Chapter 9 Channel Capacity and Coding

Text. [1] J. G. Proakis and M. Salehi, Communication Systems Engineering, 2/e. Prentice Hall, 2002.

The goal of any communication system is to transmit information from an **information source** to a **destination** via a **communication channel**.

9.1 Modeling of Communication Channels

A communication channel is a medium over which information can be transmitted or in which information can be stored, such as coaxial cables, ionospheric propagation, free space, fiber optic cables, and magnetic and optical disks.

They accept a signal at their inputs and deliver a signal at their outputs at another location (in case of transmission) or at a later time (in case of storage).

The output of a communication channel becomes different from its input due to various factors such as attenuation, nonlinearities, bandwidth limitations, multipath propagation, and noise.

Due to the presence of noise and fading, the input-output relation in a communication channel is, generally, a stochastic relation.

A waveform channels accepts a (continuous-time) waveform signal as its input and produce a waveform signal as its output.

Because the bandwidth of any practical channel is limited, a waveform channel becomes equivalent to a discrete-time channel by using the sampling theorem.

In a discrete-time channel, both input and output are discrete-time signals.

In a discrete-time channel, the channel is called a **discrete channel** if the values that the input and output variables can take are finite or countably infinite.

In general, a discrete channel is defined by the input alphabet $\mathcal{X}$, the output alphabet $\mathcal{Y}$, and the conditional PMF $p(\mathbf{y} \mid \mathbf{x})$ of the output sequence given the input sequence.

A schematic representation of a discrete channel is shown in Figure 9.1.

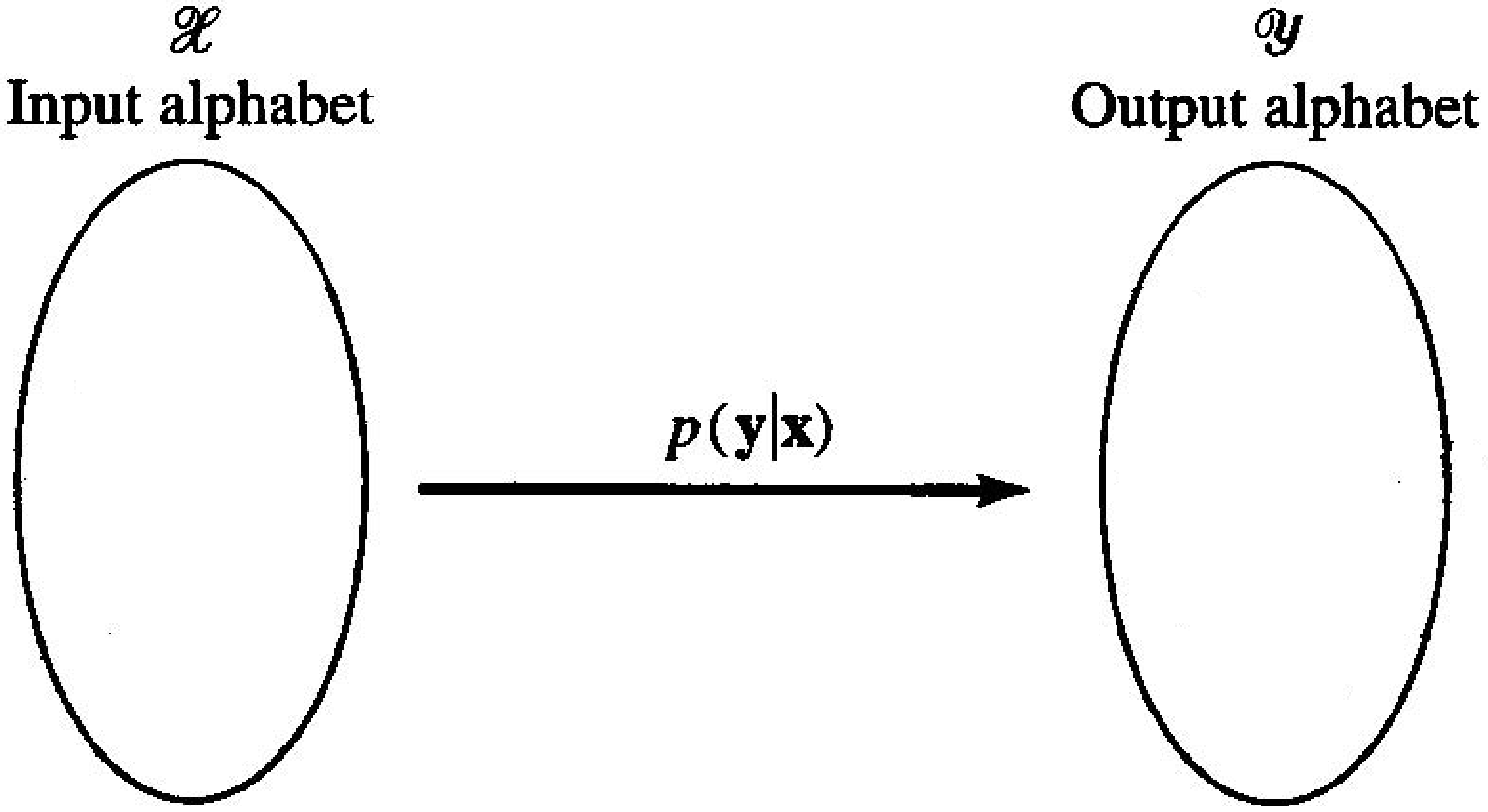


Figure 9.1 A discrete channel.

In general, the output y_i does not only depend on the input at the same time x_i but also on the previous inputs (channels with ISI, see Chapter 8), or even previous and future inputs (in storage channels).

In this case, we say that the channel has **memory**.

For a **discrete-memoryless channel**, for any $\mathbf{y} \in \mathcal{Y}^n$ and $\mathbf{x} \in \mathcal{X}^n$, we have

$$p(\mathbf{y} \mid \mathbf{x}) = \prod_{i=1}^{n} p(y_i \mid x_i) \tag{9.1.1}$$

where $p(y_i \mid x_i)$ is transition probability.

All channel models that we will discuss in this chapter are memoryless.

A special case of a discrete-memoryless channel is the **binary-symmetric channel (BSC)**.

Figure 9.2 shows a binary-symmetric channel.

In a binary-symmetric channel, $P(0 \mid 1) = P(1 \mid 0) = \epsilon$ is called the **crossover probability**.

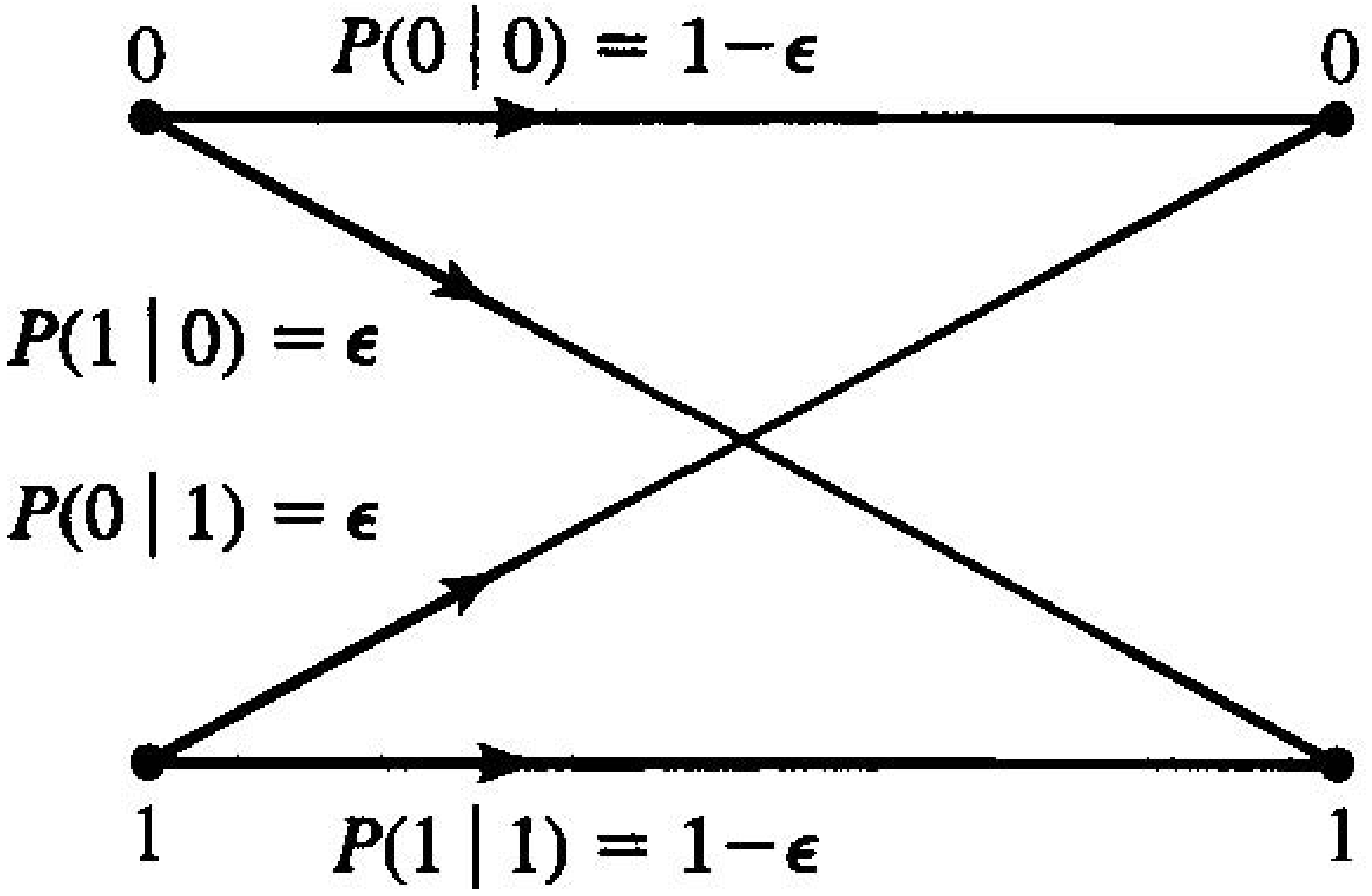


Figure 9.2 Binary-symmetric channel (BSC).

Ex. 9.1.1

Consider an additive white Gaussian noise channel with binary antipodal signaling.

The error probability of a 1 being detected as 0 or a 0 being detected as 1 is given by

$$\begin{aligned}\epsilon &= P(1|0)\\ &= P(0|1)\\ &= Q\left(\sqrt{\frac{2\varepsilon_b}{N_0}}\right) \qquad (9.1.2)\end{aligned}$$

where N_0 is the (single-sided) noise power spectral density and ε_b is the bit energy of each of the antipodal signals representing 0 and 1.

This discrete channel is a binary symmetric channel.

Ex. 9.1.2

In an AWGN channel with binary antipodal signaling, the input is either $\sqrt{\varepsilon_b}$ or $-\sqrt{\varepsilon_b}$.

The output is the sum of the input and the Gaussian noise.

For this binary-input, continuous-output channel, we have $\mathcal{X} = \{\pm\sqrt{\varepsilon_b}\}$ and $\mathcal{Y} = \mathbb{R}$ and

$$f(y \mid x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(y-x)^2}{2\sigma^2}}$$

where σ^2 is the variance of the noise.

Ex.

As an example of a continuous-amplitude channel, consider the discrete-time additive white Gaussian noise channel with an input power constraint.

In this channel, both the input alphabet $\mathcal{X}$ and output alphabet $\mathcal{Y}$ are the set of real numbers.

If the channel input is X, the channel output is given by

$$Y = +Z \tag{9.1.3}$$

where Z is the Gaussian noise in the channel with mean 0 and variance P_N.

Assume that inputs to this channel satisfy power constraint such that,

for large n, an input block of length n satisfies

$$\frac{1}{n}\sum_{i=1}^{n} x_i^2 \le P \tag{9.1.4}$$

where P is some fixed power constraint.

This channel model is shown in Figure 9.3.

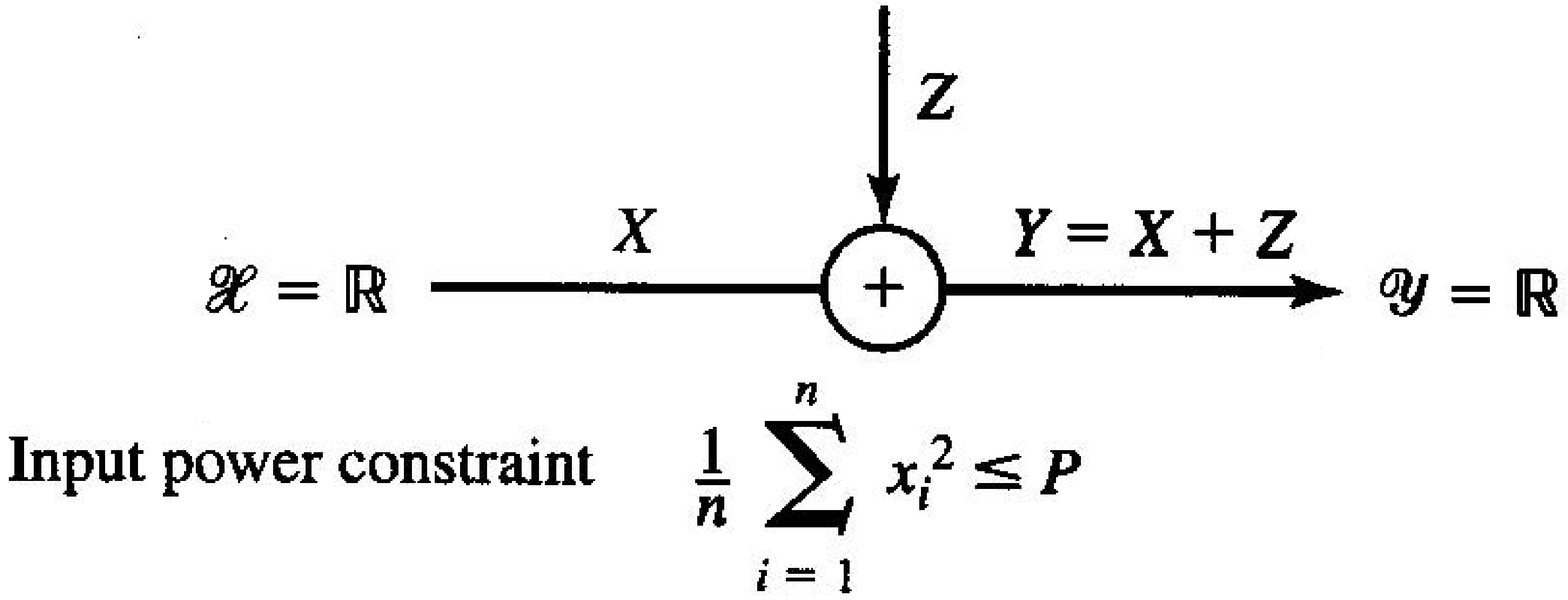

Z
X
$Y = X + Z$
$\mathscr{X} = \mathbb{R}$
$+$
$\mathscr{Y} = \mathbb{R}$
Input power constraint
$\frac{1}{n}\sum_{i=1}^{n} x_i^2 \le P$

Figure 9.3 Additive white Gaussian noise channel power constraint.

9.2 Channel Capacity

The **reliability** of transmitting information over any communication channel is measured by the probability of correct reception at the receiver.

By the **noisy channel-coding theorem** by Shannon (1948), reliable transmission (that is transmission with error probability less any given value) is possible even over a noisy channel as long as the transmission rate is less than some number called the **channel capacity**.

This implies that **the basic limitation that noise causes in a communication channel is not on the reliability of communication, but on the speed of communication**.

Figure 9.4 shows an example of a discrete-memoryless channel with four inputs and outputs.

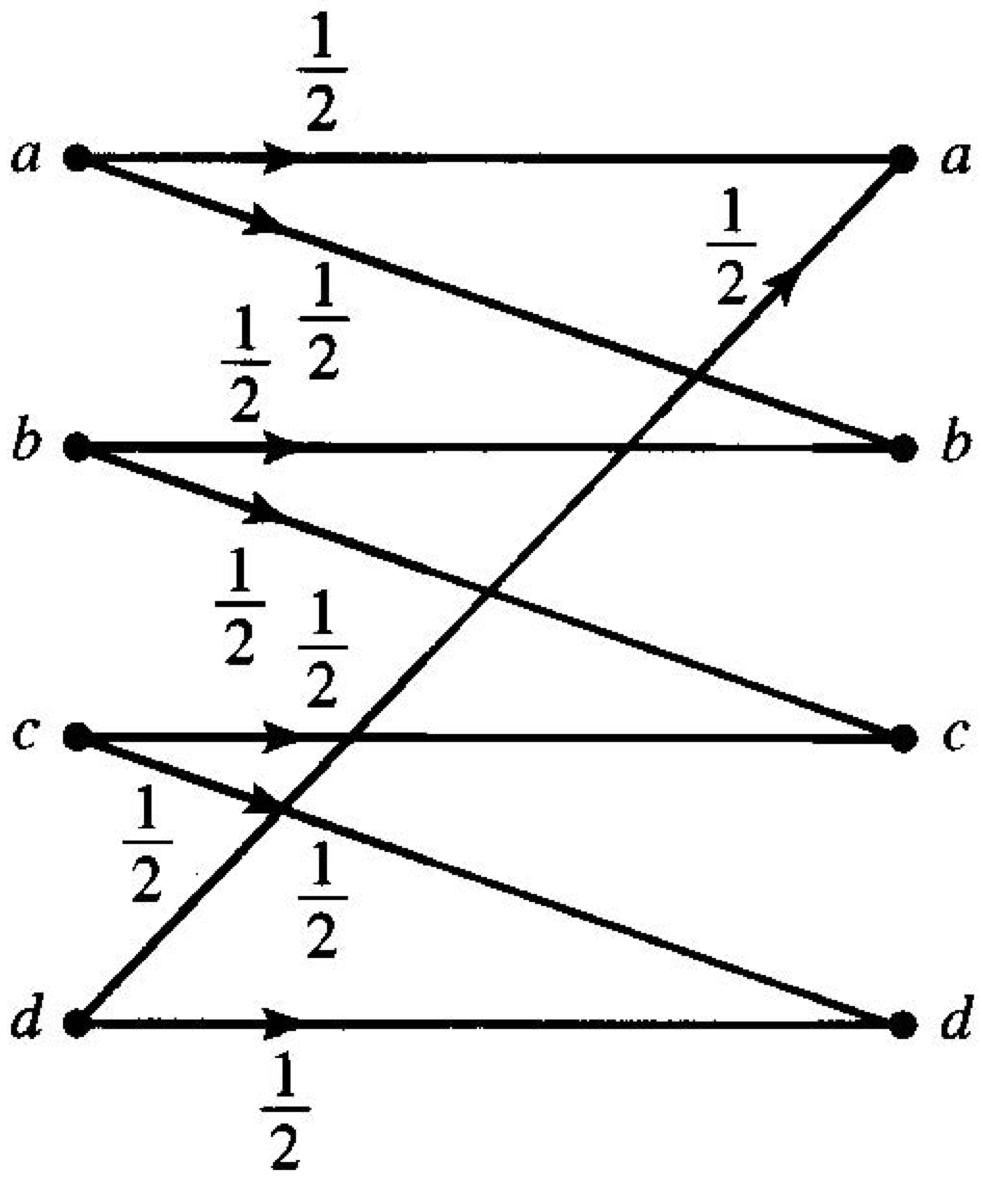


Figure 9.4 Example of a discrete channel.

Since the receiver does not know whether a or d was transmitted if it receives a;
the receiver does not know whether a or b was transmitted if it receives b, etc.;
there always exists a possibility of error.

But if the transmitter and receiver agree that the transmitter only uses letters a and c, then there exists no ambiguity.

In this case, if the receiver receives a or b it knows that a was transmitted; and if it receives c or d it knows that c was transmitted.

This means that the two symbols a and c can be transmitted over this channel without error; that is, errors can be avoided by using only a subset of all possible inputs whose corresponding possible outputs are disjoint in this example.

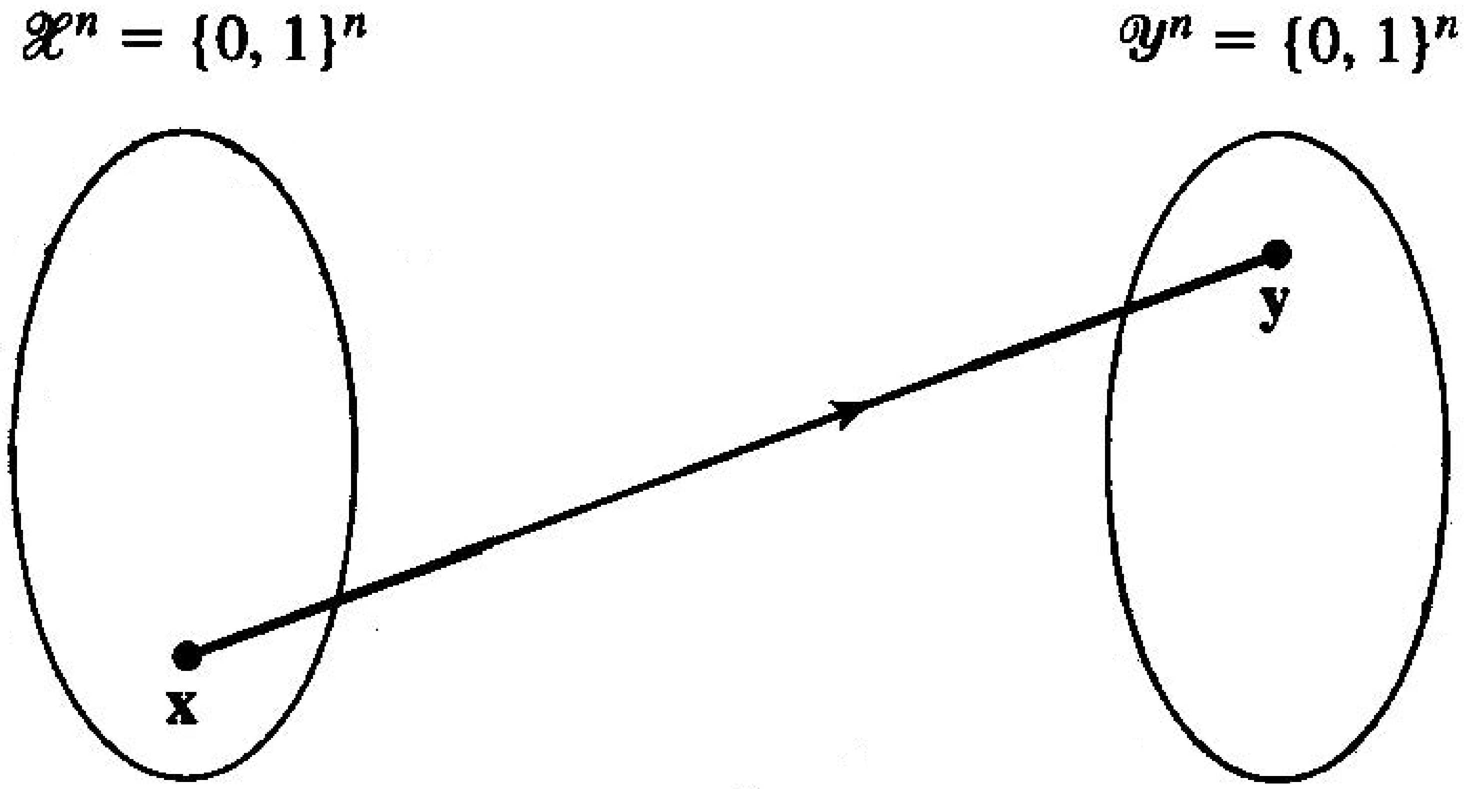
The inputs in the chosen subset of possible inputs should be “far apart” such that their “images” under the channel operation are nonoverlapping (or have negligible overlaps).

Notice that there is no way to have nonoverlapping outputs for a binary symmetric channel.

In order to use the results of the above argument for the binary symmetric channel, we need to apply it not to the channel itself, but to the **extension channel**.

The nth extension of a channel with input alphabet $\mathcal{X}$, output alphabets alphabet $\mathcal{Y}$, and conditional probabilities $P(y|x)$ is a channel with input and output alphabets $\mathcal{X}^n$ and $\mathcal{Y}^n$ and conditional probability $P(\mathbf{y}|\mathbf{x}) = \prod_{i=1}^{n} P(y_i | x_i)$.

The nth extension of a binary symmetric channel takes binary sequences of length n as its input and output which is shown in Figure 9.5.



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

x: A binary sequence of length n.

y: A binary sequence of length n.

Figure 9.5 n th extension of a binary symmetric channel.

By the law of large numbers (see Chapter 4), for n large enough, if a binary sequence of length n is transmitted over the channel, the output will disagree with the input with high probability at $n\epsilon$ positions.

The number of possible sequences that disagree with a sequence of length n at $n\epsilon$ positions is given by $\binom{n}{n\epsilon}$.

By using Stirling's approximation $n! \approx n^n e^{-n}\sqrt{2\pi n}$, we obtain

$$\binom{n}{n\epsilon} \approx 2^{nH_b(\epsilon)} \tag{9.2.1}$$

where $H_b(\epsilon) = -\epsilon \log_2 \epsilon - (1-\epsilon)\log_2(1-\epsilon)$ is the binary entropy function (see Chapter 6).

This means that, for any input sequence of length n, there exist roughly $2^{nH_b(\epsilon)}$ highly probable corresponding output sequences.

On the other hand, the total number of highly probable output sequences is roughly $2^{nH(Y)}$.

Therefore, the maximum number of input sequences that produce almost nonoverlapping output sequences, is at most equal to

$$\begin{aligned} M &= \frac{2^{nH(Y)}}{2^{nH_b(\epsilon)}} \\ &= 2^{n\left(H(Y)-H_b(\epsilon)\right)} \end{aligned} \tag{9.2.2}$$

and the transmission rate per channel use is given by

$$\begin{aligned} R &= \frac{\log M}{n} \\ &= H(Y) - H_b(\epsilon) . \end{aligned} \tag{9.2.3}$$

Note that ϵ depends on the channel and we cannot control it.

However, the probability distribution of the random variable Y depends both on the input distribution $p(x)$ and the channel properties characterized by ϵ.

Figure 9.6 gives a schematic representation of this case.

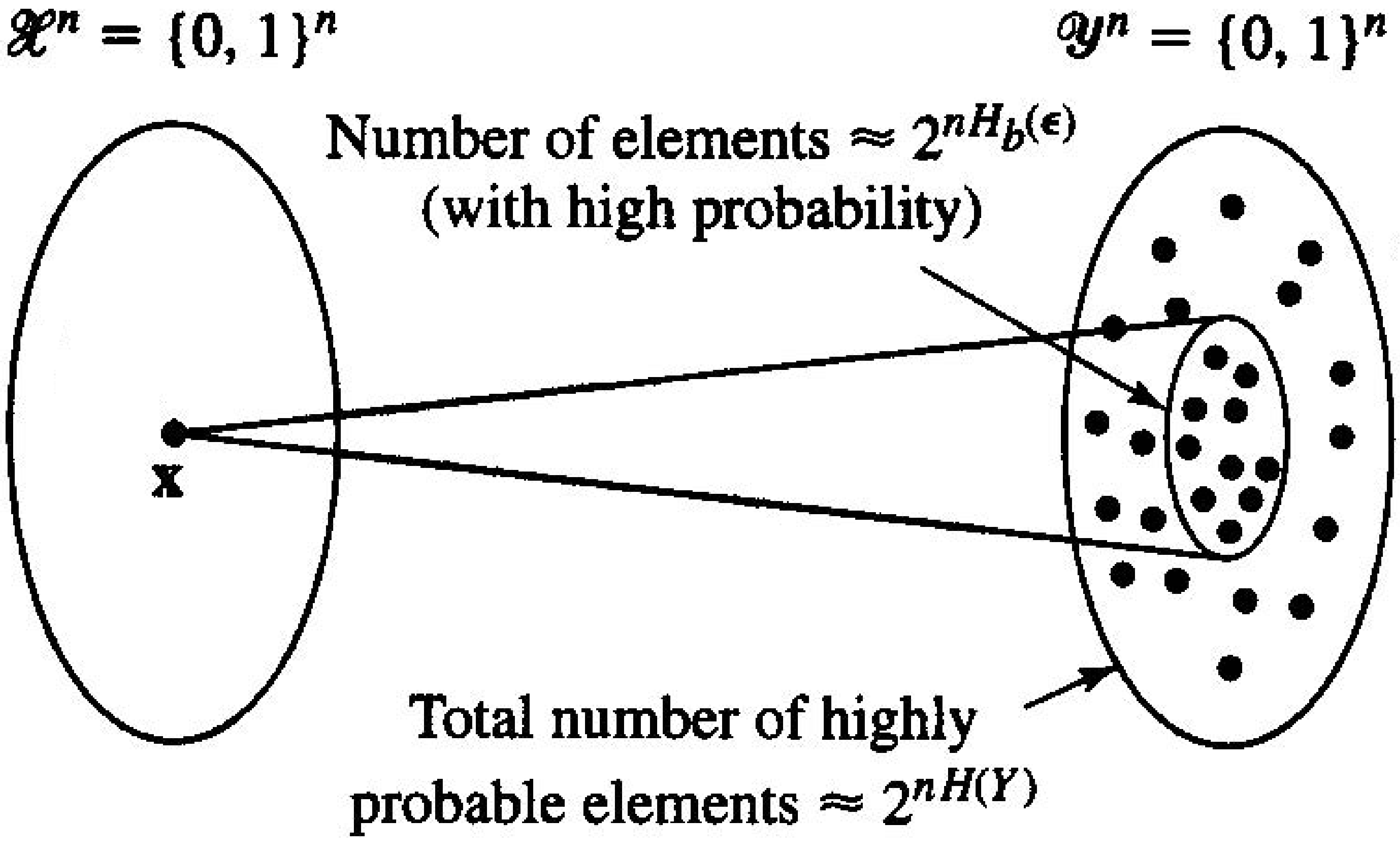

$\mathscr{X}^n = \{0, 1\}^n$
$\mathscr{Y}^n = \{0, 1\}^n$
Number of elements $\approx 2^{nH_b(\epsilon)}$ (with high probability)
x
Total number of highly probable elements $\approx 2^{nH(Y)}$

Figure 9.6 Schematic representation of a BSC.

To maximize the transmission rate over the channel, one has to choose $p(x)$ that maximizes $H(Y)$.

If X is chosen to be a uniformly distributed random variable, that is, $P(X=0)=P(X=1)=0.5$, then Y is uniformly distributed and $H(Y)$ is maximized.

Since the maximum value of $H(Y)$ is 1, from (9.2.3) the maximum of the transmission rate is given by

$$R = 1 - H_b(\epsilon). \tag{9.2.4}$$

It can be proved that the above rate is the maximum rate at which reliable transmission over the BSC is possible.

By **reliable transmission** we mean that the error probability can be made to tend to 0 as the sequence length n tends to infinity.

The channel capacity in this case is shown in Figure 9.7.

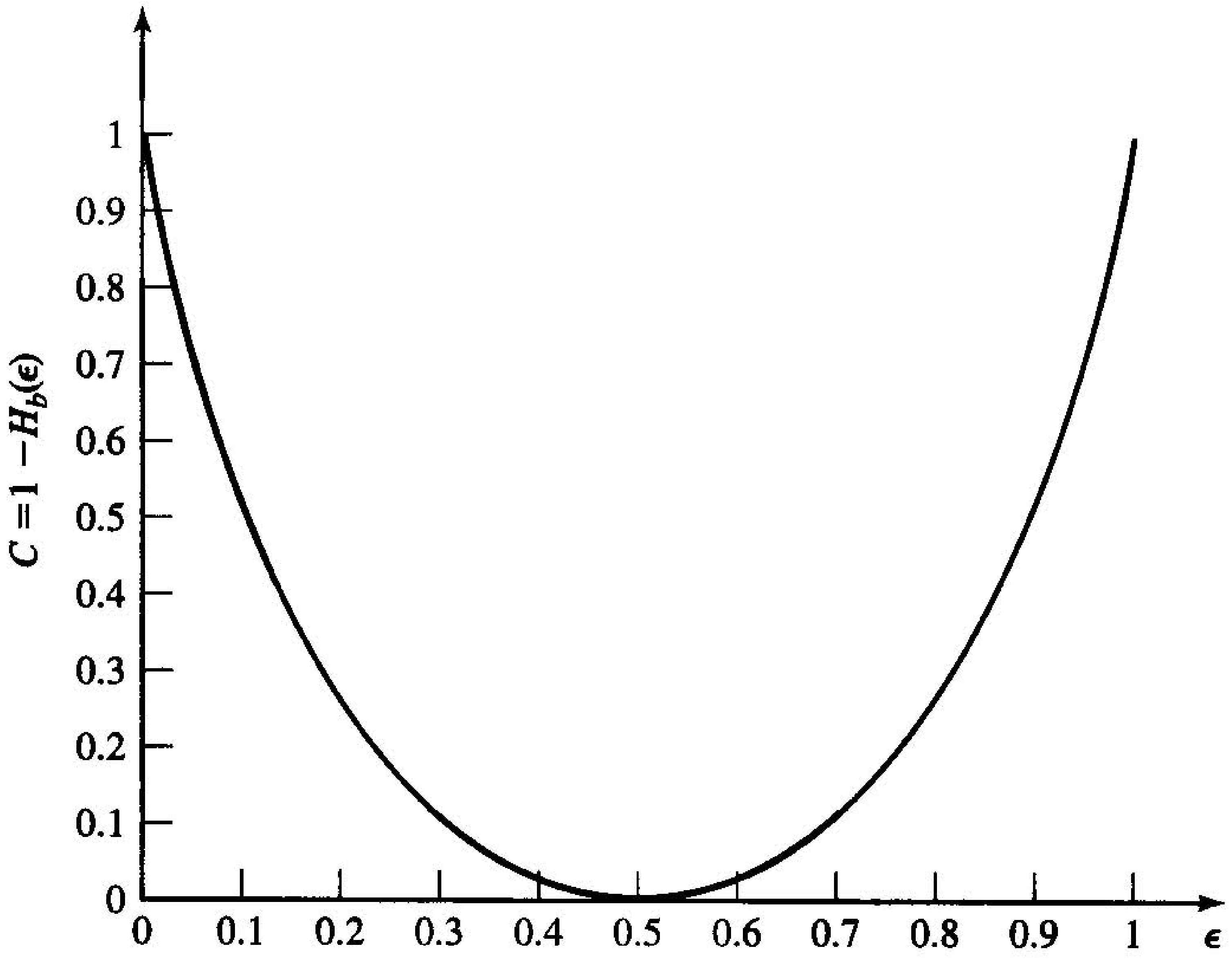


Figure 9.7 The capacity of a BSC.

Note that the both cases with $\epsilon = 0$ and $\epsilon = 1$ result in a **perfect channel** having channel capapcity $C = 1$.

This means that a channel that always flips the input is as good as the channel that transmits the input without error.

The worst case happens when the channel flips the input with probability $\frac{1}{2}$ which leads to a **useless channel** having channel capacity $C = 0$.

The maximum rate at which one can communicate over a discrete-memoryless channel and still make the error probability to approach 0 as the code block length increases, is called the **channel capacity** which is denoted by C.

Theorem 9.2.1 [Noisy Channel-Coding Theorem]

The capacity of a discrete-memoryless channel is given by

$$C = \max_{p(x)} I(X;Y) \tag{9.2.5}$$

where $I(X;Y)$ is the mutual information between the channel input X and the channel output Y (see

Chapter 6).

Definition 6.4.1

The **mutual information** between two discrete random variables X and Y is denoted by $I\left(X;Y\right)$ and defined by

$$I\left(X;Y\right)=H\left(X\right)-H\left(X\middle|Y\right). \tag{6.4.1}$$

If the transmission rate R is less than C, then for any $\delta>0$ there exists a code with block length n large enough whose error probability is less than δ.

If $R>C$, the error probability of any code with any block length is bounded away from 0.

According to this theorem, any communication channel is characterized by a number called capacity that determines how much information can be transmitted over it, regardless of all other properties.

Therefore, to compare two channels from an information transmission point of view, it is enough to compare their capacities.

Ex. 9.2.1

Find the capacity of the channel shown in Figure 9.8.

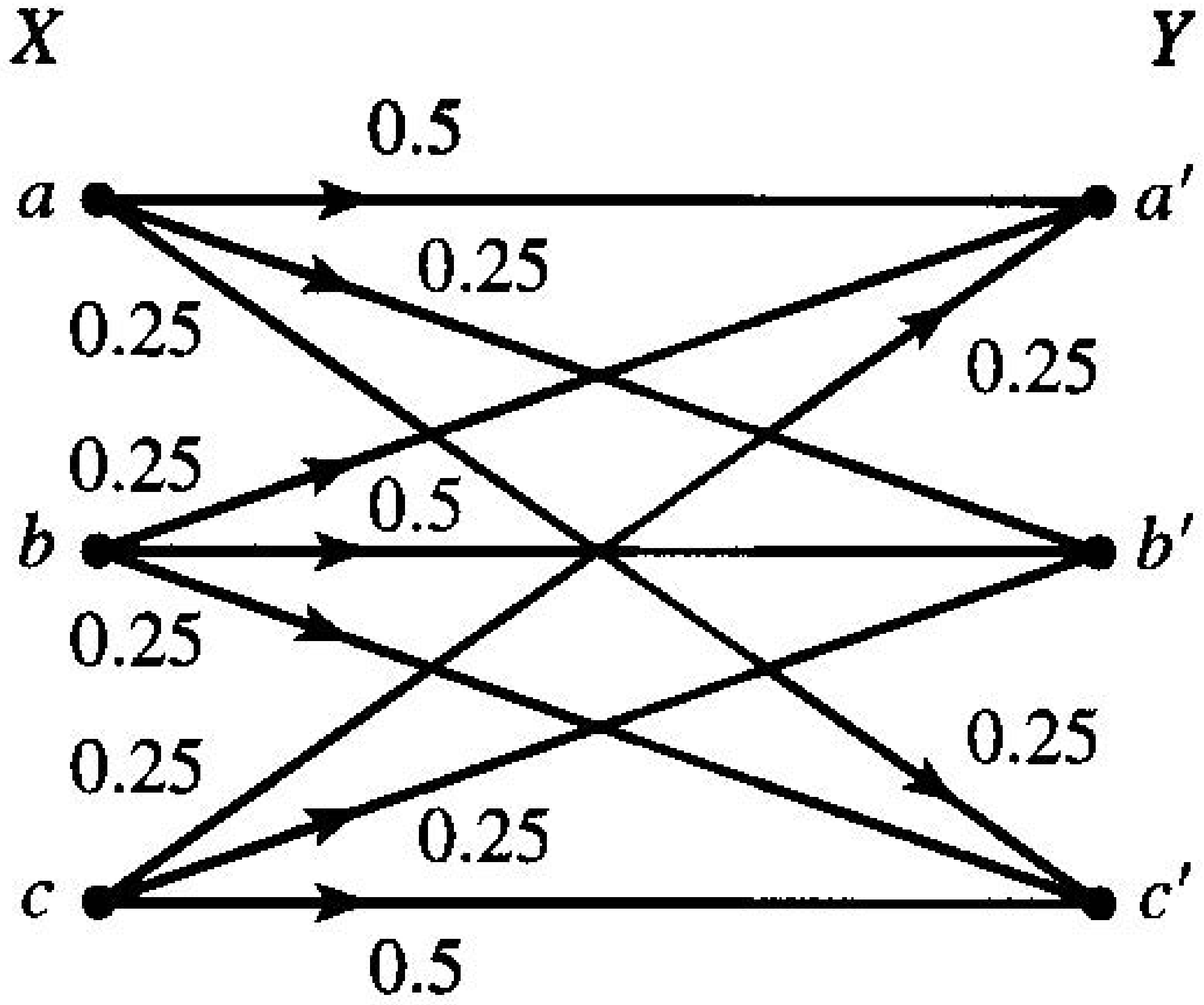


Figure 9.8 DMC of Ex. 9.2.1.

Solution

We have $I(X;Y) = H(Y) - H(Y \mid X)$.

But

$H(Y \mid X) = P(X = a)H(Y \mid X = a) + P(X = b)H(Y \mid X = b) + P(X = c)H(Y \mid X = c)$.

From Figure 9.8, for all three cases of $X = a$, $X = b$, and $X = c$, the random variable Y is ternary with probabilities 0.25, 0.25, and 0.5.

Therefore,

$$\begin{aligned} H(Y \mid X = a) &= H(Y \mid X = b) \\ &= H(Y \mid X = c) \\ &= 1.5. \end{aligned}$$

Because $P(X = a) + P(X = b) + P(X = c) = 1$,

we have

$H(Y \mid X) = 1.5$

and

$$I(X;Y) = H(Y) - H(Y|X)$$
$$= H(Y) - 1.5.$$

To maximize $I(X;Y)$, it needs to maximize $H(Y)$, which is maximized when the random variable Y is equiprobable.

It is not clear in general if there exists an input distribution that results in a uniform distribution on the output.

However, in this special case (due to the symmetry of the channel) a uniform input distribution results in a uniform output distribution and

$$H(Y) = \log_2 3$$
$$= 1.585 \text{ (bits).}$$

Hence, the capacity of this channel is given by

$$C = \max_{p(x)} I(X;Y)$$
$$= H(Y) - H(Y|X)$$
$$= 1.585 - 1.5 = 0.085 \qquad \text{bits/transmission.}$$

9.2.1 Gaussian Chanel Capacity

A discrete-time Gaussian channel with input power constraint is characterized by the input-output relation

$$Y = X + Z \tag{9.2.6}$$

where Z is a zero-mean Gaussian random variable with variance P_N, and an input power constraint that

$$\frac{1}{n}\sum_{i=1}^{n} x_i^2 \le P \tag{9.2.7}$$

applies to any input sequence of length n for n large enough.

For blocks of length n at the input, output, and noise, we have

$$\mathbf{y} = \mathbf{x} + \mathbf{z}\,. \tag{9.2.8}$$

By the law of large numbers, if n is large, we have

$$\begin{aligned}\frac{1}{n}\sum_{i=1}^{n} z_i^2 &= \frac{1}{n}\sum_{i=1}^{n}(y_i - x_i)^2 \\ &\le P_N \end{aligned} \tag{9.2.9}$$

or

$$\|\mathbf{y} - \mathbf{x}\|^2 \le n\,P_N\,. \tag{9.2.10}$$

This means that, with probability approaching 1 (as n increases), $\mathbf{y}$ will be located within an n-dimensional sphere (hypersphere) of radius $\sqrt{nP_N}$ centered at $\mathbf{x}$.

On the other hand, due to the power constraint of P on the input and the independence of the input and noise, the output power is the sum of the input power and the noise power, that is,

$$\begin{aligned}\frac{1}{n}\sum_{i=1}^{n} y_i^2 &= \frac{1}{n}\sum_{i=1}^{n}(x_i + z_i)^2 \\ &= \frac{1}{n}\sum_{i=1}^{n}(x_i^2 + z_i^2) \\ &\le P + P_N \end{aligned} \qquad (9.2.11)$$

or

$$\|\mathbf{y}\|^2 \le n(P + P_N)\,. \qquad (9.2.12)$$

This implies that the output sequences (again, asymptotically and with high probability) will be located within an n-dimensional hypersphere of radius $\sqrt{n(P+P_N)}$ centered at the origin.

Figure 9.9 shows the sequences in the output space.

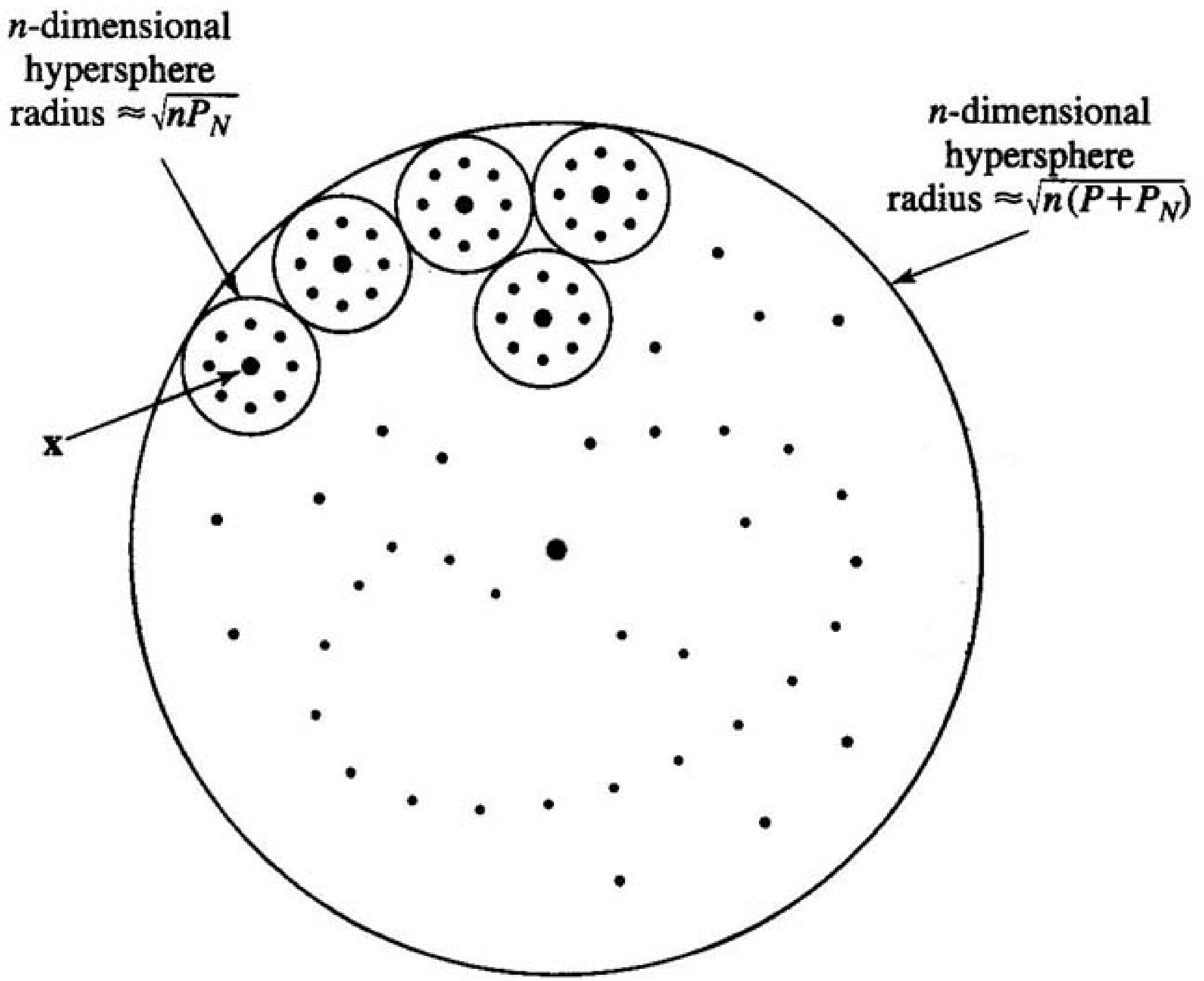

n-dimensional hypersphere radius $\approx \sqrt{nP_N}$
n-dimensional hypersphere radius $\approx \sqrt{n(P+P_N)}$
$\mathbf{x}$

Figure 9.9 The output sequences of a Gaussian channel with power constraint.

Now the question is: how many $\mathbf{x}$ sequences can we transmit over this channel such that the hyperspheres corresponding to these sequences do not overlap in the output space?

An equivalent question is: how many hyperspheres of radius $\sqrt{nP_N}$ can we pack in the hypersphere of radius $\sqrt{n(P_N+P)}$?

The answer is roughly the ratio of the volumes of the two hyperspheres.

Let the volume of an n-dimensional hypersphere is given by

$$V_n = K_n R^n \tag{9.2.13}$$

where R is the radius of the hypersphere and K_n is a constant independent of R.

Then, the number of messages that can be reliably transmitted over this channel is given by

$$M = \frac{K_n\left(n(P_N+P)\right)^{\frac{n}{2}}}{K_n(nP_N)^{\frac{n}{2}}}$$

$$= \left(\frac{P_N+P}{P_N}\right)^{\frac{n}{2}}$$

$$= \left(1 + \frac{P}{P_N}\right)^{\frac{n}{2}}. \tag{9.2.14}$$

Therefore, the capacity of a discrete-time additive white Gaussian noise channel with input power constraint P is given by

$$\begin{aligned} C &= \frac{1}{n}\log_2 M \\ &= \frac{1}{n}\cdot\frac{n}{2}\log_2\left(1 + \frac{P}{P_N}\right) \\ &= \frac{1}{2}\log_2\left(1 + \frac{P}{P_N}\right) \text{ bits/transmission}. \end{aligned} \tag{9.2.15}$$

We can sample a continuous-time, bandlimited, additive white Gaussian noise channel with noise power spectral density $\frac{N_0}{2}$, input power constraint P, and bandwidth W at the Nyquist rate to obtain a discrete-time channel.

Then, the power per sample becomes P and the noise power per sample is given by

$$P_N = \int_{-W}^{+W} \frac{N_0}{2} df$$
$$= WN_0 .$$

Substituting these results into (9.2.15), we obtain

$$C = \frac{1}{2}\log_2\left(1+\frac{P}{N_0 W}\right) \quad \text{bits/transmission}. \tag{9.2.16}$$

If we multiply this result by the number of transmissions/sec, which is $2W$, we obtain the channel capacity

$$C = W\log_2\left(1+\frac{P}{N_0 W}\right) \quad \text{bits/sec}. \tag{9.2.17}$$

Ex. 9.2.2

Find the capacity of a telephone channel with bandwidth $W = 3000$ Hz and SNR of 39 dB.

Solution

The SNR of 39 dB is equivalent to 7943.

Using Shannon’s relation in (9.2.17) we have

$$\begin{aligned} C &= W\log_2\left(1+\frac{P}{N_0W}\right) \\ &= 3000\log_2(1+7943) \\ &\approx 38{,}867 \quad \text{bits/sec}. \end{aligned}$$

9.3 Bounds on Communication

There exists a trade-off between signal power P and bandwidth W in the sense that one can compensate for the other.

Increasing the input signal power increases the channel capacity, because when one has more power to spend, one can choose a larger number of input levels which are far apart and, hence, more information bits/transmission are possible.

However, the increase in capacity as a function of power is logarithmic and slow.

This is because if one is transmitting with a certain number of input levels that are Δ apart to allow a certain level of immunity against noise and wants to increase the number of input levels,
one has to introduce new levels with amplitudes higher than the existing levels, and this requires a lot of power.

Nevertheless, the capacity of the channel can be increased to any value by increasing the input power.

Increasing channel bandwidth W has two contrasting effects.

On one hand, on a larger bandwidth channel one can transmit more samples/sec and, therefore, increase the transmission rate.

On the other hand, a larger bandwidth means a larger input noise to the receiver and this degrades the performance of the channel.

These effects are seen from the two W's which appear in (9.2.17).

Applying L'Hospital's rule to (9.2.17), we obtain

$$\begin{aligned}\lim_{W\to\infty} C &= \frac{P}{N_0}\log e \\ &= 1.44\frac{P}{N_0} \qquad (9.3.1)\end{aligned}$$

which implies that by increasing the bandwidth alone the capacity is not increased to any desired value, contrary to the case of power.

Figure 9.10 shows the channel capacity C versus bandwidth W.

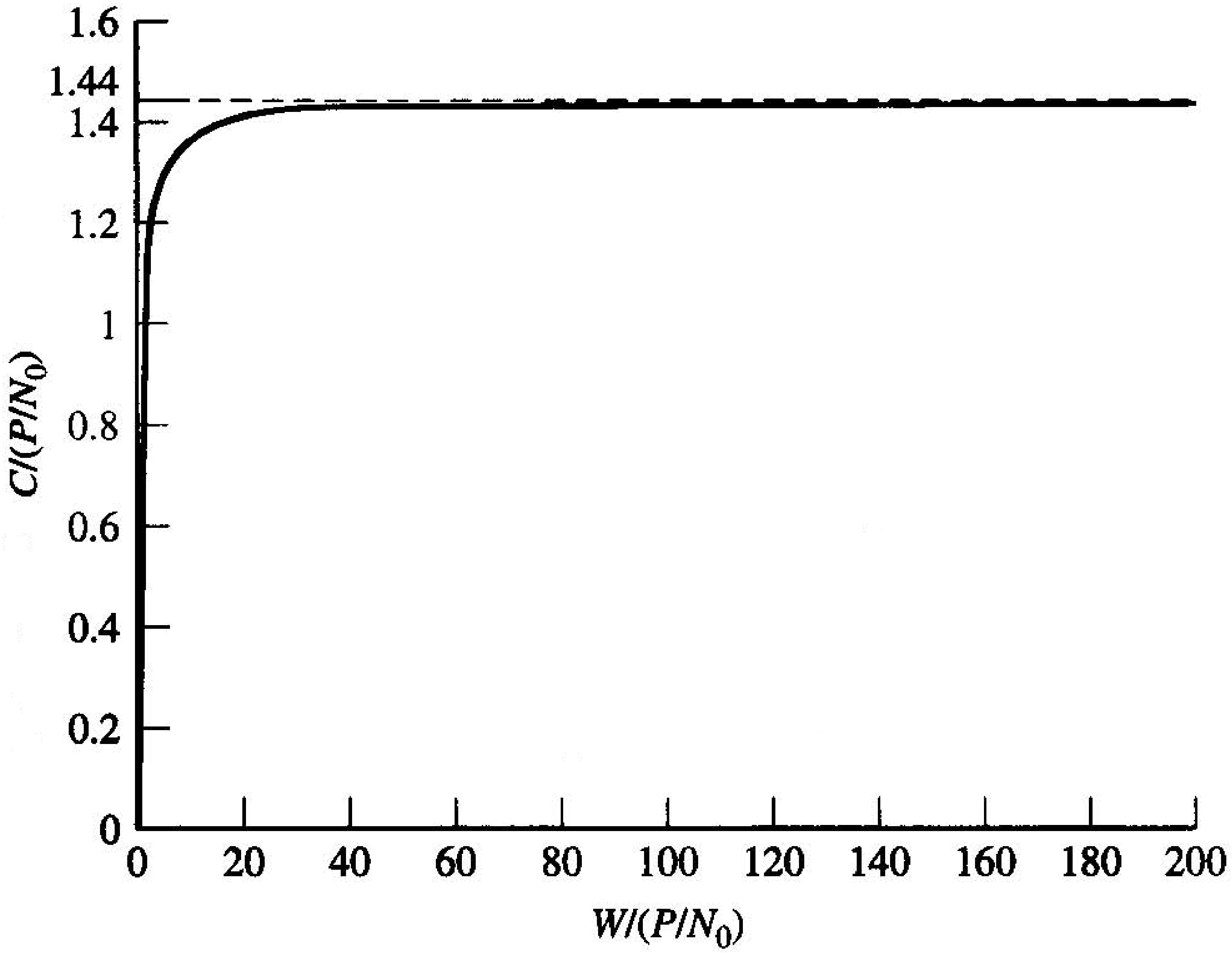


Figure 9.10 Channel capacity versus bandwidth.

In any practical communication system, we must have $R < C$.

If an AWGN channel is employed, we have

$$R < W \log\left(1 + \frac{P}{N_0 W}\right). \tag{9.3.2}$$

By dividing both sides by W and defining the spectral bit rate $r = \frac{R}{W}$, we obtain

$$r < \log\left(1 + \frac{P}{N_0 W}\right). \tag{9.3.3}$$

The energy per bit is given by $\varepsilon_b = \frac{P}{R}$.

By substituting this into (9.3.3), we obtain

$$r < \log\left(1 + r\frac{\varepsilon_b}{N_0}\right) \tag{9.3.4}$$

or, equivalently,

$$\frac{\mathcal{E}_b}{N_0} > \frac{2^r - 1}{r}. \tag{9.3.5}$$

Figure 9.11 shows the spectral bit rate r versus $\frac{\mathcal{E}_b}{N_0}$.

In Figure 9.11, the curve is given by

$$r = \log\left(1 + r\frac{\mathcal{E}_b}{N_0}\right) \tag{9.3.6}$$

which divides the plane into two regions.

In the region below the curve, reliable communication is possible; and in the region above the curve, reliable communication is not possible

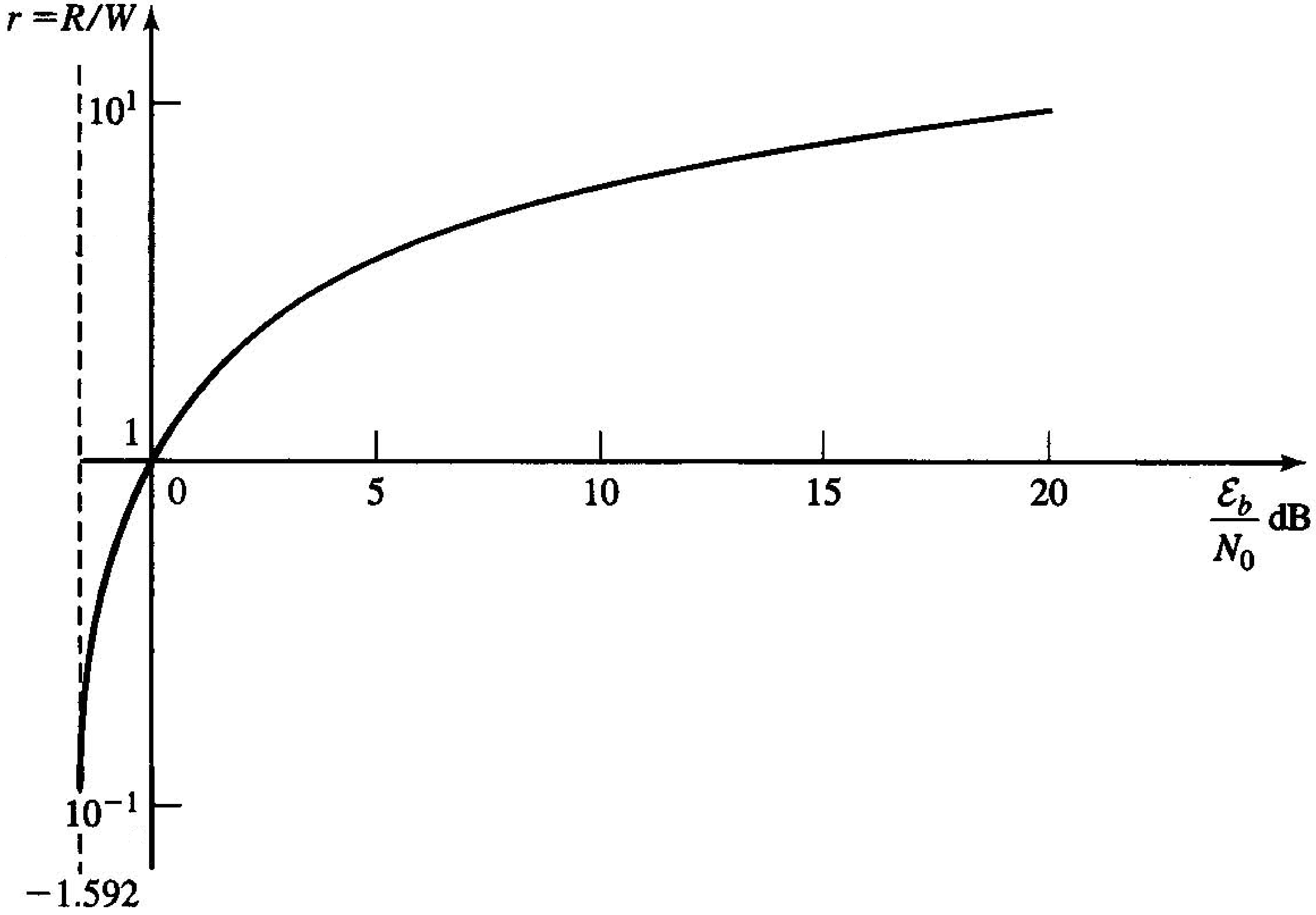


Figure 9.11 Spectral bit rate versus SNR/bit in an optimal system.

The performance of any communication system can be marked by a point in Figure 9.11 and the closer the point is to the curve, the closer is the performance of the system to that of an optimal system.

From this curve, it is seen that (as r tends to 0),

$$\begin{aligned}\frac{\mathcal{E}_b}{N_0} &= \ln 2 \\ &= 0.693 \\ &\sim -1.6\text{dB} \qquad (9.3.7)\end{aligned}$$

is an absolute minimum which is needed for reliable communication.

In other words, for reliable communication, we must have

$$\frac{\mathcal{E}_b}{N_0} > 0.693 . \qquad (9.3.8)$$

In Figure 9.11, if the spectral bit rate $r << 1$, the channel bandwidth is large and the main concern is limitation on power.

This case is usually referred to as the **power-limited case**.

Signaling schemes with high dimensionality such as orthogonal, biorthogonal, and simplex are used in this case.

If the spectral bit rate $r >> 1$, the channel bandwidth is small and, therefore, it is referred to as the **bandwidth-limited case**.

Signaling schemes with low dimensionality (that is, with crowded constellations) such as 256-QAM are used in this case.

Entropy gives a lower bound on the rate of the codes that are capable of reproducing the source output with no error, and the rate-distortion function gives a lower bound on the rate of the source with no error, and the rate-distortion function gives a lower bound on the rate of the codes capable of reproducing the source with distortion D.

If we want to transmit a source U reliably via a channel with capacity C, we require that

$$H(U) < C. \tag{9.3.9}$$

If transmission with a maximum distortion equal to D is desired, then the condition is given by

$$R(D) < C. \tag{9.3.10}$$

These two relations define fundamental limits on the transmission of information.

In both cases, we have assumed that one transmission over the channel is possible for each source output.

Ex. 9.3.1

A zero-mean Gaussian source with power-spectral density

$$S_X(f) = \Pi\left(\frac{f}{20,000}\right)$$

is to be transmitted via a telephone channel described in Example 9.2.2.

Find the minimum distortion achievable if mean-squared distortion is used.

Solution

The bandwidth of the source is $10,000$ Hz, and hencee, it can be sampled at a rate of $20,000$ samples/sec.

The power of the source is given by

$$P_X = \int_{-\infty}^{\infty} S_X(f)df$$

$$= 20,000$$

The variance of each sample is given by

$$\sigma^2 = 20,000$$

and the rate distortion function for $D < 20,000$ is given by

$$R(D) = \frac{1}{2}\log\frac{\sigma^2}{D}$$

$$= \frac{1}{2}\log\frac{20,000}{D} \quad \text{bits/sample}$$

which is equivalent to

$$R(D) = 10,000\log\frac{20,000}{D} \quad \text{bits/sec}.$$

Since the capacity of the channel (derived in Example 9.2.2) $R(D)$ is $38,867$ bits/sec, we can derive the least possible distortion by solving

$$38,867 = 10,000\log\frac{20,000}{D}$$

for D, which results in $D = 1352$.

In general, it can be shown that the rate-distortion function for a waveform Gaussian source with power-spectral density

$$S_X(f) = \begin{cases} A, & |f| < W_S, \\ 0, & \text{otherwise}, \end{cases} \tag{9.3.11}$$

and with distortion measure

$$d\big(u(t), v(t)\big) = \lim_{T\to\infty} \frac{1}{T} \int_{-\frac{T}{2}}^{\frac{T}{2}} [u(t) - v(t)]^2 dt \tag{9.3.12}$$

is given by

$$R(D) = \begin{cases} W_S \log\left(\dfrac{2AW_S}{D}\right), & D < 2AW_S, \\ 0, & D \geq 2AW_S. \end{cases} \tag{9.3.13}$$

If this source is to be transmitted via an additive white Gaussian noise channel with bandwidth W_c, power P, and noise-spectral density N_0, the minimum achievable distortion is obtained by solving the equation

$$W_c \log\left(1 + \frac{P}{N_0 W_c}\right) = W_S \log\left(\frac{2AW_S}{D}\right). \tag{9.3.14}$$

From (9.3.14), we obtain

$$D = 2AW_S\left(1+\frac{P}{N_0W_c}\right)^{-\frac{W_c}{W_S}}. \qquad (9.3.15)$$

It is seen that, in an optimal system, distortion decreases exponentially as $\frac{W_c}{W_S}$ increases.

The ratio $\frac{W_c}{W_S}$ is called the bandwidth-expansion factor.

We can also find the SQNR as

$$\begin{aligned}\text{SQNR} &= \frac{2AW_S}{D}\\ &= \left(1+\frac{P}{N_0W_c}\right)^{\frac{W_c}{W_S}} \qquad (9.3.16)\end{aligned}$$

In an optimal system, the final signal-to-noise (distortion) ratio increases exponentially with the bandwidth expansion factor.

9.3.1 Transmission of Analog Sources by PCM (skipped)

PCM is one of the most commonly used schemes for transmission of analog data.

In a PCM system the quantization noise (distortion) is given by [Section 6.6]

$$E[\tilde{X}^2] = \frac{x_{\max}^2}{3 \cdot 4^v}$$

where $x_{\max}$ is the maximum input amplitude and v is the number of bits/sample.

If the source bandwidth is W_s, then the sampling frequency in an optimal system is $f_s = 2W_s$, and the bit rate is $R = 2vW_s$.

Assuming that the binary data is directly transmitted over the channel, the minimum channel bandwidth that can accommodate this bit rate [Nyquist criterion, see Chapter 8] is given by

$$\begin{aligned} W_c &= \frac{R}{2} \\ &= v\,W_s. \end{aligned} \tag{9.3.17}$$

Therefore, the quantization noise is given by

$$E[\tilde{X}^2] = \frac{x_{\max}^2}{3 \cdot 4^{\frac{W_c}{W_s}}}. \tag{9.3.18}$$

If the output of a PVM system is transmitted over a noisy channel with error probability P_b, some bits will be received in error and another type of distortion, due to transmission, will also be introduced.

This distortion will be independent from the quantization distortion and since the quantization error is zero mean, the total distortion will be the sum of these distortions [see Section 4.3.2].

Assume that P_b is small enough such that in a block of length v, either no error or one error can occur.

This one error can occur at any of the v locations and, therefore, its contribution to the total distortion varies accordingly.

If an error occurs in the least significant bit, it results in a change of Δ in the output level;
if it happens in the next bit, it results in 2Δ change in the output; and
if it occurs at the most significant bit, it causes a level change equal to $2^{v-1}\Delta$ at the output.

Assuming these are the only possible errors, the transmission distortion is obtained as

$$E[D_T^2] = \sum_{i=1}^{v} P_b (2^{i-1}\Delta)^2$$

$$= P_b \Delta^2 \frac{4^{\frac{W_c}{W_s}} - 1}{3} \qquad (9.3.19)$$

and the total distortion is given by

$$D_{\text{total}} = \frac{x_{\max}^2}{3 \cdot 4^{\frac{W_c}{W_s}}} + P_b \Delta^2 \frac{4^{\frac{W_c}{W_s}} - 1}{3}. \qquad (9.3.20)$$

Note that

$$\Delta = \frac{2x_{\max}}{2^v}$$

$$= \frac{x_{\max}}{2^v - 1}. \qquad (9.3.21)$$

Substituting for Δ, and assuming $4^v \gg 1$, (9.3.20) is simplified to

$$D_{\text{total}} \approx \frac{x_{\max}^2}{3} (4^{-\frac{W_c}{W_s}} + 4P_b). \qquad (9.3.22)$$

The signal-to-noise (distortion) ratio at the receiving end is given by

$$\text{SNR} = \frac{3\overline{X^2}}{x_{\max}^2\left(4^{-\frac{W_c}{W_s}} + 4P_b\right)}. \tag{9.3.23}$$

P_b depends on the modulation scheme employed to transmit the outputs of the PCM system.

If binary antipodal signaling is employed is employed, then

$$P_b = Q\left(\sqrt{\frac{2\varepsilon_b}{N_0}}\right) \tag{9.3.24}$$

and if binary orthogonal signaling with coherent detection is used, then

$$P_b = Q\left(\sqrt{\frac{\varepsilon_b}{N_0}}\right). \tag{9.3.25}$$

In (9.3.23), it is seen that for small P_b, the SNR grows almost exponentially with $\frac{W_c}{W_s}$, the bandwidth-expansion factor, and in this sense, a PCM system uses the available bandwidth efficiently.

Recall from Section 5.3, that another frequently used system that trades bandwidth for noise immunity is an

FM system.

However, the SNR of an FM system is a quadratic function of the bandwidth-expansion factor [see (5.3.26)] and, therefore, an FM system is not as bandwidth efficient as a PCM system.

9.4 Coding for Reliable Communication (skipped)

In Chapter 7, it was shown that both in baseband and carrier-modulation schemes, the error probability is a function of the distance between the points in the signal constellation.

In fact, for binary equiprobable signals the error probability can be expressed as [see (7.6.10)]

$$P_e = Q\left(\frac{d_{12}}{\sqrt{2N_0}}\right) \tag{9.4.1}$$

where d_{12} is the Euclidean distance between $s_1(t)$ and $s_2(t)$, given by

$$d_{12}^2 = \int_{-\infty}^{\infty} [s_1(t) - s_2(t)]^2 dt\,. \tag{9.4.2}$$

On the other hand, antipodal signaling with coherent demodulation performs best and has a Euclidean distance $d_{12} = 2\sqrt{\varepsilon_b}$.

Therefore,

$$P_e = Q\left(\sqrt{\frac{2\varepsilon_b}{N_0}}\right). \tag{9.4.3}$$

From the above, it is seen that to decrease the error probability, one has to increase the signal energy.

Increasing the signal energy can be done in two ways: 1) increasing the transmitter power 2) increasing the transmission duration.

Increasing the transmitter power is not always feasible because each transmitter has a limitation on its average power.

Increasing the transmission duration, in turn, decreases the transmission rate and, therefore, it seems that the only way to make the error probability vanish is to let the transmission rate vanish.

In fact, this was the communication engineers' viewpoint in the pre-Shannon era.

By employing orthogonal signals, one can achieve reliable transmission at nonzero rate, as long as $\frac{\mathcal{E}_b}{N_0} > 2\ln 2$ (see Chapter 7).

Here, we show that $\frac{\mathcal{E}_b}{N_0} > \ln 2$ is sufficient to achieve reliable transmission.

Note that this is the same condition which guarantees reliable transmission over an additive white Gaussian noise channel when the bandwidth goes to infinity (see Figure 9.11 and (9.3.7)).

<Skipped.>

Comparing the two bounds, it is concluded that coding results in a power gain equivalent to

$$G_{\text{coding}} = d_{\min}^{H} R_c \tag{9.4.55}$$

which is called the **asymptotic-coding gain**, or simply, **the coding gain**.

The coding gain is a function of two main code parameters, the minimum Hamming distance and the code rate.

Note that in general, $R_c < 1$ and $d_{H,\min} \geq 1$ and, therefore, the coding gain can be greater or less than one.†

There exist many codes that can provide good coding gains.

For a given n and k the best code is the code that can provide the highest minimum Hamming distance.

To study the bandwidth requirements of coding, we observe that when no coding is used, the width of the pulses employed to transmit one bit is given by

$$T_b = \frac{1}{R}. \tag{9.4.56}$$

After using coding, in the same time duration that k pulses were transmitted, we must now transmit n pulses, which means that the duration of each pulse is reduced by a factor of $\frac{k}{n} = R_c$.

† Although in a very bad code design we can even have $d_{H,\min} = 0$, we will ignore such cases.

Therefore, the **bandwidth-expansion ratio** is given by

$$B = \frac{W_{\text{coding}}}{W_{\text{no coding}}} = \frac{1}{R_c} = \frac{n}{k} \tag{9.4.57}$$

which implies that the bandwidth has increased linearly.

It can be proved that, in an AWGN channel, there exists a sequence of codes with parameters (n_i, k_i) with fixed rate ($\frac{k_i}{n_i} = R_c$ independent of i) satisfying

$$R_c = \frac{k}{n} < \frac{1}{2}\log\left(1+\frac{P}{N_0 W}\right) \tag{9.4.58}$$

where $\frac{1}{2}\log\left(1+\frac{P}{N_0 W}\right)$ is the capacity of the channel in bits/transmission,† for which the error probability goes to zero as n_i becomes larger and larger.

† Recall that in the capacity of this channel in bits/sec is $W\log\left(1+\frac{P}{N_0 W}\right)$

For such a scheme, the bandwidth expands by a modest factor and does not, as in orthogonal signaling, grow exponentially.

There are two major types of codes: **block codes**, and **convolutional codes**.

In a block code, the information sequence is broken into blocks of length k and each block is mapped into channel inputs of length n.

This mapping is independent from the previous blocks; i.e., there exists no memory from one block to another block.

In convolutional codes, there exists a shift register of length $k_0 L$ as shown in Figure 9.15.

The information bits enter the shift register k_0 bits at a time and then n_0 bits which are linear combinations of various shift register bits are transmitted over the channel.

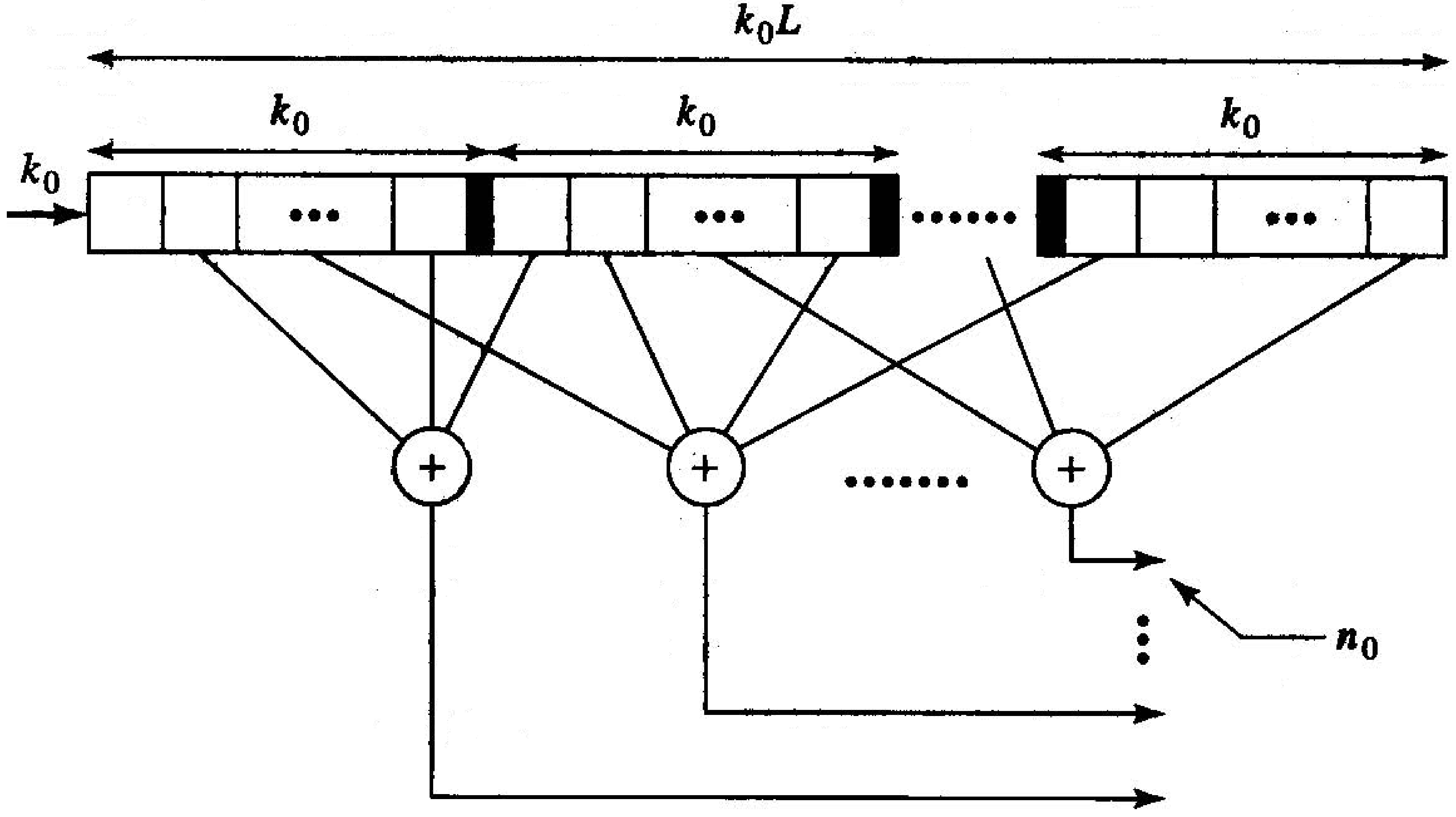


Figure 9.15 A convolutional encoder.

These n_0 bits depend not only on the recent k_0 bits that just entered the shift register, but also on the $(L-1)k_0$ previous contents of the shift register that constitute its **state**.

The quantity

$$m_c = L \tag{9.4.59}$$

is defined as the **constraint length** of the convolutional code.

The number of states of the convolutional code are equal to $2^{(L-1)k_0}$.

The rate of a convolutional code is defined as

$$R_c = \frac{k_0}{n_0}. \tag{9.4.60}$$

The main difference between block codes and convolutional codes is the existence of memory in convolutional codes.

9.5 Linear Block Codes

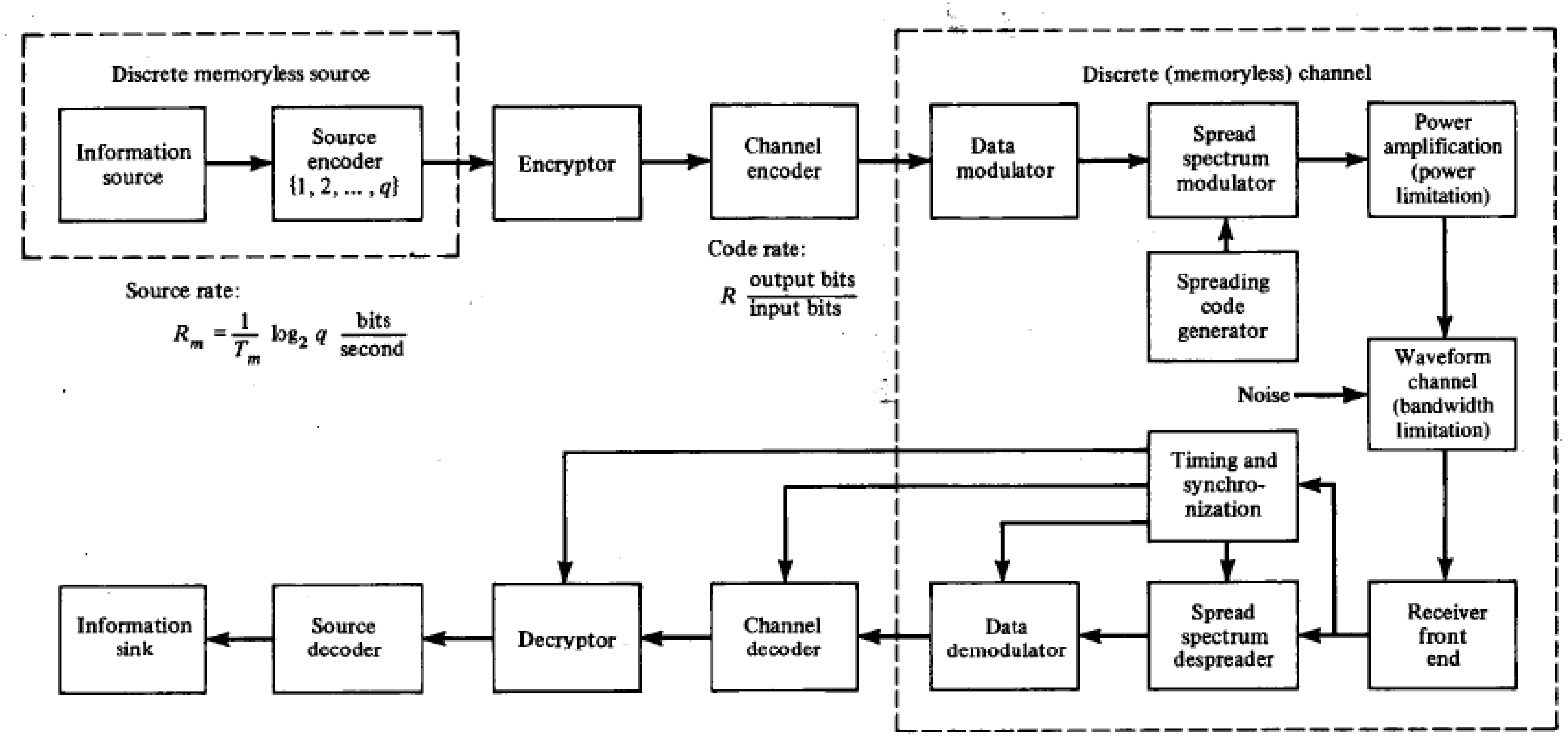

Discrete memoryless source
Information source
Source encoder {1, 2, ... , q}
Encryptor
Channel encoder
Discrete (memoryless) channel
Data modulator
Spread spectrum modulator
Power amplification (power limitation)
Spreading code generator
Waveform channel (bandwidth limitation)
Noise
Timing and synchro-nization
Receiver front end
Spread spectrum despreader
Data demodulator
Channel decoder
Decryptor
Source decoder
Information sink
Source rate:
$R_m = \frac{1}{T_m} \log_2 q \; \frac{\text{bits}}{\text{second}}$
Code rate:
$R \; \frac{\text{output bits}}{\text{input bits}}$

FIGURE P-1. Block diagram of a typical digital communication system.

[R. Peterson, R. Ziemer, and D. Borth, Introduction to Spread Spectrum Communications. Prentice-Hall, 1995.]

An (n, k) (binary) block code[†] is completely defined by $M = 2^k$ binary sequences of length n called **codewords**.

For the time being we consider only binary codes.

A code C consists of M codewords $\mathbf{c}_i$ for $1 \le i \le 2^k$, that is,

$C = \{\mathbf{c}_1, \mathbf{c}_2, \cdots, \mathbf{c}_M\}$

where each $\mathbf{c}_i$ is a sequence if length n with components equal to 0 or 1.

Definition 9.5.1.

A block code is linear if any linear combination of two codewords is also a codeword.
In the binary case this requires that if $\mathbf{c}_i$ and $\mathbf{c}_j$ are codewords then $\mathbf{c}_i \oplus \mathbf{c}_j$ is also a codeword, where $\oplus$ stands for component-wise modulo-2 addition.

A linear block code is a k-dimensional subspace of an n-dimensional space.

It is also obvious that the all zero sequence $\mathbf{0}$ is a codeword of any linear block code since it can be written

[†] From now on we will deal with binary codes unless otherwise specified.

as $\mathbf{c}_i \oplus \mathbf{c}_i$ for any codeword $\mathbf{c}_i$.

Note that linearity of a code only depends on the codewords and not on the way that the information sequences (messages) are mapped to the codewords.

From now on we assume the linear codes we study satisfies a special property that if the information sequence $\mathbf{x}_1$ (of length k) is mapped into the codeword $\mathbf{c}_1$ (of length n) and the information sequence $\mathbf{x}_2$ is mapped into $\mathbf{c}_2$, then $\mathbf{x}_1 \oplus \mathbf{x}_2$ is mapped into $\mathbf{c}_1 \oplus \mathbf{c}_2$.

Ex.9.5.1

A $(5, 2)$ code is defined by

$C = \{00000, 10100, 01111, 11011\}$.

Verify that this code is linear.

Solution

a) If the mapping between the information sequences and codewords is given by

$00 \rightarrow 00000$

$01 \rightarrow 01111$

$10 \rightarrow 10100$

$11 \rightarrow 11011$

then, the special property mentioned above is satisfied as well.

b) If the mapping is given by

$00 \rightarrow 10100$

$01 \rightarrow 01111$

$10 \rightarrow 00000$

$11 \rightarrow 11011,$

then, the special property is not satisfied.

However, in both cases the code is linear.

Definition 9.5.3

The **Hamming weight,** or simply the **weight**, $w(\mathbf{c}_i)$ of a codeword $\mathbf{c}_i$ is the number of nonzero components of the codeword.

Definition 9.5.2

The **Hamming distance** $d(\mathbf{c}_i, \mathbf{c}_j)$ between two codewords $\mathbf{c}_i$ and $\mathbf{c}_j$ is the number of components at which the two codewords differ,[†] that is,

$$d(\mathbf{c}_i, \mathbf{c}_j) = w(\mathbf{c}_i - \mathbf{c}_j)$$

Definition 9.5.4

The minimum distance of a code $d_{\min}$ is the minimum Hamming distance between any two different codewords in the code, that is,

$$d_{\min} = \min_{\substack{\mathbf{c}_i, \mathbf{c}_j \\ i \neq j}} d(\mathbf{c}_i, \mathbf{c}_j). \tag{9.5.1}$$

Definition 9.5.5

The minimum weight of a code $w_{\min}$ is the minimum of the weights of the codewords except the all-zero codeword, that is,

$$w_{\min} = \min_{\mathbf{c}_i \neq \mathbf{0}} w(\mathbf{c}_i). \tag{9.5.2}$$

Theorem 9.5.1

In any linear code, $d_{\min} = w_{\min}$.

[†] From now on Hamming distance is denoted by d and Euclidean distance is denoted by d_{E}.

Proof

If $\mathbf{c}$ is a codeword, then $w(\mathbf{c}) = d(\mathbf{c}_i, \mathbf{0})$.

$$\begin{aligned} d_{\min} &= \min_{\substack{\mathbf{c}_i, \mathbf{c}_j \\ i \neq j}} d(\mathbf{c}_i, \mathbf{c}_j) \\ &= \min_{\substack{\mathbf{c}_i, \mathbf{c}_j \\ i \neq j}} w(\mathbf{c}_i - \mathbf{c}_j) \\ &= \min_{\substack{\mathbf{c}_l \\ \mathbf{c}_l \neq \mathbf{0}}} w(\mathbf{c}_l) \\ &= w_{\min}. \end{aligned}$$

This implies that, in a linear code, corresponding to any weight of a codeword, there exists a Hamming distance between two codewords; and corresponding to any Hamming distance, there exists a weight of a codeword.

Generator and Parity Check Matrices

In an (n, k) (binary) linear block code, let the codeword corresponding to the information sequences $\mathbf{e}_1 = (1000\cdots0)$, $\mathbf{e}_2 = (0100\cdots0)$, $\mathbf{e}_3 = (0010\cdots0)$, $\cdots$, $\mathbf{e}_k = (0000\cdots1)$ of length k be denoted by $\mathbf{g}_1, \mathbf{g}_2, \mathbf{g}_3, \cdots, \mathbf{g}_k$ of length n, respectively, where each of the $\mathbf{g}_i$ sequences is a binary sequences of length

n.

Now, any information sequence $\mathbf{x} = (x_1, x_2, x_3, \cdots, x_k)$ can be written as

$$\mathbf{x} = \sum_{i=1}^{n} x_i \mathbf{e}_i \tag{9.5.3}$$

and the corresponding codeword is given by

$$\mathbf{c} = \sum_{i=1}^{n} x_i \mathbf{g}_i . \tag{9.5.4}$$

If we define the **generator matrix** for this code as

$$\mathbf{G} \triangleq \begin{bmatrix} \mathbf{g}_1 \\ \mathbf{g}_2 \\ \vdots \\ \mathbf{g}_k \end{bmatrix}$$

$$= \begin{bmatrix} g_{11} & g_{12} & \cdots & g_{1n} \\ g_{21} & g_{22} & \cdots & g_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ g_{k1} & g_{k2} & \cdots & g_{kn} \end{bmatrix}, \tag{9.5.5}$$

then, we can write

$$\mathbf{c} = \mathbf{x}\mathbf{G} . \tag{9.5.6}$$

This shows that any linear combination of the rows of the generator matrix is a codeword.

The generator matrix for any linear block code is a $k \times n$ matrix of rank k (because the dimension of the subspace is k, by definition).

The generator matrix of a code completely describes the code, that is, when the generator matrix is given, the structure of an encoder is quite simple.

Ex. 9.5.2

Find the generator matrix for the first code given in Example 9.5.1.

Solution

We have to find the codewords corresponding to information sequences 10 and 01 which are orthogoanl.

These are 10100 and 01111, respectively.

Therefore, we have

$$\mathbf{G} = \begin{bmatrix} 10100 \\ 01111 \end{bmatrix} \tag{9.5.7}$$

It is seen that for the information sequence (x_1, x_2), the codeword is given by

$$(c_1, c_2, c_3, c_4, c_5) = (x_1, x_2)\mathbf{G} \tag{9.5.8}$$

or

$$c_1 = x_1$$

$$c_2 = x_2$$

$$c_3 = x_1 \oplus x_2$$

$$c_4 = x_2$$

$$c_5 = x_2$$

The codeword corresponding to each information sequence starts with a replica of the information sequence itself followed by some extra bits.

Such a code is called a **systematic code** and the extra bits following the information sequence in the codeword are called the **parity check bits** or **redundant bits** (or **redundancy**).

A code which is not systematic is called a **non-systematic code**.

A necessary and sufficient condition for a code to be systematic is that the generator matrix be in the form

$$\mathbf{G} = [\mathbf{I}_k \mid \mathbf{P}] \tag{9.5.9}$$

where $\mathbf{I}_k$ denotes a $k \times k$ identity matrix and $\mathbf{P}$ is a $k \times (n-k)$ binary matrix.

In a systematic code, we have

$$c_i = \begin{cases} x_i, & 1 \le i \le k, \\ \sum_{j=1}^{k} p_{j\,i} x_j, & k+1 \le i \le n, \end{cases} \tag{9.5.10}$$

where summations are modulo-2.

By definition, a linear block code C is a k-dimensional linear subspace of the n-dimensional space.

If we take all sequences of length n that are orthogonal to all vectors in this k-dimensional linear subspace, the result will be an $(n-k)$-dimensional linear subspace called the **orthogonal complement** of the k-dimensional subspace.

This $(n-k)$-dimensional linear subspace defines an $(n, n-k)$ linear code which is known as the **dual** of the original (n, k) code C.

The dual code is denoted by $C^{\perp}$.

The codewords of the original code C and the dual code $C^{\perp}$ are orthogonal to each other.

In particular, if we denote the generator matrix of the dual code by $\mathbf{H}$, which is an $(n-k)\times n$ matrix, then any codeword of the original code is orthogonal to all rows of $\mathbf{H}$, that is,

$$\mathbf{c}\,\mathbf{H}^T = \mathbf{0} \quad \text{for all} \quad \mathbf{c} \in C. \tag{9.5.11}$$

The matrix $\mathbf{H}$, which is the generator matrix of the dual code $C^{\perp}$, is called the **parity check matrix** of the original code C.

Since all rows of the generator matrix are codewords, we have

$$\mathbf{G}\mathbf{H}^T = \mathbf{0}. \tag{9.5.12}$$

In the special case of a systematic code where

$$\mathbf{G} = [\mathbf{I}_k \mid \mathbf{P}], \tag{9.5.13}$$

the parity check matrix is given in the form of

$$\mathbf{H} = [-\mathbf{P}^T \mid \mathbf{I}_k]. \tag{9.5.14}$$

Note that $-\mathbf{P}^T = \mathbf{P}^T$ in the case of binary code.

Ex. 9.5.3

Find the parity check matrix for the code given in Example 9.5.1.

Solution

Here

$$\mathbf{G} = \begin{bmatrix} 10100 \\ 01111 \end{bmatrix}$$

$$\mathbf{I} = \begin{bmatrix} 10 \\ 01 \end{bmatrix}$$

$$\mathbf{P} = \begin{bmatrix} 100 \\ 111 \end{bmatrix}.$$

Since $-\mathbf{P}^T = \mathbf{P}^T$ in the case of binary code, we have

$$\mathbf{P}^t = \begin{bmatrix} 11 \\ 01 \\ 01 \end{bmatrix}$$

and, hence,

$$\mathbf{H} = \left[\begin{array}{c|c} 11 & 100 \\ 01 & 010 \\ 01 & 001 \end{array}\right].$$

Hamming Codes

Hamming codes are a class of (n, k) linear block codes with $n = 2^m - 1,$ $k = 2^m - m - 1,$ and $d_{\min} = 3$, for some integer $m \geq 2$.

These codes are capable of correcting all single errors.

The parity check matrix for a Hamming code consists of all binary sequences of length m except the all-zero sequence.

The rate of a Hamming code is given by

$$R_c = \frac{2^m - m - 1}{2^m - 1} \tag{9.5.15}$$

which becomes close to 1 as m takes large values.

Hamming codes are high-rate codes with relatively small minimum distance $(d_{\min} = 3)$.

Ex. 9.5.4

Find the parity check matrix and the generator matrix of a $(7, 4)$ Hamming code in the systematic form.

Solution

Since $n = 2^m - 1 = 7$, $m = 3$.

Hence, the parity check matrix $\mathbf{H}$ consists of all binary sequences of length 3 except the all-zero sequence.

The parity check matrix in the systematic form is given by

$$\mathbf{H} = \left[\begin{array}{cccc|ccc} 1 & 0 & 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 & 1 \end{array}\right]$$

$$= [-\mathbf{P}^T \,|\, \mathbf{I}_k]$$

and the generator matrix is obtained as

$$\mathbf{G} = [\mathbf{I}_k \,|\, \mathbf{P}]$$

$$\mathbf{G} = \left[\begin{array}{cccc|ccc} 1 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{array}\right].$$

9.5.1 Decoding and Performance of Linear Block Codes

The purpose of using channel coding in communication systems is to reduce the error probability at a given transmitted power by increasing the Euclidean distance between the transmitted signals.

Hence, codes are compared is based on the minimum distance of the code, which is equal to the minimum weight for linear codes.

For a given n and k, a code with a larger minimum distance, $d_{\min}$ (or $w_{\min}$), usually has better performance than a code with a smaller minimum distance.

Soft-Decision Decoding

In the optimum signal-detection scheme on an additive white Gaussian noise channel, detection is made such that the Euclidean distance between the received signal and the transmitted signal is minimized.

In using coded waveforms the concept is the same.

Assuming that binary PSK is adopted for transmission of the coded message, a codeword $\mathbf{c}_i = (c_{i1}, c_{i2}, \cdots, c_{in})$ is mapped into the sequence $s_i(t) = \sum_{k=1}^{n} \psi_{ik}\left(t-(k-1)T\right)$, where

$$\psi_{ik}(t) = \begin{cases} \psi(t), & c_{ik} = 1, \\ -\psi(t), & c_{ik} = 0, \end{cases} \tag{9.5.16}$$

and the waveform $\psi(t)$ has duration T and energy ε, and is equal to zero outside the interval $[0, T]$.

The Euclidean distance between any two signal waveforms is given by

$$\begin{aligned}(d_{ij}^{E})^2 &= \sum_{\substack{1\le l\le n\\ l:c_{il}\ne c_{jl}}} (c_{il}-c_{jl})^2 \\ &= \sum_{\substack{1\le l\le n\\ l:c_{il}\ne c_{jl}}} (2\sqrt{\varepsilon})^2 \\ &= 4d_{ij}^{\mathrm{H}}\varepsilon . \end{aligned} \qquad (9.5.17)$$

For orthogonal signaling, where $\psi_1(t)$ and $\psi_2(t)$ are orthogonal, the Euclidean distance between any two signal waveforms is given by

$$\begin{aligned}(d_{ij}^{E})^2 &= \sum_{\substack{1\le l\le n\\ l:c_{il}\ne c_{jl}}} \int_0^T \left(\psi_1(t)-\psi_2(t)\right)^2 dt \\ &= \sum_{\substack{1\le l\le n\\ l:c_{il}\ne c_{jl}}} (\sqrt{2\varepsilon})^2 \\ &= 2d_{ij}^{H}\varepsilon . \end{aligned} \qquad (9.5.18)$$

Using the relation

$$P_e = Q\left(\frac{d^E}{\sqrt{2N_0}}\right), \tag{9.5.19}$$

we obtain

$$P(j \text{ received} \mid i \text{ sent}) = \begin{cases} Q\left(\sqrt{\dfrac{d_{ij}\varepsilon}{N_0}}\right), & \text{for orthogonal signaling,} \\ Q\left(\sqrt{\dfrac{d_{ij}2\varepsilon}{N_0}}\right), & \text{for antipodal signaling.} \end{cases} \tag{9.5.20}$$

Since $Q(x)$ is a decreasing function of x and $d_{ij} \geq d_{\min}$, we have

$$P(j \text{ received} \mid i \text{ sent}) = \begin{cases} Q\left(\sqrt{\dfrac{d_{\min}\varepsilon}{N_0}}\right), & \text{for orthogonal signaling,} \\ Q\left(\sqrt{\dfrac{d_{\min}2\varepsilon}{N_0}}\right), & \text{for antipodal signaling.} \end{cases} \tag{9.5.21}$$

Using the union bound (see Chapter 7), we obtain

$$P(\text{error} \mid i \text{ sent}) = \begin{cases} (M-1)Q\left(\sqrt{\dfrac{d_{\min}\varepsilon}{N_0}}\right), & \text{for orthogonal signaling,} \\ (M-1)Q\left(\sqrt{\dfrac{2d_{\min}\varepsilon}{N_0}}\right), & \text{for antipodal signaling.} \end{cases} \tag{9.5.22}$$

Assuming equiprobable messages, we have

$$P_e \le \begin{cases} (M-1)Q\left(\sqrt{\dfrac{d_{\min}\varepsilon}{N_0}}\right), & \text{for orthogonal signaling,} \\ (M-1)Q\left(\sqrt{\dfrac{2d_{\min}\varepsilon}{N_0}}\right), & \text{for antipodal signaling.} \end{cases} \tag{9.5.23}$$

In optimal demodulation, the received signal $r(t)$ is passed through a bank of matched filters to obtain the received vector $\mathbf{r}$, and then the closet point in the constellation to $\mathbf{r}$ is found in the Euclidean distance sense.

The type of decoding which involves finding the minimum Euclidean distance is called **soft-decision decoding**, and requires real number computation.

Ex. 9.5.5

Compare the performances of an uncoded data transmission system with a coded transmission using the $(7, 4)$ Hamming code given in Example 9.5.4.

Assume that a binary source has rate $R = 10^4$ bits/sec and the modulation scheme is binary PSK.

Also assume that the channel is an additive white Gaussian noise channel, the received power is $1\,\mu\mathrm{W}$, and the noise power-spectral density is $\frac{N_0}{2} = 10^{-11}$.

Solution

1. If no coding is employed, we have

$$P_b = Q\left(\sqrt{\frac{2\varepsilon_b}{N_0}}\right)$$

$$= Q\left(\sqrt{\frac{2P}{RN_0}}\right). \tag{9.5.24}$$

As $\frac{2P}{RN_0} = \frac{10^{-6}}{10^4 \times 10^{-11}} = 10$, we have

$$\begin{aligned} P_b &= Q(\sqrt{10}) \\ &= Q(3.16) \\ &\approx 7.86\times 10^{-4}. \end{aligned} \tag{9.5.25}$$

The error probability for a four-bit sequence is given by

$$\begin{aligned} P_{\text{error in 4 bits}} &= 1-(1-P_b)^4 \\ &\approx 3.1\times 10^{-3}. \end{aligned} \tag{9.5.26}$$

2. If coding is employed, we have $d_{\min} = 3$, and

$$\begin{aligned} \frac{\mathcal{E}}{N_0} &= R_c \frac{\mathcal{E}_b}{N_0} \\ &= R_c \frac{P}{RN_0} \\ &= \frac{4}{7}\times 5 \\ &= \frac{20}{7}. \end{aligned}$$

The **message** error probability is given by

$$P_e \le (M-1)Q\left(\sqrt{\frac{2d_{\min}\varepsilon}{N_0}}\right)$$

$$= 15Q\left(\sqrt{3\frac{40}{7}}\right)$$

$$= 15Q(4.14)$$

$$\approx 2.6\times 10^{-4}$$

which implies that the error probability is reduced by a factor of 12 by using this simple code.

Note that the cost of using the code is an increase in the bandwidth required for transmission of the messages.

The bandwidth expansion ratio is given by

$$\frac{\mathrm{W}_{\text{coded}}}{\mathrm{W}_{\text{uncoded}}} = \frac{1}{R_c}$$

$$= \frac{7}{4}$$

$$= 1.75.$$

Hard-Decision Decoding

A simpler decoding scheme is to make hard binary decisions on the components of the received vector $\mathbf{r}$, and then to find the codeword which is closest to it in the sense of Hamming distance.

Ex.9.5.6

A $(3,1)$ code consists of the two codewords 000 and 111.

The codewords are transmitted using binary PSK modulation with $\varepsilon = 1$.

Suppose that the received vector (that is, the sampled outputs of the matched filters) is $\mathbf{r} = (0.5, 0.5, -3)$.

In soft decision decoding, we compare the Euclidean distance between $\mathbf{r}$ and the two signal points $(1,1,1)$ which corresponds to 111 and $(-1,-1,-1)$ which corresponds to 000 and choose the signal point which gives smaller distance.

We have

$\left(d^{\mathrm{E}}(\mathbf{r}, (1,1,1)\right)^2 = 0.5^2 + 0.5^2 + 4^2 = 16.5$ and

$$\left(d^{\mathrm{E}}(\mathbf{r},(-1,-1,-1)\right)^2 = 1.5^2 + 1.5^2 + (-2)^2 = 8.5$$

Hence, In soft-decision decoding the decoder would decode $\mathbf{r}$ as $(-1,-1,-1)$ or, equivalently, $(0,0,0)$.

However, if hard-decision decoding is employed, $\mathbf{r}$ is first component-wise detected as 1 or 0.

The resulting vector $\mathbf{y}$ is therefore $\mathbf{y} = (1,1,0)$.

Hence, in hard-decision decoding the decoder compare $\mathbf{y}$ with the $(1,1,1)$ and $(0,0,0)$ and find the closer one in the Hamming distance sense, of which result is $(1,1,1)$.

Notice that the results of soft-decision decoding and hard-decision decoding can be different.

Soft-decision decoding is the optimal detection method and achieves a lower probability of error than hard-decision decoding

There are three basic steps involved in hard-decision decoding.

First, we perform demodulation by passing the received $r(t)$ through the matched filters and sampling the output to obtain the $\mathbf{r}$ vector.

Second, we compare the components of $\mathbf{r}$ with the thresholds and quantize each component to one of the two levels to obtain $\mathbf{y}$ vector.

Third, we perform decoding by finding the codeword that is closest to $\mathbf{y}$ in the sense of Hamming distance.

In this section, we present a systematic approach to perform hard-decision decoding.

First, we introduce a **standard array**.

Let the codewords of the code be denoted by $\mathbf{c}_1, \mathbf{c}_2, \cdots, \mathbf{c}_M$, where each of them is of length n and $M = 2^k$, and let $\mathbf{c}_1$ denote the all-zero codeword.

A standard array is a $2^{n-k} \times 2^k$ array whose elements are binary sequences of length n and is generated by writing all the codewords in a row starting with the all zero codeword.

This constitutes the first row of the standard array.

To write the second row, we look among all the binary sequences of length n that are not in the first row of the array.

Choose one of these sequences that has the minimum weight and denote it by $\mathbf{e}_1$.

Write it under[†] $\mathbf{c}_1$ and write $\mathbf{e}_1 \oplus \mathbf{c}_i$ under $\mathbf{c}_i$ for $2 \le i \le M$.

The third row of the array is written similarly to the second row as follows.

From the binary n-tuples that **have not been used in the first two rows**, we choose one with minimum weight and call it $\mathbf{e}_2$.

Then, the elements of the third row become $\mathbf{c}_i \oplus \mathbf{e}_2$, $2 \le i \le M$.

[†] Note that $\mathbf{c}_1 \oplus \mathbf{e}_1 = \mathbf{e}_1$, since $c_1 = (0, 0, \cdots, 0)$.

This process is continued until no binary n-tuples remains to start a new row.

Figure 9.16 shows the standard array generated as explained above.

$$\begin{array}{cccc}
\mathbf{c}_1 & \mathbf{c}_2 & \mathbf{c}_3 & \mathbf{c}_M \\
\mathbf{e}_1 & \mathbf{e}_1 \oplus \mathbf{c}_2 & \mathbf{e}_1 \oplus \mathbf{c}_3 & \mathbf{e}_1 \oplus \mathbf{c}_M \\
\mathbf{e}_2 & \mathbf{e}_2 \oplus \mathbf{c}_2 & \mathbf{e}_2 \oplus \mathbf{c}_3 & \mathbf{e}_2 \oplus \mathbf{c}_M \\
\vdots & \vdots & \vdots & \vdots \\
\mathbf{e}_{2^{(n-k)}-1} & \mathbf{e}_{2^{(n-k)}-1} \oplus \mathbf{c}_2 & \mathbf{e}_{2^{(n-k)}-1} \oplus \mathbf{c}_3 & \mathbf{e}_{2^{(n-k)}-1} \oplus \mathbf{c}_M
\end{array}$$

Figure 9.16 Standard array.

Each row of the standard array is called a **coset** and the first element of each coset ($\mathbf{e}_l$, in general) is called the **coset leader**.

Theorem 9.5.2.

All elements of the standard array are distinct.

Proof.

Assume that two elements of the standard array are equal.

This can happen in two ways.

1. The two equal elements belong to the same coset.

   In this case we have $\mathbf{c}_l \oplus \mathbf{c}_i = \mathbf{e}_l \oplus \mathbf{c}_j$, from which we conclude $\mathbf{c}_i = \mathbf{c}_j$ which is impossible.

2. The two equal elements belong to two different cosets.

   In this case we have $\mathbf{e}_l \oplus \mathbf{c}_i = \mathbf{e}_k \oplus \mathbf{c}_j$, for $l \neq m$, which means $\mathbf{e}_l = \mathbf{e}_k \oplus (\mathbf{c}_j \oplus \mathbf{c}_i^c) = \mathbf{e}_k \oplus (\mathbf{c}_j \oplus \mathbf{c}_i)$ where $\mathbf{c}_i^c$ is a complement of $\mathbf{c}_i$ modulo 2

By linearity of the code $\mathbf{c}_i \oplus \mathbf{c}_j$ is also a codeword which is denoted by $\mathbf{c}_m$.

Therefore, $\mathbf{e}_l = \mathbf{e}_k \oplus \mathbf{c}_m$ and, hence, $\mathbf{e}_l$ and $\mathbf{e}_k$ belong to the same coset, which is impossible since by assumption $k \neq l$.

From Theorem 9.5.2, we conclude that the standard array contains exactly

$$\frac{\#\text{ of sequences of length } n}{\#\text{ of codewords}} = \frac{2^n}{2^k} = 2^{n-k} \text{ rows.}$$

Theorem 9.5.3.

If $\mathbf{y}_1$ and $\mathbf{y}_2$ are elements of the same coset, we have $\mathbf{y}_1\mathbf{H}^T = \mathbf{y}_2\mathbf{H}^T$.

Proof.

Since $\mathbf{y}_1$ and $\mathbf{y}_2$ are in the same coset, $\mathbf{y}_1 = \mathbf{e}_l \oplus \mathbf{c}_i$ and $\mathbf{y}_2 = \mathbf{e}_l \oplus \mathbf{c}_j$, where $\mathbf{c}_i \neq \mathbf{c}_j$, therefore

$$\begin{aligned}
\mathbf{y}_1\mathbf{H}^T &= (\mathbf{e}_l \oplus \mathbf{c}_i)\mathbf{H}^T \\
&= \mathbf{e}_l\mathbf{H}^T + \mathbf{0} \\
&= (\mathbf{e}_l \oplus \mathbf{c}_j)\mathbf{H}^T \\
&= \mathbf{y}_2\mathbf{H}^T.
\end{aligned}$$

From Theorem 9.5.3, we conclude that each coset of the standard array is uniquely identified by the product $\mathbf{e}_l\mathbf{H}^T$.

For any binary sequence $\mathbf{y}$ of length n, we define the **syndrome** $\mathbf{s}$ as

$$\mathbf{s} = \mathbf{y}\mathbf{H}^T . \tag{9.5.27}$$

If $\mathbf{y} = \mathbf{e}_l \oplus \mathbf{c}_i$, that is, $\mathbf{y}$ belongs to the $(l+1)$st coset, then obviously $\mathbf{s} = \mathbf{e}_l\mathbf{H}^T$.

The syndrome is a binary sequence of length $n-k$ and there exists a unique syndrome corresponding to each coset.

Obviously, the syndrome corresponding to the first coset, which consists of the codewords, is $\mathbf{s} = \mathbf{0}\mathbf{H}^T = \mathbf{0}$.

Ex.9.5.7

Find the standard array for the $(5, 2)$ code with codewords 00000, 10100, 01111, 11011.

Also find the syndromes corresponding to each coset.

Solution

The generator polynomial of the code is given by

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

and the parity check matrix corresponding to $\mathbf{G}$ is given by

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

Using the construction procedure described before, the standard array is obtained as

00000	01111	10100	11011	syndrome = 000
10000	11111	00100	01011	syndrome = 100
01000	00111	11100	10011	syndrome = 111
00010	01101	10110	11001	syndrome = 010
00001	01110	10101	11010	syndrome = 001
11000	10111	01100	00011	syndrome = 011
10010	11101	00110	01001	syndrome = 110
10001	11110	00101	01010	syndrome = 101

Assume that that the received vector $\mathbf{r}$ is compared component-wise with a threshold for hard decision and the resulting binary vector is $\mathbf{y}$.

Then, find the codeword which is at the shortest Hamming distance from $\mathbf{y}$.

First, we find the coset in which $\mathbf{y}$ is located .

To do this, calculate the syndrome of $\mathbf{y}$ by $\mathbf{s} = \mathbf{y}\mathbf{H}^T$ and refer to the standard array to find the coset corresponding to $\mathbf{s}$.

Suppose that the coset leader corresponding to this coset is $\mathbf{e}_l$.

Then, $\mathbf{y} = \mathbf{e}_l \oplus \mathbf{c}_i$ for some i , because $\mathbf{y}$ belongs to this coset.

Hence, the Hamming distance of $\mathbf{y}$ from any codeword $\mathbf{c}_j$ is given by

$$\begin{aligned} d(\mathbf{y}, \mathbf{c}_j) &= w(\mathbf{y} \oplus \mathbf{c}_j) \\ &= w(\mathbf{e}_l \oplus \mathbf{c}_i \oplus \mathbf{c}_j) . \end{aligned} \tag{9.5.28}$$

Because the code C is linear, $\mathbf{c}_i \oplus \mathbf{c}_j$ is also a codeword, that is, $\mathbf{c}_i \oplus \mathbf{c}_j = \mathbf{c}_k$ for some integer k , $1 \le k \le M$.

This implies that

$$d(\mathbf{y},\mathbf{c}_j) = w(\mathbf{c}_k \oplus \mathbf{e}_l) \tag{9.5.29}$$

where $\mathbf{c}_k \oplus \mathbf{e}_l$ belongs to the same coset which $\mathbf{y}$ belongs to.

Therefore, to minimize $d(\mathbf{y},\mathbf{c}_j)$, we have to find the minimum weight element in the coset to which $\mathbf{y}$ belongs.

By the construction procedure of the standard array, this element is the coset leader, that is, we choose $\mathbf{c}_k = \mathbf{0}$ and, therefore, $\mathbf{c}_j = \mathbf{c}_i$.

This implies that $\mathbf{y}$ is decoded into $\hat{\mathbf{c}} = \mathbf{c}_i$ by finding

$$\begin{aligned}\hat{\mathbf{c}} &= \mathbf{y} \oplus \mathbf{e}_l \\ &= \mathbf{c}_i .\end{aligned} \tag{9.5.30}$$

This procedure for hard-decision decoding can be summarized as follows.

1. Find $\mathbf{r}$, the vector representation of the received signal.

2. Compare each component of $\mathbf{r}$ to the optimal threshold (usually 0) and make a binary decision on it to

obtain the binary vector $\mathbf{y}$.

3. Find $\mathbf{s} = \mathbf{y}\mathbf{H}^T$ the syndrome of $\mathbf{y}$.

4. Find the coset corresponding to $\mathbf{s}$ by using the standard array.

5. Find the coset leader $\mathbf{e}$ and decoded $\mathbf{y}$ as $\hat{\mathbf{c}} = \mathbf{y} \oplus \mathbf{e}$.

Because in this decoding scheme the difference between the vector $\mathbf{y}$ and the decoded vector $\mathbf{c}$ is $\mathbf{e}$, the binary n-tuple $\mathbf{e}$ is frequently referred to as the **error pattern**.

This means that the **coset leaders constitute the set of all correctable error patterns**.

In hard-decision decoding the error probability for each bit for antipodal signaling is given by

$$P_b = Q\left(\sqrt{\frac{2\mathcal{E}_b}{N_0}}\right) \tag{9.5.31}$$

and that for orthogonal signaling is given by

$$P_b = Q\left(\sqrt{\frac{\mathcal{E}_b}{N_0}}\right). \tag{9.5.32}$$

The channel between the input codeword $\mathbf{c}$ and the output of the hard limiter $\mathbf{y}$ is a binary-input binary-output channel that can be modeled by a binary-symmetric channel with crossover probability P_b.

Because the code is linear, the distance between any two codeword $\mathbf{c}_i$ and $\mathbf{c}_j$ is the distance between the all-zero codeword $\mathbf{0}$ and $\mathbf{c}_i \oplus \mathbf{c}_j = \mathbf{c}_k$.

Assume that the all-zero codeword $\mathbf{0}$ is transmitted without loss of generality.

By the union bound, the error probability cannot exceed $(M-1)$ times the probability of decoding into the codeword that is closet to $\mathbf{0}$ in the sense of Hamming distance.

For this codeword, denoted by $\mathbf{c}$, which is at distance $d_{\min}$ from $\mathbf{0}$, we have

$$P(\text{decode to } \mathbf{c} \,|\, \mathbf{0} \text{ sent}) \le \begin{cases} \displaystyle\sum_{i=\frac{d_{\min}+1}{2}}^{d_{\min}} \binom{d_{\min}}{i} P_b^i (1-P_b)^{d_{\min}-i}, & d_{\min} \text{ odd}, \\ \displaystyle\sum_{i=\frac{d_{\min}}{2}+1}^{d_{\min}} \binom{d_{\min}}{i} P_b^i (1-P_b)^{d_{\min}-i} + \frac{1}{2}\binom{d_{\min}}{\frac{d_{\min}}{2}} P_b^{\frac{d_{\min}}{2}} (1-P_b)^{\frac{d_{\min}}{2}}, & d_{\min} \text{ even}, \end{cases}$$

or, in general,

$$P(\text{decode to } \mathbf{c} \,|\, \mathbf{0} \text{ sent}) \le \sum_{i=\left\lfloor \frac{d_{\min}+1}{2} \right\rfloor}^{d_{\min}} \binom{d_{\min}}{i} P_b^i (1-P_b)^{d_{\min}-i}. \tag{9.5.33}$$

Therefore,

$$\begin{aligned} P_e &\le (M-1) P(\text{decode to } \mathbf{c} \,|\, \mathbf{0} \text{ sent}) \\ &\le (M-1) \sum_{i=\left\lfloor \frac{d_{\min}+1}{2} \right\rfloor}^{d_{\min}} \binom{d_{\min}}{i} P_b^i (1-P_b)^{d_{\min}-i} \end{aligned} \tag{9.5.34}$$

which is an upper bound on the error probability of a linear block code using hard-decision decoding.

Both in soft-decision and hard-decision decoding, $d_{\min}$ plays a major role in bounding the error probability.

For a given (n, k), it is desirable to have codes with large $d_{\min}$.

It can be shown that soft-decision decoding has roughly 2 dB better performance than hard-decision decoding for an additive white Gaussian noise channel.

It can also be shown that if an 8-level quantizer (three bits/component) is employed instead of to a 2-level quantizer for each component of $\mathbf{r}$, performance difference with soft decision (with infinite precision) reduces to 0.1 dB.

This multilevel quantization scheme, which is a compromise between soft decision (having infinite precision) and hard decision, is also referred to as soft decision in the literature.

Ex.9.5.8

If hard-decision decoding is employed in Example 9.5.5, how will the results change?

Solution

Here $P_b = Q\left(\sqrt{\frac{40}{7}}\right) = Q(2.39) = 0.0084$ and

$d_{\min} = 3$.

Therefore,

$$P_e \le \binom{7}{2} P_b^2 (1-P_b)^5 + \binom{7}{3} P_b^3 (1-P_b)^4 + \cdots + P_b^7$$

$$\approx 21\, P_b^2$$

$$\approx 1.5 \times 10^{-3}.$$

In this case, coding has decreased the error probability by a factor of 2, compared to 12 in the soft-decision case.

Error Detection versus Error Correcting

Let C be a linear block code with minimum distance $d_{\min}$.

Then, if $\mathbf{c}$ is transmitted and hard-decision decoding is employed, any codeword will be decoded correctly if the received $\mathbf{y}$ is closer to $\mathbf{c}$ than any other codeword as shown in Figure 9.17.

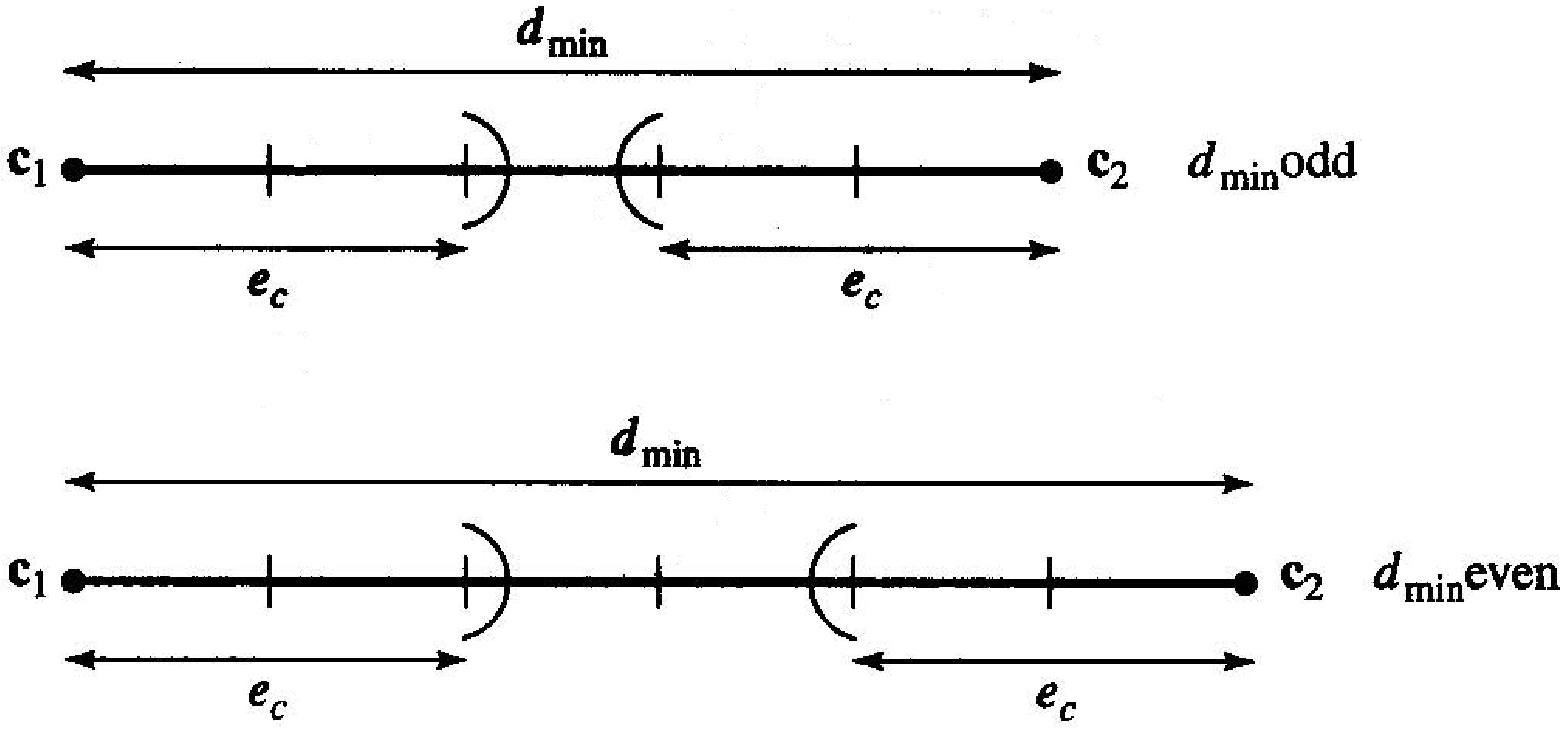


Figure 9.17 Relation between e_c and $d_{\min}$.

As shown in Figure 9.17, around each codeword there is a “Hamming sphere” of radius e_c where e_c is the number of correctable errors.

As long as these spheres are disjoint, the code is capable of correcting e_c errors.

The condition for nonoverlapping spheres is expressed as

$$\begin{cases} 2e_c + 1 = d_{\min}, & d_{\min} \text{ odd}, \\ 2e_c + 2 = d_{\min}, & d_{\min} \text{ even}, \end{cases}$$

or

$$e_c = \begin{cases} \dfrac{d_{\min} - 1}{2}, & d_{\min} \text{ odd}, \\ \dfrac{d_{\min} - 2}{2}, & d_{\min} \text{ even}, \end{cases} \tag{9.5.35}$$

which can be summarized as

$$e_c = \left\lfloor \frac{d_{\min} - 1}{2} \right\rfloor. \tag{9.5.36}$$

In a communication system where a feedback link is available from the receiver to the transmitter, it might be desirable to detect if an error has occurred and, if so, to ask the transmitter via the feedback channel to retransmit the message.

The error-detection capability of a code in the absence of error correction is given by $e_d = d_{\min} - 1$, because if $d_{\min} - 1$ or fewer errors occur, the transmitted codeword will be converted to a word sequence which is not a codeword and, therefore, an error is detected.

If both error correction and error detection are desirable, then there is a trade-off between them as shown Figure 9.18.

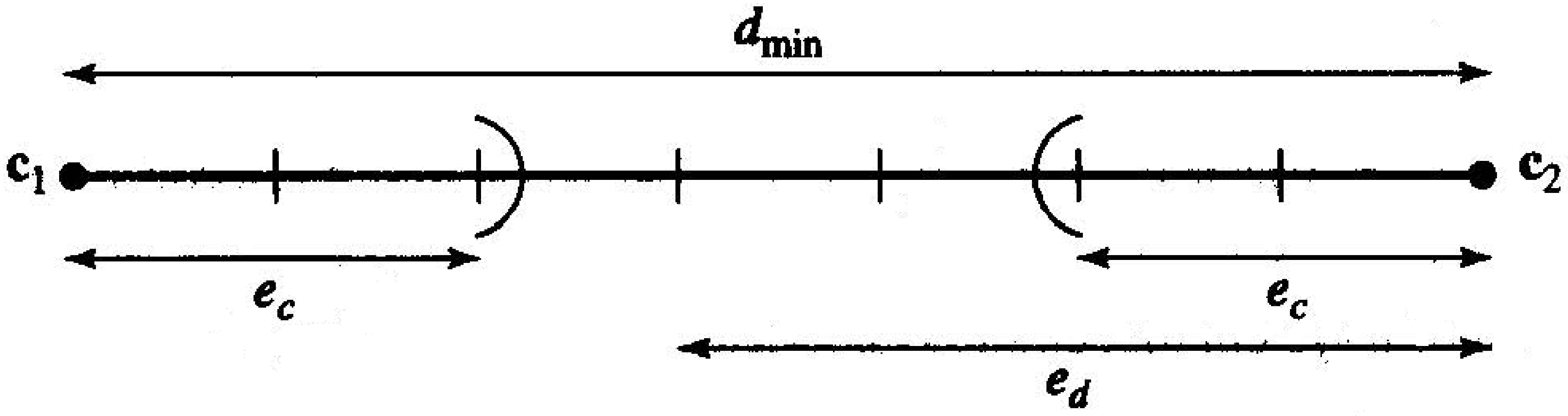


Figure 9.18 Relation between e_c, e_d and $d_{\min}$.

In Figure 9.18, we see that

$$e_c + e_d = d_{\min} - 1 \tag{9.5.37}$$

with the extra condition $e_c \le e_d$.

9.5.2 Burst-Error-Correcting-Codes

Most of the linear block codes are designed for correcting **random errors**, that is, errors that occur independently from the location of other channel errors.

Channel models including the additive white Gaussian noise channel can be modeled as channels with random errors.

However, the assumption of independently generated errors is not a valid in some other physical channels such as a fading channel.

In such a channel, if the channel is in deep fade, a large number of errors occur in sequence, that is, the errors are **bursty**.

In this channel, the probability of error at a certain location or time depends on whether its adjacent bits are received correctly or not.

Another example of a channel with burst error is a compact disc.

Any physical damage to a compact disc, such as a scratch, damages a sequence of bits and, therefore, the errors tend to occur in **bursts**.

Any random error-correcting code can be used to correct burst errors as long as the number of errors is less than half of the minimum distance of the code.

But there are more efficient coding schemes to correct burst errors such as Fire codes and Burton codes.

An effective method for correction of error bursts is to **interleave** the coded data such that the location of errors looks random and is distributed over many codewords rather than a few codewords.

In this way, the number of errors that occur in each block becomes small and can be corrected by using a random error correcting code.

At the receiver, a **deinterleaver** is employed to reverse the effect of the interleaver.

A block diagram of a coding system employing interleaving/deinterleaving is shown in Figure 9.19.

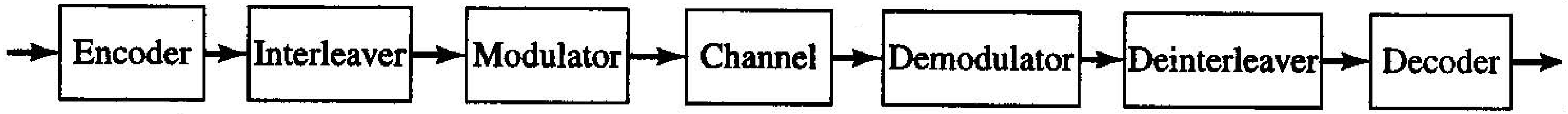


Figure 9.19 Block diagram of a system that employs interleaving/deinterleaving for burst-error-correction.

An **interleaver of depth** m reads m codewords of length n each and arranges them in a block with m rows and n columns.

Then, this block is read by column and the output is sent to the digital modulator.

At the receiver, the output of the detector is supplied to the deinterleaver, which generates the same $m \times n$ block structure and then reads by row and sends the output to the channel decoder as shown in Figure 9.20.

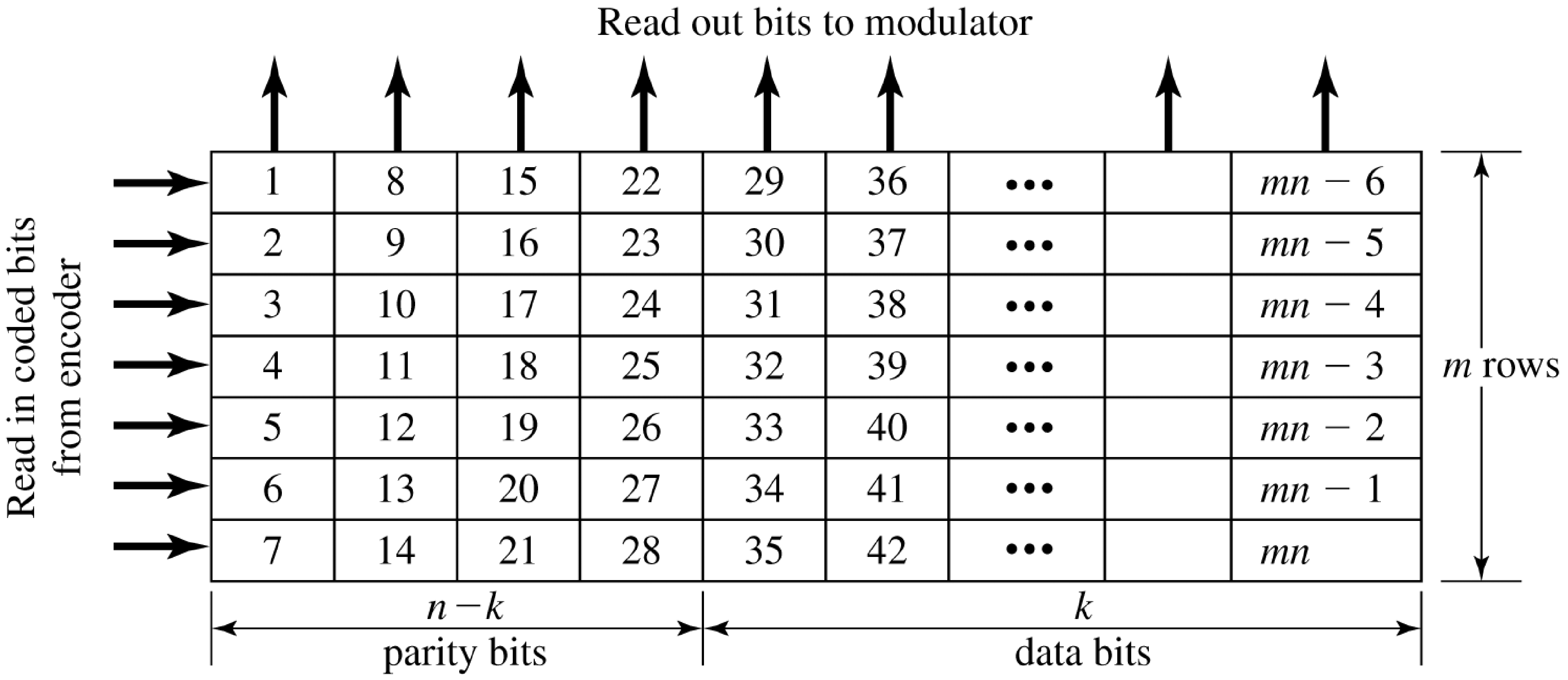

Read out bits to modulator
Read in coded bits from encoder
1 8 15 22 29 36 ••• mn − 6
2 9 16 23 30 37 ••• mn − 5
3 10 17 24 31 38 ••• mn − 4
4 11 18 25 32 39 ••• mn − 3
5 12 19 26 33 40 ••• mn − 2
6 13 20 27 34 41 ••• mn − 1
7 14 21 28 35 42 ••• mn
m rows
n − k
parity bits
k
data bits

Figure 9.20

A block interleaver for coded data.

Assume that $m = 8$ and the code in use is a $(15, 11)$ Hamming code capable of correcting one error/codeword.

Then, the block generated by the interleaver is a 8×15 block containing 120 binary symbols.

Any burst of errors of length 8 or less will result in at most one error/codeword and, therefore, can be corrected by the decoding of $(15, 11)$ Hamming code.

If interleaving/deinterleaving was not employed, an error burst of length 8 could possibly result in erroneous detection in 2 codewords (up to 22 information bits).

9.6 Cyclic Codes

Cyclic codes are a subset of linear block codes for which easily implementable encoders and decoders exists.

Definition 9.6.1.

A cyclic code is a linear block code with the extra condition that if $\mathbf{c}$ is a codeword, a cyclic shift[†] of it is also a codeword.

Ex.9.6.1

The code $\{000, 110, 101, 011\}$ is a cyclic code because it is easily verified to be linear and a cyclic shift of any codeword is also a codeword.

The code $\{000, 010, 101, 111\}$ is not cyclic because, although it is linear, a cyclic shift of 101 is not a codeword.

[†] A cyclic shift of the codeword $\mathbf{c} = (c_1, c_2, \cdots, c_{n-1}, c_n)$ is defined to be $\mathbf{c}^{(1)} = (c_2, c_3, \cdots, c_{n-1}, c_n, c_1)$.

9.6.1 Structure of Cyclic Codes

To study the properties of cyclic codes, it is easier to represent each codeword as a polynomial, called the codeword polynomial.

The codeword polynomial corresponding to $\mathbf{c} = (c_1, c_2, \cdots, c_{n-1}, c_n)$ is simply defined to be

$$c(p) = \sum_{i=1}^{n} c_i p^{n-1} = c_1 p^{n-1} + c_2 p^{n-2} + \cdots + c_{n-1} p + c_n \,. \tag{9.6.1}$$

The codeword polynomial of $\mathbf{c}^{(1)} = (c_2, c_3, \cdots, c_{n-2}, c_{n-1}, c_n, c_1)$, the cyclic shift of $\mathbf{c}$, is

$$c^{(1)}(p) = c_2 p^{n-1} + c_3 p^{n-2} + \cdots + c_{n-1} p^2 + c_n p + c_1 \tag{9.6.2}$$

which can be written as

$$c^{(1)}(p) = pc(p) + c_1(p^n + 1) \,. \tag{9.6.3}$$

Nothing that in the binary field addition and subtraction are equivalent, this reduces to

$$pc(p) = c^{(1)}(p) + c_1(p^n + 1) \tag{9.6.4}$$

or

$$c^{(1)}(p) = pc(p) \quad (\text{mod } (p^n + 1)) \,. \tag{9.6.5}$$

If we shift $\mathbf{c}^{(1)}$ once more the result will also be a codeword and its codeword polynomial will be

$$c^{(2)}(p) = pc^{(1)}(p) \quad (\text{mod } (p^n + 1)) \tag{9.6.6}$$

$$= p^2 c(p) \quad (\text{mod } (p^n + 1)) \tag{9.6.7}$$

where

$$a(p) = b(p) \quad (\text{mod } d(p))$$

means that $a(p)$ and $b(p)$ have the same remainder when divided by $d(p)$.

In general, for i shifts we have the codeword polynomial

$$c^{(i)}(p) = p^i c(p) \quad (\text{mod}(p^n + 1)). \tag{9.6.8}$$

For $i = n$, we have

$$c^{(n)}(p) = p^n c(p) \quad (\text{mod } (p^n + 1)) \tag{9.6.9}$$

$$= (p^n + 1)c(p) + c(p) \quad (\text{mod}(p^n + 1)) \tag{9.6.10}$$

$$= c(p) \tag{9.6.11}$$

which is obvious because shifting any codeword n times leaves it unchanged.

Theorem 9.6.1.

In any (n,k) cyclic code all codeword polynomials are multiples of a polynomial of degree $n-k$ of the form

$$g(p) = p^{n-k} + g_2 p^{n-k-1} + g_3 p^{n-k-2} + \cdots + g_{n-k} p + 1$$

called the **generator polynomial**, where $g(p)$ divides $p^n + 1$.

Furthermore, for any information sequence $\mathbf{x} = (x_1, x_2, \cdots, x_{k-1}, x_k)$, we have the **information sequence polynomial** $X(p)$ defined by

$$X(p) = x_1 p^{k-1} + x_2 p^{k-2} + \cdots + x_{k-1} p + x_k$$

and the codeword polynomial corresponding to $\mathbf{x}$ is given by $c(p) = X(p)g(p)$.

Any codeword polynomial is the product of the generator polynomial and the information polynomial.

That is, $\mathbf{c} = (c_1, c_2, \cdots, c_{n-1}, c_n)$ is the discrete convolution of the two sequences $\mathbf{x} = (x_1, x_2, \cdots, x_k)$ and $\mathbf{g} = (1, g_2, \cdots, g_{n-k}, 1)$.

9.7 Convolutional Codes

Convolutional codes are different from the block codes by the existence of memory in the encoding scheme.

In block codes, each block (or sequence) of k input bits is mapped into a block of length n of output bits by a rule defined by the code (for example by $\mathbf{G}$ or $g(p)$) and regardless of the previous inputs to the encoder.

The rate of a block code is given by

$$R_c = \frac{k}{n}. \tag{9.7.1}$$

In convolutional codes, each block of k bits is again mapped into a block of n bits to be transmitted over the channel, but these n bits are not only determined by the present k-information bits but also by the previous information bits.

The block diagram of a convolutional encoder is given in Figure 9.24.

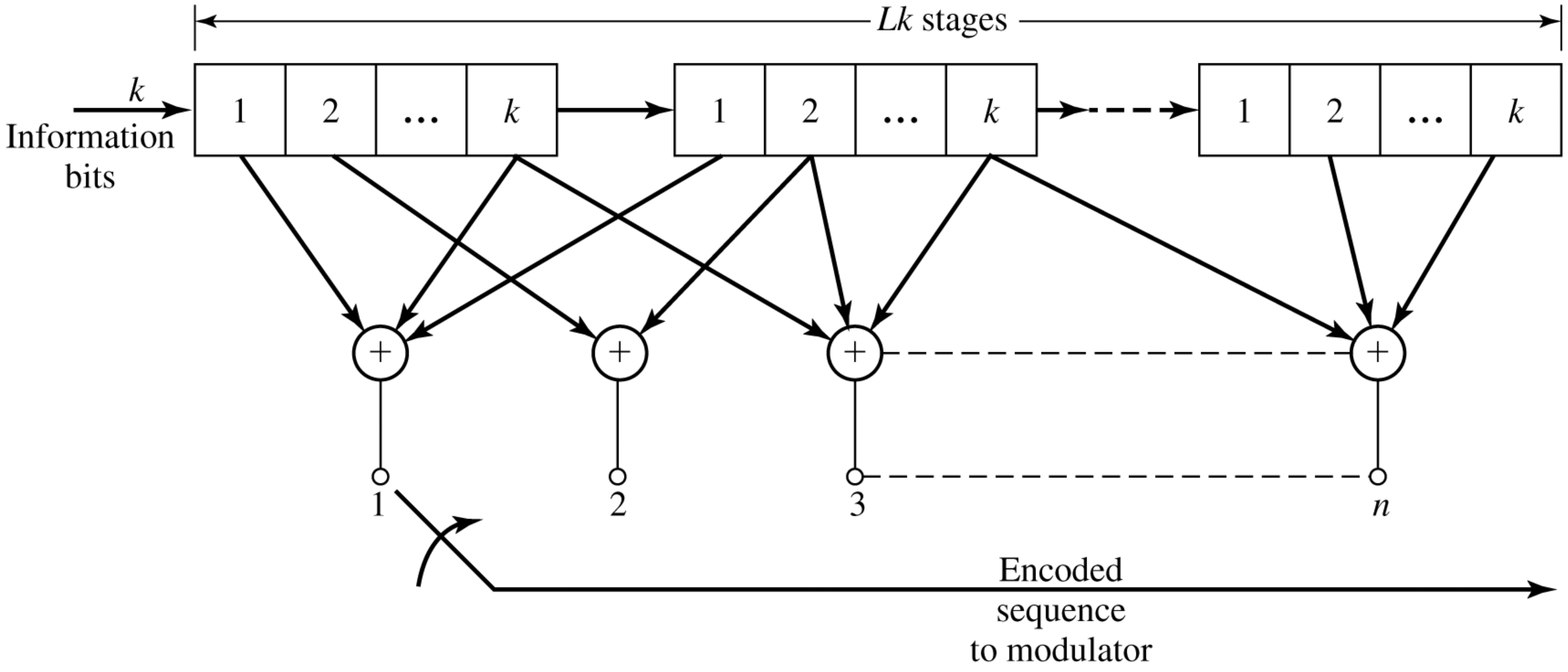


Figure 9.24

The block diagram of a convolutional encoder.

The convolutional encoder consists of a shift register with kL stages where L is called the **constraint length** of the code.

At each instant of time, k-information bits enter the shift register and the contents of the last k stages of the shift register are dropped.

After the k bits have entered the shift register, n-linear combinations of the contents of the shift register, as shown in Figure 9.24, are computed to generate the encoded sequence.

The n-encoder outputs not only depend on the most recent k bits that have entered the encoder but also on the $(L-1)k$ contents of the first $(L-1)k$ stages of the shift register before the k bits arrived.

Therefore, the shift register is a finite state machine with $2^{(L-1)k}$ states.

Because for each k-input bits, we have n-output bits, the rate of this code is simply given by

$$R_c = \frac{k}{n}. \tag{9.7.2}$$

Ex.9.7.1

An encoder for a convolutional code is shown in Figure 9.25.

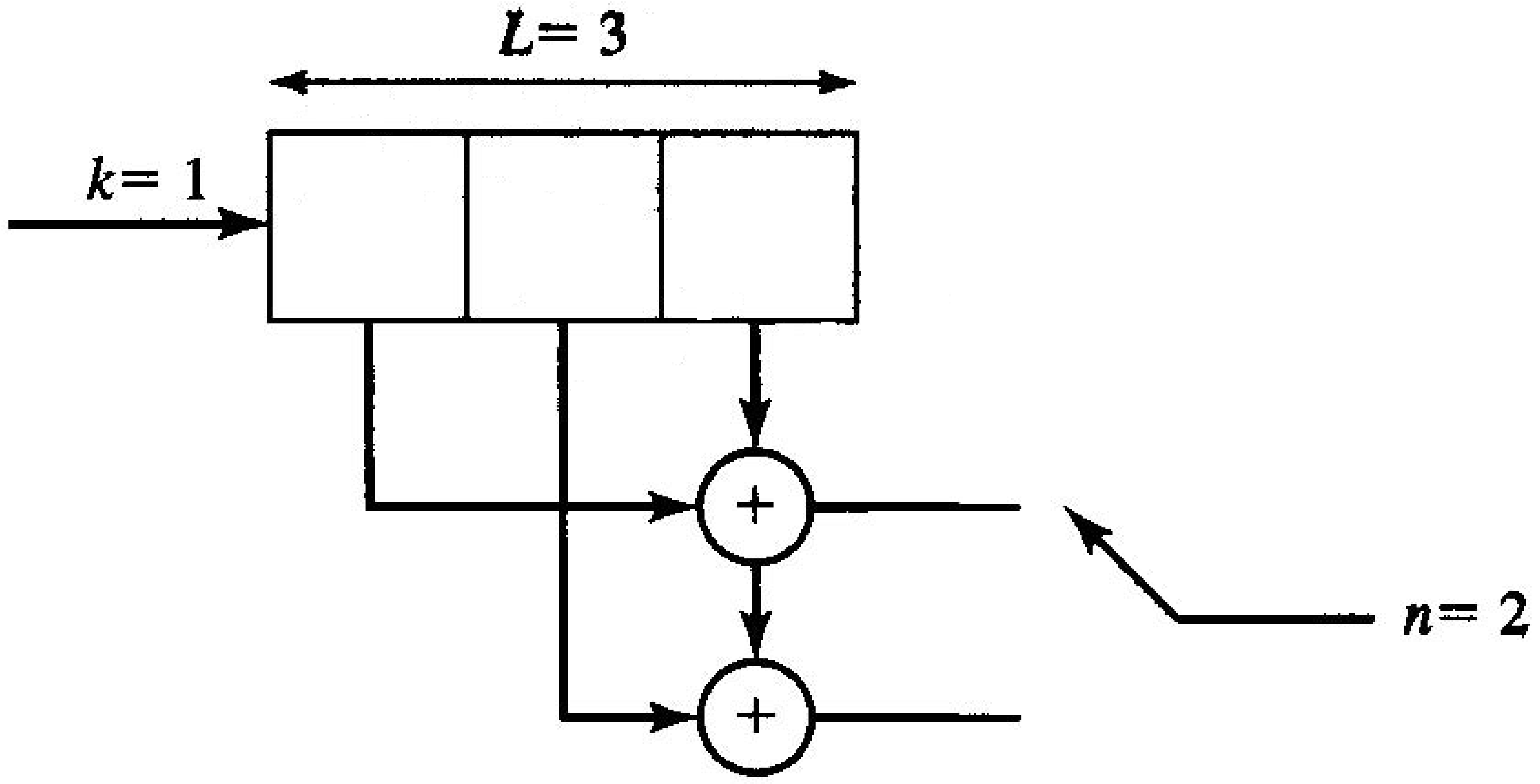


Figure 9.25 Rate $\frac{1}{2}$ convolutional encoder.

For the encoder in Figure 9.25, $k=1$, $n=2$, and $L=3$.

The rate of the code is $R_c = \frac{k}{n} = \frac{1}{2}$ and the number of states is $2^{(L-1)k} = 2^{(3-1)\cdot 1} = 4$.

One way to describe a convolutional code (other than drawing the encoder) is to specify how the two output bits of the encoder depend on the contents of the shift register by n vectors $\mathbf{g}_1, \mathbf{g}_2, \cdots, \mathbf{g}_n$, known as **generator sequences** of the convolutional code.

The i th, $1 \le i \le 2^{kL}$, component of $\mathbf{g}_j$, $1 \le j \le n$, is 1 if the i th stage of the shift register is connected to the combiner corresponding to the j th bit in the output and 0 otherwise.

The generator sequences are given by

$\mathbf{g}_1 = [1 \quad 0 \quad 1]$

$\mathbf{g}_2 = [1 \quad 1 \quad 1]$.

9.7.1 Basic Properties of Convolutional Codes

Because a convolutional encoder has finite memory, it can be represented by a **state-transition diagram**.

In the state-transition diagram, each state of the convolutional encoder is represented by a node and transitions between states are denoted by branches connecting these nodes.

The number of branches emerging from each state is equal to 2^k.

Figure 9.26 shows the state transition diagram for the convolutional code in Figure 9.25.

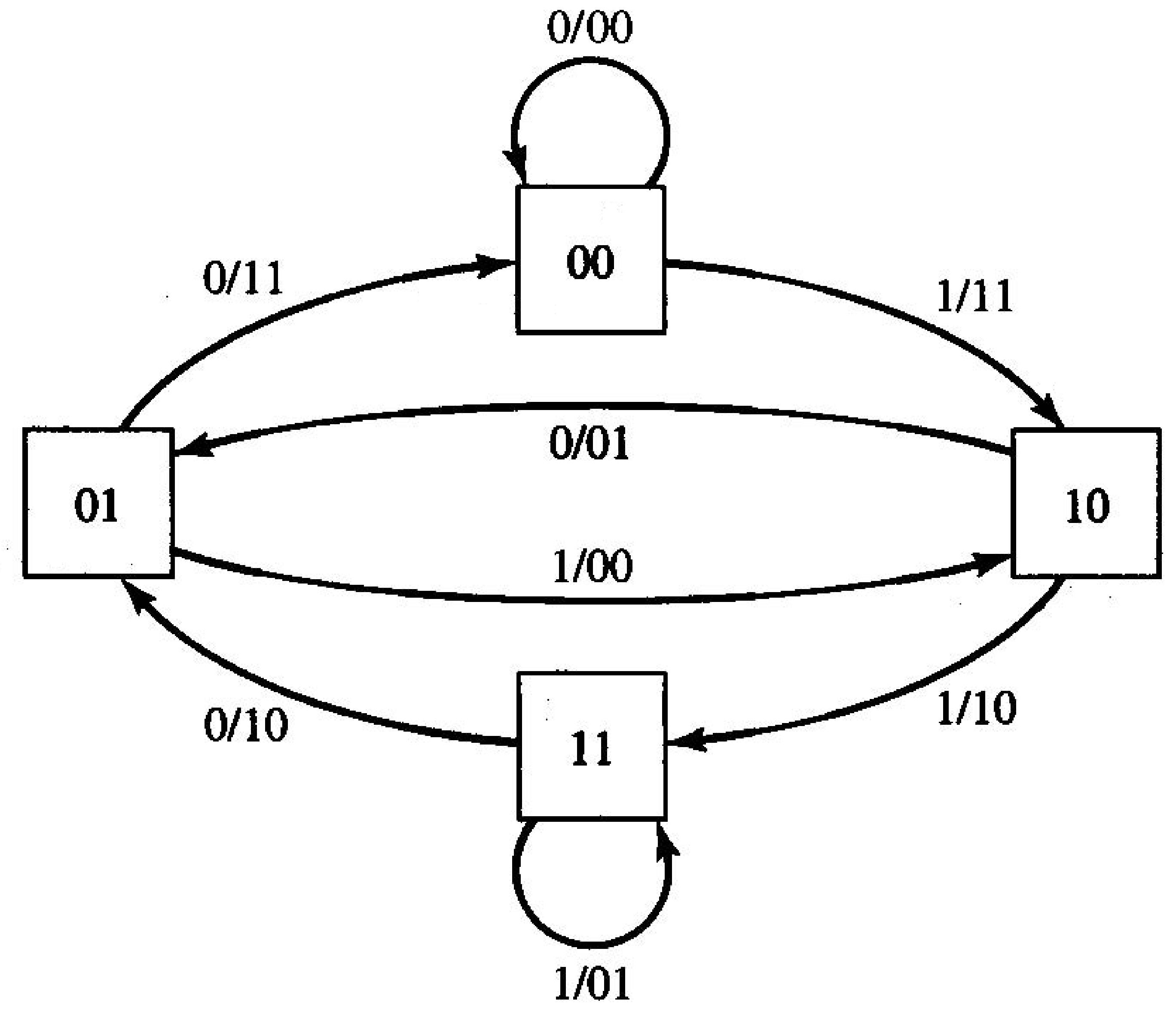


Figure 9.26 State transition diagram for the encoder in Figure 9.25.

A second method to describe a convolutional code is to specify its **trellis diagram**.

The trellis diagram is obtained by specifying all states on a vertical axis and repeating this vertical axis along the time on a horizontal axis.

Then, each transition from a state to another state is denoted by a branches connecting the two states on two adjacent vertical axes.

As was the case with the state transition diagram, we have 2^k branches of the trellis leaving each state and 2^k branches merging at each state.

Figure 9.27 shows the trellis diagram for the code described by the encoder in Figure 9.25.

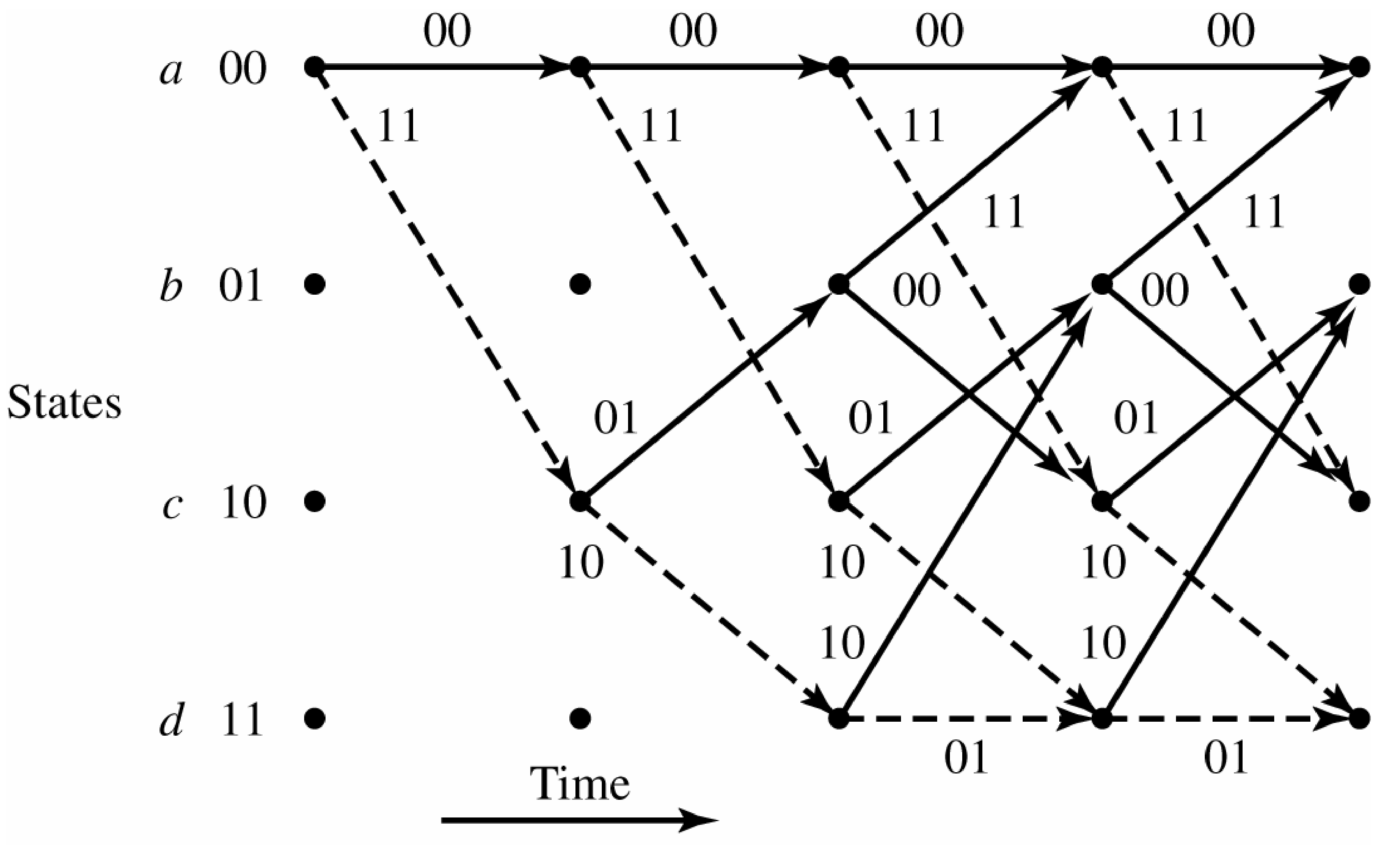


Figure 9.27

Trellis diagram for the encoder of Figure 9.25.

Encoding

Assume that the encoder, before the first information bit enters it, is loaded with zeros (that is, the encoder is in the all-zero state).

The information bits enter the encoder k bits at a time and the corresponding n-output bits are transmitted over the channel.

This procedure is continued until the last group of k bits is loaded into the encoder and the corresponding n-outputs are transmitted over the channel.

Assume that after the last group of k bits, another group of $k(L-1)$ enters encoder so that the corresponding n outputs are transmitted over the channel. This returns the encoder to the all-zero state to make it ready for next transmission.

Ex.9.7.2

In the convolutional code shown in Figure 9.25, what is the encoded sequence corresponding to the information sequence $\mathbf{x} = (1101011)$?

Solution

After transmission of the last information bit, two 0 bits are transmitted which means that the transmitted sequence is $\mathbf{x}_1 = (110101100)$.

We have the codeword $\mathbf{c} = (111010000100101011)$.

Transfer Function

Any codeword of a convolutional encoder corresponds to a path through the trellis that starts from the all-zero state and returns to the all-zero state.

To obtain the transfer function of a convolutional code, we split the all-zero state into two states, one denoting the starting state and one denoting the first return to the all-zero state.

All other states are denoted as in-between states.

Corresponding to each branch connecting two states, a function of the form $D^{\alpha} N^{\beta} J$ is defined where α denotes the number of ones in the output bit sequence for that branch and β is the number of ones in the corresponding input sequence for that branch.

The **transfer function** of the convolutional code is, then, the transfer function of the flow graph between the starting all-zero state and the final all-zero state, and will be a function of the tree parameters D, N, J and denoted by $T(D, N, J)$.

Each element of $T(D, N, J)$ corresponds to a path through the trellis starting from zero the all-zero state and ending at the all-zero state.

The exponent of J indicates the number of branches spanned by that path, the exponent of D shows the number of ones in the codeword corresponding to that path (or equivalently the Hamming weight if the codeword), and finally, the exponent of N indicates the number of ones in the input information sequence.

Since $T(D, N, J)$ indicates the properties of all paths through the trellis starting from the all-zero path and returning to it **for the first time**, then, in deriving it, any self loop at the all-zero state is ignored.

Ex.9.7.3

Find the transfer function of the convolutional code of Figure 9.25.

Solution

Figure 9.28 shows the diagram used to find the transfer function of this code.

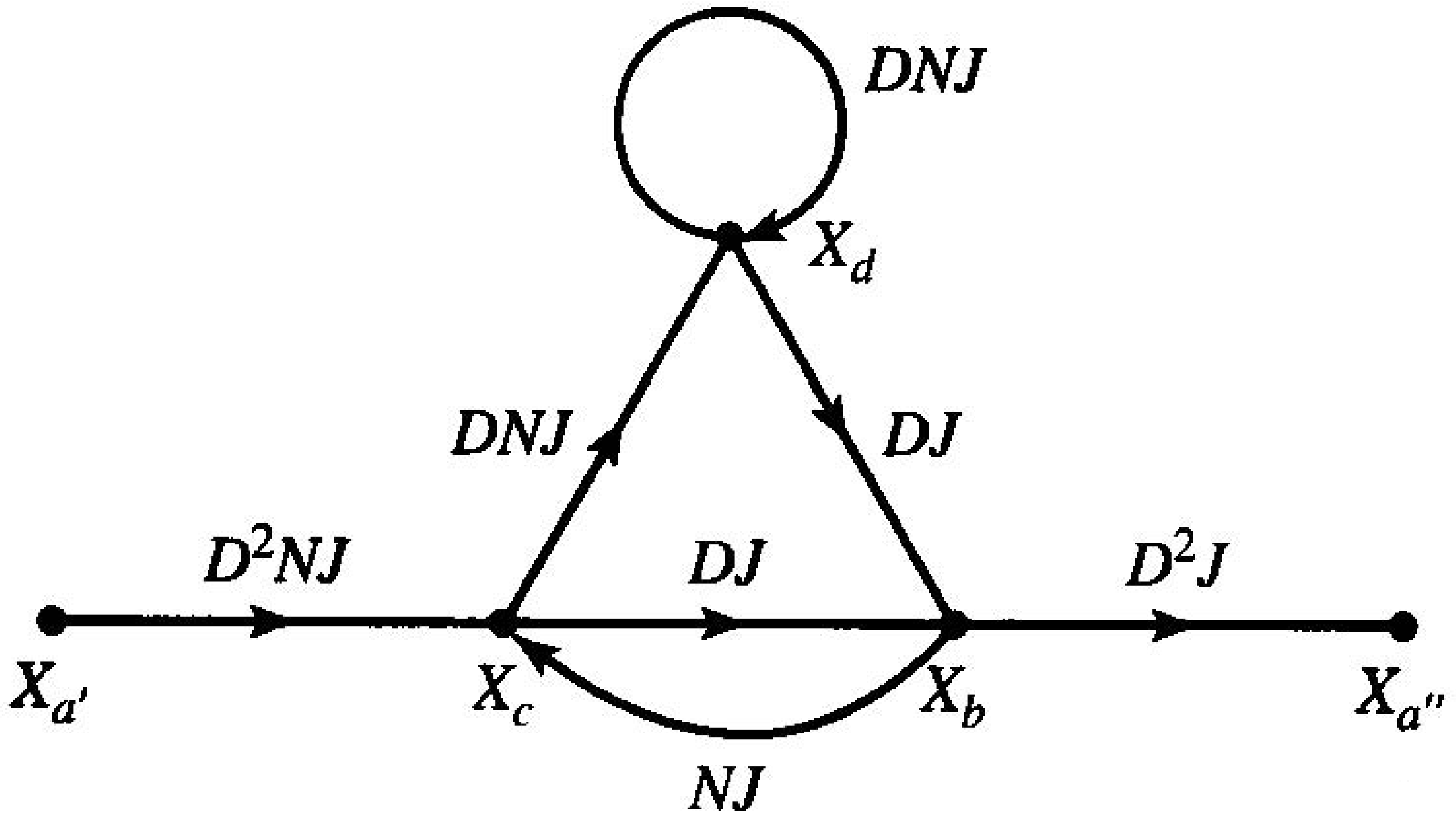


Figure 9.28 Flow graph for finding the transfer function.

The code has a total of four states denoted by the contents of the first two stages of the shift register.

We denote these states by the following letters

$00 \to a$

$01 \to b$

$10 \to c$

$11 \to d.$

As seen in the figure, state a is split into states a' and a'' denoting the starting and returning state.

We can write

$$X_c = X_{a'} D^2 NJ + NJX_b$$

$$X_b = DJX_d + DJX_b$$

$$X_d = DNJX_c + DNJX_d$$

$$X_{a''} = D^2 JX_b.$$

Eliminating X_b, X_c, and X_d results in

$$\begin{aligned} T(D,N,J) &= \frac{X_{a''}}{X_{a'}} \\ &= \frac{D^5NJ^3}{1-DNJ-DNJ^2}. \end{aligned} \tag{9.7.3}$$

Expanding $T(D,N,J)$ in a polynomial form, we obtain

$$T(D,N,J) = D^5NJ^3 + D^6N^2J^4 + D^6N^2J^5 + D^7N^3J^5 + \cdots. \tag{9.7.4}$$

The term D^5NJ^3 indicates that there exists a path through the trellis starting from the all-zero state and returning to the all-zero state for the first time, which spans three branches, corresponding to an input-information sequence containing one 1 (and, therefore, two 0's), and the codeword for this path has Hamming weight equal to 5.

This oath is indicated with bole lines in the Figure 9.29.

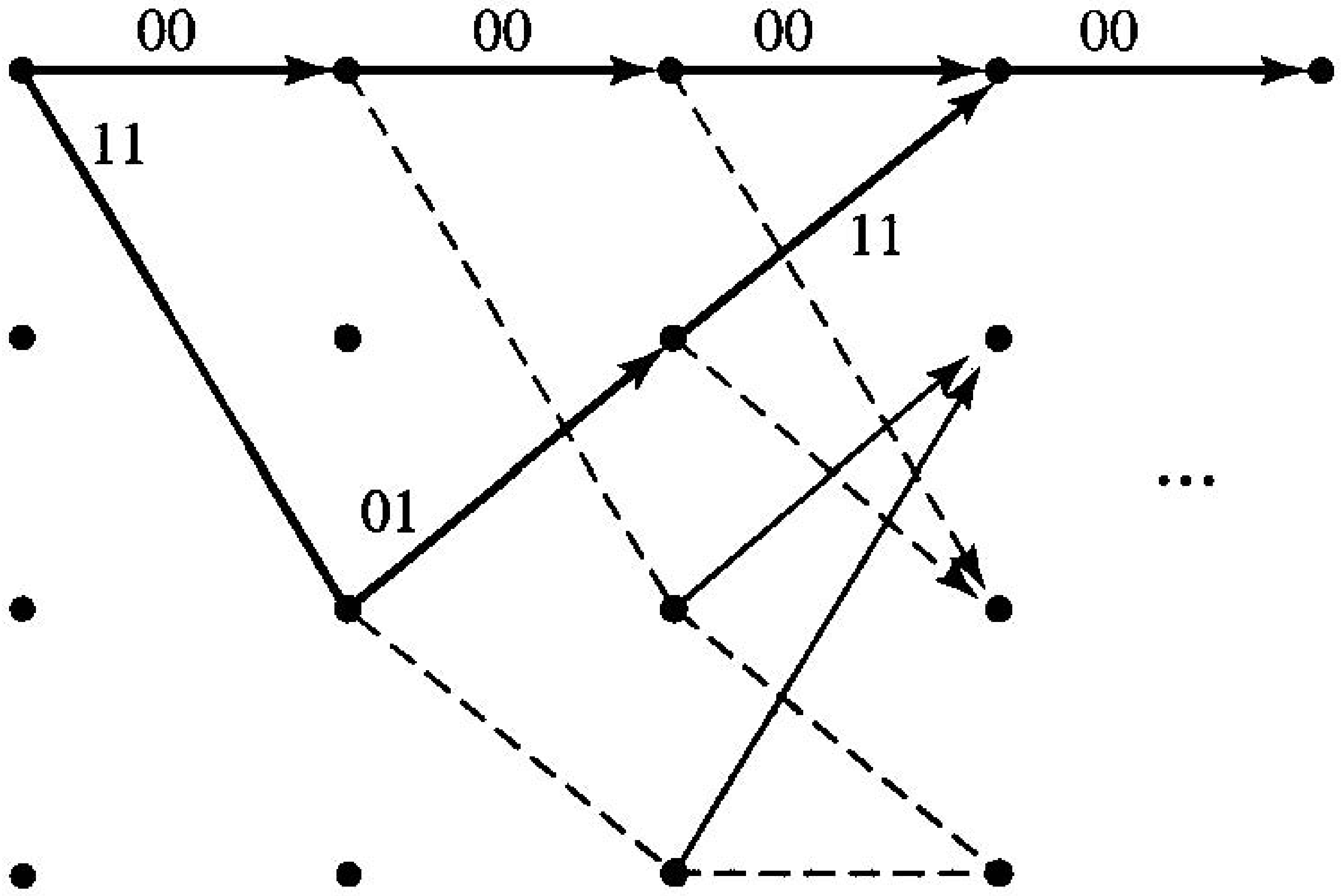


Figure 9.29 Path corresponding to D^5NJ^3 in code represented in Figure 9.25.

This path is somewhat similar to the minimum-weight codeword on block codes. In fact, this path corresponds to the codeword that is equal to the minimum power of D in the expansion of $T(D, N, J)$ is called the **free distance of the code** and denoted by d_{free}.

The free distance of the above code is equal to 5.

The general form of the transfer function is, therefore, given by

$$T(D, N, J) = \sum_{d=d_{\text{free}}}^{\infty} a_d D^d N^{f(d)} J^{g(d)}. \tag{9.7.5}$$

A shorter form of transfer function, which only provides information about the weight of the codewords, can be obtained from $T(D, N, J)$ by setting $N = J = 1$.

This shorter form is given by

$$T_1(D) = \sum_{d=d_{\text{free}}}^{\infty} a_d D^d \tag{9.7.6}$$

which will be used in deriving bounds on the error probabilities of the convolutional codes.

Ex.9.7.4

For the code of Figure 9.25 we have

$$T_1(D) = \left.\frac{D^5 N J^3}{1 - DNJ - DNJ^2}\right|_{N=J=1}$$

$$= \frac{D^5}{1-2D}$$

$$= D^5 + 2D^6 + 4D^7 + \cdots$$

$$= \sum_{i=0}^{\infty} 2^i D^{5+i} .$$

Catastrophic Codes

The purpose of coding is to provide higher levels of protection against channel noise.

A code that maps information sequences that are far apart into codewords that are not far apart into codewords that are not far apart is not a good code since these two codewords can be mistaken rather easily and the result would be a large number of bit errors in the information stream.

A limiting case of this undesirable property happens when two information sequences that are different in infinitely many positions are mapped into codewords that differ only in a finite number of positions.

In such a case, since the codewords differ in a finite number of bits, there always exists the probability that they will be erroneously decoded and this, in turn, results in an infinite number of errors in detecting the input information sequence.

Codes that exhibit this property are called **catastrophic codes** and should be avoided in practice.

As an example of a catastrophic code, let us consider the $(2,1)$ code described by

$\mathbf{g}_1 = [1 \quad 1 \quad 0]$

$\mathbf{g}_2 = [0 \quad 1 \quad 1]$.

The encoder and the state-transition diagram for this code are given in Figure 9.30.

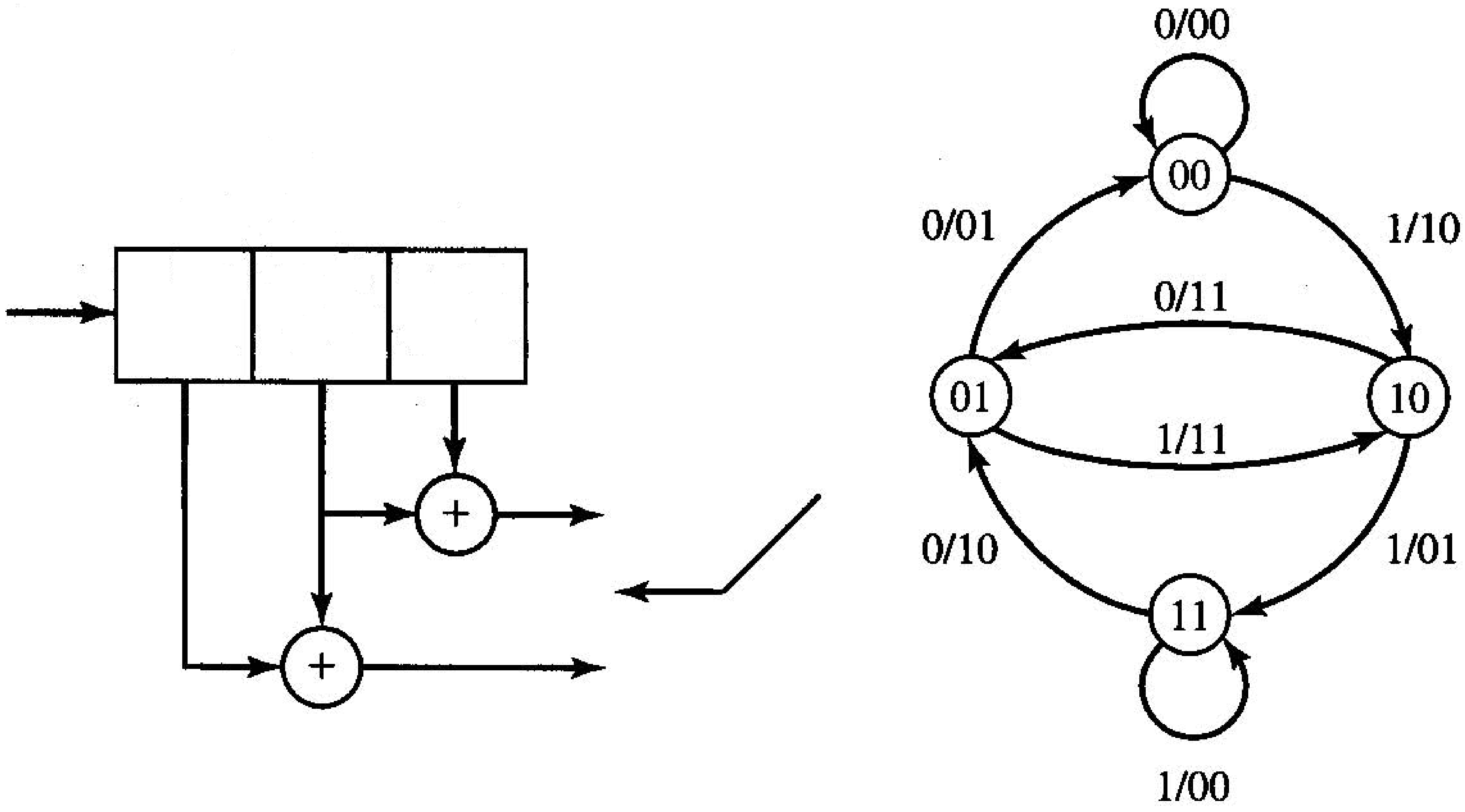


Figure 9.30 Encoder and the state-transition diagram for a catastrophic code.

There exists a self loop in state “11” that corresponds to a “1” input to the encoder and the corresponding output consists of all zeros.

Therefore, if an input-information stream consists of all ones, the corresponding output will be

$\mathbf{c} = (10010000\cdots0001001)$.

Comparing this codeword to the codeword corresponding to the all-zero information sequence

$\mathbf{c}_0 = (0000\cdots000)$,

we observe that, although the information sequences are different in a large number of positions, the corresponding output sequences are quite close (the Hamming distance being only 4) and they can be mistaken very easily.

The existence of such a self loop corresponding to k inputs which are not all zeros and for which the n output bits are all zeros, shows that a code is catastrophic and should be avoided.

9.7.2 Optimum Decoding of Convolutional Codes—Viterbi Algorithm

In hard-decision decoding of convolutional codes, we want to choose a path through the trellis whose codeword, denoted by $\mathbf{c}$, is at minimum Hamming distance from the quantized received sequence $\mathbf{y}$.

In hard-decision decoding, the channel is binary memoryless (the fact that the channel is memoryless follows from the fact that the channel noise is assumed to be white).

Because the desired path starts from the all-zero state and returns back to the all-zero state, we assume that this path spans a total of m branches, and since each branch corresponds to n bits of the encoder output, the total number of bits in $\mathbf{c}$, and also in $\mathbf{y}$, is mn.

Denote the sequence of bits corresponding to the i th branch by $\mathbf{c}_i$ and $\mathbf{y}_i$, respectively, where $1 \le i \le m$, and each $\mathbf{c}_i$ and $\mathbf{y}_i$ is of length n.

Then, the Hamming distance between $\mathbf{c}$ and $\mathbf{y}$ is given by

$$d(\mathbf{c},\mathbf{y}) = \sum_{i=1}^{m} d(\mathbf{c}_i,\mathbf{y}_i) . \tag{9.7.7}$$

In soft-decision decoding, we have a similar situation with three differences.

1. Instead of $\mathbf{y}$, we are dealing directly with the vector $\mathbf{r}$, the vector output of the optimal (matched-filter type or correlators-type) digital demodulator.

2. Instead of the binary 0, 1 sequence $\mathbf{c}$, we are dealing with the corresponding sequence $\mathbf{c}'$ with

$$c_{ij}' = \begin{cases} \sqrt{\varepsilon}, & c_{ij} = 1, \\ -\sqrt{\varepsilon}, & c_{ij} = 0, \end{cases}$$

for $1 \le i \le m$ and $1 \le j \le n$.

3. Instead of Hamming distance, we are using Euclidean distance because the channel is an additive white Gaussian noise channel.

Then, we have

$$d_{\mathrm{E}}^2(\mathbf{c}', \mathbf{r}) = \sum_{i=1}^{m} d_{\mathrm{E}}^2(\mathbf{c}_i', \mathbf{r}_i)\,. \tag{9.7.8}$$

From (9.7.7) and (9.7.8), the problem we have to solve is stated as follows.

Given a vector (or sequence) $\mathbf{a}$, find a **path** through the trellis, starting at the all-zero state $S_1 = 0$ and ending at the all-zero state $S_m = 0$, such that some distance measure between the sequence $\mathbf{a}$ and a sequence $\mathbf{b}$ corresponding to that path is minimized.[†]

From (9.7.7) and (9.7.8), it is shown that the distance between $\mathbf{a}$ and $\mathbf{b}$ in both hard- and soft-decision decoding can be written as the sum of distance corresponding to individual **branche**s of the path.

Assume that the problem is formulated as minimizing a metric μ given by

$$\mu(\mathbf{a},\mathbf{b}) = \sum_{i=1}^{m} \mu(\mathbf{a}_i,\mathbf{b}_i).$$

First, observe that if the path $(S_1 = 0, S_i = l, S_m = 0)$, $1 \le i \le m$, is the optimal path, where $0 \le l \le M-1$ (where $M = 2^{(L-1)k}$ is the number of states), then the metric contribution of the path $(S_1 = 0, S_i = l)$ is lower than the metric contribution of any other path $\overline{(S_1 = 0, S_i = l)}$ connecting $S_1 = 0$ to $S_i = l$.

Otherwise, the path consisting of $\overline{(S_1 = 0, S_i = l)}$ and $(S_i = l, S_m = 0)$ would be the optimal path.

This is shown in Figure 9.31.

[†] The problem can also be formulated as a maximization problem. For example, instead of minimizing the Euclidean distance, one could maximize the correlation.

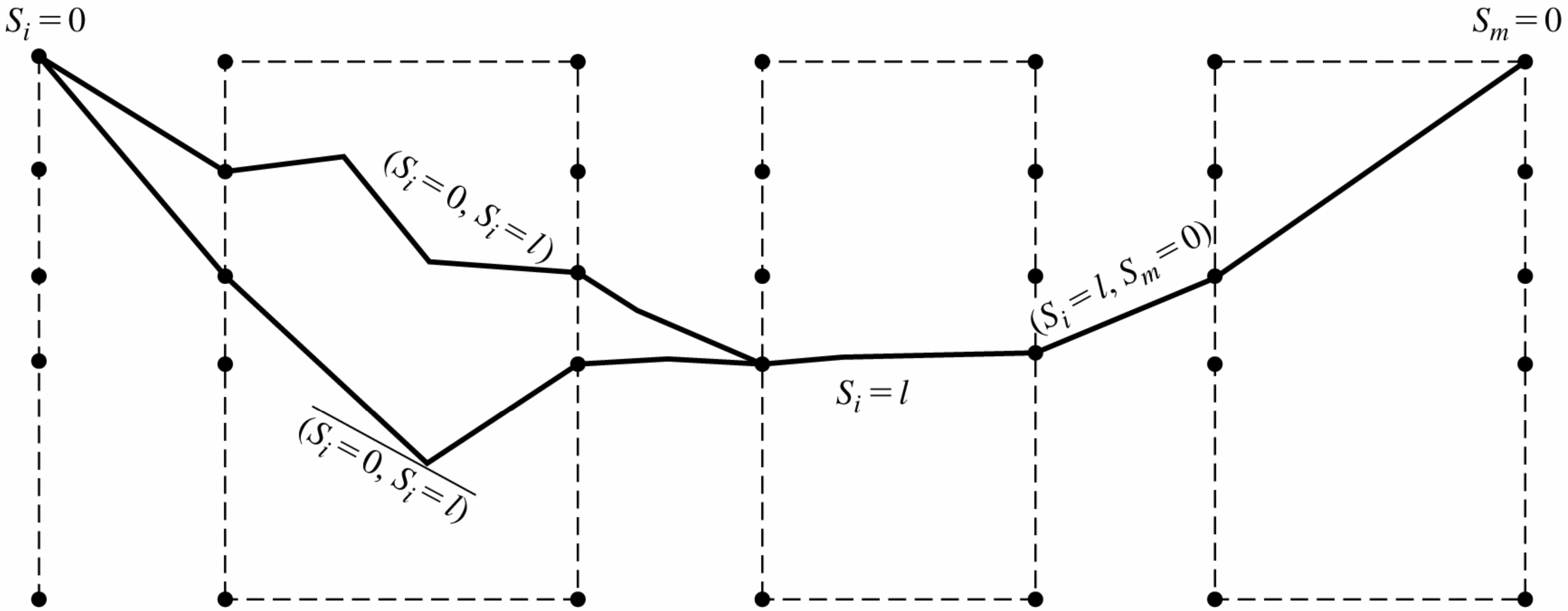


Figure 9.31

Comparison of the optimal path, $(S_1 = 0, S_i = l, S_m = 0)$, with a suboptimal path consisting of the concatenation of $\overline{(S_1 = 0, S_i = l)}$ and $(S_i = l, S_m = 0)$.

Let Λ_{i-1} denote the set of 2^k states that are connected with a branch to $S_i = l$.

If the path from $S_1 = 0$ to $S_i = l$ is the optimal path between these two states, then this path must be the concatenation of an optimal path connecting $S_1 = 0$ to a state $S_{i-1} = \lambda$ for some $\lambda \in \Lambda_{i-1}$ and the branch connecting $S_{i-1} = \lambda$ to $S_i = l$.

The optimal path connectings $S_1 = 0$ to $S_{i-1} = \lambda$ is called a **survivor path** at state $S_{i-1} = \lambda$, or simply a survivor.

Therefore, in order to find the survivor path at state $S_i = l$, it is sufficient to have the survivors (and their metrics) for all $S_{i-1} = \lambda$, $\lambda \in \Lambda_{i-1}$, append them with the branches connecting elements of Λ_{i-1} to $S_i = l$, find the metric of the resulting paths from $S_1 = 0$ to $S_i = l$, and pick the one with minimum metric; this will be the new survivor at $S_i = l$.

This process is started at $S_1 = 0$ and is finished at $S_m = 0$; the final survivor at $S_m = 0$ is the optimal path and the best (maximum-likelihood) match to the received sequence as shown in Figure 9.32.

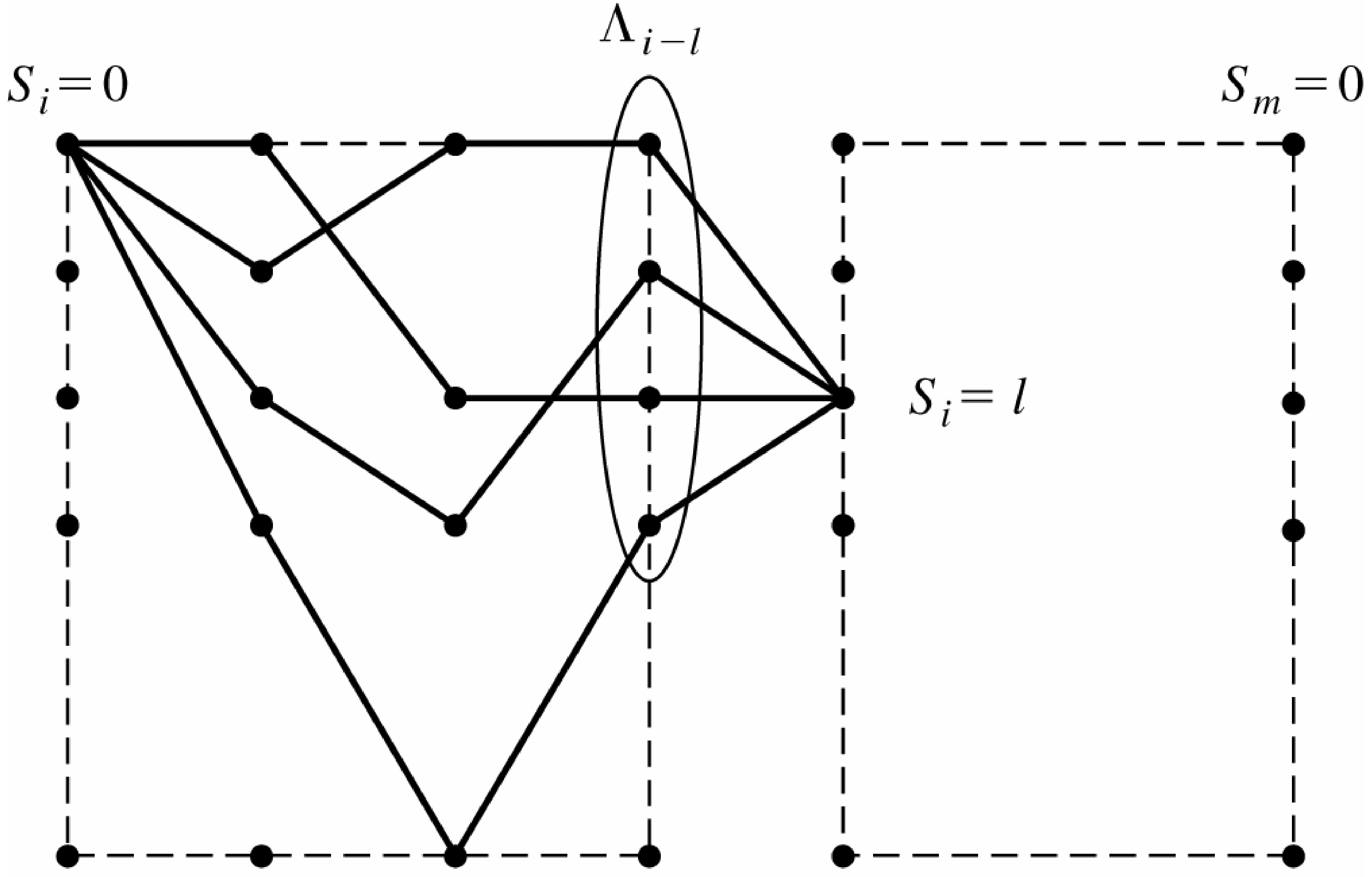


Figure 9.32

Finding a new survivor from old survivors.

At each step, the new survivor metric is given by

$$\mu(S_1=0,S_i=l)=\min_{\lambda\in\Lambda_{i-1}}\{\mu(S_1=0,S_{i-1}=\lambda)+\mu(S_{i-1}=\lambda,S_i=l)\}. \qquad (9.7.9)$$

Having found the survivor metric, the new survivor at $S_i=l$ is the path $(S_1=0,S_{i-1}=\lambda,S_i=l)$ for the λ that minimizes the survivor matric in (9.7.9).

The above procedure can be summarized with the Viterbi algorithm.

1. Parse the received sequence into m subsequences each of length n.
2. Draw a trellis of depth m for the code under study.

   For the last $L-1$ stages of the trellis, draw only paths corresponding to the all-zero input sequences (because we know that the input sequence has been padded with $k(L-1)$ zeros).
3. Set $i=1$, and set the metric of the initial all-zero state equal to zero.
4. Find the distance of the i th stage states to the $(i+1)$ st-stage states of the trellis.
5. Add these distances to the metrics of the i th stage states to obtain the metric candidates for the $(i+1)$ st-stage states.

   For each state of the $(i+1)$ st stage, there are 2^k metric candidates, each corresponding to one branch ending at that state.

6. For each state at the $(i+1)$st stage, choose the minimum of the metric candidates, label the branch corresponding th this minimum value as the survivor, and assign the minimum of the metric candidates as the metrics of the $(i+1)$st-stage states.
7. If $i=m$, go to step 8, otherwise increase i by 1 and go to step 4.
8. Starting with the all-zero state at the final stage, go back through the trellis along the survivors to reach the initial all-zero state.

   This path is the optimal path and the input-bit sequence corresponding to it is the ML decoded information sequence.

To obtain the best guess about the input bit sequence, remove the last $k(L-1)$ zeros from this sequence.

In the above algorithm, the decoding delay and the amount of memory required for decoding a long information sequence is unacceptable.

The decoding cannot be started until the whole sequence is received, and all surviving paths have to be stored.

A suboptimal solution that does not cause problems is desirable.

One such approach, referred to as **path-memory truncation**, is that the decoder at each stage only searches δ stages back in the trellis instead of searching back to the start of the trellis.

With this approach at the $(\delta+1)$th stage, the decoder makes a decision on the input bits corresponding to the first stage of the trellis (the first k bits) and future received bits do not change this decision.

This means that the decoding delay will be $k\delta$ bits and it is only required to keep the surviving paths corresponding to the last δ stages.

Computer simulation have shown that if $\delta \approx 5L$, the degradation in performance due to path-memory truncation is negligible.

Ex.9.7.5

Assume that in hard-decision decoding.

Suppose the quantized received sequence is given by

$\mathbf{y} = (0110111101001)$.

Suppose that the convolutional code shown in Figure 9.25 is used.

Find the ML information sequence and the number of errors.

Solution

The code is a $(2,1)$ code with $L=3$.

The length of the received sequence $\mathbf{y}$ is 14 which means that $m=7$ and we have to draw a trellis of depth 7.

Because the input-information sequence is padded with $k(L-1)=2$ zeros, for the final two stages of the trellis we will only draw the branches corresponding to all zero inputs.

This means that the actual length of the input sequence is 5, which, after padding with 2 zeros, has increased to 7.

The trellis diagram is shown in Figure 9.33.

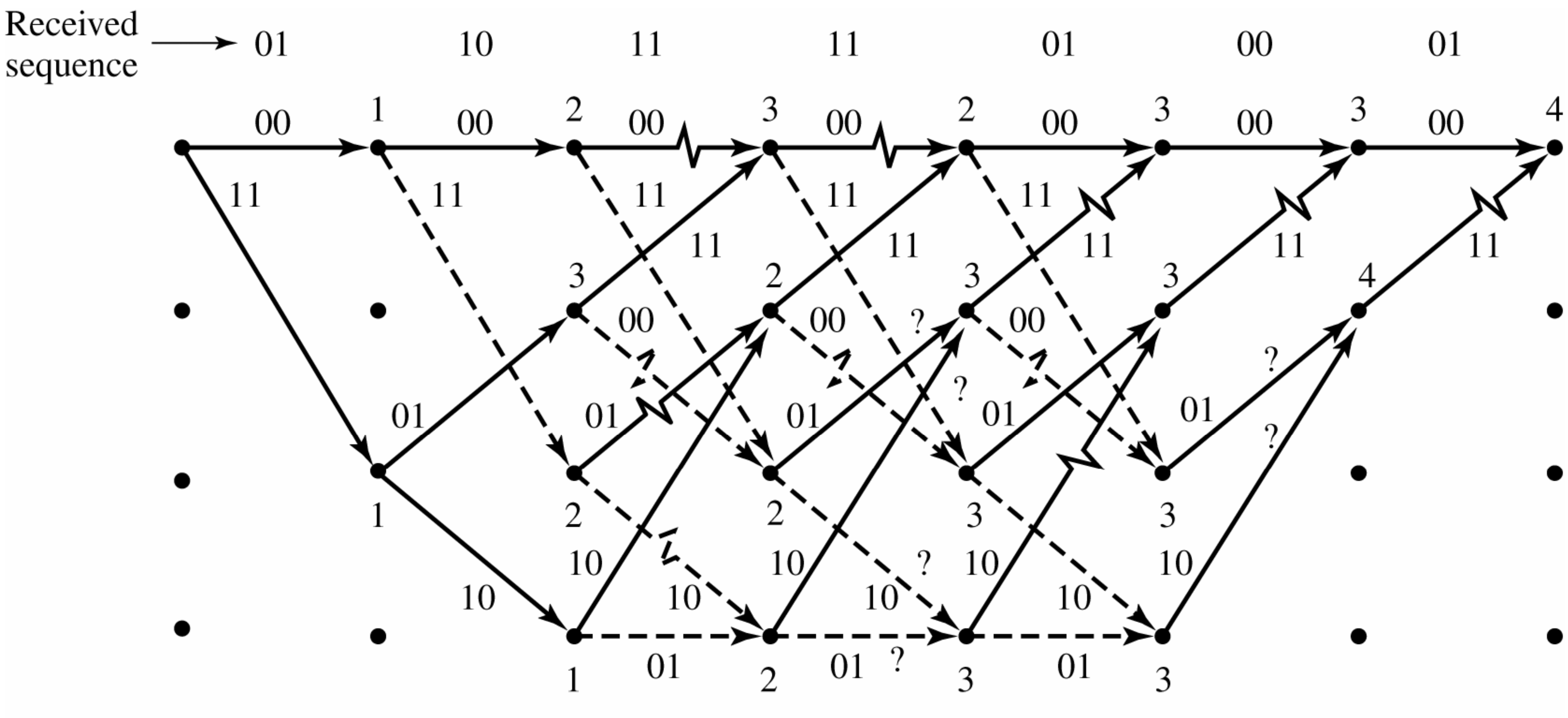


Figure 9.33

The trellis diagram for Viterbi decoding of the sequence (01101111010001).

In the trellis in the last two stages, we have only considered the zero inputs to the encoder.

The metric of the initial all-zero state is set to zero and the metrics of the next stage are computed.

In the third stage, a comparison has to be made here and survivors are to be chosen.

From the two branches that enter each state, one which corresponds to the least total accumulated metric remains as a survivor and the other branches are deleted.

If at any stage, two paths result in the same metric, each one of them can be a survivor which is marked by a “?” in the trellis diagram.

The procedure is continued to the final all-zero state of the trellis and then starting from that state, we move along the surviving paths to the initial all-zero state.

The input bit sequence corresponding to this path is 1100000 where the last two zeros are not information bits but were added to return the encoder to the all-zero state.

Therefore, the information sequence is 11000.

The corresponding codeword for the selected path is 11101011000000, which is at Hamming distance 4 from the received sequence.

No other path through the trellis is at a Hamming distance less than 4 from the received $\mathbf{y}$.

For soft-decision decoding a similar procedure is followed with squared Euclidean distance substituted for Hamming distance.

9.7.3 Other Decoding Algorithms for Convolutional Codes

The Viterbi algorithm provides maximum-likelihood decoding for convolutional codes.

However, as the complexity of the algorithm is proportional to the number of states in the trellis diagram, the complexity increase exponentially with the constraint length of the convolutional codes.

Therefore, the Viterbi algorithm can be allied only to codes with low-constraint lengths.

For higher constraint-length codes, other suboptimal decoding schemes have been proposed, such as the sequential decoding of Wozencraft (1957); the Fano algorithm (1963), the stack algorithm (Zigangirov (1966) and Jelinek (1969)); the feedback-decoding algorithm (Heller (1975)); and majority logic decoding (Massey (1963)).

9.7.4 Bounds on Error Probability of Convolutional Codes

Finding bounds on the error performance of convolutional codes is different from the method used to find error bounds for block codes, because, here we are dealing with sequences of very large length and, since the free distance of these codes is usually small, some errors will eventually occur.

The number of errors is a random variable that depends both on the channel characteristics (SNR in soft-decision decoding and crossover probability in hard-decision decoding) and the length of the input sequence.

The longer the input sequence, the higher the probability of making errors. Therefore, it makes sense to normalize the number of bit errors to the length of the input sequence, that is, the expected number of bits received in error/input bit.

To find a bound on the average number of bits in error for each input bit, we first derive a bound on the average number of bits in error for ach input sequence of length k.

Assume that the all-zeros sequence is transmitted† and, up to stage l in the decoding, there has been no error.

† Because of the linearity of convolutional codes we can, without loss of generality, make this assumption.

Now k-information bits enter the encoder and result in moving to the next stage in the trellis.

Because we are assuming that up to stage l there has been no error, then up to this stage the all-zero path through the trellis has the minimum metric.

Now, moving to the next stage (stage $(l+1)$th), it is possible that another path through the trellis will have a metric less than the all-zero path and, therefore, cause errors.

If this happens, we must have a path through the trellis that merges with the all-zero path, for for the first time, at the $(l+1)$th stage and has a metric less than the all-zero path as shown in Figure 9.34.

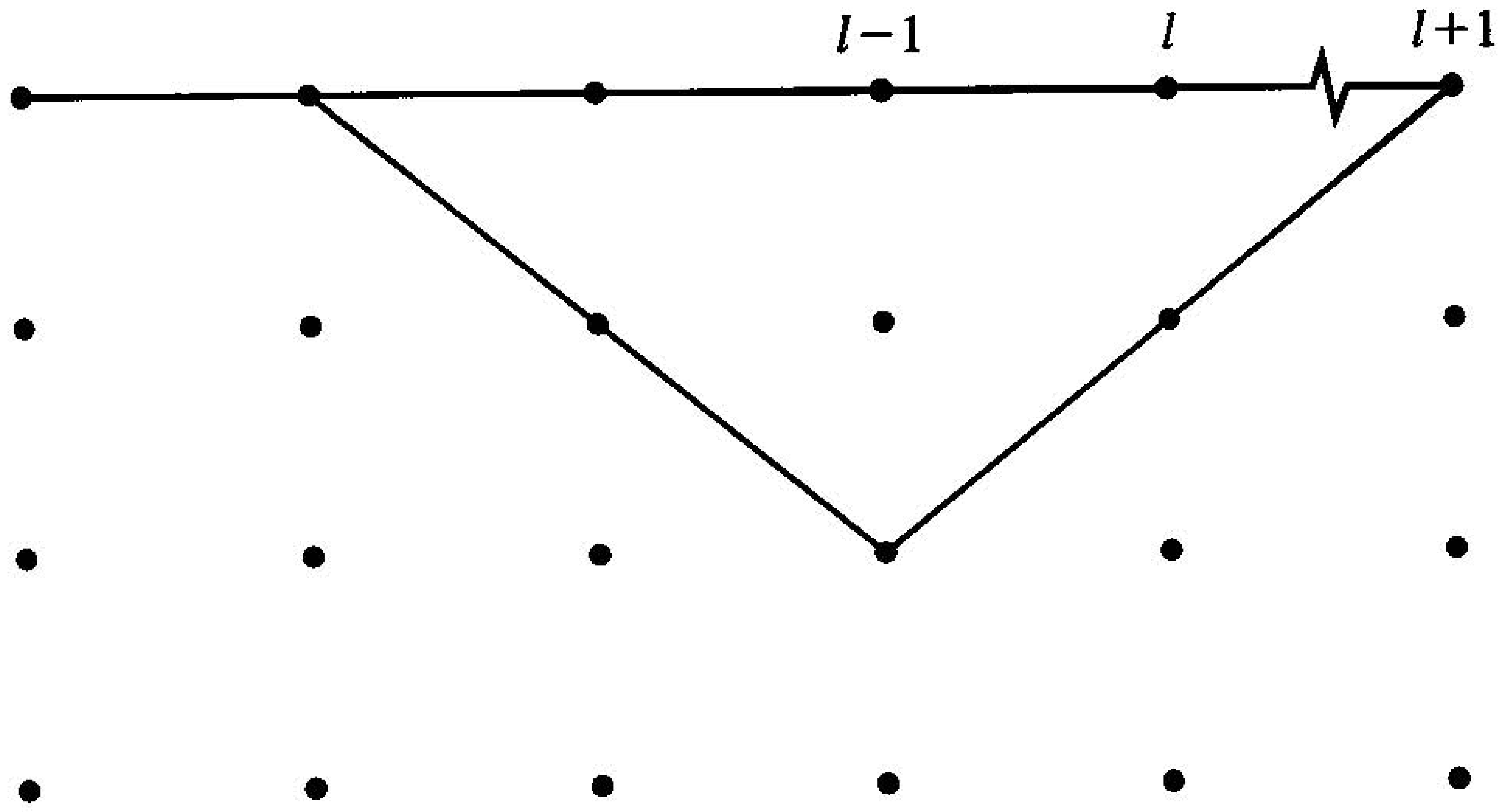


Figure 9.34 Path corresponding to the first-error-event.

Such an event is called the **first-error-event** and the corresponding probability is called **the first-error-event probability**.

Let $P_2(d)$ denote the probability that a path through the trellis which is at Hamming distance d from the all-zero path, is the survivor at the $(l+1)$st stage.

Denoting the number of paths of weight d by a_d, we can bound the first-error-event probability by

$$P_e \le \sum_{d=d_{\text{free}}}^{\infty} a_d P_2(d) \tag{9.7.10}$$

where on the right-hand side, we have included all paths through the trellis that merge with the all-zero path at the $(l+1)$st stage.

$P_2(d)$ depends on whether soft-or hard-decision decoding is employed.

<Skipped.>

Therefore, the coding gain is given by

$$\begin{aligned} G_{\text{coding}} &= \frac{d^2_{\text{coded}}}{d^2_{\text{uncoded}}} \\ &= 2 \sim 3\text{dB} \end{aligned} \tag{9.9.13}$$

which implies this simple coding scheme is capable of achieving a 3-dB coding gain without increasing bandwidth but at increased complexity in encoding and decoding.

Instead of a 8-state trellis, a trellis with a higher number of states yields higher coding gains.

Extensive computer simulations by Ungerboeck indicate that with 8, 16, 32, 64, 128, and 256 states coding gains in the range of $3.6 - 5.75$ dB can be achieved.

The trellis diagram for an 8-state trellis is shown in Figure 9.50.

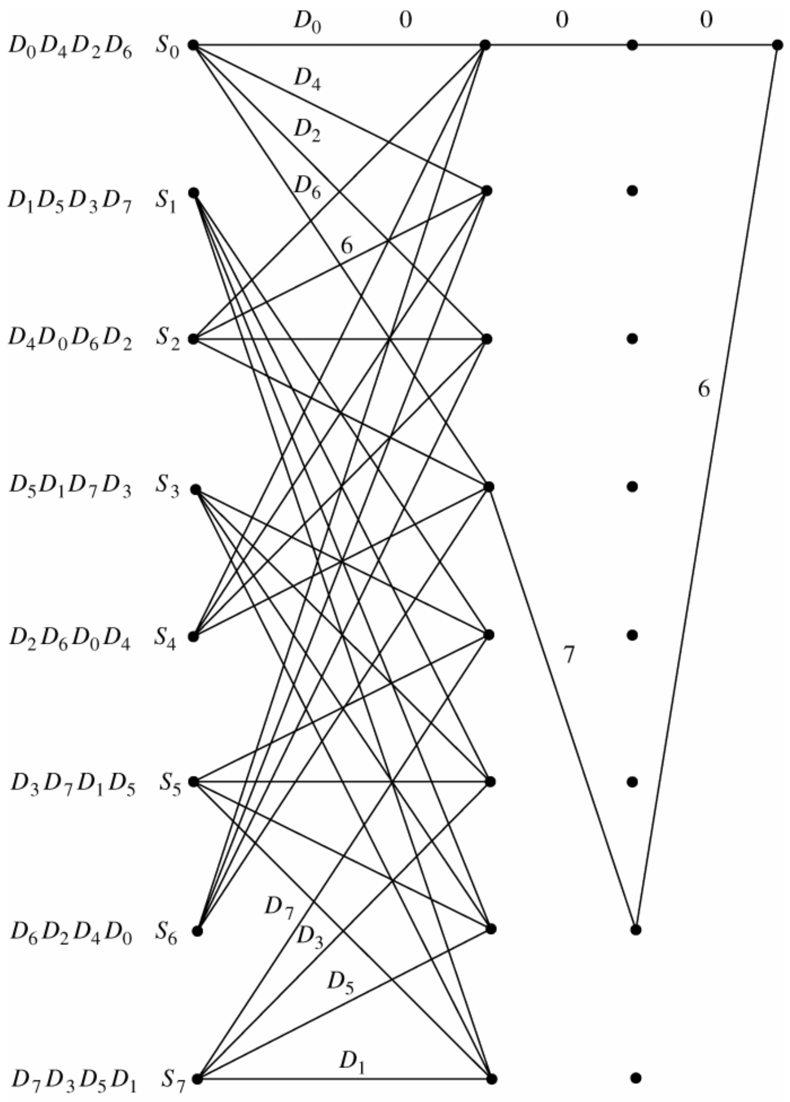


Figure 9.50

An 8-state Ungerboeck encoder.

9.10 Practical Applications of Coding

Coding can be employed to improve the effective SNR and, thus, enhance the performance of the digital communication system.

Block and convolutional codes and combinations of them in the form of concatenated and turbo codes as discussed earlier, have been applied to communication where bandwidth is not a major concern.

In cases where bandwidth is a major concern, as in digital communication over telephone channels, coded modulation can be employed.

By using coding, performance of digital communication systems are improved by up to 9 dB, depending on the application and the type of the code employed.

These include deep-space communications, telephone-line modems, and coding for compact discs (CDs).

9.10.1 Coding for Deep-Space Communications

Deep-space communication channels have very low SNRs and practically no bandwidth limitations.

The transmitter power is usually obtained from on-board solar cells and, therefore, is typically limited $20-30\ \mathrm{W}$.

The physical dimensions of the transmitting antenna are also limited and, therefore, its gain is limited too.

The enormous distance between the transmitter and the receiver and lack of repeaters results in a very low SNR at the receiver.

The channel noise can be characterized by a white Gaussian random process.

These channels are very well modeled as AWGN channels.

Because bandwidth is not a major concern on these channels both block and convolution codes can be applied.

In the Viking orbiters and landers mission to Mars, a $(32, 6)$ block code (Reed-Muller code) was employed that provided a coding gain a gain of approximately 4 dB with respect to an uncoded PSK system at an error rate of 10^{-6}.

Later, in the Voyager space mission to outer planets (Mars, Jupiter, and Saturn), convolutional codes with Viterbi decoding were employed.

Two codes that were designed at the Jet Propulsion Laboratory (JPL) for that mission were a $(2,1)$ convolutional code with a constraint length of $L=7$ with

$$\mathbf{g}_1 = [1 \quad 1 \quad 0 \quad 1 \quad 1 \quad 0 \quad 1]$$

$$\mathbf{g}_2 = [1 \quad 0 \quad 0 \quad 1 \quad 1 \quad 1 \quad 1]$$

and a (3, 1) convolutional code with $L=7$ and

$$\mathbf{g}_1 = [1 \quad 1 \quad 0 \quad 1 \quad 1 \quad 0 \quad 1]$$

$$\mathbf{g}_2 = [1 \quad 0 \quad 0 \quad 1 \quad 1 \quad 1 \quad 1]$$

$$\mathbf{g}_3 = [1 \quad 0 \quad 1 \quad 0 \quad 1 \quad 1 \quad 1].$$

The first code has a free distance of $d_{\text{free}} = 10$ and for the second code $d_{\text{free}} = 15$.

Both codes were decoded using the Viterbi algorithm and a soft-decoding scheme in which the output was quantized to $Q = 8$ levels.

The first code provides a coding gain of 5.1 dB with respect to an uncoded PSK system operating at an error rate of 10^{-5}.

The second code provides a gain of 5.7 dB.

Both codes operate about 4.5 dB above the theoretical limit predicted by Shannon’s formula.

In subsequent missions of the Voyager to Uranus, in 1986, the $(2,1)$ convolutional code with $L = 7$ was used as an inner code in a concatenated coding scheme where a $(255, 223)$ Reed-Solomon code served as the outer code.

Viterbi decoding followed by a Reed-Solomon decoder at the Earth terminal provided a total coding gain of 8 dB at an error rate of 10^{-6}.

This system operated at a data rate of approximately 30 Kbits/sec. For NASA's Pioneer 9 mission a $(2,1)$ convolutional code with a constraint length of $L=21$ was designed with generator sequences (in octal representation)

$\mathbf{g}_1 = [4 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0]$

$\mathbf{g}_2 = [7 \quad 1 \quad 5 \quad 4 \quad 7 \quad 3 \quad 7]$

which employed Fano's algorithm with a soft-decision decoding scheme and eight levels of output quantization.

Pioneers 10, 11, and 12 and also in Helios A and B German solar orbiter missions employed a $(2,1)$ convolutional code with a constraint length of $L=32$.

The generator sequences for this code (in octal representation) are given by

$\mathbf{g}_1 = [7 \quad 3 \quad 3 \quad 5 \quad 3 \quad 3 \quad 6 \quad 7 \quad 6 \quad 7 \quad 2]$

$\mathbf{g}_2 = [5 \quad 3 \quad 3 \quad 5 \quad 3 \quad 3 \quad 6 \quad 7 \quad 6 \quad 7 \quad 2]$.

This code has a free distance of $d_{\text{free}} = 23$.

For decoding, the Fano decoding algorithm with eight-level output quantization was employed.

Majority logic decoding has also been used in a number of coding schemes designed for the INTELSAT communication satellites.

As an example a (8, 7) code with $L = 48$ designed to operate at 64 Kbits/sec on an INTELSAT satellite was capable of improving the error rate from 10^{-4} to 5×10^{-8}.

In the Galileo space mission a (4, 1, 14) convolutional code was used, resulting in a spectral-bit rate of $\frac{1}{4}$, which at $\frac{\mathcal{E}_b}{N_0}$ of 1.75 dB, achieved an error probability of 10^{-5}.

This code performed 2.5 dB from the Shannon limit.

Using an outer (255, 223) Reed-Solomon code improves the coding gain by another 0.8 dB, resulting in a concatenated code operating 1.7-dB from the Shannon limit.

Turbo codes performing at a spectral-bit rate of 0.5 and operating only 0.7-dB from the Shannon limit, compare quite favorably with all systems described above.

9.10.2 Coding for Telephone-Line Modems

Telephone-line channels have a limited bandwidth, typically between 300-3000 Hz, and a rather high SNR, which is usually 28 dB or more.

Therefore, telephone-line channels is bandwidth limited, while the deep-space communication channel is power limited.

This corresponds to the case of $r \gg 1$ in Figure 9.11.

Because bandwidth is limited, we have to use low dimensional signaling schemes, and since power is rather abundant, we can employ multilevel modulation schemes.

Trellis-coded modulation is an appropriate scheme to be employed in such a case (see Section 9.5).

Historically, the first modems on telephone channels (prior to 1960s) employed frequency-shift keying with asynchronous detection and achieved bit rates in the range of $300-1200$ bits/sec.

Later, in the early 1960s, the first generation of synchronous modems employing 4-PSK modulation achieved bit rates of up to 2400 bit/sec.

Advances in equalization techniques allowed for more sophisticated constellations which resulted in higher bit rates.

These included 8-PSK modems achieving a bit rate of 4800 bit/sec and 16-point QAM modems that increased the bit rate to 9600 bits/sec.

In early 1980s, modems with a bit rate of $14,400$ bits/sec were introduced that employed a 64-point QAM signal constellation.

All these improvements were results of advances in equalization and signal-processing techniques and also improvments in the characteristics of telephone lines.

The advent of trellis-coded modulation made it possible to design coded modulation systems that improves overall system performance without requiring excess bandwidth.

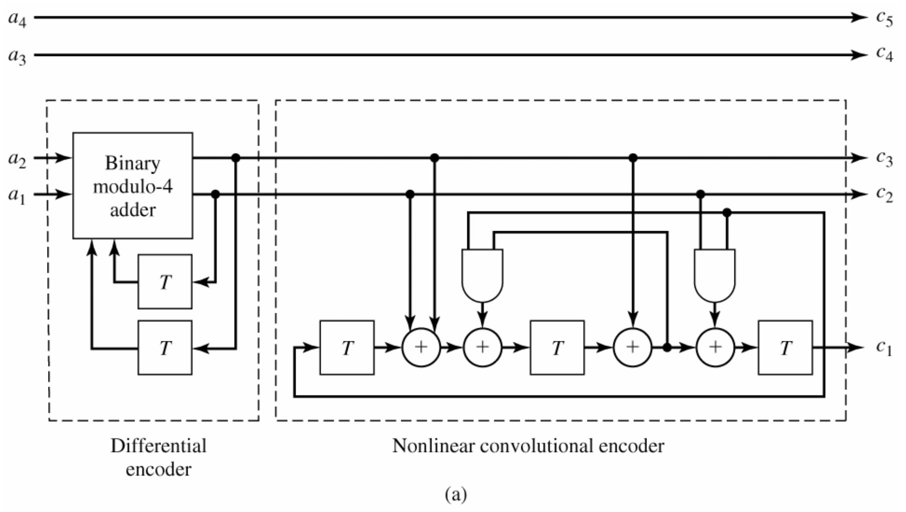
Trellis-coded modulation schemes based on variations of the original Ungerboeck’s codes and introduced by Wei (1984) were adopted as standard by the CCITT standard committees.

These codes are based on linear or nonlinear convolutional codes to guarantee invariance to 180°-or 90°-phase rotations.

This is crucial in applications where differential encoding is employed to avoid phase ambiguities when a PLL is employed for carrier-phase estimation at the receiver.

These codes achieve a coding gain comparable to Ungerboeck’s codes with the same number of states but, at the same time, provide the required phase invariance.

In Figure 9.51 we show the combination for the 8-state trellis-coded modulation system that is adopted in the CCITT V.32 standard.



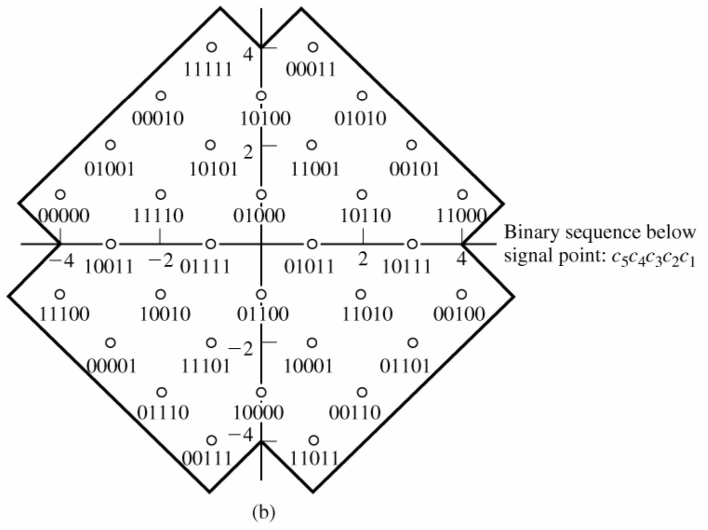


Figure 9.51

(a) Differential encoder, nonlinear convolutional encoder, and (b) signal constellation adopted in the V.32 standard.

9.10.3 Coding for Compact Discs

The storage medium for digital audio recording on a compact disc is a plastic disc with a diameter of 120 mm, a thickness of 1.2 mm and a track pitch of $1.6\ \mu\text{m}$.

At playing time, this disc is read by a laser beam at a velocity of 1.25 m/s.

Inside the spiral track on the disc, there are depressions called "pits" and flat areas between pits called "lands."

The digital audio is stored by the length of these pits and lands.

A 1 is represented by a transition from a pit to a land or vice versa, whereas a 0 corresponds to no transition (NRZI modulation).

Constraints on the physical length of pits and lands make it necessary to employ runlength-limited (RLL) codes (see Chapter 8).

The eight-to-fourteen modulation (EFM) code with $d = 2$ and $\kappa = 10$ is used in compact-disc recording.

The main source of errors in a compact disc is imperfections in the manufacturing of the disc, such as air bubbles in the plastic material or pit inaccuracies; and damages to the disc, such as fingerprints or scratches, dust, dirt, and surface abrasions.

Since each pit is almost $0.5\,\mu\text{m}$ wide and between 0.9 and $3.3\,\mu\text{m}$ long, these sources of errors result in error bursts, affecting many adjacent information bits.

This means that a good model for this storage channel is a channel with bursts of error, and well-known techniques for corresponding bursts of errors can be employed. Reed-Solomon codes are particularly attractive for such applications (see Section 9.6.1).

The left and the right channel are sampled at a rate of 44.1 KHz and then each sample is quantized to 16 levels (see Chapter 6).

Therefore, at each sampling instant there are 32 bits, or four 8-bit sequences to be encoded.

Each 8-bit sequence is called a “symbol.”

For error correction and detection, two Reed-Solomon codes are employed,† as shown in Fig.9.52.

† These are in fact, **shortened** Reed-Solomon codes, obtained from Reed-Solomon codes by putting some information bits equal to zero, and therefore reducing both k and n by a constant, while keeping the minimum distance intact.

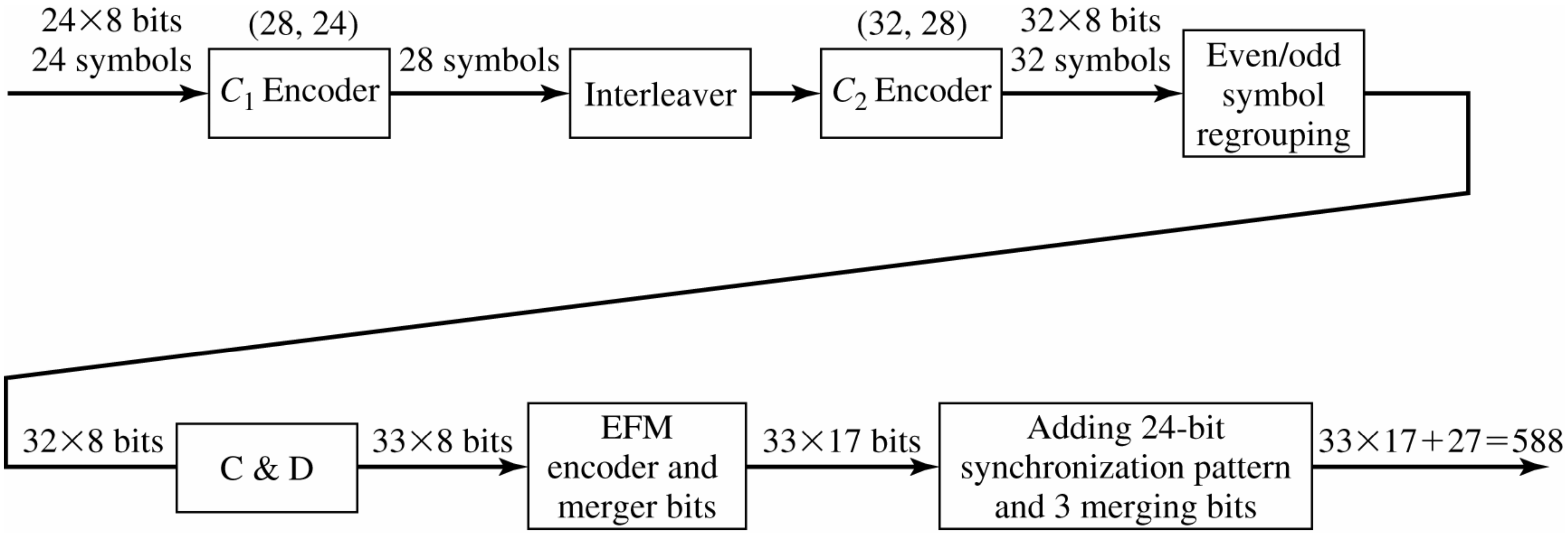


Figure 9.52

Encoding for compact-disc recording.

The first code, denoted by C_1 is a (28, 24) RS code and the second code, C_2, is a (32, 28) RS code.

The alphabet on which these codes are defined consists of binary sequences of length 8, which coincides with our definition of a symbol.

The input sequence to the C_1 encoder consists of 24 symbols (usually known as a "frame"), which are encoded into 28 symbols.

The 28 symbol at the output of the C_1 encoder are **interleaved** (see Section 9.5.2) to reduce the effect of error bursts and spread them over a longer interval, making them look more "random."

These are encoded by the C_2 encoder to 32 symbols.

At the output of the C_2 encoder, the odd-numbered symbols of each frame are grouped with the even-numbered symbols of the next frame to form a new frame.

At the output of the C_2 encoder corresponding to each set of six audio samples, we have 32 eight-bit symbols.

One more 8-bit symbol is added that contains the control and display (C&D) information, bringing the total to 33 symbols/frame.

The output is then applied to the eight-to-fourteen (EFM) runlength-limited encoder, which maps each symbol into a binary sequence of length 14.

Three more bits, called "merger" bits, are added for each symbol to make sure that the merger of codewords satisfies the runlength constraint (see Chapter 8).

These bring the length of each sequence to 17.

Next, a frame is completed by adding a 24 synchronization pattern and 3 additional "merger" bits to guarantee the runlength constraint after a merger.

This brings the total number of encoded bits/frame (six samples, or $6 \times 2 \times 16 = 192$ audio bits) to

$$33 \times 17 + 24 + 3 = 588$$

channel bits as shown Figure 8.29.

This means that an expansion of, roughly, three times has taken place.

These extra bits are used to protect the digital audio information bits from errors (the RS codes) and also make sure that the runlength constraint is satisfied (the EFM code).

On the playback, first synchronization and merger bits are separated, and then the 32 symbols are deinterleaved.

The result then enters the decoder for the code C_2.

This code has a minimum distance of 5 and, therefore, is capable of correcting up to 2 errors. The decoder, however, is designed to correct only 1 error.

Then according to the relation in (9.5.37)

$$e_c + e_d = d_{\min} - 1$$

it can detect up to 3 errors with certainty and 4 or more errors with high probability.

If a single error is encountered, it is corrected, of multiple errors are detected, then all 28 symbols are flagged as “unreliable.”

After deinterleaving, these symbols are passed to the decoder for the code C_1.

Decoder C_1 tries single error, or 2 erasure corrections.

If it fails, all output symbols are flagged; if there are more 3 or more flags at its input, it copies them to its output.

At the output of the second decoder, the symbol corresponding to “unreliable” positions are filled in by interpolation of the other positions.

Using this rather complex encoding-decoding technique together with the signal processing methods, burst errors of up to $12{,}000$ data bits, which correspond to a track length of 7.5 mm on the disc, can be conce