{i'}), \quad y_i \in \mathcal{Y}_i. \tag{12.1}$$

Similarly, if $\mathcal{C}_j(s_j)$ denotes the subset of codewords that are consistent with the state variable Σ_j having the value s_j in the state alphabet $\mathcal{S}_j$, then, up to a scale factor, the state APP vector $\{p(\Sigma_j = s_j \mid \mathbf{r}), s_j \in \mathcal{S}_j\}$ is given by

$$p(\Sigma_j = s_j \mid \mathbf{r}) \quad \propto \sum_{\mathbf{y} \in \mathcal{C}_j(s_j)} \prod_{i \in \mathcal{I}} p(r_i \mid y_i), \quad s_j \in \mathcal{S}_j. \tag{12.2}$$

We see that the components of APP vectors are naturally expressed as sums of products. The sum-product algorithm aims to compute these APP vectors for every state and symbol variable.

12.1.1 Past-future decomposition rule

The sum-product algorithm is based on two fundamental principles, which we shall call here the *past/future decomposition rule* and the *sum-product update rule*. Both of these rules are based on set-theoretic decompositions that are derived from the code graph.

The past/future decomposition rule is based on the Cartesian-product decomposition of the cut-set bound (Chapter 11). In this case every edge Σ_j is a cut set, so the subset of codewords that are consistent with the state variable Σ_j having the value s_j is the Cartesian product

$$\mathcal{C}_j(s_j) = Y_{|\mathcal{P}}(s_j) \times Y_{|\mathcal{F}}(s_j), \tag{12.3}$$

where $\mathcal{P}$ and $\mathcal{F}$ denote the two components of the disconnected graph which results from deleting the edge representing Σ_j, and $Y_{|\mathcal{P}}(s_j)$ and $Y_{|\mathcal{F}}(s_j)$ are the sets of symbol values in each component that are consistent with Σ_j taking the value s_j.

We now apply an elementary Cartesian-product lemma:

Lemma 12.1 (Cartesian-product distributive law) *If $\mathcal{X}$ and $\mathcal{Y}$ are disjoint discrete sets and $f(x)$ and $g(y)$ are any two functions defined on $\mathcal{X}$ and $\mathcal{Y}$, then*

$$\sum_{(x,y) \in \mathcal{X} \times \mathcal{Y}} f(x)g(y) = \left(\sum_{x \in \mathcal{X}} f(x) \right) \left(\sum_{y \in \mathcal{Y}} g(y) \right). \tag{12.4}$$

This lemma may be proved simply by writing the terms on the right in a rectangular array and then identifying them with the terms on the left. It says that rather than computing the sum of $|\mathcal{X}||\mathcal{Y}|$ products, we can just compute a single product of independent sums over $\mathcal{X}$ and $\mathcal{Y}$. This simple lemma lies at the heart of many "fast" algorithms.

Using (12.3) and applying this lemma in (12.2), we obtain the past/future decomposition rule

$$\begin{aligned} p(\Sigma_j = s_j \mid \mathbf{r}) &\propto \left(\sum_{\mathbf{y}_{|\mathcal{P}} \in Y_{|\mathcal{P}}(s_j)} \prod_{i \in \mathcal{I}_\mathcal{P}} p(r_i \mid y_i) \right) \left(\sum_{\mathbf{y}_{|\mathcal{F}} \in Y_{|\mathcal{F}}(s_j)} \prod_{i \in \mathcal{I}_\mathcal{F}} p(r_i \mid y_i) \right) \\ &\propto p(\Sigma_j = s_j \mid \mathbf{r}_{|\mathcal{P}}) p(\Sigma_j = s_j \mid \mathbf{r}_{|\mathcal{F}}), \end{aligned} \tag{12.5}$$

in which the two terms $p(\Sigma_j = s_j \mid \mathbf{r}_{|\mathcal{P}})$ and $p(\Sigma_j = s_j \mid \mathbf{r}_{|\mathcal{F}})$ depend only on the likelihoods of the past symbols $\mathbf{y}_{|\mathcal{P}} = \{Y_i, i \in \mathcal{I}_\mathcal{P}\}$ and future symbols $\mathbf{y}_{|\mathcal{F}} = \{Y_i, i \in \mathcal{I}_\mathcal{F}\}$, respectively.

The sum-product algorithm therefore computes the APP vectors $\{p(\Sigma_j = s_j \mid \mathbf{r}_{|\mathcal{P}})\}$ and $\{p(\Sigma_j = s_j \mid \mathbf{r}_{|\mathcal{F}})\}$ separately, and multiplies them componentwise to obtain $\{p(\Sigma_j = s_j \mid \mathbf{r})\}$. This is the past/future decomposition rule for state variables.

APP vectors for symbol variables are computed similarly. In this case, since symbol variables have degree 1, one of the two components of the graph induced by a cut is just the symbol variable itself, while the other component is the rest of the graph. The past/future decomposition rule thus reduces to the following simple factorization of (12.1):

$$p(Y_i = y_i \mid \mathbf{r}) \propto p(r_i \mid y_i) \left(\sum_{\mathbf{y} \in \mathcal{C}_i(y_i)} \prod_{i' \neq i} p(r_{i'} \mid y_{i'}) \right) \propto p(y_i \mid r_i) p(y_i \mid \mathbf{r}_{|i' \neq i}). \tag{12.6}$$

In the turbo code literature, the first term, $p(y_i \mid r_i)$, is called the *intrinsic* information, while the second term, $p(y_i \mid \mathbf{r}_{|i' \neq i})$, is called the *extrinsic* information.

12.1.2 Sum-product update rule

The second fundamental principle of the sum-product algorithm is the sum-product update rule. This is a local rule for the calculation of an APP vector, *e.g.*, $\{p(\Sigma_j = s_j \mid \mathbf{r}_{|\mathcal{P}}), s_j \in \mathcal{S}_j\}$, from APP vectors that lie one step further upstream.

The local configuration with respect to the edge corresponding to the state variable Σ_j is illustrated in Figure 1. The edge must be incident on a unique past vertex corresponding to a constraint code $\mathcal{C}_k$. If the degree of $\mathcal{C}_k$ is δ_k, then there are $\delta_k - 1$ edges further upstream of $\mathcal{C}_k$, corresponding to further past state or symbol variables. For simplicity, we suppose that these are all state variables $\{\Sigma_{j'}, j' \in \mathcal{K}_{jk}\}$, where we denote their index set by $\mathcal{K}_{jk} \subseteq \mathcal{K}_{|\mathcal{P}}$.

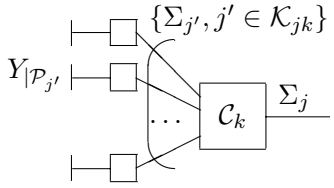


Figure 1. Local configuration for sum-product update rule.

Since the graph is cycle-free, each of these past edges has its own independent past $\mathcal{P}_{j'}$. The corresponding sets $Y_{|\mathcal{P}_{j'}}$ of input symbols must be disjoint, and their union must be $Y_{|\mathcal{P}}$. Thus if $\mathcal{C}_k(s_j)$ is the set of codewords in the local constraint code $\mathcal{C}_k$ that are consistent with $\Sigma_j = s_j$, and $Y_{|\mathcal{P}_{j'}}(s_{j'})$ is the set of $\mathbf{y}_{|\mathcal{P}_{j'}} \in Y_{|\mathcal{P}_{j'}}$ that are consistent with $\Sigma_{j'} = s_{j'}$, then we have

$$Y_{|\mathcal{P}}(s_j) = \bigoplus_{\mathcal{C}_k(s_j)} \bigotimes_{j' \in \mathcal{K}_{jk}} Y_{|\mathcal{P}_{j'}}(s_{j'}), \tag{12.7}$$

where the plus sign indicates a disjoint union, and the product sign indicates a Cartesian product. In other words, for each codeword in $\mathcal{C}_k$ for which $\Sigma_j = s_j$, the set of possible pasts is the Cartesian product of possible pasts of the other state values $\{s_{j'}, j' \in \mathcal{K}_{jk}\}$, and the total set of possible pasts is the disjoint union of these Cartesian products.

Now, again using the Cartesian-product distributive law, it follows from (12.7) that

$$p(\Sigma_j = s_j \mid \mathbf{r}_{|\mathcal{P}}) = \sum_{\mathcal{C}_k(s_j)} \prod_{j' \in \mathcal{K}_{jk}} p(\Sigma_{j'} = s_{j'} \mid \mathbf{r}_{|\mathcal{P}_{j'}}). \tag{12.8}$$

Thus if we know all the upstream APP vectors $\{p(\Sigma_{j'} = s_{j'} \mid \mathbf{r}_{|\mathcal{P}_{j'}}), s_{j'} \in \mathcal{S}_{j'}\}$, then we can compute the APP vector $\{p(\Sigma_j = s_j \mid \mathbf{r}_{|\mathcal{P}}), s_j \in \mathcal{S}_j\}$.

Equation (12.8) is the sum-product update rule. We can see that for each $s_j \in \mathcal{S}_j$ it involves a sum of $|\mathcal{C}_k|$ products of $\delta_k - 1$ terms. Its complexity is thus proportional to the size $|\mathcal{C}_k|$ of the constraint code $\mathcal{C}_k$. In a trellis, this is what we call the branch complexity.

In the special case where $\mathcal{C}_k$ is a repetition code, there is only one codeword corresponding to each $s_j \in \mathcal{S}_j$, so (12.8) becomes simply the following *product update rule*:

$$p(\Sigma_j = s_j \mid \mathbf{r}_{|\mathcal{P}}) = \prod_{j' \in \mathcal{K}_{jk}} p(\Sigma_{j'} = s_j \mid \mathbf{r}_{|\mathcal{P}_{j'}}); \tag{12.9}$$

i.e., the components of the upstream APP vectors are simply multiplied componentwise. When the sum-product algorithm is described for Tanner graphs, the product update rule is often stated as a separate rule for variable nodes, because variable nodes in Tanner graphs correspond to repetition codes in normal graphs.

Note that for a repetition code of degree 2, the product update rule of (12.9) simply becomes a pass-through of the APP vector; no computation is required. This seems like a good reason to suppress state nodes of degree 2, as we do in normal graphs.

12.1.3 The sum-product algorithm

Now we describe the complete sum-product algorithm for a finite cycle-free normal graph, using the past/future decomposition rule (12.5) and the sum-product update rule (12.8).

Because the graph is cycle-free, it is a tree. Symbol variables have degree 1 and correspond to leaves of the tree. State variables have degree 2 and correspond to branches.

For each edge, we wish to compute two APP vectors, corresponding to past and future. These two vectors can be thought of as two messages going in opposite directions.

Using the sum-product update rule, each message may be computed after all upstream messages have been received at the upstream vertex (see Figure 1). Therefore we can think of each vertex as a processor that computes an outgoing message on each edge after it has received incoming messages on all other edges.

Because each edge is the root of a finite past tree and a finite future tree, there is a maximum number d of edges to get from any given edge to the furthest leaf node in either direction, which is called its *depth* d. If a message has depth d, then the depth of any upstream message can be no greater than $d - 1$. All symbol half-edges have depth $d = 0$, and all state edges have depth $d \geq 1$. The *diameter* $d_{\max}$ of the tree is the maximum depth of any message.

Initially, incoming messages (intrinsic information) are available at all leaves of the tree. All depth-1 messages can then be computed from these depth-0 messages; all depth-2 messages can then be computed from depth-1 and depth-0 messages; etc. In a synchronous (clocked) system, all messages can therefore be computed in $d_{\max}$ clock cycles.

Finally, given the two messages on each edge in both directions, all *a posteriori* probabilities (APPs) can be computed using the past/future decomposition rule (12.5).

In summary, given a finite cycle-free normal graph of diameter $d_{\max}$ and intrinsic information for each symbol variable, the sum-product algorithm computes the APPs of all symbol and state variables in $d_{\max}$ clock cycles. One message (APP vector) of size $|\mathcal{S}_j|$ is computed for each state variable Σ_j in each direction. The computational complexity at a vertex corresponding to a constraint code $\mathcal{C}_k$ is of the order of $|\mathcal{C}_k|$. (More precisely, the number of pairwise multiplications required is $\delta_k(\delta_k - 2)|\mathcal{C}_k|$.)

The sum-product algorithm does not actually require a clock. In an asynchronous implementation, each vertex processor can continuously generate outgoing messages on all incident edges, using whatever incoming messages are available. Eventually all messages must be correct. An analog asynchronous implementation can be extremely fast.

We see that there is a clean separation of functions when the sum-product algorithm is implemented on a normal graph. All computations take place at vertices, and the computational complexity at a vertex is proportional to the vertex (constraint code) complexity. The function of ordinary edges (state variables) is purely message-passing (communications), and the communications complexity (bandwidth) is proportional to the edge complexity (state space size). The function of half-edges (symbol variables) is purely input/output; the inputs are the intrinsic APPs, and the ultimate outputs are the extrinsic APP vectors, which combine with the inputs to form the symbol APPs. In integrated-circuit terminology, the constraint codes, state variables and symbol variables correspond to logic, interconnect, and I/O, respectively.

12.2 The BCJR algorithm

The chain graph of a trellis (state-space) representation is the archetype of a cycle-free graph. The sum-product algorithm therefore may be used for exact APP decoding on any trellis (state-space) graph. In coding, the resulting algorithm is known as the Bahl-Cocke-Jelinek-Raviv (BCJR) algorithm. (In statistical inference, it is known as the forward-backward algorithm. If all probability distributions are Gaussian, then it becomes the Kalman smoother.)

Figure 2 shows the flow of messages and computations when the sum-product algorithm is applied to a trellis.

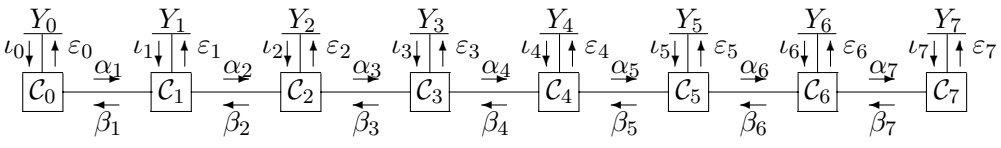


Figure 2. Flow of messages and computations in the sum-product algorithm on a trellis.

The input messages are the intrinsic APP vectors $\iota_i = \{p(r_i \mid y_i), y_i \in \mathcal{Y}_i\}$, derived from the observations r_i; the output messages are the extrinsic APP vectors $\varepsilon_i = \{p(y_i \mid \mathbf{r}_{|i' \neq i}), y_i \in \mathcal{Y}_i\}$. The intermediate messages are the forward state APP vectors $\alpha_j = p(s_j \mid \mathbf{r}_{|\mathcal{P}_j}), s_j \in \mathcal{S}_j\}$ and the backward state APP vectors $\beta_j = p(s_j \mid \mathbf{r}_{|\mathcal{F}_j}), s_j \in \mathcal{S}_j\}$, where $\mathbf{r}_{|\mathcal{P}_j}$ and $\mathbf{r}_{|\mathcal{F}_j}$ denote the observations before and after s_j, respectively.

The algorithm proceeds independently in the forward and backward directions. In the forward direction, the messages α_j are computed from left to right; α_j may be computed by the sum-product rule from the previous message α_{j-1} and the most recent input message ι_{j-1}. In the backward direction, the messages β_j are computed from right to left; β_j may be computed by the sum-product rule from β_{j+1} and ι_j.

Finally, each output message ε_i may be computed by the sum-product update rule from the messages α_i and β_{i+1}, giving the extrinsic information for each symbol. To find the APP vector of an input symbol Y_i, the intrinsic and extrinsic messages ι_i and ε_i are multiplied componentwise, according to the past/future decomposition rule. (In turbo decoding, the desired output is actually the extrinsic likelihood vector, not the APP vector.) Similarly, to find the APP vector of a state variable Σ_j, the forward and backward messages α_j and β_j are multiplied componentwise.

Exercise 1. Consider the two-state trellis diagram for the binary $(7, 6, 2)$ SPC code shown in Figure 3 of Chapter 10. Suppose that a codeword is chosen equiprobably at random, that the transmitter maps $\{0, 1\}$ to $\{\pm 1\}$ as usual, that the resulting real numbers are sent through a discrete-time AWGN channel with noise variance $\sigma^2 = 1$ per symbol, and that the received sequence is $\mathbf{r} = (0.4, -1.0, -0.1, 0.6, 0.7, -0.5, 0.2)$. Use the sum-product algorithm to determine the APP that each input bit Y_i is a 0 or a 1.

12.3 The min-sum algorithm and ML decoding

We now show that with minor modifications the sum-product algorithm may be used to perform a variant of maximum-likelihood (ML) sequence decoding rather than APP decoding. On a trellis, the resulting "min-sum" algorithm becomes a variant of the Viterbi algorithm.

With the same notation as in the previous section, the min-sum algorithm is defined as follows. Again, let $\mathcal{C}_i(y_i)$ denote the subset of codewords in which the symbol variable Y_i has the value $y_i \in \mathcal{Y}_i$. Then the *metric* $m_i(y_i)$ of y_i is defined as the maximum likelihood of any codeword $\mathbf{y} \in \mathcal{C}_i(Y_i)$; *i.e.,*

$$m_i(y_i) = \max_{\mathbf{y} \in \mathcal{C}_i(Y_i)} p(\mathbf{r} \mid \mathbf{y}) = \max_{\mathbf{y} \in \mathcal{C}_i(Y_i)} \prod_{i' \in I} p(r_{i'} \mid y_{i'}), \quad y_i \in \mathcal{Y}_i. \tag{12.10}$$

It is clear that the symbol value y_i with the maximum metric $m_i(y_i)$ will be the value of y_i in the codeword $\mathbf{y} \in \mathcal{C}$ that has the maximum global likelihood.

Similarly, if $\mathcal{C}_j(s_j)$ denotes the subset of codewords that are consistent with the state variable Σ_j having the value s_j in the state alphabet $\mathcal{S}_j$, then the metric $m_j(s_j)$ of s_j will be defined as the maximum likelihood of any codeword $\mathbf{y} \in \mathcal{C}_j(s_j)$:

$$m_j(s_j) \quad = \quad \max_{\mathbf{y} \in \mathcal{C}_j(s_j)} \prod_{i \in I} p(r_i \mid y_i), \quad s_j \in \mathcal{S}_j. \tag{12.11}$$

We recognize that (12.10) and (12.11) are almost identical to (12.1) and (12.2), with the exception that the sum operator is replaced by a max operator. This suggests that these metrics could be computed by a version of the sum-product algorithm in which "sum" is replaced by "max" everywhere, giving what is called the "max-product algorithm."

In fact this works. The reason is that the operators "max" and "product" operate on probability vectors defined on sets according to the same rules as "sum" and "product." In particular, assuming that all quantities are non-negative, we have

(a) the distributive law: $a \max\{b, c\} = \max\{ab, ac\}$;

(b) the Cartesian-product distributive law:

$$\max_{(x,y)\in\mathcal{X}\times\mathcal{Y}} f(x)g(y) = \left(\max_{x\in\mathcal{X}} f(x)\right) \left(\max_{y\in\mathcal{Y}} g(y)\right). \tag{12.12}$$

Consequently, the derivation of the previous section goes through with just this one change. From (12.3), we now obtain the past/future decomposition rule

$$m_j(s_j) = \left(\max_{\mathbf{y}_{|\mathcal{P}}\in Y_{|\mathcal{P}}(s_j)} \prod_{i\in\mathcal{I}_\mathcal{P}} p(r_i \mid y_i)\right) \left(\max_{\mathbf{y}_{|\mathcal{F}}\in Y_{|\mathcal{F}}(s_j)} \prod_{i\in\mathcal{I}_\mathcal{F}} p(r_i \mid y_i)\right) = m_j(s_j \mid \mathbf{r}_{|\mathcal{P}}) m_j(s_j \mid \mathbf{r}_{|\mathcal{F}}),$$

in which the partial metrics $m_j(s_j \mid \mathbf{r}_{|\mathcal{P}})$ and $m_j(s_j \mid \mathbf{r}_{|\mathcal{F}})$ are the maximum likelihoods over the past symbols $\mathbf{y}_{|\mathcal{P}} = \{y_i, i \in \mathcal{I}_\mathcal{P}\}$ and future symbols $\mathbf{y}_{|\mathcal{F}} = \{y_i, i \in \mathcal{I}_\mathcal{F}\}$, respectively. Similarly, we obtain the max-product update rule

$$m_j(s_j \mid \mathbf{r}_{|\mathcal{P}}) = \max_{\mathcal{C}_k(s_j)} \prod_{j'\in\mathcal{K}_{jk}} m_{j'}(s_{j'} \mid \mathbf{r}_{|\mathcal{P}_{j'}}), \tag{12.13}$$

where the notation is as in the sum-product update rule (12.8).

In practice, likelihoods are usually converted to log likelihoods, which converts products to sums and yields the max-sum algorithm. Or, log likelihoods may be converted to negative log likelihoods, which converts max to min and yields the min-sum algorithm. These variations are all trivially equivalent.

On a trellis, the forward part of any of these algorithms is equivalent to the Viterbi algorithm (VA). The update rule (12.13) becomes the add-compare-select operation, which is carried out at each state to determine the new metric $m_j(s_j \mid \mathbf{r}_{|\mathcal{P}})$ of each state. The VA avoids the backward part of the algorithm by also remembering the survivor history at each state, and then doing a traceback when it gets to the end of the trellis; this traceback corresponds in some sense to the backward part of the sum-product algorithm.

Exercise 2. Repeat Exercise 1, using the min-sum algorithm instead of the sum-product algorithm. Decode the same sequence using the Viterbi algorithm, and show how the two computations correspond. Decode the same sequence using Wagner decoding, and show how Wagner decoding relates to the other two methods.

12.4 The sum-product algorithm on graphs with cycles

On a graph with cycles, there are several basic approaches to decoding.

One approach is to agglomerate the graph enough to eliminate the cycles, and then apply the sum-product algorithm, which will now be exact. The problem is that the complexity of decoding of a cycle-free graph of $\mathcal{C}$ cannot be significantly less than the complexity of decoding some trellis for $\mathcal{C}$, as we saw in Chapter 11. Moreover, as we saw in Chapter 10, the complexity of a minimal trellis for a sequence of codes with positive rates and coding gains must increase exponentially with code length.

A second approach is simply to apply the sum-product algorithm to the graph with cycles and hope for the best.

Because the sum-product rule is local, it may be implemented at any vertex of the graph, using whatever incoming messages are currently available. In a *parallel* or "flooding" schedule, the sum-product rule is computed at each vertex at all possible times, converting the incoming messages to a set of outgoing messages on all edges. Other schedules are possible, as we will discuss in the next chapter.

There is now no guarantee that the sum-product algorithm will converge. In practice, the sum-product algorithm converges with probability near 1 when the code rate is below some threshold which is below but near the Shannon limit. Convergence is slow when the code rate is near the threshold, but rapid when the code rate is somewhat lower. The identification of fixed points of the sum-product algorithm is a topic of current research.

Even if the sum-product algorithm converges, there is no guarantee that it will converge to the correct likelihoods or APPs. In general, the converged APPs will be too optimistic (overconfident), because they assume that all messages are from independent inputs, whereas in fact messages enter repeatedly into sum-product updates because of graph cycles. Consequently, decoding performance is suboptimal. In general, the suboptimality is great when the graph has many short cycles, and becomes negligible as cycles get long and sparse (the graph becomes "locally tree-like"). This is why belief propagation has long been considered to be inapplicable to most graphical models with cycles, which typically are based on physical models with inherently short cycles; in coding, by contrast, cycles can be designed to be very long with high probability.

A third approach is to beef up the sum-product algorithm so that it still performs well on certain classes of graphs with cycles. Because the sum-product algorithm already works so well in coding applications, this approach is not really needed for coding. However, this is a current topic of research for more general applications in artificial intelligence, optimization and physics.

Chapter 13

Capacity-approaching codes

We have previously discussed codes on graphs and the sum-product decoding algorithm in general terms. In this chapter we will give a brief overview of some particular classes of codes that can approach the Shannon limit quite closely: low-density parity-check (LDPC) codes, turbo codes, and repeat-accumulate (RA) codes. We will analyze long LDPC codes on the binary erasure channel (BEC), where exact results can be obtained. We will also sketch how to analyze any of these codes on symmetric binary-input channels.

13.1 LDPC codes

The oldest of these classes of codes is LDPC codes, invented by Gallager in his doctoral thesis (1961). These codes were introduced long before their time ("a bit of 21st-century coding that happened to fall in the 20th century"), and were almost forgotten until the mid-1990s, after the introduction of turbo codes, when they were independently rediscovered. Because of their simple structure, they have been the focus of much analysis. They have also proved to be capable of approaching the Shannon limit more closely than any other class of codes.

An LDPC code is based on the parity-check representation of a binary linear (n, k) block code $\mathcal{C}$; *i.e.,* $\mathcal{C}$ is the set of all binary n-tuples that satisfy the $n - k$ parity-check equations

$$\mathbf{x}H^T = \mathbf{0},$$

where H is a given $(n - k) \times n$ parity-check matrix. The basic idea of an LDPC code is that n should be large and H should be sparse; *i.e.,* the density of ones in H should be of the order of a small constant times n rather than n^2. As we will see, this sparseness makes it feasible to decode $\mathcal{C}$ by iterative sum-product decoding in linear time. Moreover, H should be pseudo-random, so that $\mathcal{C}$ will be a "random-like" code.

In Chapter 11, we saw that a parity-check representation of an (n, k) linear code leads to a Tanner graph with n variable nodes, $n - k$ constraint (zero-sum) nodes, and no state nodes. The number of edges is equal to the number of ones in the parity-check matrix H. This was illustrated by the Tanner graph of a parity-check representation for an $(8, 4, 4)$ code, shown again here as Figure 1.

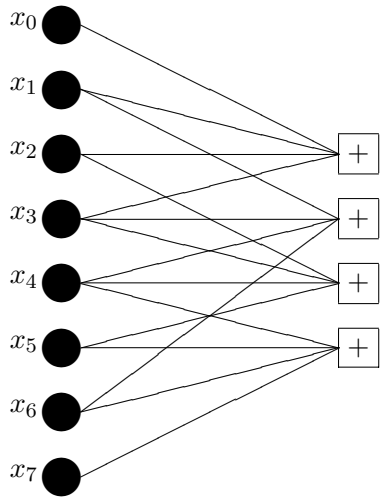


Figure 1. Tanner graph of parity-check representation for $(8, 4, 4)$ code.

Gallager's original LDPC codes were *regular*, meaning that every variable is involved in the same number d_λ of constraints, whereas every constraint checks the same number d_ρ of variables, where d_ρ and d_λ are small integers. The number of edges is thus $nd_\lambda = (n-k)d_\rho$, so the nominal rate $R = k/n$ of the code satisfies[1]

$$1 - R = \frac{n-k}{n} = \frac{d_\lambda}{d_\rho}.$$

Subject to this constraint, the connections between the nd_λ variable "sockets" and the $(n-k)d_\rho$ constraint sockets are made pseudo-randomly.

For example, the normal graph of a Gallager code with $d_\lambda = 3, d_\rho = 6$ and $R = \frac{1}{2}$ is shown in Figure 2. The large box labelled Π represents a pseudo-random permutation ("interleaver").

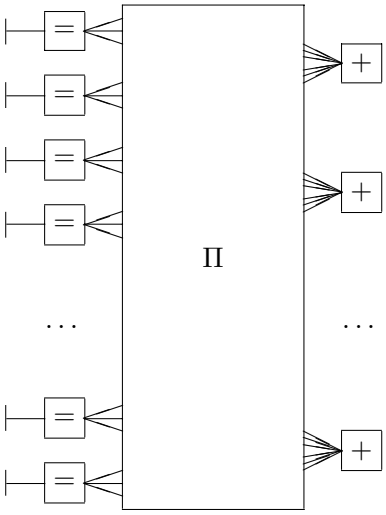


Figure 2. Normal graph of a regular $d_\lambda = 3, d_\rho = 6$ LDPC code.

[1]The actual rate of the code will be greater than R if the checks are not linearly independent. However, this fine point may be safely ignored.

Decoding an LDPC code is done by an iterative version of the sum-product algorithm, with a schedule that alternates between left nodes (repetition constraints) and right nodes (zero-sum constraints). The sparseness of the graph tends to insure that its girth (minimum cycle length) is reasonably large, so that the graph is locally tree-like. In decoding, this implies that the independence assumption holds for quite a few iterations. However, typically the number of iterations is large, so the independence assumption eventually becomes invalid.

The minimum distance of an LDPC code is typically large, so that actual decoding errors are hardly ever made. Rather, the typical decoding failure mode is a failure of the decoding algorithm to converge, which is of course a detectable failure.

The main improvement in recent years to Gallager's LDPC codes has been the use of irregular codes— *i.e.,* LDPC codes in which the left (variable) vertices and right (check) vertices have arbitrary degree distributions. The behavior of the decoding algorithm can be analyzed rather precisely using a technique called "density evolution," and the degree distributions can consequently be optimized, as we will discuss later in this chapter.

13.2 Turbo codes

The invention of turbo codes by Berrou *et al.* (1993) ignited great excitement about capacity-approaching codes. Initially, turbo codes were the class of capacity-approaching codes that were most widely used in practice, although LDPC codes may now be superseding turbo codes. Turbo codes achieve fairly low error rates within 1–2 dB of the Shannon limit at moderate block lengths ($n = 10^3$ to 10^4).

The original turbo codes of Berrou *et al.* are still some of the best that are known. Figure 3 shows a typical Berrou-type turbo code. An information bit sequence is encoded twice: first by an ordinary rate-$\frac{1}{2}$ systematic recursive (with feedback) convolutional encoder, and then, after a large pseudo-random permutation Π, by a second such encoder. The information sequence and the two parity sequences are transmitted, so the overall rate is $R = \frac{1}{3}$.

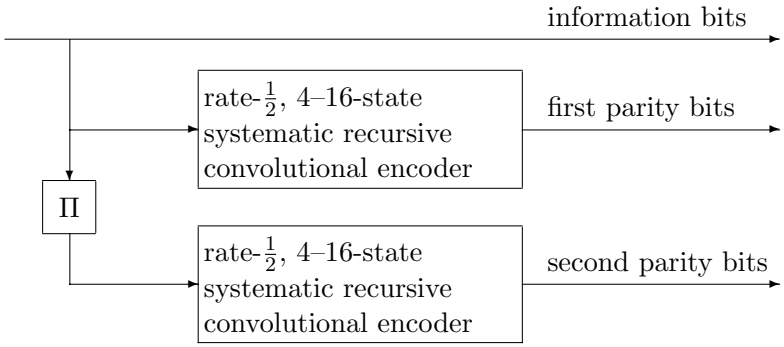


Figure 3. Rate-$\frac{1}{3}$ Berrou-type turbo code.

The two convolutional encoders are not very complicated, typically 4–16 states, and are often chosen to be identical. The convolutional codes are usually terminated to a finite block length. In practice the rate is often increased by puncturing.

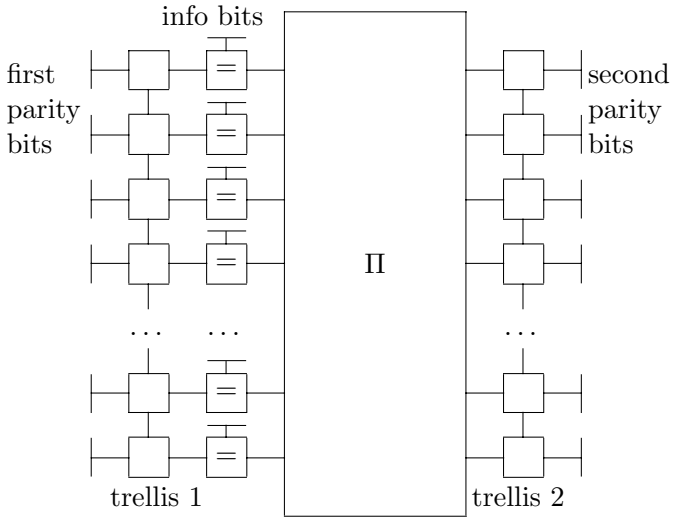


Figure 4. Normal graph of a Berrou-type turbo code.

Figure 4 is a normal graph of such a code. On both sides of the permutation Π are normal graphs representing the trellises of the two constituent convolutional codes, as in Figure 4(b) of Chapter 11. The information and parity bits are shown separately. The information bit sequences for the two trellises are identical, apart from the permutation Π.

Decoding a turbo code is done by an iterative version of the sum-product algorithm, with a schedule that alternates between the left trellis and the right trellis. On each trellis the sum-product algorithm reduces to the BCJR (APP) algorithm, so the decoder can efficiently decode the entire trellis before exchanging the resulting "extrinsic information" with the other trellis.

Again, the large permutation Π, combined with the recursive property of the encoders, tends to ensure that the girth of the graph is reasonably large, so the independence assumption holds for quite a few iterations. In turbo decoding the number of iterations is typically only 10–20, since much computation can be done along a trellis in each iteration.

The minimum distance of a Berrou-type turbo code is typically not very large. Although at low SNRs the decoding error probability tends to drop off rapidly above a threshold SNR (the "waterfall region") down to $10^{-4}, 10^{-5}$ or lower, for higher SNRs the error probability falls off more slowly due to low-weight error events (the "noise floor" region), and the decoder actually makes undetectable errors.

For applications in which these effects are undesirable, a different arrangement of two constituent codes is used, namely one after the other as in classical concatenated coding. Such codes are called "serial concatenated codes," whereas the Berrou-type codes are called "parallel concatenated codes." Serial concatenated codes usually still have an error floor, but typically at a considerably lower error rate than parallel concatenated codes. On the other hand, their threshold in the waterfall region tends to be worse than that of parallel concatenated codes. The "repeat-accumulate" codes of the next section are simple serial concatenated codes.

Analysis of turbo codes tends to be more *ad hoc* than that of LDPC codes. However, good *ad hoc* techniques for estimating decoder performance are now known (*e.g.,* the "extrinsic information transfer (EXIT) chart;" see below), which allow optimization of the component codes.

13.3 Repeat-accumulate codes

Repeat-accumulate (RA) codes were introduced by Divsalar, McEliece *et al.* (1998) as extremely simple "turbo-like" codes for which there was some hope of proving theorems. Surprisingly, even such simple codes proved to work quite well, within about 1.5 dB of the Shannon limit— *i.e.,* better than the best schemes known prior to turbo codes.

RA codes are a very simple class of serial concatenated codes. The outer code is a simple $(n, 1, n)$ repetition code, which simply repeats the information bits n times. The resulting sequence is then permuted by a large pseudo-random permutation Π. The inner code is a rate-1 2-state convolutional code with generator $g(D) = 1/(1 + D)$; *i.e.,* the input/output equation is $y_k = x_k + y_{k-1}$, so the output bit is simply the "accumulation" of all previous input bits (mod 2). The complete RA encoder is shown in Figure 5.

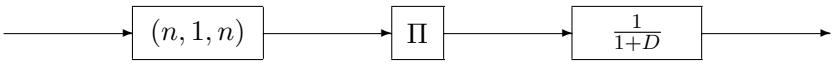


Figure 5. Rate-$\frac{1}{n}$ RA encoder.

The normal graph of a rate-$\frac{1}{3}$ RA code is shown in Figure 6. Since the original information bits are not transmitted, they are regarded as hidden state variables, repeated three times. On the right side, the states of the 2-state trellis are the output bits y_k, and the trellis constraints are represented explicitly by zero-sum nodes that enforce the constraints $y_k + x_k + y_{k-1} = 0$.

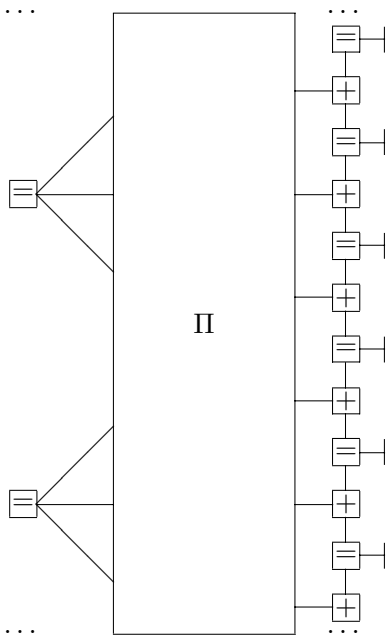


Figure 6. Normal graph of rate-$\frac{1}{3}$ RA code.

Decoding an RA code is again done by an iterative version of the sum-product algorithm, with a schedule that alternates between the left constraints and the right constraints, which in this case form a 2-state trellis. For the latter, the sum-product algorithm again reduces to the BCJR (APP) algorithm, so the decoder can efficiently decode the entire trellis in one iteration. A left-side iteration does not accomplish as much, but on the other hand it is extremely simple. As with LDPC codes, performance may be improved by making the left degrees irregular.

13.4 Analysis of LDPC codes on the binary erasure channel

In this section we will analyze the performance of iterative decoding of long LDPC codes on a binary erasure channel. This is one of the few scenarios in which exact analysis is possible. However, the results are qualitatively (and to a considerable extent quantitatively) indicative of what happens in more general scenarios.

13.4.1 The binary erasure channel

The binary erasure channel (BEC) models a memoryless channel with two inputs $\{0, 1\}$ and three outputs, $\{0, 1, ?\}$, where "?" is an "erasure symbol." The probability that any transmitted bit will be received correctly is $1 - p$, that it will be erased is p, and that it will be received incorrectly is zero. These transition probabilities are summarized in Figure 7 below.

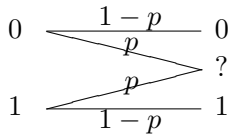


Figure 7. Transition probabilities of the binary erasure channel.

The binary erasure channel is an exceptional channel, in that a received symbol either specifies the transmitted symbol completely, or else gives no information about it. There are few physical examples of such binary-input channels. However, a Q-ary erasure channel (QEC) is a good model of packet transmission on the Internet, where (because of internal parity checks on packets) a packet is either received perfectly or not at all.

If a code sequence $\mathbf{c}$ from a binary code $\mathcal{C}$ is transmitted over a BEC, then the received sequence $\mathbf{r}$ will agree with $\mathbf{c}$ in all unerased symbols. If there is no other code sequence $\mathbf{c}' \in \mathcal{C}$ that agrees with $\mathbf{r}$ in all unerased symbols, then $\mathbf{c}$ is the only possible transmitted sequence, so a maximum-likelihood (ML) decoder can decide that $\mathbf{c}$ was sent with complete confidence. On the other hand, if there is another code sequence $\mathbf{c}' \in \mathcal{C}$ that agrees with $\mathbf{r}$ in all unerased symbols, then there is no way to decide between $\mathbf{c}$ and $\mathbf{c}'$, so a detectable decoding failure must occur. (We consider a random choice between $\mathbf{c}$ and $\mathbf{c}'$ to be a decoding failure.)

The channel capacity of a BEC with erasure probability p is $1 - p$, the fraction of unerased symbols, as would be expected intuitively. If feedback is available, then the channel capacity may be achieved simply by requesting retransmission of each erased symbol (the method used on the Internet Q-ary erasure channel).

Even without feedback, if we choose the 2^{nR} code sequences in a block code $\mathcal{C}$ of length n and rate R independently at random with each bit having probability $\frac{1}{2}$ of being 0 or 1, then as $n \to \infty$ the probability of a code sequence $\mathbf{c}' \in \mathcal{C}$ agreeing with the transmitted sequence $\mathbf{c}$ in all $\approx n(1-p)$ unerased symbols is about $2^{-n(1-p)}$, so by the union bound estimate the probability of decoding failure is about

$$\Pr(E) \approx 2^{nR} \cdot 2^{-n(1-p)},$$

which decreases exponentially with n as long as $R < 1 - p$. Thus capacity can be approached arbitrarily closely without feedback. On the other hand, if $R > 1-p$, then with high probability there will be only $\approx n(1-p) < nR$ unerased symbols, which can distinguish between at most $2^{n(1-p)} < 2^{nR}$ code sequences, so decoding must fail with high probability.

13.4.2 Iterative decoding of LDPC codes on the BEC

On the binary erasure channel, the sum-product algorithm is greatly simplified, because at any time every variable corresponding to every edge in the code graph is either known perfectly (unerased) or not known at all (erased). Iterative decoding using the sum-product algorithm therefore reduces simply to the propagation of unerased variables through the code graph.

There are only two types of nodes in a normal graph of an LDPC code (*e.g.,* Figure 2): repetition nodes and zero-sum nodes. If all variables are either correct or erased, then the sum-product update rule for a repetition node reduces simply to:

> If any incident variable is unerased, then all other incident variables may be set equal to that variable, with complete confidence; otherwise, all incident variables remain erased.

For a zero-sum node, the sum-product update rule reduces to:

> If all but one incident variable is unerased, then the remaining incident variable may be set equal to the mod-2 sum of those inputs, with complete confidence; otherwise, variable assignments remain unchanged.

Since all unerased variables are correct, there is no chance that these rules could produce a variable assignment that conflicts with another assignment.

Exercise 1. Using a graph of the $(8, 4, 4)$ code like that of Figure 1 for iterative decoding, decode the received sequence $(1, 0, 0, ?, 0, ?, ?, ?)$. Then try to decode the received sequence $(1, 1, 1, 1, ?, ?, ?, ?)$. Why does decoding fail in the latter case? Give both a local answer (based on the graph) and a global answer (based on the code). For the received sequence $(1, 1, 1, 1, ?, ?, ?, 0)$, show that iterative decoding fails but that global (*i.e.,* ML) decoding succeeds. □

13.4.3 Performance of large random LDPC codes

We now analyze the performance of iterative decoding for asymptotically large random LDPC codes on the BEC. Our method is a special case of a general method called *density evolution*, and is illustrated by a special case of the *EXtrinsic Information Transfer (EXIT) chart* technique.

We first analyze a random ensemble of regular (d_ρ, d_λ) LDPC codes of length n and rate $R = 1 - d_\lambda/d_\rho$. In this case all left nodes are repetition nodes of degree $d_\lambda + 1$, with one external (input) incident variable and d_λ internal (state) incident variables, and all right nodes are zero-sum nodes of degree d_ρ, with all incident variables being internal (state) variables. The random element is the permutation Π, which is chosen equiprobably from the set of all $(nd_\lambda)!$ permutations of $nd_\lambda = (n - k)d_\rho$ variables. We let $n \to \infty$ with (d_ρ, d_λ) fixed.

We use the standard sum-product algorithm, which alternates between sum-product updates of all left nodes and all right nodes. We track the progress of the algorithm by the expected fraction q of internal (state) variables that are still erased at each iteration. We will assume that at each node, the incident variables are independent of each other and of everything else; as $n \to \infty$, this "locally tree-like" assumption is justified by the large random interleaver.

At a repetition node, if the current probability of internal variable erasure is q_{in}, then the probability that a given internal incident variable will be erased as a result of its sum-product update is the probability that both the external incident variable and all $d_\lambda - 1$ other internal incident variables are erased, namely $q_{\text{out}} = p(q_{\text{in}})^{d_\lambda - 1}$.

On the other hand, at a zero-sum node, the probability that a given incident variable will not be erased is the probability that all $d_\rho - 1$ other internal incident variables are not erased, namely $(1 - q_{\text{in}})^{d_\rho - 1}$, so the probability that it *is* erased is $q_{\text{out}} = 1 - (1 - q_{\text{in}})^{d_\rho - 1}$.

These two functions are plotted in the "EXIT chart" of Figure 8 in the following manner. The two variable axes are denoted by $q_{r\to\ell}$ and $q_{\ell\to r}$, where the subscripts denote respectively left-going and right-going erasure probabilities. The range of both axes is from 1 (the initial value) to 0 (hopefully the final value). The two curves represent the sum-product update relationships derived above:

$$q_{\ell\to r} = p(q_{r\to\ell})^{d_\lambda - 1}; \qquad q_{r\to\ell} = 1 - (1 - q_{\ell\to r})^{d_\rho - 1}.$$

These curves are plotted for a regular $(d_\lambda = 3, d_\rho = 6)$ $(R = \frac{1}{2})$ LDPC code on a BEC with $p = 0.4$.

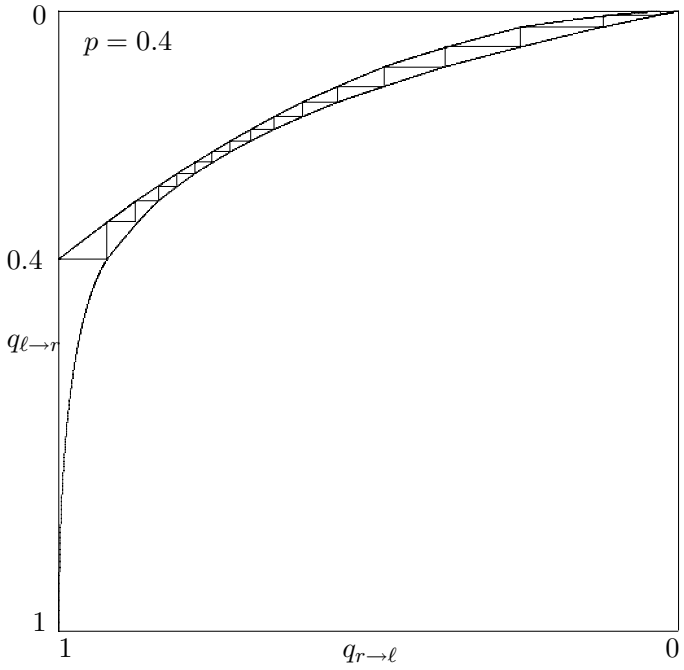


Figure 8. EXIT chart for iterative decoding of a regular $(d_\lambda = 3, d_\rho = 6)$ LDPC code on a BEC with $p = 0.4$.

A "simulation" of iterative decoding may then be performed as follows (also plotted in Figure 8). Initially, the left-going erasure probability is $q_{r\to\ell} = 1$. After a sum-product update in the left (repetition) nodes, the right-going erasure probability becomes $q_{\ell\to r} = p = 0.4$. After a sum-product update in the right (zero-sum) nodes, the left-going erasure probability becomes $q_{r\to\ell} = 1 - (0.6)^5 = 0.922$. Continuing, the erasure probability evolves as follows:

$$\begin{array}{lccccccccccccccc} q_{r\to\ell}: & 1 & & 0.922 & & 0.875 & & 0.839 & & 0.809 & & 0.780 & & 0.752 & \cdots \\ & & \searrow & & \nearrow & & \searrow & & \nearrow & & \searrow & & \nearrow & & \\ q_{\ell\to r}: & & 0.400 & & 0.340 & & 0.306 & & 0.282 & & 0.262 & & 0.244 & & \end{array}$$

$$\begin{array}{lccccccccccccccc} q_{r\to\ell}: & 0.723 & & 0.690 & & 0.653 & & 0.607 & & 0.550 & & 0.475 & & 0.377 & \cdots \\ & & \searrow & & \nearrow & & \searrow & & \nearrow & & \searrow & & \nearrow & & \\ q_{\ell\to r}: & & 0.227 & & 0.209 & & 0.191 & & 0.170 & & 0.147 & & 0.121 & & 0.090 & & 0.057 \end{array}$$

We see that iterative decoding must eventually drive the erasure probabilities to the top right corner of Figure 8, $q_{\ell\to r} = q_{r\to\ell} = 0$, because the two curves do not cross. It takes quite a few iterations, about 15 in this case, to get through the narrow "tunnel" where the two curves

approach each other closely. However, once past the "tunnel," convergence is rapid. Indeed, when both erasure probabilities are small, the result of one complete (left and right) iteration is

$$q_{r\to\ell}^{\text{new}} \approx 5q_{\ell\to r} = 5p(q_{r\to\ell}^{\text{old}})^2.$$

Thus both probabilities decrease rapidly (doubly exponentially) with the number of iterations.

The iterative decoder will fail if and only if the two curves touch; *i.e.,* if and only if there exists a pair $(q_{\ell\to r}, q_{r\to\ell})$ such that $q_{\ell\to r} = p(q_{r\to\ell})^{d_\lambda - 1}$ and $q_{r\to\ell} = 1 - (1 - q_{\ell\to r})^{d_\rho - 1}$, or equivalently the single-iteration update equation

$$q_{r\to\ell}^{\text{new}} = 1 - (1 - p(q_{r\to\ell}^{\text{old}})^{d_\lambda - 1})^{d_\rho - 1}$$

has a fixed point with $q_{r\to\ell}^{\text{new}} = q_{r\to\ell}^{\text{old}}$. For example, if $p = 0.45$ and $d_\lambda = 3, d_\rho = 6$, then this equation has a (first) fixed point at about $(q_{\ell\to r} \approx 0.35, q_{r\to\ell} \approx 0.89)$, as shown in Figure 9.

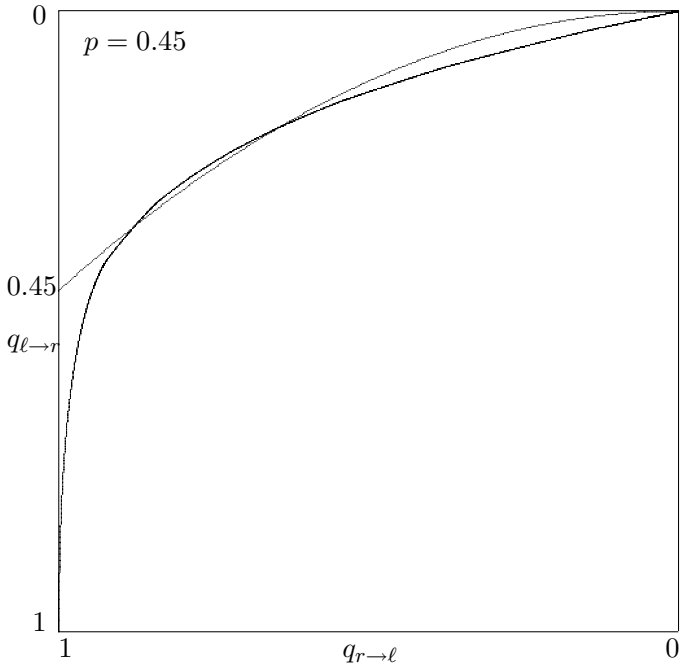


Figure 9. EXIT chart for iterative decoding of a regular $(d_\lambda = 3, d_\rho = 6)$ LDPC code on a BEC with $p = 0.45$.

Exercise 2. Perform a simulation of iterative decoding of a regular $(d_\lambda = 3, d_\rho = 6)$ LDPC code on a BEC with $p = 0.45$ (*i.e.,* on Figure 9), and show how decoding gets stuck at the first fixed point $(q_{\ell\to r} \approx 0.35, q_{r\to\ell} \approx 0.89)$. About how many iterations does it take to get stuck? By simulation of iterative decoding, compute the coordinates of the fixed point to six significant digits. □

We conclude that iterative decoding of a large regular $(d_\lambda = 3, d_\rho = 6)$ LDPC code, which has nominal rate $R = \frac{1}{2}$, will succeed on a BEC when the channel erasure probability is less than some threshold p^*, where $0.4 < p^* < 0.45$. The threshold p^* is the smallest p such that the equation $x = 1 - (1 - px^2)^5$ has a solution in the interval $0 < x < 1$.

Exercise 3. By analysis or simulation, show that $p^* = 0.429....$

13.4.4 Analysis of irregular LDPC codes

Now let us apply a similar analysis to large *irregular* LDPC codes, where the left nodes and/or right nodes do not necessarily all have the same degree. We characterize ensembles of such codes by the following parameters.

The number of external variables and left (repetition) nodes is the code length n; again we let $n \to \infty$. The number of internal variables (edges) will be denoted by E, which we will allow to grow linearly with n. The number of right (zero-sum) nodes will be $n - k = n(1 - R)$, yielding a nominal code rate of R.

A left node will be said to have degree d if it has d incident internal edges (*i.e.,* the external variable is not counted in its degree). The number of left nodes of degree d will be denoted by L_d. Thus $n = \sum_d L_d$. Similarly, the number of right nodes of degree d will be denoted by R_d, and $n(1 - R) = \sum_d R_d$. Thus the nominal rate R is given by

$$R = 1 - \frac{\sum_d R_d}{\sum_d L_d}.$$

An edge will be said to have left degree d if it is incident on a left node of degree d. The number of edges of left degree d will be denoted by ℓ_d; thus $\ell_d = dL_d$. Similarly, the number of edges of right degree d is $r_d = dR_d$. The total number of edges is thus

$$E = \sum_d \ell_d = \sum_d r_d.$$

It is helpful to define generating functions of these degree distributions as follows:

$$\begin{aligned} L(x) &= \sum_d L_d x^d; \\ R(x) &= \sum_d R_d x^d; \\ \ell(x) &= \sum_d \ell_d x^{d-1}; \\ r(x) &= \sum_d r_d x^{d-1}. \end{aligned}$$

Note that $L(1) = n$, $R(1) = n(1-R)$, and $\ell(1) = r(1) = E$. Also, note that $\ell(x)$ is the derivative of $L(x)$,

$$L'(x) = \sum_d L_d d x^{d-1} = \sum_d \ell_d x^{d-1} = \ell(x),$$

and similarly $R'(x) = r(x)$. Conversely, we have the integrals

$$\begin{aligned} L(x) &= \sum_d (\ell_d/d) x^d = \int_0^x \ell(y) dy; \\ R(x) &= \sum_d (r_d/d) x^d = \int_0^x r(y) dy. \end{aligned}$$

Finally, we have

$$R = 1 - \frac{R(1)}{L(1)} = 1 - \frac{\int_0^1 r(x)dx}{\int_0^1 \ell(x)dx}.$$

In the literature, it is common to normalize all of these generating functions by the total number of edges E; *i.e.,* we define $\lambda(x) = \ell(x)/E = \ell(x)/\ell(1)$, $\rho(x) = r(x)/E = r(x)/r(1)$, $\Lambda(x) = L(x)/E = \int_0^x \lambda(y)dy$, and $\mathrm{P}(x) = R(x)/E = \int_0^x \rho(y)dy$. In these terms, we have

$$R = 1 - \frac{\mathrm{P}(1)}{\Lambda(1)} = 1 - \frac{\int_0^1 \rho(x)dx}{\int_0^1 \lambda(x)dx}.$$

The *average left degree* is defined as $\overline{d}_\lambda = E/n$, and the *average right degree* as $\overline{d}_\rho = E/n(1-R)$; therefore $1/\overline{d}_\lambda = \Lambda(1)$, $1/\overline{d}_\rho = \mathrm{P}(1)$, and

$$R = 1 - \frac{\overline{d}_\lambda}{\overline{d}_\rho}.$$

The analysis of iterative decoding of irregular LDPC codes may be carried out nicely in terms of these generating functions. At a left node, if the current left-going probability of variable erasure is $q_{r\to\ell}$, then the probability that a given internal variable of left degree d will be erased as a result of a sum-product update is $p(q_{r\to\ell})^{d-1}$. The expected fraction of erased right-going variables is thus

$$q_{\ell\to r} = p\sum_d \lambda_d (q_{r\to\ell})^{d-1},$$

where $\lambda_d = \ell_d/E$ is the fraction of edges of left degree d. Thus

$$q_{\ell\to r} = p\lambda(q_{r\to\ell}),$$

where $\lambda(x) = \sum_d \lambda_d x^{d-1}$. Similarly, the expected fraction of erased left-going variables after a sum-product update at a right node is

$$q_{r\to\ell} = \sum_d \rho_d \left(1 - (1 - q_{\ell\to r})^{d-1}\right) = 1 - \rho(1 - q_{\ell\to r}),$$

where $\rho_d = r_d/E$ is the fraction of edges of right degree d, and $\rho(x) = \sum_d \rho_d x^{d-1}$.

These equations generalize the equations $q_{\ell\to r} = p(q_{r\to\ell})^{\lambda-1}$ and $q_{r\to\ell} = 1 - (1 - q_{\ell\to r})^{\rho-1}$ for the regular case. Again, these two curves may be plotted in an EXIT chart, and may be used for an exact calculation of the evolution of the erasure probabilities $q_{\ell\to r}$ and $q_{r\to\ell}$ under iterative decoding. And again, iterative decoding will be successful if and only if the two curves do not cross. The fixed-point equation now becomes

$$x = 1 - \rho(1 - p\lambda(x)).$$

Design of a capacity-approaching LDPC code therefore becomes a matter of choosing the left and right degree distributions $\lambda(x)$ and $\rho(x)$ so that the two curves come as close to each other as possible, without touching.

13.4.5 Area theorem

The following lemma and theorem show that in order to approach the capacity $C = 1 - p$ of the BEC arbitrarily closely, the two EXIT curves must approach each other arbitrarily closely; moreover, if the rate R exceeds C, then the two curves must cross.

Lemma 13.1 (Area theorem) *The area under the curve $q_{\ell\to r} = p\lambda(q_{r\to\ell})$ is $p/\overline{d}_\lambda$, while the area under the curve $q_{r\to\ell} = 1-\rho(1-q_{\ell\to r})$ is $1-1/\overline{d}_\rho$.*

Proof.

$$\begin{aligned}\int_0^1 p\lambda(x)dx &= p\Lambda(1) = \frac{p}{\overline{d}_\lambda};\\ \int_0^1 (1-\rho(1-x))dx &= \int_0^1 (1-\rho(y))dy = 1-\mathrm{P}(1) = 1-\frac{1}{\overline{d}_\rho}.\end{aligned}$$

Theorem 13.2 (Converse capacity theorem) *For successful iterative decoding of an irregular LDPC code on a BEC with erasure probability p, the code rate R must be less than the capacity $C = 1-p$. As $R \to C$, the two EXIT curves must approach each other closely, but not cross.*

Proof. For successful decoding, the two EXIT curves must not intersect, which implies that the regions below the two curves must be disjoint. This implies that the sum of the areas of these regions must be less than the area of the EXIT chart, which is 1:

$$p\Lambda(1) + 1 - \mathrm{P}(1) < 1.$$

This implies that $p < \mathrm{P}(1)/\Lambda(1) = 1-R$, or equivalently $R < 1-p = C$. If $R \approx C$, then $p\Lambda(1)+1-\mathrm{P}(1) \approx 1$, which implies that the union of the two regions must nearly fill the whole EXIT chart, whereas for successful decoding the two regions must remain disjoint. □

13.4.6 Stability condition

Another necessary condition for the two EXIT curves not to cross is obtained by considering the curves near the top right point $(0,0)$. For $q_{r\to\ell}$ small, we have the linear approximation

$$q_{\ell\to r} = p\lambda(q_{r\to\ell}) \approx p\lambda'(0)q_{r\to\ell}.$$

Similarly, for $q_{\ell\to r}$ small, we have

$$q_{r\to\ell} = 1-\rho(1-q_{\ell\to r}) \approx 1-\rho(1)+\rho'(1)q_{\ell\to r} = \rho'(1)q_{\ell\to r},$$

where we use $\rho(1) = \sum_d \rho_d = 1$. After one complete iteration, we therefore have

$$q_{r\to\ell}^{\text{new}} \approx p\lambda'(0)\rho'(1)q_{r\to\ell}^{\text{old}},$$

The erasure probability $q_{r\to\ell}$ is thus reduced on a complete iteration if and only if

$$p\lambda'(0)\rho'(1) < 1.$$

This is known as the *stability condition* on the degree distributions $\lambda(x)$ and $\rho(x)$. Graphically, it ensures that the line $q_{\ell\to r} \approx p\lambda'(0)q_{r\to\ell}$ lies above the line $q_{r\to\ell} \approx \rho'(1)q_{\ell\to r}$ near the point $(0,0)$, which is a necessary condition for the two curves not to cross.

Exercise 4. Show that if the minimum left degree is 3, then the stability condition necessarily holds. Argue that such a degree distribution $\lambda(x)$ cannot be capacity-approaching, however, in view of Theorem 13.2. □

Luby, Shokrollahi *et al.* have shown that for any $R < C$ it is possible to design $\lambda(x)$ and $\rho(x)$ so that the nominal code rate is R and the two curves do not cross, so that iterative decoding will be successful. Thus iterative decoding of LDPC codes solves the longstanding problem of approaching capacity arbitrarily closely on the BEC with a linear-time decoding algorithm, at least for asymptotically long codes.

13.5 LDPC code analysis on symmetric binary-input channels

In this final section, we will sketch how to analyze a long LDPC code on a general symmetric binary-input channel (SBIC). Density evolution is exact, but is difficult to compute. EXIT charts give good approximate results.

13.5.1 Symmetric binary-input channels

The binary symmetric channel (BSC) models a memoryless channel with binary inputs $\{0, 1\}$ and binary outputs $\{0, 1\}$. The probability that any transmitted bit will be received correctly is $1 - p$, and that it will be received incorrectly is p. This model is depicted in Figure 10 below.

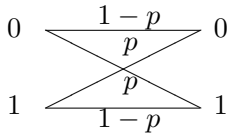


Figure 10. Transition probabilities of the binary symmetric channel.

By symmetry, the capacity of a BSC is attained when the two input symbols are equiprobable. In this case, given a received symbol y, the *a posteriori* probabilities $\{p(0 \mid y), p(1 \mid y)\}$ of the transmitted symbols are $\{1 - p, p\}$ for $y = 0$ and $\{p, 1 - p\}$ for $y = 1$. The conditional entropy $H(X \mid y)$ is thus equal to the binary entropy function $\mathcal{H}(p) = -p \log_2 p - (1 - p) \log_2(1 - p)$, independent of y. The channel capacity is therefore equal to $C = H(X) - H(X \mid Y) = 1 - \mathcal{H}(p)$.

A general *symmetric binary-input channel* (SBIC) is a channel with binary inputs $\{0, 1\}$ that may be viewed as a mixture of binary symmetric channels, as well as possibly a binary erasure channel. In other words, the channel output alphabet may be partitioned into pairs $\{a, b\}$ such that $p(a \mid 0) = p(b \mid 1)$ and $p(a \mid 1) = p(b \mid 0)$, as well as possibly a singleton $\{?\}$ such that $p(? \mid 0) = p(? \mid 1)$.

Again, by symmetry, the capacity of a SBIC is attained when the two input symbols are equiprobable. Then the *a posteriori* probabilities (APPs) are symmetric, in the sense that for any such pair $\{a, b\}$, $\{p(0 \mid y), p(1 \mid y)\}$ equals $\{1 - p, p\}$ for $y = a$ and $\{p, 1 - p\}$ for $y = b$, where

$$p = \frac{p(a \mid 1)}{p(a \mid 0) + p(a \mid 1)} = \frac{p(b \mid 0)}{p(b \mid 0) + p(b \mid 1)}.$$

Similarly, $\{p(0 \mid ?), p(1 \mid ?)\} = \{\frac{1}{2}, \frac{1}{2}\}$.

We may characterize a symmetric binary-input channel by the probability distribution of the APP parameter $p = p(1 \mid y)$ under the conditional distribution $p(y \mid 0)$, which is the same as the distribution of $p = p(0 \mid y)$ under $p(y \mid 1)$. This probability distribution is in general continuous if the output distribution is continuous, or discrete if it is discrete.[2]

Example 1. Consider a binary-input channel with five outputs $\{y_1, y_2, y_3, y_4, y_5\}$ such that $\{p(y_j \mid 0)\} = \{p_1, p_2, p_3, p_4, p_5\}$ and $\{p(y_j \mid 1)\} = \{p_5, p_4, p_3, p_2, p_1\}$. This channel is a SBIC, because the outputs may be grouped into symmetric pairs $\{y_1, y_5\}$ and $\{y_2, y_4\}$ and a singleton $\{y_3\}$ satisfying the symmetry conditions given above. The values of the APP parameter p are then $\{\frac{p_5}{p_1+p_5}, \frac{p_4}{p_2+p_4}, \frac{1}{2}, \frac{p_2}{p_2+p_4}, \frac{p_1}{p_1+p_5}\}$; their probability distribution is $\{p_1, p_2, p_3, p_4, p_5\}$. □

Example 2. Consider a binary-input Gaussian-noise channel with inputs $\{\pm 1\}$ and Gaussian conditional probability density $p(y \mid x) = (2\pi\sigma)^{-1/2} \exp -(y-x)^2/2\sigma^2$. This channel is a SBIC, because the outputs may be grouped into symmetric pairs $\{\pm y\}$ and a singleton $\{0\}$ satisfying the symmetry conditions given above. The APP parameter is then

$$p = \frac{p(y \mid -1)}{p(y \mid 1) + p(y \mid -1)} = \frac{e^{-y/\sigma^2}}{e^{y/\sigma^2} + e^{-y/\sigma^2}},$$

whose probability density is induced by the conditional density $p(y \mid +1)$. □

Given an output y, the conditional entropy $H(X \mid y)$ is given by the binary entropy function, $H(X \mid y) = \mathcal{H}(p(0 \mid y)) = \mathcal{H}(p(1 \mid y))$. The average conditional entropy is thus

$$H(X \mid Y) = \mathsf{E}_{Y|0}[H(X \mid y)] = \int dy \; p(y \mid 0) \; \mathcal{H}(p(0 \mid y)),$$

where we use notation that is appropriate for the continuous case. The channel capacity is then $C = H(X) - H(X \mid Y) = 1 - H(X \mid Y)$ bits per symbol.

Example 1. For the five-output SBIC of Example 1,

$$H(X \mid Y) = (p_1 + p_5)\mathcal{H}\left(\frac{p_1}{p_1 + p_5}\right) + (p_2 + p_4)\mathcal{H}\left(\frac{p_2}{p_2 + p_4}\right) + p_3. \qquad \square$$

Example 3. A binary erasure channel with erasure probability p is a SBIC with probabilities $\{1-p, p, 0\}$ of the APP parameter being $\{0, \frac{1}{2}, 1\}$, and thus of $H(X \mid y)$ being $\{0, 1, 0\}$. Therefore $H(X \mid Y) = p$ and $C = 1 - p$. □

13.5.2 Sum-product decoding of an LDPC code on a SBIC

For an LDPC code on a SBIC, the sum-product iterative decoding algorithm is still quite simple, because all variables are binary, and all constraint nodes are either repetition or zero-sum nodes.

For binary variables, APP vectors are always of the form $\{1-p, p\}$, up to scale; *i.e.,* they are specified by a single parameter. We will not worry about scale in implementing the sum-product algorithm, so in general we will simply have unnormalized APP weights $\{w_0, w_1\}$, from which can recover p using the equation $p = w_1/(w_0 + w_1)$. Some implementations use likelihood ratios $\lambda = w_0/w_1$, from which we can recover p using $p = 1/(1+\lambda)$. Some use log likelihood ratios $\Lambda = \ln \lambda$, from which we can recover p using $p = 1/(1+e^{\Lambda})$.

[2]Note that any outputs with the same APP parameter may be combined without loss of optimality, since the APPs form a set of sufficient statistics for estimation of the input from the output.

For a repetition node, the sum-product update rule reduces simply to the product update rule, in which the components of the incoming APP vectors are multiplied componentwise: *i.e.,*

$$w_0^{\text{out}} = \prod_j w_0^{\text{in},j}; \qquad w_1^{\text{out}} = \prod_j w_1^{\text{in},j},$$

where $\{w_{x_j}^{\text{in},j} \mid x_j \in \{0,1\}\}$ is the jth incoming APP weight vector. Equivalently, the output likelihood ratio is the product of the incoming likelihood ratios; or, the output log likelihood ratio is the sum of the incoming log likelihood ratios.

For a zero-sum node, the sum-product update rule reduces to

$$w_0^{\text{out}} = \sum_{\sum x_j = 0} \prod_j w_{x_j}^{\text{in},j}; \qquad w_1^{\text{out}} = \sum_{\sum x_j = 1} \prod_j w_{x_j}^{\text{in},j}.$$

An efficient implementation of this rule is as follows:[3]

(a) Transform each incoming APP weight vector $\{w_0^{\text{in},j}, w_1^{\text{in},j}\}$ by a 2×2 Hadamard transform to $\{W_0^{\text{in},j} = w_0^{\text{in},j} + w_1^{\text{in},j}, W_1^{\text{in},j} = w_0^{\text{in},j} - w_1^{\text{in},j}\}$.

(b) Form the componentwise product of the transformed vectors; *i.e.,*

$$W_0^{\text{out}} = \prod_j W_0^{\text{in},j}; \qquad W_1^{\text{out}} = \prod_j W_1^{\text{in},j}.$$

(c) Transform the outgoing APP weight vector $\{W_0^{\text{out}}, W_1^{\text{out}}\}$ by a 2×2 Hadamard transform to $\{w_0^{\text{out}} = W_0^{\text{out}} + W_1^{\text{out}}, w_1^{\text{out}} = W_0^{\text{out}} - W_1^{\text{out}}\}$.

Exercise 5. (Sum-product update rule for zero-sum nodes)

(a) Prove that the above algorithm implements the sum-product update rule for a zero-sum node, up to scale. [Hint: observe that in the product $\prod_j (w_0^{\text{in},j} - w_1^{\text{in},j})$, the terms with positive signs sum to w_0^{out}, whereas the terms with negative signs sum to w_1^{out}.]

(b) Show that if we interchange $w_0^{\text{in},j}$ and $w_1^{\text{in},j}$ in an even number of incoming APP vectors, then the outgoing APP vector $\{w_0^{\text{out}}, w_1^{\text{out}}\}$ is unchanged. On the other hand, show that if we interchange $w_0^{\text{in},j}$ and $w_1^{\text{in},j}$ in an odd number of incoming APP vectors, then the components w_0^{out} and w_1^{out} of the outgoing APP vector are interchanged.

(c) Show that if we replace APP weight vectors $\{w_0, w_1\}$ by log likelihood ratios $\Lambda = \ln w_0/w_1$, then the zero-sum sum-product update rule reduces to the "tanh rule"

$$\Lambda^{\text{out}} = \ln \left(\frac{1 + \prod_j \tanh \Lambda^{\text{in},j}/2}{1 - \prod_j \tanh \Lambda^{\text{in},j}/2} \right),$$

where the hyperbolic tangent is defined by $\tanh x = (e^x - e^{-x})/(e^x + e^{-x})$.

(d) Show that the "tanh rule" may alternatively be written as

$$\tanh \Lambda^{\text{out}}/2 = \prod_j \tanh \Lambda^{\text{in},j}/2.$$

[3] This implementation is an instance of a more general principle: the sum-product update rule for a constraint code $\mathcal{C}$ may be implemented by Fourier-transforming the incoming APP weight vectors, performing a sum-product update for the dual code $\mathcal{C}^\perp$, and then Fourier-transforming the resulting outgoing APP weight vectors. For a binary alphabet, the Fourier transform reduces to the 2×2 Hadamard transform.

13.5.3 Density evolution

The performance of iterative decoding for asymptotically large random LDPC codes on a general SBIC may in principle be simulated exactly by tracking the probability distribution of the APP parameter p (or an equivalent parameter). This is called *density evolution*. However, whereas on the BEC there are only two possible values of p, $\frac{1}{2}$ (complete ignorance) and 0 (complete certainty), so that we need to track only the probability q of the former, in density evolution we need to track a general probability distribution for p. In practice, this cannot be done with infinite precision, so density evolution becomes inexact to some extent.

On an SBIC, the channel symmetry leads to an important simplification of density evolution: we may always assume that the all-zero codeword was sent, which means that for each variable we need to track only the distribution of p given that the value of the variable is 0.

To justify this simplification, note that any other codeword imposes a configuration of variables on the code graph such that all local constraints are satisfied; *i.e.,* all variables incident on a repetition node are equal to 0 or to 1, while the values of the set of variables incident on a zero-sum node include an even number of 1s. Now note that at a repetition node, if we interchange $w_0^{\text{in},j}$ and $w_1^{\text{in},j}$ in all incoming APP vectors, then the components w_0^{out} and w_1^{out} of the outgoing APP vector are interchanged. Exercise 5(b) proves a comparable result for zero-sum nodes. So if the actual codeword is not all-zero, but the channel is symmetric, so for every symbol variable which has a value 1 the initial APP vectors are simply interchanged, then the APP vectors will evolve during sum-product decoding in precisely the same way as they would have evolved if the all-zero codeword had been sent, except that wherever the underlying variable has value 1, the APP components are interchanged.

In practice, to carry out density evolution for a given SBIC with given degree distributions $\lambda(x)$ and $\rho(x)$, the probability density is quantized to a discrete probability distribution, using as many as 12–14 bits of accuracy. The repetition and check node distribution updates are performed in a computationally efficient manner, typically using fast Fourier transforms for the former and a table-driven recursive implementation of the "tanh rule" for the latter. For a given SBIC model, the simulation is run iteratively until either the distribution of p tends to a delta function at $p = 0$ (success), or else the distribution converges to a nonzero fixed point (failure). Repeated runs allow a success threshold to be determined. Finally, the degree distributions $\lambda(x)$ and $\rho(x)$ may be optimized by using hill-climbing techniques (iterative linear programming).

For example, in his thesis (2000), Chung designed rate-$\frac{1}{2}$ codes with asymptotic thresholds within 0.0045 dB of the Shannon limit, and with performance within 0.040 dB of the Shannon limit at an error rate of 10^{-6} with a block length of $n = 10^7$; see Figure 11. The former code has left degrees $\{2, 3, 6, 7, 15, 20, 50, 70, 100, 150, 400, 900, 2000, 3000, 6000, 8000\}$, with average left degree $\overline{d}_\lambda = 9.25$, and right degrees $\{18, 19\}$, with average right degree $\overline{d}_\rho = 18.5$. The latter code has left degrees $\{2, 3, 6, 7, 18, 19, 55, 56, 200\}$, with $\overline{d}_\lambda = 6$, and all right degrees equal to 12.

In current research, more structured constructions of the parity-check matrix H (or equivalently the permutation Π) are being sought for shorter block lengths, of the order of 1000.

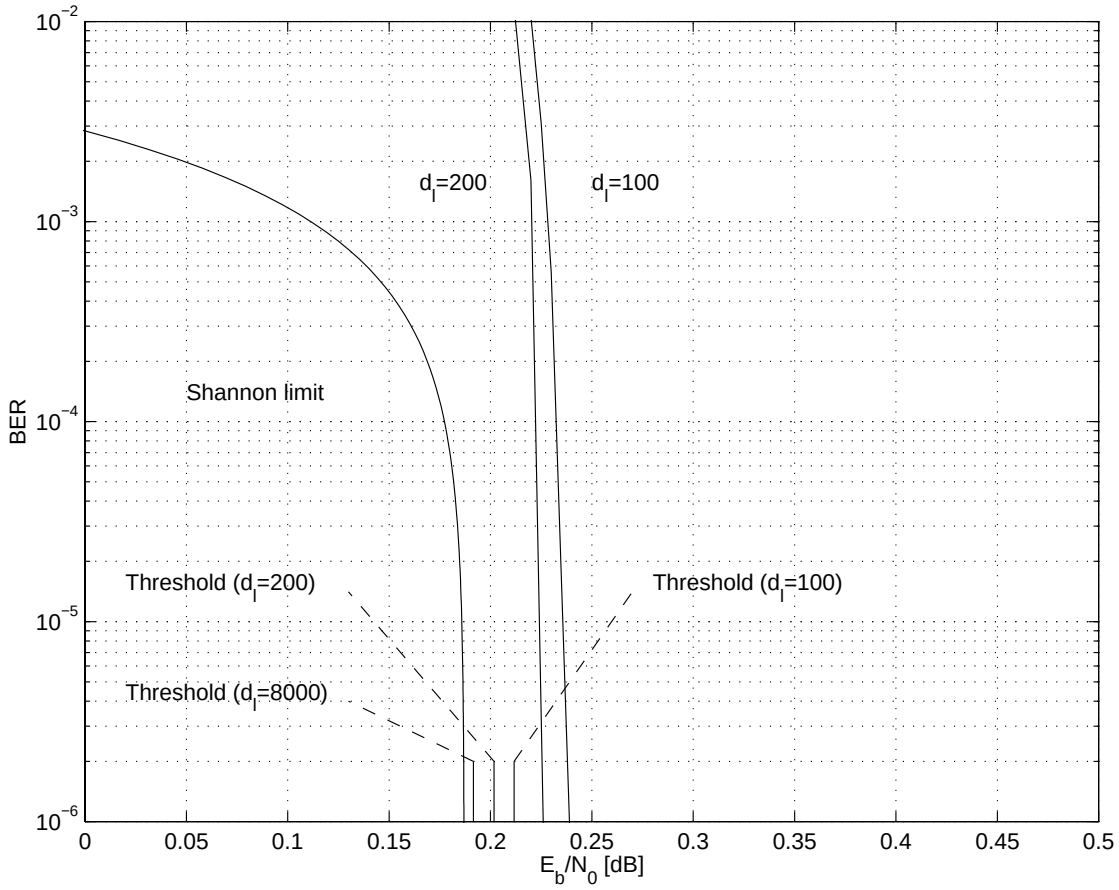


Figure 11. Asymptotic analysis with maximum left degree $d_l = 100, 200, 8000$ and simulations with $d_l = 100, 200$ and $n = 10^7$ of optimized rate-$\frac{1}{2}$ irregular LDPC codes [Chung *et al.*, 2001].

13.5.4 EXIT charts

For large LDPC codes, an efficient and quite accurate heuristic method of simulating sum-product decoding performance is to replace the probability distribution of the APP parameter p by a single summary statistic: most often, the conditional entropy $H(X \mid Y) = \mathsf{E}_{Y|0}[H(X \mid y)]$, or equivalently the mutual information $I(X;Y) = 1 - H(X \mid Y)$. We have seen that on the BEC, where $H(X \mid Y) = p$, this approach yields an exact analysis.

Empirical evidence shows that the relations between $H(X \mid Y)^{\text{out}}$ and $H(X \mid Y)^{\text{in}}$ for repetition and zero-sum nodes are very similar for all SBICs. This implies that degree distributions $\lambda(z), \rho(z)$ designed for the BEC may be expected to continue to perform well for other SBICs.

Similar EXIT chart analyses may be performed for turbo codes, RA codes, or for any codes in which there are left codes and right codes whose variables are shared through a large pseudo-random interleaver. In these cases the two relations between $H(X \mid Y)^{\text{out}}$ and $H(X \mid Y)^{\text{in}}$ often cannot be determined by analysis, but rather are measured empirically during simulations. Again, design is done by finding distributions of left and right codes such that the two resulting curves approach each other closely, without crossing. All of these classes of codes have now been optimized to approach capacity closely.

Chapter 14

Introduction to lattice and trellis codes

In this chapter we discuss coding techniques for bandwidth-limited (high-SNR) AWGN channels.

On bandwidth-limited channels, nonbinary signal alphabets such as M-PAM must be used to approach capacity. Furthermore, the signals should be used with a nonuniform, Gaussian-like probability distribution.

Using large-alphabet approximations, we show that the total coding gain of a coded modulation scheme for the bandwidth-limited AWGN channel is the sum of a coding gain due to a denser packing than the baseline M-PAM scheme, plus a shaping gain due to constellation shaping (or equivalently to use of a nonuniform distribution). At high SNRs, the coding and shaping problems are separable.

The maximum possible shaping gain is a factor of $\pi e/6$ (1.53 dB). Simple shaping methods such as shell mapping and trellis shaping can easily obtain of the order of 1 dB of shaping gain.

For moderate coding gains at moderate complexity, the two principal classes of packings are lattices and trellis codes, which are analogous to block and convolutional codes, respectively. By now the principles of construction of the best such codes are well understood, and it seems likely that the best codes have been found. We plot the effective coding gains of these known moderate-complexity lattices and trellis codes versus the branch complexity of their minimal trellises, assuming ML decoding. Trellis codes are somewhat superior, due mainly to their lower error coefficients.

We briefly mention higher-performance schemes, including multilevel schemes with multistage decoding and bit-interleaved coded modulation, which allow the use of high-performance binary codes such as those described in the previous chapter to approach capacity.

14.1 Lattices

It is clear from Shannon's capacity theorem that an optimal block code for a bandwidth-limited AWGN channel consists of a dense packing of code points within a sphere in a high-dimensional Euclidean space. Most of the densest known packings are lattices.

In this section we briefly describe lattice constellations, and analyze their performance using the union bound estimate and large-constellation approximations.

An n-dimensional (n-D) *lattice* Λ is a discrete subset of n-space $\mathbb{R}^n$ that has the group property. Without essential loss of generality, Λ may be assumed to span $\mathbb{R}^n$. The points of the lattice then form a uniform infinite packing of $\mathbb{R}^n$.

Example 1. The set of integers $\mathbb{Z}$ is a one-dimensional lattice, since $\mathbb{Z}$ is a discrete subgroup of $\mathbb{R}$. Any 1-dimensional lattice is of the form $\Lambda = \alpha\mathbb{Z}$ for some scalar $\alpha > 0$. □

Example 2. The *integer lattice* $\mathbb{Z}^n$ (the set of integer n-tuples) is an n-dimensional lattice for any $n \geq 1$. □

Example 3. The *hexagonal lattice* $A_2 = \{a(1,0) + b(\frac{1}{2}, \frac{\sqrt{3}}{2}) \mid (a,b) \in \mathbb{Z}^2\}$ is illustrated in Figure 1. This lattice is the densest packing of $\mathbb{R}^2$. □

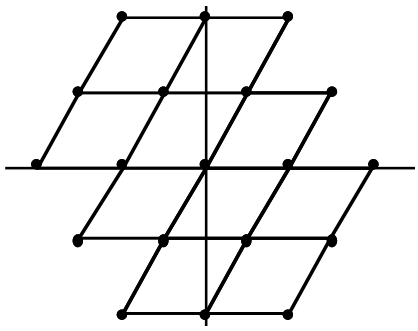


Figure 1. *The hexagonal lattice A_2.*

Exercise 1. Let $\mathcal{C}$ be an (n,k,d) binary linear block code. Show that

$$\Lambda_{\mathcal{C}} = \{\mathbf{x} \in \mathbb{Z}^n \mid \mathbf{x} \equiv \mathbf{c} \bmod 2 \text{ for some } \mathbf{c} \in \mathcal{C}\} \tag{14.1}$$

is an n-dimensional sublattice of $\mathbb{Z}^n$ (called a "Construction A" or "mod-2" lattice).

A general n-dimensional lattice Λ that spans $\mathbb{R}^n$ may be characterized by a set of linearly independent generators $G = \{\mathbf{g}_j,\ 1 \leq j \leq n\}$ such that Λ is the set of all integer linear combinations of the generators:

$$\Lambda = \{\mathbf{a}G = \sum_j a_j \mathbf{g}_j \mid \mathbf{a} \in \mathbb{Z}^n\}. \tag{14.2}$$

Thus Λ may be viewed as the image of the integer lattice $\mathbb{Z}^n$ under a linear transformation of n-space $\mathbb{R}^n$ by the linear operator G, as illustrated by Figure 1.

By the group property of Λ, any translate $\Lambda + \mathbf{x}$ by a lattice point $\mathbf{x} \in \Lambda$ is just Λ again. This implies that a lattice is "geometrically uniform;" every point of the lattice has the same number of neighbors at each distance, and all decision regions of a minimum-distance decoder ("Voronoi regions") are congruent and form a tessellation of $\mathbb{R}^n$. Indeed, any lattice translate $\Lambda + \mathbf{t}$ is geometrically uniform.

The key geometrical parameters of a lattice are:

- the *minimum squared distance* $d^2_{\min}(\Lambda)$ between lattice points;
- the *kissing number* $K_{\min}(\Lambda)$ (the number of nearest neighbors to any lattice point);
- the *volume* $V(\Lambda)$ of n-space per lattice point. As indicated in Figure 1, this volume is the volume of the *fundamental parallelotope*

$$[0,1)^n G = \{\mathbf{a}G \mid \mathbf{a} \in [0,1)^n\}.$$

Since the volume of the n-cube $[0,1)^n$ is 1 and the Jacobian of the linear transformation G is its determinant $|G|$, it follows that $V(\Lambda) = |G|$ for any generator matrix G of Λ.

The *Hermite parameter* of Λ is the normalized density parameter

$$\gamma_c(\Lambda) = \frac{d^2_{\min}(\Lambda)}{V(\Lambda)^{2/n}}, \tag{14.3}$$

which we will shortly identify as its nominal coding gain. The quantity $V(\Lambda)^{2/n}$ may be thought of as the normalized volume of Λ per two dimensions.

Example 3 (cont.) For the hexagonal lattice A_2, the minimum squared distance is $d^2_{\min}(A_2) = 1$, the kissing number is $K_{\min}(A_2) = 6$, the volume is $V(A_2) = \sqrt{3}/2$, and the Hermite parameter is $\gamma_c(A_2) = 2/\sqrt{3} = 1.155$ (0.62 dB). Therefore A_2 is denser than the integer lattice $\mathbb{Z}^2$, for which $d^2_{\min}(\mathbb{Z}^2) = V(\mathbb{Z}^2) = \gamma_c(\mathbb{Z}^2) = 1$. □

Exercise 1 (cont.) Show that if $\mathcal{C}$ is an (n,k,d) binary linear block code with N_d weight-d words, then the mod-2 lattice $\Lambda_{\mathcal{C}}$ has the following geometrical parameters:

$$d^2_{\min}(\Lambda_{\mathcal{C}}) = \min\{d, 4\}; \tag{14.4}$$

$$K_{\min}(\Lambda_{\mathcal{C}}) = \begin{cases} 2^d N_d, & \text{if } d < 4; \\ 2n, & \text{if } d > 4; \\ 2^d N_d + 2n, & \text{if } d = 4; \end{cases} \tag{14.5}$$

$$V(\Lambda_{\mathcal{C}}) = 2^{n-k}; \tag{14.6}$$

$$\gamma_c(\Lambda_{\mathcal{C}}) = \frac{d^2_{\min}(\Lambda_{\mathcal{C}})}{2^{\eta(\mathcal{C})}}, \tag{14.7}$$

where $\eta(\mathcal{C}) = 2(n-k)/n$ is the redundancy of $\mathcal{C}$ in bits per two dimensions. □

Exercise 2. Show that $\gamma_c(\Lambda)$ is invariant to scaling, orthogonal transformations, and Cartesian products; *i.e.*, $\gamma_c(\alpha U \Lambda^m) = \gamma_c(\Lambda)$, where $\alpha > 0$ is any scale factor, U is any orthogonal matrix, and $m \geq 1$ is any positive integer. Show that $\gamma_c(\alpha U \mathbb{Z}^n) = 1$ for any version $\alpha U \mathbb{Z}^n$ of any integer lattice $\mathbb{Z}^n$. □

14.2 Lattice constellations

A *lattice constellation*

$$\mathcal{C}(\Lambda, \mathcal{R}) = (\Lambda + \mathbf{t}) \cap \mathcal{R} \tag{14.8}$$

is the finite set of points in a lattice translate $\Lambda + \mathbf{t}$ that lie within a compact bounding region $\mathcal{R}$ of n-space.

Example 4. An M-PAM constellation $\alpha\{\pm 1, \pm 3, \ldots, \pm(M-1)\}$ is a one-dimensional lattice constellation $\mathcal{C}(2\alpha\mathbb{Z}, \mathcal{R})$ with $\Lambda + \mathbf{t} = 2\alpha(\mathbb{Z} + 1)$ and $\mathcal{R} = [-\alpha M, \alpha M]$. □

The key geometric properties of the region $\mathcal{R}$ are

- its *volume* $V(\mathcal{R}) = \int_{\mathcal{R}} d\mathbf{x}$;
- the *average energy* $P(\mathcal{R})$ per dimension of a uniform probability density function over $\mathcal{R}$:

$$P(\mathcal{R}) = \int_{\mathcal{R}} \frac{\|\mathbf{x}\|^2}{n} \frac{d\mathbf{x}}{V(\mathcal{R})}. \tag{14.9}$$

The *normalized second moment* of $\mathcal{R}$ is defined as the dimensionless parameter

$$G(\mathcal{R}) = \frac{P(\mathcal{R})}{V(\mathcal{R})^{2/n}}. \tag{14.10}$$

Example 4 (cont.). The key geometrical parameters of $\mathcal{R} = [-\alpha M, \alpha M]$ are $V(\mathcal{R}) = 2\alpha M$, $P(\mathcal{R}) = \alpha^2 M^2/3$, and $G(\mathcal{R}) = 1/12$. □

Exercise 3. Show that $G(\mathcal{R})$ is invariant to scaling, orthogonal transformations, and Cartesian products; *i.e.,* $G(\alpha U \mathcal{R}^m) = G(\mathcal{R})$, where $\alpha > 0$ is any scale factor, U is any orthogonal matrix, and $m \geq 1$ is any positive integer. Show that $G(\alpha U[-1,1)^n) = 1/12$ for any version $\alpha U[-1,1)^n$ of any n-cube $[-1,1)^n$ centered at the origin. □

For performance analysis of large lattice constellations, one may use the following approximations, the first two of which are together known as the *continuous approximation*:

- The *size* of the constellation is

$$|\mathcal{C}(\Lambda, \mathcal{R})| \approx \frac{V(\mathcal{R})}{V(\Lambda)}; \tag{14.11}$$

- The *average energy per dimension* of a uniform discrete distribution over $\mathcal{C}(\Lambda, \mathcal{R})$ is

$$P(\mathcal{C}(\Lambda, \mathcal{R})) \approx P(\mathcal{R}); \tag{14.12}$$

- The *average number of nearest neighbors* to any point in $\mathcal{C}(\Lambda, \mathcal{R})$ is $\approx K_{\min}(\Lambda)$.

Again, the union bound estimate (UBE) on probability of block decoding error is

$$\Pr(E) \approx K_{\min}(\Lambda) Q^{\surd}\left(\frac{d^2_{\min}(\Lambda)}{4\sigma^2}\right). \tag{14.13}$$

Since

$$\begin{aligned}
\rho &= \frac{2}{n}\log_2|\mathcal{C}(\Lambda,\mathcal{R})| \approx \frac{2}{n}\log_2\frac{V(\mathcal{R})}{V(\Lambda)};\\
\mathrm{SNR} &= \frac{P(\mathcal{C}(\Lambda,\mathcal{R}))}{\sigma^2} \approx \frac{P(\mathcal{R})}{\sigma^2};\\
\mathrm{SNR}_{\mathrm{norm}} &\approx \frac{\mathrm{SNR}}{2^\rho} = \frac{V(\Lambda)^{2/n}}{V(\mathcal{R})^{2/n}}\frac{P(\mathcal{R})}{\sigma^2},
\end{aligned}$$

we may write the UBE as

$$\Pr(E) \approx K_{\min}(\Lambda) Q^{\surd}\left(\gamma_{\mathrm{c}}(\Lambda)\gamma_s(\mathcal{R})(3\,\mathrm{SNR}_{\mathrm{norm}})\right), \tag{14.14}$$

where the *nominal coding gain* of Λ and the *shaping gain* of $\mathcal{R}$ are defined respectively as

$$\gamma_{\mathrm{c}}(\Lambda) = \frac{d^2_{\min}(\Lambda)}{V(\Lambda)^{2/n}}; \tag{14.15}$$

$$\gamma_s(\mathcal{R}) = \frac{V(\mathcal{R})^{2/n}}{12P(\mathcal{R})} = \frac{1/12}{G(\mathcal{R})}. \tag{14.16}$$

For a baseline M-PAM constellation with $\Lambda = 2\alpha\mathbb{Z}$ and $\mathcal{R} = [-\alpha M, \alpha M]$, we have $\gamma_{\mathrm{c}}(\Lambda) = \gamma_s(\mathcal{R}) = 1$ and $K_{\min}(\Lambda) \approx 2$, so the UBE reduces to the baseline expression

$$\Pr(E) \approx 2Q^{\surd}(3\,\mathrm{SNR}_{\mathrm{norm}}).$$

The nominal coding gain $\gamma_{\mathrm{c}}(\Lambda)$ measures the increase in density of Λ over the baseline integer lattice $\mathbb{Z}$ (or $\mathbb{Z}^n$). The shaping gain $\gamma_s(\mathcal{R})$ measures the decrease in average energy of $\mathcal{R}$ relative to an interval $[-\alpha,\alpha]$ (or an n-cube $[-\alpha,\alpha]^n$). Both contribute a multiplicative factor of gain to the argument of the $Q^{\surd}(\cdot)$ function.

As before, the effective coding gain is reduced by the error coefficient $K_{\min}(\Lambda)$. The probability of block decoding error per two dimensions is

$$P_s(E) \approx K_s(\Lambda) Q^{\surd}(\gamma_{\mathrm{c}}(\Lambda)\gamma_s(\mathcal{R})(3\,\mathrm{SNR}_{\mathrm{norm}})), \tag{14.17}$$

in which the normalized error coefficient per two dimensions is $K_s(\Lambda) = 2K_{\min}(\Lambda)/n$.

Graphically, a curve of the form $P_s(E) \approx K_s(\Lambda) Q^{\surd}(\gamma_{\mathrm{c}}(\Lambda)\gamma_s(\mathcal{R})(3\,\mathrm{SNR}_{\mathrm{norm}}))$ may be obtained simply by moving the baseline curve $P_s(E) = 4Q^{\surd}(3\,\mathrm{SNR}_{\mathrm{norm}})$ to the left by $\gamma_{\mathrm{c}}(\Lambda)$ and $\gamma_s(\mathcal{R})$ (in dB), and upward by a factor of $K_s(\Lambda)/4$. Such simple manipulations of the baseline curve as a function of $\gamma_{\mathrm{c}}(\Lambda)$, $\gamma_s(\mathcal{R})$ and $K_s(\Lambda)$ again are an easy and useful design tool for lattice constellations of moderate complexity.

14.3 Shaping gain and shaping techniques

Although shaping is a newer and less important topic than coding, we discuss it first because its story is quite simple.

The n-dimensional shaping region $\mathcal{R}$ that minimizes $G(\mathcal{R})$ is obviously an n-sphere. The key geometrical parameters of an n-sphere of radius r (for n even) are:

$$\begin{aligned} V_{\otimes}(n,r) &= \frac{(\pi r^2)^{n/2}}{(n/2)!}; \\ P_{\otimes}(n,r) &= \frac{r^2}{n+2}; \\ G_{\otimes}(n,r) &= \frac{P_{\otimes}(n,r)}{V_{\otimes}(n,r)^{2/n}} = \frac{((n/2)!)^{2/n}}{\pi(n+2)}. \end{aligned}$$

By Stirling's approximation, $m! \to (m/e)^m$ as $m \to \infty$, which implies

$$\begin{aligned} G_{\otimes}(n,r) &\to \frac{1}{2\pi e}; \\ \gamma_{s\otimes}(n,r) &= \frac{1/12}{G_{\otimes}(n,r)} \to \frac{\pi e}{6} \ (1.53 \text{ dB}). \end{aligned}$$

Thus shaping gain is limited to a finite value as $n \to \infty$, namely $\pi e/6$ (1.53 dB), which is called the *ultimate shaping gain*.

The shaping gain of an n-sphere is plotted for dimensions $n \leq 24$ in Figure 2. Note that the shaping gain of a 16-sphere already exceeds 1 dB.

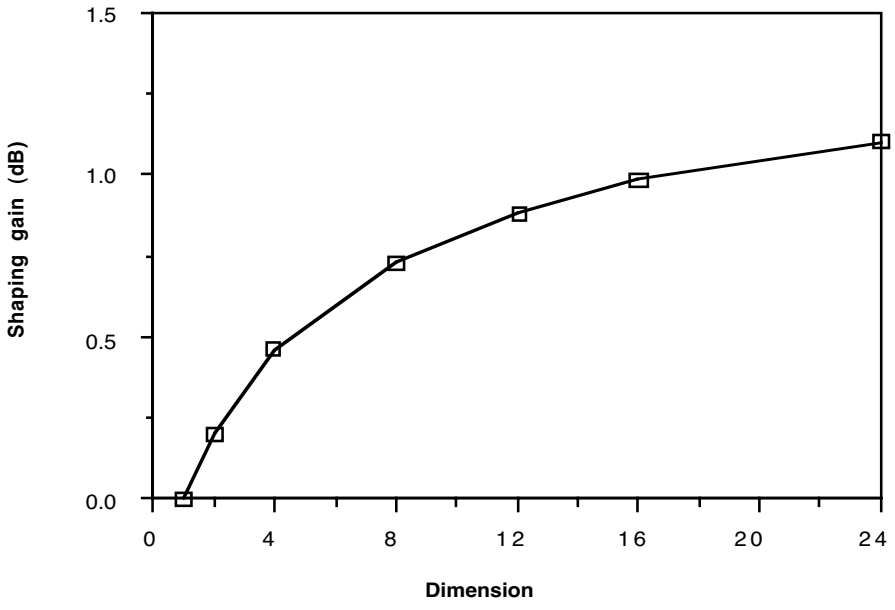


Figure 2. *Shaping gains of n-spheres for $n \leq 24$.*

The projection of a uniform probability distribution over an n-sphere onto one or two dimensions is a nonuniform probability distribution that approaches a Gaussian distribution

as $n \to \infty$. The ultimate shaping gain of $\pi e/6$ (1.53 dB) may alternatively be derived as the difference between the average power of a uniform distribution over an interval and that of a Gaussian distribution with the same differential entropy.

Shaping thus induces a Gaussian-like probability distribution on a one-dimensional PAM or two-dimensional QAM constellation, rather than an equiprobable distribution. In principle, with spherical shaping, the lower-dimensional constellation will become arbitrarily large, even with fixed average power. In practice, the lower-dimensional constellation is constrained by design to a certain region $\mathcal{R}$ to limit "shaping constellation expansion." The n-dimensional shape then only approximates spherical shaping subject to this constraint, and the lower-dimensional probability distribution approaches a truncated Gaussian distribution within the region $\mathcal{R}$.

With large constellations, shaping can be implemented almost independently of coding by operations on the "most significant bits" of M-PAM or $(M \times M)$-QAM constellation labels, which affect the gross shape of the n-dimensional constellation. In contrast, coding affects the "least significant bits" and determines fine structure.

Two practical schemes that can easily obtain shaping gains of 1 dB or more while limiting 2D shaping constellation expansion to a factor of 1.5 or less are "trellis shaping," a kind of dual to trellis coding, and "shell mapping," which uses generating-function techniques to enumerate the points in a Cartesian product constellation in approximate increasing order of energy.

14.4 Coding gains of dense lattices

Finding the densest lattice packings in a given number of dimensions is a mathematical problem of long standing. A summary of the densest known packings is given in [Conway and Sloane, *Sphere Packings, Lattices and Groups*]. The nominal coding gains of these lattices in up to 24 dimensions is plotted in Figure 3.

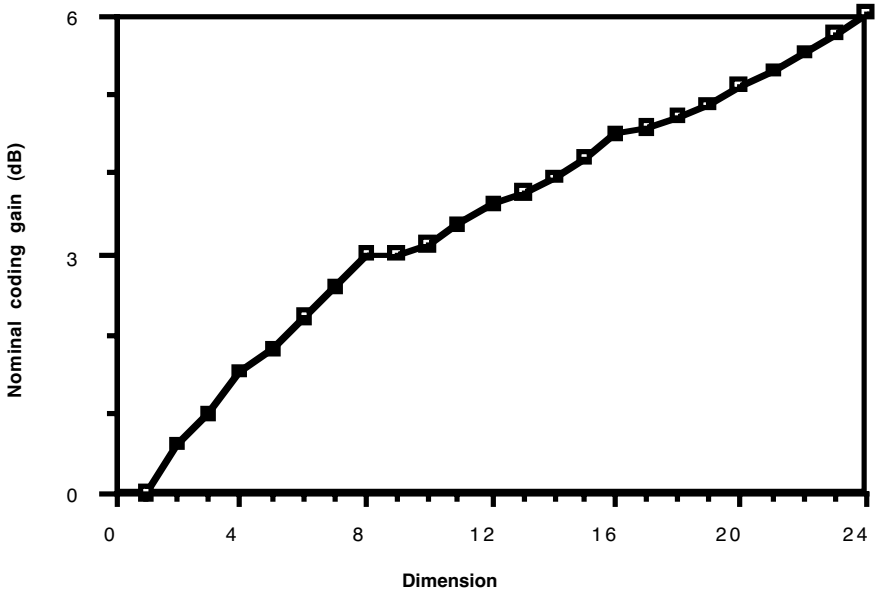


Figure 3. *Nominal coding gains of densest lattices in dimensions $n \leq 24$.*

In contrast to shaping gain, the nominal coding gains of dense n-dimensional lattices become infinite as $n \to \infty$.

Example 5 (Barnes-Wall lattices). For all integer $m \geq 0$, there exists a 2^{m+1}-dimensional *Barnes-Wall lattice* $BW_{2^{m+1}}$ whose nominal coding gain is $2^{m/2}$ (see next subsection). The two-dimensional BW lattice is $\mathbb{Z}^2$. In 4, 8, and 16 dimensions the BW lattices (denoted by D_4, E_8 and Λ_{16}, respectively) are the densest lattices known. For large m, considerably denser lattices are known. □

Exercise 1 (cont.) Show that the mod-2 lattices corresponding to the $(4, 3, 2)$ and $(4, 1, 4)$ binary linear block codes have coding gain $2^{1/2}$ (1.51 dB); these lattices are in fact versions of D_4. Show that the mod-2 lattice corresponding to the $(8, 4, 4)$ binary linear block code has coding gain 2 (3.01 dB); this lattice is in fact a version of E_8. Show that no mod-2 lattice has a nominal coding gain more than 4 (6.02 dB). □

However, effective coding gains cannot become infinite. Indeed, the Shannon limit shows that no lattice can have a combined effective coding gain and shaping gain greater than 9 dB at $P_s(E) \approx 10^{-6}$. This limits the maximum possible effective coding gain to 7.5 dB, since shaping gain can contribute up to 1.53 dB.

What limits effective coding gain is the number of near neighbors, which becomes very large for high-dimensional dense lattices.

Example 5 (cont.) The kissing number of the 2^{m+1}-dimensional Barnes-Wall lattice is

$$K_{\min}(BW_{2^{m+1}}) = \prod_{1 \leq i \leq m+1} (2^i + 2).$$

For $m = 0, 1, 2, 3, 4, \ldots$ these numbers are 4, 24, 240, 4320, 146880, ... Thus while BW_{32} has a nominal coding gain of 4 (6.02 dB), its kissing number is 146880, so its effective coding gain by our rule of thumb is only about 3.8 dB. BW_{128} has a nominal coding gain of 8 (9.03 dB), but a kissing number of 1 260 230 400, so its effective coding gain by our rule of thumb is only about 4.6 dB. These calculations indicate how the effective coding gain of higher-dimensional lattices eventually saturates. □

Example 6 (Leech Lattice). The *Leech lattice* L_{24}, a remarkably dense lattice in 24 dimensions, has a nominal coding gain of 4 (6.02 dB), but it has a kissing number of 196560, so its effective coding gain by our rule of thumb is only about 3.6 dB. □

14.4.1 Barnes-Wall lattices

The Barnes-Wall lattices (1959) are an infinite family of n-dimensional lattices that are analogous to the Reed-Muller binary block codes. For $n \leq 16$, they are the best lattices known. For greater n, they are not in general the best lattices known, but in terms of performance *vs.* decoding complexity they are still quite good, since they admit relatively simple decoding algorithms.

For any integer $m \geq 0$, there exists an $(n = 2^{m+1})$-dimensional BW lattice, denoted $BW_{2^{m+1}}$, that has minimum squared Euclidean distance $d^2_{\min}(BW_{2^{m+1}}) = 2^m$, normalized volume $V(BW_{2^{m+1}})^{2/n} = 2^{m/2}$, and therefore nominal coding gain $\gamma_c(BW_{2^{m+1}}) = 2^{m/2}$.

In 2 dimensions, the Barnes-Wall lattice BW_2 is the integer lattice $\mathbb{Z}^2$, which is the mod-2 lattice corresponding to the $(2, 2, 1)$ code.

The mod-2 lattice $R\mathbb{Z}^2$ corresponding to the $(2,1,2)$ code is a sublattice of $\mathbb{Z}^2$; it is the set of all integer 2-tuples in which both integers are even or both integers are odd. It can be obtained by rotating $\mathbb{Z}^2$ by 45° and scaling by $\sqrt{2}$; *i.e.,* by transforming $\mathbb{Z}^2$ by the 2×2 Hadamard matrix

$$R = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}.$$

Consequently $d^2_{\min}(R\mathbb{Z}^2) = 2$ and $V(R\mathbb{Z}^2) = 2$.

The lattice $2\mathbb{Z}^2$ (the mod-2 lattice corresponding to the $(2,0,\infty)$ code) is a sublattice of $R\mathbb{Z}^2$ with $d^2_{\min}(2\mathbb{Z}^2) = 4$ and $V(2\mathbb{Z}^2) = 4$. Note that $2\mathbb{Z}^2 = R(R\mathbb{Z}^2)$, since $R^2 = 2I$.

In fact, we see that there is a lattice chain $\mathbb{Z}^2/R\mathbb{Z}^2/2\mathbb{Z}^2/2R\mathbb{Z}^2/4\mathbb{Z}^2/\ldots$ with minimum squared distances $1/2/4/8/16/\ldots$.

The remaining BW lattices may be constructed recursively from this chain by the $|u|u+v|$ construction. $BW_{2^{m+1}}$ is constructed from BW_{2^m} and RBW_{2^m} as

$$BW_{2^{m+1}} = \{(\mathbf{u}, \mathbf{u}+\mathbf{v}) \mid \mathbf{u} \in BW_{2^m}, \mathbf{v} \in RBW_{2^m}\}.$$

More generally, for any $j \geq 0$, $R^j BW_{2^{m+1}} = \{(\mathbf{u}, \mathbf{u}+\mathbf{v}) \mid \mathbf{u} \in R^j BW_{2^m}, \mathbf{v} \in R^{j+1} BW_{2^m}\}$.

It is then easy to prove the following facts by recursion:

(a) The dimension of $BW_{2^{m+1}}$ is $n = 2^{m+1}$.

(b) The volume of $BW_{2^{m+1}}$ is

$$V(BW_{2^{m+1}}) = V(BW_{2^m})V(R(BW_{2^m})) = 2^{2^{m-1}} V(BW_{2^m})^2.$$

This recursion yields $V(BW_{2^{m+1}}) = 2^{m2^{m-1}}$, or $V(BW_{2^{m+1}})^{2/n} = 2^{m/2}$.

(c) The minimum squared distance of $BW_{2^{m+1}}$ is $d^2_{\min}(BW_{2^{m+1}}) = 2^m$.

(d) $\{R^j BW_{2^{m+1}}, j \geq 1\}$ is a chain of sublattices with minimum squared distances and normalized volumes increasing by a factor of 2 for each increment of j.

We verify that these assertions hold for $BW_2 = \mathbb{Z}^2$. For $m \geq 1$, the dimension and volume follow from the construction. We verify the distance as follows:

(a) if $\mathbf{u} = \mathbf{0}$, then $||(\mathbf{0}, \mathbf{v})||^2 = ||\mathbf{v}||^2 \geq 2^m$ if $\mathbf{v} \neq \mathbf{0}$, since $\mathbf{v} \in RBW_{2^m}$.

(b) if $\mathbf{u} + \mathbf{v} = \mathbf{0}$, then $\mathbf{u} = -\mathbf{v} \in RBW_{2^m}$ and $||(-\mathbf{v}, \mathbf{0})||^2 \geq 2^m$ if $\mathbf{v} \neq \mathbf{0}$.

(c) if $\mathbf{u} \neq \mathbf{0}$ and $\mathbf{u} + \mathbf{v} \neq \mathbf{0}$, then both $\mathbf{u}$ and $\mathbf{u} + \mathbf{v}$ are in BW_{2^m} (since RBW_{2^m} is a sublattice of BW_{2^m}), so

$$||(\mathbf{u}, \mathbf{u}+\mathbf{v})||^2 = ||\mathbf{u}||^2 + ||\mathbf{u}+\mathbf{v}||^2 \geq 2 \cdot 2^{m-1} = 2^m.$$

Equality clearly holds for $(\mathbf{0}, \mathbf{v})$, $(\mathbf{v}, \mathbf{0})$ or $(\mathbf{u}, \mathbf{u})$ if we choose $\mathbf{v}$ or $\mathbf{u}$ as a minimum-weight vector from their respective lattices.

Finally, the sublattice chain for m follows from the sublattice chain for $m-1$ by construction.

The $|u|u+v|$ construction suggests the following tableau of BW lattices. Here $D_4 = BW_4$, $E_8 = BW_8$, and $\Lambda_n = BW_n$ for $n = 2^{m+1} \geq 16$. Also, we use $R^2 = 2I_{2^m}$.

$$
\begin{array}{ccccccccc}
\mathbb{Z}^2 & & & & & & & & \\
& D_4 & & & & & & & \\
R\mathbb{Z}^2 & & E_8 & & & & & & \\
& RD_4 & & \Lambda_{16} & & & & & \\
2\mathbb{Z}^2 & & RE_8 & & \Lambda_{32} & & & & \\
& 2D_4 & & R\Lambda_{16} & & \Lambda_{64} & & & \\
2R\mathbb{Z}^2 & & 2E_8 & & R\Lambda_{32} & & \Lambda_{128} & & \\
& 2RD_4 & & 2\Lambda_{16} & & R\Lambda_{64} & & \Lambda_{256} & \\
4\mathbb{Z}^2 & & 2RE_8 & & 2\Lambda_{32} & & R\Lambda_{128} & & \Lambda_{512}
\end{array}
$$

Figure 4. *Tableau of Barnes-Wall lattices.*

In this tableau each BW lattice lies halfway between the two lattices of half the dimension that are used to construct it in the $|u|u+v|$ construction, from which we can immediately deduce its normalized volume.

For example, E_8 has the same normalized volume as $R\mathbb{Z}^2$, namely $V(E_8)^{2/8} = 2$. However, $d^2_{\min}(E_8) = 4$, whereas $d^2_{\min}(R\mathbb{Z}^2) = 2$. Therefore the nominal coding gain of E_8 is twice that of $R\mathbb{Z}^2$, namely $\gamma_c(E_8) = 2$ (3.01 dB).

14.5 Trellis codes

Trellis codes are dense packings of Euclidean-space sequences in a sequence space which is in principle infinite-dimensional. Trellis codes are to lattices as convolutional codes are to block codes. We will see that, just as binary convolutional codes provide a better performance/complexity tradeoff than binary block codes in the power-limited regime, trellis codes provide a better performance/complexity tradeoff than lattices in the bandwidth-limited regime, although the difference is not as dramatic.

The key ideas in the invention of trellis codes were:

- use of minimum squared Euclidean distance as the design criterion;
- coding on subsets of signal sets using convolutional coding principles (*e.g.,* trellises and the Viterbi algorithm).

A typical large-constellation trellis code is designed as follows. One starts with a large low-dimensional constellation, which in practice is almost always a lattice constellation $\mathcal{C}(\mathbb{Z}^n, \mathcal{R})$ based on a version of an n-dimensional integer lattice $\mathbb{Z}^n$, such as M-PAM or $(M \times M)$-QAM. (M-PSK constellations are sometimes used in the intermediate ($\rho \approx 2$ b/2D) regime because of their constant-energy property, but we will not discuss M-PSK trellis codes here.)

One can then form an m-fold Cartesian product constellation

$$\mathcal{C}(\mathbb{Z}^n, \mathcal{R})^m = \mathcal{C}(\mathbb{Z}^{mn}, \mathcal{R}^m),$$

which is still based on an mn-dimensional integer lattice $\mathbb{Z}^{mn}$.

The constellation $\mathcal{C}(\mathbb{Z}^{mn}, \mathcal{R}^m)$ is partitioned into subsets of equal size, where the number of subsets is typically a power of two, say 2^b. Initially this was done by a sequence of two-way partitions in which the minimum squared distance within subsets was maximized at each level. Subsequently it was recognized that the resulting constellations were almost always lattice constellations $\mathcal{C}(\Lambda', \mathcal{R}^m)$ based on a sublattice Λ' of index $|\mathbb{Z}^{mn}/\Lambda'| = 2^b$ in $\mathbb{Z}^{mn}$. In other words, $\mathbb{Z}^{mn}$ is the union of 2^b cosets of Λ', and the 2^b subsets are the points of $\mathcal{C}(\mathbb{Z}^{mn}, \mathcal{R}^m)$ that lie in each such coset. The sublattice Λ' is usually chosen to be as dense as possible.

Example 7 (1D partitions). In one dimension, there is a chain of sublattices of $\mathbb{Z}$ as follows:

$$\mathbb{Z} \supseteq 2\mathbb{Z} \supseteq 4\mathbb{Z} \supseteq 8\mathbb{Z} \supseteq \cdots,$$

which may alternatively be written as $\mathbb{Z}/2\mathbb{Z}/4\mathbb{Z}/8\mathbb{Z}/\cdots$. Each partition is two-way; that is, each lattice is the union of two cosets of the next sublattice. The corresponding minimum squared distances are $1/4/16/64/\cdots$. Thus an M-PAM constellation $\mathcal{C}(\mathbb{Z}, [-M/2, M/2])$ with minimum squared distance 1 may be partitioned into 2 subsets of the form $\mathcal{C}(2\mathbb{Z}, [-M/2, M/2])$ with minimum squared distance 4 within subsets, or 4 subsets of the form $\mathcal{C}(4\mathbb{Z}, [-M/2, M/2])$ with minimum squared distance 16 within subsets, and so forth. □

Example 8 (2D partitions). In two dimensions, there is a chain of sublattices of $\mathbb{Z}^2$ as follows:

$$\mathbb{Z}^2 \supseteq R\mathbb{Z}^2 \supseteq 2\mathbb{Z}^2 \supseteq 2R\mathbb{Z}^2 \supseteq \cdots,$$

where R is the 2×2 Hadamard matrix as above. This chain may alternatively be written as $\mathbb{Z}^2/R\mathbb{Z}^2/2\mathbb{Z}^2/2R\mathbb{Z}^2/\cdots$. Each partition is two-way. The corresponding minimum squared distances are $1/2/4/8/\cdots$. Thus a QAM constellation $\mathcal{C}(\mathbb{Z}^2, \mathcal{R})$ with minimum squared distance 1 may be partitioned into 2 subsets of the form $\mathcal{C}(R\mathbb{Z}^2, \mathcal{R})$ with minimum squared distance 2 within subsets, or 4 subsets of the form $\mathcal{C}(2\mathbb{Z}^2, \mathcal{R})$ with minimum squared distance 4 within subsets, and so forth. The bounding region $\mathcal{R}$ should contain an equal number of points in each subset. □

Example 9 (4D partitions). In four dimensions, there is a chain of sublattices of $\mathbb{Z}^4$ as follows:

$$\mathbb{Z}^4 \supseteq D_4 \supseteq R\mathbb{Z}^4 \supseteq RD_4 \supseteq \cdots,$$

where D_4 is the 4-dimensional Barnes-Wall lattice and R is the 4×4 matrix

$$R = \begin{bmatrix} 1 & 1 & 0 & 0 \\ 1 & -1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & -1 \end{bmatrix}.$$

(Alternatively, this is the chain of mod-2 lattices corresponding to the $(4,4,1)$, $(4,3,2)$, $(4,2,2)$ and $(4,1,4)$ binary linear block codes.) This chain may alternatively be written as $\mathbb{Z}^4/D_4/R\mathbb{Z}^4/RD_4/\cdots$. Each partition is two-way. The corresponding minimum squared distances are $1/2/2/4/\cdots$. Thus a 4D constellation $\mathcal{C}(\mathbb{Z}^4, \mathcal{R})$ with minimum squared distance 1 may be partitioned into 2 subsets of the form $\mathcal{C}(D_4, \mathcal{R})$ with minimum squared distance 2 within subsets, 8 subsets of the form $\mathcal{C}(2D_4, \mathcal{R})$ with minimum squared distance 4 within subsets, etc. Again, the bounding region $\mathcal{R}$ should contain an equal number of points in each subset. □

A trellis code encoder then operates as shown in Figure 5. Some of the input data bits are encoded in a rate-k/b 2^ν-state binary convolutional encoder. Almost always k is chosen to equal $b-1$, so the code redundancy is 1 bit per mn dimensions. The encoder output sequence of b-tuples selects a corresponding sequence of subsets of $\mathcal{C}(\mathbb{Z}^{mn}, \mathcal{R}^m)$ (cosets of Λ'). The convolutional code and the labeling of the subsets are chosen primarily to maximize the minimum squared distance $d^2_{\min}(\mathcal{C})$ between signal point sequences in any possible encoded subset sequence, and secondarily to minimize the maximum possible number $K_{\min}(\mathcal{C})$ of nearest-neighbor sequences. Finally, other input data bits select the actual signal points to be transmitted from the selected subsets. If there is any shaping, it is done at this level.

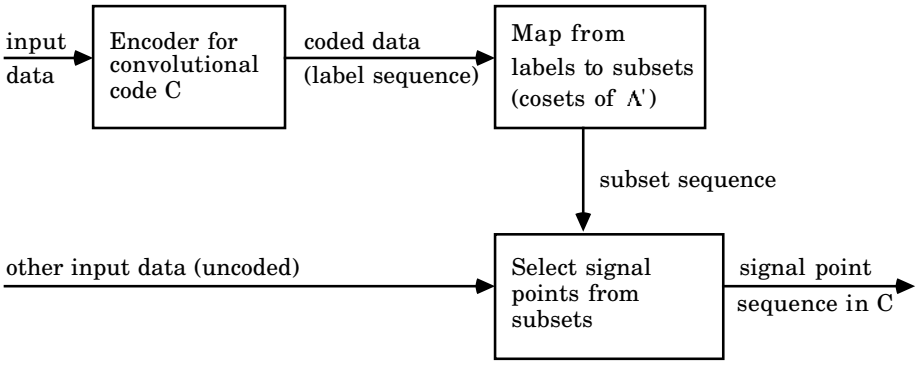


Figure 5. *Trellis code encoder.*

The nominal coding gain of such a trellis code is

$$\gamma_{\rm c}(\mathcal{C}) = d^2_{\min}(\mathcal{C}) 2^{-\eta(\mathcal{C})}, \tag{14.18}$$

where $\eta(\mathcal{C}) = 2/mn$ is the redundancy of the convolutional code in bits per two dimensions. The factor $2^{\eta(\mathcal{C})}$ may be thought of as the normalized volume of the trellis code per two dimensions, if the signal constellation is a lattice constellation based on an integer lattice $\mathbb{Z}^{mn}$. The effective coding gain is reduced by the amount that the error coefficient $2K_{\min}(\mathcal{C})/mn$ per two dimensions exceeds the baseline M-PAM error coefficient of 4 per two dimensions, again according to the rule of thumb that a factor of 2 increase costs 0.2 dB.

Exercise 1 (cont.) Let $\mathcal{C}$ be a rate-k/n binary linear convolutional code with free distance d and N_d minimum-weight code sequences per n dimensions. Define the corresponding mod-2 trellis code $\Lambda_{\mathcal{C}}$ to be the set of all integer sequences $\mathbf{x}$ with D-transform $x(D)$ such that $x(D) \equiv c(D) \bmod 2$ for some code sequence $c(D)$ in $\mathcal{C}$.

(a) Show that an encoder as in Figure 5 based on the convolutional code $\mathcal{C}$ and the lattice partition $\mathbb{Z}^n/2\mathbb{Z}^n$ is an encoder for this mod-2 trellis code.

(b) Show that $\Lambda_{\mathcal{C}}$ has the group property.

(c) Show that $\Lambda_{\mathcal{C}}$ has the following parameters:

$$d^2_{\min}(\Lambda_{\mathcal{C}}) = \min\{d, 4\}; \tag{14.19}$$

$$K_{\min}(\Lambda_{\mathcal{C}}) = \begin{cases} 2^d N_d, & \text{if } d < 4; \\ 2n, & \text{if } d > 4; \\ 2^d N_d + 2n, & \text{if } d = 4; \end{cases} \tag{14.20}$$

$$\gamma_{\rm c}(\Lambda_{\mathcal{C}}) = d^2_{\min}(\Lambda_{\mathcal{C}}) 2^{-\eta(\mathcal{C})}, \tag{14.21}$$

where $\eta(\mathcal{C}) = 2(n-k)/n$ is the redundancy of $\mathcal{C}$ in bits per two dimensions. □

The encoder redundancy $\eta(\mathcal{C})$ also leads to a "coding constellation expansion ratio" which is a factor of $2^{\eta(\mathcal{C})}$ per two dimensions—*i.e.,* a factor of $4, 2, \sqrt{2}, \ldots$ for 1D, 2D, 4D,... codes, respectively. Minimization of coding constellation expansion has motivated the increasing use of higher-dimensional trellis codes.

A trellis code may be decoded by a Viterbi algorithm (VA) decoder, as follows. Given a received point $\mathbf{r}$ in $\mathbb{R}^{mn}$, the received first finds the closest signal point to $\mathbf{r}$ in each subset. A VA decoder then finds the closest code sequence to the entire received sequence. The decoding complexity is usually dominated by the complexity of the VA decoder, which to first order is dominated by the branch complexity $2^{\nu+k}$ of the convolutional code, normalized by the dimension mn.

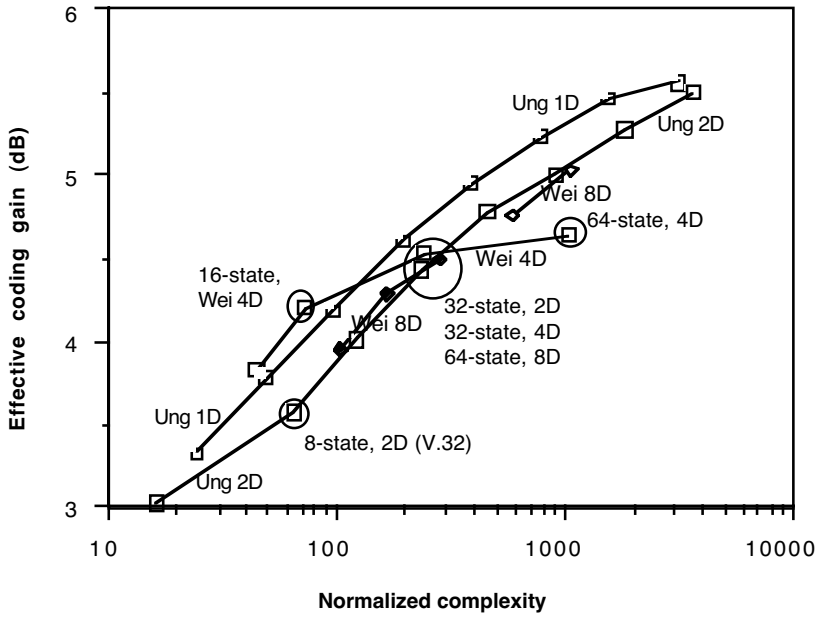


Figure 6. *Effective coding gain vs. complexity for Ungerboeck and Wei codes.*

Figure 6 shows the effective coding gains of certain important families of trellis codes versus their decoding complexity, measured by a detailed operation count. The codes considered are:

(a) The original 1D (PAM) trellis codes of Ungerboeck (1982), which are based on rate-1/2 convolutional codes ($\eta(\mathcal{C}) = 2$) with $2 \le \nu \le 9$ and the 4-way partition $\mathbb{Z}/4\mathbb{Z}$.

(b) The 2D (QAM) trellis codes of Ungerboeck, which (apart from the simplest 4-state code) are based on rate-2/3 convolutional codes ($\eta(\mathcal{C}) = 1$) with $3 \le \nu \le 9$ and the 8-way partition $\mathbb{Z}^2/2R\mathbb{Z}^2$.

(c) The 4D trellis codes of Wei (1987), all with $\eta(\mathcal{C}) = 1/2$, based on

  (a) rate-2/3 8- and 16-state convolutional codes and the 8-way partition $\mathbb{Z}^4/RD_4$;
  (b) a rate-3/4 32-state convolutional code and the 16-way partition $\mathbb{Z}^4/2\mathbb{Z}^4$;
  (c) a rate-4/5 64-state convolutional code and the 32-way partition $\mathbb{Z}^4/2D_4$.

(d) Two families of 8D trellis codes of Wei ($\eta(\mathcal{C}) = 1/4$).

The V.32 modem (1984) uses an 8-state 2D trellis code, also due to Wei (1984), whose performance/complexity tradeoff is the same as that as that of the original 8-state 2D Ungerboeck code, but which uses a nonlinear convolutional encoder to achieve 90° rotational invariance. This code has an effective coding gain of about 3.6 dB, a branch complexity of 2^5 (per two dimensions), and a coding constellation expansion ratio of 2.

The V.34 modem (1994) specifies three 4D trellis codes, with performance and complexity equivalent to the 4D Wei codes circled on Figure 6. All have a coding constellation expansion ratio of $\sqrt{2}$. The 16-state code is the original 16-state 4D Wei code, which has an effective coding gain of about 4.2 dB and a branch complexity of 2^6 (per four dimensions). The 32-state code is due to Williams and is based on the 16-way partition $\mathbb{Z}^4/H\mathbb{Z}^4$, where H is a 4×4 Hadamard matrix, to ensure that there are no minimum-distance error events whose length is only two dimensions; it has an effective coding gain of about 4.5 dB and a branch complexity of 2^8 (per four dimensions) The 64-state code is a modification of the original 4D Wei code, modified to prevent quasicatastrophic error propagation; it has an effective coding gain of about 4.7 dB and a branch complexity of 2^{10} (per four dimensions).

It is noteworthy that no one has improved on the performance vs. complexity tradeoff of the original 1D and 2D trellis codes of Ungerboeck or the subsequent multidimensional codes of Wei, and by this time it seems safe to predict that no one will ever do so. There have however been new trellis codes that enjoy other properties with about the same performance and complexity, such as those described in the previous two paragraphs, and there may still be room for further improvements of this kind.

Finally, we see that trellis codes have a performance/complexity advantage over lattice codes, when used with maximum-likelihood decoding. Effective coding gains of 4.2–4.7 dB, better than that of the Leech lattice L_{24} or of BW_{32}, are attainable with less complexity (and much less constellation expansion). 512-state 1D or 2D trellis codes can achieve effective coding gains of the order of 5.5 dB, which is superior to that of lattice codes of far greater complexity.

On the other hand, it seems very difficult to obtain effective coding gains of greater than 6 dB. This is not surprising, because at $P_s(E) \approx 10^{-6}$ the effective coding gain at the Shannon limit would be about 7.5 dB, and at the cutoff rate limit it would be about 5.8 dB. To approach the Shannon limit, much more complicated codes and decoding methods are necessary.

14.6 Sequential decoding in the high-SNR regime

In the bandwidth-limited regime, the cutoff rate limit is a factor of $4/e$ (1.68 dB) less than capacity. Therefore sequential decoders should be able to operate within about 1.7 dB of the Shannon limit; *i.e.,* sequential decoders should be able to achieve an effective coding gain of about 6 dB at $P_s(E) \approx 10^{-6}$. Several theses (Wang, Ljungberg, Maurer) have confirmed that sequential decoders are indeed capable of such performance.

14.7 Multilevel codes and multistage decoding

To approach the Shannon limit even more closely, it is clear that much more powerful codes must be used, with non-ML but near-ML decoding. Multilevel codes and multistage decoding may be used for this purpose. Multilevel coding may be based on a chain of sublattices of $\mathbb{Z}^n$,

$$\Lambda_0 = \mathbb{Z}^n \supseteq \Lambda_1 \supseteq \cdots \supseteq \Lambda_{r-1} \supseteq \Lambda_r,$$

which induce a chain of lattice partitions Λ_{j-1}/Λ_j, $1 \le j \le r$. A different encoder as in Figure 5 may be used independently on each such lattice partition. Moreover, with multistage decoding, each level is decoded independently.

Remarkably, such a multilevel scheme incurs no loss in channel capacity, compared to a single-level code based on the partition $\mathbb{Z}^n/\Lambda_r$; the capacity $C(\mathbb{Z}^n/\Lambda_r)$ of the partition $\mathbb{Z}^n/\Lambda_r$ is equal to the sum of the capacities $C(\Lambda_{j-1}/\Lambda_j)$ at each level. If the partition $\mathbb{Z}^n/\Lambda_r$ is "large enough" and appropriately scaled, then $C(\mathbb{Z}^n/\Lambda_r)$ approaches the capacity of the Gaussian channel.

All of the partitions Λ_{j-1}/Λ_j may even be binary; *e.g.,* one may use the standard one-dimensional or two-dimensional chains

$$\mathbb{Z} \supseteq 2\mathbb{Z} \supseteq 4\mathbb{Z} \supseteq 8\mathbb{Z} \supseteq \cdots;$$
$$\mathbb{Z}^2 \supseteq R\mathbb{Z}^2 \supseteq 2\mathbb{Z}^2 \supseteq 2R\mathbb{Z}^2 \supseteq 4\mathbb{Z}^2 \supseteq \cdots.$$

Then one can use a binary code of rate close to $C(\Lambda_{j-1}/\Lambda_j)$ at each level to approach the Shannon limit.

In particular, by using binary turbo codes of appropriate rate at each level, it has been shown that one can get within 1 dB of the Shannon limit (Wachsmann and Huber).

Powerful probabilistic coding methods such as turbo codes are really needed only at the higher levels. At the lower levels, the channels become quite clean and the capacity $C(\Lambda_{j-1}/\Lambda_j)$ approaches $\log_2 |\Lambda_{j-1}/\Lambda_j|$, so that the desired redundancy approaches zero. For these levels, algebraic codes and decoding methods may be more appropriate.

In summary, multilevel codes and multistage decoding allow the Shannon limit to be approached as closely in the bandwidth-limited regime as it can be approached in the power-limited regime with binary codes.

14.8 Multilevel turbo codes

A number of varieties of multilevel turbo codes based on multiple component trellis codes have been developed for the bandwidth-limited regime by several authors (*e.g.,* Berrou *et al.*, Benedetto *et al.*, Robertson and Wörz, Divsalar *et al.*). The performance of these codes seems to be comparable to that of binary turbo codes in the power-limited regime: *i.e.,* within about 1 dB of the Shannon limit. However, such capacity-approaching codes do not seem to have been implemented yet in practice, to the best of our knowledge.

14.9 Bit-interleaved coded modulation

In bit-interleaved coded modulation (BICM), the signals in a nonbinary constellation of size 2^b are selected by b randomly interleaved encoded bits from a binary encoder. The effective binary channel is then an equiprobable mixture of b parallel channels. The receiver knows which channel is used for each bit, and therefore can compute the correct APP vector for each symbol. Capacity-approaching codes may be designed for this mixture channel. While capacity is typically slightly reduced on an AWGN channel, the "pragmatic" BICM approach has become quite popular.

References

[1] C. Berrou, A. Glavieux and P. Thitimajshima, "Near Shannon limit error-correcting coding and decoding: Turbo codes," *Proc. 1993 Int. Conf. Commun.* (Geneva), pp. 1064–1070, May 1993.

[2] S.-Y. Chung, G. D. Forney, Jr., T. J. Richardson and R. Urbanke, "On the design of low-density parity-check codes within 0.0045 dB from the Shannon limit," *IEEE Commun. Letters*, vol. 5, pp. 58–60, Feb. 2001.

[3] D. J. Costello, Jr., J. Hagenauer, H. Imai and S. B. Wicker, "Applications of error-control coding," *IEEE Trans. Inform. Theory*, vol. 44, pp. 2531–2560, Oct. 1998.

[4] R. G. Gallager, *Low-Density Parity-Check Codes.* Cambridge, MA: MIT Press, 1962.

[5] E. A. Lee and D. G. Messerschmitt, *Digital Communication* (first edition). Boston: Kluwer, 1988.

[6] E. A. Lee and D. G. Messerschmitt, *Digital Communication* (second edition). Boston: Kluwer, 1994.

[7] C. E. Shannon, "A mathematical theory of communication," *Bell Syst. Tech. J.*, vol. 27, pp. 379–423 and 623–656, 1948.

6.451 Principles of Digital Communication II
Spring 2005