1
The principles of coding in digital communications

1.1 ERROR CONTROL SCHEMES

Error control coding is concerned with methods of delivering information from a source to a destination with a minimum of errors. As such it can be seen as a branch of information theory and traces its origins to Shannon's work in the late 1940s. The early theoretical work indicates what is possible and provides some insights into the general principles of error control. On the other hand, the problems involved in finding and implementing codes have meant that the practical effects of employing coding are often somewhat different from what was originally expected.

Shannon's work showed that any communication channel could be characterized by a capacity at which information could be reliably transmitted. At any rate of information transmission up to the channel capacity, it should be possible to transfer information at error rates that can be reduced to any desired level. Error control can be provided by introducing redundancy into transmissions. This means that more symbols are included in the message than are strictly needed just to convey the information, with the result that only certain patterns at the receiver correspond to valid transmissions. Once an adequate degree of error control has been introduced, the error rates can be made as low as required by extending the length of the code, thus averaging the effects of noise over a longer period.

Experience has shown that to find good long codes with feasible decoding schemes is more easily said than done. As a result, practical implementations may concentrate on the improvements that can be obtained, compared with uncoded communications. Thus the use of coding may increase the operational range of a communication system, reduce the error rates, reduce the transmitted power requirements or obtain a blend of all these benefits.

Apart from the many codes that are available, there are several general techniques for the control of errors, and the choice will depend on the nature of the data and the user's requirements for error-free reception. The most complex techniques fall into the category of forward error correction, where it is assumed that a code capable of correcting any errors will be used. Alternatives are to detect errors and request retransmission, which is known as retransmission error control, or to use inherent redundancy to process the erroneous data in a way that will make the errors subjectively important, a method known as error concealment.

This chapter first looks at the components of a digital communication system. Sections 1.3 to 1.8 then look in more detail at each of the components. Section 1.8 gives a simple example of a code that is used to show how error detection and correction may in principle be achieved. Section 1.9 discusses the performance of error correcting codes and Section 1.10 looks at the theoretical performance available. A number of more advanced topics are considered in Sections 1.11 to 1.14, namely coding for bandwidth-limited conditions, coding for burst errors, multistage coding (known as *concatenation*) and the alternatives to forward error correction. Finally, Section 1.15 summarizes the various considerations in choosing a coding scheme.

1.2 ELEMENTS OF DIGITAL COMMUNICATION SYSTEMS

A typical communication system incorporating coding is shown in Figure 1.1. Error control coding is applied after the source information is converted into digital format by the source encoder. The separation between coding and modulation is conventional, although it will be found later that there are instances where the two must be designed together. At the receiver, the operations are carried out in reverse order relative to the transmitter.

The functions are described in more detail in the following sections.

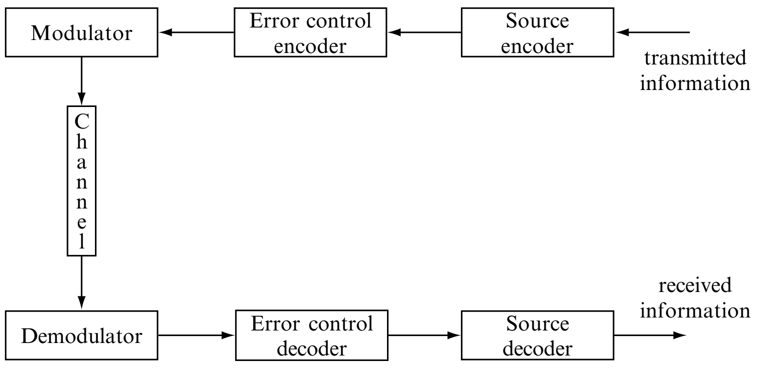

Figure 1.1 Coded communication system

1.3 SOURCE ENCODING

Information is given a digital representation, possibly in conjunction with techniques for removal of any inherent redundancy within the data. The amount of information contained in any message is defined in terms of the probability p that the message is selected for transmission. The information content H, measured in bits, is given by

$$H = \log_2 (1/p)$$

For example, a message with a 1% chance of being selected would contain approximately 6.64 bits of information.

If there are M messages available for selection and the probability of message m is denoted p_m, the average amount of information transferred in a message is

$$\overline{H} = \sum_{m=0}^{M-1} p_m \log_2 \left(\frac{1}{p_m} \right)$$

subject to the constraint that $\sum_{m=0}^{M-1} p_m = 1$.

If the messages are equiprobable, i.e. $p_m = 1/M$, then the average information transferred is just $\log_2 (M)$. This is the same as the number of bits needed to represent each of the messages in a fixed-length coding scheme. For example, with 256 messages an 8-bit code can be used to represent any messages and, if they are equally likely to be transmitted, the information content of any message is also 8 bits.

If the messages are not equally likely to be transmitted, then the average information content of a message will be less than $\log_2 (M)$ bits. It is then desirable to find a digital representation that uses fewer bits, preferably as close as possible to the average information content. This may be done by using variable length codes such as Huffman codes or arithmetic codes, where the length of the transmitted sequence matches as closely as possible the information content of the message. Alternatively, for subjective applications such as speech, images or video, lossy compression techniques can be used to produce either fixed-length formats or variable-length formats. The intention is to allow the receiver to reconstitute the transmitted information into something that will not exactly match the source information, but will differ from it in a way that is subjectively unimportant.

1.4 ERROR CONTROL CODING

Error control coding is in principle a collection of digital signal processing techniques aiming to average the effects of channel noise over several transmitted signals. The amount of noise suffered by a single transmitted symbol is much less predictable than that experienced over a longer interval of time, so the noise margins built into the code are proportionally smaller than those needed for uncoded symbols.

An important part of error control coding is the incorporation of redundancy into the transmitted sequences. The number of bits transmitted as a result of the error correcting code is therefore greater than that needed to represent the information. Without this, the code would not even allow us to detect the presence of errors and therefore would not have any error controlling properties. This means that, in theory, any incomplete compression carried out by a source encoder could be regarded as having error control capabilities. In practice, however, it will be better to compress the source information as completely as possible and then to re-introduce redundancy in a way that can be used to best effect by the error correcting decoder.

The encoder is represented in Figure 1.2. The information is formed into frames to be presented to the encoder, each frame consisting of a fixed number of symbols. In most cases the symbols at the input of the encoder are bits; in a very few cases symbols consisting of several bits are required by the encoder. The term symbol will be used to maintain generality.

To produce its output, the encoder uses the symbols in the input frame and possibly those in a number of previous frames. The output generally contains more symbols than the input, i.e. redundancy has been added. A commonly used descriptor of a code is the code rate (R) which is the ratio of input to output symbols in one frame. A low code rate indicates a high degree of redundancy, which is likely to provide more effective error control than a higher rate, at the expense of reducing the information throughput.

If the encoder uses only the current frame to produce its output, then the code is called a (n, k) block code, with the number of input symbols per frame designated k and the corresponding number of output symbols n. If the encoder remembers a number of previous frames and uses them in its algorithm, then the code is called a tree code and is usually a member of a subset known as convolutional codes. In this case the number of symbols in the input frame will be designated k_0 with n_0 symbols in the output frame. The encoder effectively performs a sliding window across the data moving in small increments that leave many of the same symbols still within the encoder window, as shown in Figure 1.3. The total length of the window, known as

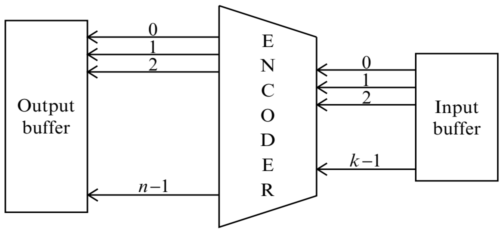

Figure 1.2 Encoder

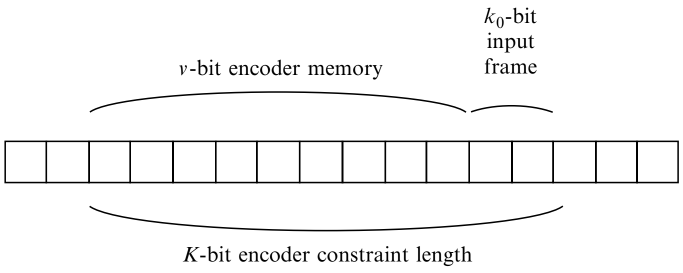

Figure 1.3 Sliding window for tree encoder

the *input constraint length* (K), consists of the input frame of k_0 symbols plus the number of symbols in the memory. This latter parameter is known as *memory constraint length* (v).

In more complex systems the encoding may consist of more than one stage and may incorporate both block and convolutional codes and, possibly, a technique known as interleaving. Such systems will be considered in later sections.

One property that will be shared by all the codes in this book is *linearity*. If we consider a linear system we normally think in terms of output being proportional to input (scaling property). For a linear system we can also identify on the output the sum of the separate components deriving from the sum of two different signals at the input (superposition property). More formally, if the system performs a function f on an input to produce its output, then

$$f(c\mathbf{x}) = c \times f(\mathbf{x}) \quad \text{(scaling)}$$

$$f(\mathbf{x} + \mathbf{y}) = f(\mathbf{x}) + f(\mathbf{y}) \quad \text{(superposition)}$$

where c is a scalar quantity, $\mathbf{x}$ and $\mathbf{y}$ are vectors.

Now the definition of a linear code is less restrictive than this, in that it does not consider the mapping from input to output, but merely the possible outputs from the encoder. In practice, however, a linear system will be used to generate the code and so the previous definition will apply in all real-life cases.

The standard definition of a linear code is as follows:

- Multiplying a code sequence by a valid scalar quantity produces a code sequence.
- Adding two code sequences produces a code sequence.

The general rules to be followed for multiplication and addition are covered in Chapter 5 but for binary codes, where the only valid scalars are 0 and 1, multiplication of a value by zero always produces zero and multiplication by 1 leaves the value unchanged. Addition is carried out as a modulo-2 operation, i.e. by an exclusive-OR function on the values.

A simple example of a linear code will be given in Section 1.8. Although the definition of a linear code is less restrictive than that of a linear system, in practice linear codes will always be produced by linear systems. Linear codes must contain the all-zero sequence, because multiplying any code sequence by zero will produce an all-zero result.

1.5 MODULATION

The modulator can be thought of as a kind of digital to analogue converter, preparing the digital code-stream for the real, analogue world. Initially the digital stream is put into a *baseband* representation, i.e. one in which the signal changes at a rate comparable with the rate of the digital symbols being represented. A convenient representation is the *Non Return to Zero* (NRZ) format, which represents bits by signal levels of $+V$ or $-V$ depending on the bit value. This is represented in Figure 1.4.

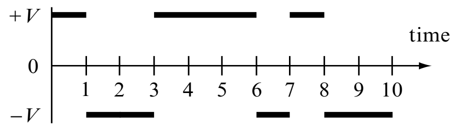

Figure 1.4 Binary NRZ stream

Although it would be possible to transmit this signal, it is usual to translate it into a higher frequency range. The reasons for this include the possibility of using different parts of the spectrum for different transmissions and the fact that higher frequencies have smaller wavelengths and need smaller antennas. For most of this text, it will be assumed that the modulation is produced by multiplying the NRZ baseband signal by a sinusoidal carrier whose frequency is chosen to be some multiple of the transmitted bit rate (so that a whole number of carrier cycles are contained in a single-bit interval). As a result, the signal transmitted over a single-bit interval is either the sinusoidal carrier or its inverse. This scheme is known as *Binary Phase Shift Keying* (BPSK).

It is possible to use a second carrier at 90° to the original one, modulate it and add the resulting signal to the first. In other words, if the BPSK signal is $\pm\cos(2\pi f_c t)$, where f_c is the carrier frequency and t represents time, the second signal is $\pm\sin(2\pi f_c t)$ and the resultant is

$$s(t) = \sqrt{2}\cos(2\pi f_c t + i\pi/4) \quad i = -3,\ -1,\ +1,\ +3$$

This is known as *Quadriphase Shift Keying* (QPSK) and has the advantage over BPSK that twice as many bits can be transmitted in the same time and the same bandwidth, with no loss of resistance to noise. The actual bandwidth occupied by the modulation depends on implementation details, but is commonly taken to be 1 Hz for one bit per second transmission rate using BPSK or 0.5 Hz using QPSK.

A phase diagram of QPSK is shown in Figure 1.5. The mapping of the bit values onto the phases assumes that each of the carriers is independently modulated using alternate bits from the coded data stream. It can be seen that adjacent points in the diagram differ by only one bit because the phase of only one of the two carriers has changed. A mapping that ensures that adjacent points differ by only one bit is known as *Gray Coding*.

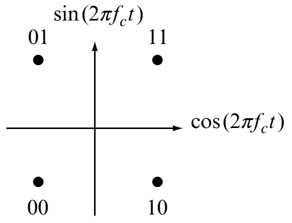

Figure 1.5 Gray-coded QPSK phase diagram

Other possible modulations include *Frequency Shift Keying* (FSK), in which the data determines the frequency of the transmitted signal. The advantage of FSK is simplicity of implementation, although the resistance to noise is less than BPSK or QPSK. There are also various modulations of a type known as *Continuous Phase Modulation*, which minimize phase discontinuities between transmitted waveforms to improve the spectral characteristics produced by nonlinear power devices.

In bandwidth-limited conditions, multilevel modulations may be used to achieve higher bit rates within the same bandwidth as BPSK or QPSK. In *M-ary Phase Shift Keying* (MPSK) a larger number of transmitted phases is possible. In *Quadrature Amplitude Modulation* (QAM) a pair of carriers in phase quadrature are each given different possible amplitudes before being added to produce a transmitted signal with different amplitudes as well as phases. QAM has more noise resistance than equivalent MPSK, but the variations in amplitude may cause problems for systems involving nonlinear power devices. Both QAM and MPSK require special approaches to coding which consider the code and the modulation together.

1.6 THE CHANNEL

The transmission medium introduces a number of effects such as attenuation, distortion, interference and noise, making it uncertain whether the information will be received correctly. Although it is easiest to think in terms of the channel as introducing errors, it should be realized that it is the effects of the channel on the demodulator that produce the errors.

The way in which the transmitted symbols are corrupted may be described using the following terms:

- Memoryless channel – the probability of error is independent from one symbol to the next.
- Symmetric channel – the probability of a transmitted symbol value *i* being received as a value *j* is the same as that of a transmitted symbol value *j* being received as *i*, for all values of *i* and *j*. A commonly encountered example is the binary symmetric channel (BSC) with a probability *p* of bit error, as illustrated in Figure 1.6.

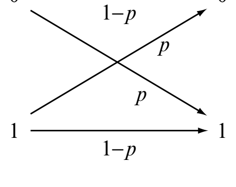

Figure 1.6 Binary symmetric channel

- Additive White Gaussian Noise (AWGN) channel – a memoryless channel in which the transmitted signal suffers the addition of wide-band noise whose amplitude is a normally (Gaussian) distributed random variable.

- Bursty channel – the errors are characterized by periods of relatively high symbol error rate separated by periods of relatively low, or zero, error rate.

- Compound (or diffuse) channel – the errors consist of a mixture of bursts and random errors. In reality all channels exhibit some form of compound behaviour.

Many codes work best if errors are random and so the transmitter and receiver may include additional elements, an *interleaver* before the modulator and a *deinterleaver* after the demodulator to randomize the effects of channel errors. This will be discussed in Section 1.12.

1.7 DEMODULATION

1.7.1 Coherent demodulation

The demodulator attempts to decide on the values of the symbols that were transmitted and pass those decisions on to the next stage. This is usually carried out by some sort of correlation with replicas of possible transmitted signals. Consider, for example, the case of BPSK. The correlation is with a single fixed phase of the carrier, producing either a positive or a negative output from the detector. In the absence of noise, the detected signal level can be taken as $\pm\sqrt{E_r}$ where E_r is the energy in each received bit. The effect of an AWGN channel will be to add a noise level n sampled from a Gaussian distribution of zero mean and standard deviation σ. The probability density is given by

$$p(n)\,dn = \frac{1}{\sigma\sqrt{2\pi}}e^{-n^2/2\sigma^s}$$

Gaussian noise has a flat spectrum and the noise level is often described by its *Single-sided Noise Power Spectral Density*, which is written N_0. The variance, σ^2, of the Gaussian noise, integrated over a single-bit interval, will be $N_0/2$. In fact it can be considered that there is a total noise variance of N_0 with half of this acting either in phase or in antiphase to the replica and the other half in phase quadrature, therefore not affecting the detector. The performance of Gray-coded QPSK is therefore exactly the same as BPSK because the two carriers can be demodulated as independent BPSK signals, each affected by independent Gaussian noise values with the same standard deviation.

The demodulator will make its decision based on the sign of the detected signal. If the received level is positive, it will assume that the value corresponding to the replica was transmitted. If the correlation was negative, it will assume that the other value was transmitted. An error will occur, therefore, if the noise-free level is $-\sqrt{E_r}$

and a noise value greater than $+\sqrt{E_r}$ is added, or if the noise-free level is $+\sqrt{E_r}$ and a noise value less than $-\sqrt{E_r}$ is added. Considering only the former case, we see that the probability of error is just the probability that the Gaussian noise has a value greater than $+\sqrt{E_r}$:

$$p = \frac{1}{\sqrt{\pi N_0}} \int_{\sqrt{E_r}}^{\infty} e^{-n^2/N_0}\, dn$$

Substituting $t = n/\sqrt{N_0}$ gives

$$p = \frac{1}{\sqrt{\pi}} \int_{\sqrt{\frac{E_b}{N_0}}}^{\infty} e^{-t^2/N_0}\, dt$$

$$p = \frac{1}{2} erfc\sqrt{\frac{E_b}{N_0}} \quad \text{where} \quad erfc(x) = \frac{2}{\sqrt{\pi}} \int_{t}^{\infty} e^{-t^2}\, dt \tag{1.1}$$

The function $erfc(x)$ is known as the complementary error function and its values are widely available in tabulated form. Note that the maximum value of $erfc(x)$ is 1.0, so that the maximum bit error probability is 0.5. This makes sense because we could guess bit values with 50 % probability without attempting to receive them at all.

1.7.2 Differential demodulation

One further complication is commonly encountered with BPSK or QPSK transmissions. In the absence of other information, it is impossible for the receiver to determine absolute phase so as to know which of the two phases represents the value 0 and which represents 1. This is because delay in the transmission path, which is equivalent to phase shift, will not be known. The representation of bit values is therefore often based on the difference between phases. Depending on the precise demodulation method, this is known either as *Differentially Encoded Phase Shift Keying* (DEPSK) or as *Differential Phase Shift Keying* (DPSK). The two are identical from the modulation point of view, with the bit value 0 normally resulting in a change of phase and the bit value 1 resulting in the phase remaining the same. The receiver may not know absolute phase values, but should be able to tell whether the phase has changed or remained the same. The differences in the demodulator implementation may be summarized as follows:

- *DEPSK* – The demodulator maintains a replica of one of the two carrier phases and correlates the received signal with this replica as for normal PSK. It then compares the sign of the correlation with the previous correlation value; a change of sign indicates data bit 0 and the same sign indicates data bit 1. Compared with PSK, there will now be a bit error either when the phase is received wrongly and the previous phase was correct or when the phase is received correctly and the previous phase was wrong. Thus noise that would cause a single-bit error in a BPSK demodulator will cause two consecutive bit errors in the DEPSK demodulator and the bit error probability is approximately twice the above BPSK expression.

- *DPSK* – The demodulator uses the previously received phase as the replica for the next bit. Positive correlation indicates data value 1, negative correlation indicates data value 0. The bit errors again tend to correlate in pairs, but the overall performance is worse. In fact the bit error probability of DPSK follows a different shape of curve:

$$p = \frac{1}{2} e^{-E_r/N_0}$$

1.7.3 Soft-decision demodulation

In some cases the demodulator's decision will be easy; in other cases it will be difficult. In principle if errors are to be corrected it is better for the demodulator to pass on the information about the certainty of its decisions because this might assist the decoder in pinpointing the positions of the likely errors; this is called soft-decision demodulation. We could think of it as passing on the actual detected level as a real number, although in practice it is likely to have some sort of quantization. Eight-level quantization is found to represent a good compromise between complexity and performance.

Since the purpose of soft decisions is to assist decoding, it is useful to relate the demodulator output to probabilistic measures of performance. One commonly adopted measure is known as the *log-likelihood ratio*, defined as $\log[p(1|r_i)/p(0|r_i)]$. This metric is required in an important decoding method to be described in Chapter 10 and can be used for other decoding methods too. The computation of the value may appear difficult, however we note that

$$\log\left[\frac{p(1|r_i)}{p(0|r_i)}\right] = \log\left[\frac{p(r_i|1)}{p(r_i|0)}\right] + \log\left[\frac{p(1)}{p(0)}\right]$$

Assuming that values 0 and 1 are equiprobable, $\log[p(1)/p(0)] = 0$ and so the assigned bit value for received level r_i is equal to $\log[p(r_i|1)/p(r_i|0)]$. This value can be calculated given knowledge of the signal level and the noise statistics. Note that it ranges from $-\infty$ (certain 0) to $+\infty$ (certain 1).

Assuming that we have Gaussian noise, the probability density function at a received value r_i from a noise-free received value x is

$$p(r_i) = \frac{1}{\sqrt{\pi N_0}} e^{-(r_i - x)^2/N_0}$$

The appropriate values of x for bit values 1 and 0 are $+\sqrt{E_r}$ and $-\sqrt{E_r}$. Thus the log-likelihood ratio is proportional to $\log\left[e^{-(r_i-\sqrt{E_r})^2/N_0} / e^{-(r_i+\sqrt{E_r})^2/N_0}\right]$.

Now

$$\log\left[\frac{e^{-(r_i-\sqrt{E_r})^2/N_0}}{e^{-(r_i+\sqrt{E_r})^2/N_0}}\right] = -\frac{(r_i-\sqrt{E_r})^2}{N_0} + \frac{(r_i+\sqrt{E_r})^2}{N_0}$$

Hence we find that

$$\log\left[\frac{p(r_i|1)}{p(r_i|0)}\right] = \frac{4r_i\sqrt{E_r}}{N_0} = \frac{4r_i}{\sqrt{E_r}}\frac{E_r}{N_r}$$

In other words, the log-likelihood ratio is linear with the detected signal level and is equal to the channel E_r/N_0, multiplied by four times the detected signal (normalized to make the noise-free levels equal to $+/-1$).

Note that the mapping adopted here from code bit values to detected demodulator levels is opposite to that conventionally used in other texts. The conventional mapping is that bit value 0 maps onto $+1$ and bit value 1 onto -1. The advantage is that the exclusive-OR operation in the digital domain maps onto multiplication in the analog domain. The disadvantage is the potential confusion between bit value 1 and analog value $+1$.

Because of the linearity of the log-likelihood ratio, the quantization boundaries of the demodulator can be set in roughly linear steps. The question remains, however, as to what size those steps should be. It can be shown that, for Q-level quantization, the optimum solution is one that minimizes the value of

$$\sum_{j=0}^{Q-1}\sqrt{p(j|1)\,p(j|0)}$$

where $p(j|c)$ represents the probability of a received value j given that symbol c was transmitted. Massey [1] described an iterative method of finding the optimum solution with nonuniform arrangement of boundaries, but the above value can easily be calculated for different linear spacings to find an approximate optimum. For example, with E_r/N_0 around 2 dB, it is found that uniformly spaced quantization boundaries are close to optimum if the spacing is $1/3$, i.e. the boundaries are placed at -1, $-2/3$ $-1/3, 0,$ $+1/3$, $+2/3$, $+1$. The use of such a scheme will be described in Section 1.8.2.

1.8 DECODING

The job of the decoder is to decide what the transmitted information was. It has the possibility of doing this because only certain transmitted sequences, known as codewords, are possible and any errors are likely to result in reception of a non-code sequence. On a memoryless channel, the best strategy for the decoder is to compare the received sequence with all the codewords, taking into account the confidence in the received symbols, and select the codeword which is closest to the received sequence as discussed above. This is known as *maximum likelihood decoding*.

1.8.1 Encoding and decoding example

Consider, for example, the block code shown in Table 1.1. This code is said to be *systematic*, meaning that the codeword contains the information bits and some other bits known as *parity checks* because they have been calculated in some way from the information. It can be seen that any codeword differs in at least three places from any other codeword. This value is called the *minimum Hamming distance* or, more briefly, *minimum distance* of the code. Consequently, if a single bit is wrong, the received sequence is still closer to the transmitted codeword, but if two or more bits are wrong, then the received sequence may be closer to one of the other codewords.

This code is linear and for any linear code it is found that the *distance structure* is the same from any codeword. For this example, starting from any codeword there are two sequences at a distance of 3 and one at a distance of 4. Thus the code properties and the error-correction properties are independent of the sequence transmitted. As a consequence, the minimum distance of the code can be found by comparing each nonzero sequence with the all-zero sequence, finding the nonzero codeword with the smallest number nonzero symbols. The count of nonzero symbols is known as the *weight* of the sequence and the minimum weight of the code is equal to the minimum distance.

Let us now assume that information 10 has been selected and that the sequence 10101 is therefore transmitted. Let us also assume that the received bits are hard-decision quantized. If the sequence is received without error, it is easy to identify it in the table and to decode. If there are errors, however, things will be more difficult and we need to measure the number of differences between the received sequence and each codeword. The measure of difference between sequences is known as *Hamming distance*, or simply as distance between the sequences. Consider first the received sequence 00101. The distance to each codeword is shown in Table 1.2.

In this case we can see that we have a clear winner. The transmitted sequence has been selected as the most likely and the decoding is correct.

Table 1.1 Example block code

Information	Codeword
00	00000
01	01011
10	10101
11	11110

Table 1.2 Distances for sequence 00101

Codeword	Distance
00000	2
01011	3
10101	1
11110	4

The previous example had an error in an information bit, but the result will be the same if a parity check bit is wrong. Consider the received sequence 10111. The distances are shown in Table 1.3. Again the sequence 10101 is chosen. Further examples are left to the reader, but it will be found that any single-bit error can be recovered, regardless of the position or the codeword transmitted.

Now let us consider what happens if there are two errors. It will be found that there are two possibilities.

Firstly, consider the received sequence 11111. The distances are shown in Table 1.4. In this case, the codeword 11110 is chosen, which is wrong. Moreover, the decoder has decided that the final bit was wrong when in fact it was correct. Because there are at least three differences between any pair of codewords, the decoder has made an extra error on top of the two made by the channel, in effect making things worse.

Finally, consider the received sequence 11001, whose distances to the codewords are shown in Table 1.5. In this case, there are two problems in reaching a decision. The first, and obvious, problem is that there is no clear winner and, in the absence of other information, it would be necessary to choose randomly between the two most likely codewords. Secondly, we predicted at the outset that only single errors would be correctable and the decoder may have been designed in such a way that it refuses to decode if there is no codeword within a distance 1 of the received sequence. The likely outcome for this example, therefore, is that the decoder will be unable to

Table 1.3 Distances for sequence 10111

Codeword	Distance
00000	4
01011	3
10101	1
11110	2

Table 1.4 Distances for sequence 11111

Codeword	Distance
00000	5
01011	2
10101	2
11110	1

Table 1.5 Distances for sequence 11001

Codeword	Distance
00000	3
01011	2
10101	2
11110	3

choose the most likely transmitted codeword and will indicate to the user the presence of *detected uncorrectable errors*. This is an important outcome that may occur frequently with block codes.

1.8.2 Soft-decision decoding

The probability that a sequence **c** of length n was transmitted, given the received sequence **r**, is $\Pi_{i=0}^{n-1} p(c_i|r_i)$. We wish to maximize this value over all possible code sequences. Alternatively, and more conveniently, we take logarithms and find the maximum of $\sum_{i=0}^{n-1} \log[p(c_i|r_i)]$. This can be carried out by a correlation process, which is a symbol-by-symbol multiplication and accumulation, regarding the code bits as having values $+1$ or -1. Therefore we would be multiplying the assigned probability by 1 for a code bit of 1 and by -1 for a code bit of 0. For hard decisions, a codeword of length n at a distance d from the received sequence would agree in $n - d$ places and disagree in d places with the received sequence, giving a correlation metric of $2n - d$. Obviously choosing the codeword to maximize this metric would yield the same decoding result as the minimum distance approach.

Even with soft decisions, we can adopt a minimum distance view of decoding and minimize $\sum_{i=0}^{n-1} \{1 - \log[p(c_i|r_i)]\}$. The correlation and minimum distance approaches are again identical provided we have an appropriate measure of distance. If the received bits are given values v_i equal to $\log[p(1|r_i)]$, then the distance to a bit value 1 is $1 - v_i$, the distance to a bit value 0 is v_i and we maximize probability by minimizing this measure of distance over all codewords.

The maximization of probability can also be achieved by maximizing some other function that increases monotonically with it. This is the case for the log-likelihood ratio $\log[p(1|r_i)/p(0|r_i)]$. To decode, we can maximize $\sum_i c_i \log[p(r_i|1)/p(r_i|1)]$ where c_i is taken as having values ± 1. This again corresponds to carrying out a correlation of received log-likelihood ratios with code sequences.

As discussed in Section 1.7.3, it is likely that the received levels will be quantized. For 8-level quantization, it might be convenient to use some uniform set of metric values depending on the range within which the detected bit falls. Such a scheme is shown in Figure 1.7.

Bearing in mind the fact that the log-likelihood ratio is linear with the analog detected level from the demodulator, then the only deviation from an ideal 8-level

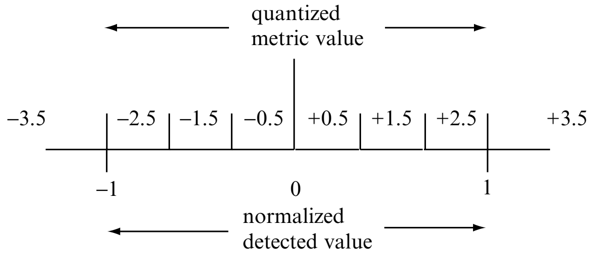

Figure 1.7 Quantization boundaries for soft-decision demodulation

quantization is that the end categories (000 and 111) extend to $-\infty$ and $+\infty$ and therefore should have larger metrics associated with them. The effect on performance, however, is negligible. For $E_r/N_0 = 2\,\text{dB}$, the optimum soft-decision metric values associated with this quantization arrangement are -3.85, -2.5, -1.5, -0.5, $+0.5$, $+1.5$, $+2.5$, $+3.85$. Therefore the proposed metrics of -3.5 to $+3.5$ are very close to optimum.

The assigned values can be scaled and offset in any convenient manner, so the scheme in Figure 1.7 is equivalent to having bit values of (-7, -5, -3, -1, $+1$, $+3$, $+5$, $+7$) or (0, 1, 2, 3, 4, 5, 6, 7). This last form is convenient for implementation of a 3-bit interface to the decoder.

Applying the correlation approach to a soft-decision case, the example in Table 1.4 might become a received sequence $+2.5$ $+0.5$ $+1.5$ $+0.5$ $+3.5$ with correlation values as shown in Table 1.6.

The maximum correlation value indicates the decoder decision. In this case, the decoder selects the correct codeword, illustrating the value of soft decisions from the demodulator.

Table 1.6 Correlations for soft-decision sequence $+2.5$ $+0.5$ $+1.5$ $+0.5$ $+3.5$

Codeword	Correlation
-1 -1 -1 -1 -1	-8.5
-1 $+1$ -1 $+1$ $+1$	$+0.5$
$+1$ -1 $+1$ -1 $+1$	$+6.5$
$+1$ $+1$ $+1$ $+1$ -1	$+1.5$

1.8.3 Alternative decoding approaches

Although conceptually very simple, the method described above is very complex to implement for many realistic codes where there may be very many codewords. As a result, other decoding methods will need to be studied. For example, the parity checks for the above code were produced according to very simple rules. Numbering the bits from left to right as bits 4 down to 0, bits 4 and 3 constitute the information and the parity bits are

$$\begin{aligned}
\text{bit } 2 &= \text{bit } 4 \\
\text{bit } 1 &= \text{bit } 3 \\
\text{bit } 0 &= \text{bit } 4 \oplus \text{bit } 3
\end{aligned}$$

The symbol $\oplus$ denotes modulo-2 addition or exclusive-OR operation. Considering only hard decisions, when a sequence is received, we can simply check whether the parity rules are satisfied and we can easily work out the implications of different error patterns. If there are no errors, all the parity checks will be correct. If there is a single-bit error affecting one of the parity bits, only that parity check will fail. If bit 4 is in error, parity bits 2 and 0 will be wrong. If bit 3 is in error, parity bits 1 and 0 will be wrong. If both parity bits 2 and 1 fail, the error is uncorrectable, regardless of whether parity bit 0 passes or fails.

We can now construct some digital logic to check the parity bits and apply the above rules to correct any correctable errors. It will be seen that applying the rules will lead to the same decodings as before for the examples shown. In the final example case, where the sequence 11001 was received, all three parity checks fail.

This type of decoding procedure resembles the methods applied for error correction to many block codes. Note, however, that it is not obvious how such methods can incorporate soft decisions from the demodulator. Convolutional codes, however, are decoded in a way that is essentially the same as the maximum likelihood method and soft decisions can be used.

1.9 CODE PERFORMANCE AND CODING GAIN

We saw earlier that we can obtain a theoretical expression for the bit error probability of BPSK or QPSK on the AWGN channel in terms of the ratio of energy per received bit to single-sided noise power spectral density, E_r/N_0. It is convenient to do the same for systems that employ coding, however we first have to solve a problem of comparability. Coding introduces extra bits and therefore we have to increase either the time to send a given message or else the bandwidth (by transmitting faster). Either case will increase the total noise in the message; in the first case because we get noise from the channel for a longer time, in the second case because more noise falls within the bandwidth.

The answer to this problem is to assess the error performance of the link in terms of E_b/N_0, the ratio of energy per bit of information to noise power spectral density. Thus when coding is added, the number of bits of information is less than the number of transmitted bits, resulting in an increase in E_b/N_0 relative to E_r/N_0. For example, if 100 bits of information are to be sent using a rate 1/2 code, 200 bits must be transmitted. Assuming that we maintain the transmitted bit rate and power, the energy in the message is doubled, but the amount of information remains the same. Energy per bit of information is therefore doubled, an increase of 3 dB. This increase acts as a penalty that the code must overcome if it is to provide real gains. The performance curve is built up in three stages as explained below.

As the first stage, the curve of bit error rate (BER) against E_b/N_0 (the same as E_r/N_0 in this case) is plotted for the modulation used. The value of E_b/N_0 is usually measured in dB and the bit error rate is plotted as a logarithmic scale, normally covering several decades, e.g. from 10^{-1} to 10^{-6}. The second stage is the addition of coding without consideration of the changes to bit error rates. For a fixed number of transmitted bits, the number of information bits is reduced, thus increasing the value of E_b/N_0 relative to E_r/N_0 by a factor $1/R$, or by $10\log_{10}(1/R)$ dB. The third stage is to consider the effect of coding on bit error rates; this may be obtained either by simulation or by calculation. For every point on the uncoded performance curve, there will therefore be a corresponding point a fixed distance to the right of it on the coded performance curve showing a different, in many cases lower, bit error rate.

An example is shown in Figure 1.8 which shows the theoretical performance of a BPSK (or QPSK) channel, uncoded and with a popular rate 1/2 convolutional code. The code performance is plotted both with hard-decision demodulation and with unquantized soft decisions, i.e. real number output of detected level from the demodulator.

It can be seen that without coding, the value of E_b/N_0 needed to achieve a bit error rate of 10^{-5} is around 9.6 dB. This error rate can be achieved with coding at E_b/N_0 around 7.1 dB using hard-decision demodulation or around 4.2 dB using unquantized soft-decision demodulation. This is expressed by saying that the *coding gain* at a BER of 10^{-5} is 2.5 dB (hard-decision) or 5.4 dB (soft-decision). Real life decoding gains would not be quite so large. The use of 8-level, or 3-bit, quantization of the soft decisions reduces the gain by around 0.25 dB. There may also be other implementation issues that affect performance. Nevertheless, gains of 4.5 to 5 dB can be expected with this code.

The quoted coding gain must be attached to a desired bit error rate, which in turn will depend on the application. Note that good coding gains are available only for relatively low required bit error rates and that at higher error rates the gain may be negative (i.e. a loss). Note also that the quoted bit error rate is the error rate coming out of the decoder, *not* the error rate coming out of the demodulator. In the soft-decision example, the demodulator is working at E_r/N_0 around 1.2 dB, producing a BER of around 5×10^{-2} out of the demodulator.

If we know the minimum distance of a block code, or the value of an equivalent parameter called *free distance* for a convolutional code, we can find the *asymptotic*

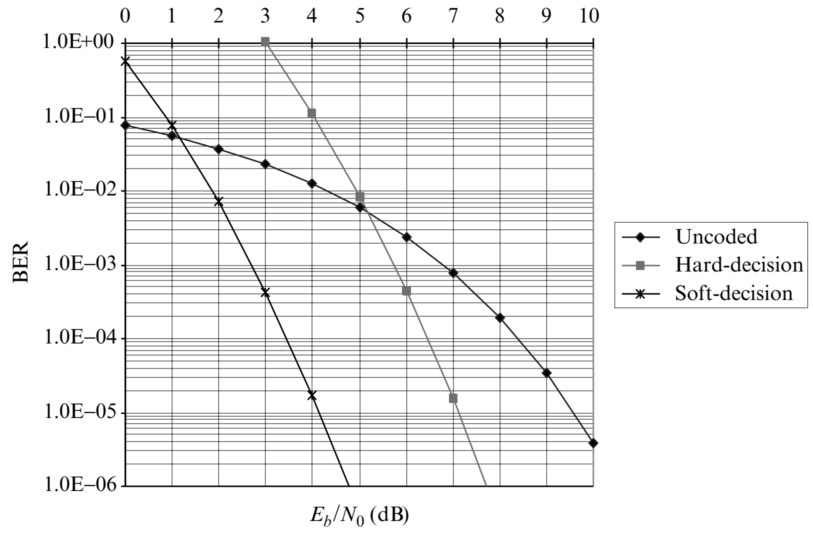

Figure 1.8 Performance of rate 1/2 convolutional code

coding gain, i.e. the gain that would be delivered if vanishingly small decoded error rates were required. For unquantized soft-decision decoding of a rate R code with distance d between the closest code sequences, the asymptotic gain is

$$G_{\text{asymptotic}} = 10\log_{10}(Rd) \tag{1.2}$$

If we have only hard decisions from the demodulator and can correct up to t errors then

$$G_{\text{asymptotic}} = 10\log_{10}[R(t+1)] \tag{1.3}$$

From the earlier simple block code example, we can see that the expected value of t would be the largest value that is less than half the value of d. Thus the value of d in Equation (1.2) is just less than twice the value of $t + 1$ in Equation (1.3). Asymptotic coding gains are therefore almost 3 dB higher when unquantized soft-decision decoding is used. As stated above, the use of 8-level quantization reduces the gain by about 0.25 dB.

Although we have solved one issue of comparability by the use of E_b/N_0, there is another that is regularly ignored. If we look at an uncoded channel and a coded channel with the same BER, the characteristics will be completely different. On the AWGN channel, the errors will occur at random intervals. On the coded channel there will be extended error-free intervals interspersed with relatively dense bursts of errors when the decoder fails. Thus if we are interested in error rates on larger units of transmission, frames, packets or messages, the coded channel at the same BER will give fewer failures but more bit errors in corrupted sections of transmission. Assessing coding gain by comparing coded and uncoded channels with the same BER may therefore be unfair to the coded channel. For example, out of 100 messages sent, an uncoded channel might result in 10 message errors with one bit wrong in each. A coded channel might produce only one message error but 10 bit errors within that message. The bit error rates are the same, but the message error rate is better on the coded channel. Add to this the fact that the detection of uncorrectable errors is rarely taken into account in a satisfactory way (a significant issue for many block codes), coding regularly delivers benefits that exceed the theoretical figures.

1.10 INFORMATION THEORY LIMITS TO CODE PERFORMANCE

We have now seen the sort of benefits that coding provides in present day practice and the ways to find asymptotic coding gain based on knowledge of simple code parameters. As yet we have not seen how to do detailed error rate calculations as these require a more detailed knowledge of code structure. Nevertheless, it is worth making a comparison with the results obtained from Shannon's work on information theory to show that, in some respects, coded systems have still some way to go.

Shannon showed that, using an average of all possible codes of length n, the error rate over the channel is characterized by a probability of message error

$$P_e \leq e^{-nE(R_{\mathrm{I}})} \tag{1.4}$$

where E, which is a function of the information rate, is called the random coding error exponent. Any specific code will have its own error exponent and the greater the error exponent the better the code, but there are calculable upper and lower bounds to the achievable value of E. In particular, a positive error exponent is achievable provided R_{I} is less than some calculable value called the channel capacity. Provided a positive error exponent can be obtained, the way to achieve lower error probabilities is to increase the length of the code.

As was seen in Section 1.9, codes have a calculable asymptotic coding gain and thus at high signal-to-noise values the error rates reduce exponentially with E_b/N_0, as in the uncoded case. The error exponent is therefore proportional to E_b/N_0. The difficulty with known codes is maintaining the error exponent while the length is increased. All known codes produced by a single stage of encoding can hold their value of error exponent only by reducing the rate to zero as the code length increases towards infinity. For example, an orthogonal signal set, which can be achieved by Frequency Shift Keying or by means of a block code, is sometimes quoted as approaching the theoretical capacity on an AWGN channel as the signal set is expanded to infinity. Unfortunately the bandwidth efficiency or the code rate reduces exponentially at the same time. This limitation can be overcome by the use of multistage encoding, known as concatenation, although even then the error exponents are less than the theoretically attainable value. Nevertheless, concatenation represents the closest practicable approach to the predictions of information theory, and as such is a technique of increasing importance. It is treated in more detail in Chapters 9 and 10.

As the most widely available performance figures for error correcting codes are for the additive white Gaussian noise (AWGN) channel, it is interesting to look at the theoretical capacity of such a channel. The channel rate is given by the Shannon–Hartley theorem:

$$C = B \log_2\left(1 + \frac{S}{N}\right) \tag{1.5}$$

where B is bandwidth, S is signal power and N is noise power within the bandwidth. This result behaves roughly as one might expect, the channel capacity increasing with increased bandwidth and signal-to-noise ratio. It is interesting to note, however, that in the absence of noise the channel capacity is not bandwidth-limited. Any two signals of finite duration are bound to show differences falling within the system bandwidth, and in the absence of noise those differences will be detectable.

Let $N = B \cdot N_0$ and $S = R_{\mathrm{I}}E_b$ (N_0 is the single-sided noise power spectral density, R_{I} is rate of information transmission ($\leq C$) and E_b is energy per bit of information), then

$$C = B\log_2\left(1 + \frac{R_I E_b}{BN_0}\right)$$

In the limit of infinite bandwidth, using the fact that $\log_2(x) = \log_e(x)/\log_e 2$ gives

$$C = 1.44\,B\log_e\left(1 + \frac{R_I E_b}{BN_0}\right)$$

As bandwidth approaches infinity, the channel capacity is given by

$$C \approx 1.44 R_I \frac{E_b}{N_0}$$

For transmission at the channel capacity, ($R_I = C$):

$$\frac{E_b}{N_0} = \frac{1}{1.44} = -1.6\,\text{dB} \tag{1.6}$$

This means that we should be able to achieve reliable communications at the channel capacity with values of E_b/N_0 as low as -1.6 dB. The channel capacity is however proportional to the information rate; increasing the rate for a fixed value of E_b/N_0 increases the signal power and therefore the channel capacity. Thus at -1.6 dB we should be able to achieve reliable communications at any rate over an AWGN channel, provided we are willing to accept infinite bandwidth.

If instead we constrain the bandwidth and set $R_I = \eta B$, where η is bandwidth efficiency of the modulation/coding scheme, then

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

For transmission at the channel capacity ($R_I = C$), therefore

$$\frac{E_b}{N_0} = \frac{1}{\eta}(2^{\eta} - 1) \tag{1.7}$$

This value can be thought of as imposing an upper limit to the coding gain achievable by a particular coding and modulation scheme. The value of E_b/N_0 to deliver the desired error rate on the uncoded channel can be determined from the modulation performance, and the corresponding coded value must be at least that given by equation (1.7). In practice, these coding gains are difficult to achieve.

If we were to use a rate 1/2 code on a QPSK channel, a fairly common arrangement, the value of η is around 1.0, giving $E_b/N_0 = 1$ (= 0 dB). As has been seen earlier, a rate 1/2 convolutional code may need over 4.5 dB to deliver a BER of 10^{-5}. It therefore falls well short of the theoretical maximum gain.

It must be stressed that Shannon merely proved that it was possible by coding to obtain reliable communications at this rate. There is no benefit, however, in having a good code if one does not know how to decode it. Practical codes are designed with a feasible decoding method in mind and the problem of constructing long codes that can be decoded is particularly severe. This seems to be the main reason why approaching the Shannon performance has proved to be so difficult.

1.11 CODING FOR MULTILEVEL MODULATIONS

The standard modulation for satellite communications is QPSK, but 8-PSK or 16-PSK could be used to obtain 3 or 4 transmitted bits (respectively) per transmitted symbol. Unfortunately, this results in reduced noise immunity. With m bits per transmitted symbol, assuming that the energy per transmitted bit is maintained, the energy per transmitted symbol can increase by a factor of m relative to binary PSK. The distance between closest points in the constellation will, however, be proportional to $\sin(\pi/M)$, where $M = 2^m$, as shown in Figure 1.9, and the noise energy required to cause an error will depend on the square of this. The uncoded performance relative to binary PSK is therefore

$$G_m\,(\text{dB}) = 10\,\log_{10}[m \sin^2(\pi/2^m)]$$

The values are shown in Table 1.7.

As can be seen, there are severe losses associated with higher level constellations, making coding all the more important. The codes, however, need to be designed specifically for the constellation to maximize the distance in signal space, the *Euclidean Distance*, between code sequences.

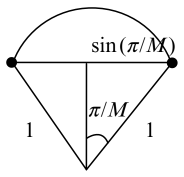

Figure 1.9 Distance between MPSK constellation points

Table 1.7 Performance of uncoded MPSK

m	M	G_m (dB)
1	2	0.0
2	4	0.0
3	8	−3.6
4	16	−8.2
5	32	−13.2
6	64	−18.4
7	128	−23.8
8	256	−29.2

The principal approach to designing codes for this type of system is to take a constellation with m bits per symbol and to use a rate $(m-1)/m$ code so that the information throughput will be the same as the uncoded constellation with $m-1$ bits per symbol and the performances can be compared directly. Convolutional codes of this type are known as Ungerboeck codes and will be described in Chapter 2.

1.12 CODING FOR BURST-ERROR CHANNELS

Coding performance curves are regularly shown for the AWGN channel. There are two reasons why this is so. Firstly, burst-error mechanisms are often badly understood and there may be no generally accepted models that fit the real behaviour. The increasing importance of mobile communications where the channel does not remotely fit the AWGN model has, however, led to considerable advances in the modelling of non-Gaussian channels. The other reason is that most codes in use are primarily designed for random error channels. The only important codes where this is not the case are Reed Solomon codes which are constructed with multibit symbols and correct a certain number of symbol errors in each codeword. A burst of errors affecting several bits close together may affect only a few symbols of the code and be correctable, as shown in Figure 1.10. The symbols each consist of 4 bits and a burst spanning 8 bits containing 5 errors has affected only 3 symbols.

For the most part, we shall be faced with trying to make a random bit-error-correcting code work on a burst-error channel, and the technique that is used is interleaving. Essentially, this consists of reordering the bits before transmission (interleaving) and putting them back into the original order on reception (deinterleaving). As the error locations are affected only by the deinterleaving, they become scattered through the code-stream so that they appear as random errors to the decoder.

There are two main types of interleaving to consider, block interleaving and convolutional interleaving. Both will be explained as if they are being used with a block code, although both can be used with convolutional codes too.

Block interleaving is illustrated in Figure 1.11. Codewords are written into the columns of an array and the total number of columns, λ, is termed the *interleaving degree*. If a burst of errors spans no more than λ symbols, then there will be at most one error in each codeword. A code that can correct up to t errors could correct, for example, up to t bursts of length λ, one burst of length (λt or a mixture of shorter bursts and random errors.

Convolutional interleaving is shown in Figure 1.12. The codewords in the columns of the array are shifted through delays which differ for each symbol. Usually these are increasing by one for each row of the array. The order of symbols on the channel

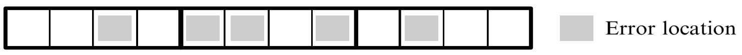

Figure 1.10 Binary burst error on multibit symbols

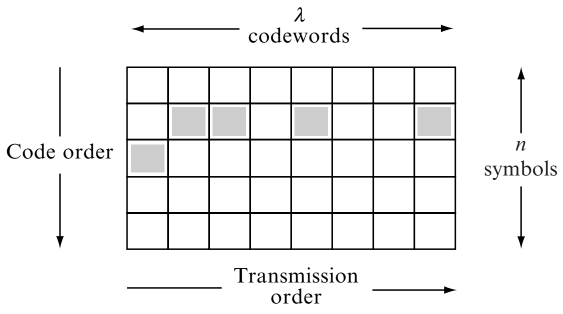

Figure 1.11 Block interleaving

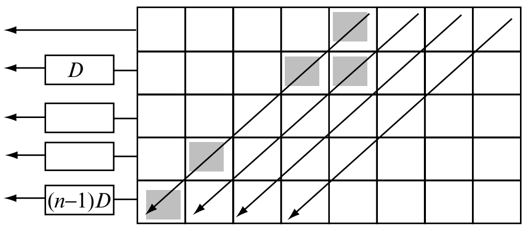

Figure 1.12 Convolutional interleaving

follows the diagonal sequence shown. Any burst of errors will affect symbols in the transmission stream as shown and it can be seen that the burst must exceed $n + 1$ symbols in length before it affects two symbols of the same codeword. If the delays are increasing by D for each symbol, then the separation of two symbols from the same codeword is $Dn + 1$. In effect this is the interleaving degree.

The main differences between the two types of interleaving are that the convolutional interleaver will extend the symbol stream through the presence of null values in the delay registers, but block interleaving will have more delay because of the need to fill the array before transmission can commence.

One might think that the block interleaver would introduce a delay of λn symbols, however it is possible to start transmission a little before the array is filled. The encoder must have $(\lambda - 1)n + 1$ symbols prepared by the time that λ symbols are transmitted; otherwise, the rightmost symbol of the top row will not be ready in time for transmission (assuming that symbol is transmitted the instant it is prepared). The delay is therefore $(\lambda - 1)n + 1 - \lambda = (\lambda - 1)(n - 1)$ symbols. The same delay will occur in the deinterleaver which writes the symbols into rows and decodes by column, giving an overall delay of $2(\lambda - 1)(n - 1)$ symbols.

The convolutional interleaver introduces $D + 2D + \cdots + (n - 1)D = n(n - 1)D/2$ dummy symbols into the stream. The deinterleaver applies a delay of $(n - 1)D$ to the top row, $(n - 2)D$ to the second row, etc., introducing the same number of dummy symbols. The overall delay is therefore $n(n - 1)D$. As the interleaving degree is $nD + 1$, the overall delay is $(\lambda - 1)(n - 1)$, half the value of the block interleaving.

1.13 MULTISTAGE CODING

The aim of making an effective long code is sometimes approached by multistage coding in which the overall code is constructed from simple components, thus providing a feasible approach to decoding. Examples of this type of approach include serial concatenation, in which information is first encoded by one code, the *outer code*, and then the encoded sequence is further encoded by a second code, the *inner code*. Reed Solomon codes are often used as outer codes because of their ability to correct the burst errors from the inner decoder. Another approach is the *product code* in which information is written into an array and the rows and columns are separately encoded.

In recent years other types of concatenation have become of interest in conjunction with iterative decoding techniques, where decoding of the second code is followed by one or more further decodings of both codes. In particular, iterative decoding is applied to *parallel concatenated codes*, namely the application of two systematic codes to a single-information stream to derive two independent sets of parity checks. This is the principle of the so-called *turbo codes* and other similar constructions, which are treated in Chapter 10.

1.14 ERROR DETECTION BASED METHODS

So far we have assumed that the intention is to correct all errors if possible; this is known as *Forward Error Correction* (FEC). We have, however, seen that detected uncorrectable errors are possible. In fact there may be good reasons not to attempt error correction provided we have some other way of dealing with erroneous data. Not attempting error correction will not make the maximum use of the received sequence, but it makes it less likely that there will be undetected errors and reduces the complexity at the receiver.

There are two main possibilities if errors are not to be corrected. The first approach is to use a reverse channel (where available) to call for retransmission. This is known as *Retransmission Error Control* (REC) or *Automatic Retransmission reQuest* (ARQ). The second approach is to process the data in such a way that the effect of errors is minimized. This is called *Error Concealment*.

1.14.1 ARQ strategies

The transmitter breaks the data into frames, each of which contains a block code used for error detection. The receiver sends back acknowledgements of correct frames and whenever it detects that a frame is in error it calls for retransmission. Often the transmitter will have carried on sending subsequent frames, so by the time it receives the call for retransmission (or fails to obtain an acknowledgement within a predefined interval) it will already have transmitted several more frames. It can then either repeat just the erroneous frame (*Selective Repeat ARQ*) or else go back to the point in the sequence where the frame error occurred and repeat all frames from that point regardless (*Go Back N ARQ*).

If Selective Repeat (SR-ARQ) is employed, the receiver must take responsibility for the correct ordering of the frames. It must therefore have sufficient buffering to reinsert the repeated frame into the stream at the correct point. Unfortunately, it is not possible to be sure how many repeats will be needed before the frame will be received. The protocols therefore need to be designed in such a way that the transmitter recognizes when the receiver's buffer is full and repeats not only erroneous frames but also those which will have been lost through buffer overflow.

Neglecting effects of finite buffers, assuming independence of errors from frame to frame and a frame error rate of p_f, the efficiency of SR-ARQ is

$$\eta_{\mathrm{SR}} = (1 - p_f)\left(\frac{k}{n}\right) \tag{1.8}$$

where n is the total frame length and k is the amount of information in the frame. The difference between n and k in this case will not be purely the parity checks of the code. It will include headers, frame numbers and other fields required by the protocol.

For Go Back N (GBN-ARQ), there is no need for receiver buffering, but the efficiency is lower. Every time x frames are received correctly, followed by one in error, the transmitter goes on to frame $x + N$ before picking up the sequence from frame $x + 1$. We can therefore say that

$$\eta_{\mathrm{GBN}} = \frac{\overline{x}}{\overline{x} + N}\frac{k}{n}$$

Now the probability of x frames being successful followed by one that fails is $p_f(1 - p_f)^x$; therefore

$$\overline{x} = \sum_{i=0}^{\infty} i p_f (1 - p_f)^i = p_f(1 - p_f)\left[1 + 2(1 - p_f) + 3(1 - p_f)^2 + \cdots\right]$$

The sum to infinity of the series in the square brackets is $1/p_f^2$, so we find that

$$\overline{x} = \frac{1 - p_f}{p_f}$$

Hence

$$\eta_{\mathrm{GBN}} = \frac{1 - p_f}{1 - p_f + p_f N}\frac{k}{n} \tag{1.9}$$

It may appear from this that an efficient GBN scheme would have a small value of N, however the value of N depends on the round trip delays and the frame length. Small values of N will mean long frames which in turn will have a higher error rate. In fact it is the frame error rate that is the most important term in the efficiency expression, with the factor k/n also playing its part to ensure that frames cannot be made too small.

The main difficulties with ARQ are that efficiency may be very low if the frame error rate is not kept low and that the delays are variable because they depend on the number of frame errors occurring. The delay problem may rule out ARQ for real time applications, particularly interactive ones. The solution to the efficiency problem may be to create some sort of hybrid between FEC and ARQ with FEC correcting most of the errors and reducing the frame error rate and additional error detection resulting in occasional use of the ARQ protocols.

1.14.2 Error concealment

Some applications carry data for subjective appreciation where there may still be some inherent redundancy. Examples include speech, music, images and video. In this case, the loss of a part of the data may not be subjectively important, provided that the right action is taken. Designing a concealment system is a signal processing task requiring knowledge of the application, the source coding and the subjective effects of errors. Possibilities include interpolation or extrapolation from previous values. Hybrids with FEC are also possible.

Error concealment is often appropriate for exactly the applications where ARQ is difficult or impossible. One example is digital speech where the vocoders represent filters to be applied to an input signal. The filter parameters change relatively slowly with time and so may be extrapolated when a frame contains errors. Another example occurs with music on compact disc where the system is designed in a way that errors in consecutive samples are unlikely to occur. The FEC codes have a certain amount of extra error detection and samples known to contain errors are given values interpolated from the previous and the following sample.

1.14.3 Error detection and correction capability of block codes

Error detection schemes or hybrids with FEC are usually based on block codes. In general, we can use block codes either for error detection alone, for error correction or for some combination of the two. Taking into account that we cannot correct an error that cannot be detected, we reach the following formula to determine the guaranteed error detection and correction properties, given the minimum distance of the code:

$$d_{\min} > s + t \tag{1.10}$$

where s is the number of errors to be detected and t ($\leq s$) is the number of errors to be corrected. Assuming that the sum of s and t will be the maximum possible then

$$d_{\min} = s + t + 1$$

Thus if $d_{\min} = 5$, the possibilities are

$s = 4$	$t = 0$
$s = 3$	$t = 1$
$s = 2$	$t = 2$

If we decided, for example, to go for single-error correction with triple-error detection, then the occurrence of four errors would be detected, but the likelihood is that the decoder would assume it was the result of a single error on a different codeword from the one transmitted.

If the code is to be used for correction of the maximum amount of errors, and if the value of minimum distance is odd, then setting $t = s$ gives

$$d_{\min} = 2t + 1 \tag{1.11}$$

1.15 SELECTION OF CODING SCHEME

The factors which affect the choice of a coding scheme are the data, the channel and specific user constraints. That includes virtually everything. The data can have an effect through its structure, the nature of the information and the resulting error-rate requirements, the data rate and any real-time processing requirements. The channel affects the solution through its power and bandwidth constraints and the nature of the noise mechanisms. Specific user constraints often take the form of cost limitations, which may affect not only the codec cost but also the possibility of providing soft-decision demodulation.

1.15.1 General considerations

The major purpose of incorporating coding into the design of any system is to reduce the costs of the other components. Reliable communications can usually be obtained by simple, yet costly, methods such as increasing power. A well-designed coding scheme should result in a lower overall system cost for an equivalent or better performance. If this objective is to be met, however, the designer needs to make a careful choice and be aware of the whole range of available techniques.

Convolutional codes are highly suitable for AWGN channels, where soft decisions are relatively straightforward. The coding gains approach the asymptotic value at relatively high bit error rates, so that at bit error rates of 10^{-5} to 10^{-7} in Gaussian conditions, convolutional codes are often the best choice. Many types of conditions, however, can give rise to non-Gaussian characteristics where the soft-decision thresholds may need to adapt to the channel conditions and where the channel coherence may mean that Viterbi decoding is no longer the maximum likelihood solution. The complexity of the decoder also increases as the code rate increases above $1/2$, so that high code rates are the exception. Even at rate $1/2$, the channel speed which can be accommodated is lower than for Reed Solomon codes, although it is still possible to work at over 100 Mbits/second, which is more than enough for many applications!

Reed Solomon codes have almost exactly complementary characteristics. They do not generally use soft decisions, but their performance is best in those conditions where soft decisions are difficult, i.e. non-Gaussian conditions. In Gaussian conditions the performance curves exhibit something of a ‘brick wall’ characteristic, with the codes working poorly at high bit error rates but showing a sudden transition to extremely effective operation as the bit error rate reduces. Thus they may show very high asymptotic coding gains but need low bit error rates to achieve such gains. Consequently they are often advantageous when bit error rates below 10^{-10} are required. Error rates as low as this are often desirable for machine-oriented data, especially if there is no possibility of calling for a retransmission of corrupted data. The decoding complexity reduces as code rate increases, and in many cases decoding can be achieved at higher transmitted data rates. They can also, of course, be combined with other codes (including convolutional codes or other RS codes) for concatenated coding.

For the future, the so-called turbo codes are going to be of increasing importance. These are tree codes of infinite constraint length, used in combination and decoded by an iterative method. Usually two codes are used with one operating on an interleaved data set. The decoding algorithms not only use soft decisions, they also provide soft decisions on the outputs, and the output of each decoder is fed to the input of the other so that successive iterations converge on a solution. The performance is extremely good, giving acceptable error rates at values of E_b/N_0 little above the Shannon levels. There are, however, several problems to be resolved including the existence of an error floor making it difficult to achieve output BERs below 10^{-5} or 10^{-6}.

The above considerations certainly do not mean that other types of codes have no place in error control. Many considerations will lead to the adoption of other solutions, as will be seen from the discussions below. Nevertheless, mainstream interests in future systems are likely to concentrate on Viterbi-decoded convolutional codes, Reed Solomon codes and turbo codes, and the designer wishing to adopt a standard, ‘off-the-shelf’ solution is most likely to concentrate on these alternatives.

1.15.2 Data structure

If information is segmented into blocks, then it will fit naturally with a block coding scheme. If it can be regarded as a continuous flow, then convolutional codes will be most appropriate. For example, protecting the contents of computer memories is usually done by block coding because the system needs to be able to access limited sections of data and decode them independently of other sections. The concept of data ordering applies only over a limited span in such applications. On the other hand, a channel carrying digitized speech or television pictures might choose a convolutional scheme. The information here is considered to be a continuous stream with a definite time order. The effects of errors will be localized, but not in a way which is easy to define.

It is important to separate the structure of the data from the characteristics of the channel. The fact that a channel carries continuous data does not necessarily mean that the data is not segmented into block form. Less obvious, but equally important, a segmented transmission does not necessarily imply segmented data. A TDMA channel, for example, may concentrate several continuous streams of information into short bursts of time, but a convolutional code may still be most appropriate. With adequate buffering, the convolutional code on any stream may be continued across the time-slots imposed by the TDMA transmission.

1.15.3 Information type

It is conventional to assess the performance of coding schemes in terms that involve bit error rates. This is not really appropriate for many types of information, and the most appropriate measure will often affect the choice of a coding scheme. Indeed it is difficult to think of any application in which the bit error rate is directly important. If discrete messages are being sent, with every bit combination representing a totally different message, then the message error rate is of crucial importance; the number of bit errors in each wrong message is not important at all. Even with information that is subjected to some kind of sensory evaluation (i.e. it is intended for humans, not for machines), not all bits are equal. In most cases there are more and less significant bits or some bits whose subjective importance is different from that of others. Digitized speech without any data compression carries a number of samples, each of which has a most and a least significant bit. Only if bit errors in all positions have equal effect will bit error rate provide a measure of subjective quality. If the speech is at all compressed, the bits will represent different types of information, such as filter poles or excitation signals, and the subjective effects will vary. Data intended for subjective evaluation may be suitable for error concealment techniques.

Errors on a coded channel can be placed into four categories. There are those which are corrected by the code and allow the information to be passed on to the destination as if those errors had never occurred. There are errors which are detected but not corrected. There are also errors which are not detected at all and errors which are detected but the attempted correction gives the wrong result. Errors are passed on to the destination in the last two cases. For many applications it is important to minimize the probability of unsuspected errors in the decoder output. This will bias the user towards block codes, which often detect errors beyond the planned decoding weight, and away from forward error correction which accepts that undetected decoding errors will occur. The strength of the bias depends on the consequence of errors. If an error could start the next world war, it is obviously of more importance than one that causes a momentary crackle on a telephone line.

Acceptable error rates will depend not only on the type of data but also on whether it will be processed on- or off-line. If data is to be processed immediately, it may be possible to detect errors and invoke some other strategy such as calling for retransmission. Off-line processing means that errors cannot be detected until it is too late to do anything about it. As a result the error rate specification will commonly be lower.

Note that there must always be some level of errors which is considered to be acceptable. It is easy to set out with a goal of eliminating all errors. Achieving this goal would require infinite time and an infinite budget.

1.15.4 Data rate

It is difficult to put figures on the data rates achievable using different codes. This is partly because any figures given can quickly become out of date as technology advances and partly because greater speeds can usually be achieved by adopting a more complex, and therefore more expensive, solution. Nevertheless, for a fixed complexity, there are some codes which can be processed more rapidly than others.

The codes which can be processed at the highest data rates are essentially simple, not very powerful, codes. Examples are codes used purely for error detection. Concatenated codes using short block inner codes are not far behind because the computations on the Reed Solomon codes are done at symbol rate, not bit rate, and the block codes used are extremely simple. It follows that Reed Solomon codes alone are in the highest data rate category. Viterbi-decoded convolutional codes are fast provided the input constraint length is not too long, say no more than 9. BCH codes can also be used at similar rates provided hard-decision decoding only is required. Soft-decision decoding of block codes and the more complex concatenated schemes, e.g. turbo codes, are capable of only moderate data rates.

Of course, the required data rate affects the choice of technology too; the more that can be done in hardware the faster the decoding. Parallelism can increase decoding speeds, but with higher hardware complexity and therefore cost. A data rate of a few thousand bits per second could allow a general-purpose microprocessor to be used for a wide range of codecs, but obviously that would be uneconomic for volume production. Many of the influences of data rate on system design will be closely bound up with economics.

1.15.5 Real time data processing

If real time data processing is required, the decoder must be able to cope with the link data rates. This may be achieved at the expense of delays by, for example, decoding one sequence while the next is being buffered. The decoding delay may in some cases become significant, especially if it is variable.

Forward error correction requires a decoding delay that, in most cases, depends on the exact errors which occur. Nevertheless, there is usually a certain maximum delay that will not be exceeded. Buffering the decoded information until the maximum delay has expired can therefore produce a smooth flow of information to the destination. Two major factors determining the delay will be the data rate and the length of the code. Information theory tells us that long codes are desirable, but for many applications long delays are not. Thus the maximum acceptable delay may limit the length of the codes that can be used.

If no maximum decoding delay can be determined, then the decoded information will come through with variable delays, which can cause havoc with real time information. The main error control strategy that exhibits variable delays is ARQ because one cannot guarantee that any retransmission will be successful. These problems may be minimized by the use of a suitable ARQ/FEC hybrid.

1.15.6 Power and bandwidth constraints

These constraints drive the solution in opposite directions. In the absence of bandwidth constraints one would use a low rate concatenated code to achieve high coding gains or very low error rates. Very tight bandwidth constraints, making binary modulation incompatible with the required data rate and error rates, require the use of specially designed codes in conjunction with multilevel modulations. Traditionally these have been convolutional codes, but block codes may be possible and turbo-coded solutions are being developed.

Assuming that the major aim of coding is to reduce the power requirement for a given error rate, high coding gains would appear to be desirable. There can be no doubt that the highest gains are achievable using turbo codes or concatenated codes. If the gain requirement is less stringent, convolutional codes with hard-decision sequential decoding or soft-decision Viterbi decoding (to be described in Chapter 2) provide the highest gains on a Gaussian channel.

1.15.7 Channel error mechanisms

Ideally one would design a coding scheme for the precise conditions encountered on the channel. In practice, the channel may not be well characterized and the coding scheme may have to show flexibility to cope with the range of possible conditions. For slowly varying channel conditions which exhibit approximate Gaussian conditions over appreciable periods of time, adaptive coding schemes are a natural choice. These often use variable rate convolutional codes, or they may be based around ARQ/FEC hybrids. For channels which may fluctuate rapidly between states, producing mixtures of bursty and random errors, a wide variety of diffuse-error correcting schemes, including interleaving, are available. Reed Solomon codes may also be considered to fall into this category; although optimized neither for random errors or general bursts, their low redundancy overhead makes them a good choice for a variety of channel conditions.

1.15.8 Cost

Any error control scheme is merely a part of a larger system and its costs must be in proportion to its importance within the system. Bearing in mind that error rates may be reduced by the use of higher transmitted power, the aim of coding is to be more cost-effective than other solutions. That, however, is often not the main way in which cost constraints are experienced in a coding system; the major part of the costs of error control are incurred at the decoder, placing the burden of the economics onto the receiving equipment. Since the owners of transmitting and receiving equipment may be different, the economic considerations may not be combined to optimize overall system costs. Decoder costs must be assessed in terms of what the receiver will be prepared to pay.

A number of fairly straightforward rules may be stated. Firstly as previously indicated, the decoder costs dominate in a forward error correction scheme. Error detection is therefore much cheaper than error correction. High data rates will cost more than low data rates. Complex codes with multiple-error correction will cost more than simpler codes. For many applications, however, the main factor affecting cost will be whether there is a codec available commercially or whether it will need to be developed specially. Development costs must be spread across the number of receivers, and if the market is small or particularly cost-sensitive it may be impossible to develop a special codec for the particular needs of the application. In that case, the choice will be severely limited.

Any very specific advice about commercially available codecs would ensure that this book would quickly be out of date. As with all modern technologies, the product range is expanding and getting cheaper. Rate 1/2 Viterbi decoders are available and popular, and may incorporate puncturing for higher rates or for adaptable coding schemes (although the latter involve many other system complexities and costs). Certain Reed Solomon codes are being adopted as standard and codecs are becoming available. Often this will be a spin-off from a particular mass market, such as compact disc players.

Although it seems a shame to sound a negative note, I believe that many interesting ideas in error control will never be implemented simply because their potential market will not make the development costs worthwhile. Similarly many engineers working on error control techniques will never be allowed to design the best system technically; they will be forced to choose the best of what is available. Those who wish to have a relatively free hand should work on applications where the potential market is large or not very cost-sensitive. The same constraints apply, of course, in many other areas. Some would say that is what engineering is all about and error control is, after all, an engineering topic rather than a mathematical one. The mathematics is the servant of the engineer, and the engineering is the really difficult part.

1.16 CONCLUSION

In this chapter we have been concerned mainly with general concepts and background. There are several good modern books on digital communications that include a treatment of error control codes [2–4]. These can be consulted for more information about digital modulations implementation issues and applications. They can also be used to provide an alternative view of error control coding issues that will be treated later in this book. Another large subject given very brief treatment here is the general topic of information theory, and other sources [5, 6] are recommended for further reading.

The next chapter of this book deals with convolutional codes, which are the most commonly adopted codes in digital communications. Chapter 3 will cover linear block codes and a subset of these codes, cyclic codes, will be treated in Chapter 4. The construction of cyclic codes and the decoding methods for multiple-error correction require a knowledge of finite field arithmetic, and this is covered in Chapter 5. Chapter 6 then deals with BCH codes, a large family of binary and nonbinary codes, but concentrating on the binary examples. The most important nonbinary BCH codes are Reed Solomon codes and these are treated in Chapter 7. Chapter 8 then deals with performance issues relevant to all block codes. Multistage coding is introduced in chapter 9. Codes using soft-in-soft-out algorithms for iterative decoding are covered in Chapter 10.

It should not be assumed that the length of the treatment of different codes indicates their relative importance. Block codes have a very strong theoretical basis, but for many applications the chapters on convolutional codes and on iterative decoding will be the most relevant. Iterative decoding can, however, be applied to block codes and familiarity with Chapters 4 and 9 will certainly help in obtaining the most from Chapter 10.

1.17 EXERCISES

1 A transmitter sends one of four possible messages. The most likely occurs with a probability of 0.5 and is represented by the value 0. The second most likely occurs with a probability of 0.25 and is given the representation 10. The other two messages each occur with a probability of 0.125 and are given the representations 110 and 111. Find the mean information content of the messages and the mean number of bits transmitted. Compare with the case where a fixed length representation is used.

2 Find the value of E_r/N_0 needed by BPSK or QPSK modulation to achieve a bit error rate of 10^{-3} over an AWGN channel.

3 Find approximate optimum uniform spacing for 16-level quantization from a BPSK or QPSK receiver, assuming an AWGN channel with $E_r/N_0 = 2\,\text{dB}$.

4 Use Table 1.1 to carry out minimum distance decoding of hard-decision sequences 01011, 01110, 10100, 11000. Use the method of Section 1.8.3 to check the results.

5 Prove that, for a linear code, the distance structure is the same viewed from any codeword.

6 An uncoded channel needs E_b/N_0 of 2 dB to achieve a BER of 10^{-2} and 10 dB to achieve a BER of 10^{-5}. A rate 1/2 code subject to random bit errors with probability of 0.01 produces an output BER of 10^{-5}. What is the coding gain at 10^{-5} BER? If the coded channel operates at $E_b/N_0 = 5\,\text{dB}$, what is the uncoded BER?

7 Find the maximum coding gain that could be achieved for BER $= 10^{-6}$ using QPSK modulation over an AWGN channel and a rate 1/3 code.

8 A block code has $d_{\min} = 8$. Find the maximum guaranteed error detection if maximum error correction is to be carried out. How would this change if only single-error patterns were to be corrected? Find the amount of error correction achievable if an *extra* three bits in error should be detectable.

9 A source transmits a sequence of numbered frames, inserting repeats of previous frames as required for ARQ. The second, sixth and ninth frames transmitted are corrupted on reception. Show the flow of frames between the transmitter and receiver to support a GB3 protocol.

10 Assuming the same propagation delays and the same corrupted frames as question 9, find how many frame intervals would be needed to transfer 7 frames if SR-ARQ were used, assuming

(a) an infinite receiver window
(b) a minimum receiver window to operate SR-ARQ.

Quantify the receiver window size needed for part (b).

1.18 REFERENCES

1 J. Massey, *Coding and Modulation in Digital Communications*, Proceedings of the International Zurich Seminar on Digital Communications, 1984.
2 S. Haykin, *Communication Systems*, 4th edition, John Wiley & Sons, 2001.
3 B. Sklar, *Digital Communications: fundamentals and applications*, 2nd edition, Prentice Hall, 2001.
4 J.G. Proakis, *Digital Communications*, 4th edition, McGraw Hill, 2001.
5 S.W. Golomb, R.E. Peile and R.A. Scholtz, *Basic Concepts in Information Theory and Coding*, Plenum Press, 1994.
6 R.E. Blahut, *Principles and Practice of Information Theory*, Addison Wesley, 1987.