Chapter 1

Introduction to Coded Digital Communications

The basics of error-control coding (or also called channel coding) and the principal characteristics of transmission channels are the subject of this introductory chapter. We start with a description of the key role and the fundamental relevance of coding for digital communication systems, and list some of the many applications for which error-control coding is of advantageous. An overview of the elements of typical communication systems is given, focusing on the stochastic description of discrete transmission channels including memoryless channels and the additive white Gaussian noise channel. The principles of error-control coding are introduced with block codes as one of the two major classes of channel codes. The chapter continues with the properties of Hamming distances and the minimum distance of a block code. The maximum-likelihood decoding rule for optimum receivers is deduced for the general discrete channel as well as for hard- and soft-decision decoding. The asymptotic coding gain is an important performance measure of a code. Finally, the fundamental concept of error-control coding is illustrated by a comparison to the basic principle of quantization for digital transmission.

1.1 Coding for Reliable Digital Transmission and Storage

The founder of the modern information and coding theory is Claude E. Shannon, who published two famous, pioneering papers entitled *A Mathematical Theory of Communication* and *Communication Theory of Secrecy Systems* in 1948 and 1949 in the Bell Systems Technical Journal, a reprint of which can be found in [129, 130]. More than half a century later, the Shannon information theory is still the foundation discipline for communication systems and determines the way we think of digital signaling with regard to a secure, reliable and efficient transmission.

The data sources as well as the transmission channels are described by stochastic models. A mathematical measure of information, the entropy, assigns a certain information content to each message. This enables us to determine the minimum number of symbols necessary for an error-free representation of the message. Each longer message with the same amount of information contains redundancy. The information theory relates the information content of a message to the information-carrying ability of the transmission channel, where the performance is defined in terms of error probabilities. Basically, coding is the mapping of a message to a set or sequence of symbols. Shannon's theory comprises three different types of coding:

Source coding. The source information stream is compressed so that no or only less significant information is lost, enabling a perfect or almost perfect restoration of the information. One advantage being that fewer symbols have to be transmitted. Thus, source coding eliminates superfluous and uncontrolled redundancy and reduces the load on the transmission system.

Error-control coding (also called **channel coding**). Error-control coding offers methods to transmit information from a source to a sink with a minimum of errors, in conjunction with lower transmit power and perhaps even less bandwidth. On the transmitter side, redundancy is added to the actual information in a controlled fashion (either as additional parity-check symbols or by expanding the modulation alphabet), so that the receiver can detect and correct the transmission errors. This guarantees high reliability of the transmitted information. Furthermore, it is possible to cancel out the effect of interference from external sources which could not be achieved by simply increasing the transmit power.

Secrecy coding (also called **cryptography**). The information is encrypted to make it unreadable to unauthorized persons or to avoid falsification or deception during transmission. For channel coding the messages are to remain readable even in the case of interference, however, for secrecy coding the encrypted messages are to be unreadable without the knowledge of the key even in the case of perfect transmission.

This book describes error-control coding schemes in connection with digital communications and modern modulation methods while observing the elementary bounds and potentialities for reliable communication described by the Shannon information theory.

Since 1948, error-control coding has developed into an application oriented science, mainly influenced by the interaction of theory and practice. Many modern digital communication systems (such as mobile radio, modems for copper lines, high-rate access systems or deep-space satellites) only achieve their enormous performance by using error-control coding, especially for channels subject to interference or where high reliability is an important issue. A special form of transmission (from *here* to *there*) are mass-storage applications (from *now* to

later), since noise and interference, similar to that during transmission, are possible during write (input) and read (output) operations. Efficient storage media with very low error-rate requirements also require error-control mechanisms.

Error-control coding is not only of practical importance, but also a very interesting science for those interested in communications and information engineering as well as for mathematicians, because the coding schemes, the algorithms and the Shannon information theory are based on a challenging and complex but elegant theory. This could be highlighted by some important key words, including the basics of communication systems; digital modulation schemes; protocols for IP and ATM packet data services; probability theory, stochastics and power spectral densities; matrices and vector spaces; the algebra of finite fields, principal ideal rings and polynomial rings; Fourier transforms on Galois fields; special metrics (including Hamming distance, Viterbi metric and a so-called soft algebra); the analysis and synthesis of shift registers; finite state machines; trellis diagrams; efficient algorithms and structures.

In the aforementioned paper from 1948, Shannon proved that each channel is described by a numerical parameter called *channel capacity*. By using channel coding, arbitrary small error rates can be guaranteed if the data rate is smaller than the channel capacity and the required processing complexity in the transmitter and receiver is small enough to allow a reasonable implementation. So the channel properties do not restrict the quality of the transmission, but only the throughput.

However, the Shannon information theory only proves the *existence* of powerful codes, but does not provide any practical rules for the *construction*. In the years following 1948, no one actually managed to find the theoretically predicted codes in practice, however, at that time technology was not sufficiently advanced to allow the realization of very complex coding schemes. After a period of disillusionment about the value of information theory further research concentrated on finding codes that were at least realizable, even if they did not reach the information theoretical bounds. The main objective was the improvement of coded over uncoded transmission, especially with regard to the reduction of the error rate or of the transmit power.

Meanwhile a multitude of special codes have been found and analyzed. However, only few codes are of great practical importance. These include the RS and BCH block codes, for instance, as well as some relatively simple convolutional codes, all of which will be primarily discussed here. The technique of concatenating codes and powerful algorithms for decoding have at least reduced the gap between coding schemes which allow a reasonable implementation and the bounds predicted by the Shannon information theory.

There are two main classes of codes, block codes (BC, Chapters 1 to 8) and convolutional codes (CC, Chapters 9 and 10). Both classes will turn out to be completely different in theory and practice. As in all other textbooks, the block codes are described at the beginning, as they help to explain a lot of elementary topics (Chapter 1) as well as the Shannon information theory (Chapter 3). For

the latter, only little knowledge of block codes as presented in Chapter 1 is needed, in contrast to the RS and BCH block codes (Chapter 8) which have a very complex mathematical structure (Chapter 7). The interaction of error-control coding and details of the transmission system will become clearer with the discussion of convolutional codes which will follow.

Block codes and convolutional codes together with simple modulation schemes could be called the *classic error-control techniques*. The redundancy added during encoding increases the data rate, so that the transmission channel has to be used more often and more bandwidth is needed. However, this only provides *power-efficient* but not *bandwidth-efficient* transmission techniques. The pioneering works of G.Ungerböck, known as TCM (Trellis Coded Modulation, see Chapter 11) since 1982, allow a power-efficient transmission without having to increase the bandwidth. By using TCM the error rate and the required transmit power (and in extreme cases even the required bandwidth) can be reduced. TCM is primarily based on convolutional codes, but also partly on block codes.

There are two basic principles of error-control coding which depend on the requirements of the tolerable end-to-end delay and whether a feedback channel for the reverse direction is available:

Forward Error Correction (FEC). The redundancy added in the transmitter is used to *correct* transmission errors in the receiver. As *error-correction codes* (ECC), block codes and convolutional codes as well as trellis coded modulation are used. Later on, we will show that convolutional codes and the TCM technique should better be referred to as *transmission codes*.

Automatic Repeat Request (ARQ). Little redundancy is added in the transmitter, as the receiver only uses an *error-detection code* (EDC) to *detect* but not to *correct* transmission errors. If errors are detected, a request is issued via the feedback channel in the reverse direction to either repeat the message or to add further redundancy to the message. Almost always block codes are used for error detection. However, there must be sufficient end-to-end time available and a reliable feedback channel in the reverse direction.

The advantage of ARQ is that for error detection less redundancy has to be transmitted than for error correction. However, if messages have to be transmitted repeatedly the delay can become quite large. Thus the throughput with ARQ depends on the quality of the channel whereas the error rate does not. These properties are reversed for forward error correction (presupposing a fixed coding scheme), since the channel quality determines the error rate, but not the throughput. FEC and ARQ techniques can also be combined, for example by dimensioning the redundancy such that a small number of errors are correctable, but that for a larger number of errors a retransmission is requested. In this book we will primarily discuss FEC techniques, however, in Chapter 15? we also take a look at ARQ techniques.

The design of high-performance coding techniques should always be oriented towards the specific constraints of a transmission system and especially to the properties of the transmission channel. So specific applications require specific codes. Some of the most important constraints which have to be considered for the selection and optimization of a coded transmission system are

- the properties of the physical transmission channel (particularly the available bandwidth, the signal-to-noise ratio, and possible distortions or other imperfections);

- the existing modulation scheme, if coding and modulation can not be optimized jointly;

- the requirements for the error probability and the error structure after decoding (the difference between single random errors and burst errors could be relevant);

- the available transmit power (where limitations could be required to simplify the power amplifier, to extend the battery lifetime in case of mobile terminals, or to minimize the interference to other systems);

- the acceptable delay caused by error-control coding (where the delay can be measured with regards to time or symbols, also the difference between a continuous data stream and individually transmitted data packets could be relevant);

- the bounds on the complexity of the signal processing in transmitter and receiver (also including requirements for synchronization)

as well as many other requirements. All this results in a quite complex task with different individual solutions depending on the weighting of the various constraints. The following listing provides an overview of the various applications and features of error-control coding.

- To save *transmit power*, for example if the physical channel has near ideal AWGN characteristics, which leads to statistically independent random single errors. Typical applications are geostationary communication satellites and especially deep space research satellites (see Section 16.1?).

- To save *bandwidth* by making sure that the redundancy added by encoding does not cause more symbols to be transmitted, but uses symbols from an increased modulation alphabet. An important application are modems for telephone channels including copper subscriber lines (see Section 16.2?).

- To provide *both power-efficient and bandwidth-efficient* communication, this is especially requested for digital mobile radio (see Sections 16.3?, 16.4?).

- For *time-variant* channels, if there are typically good and bad parts within a single codeword and if the receiver can estimate the current signal quality well enough. A typical application is mobile radio, where fading events can occur that are short compared to the block length of a codeword and long compared to the symbol duration.

- For channels with *burst errors* which typically occur for mass-storage applications like compact discs (see Section 16.6?) or for magnetic recording.

- For *very high reliability* as required for the data exchange in computer networks, for electronic financial transfers between banks, for safety-relevant services like remote control of railway networks, or generally for high-compressed data. High reliability often occurs in conjunction with high security, so error-control coding as well as secrecy codes are required.

- For source coding schemes generating symbols with *different demands on error protection*. A typical example are speech coders used in mobile radio, where the bits in a speech frame are of different relevance for the human ear (unequal error protection, see Subsection 9.3.3 and Sections 16.3?, 16.4?).

- For *error detection* instead of error correction. In combination with source coding of speech and music, error *concealment techniques* are also used to enhance the quality impression.

- In combination with *ARQ techniques* as explained above.

- For those *interferences* which can not be reduced by increasing the transmit power, for example interference from adjacent channels, click noise, slow and fast fading, multipath propagation and linear and non-linear distortions.

- To realize higher system capacity by *adaptive modulation and coding* and better statistical multiplex gains in fixed wireless cellular access systems based on point-to-multipoint architectures (see Section 16.?).

- To reduce the *error floor*, which occurs in transmitters and receivers which are not perfect, for example due to non-linearities (see Section 16.5?).

1.2 Elements of Digital Communication Systems

1.2.1 Basic Elements and Principles

The basic principle of a digital transmission system with source coding and error-control coding is shown in Figure 1.1. As mentioned above, first source coding eliminates redundancy and then error-control coding adds redundancy in

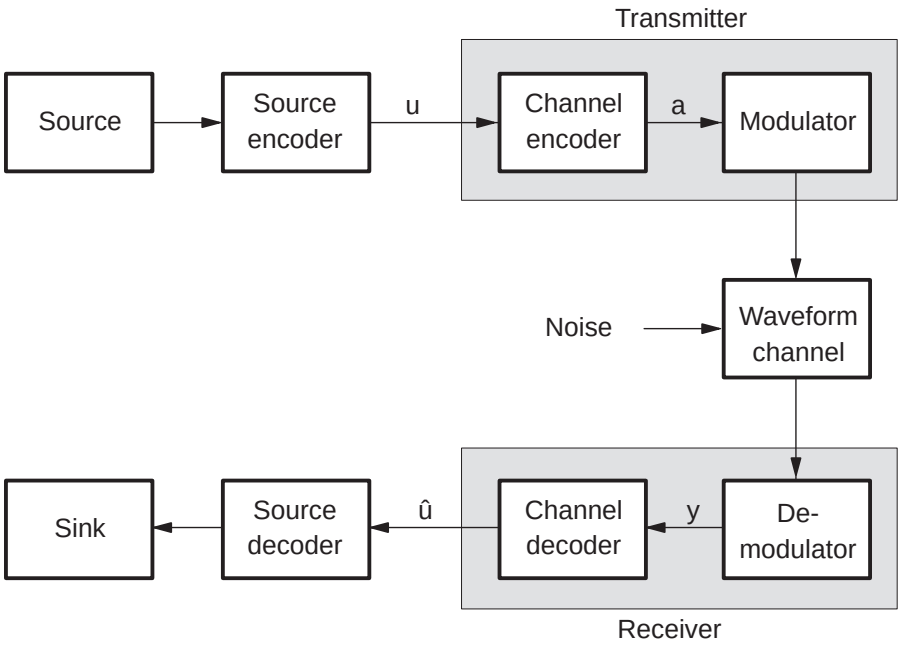

Figure 1.1. A digital communication system with source and channel coding

a controlled manner. Redundancy which may already be in the source data is useless for error-control coding, since the properties of this redundancy are not exactly controllable. The source data may even have no redundancy at all.

According to the Shannon information theory [129, 130] introduced in Chapter 3, source coding and error-control coding can be executed and optimized separately as shown in Figure 1.1, this is also called the *separation principle*. However, bounds on the *transmission delay* (also called *latency*) or on the processing complexity or other practical constraints can require the coordination and joint optimization of source coding and error-control coding (for further detailed considerations of this specific issue see Subsection 9.3.2 on RCPC codes and Sections 16.3? and 16.4? on source-channel coding for mobile radio).

Each *encoder* operation has a corresponding *decoder* operation as its counterpart. The term *coder* is not used. *Encryption*, which is not listed in Figure 1.1 but included in Figure 1.2, is usually placed between the source coding and the error-control coding. In the following, source coding and encryption will be ignored so that error-control coding could be addressed as *coding*.

The symbols u are called *information symbols* or, for the binary case, *information bits*. The names *source symbols* or *source bits* are also quite common. The *encoder* transforms the information symbols or information bits u into the *encoded symbols* or *encoded bits* a and adds redundancy which increases the data rate.

The encoder is connected to the *waveform channel* (there are also many other names like *physical channel, continuous channel, transmission channel, analog channel*) by the *modulator*. The waveform channel can not transmit any

discrete-time symbols but only continuous-time signals. Thus, the modulator's function is to assign such signals to the discrete values a, which can then be transmitted over the waveform channel. It also adapts the modulated signal to the *frequency range* of the waveform channel, and in particular shifts the baseband signal into the passband frequency range. However, for the technique of *trellis coded modulation* (TCM, see Chapter 11) it is not necessary to split the transmitter into an encoder and a separated modulator as shown in Figure 1.1.

The *waveform channel* generally does not behave ideally; the signals are altered, disturbed and deteriorated during the transmission. This applies for all types of transmission channels, including

- cables (e.g., subscriber line, coax cable or fiber),
- terrestrial radio channels (e.g., mobile communications, microwave line-of-sight radio, broadcast radio or short-wave radio),
- geostationary or deep-space satellite links,
- mass storage systems as a specific form of information transmission (e.g., magnetic recording, electronic or optical devices)

as well as for any combination of these channels. The waveform channel is characterized, for example, by thermal noise, by intersymbol interference and multipath radio propagation, by fading effects, by co-channel and adjacent-channel interference, by linear or non-linear amplitude and phase distortions, by atmospheric effects, as well as by deliberate or hostile interference (jamming).

So the *demodulator* does not receive the exact transmitted continuous-time signals, but only corrupted versions of them. However, the receiver should still be able to reconstruct the transmitted discrete-time information symbols from the received signal. In the demodulator the baseband signal is first reconstructed from the received passband signal, this includes carrier, phase and clock synchronization for coherent receivers. From the baseband signal a sequence of discrete-time symbols is derived, which we refer to as *received symbols* y, so that each y exactly corresponds to one transmitted symbol a. It is not necessarily the case that the ranges of y and a are identical. In case of *soft-decision demodulation* the demodulator is to produce values y which are to contain as much information as possible for the decoder. Hence, for example, a combination of binary a and real-valued y can be reasonable.

The *decoder* operates on the discrete-time symbols y which may be continuous-valued or discrete-valued to derive estimations $\hat{u}$ for the transmitted information symbols u. This estimation process generally causes a *decoding delay*, in particular, since the decoder should work ideally on very long *symbol sequences*, i.e., after receiving a whole sequence of received values, a whole sequence of information symbols is estimated in one step.

The modulator and the demodulator only handle individual separated symbols and know nothing of the properties of the channel encoder nor of the properties of whole symbol sequences, in other words the modulator and the demodulator are memoryless. In contrast, the encoder introduces memory to

the transmit signal and the decoder uses this memory for its task to derive best possible estimations.

1.2.2 Further Details of Mapping and Modulation

A digital transmission system should be designed according to the expected properties of the waveform channel to achieve an optimum performance of the entire system. Therefore we will not only discuss the design and the analysis of error-control codes but also the modeling of waveform channels and discrete channels.

Between the pair encoder-decoder and the transmission channel in Figure 1.2, we have now placed the pair mapping-demapping. In contrast to the abstract communication model shown in Figure 1.1, we will now elaborate on further important details, particularly the relation between modulation and coding schemes for a typical digital wireless passband transmission system. Also we compare the size of the range of symbols used by the encoder to the size of the modulation alphabet.

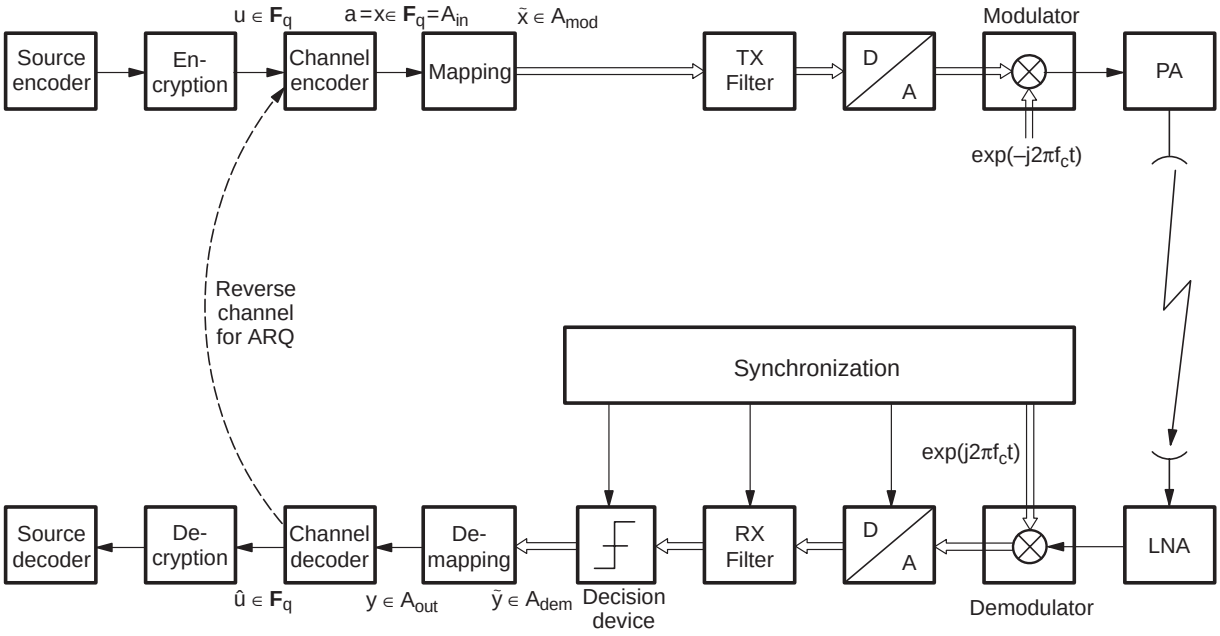

Figure 1.2. A detailed block diagram for a typical wireless digital transmission

As in Figure 1.1, the transmitter branch in Figure 1.2 contains the *source encoder* and the *channel encoder*, however, with the *encryption* operation added. The following *mapping* is discussed further below in this subsection. The *pulse shaping and transmit filter* are represented by the block TX filter and are assumed to take place in the digital domain prior to the *digital-to-analog conversion* (DAC). The transmitter filter converts a symbol stream into a continuous-time signal, this is explained in detail in Subsection 2.2.1. By multiplication

with $\exp(-j2\pi f_c t)$ the *modulator* shifts the frequency from the baseband domain to the passband domain centered around the *carrier frequency* f_c. Finally, the *power amplifier* (PA) generates the desired level of the signal which is then finally radiated from the *transmit antenna*.

In case of passband transmission the equivalent baseband signal is usually complex-valued. A complex-valued signal can also be described by a 2-dimensional real-valued signal, but the formal description is simplified by the complex notation. In Figure 1.2, the complex-valued (2-dimensional) symbols and signals are represented by double arrows, whereas real-valued (1-dimensional) symbols and signals are represented by single arrows. More details of passband transmitters and receivers are shown in Figure 2.2. A compact but serious introduction to passband modulation schemes is given in Chapter 2 and comprehensive and detailed introductions can be found in almost every textbook on digital communications, e.g., [114].

In the receiver path, the signal from the *receiver antenna* is passed to the *low-noise amplifier* (LNA). Due to basic physical principles it is inevitable that the LNA adds unwanted thermal noise to the received signal, this noise is represented very accurately by the *additive white Gaussian noise* (AWGN) channel model, which is introduced in Subsection 1.3.3 for baseband signaling and in Chapter 2 for passband signaling. The AWGN channel is of supreme importance for digital communications in general. By multiplication with $\exp(+j2\pi f_c t)$ the *demodulator* shifts the frequency from the passband domain back to the baseband domain. After *analog-to-digital conversion* (ADC), the baseband signal is passed to the *digital receiver filter* (RX filter), which is correspondingly *matched* to the transmitter filter as explained in Subsection 2.2.1 and also addressed in Problem 2.1.

The *demapping*, the *channel decoder*, the *decryption* and the *source decoder* are the corresponding counterparts to the transmitter path. The task of *synchronization* is to find the correct carrier frequency and carrier phase (assuming a coherent demodulator, see Subsection 2.2.1) and the correct sampling clock which is at least required by the decision device, but maybe also required by the ADC and the digital receiver filter, depending on the details of the implementation.

As can be seen by comparing Figures 1.1 and 1.2 the term *modulator* is used in two different ways. One is the complex multiplication for the frequency shift, the other comprises all elements between the mapping and the waveform channel. The term *demodulator* is used equivalently.

The order of the elements in the transmitter and the receiver can also be different than assumed in Figure 1.2. For example, the modulation operation can be spread across multiple intermediate frequencies. The transition from the digital to the analog domain can take place in the baseband domain, on one of the intermediate frequencies or possibly in the passband domain. In real systems filter operations will be used at almost any stage, particularly after the DAC in the transmitter and before the ADC in the receiver, which is not explicitly shown

in Figure 1.2. The transition from discrete-valued and discrete-time symbols to continuous-valued and continuous-time waveforms in the transmitter takes place during the DAC at the latest. However, the transition could also be understood as taking place in the transmitter filter, see also Figure 2.2 and the introduction to baseband and passband signaling in Section 2.2.

The arched arrow in Figure 1.2 implies a reverse channel via which the receiver can request the transmitter to repeat the whole message or simply parts of it. Such methods are called *automatic repeat request* (ARQ) and are usually placed between encryption and channel encoder as shown in the figure. ARQ schemes are covered in Chapter 15?.

Let us take a look at the mapping, the decision device und the demapping, the parts of Figure 1.2 which are the most interesting in this context. We will see that the input and output symbols of the channel encoder as well as the output symbols of the decoder are selected from a finite alphabet, which will turn out to be a *finite field* or *Galois field* denoted $\mathbb{F}_q$ with the cardinality of

$$|\mathbb{F}_q| = q = p^m. \tag{1.2.1}$$

The Galois fields will be introduced in Chapter 4 and discussed in detail in Chapter 7. The field $\mathbb{F}_q$ is the range for the information symbols u, for the encoded symbols $a = x$ and for the estimated information symbols $\hat{u}$, hence $u, a, \hat{u}$ are all q-ary. There are several cases to be considered:

- $q = 2$ is the simplest case of *binary codes*, where the symbols are simply bits. In Chapters 1 to 5, although we consider only binary codes as examples, we will develop the theory for the general case.
- $q = p^m$ is the general case, where p is prime and m is a non-negative integer. We will have to wait until Chapter 7 to understand the background of this.
- $q = 2^m$ is the normal case for most codes, where the symbols are groups of bits, for the important case of $m = 8$ the symbols are bytes.

The actual values within $\mathbb{F}_q$ are irrelevant, only the definition of the operations of addition and multiplication *between* the elements of $\mathbb{F}_q$ are important. So it does not matter whether for $\mathbb{F}_4$ we use the set $\{0, 1, z, z^2\}$, $\{A, \alpha, 7, \pi\}$ or $\{00, 01, 10, 11\}$.

Throughout the entire book, the encoded symbols are denoted a as well as x and $\mathcal{A}_{\text{in}}$ is often used instead of $\mathbb{F}_q$. This may seem confusing at first, but is very important, since we use $a \in \mathbb{F}_q$ to denote the symbols as the output of the channel encoder, but $x \in \mathcal{A}_{\text{in}}$ to emphasize the input of the mapping. The term $\mathbb{F}_q$ denotes the Galois field with $q = p^m$, but $\mathcal{A}_{\text{in}}$ refers to a general q-ary alphabet, however, without taking advantage of the special form of $q = p^m$. So,

$$\begin{aligned} &a = x \in \mathbb{F}_q = \mathcal{A}_{\text{in}}, \quad y \in \mathcal{A}_{\text{out}}, \\ &|\mathcal{A}_{\text{in}}| = q. \end{aligned} \tag{1.2.2}$$

The mapping assigns each q-ary *encoded symbol* $x \in \mathbb{F}_q$ of the Galois field to a 2^M-ary *modulation symbol* $\tilde{x} \in \mathcal{A}_{\text{mod}}$ of the *modulation alphabet*. In contrast to the $\mathbb{F}_q$-based operations in the encoder and the decoder, the digital transmitter filter and the modulation that follows operate with real- or complex-valued symbols and signals. Thus, the modulation alphabet $\mathcal{A}_{\text{mod}}$ is a subset of the real or complex numbers, depending on whether we are considering baseband signaling (also called 1-dimensional signaling) or passband signaling (also called 2-dimensional signaling):

$$\begin{aligned} &\tilde{x} \in \mathcal{A}_{\text{mod}} \subset \mathbb{R} \text{ or } \mathbb{C}, \quad \tilde{y} \in \mathcal{A}_{\text{dem}} \subset \mathbb{R} \text{ or } \mathbb{C}, \\ &|\mathcal{A}_{\text{mod}}| = 2^M \end{aligned} \tag{1.2.3}$$

After the decision device in the receiver branch we have the symbols $\tilde{y} \in \mathcal{A}_{\text{dem}}$ of the *demodulator alphabet*, which are then assigned to the *output symbols* $y \in \mathcal{A}_{\text{out}}$ by the demapping. Like $\mathcal{A}_{\text{mod}}$, $\mathcal{A}_{\text{dem}}$ is also a subset of the real or complex numbers. The 2^M-ary modulation symbols of the alphabet $\mathcal{A}_{\text{mod}}$ are called *encoded modulation symbols* or simply *symbols*. Particularly for the trellis coded modulation (TCM) introduced in Chapter 11, other terms such as *signal point* or simply *signal* are used, then $\mathcal{A}_{\text{mod}}$ is also called *signal constellation*. Throughout the entire book we shall stick to $|\mathcal{A}_{\text{mod}}| = 2^M$, only for TCM will we compare uncoded 2^M-ary to coded 2^{M+1}-ary modulation schemes. Usually M is integer, but there are some exotic modulation schemes where the number of modulation levels is not a power of two, so we can include these cases by simply also allowing non-integer values of M.

Example 1.1. Let us consider some examples of the mapping scheme for various combinations of q and M.

(1) The simplest case is the full binary case where $q = 2$ (binary coding) with $\mathbb{F}_2 = \mathcal{A}_{\text{in}} = \{0, 1\}$, for example, and $M = 1$ (binary modulation) with $\mathcal{A}_{\text{mod}} = \{+\sqrt{E_c}, -\sqrt{E_c}\} \subset \mathbb{R}$, for example. For this example the mapping can be described by the simple function

$$x \in \mathcal{A}_{\text{in}} \quad \mapsto \quad \tilde{x} = (2x - 1)\sqrt{E_c} \in \mathcal{A}_{\text{mod}}. \tag{1.2.4}$$

(2) A binary channel encoder can be combined with a high-level modulation. For example, if $q = 2$ and $M = 5$, then five consecutive bits of the channel encoder are mapped to a 32-ary modulation symbol. The mapping reduces the symbol rate (i.e., the number of modulation symbols per second) by a factor of 5.

(3) A realistic example would be the combination of $q = 256 = 2^8$ and $M = 2$. For 4-PSK modulation (see Section 2.3) we could choose $\mathcal{A}_{\text{mod}} = \{1, j, -1, -j\} \subset \mathbb{C}$ or as in Figure 2.5 $\mathcal{A}_{\text{mod}} = \{(1 + j)/\sqrt{2}, (-1 + j)/\sqrt{2}, (-1 - j)/\sqrt{2}, (1 - j)/\sqrt{2}\} \subset \mathbb{C}$, possibly both with an additional factor of $\sqrt{E_c}$ which will become clear in Subsection 1.3.3. Then a 256-ary symbol of the channel encoder with 8 bit is mapped to four 4-PSK modulation symbols. Here, the mapping increases the symbol rate by the factor 4.

(4) Now consider the combination of $q = 256 = 2^8$ and $M = 6$. Three 256-ary symbols of the channel encoder with $3 \cdot 8 = 24$ bit are mapped to four 64-ary modulation symbols with $4 \cdot 6 = 24$ bit. The mapping increases the symbol rate by the factor 4/3. ■

So depending on the specific conditions, we either have $q = 2^M$, $q < 2^M$ or $q > 2^M$. Thus the mapping can either not influence the symbol rate at all or reduce or increase it by the factor $(\log_2 q)/M$. From now on the term *symbol rate* r_s shall always refer to the modulation symbols:

$$r_s = \text{number of } 2^M\text{-ary modulation symbols per second} \tag{1.2.5}$$

$$= \frac{\log_2 q}{M} \cdot \text{number of } q\text{-ary encoded symbols per second} \tag{1.2.6}$$

$$= \frac{1}{T_s}. \tag{1.2.7}$$

The symbol rate is also called the *baud rate* in units of "baud", named after the French engineer Baudot. This is the number of uses of the waveform channel per second. The inverse of the symbol rate is equal to the duration T_s of a modulation symbol, in other words, a symbol is transmitted once every T_s seconds. We refer to Section 2.2 for more details on this issue.

1.2.3 Abstract and Inner Discrete Channel

The combination of all elements of the communication system between mapping and demapping including the pair mapping-demapping itself is called the *abstract discrete channel* (abstract DC) with $\mathcal{A}_{\text{in}}$ and $\mathcal{A}_{\text{out}}$ as input and output alphabets. Without the pair mapping-demapping it is referred to as the *inner discrete channel* (inner DC) with $\mathcal{A}_{\text{mod}}$ and $\mathcal{A}_{\text{dem}}$ as input and output alphabets. The choice of which channel model we use at any one time depends on our objectives as follows.

Abstract DC with q-ary input. A general channel with arbitrary input and output alphabets is assumed, its inner structure is completely ignored, which can make it be completely different from the example of Figure 1.2. Without considering the details of the transmission system, everything between the output of the encoder and the input of the decoder is modeled as an abstract black box. Then the discrete channel is only characterized by how the q-ary input symbols $x \in \mathcal{A}_{\text{in}}$ are mapped to the output symbols $y \in \mathcal{A}_{\text{out}}$. In Subsection 1.3.1 we will describe this matter in detail with conditional probabilities.

This abstract approach for the discrete channel is the foundation for Chapters 4 to 8 and partially for Chapter 3, where we are interested in the design and the analysis of the performance of high-sophisticated error-control

schemes, particularly in block codes and even more so in hard-decision decoders where $\mathcal{A}_{\text{mod}} = \mathcal{A}_{\text{dem}}$. Therefore it is appropriate to consider the pairs mapping-demapping and modulator-demodulator as invisible parts of the abstract discrete channel. The abstract DC is also called *digital channel* or *coding channel*.

The generalized abstract discrete channel is mainly important for theoretical considerations and for the derivation of principal concepts. However, according to Figure 1.2, in the practice of data transmission systems, there usually is a noisy channel, which is used with an especially selected modulation system. As already mentioned, thermal noise is almost always modeled as an additive white Gaussian noise (AWGN) channel.

Inner DC with a specific 2^M-ary modulation system. A physical channel with a modulation system similar to that of Figure 1.2 is assumed and the transmission path is modeled as an AWGN channel. The 2^M-ary modulation alphabet $\mathcal{A}_{\text{mod}}$ and also $\mathcal{A}_{\text{dem}}$ are subsets of the real numbers for baseband signaling or of the complex numbers for passband signaling.

This approach for the discrete channel based on the AWGN model is the foundation for Chapters 2, 9, 10, 11 and partially for Chapter 3, and also always if we are considering the joint optimization of error-control coding and modulation schemes. Here the number of modulation levels as well as the modulation scheme and in particular the pair mapping-demapping are very important. Finally, it is important to distinguish between baseband and passband signaling.

As already mentioned, it is irrelevant which elements represent the Galois field $\mathbb{F}_q$, only the definition of the algebraic operations between the elements is important. Similarly, for the discrete channel only the statistical relationships between input and output symbols are relevant, and at least for the abstract discrete channel the alphabets as such are irrelevant. Of course for the AWGN channel these statistical relationships depend on how $\mathcal{A}_{\text{mod}}$ is selected as subset of $\mathbb{R}$ or $\mathbb{C}$.

Ideally, for the AWGN channel there are usually continuous-valued symbols at the input of the *decision device* in the receiver branch, which are, of course, real- or complex-valued corresponding to $\mathcal{A}_{\text{dem}}$. However, after the ADC the digital symbols are only of a limited amplitude resolution due to implementation restrictions. For the function of the decision device and the *demapping* there are two different cases to be distinguished:

Hard decisions. The continuous-valued input symbols are mapped to symbols of the input alphabet $\mathcal{A}_{\text{in}}$ in the decision device, for example by using a hard quantization operation. The objective is to make the input of the demapping come as close to the output of the mapping as possible, i.e., as few errors as possible are to occur. The demapping performs the exact inverse operation to the mapping and again generates symbols of $\mathbb{F}_q$. So,

$$\mathcal{A}_{\text{out}} = \mathcal{A}_{\text{in}} = \mathbb{F}_q, \quad \mathcal{A}_{\text{dem}} = \mathcal{A}_{\text{mod}}. \tag{1.2.8}$$

Ideal soft decisions. There is absolutely no quantization in the decision device, the extreme case is that the continuous-valued symbols are simply passed on to the demapping. As mentioned in Subsection 1.2.1, it can be useful that the decoder obtains as much channel state information as possible from the input y, whereas in the case of hard quantization in the decision device useful information for the decoder might be lost. In the demapping device the real- or complex-valued symbols (in case of baseband or passband signaling) are simply passed on together with a possible symbol rate conversion. So

$$\begin{array}{ll} |\mathcal{A}_{\rm dem}| > |\mathcal{A}_{\rm mod}| = 2^M, & \mathcal{A}_{\rm dem} \subseteq \mathbb{R} \text{ or } \mathbb{C}, \\ |\mathcal{A}_{\rm out}| \ > |\mathcal{A}_{\rm in}| = q, & \mathcal{A}_{\rm out} \subseteq \mathbb{R} \text{ or } \mathbb{C}. \end{array} \tag{1.2.9}$$

For ideal soft-decisions even $\mathcal{A}_{\rm dem}$ and therefore also $\mathcal{A}_{\rm out}$ could be continuous-valued, ignoring for the moment the technical restrictions mentioned above, such as, for example, a finite wordlength after the ADC. Of course, apart from this case on the one hand and hard decisions with $\mathcal{A}_{\rm out} = \mathbb{F}_q$ on the other hand, all cases in between are also possible. An important, illustrating example with octal quantization is given in Subsection 1.3.5. By the way, in the case of ideal soft decisions, the term *decision device* is obviously exaggerated, since its only non-trivial operation is the sampling.

1.3 Discrete Channel Models

1.3.1 The Basic Concept of a Discrete Channel

As shown in Figure 1.3, the *abstract discrete channel* (abstract DC, also called *digital channel* or *coding channel*) is the combination of the mapping, modulator, waveform channel, demodulator and demapping. So the abstract DC formed by the modulation system is quite general, it might contain very complex modulation and synchronization schemes as we saw in the previous section. In this section we introduce a formal description of idealized discrete channels. To be absolutely clear, the term "discrete" refers to discrete-time symbols, and not to discrete-valued symbols.

The abstract discrete channel can be characterized formally by the triplet $(\mathcal{A}_{\rm in}, \mathcal{A}_{\rm out}, P_{y|x})$, where

(1) $\mathcal{A}_{\rm in}$ is the *input alphabet* of size q.

(2) $\mathcal{A}_{\rm out}$ is the *output alphabet*. As explained in Subsection 1.2.2 there are various cases for the demodulator (i.e., for the decision device contained therein):

Hard decisions means that the demodulator estimates the transmitted symbols a directly, hence $\mathcal{A}_{\rm out} = \mathcal{A}_{\rm in} = \mathbb{F}_q$. This is the case for most block codes, which are presented in Chapters 4 to 8 (with the exception of erasure

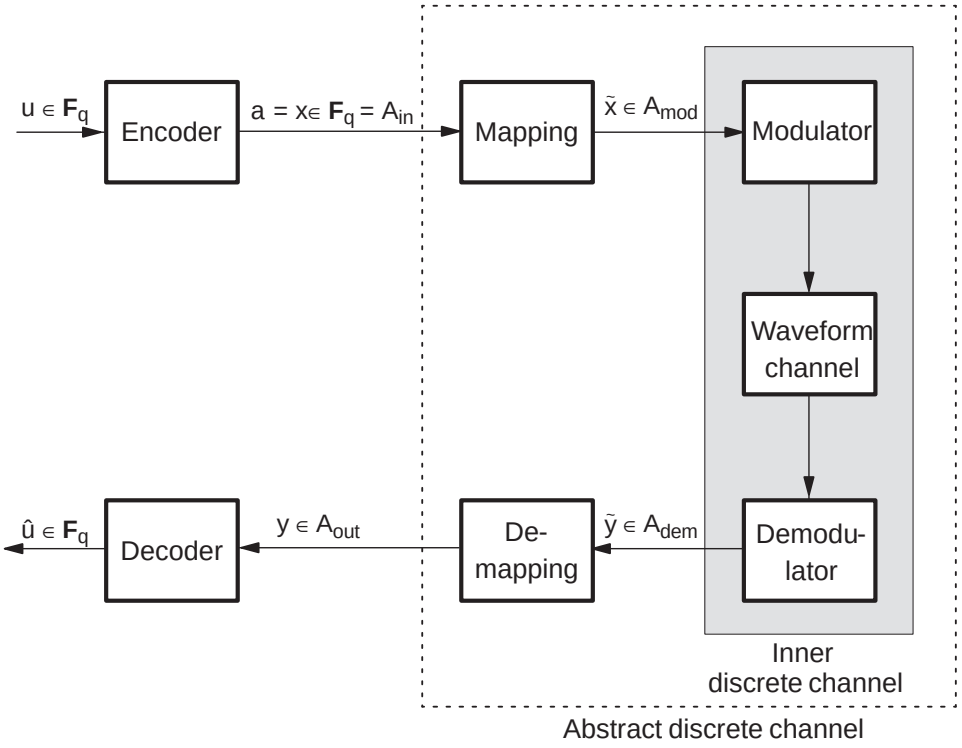

Figure 1.3. The discrete channel model as formed by the modulation system

decoding discussed in Section 8.6). The very simplest case of hard-decision demodulation are binary discrete channels with $\mathcal{A}_{\text{in}} = \mathcal{A}_{\text{out}} = \mathbb{F}_2 = \{0, 1\}$.

Soft decisions means that $\mathcal{A}_{\text{out}}$ contains more values than $\mathcal{A}_{\text{in}}$, for a continuous-valued demodulator output we may even have $\mathcal{A}_{\text{out}} = \mathbb{R}$ or $\mathbb{C}$. In many applications the demodulator can provide some information on the current quality or state of the channel, including the quality of its own decisions (see Section 10.?). For example, the demodulator tells the decoder with which certainty it made its decisions; that can vary from very reliable over uncertain to nearly random decisions. In principle, all coding schemes can use this kind of information, however, it is mainly only of real use for convolutional codes (see Chapters 9 to 11). A typical case is the combination of a binary $\mathcal{A}_{\text{in}} = \{0, 1\}$ and an octal $\mathcal{A}_{\text{out}}$, where the received symbols are quantized with 3 bits (see Figures 1.8, 1.9, 3.3 and Subsection 10.2.3?)

(3) $P_{y|x}$ is the *transition probability* (also called *channel statistic*). The more detailed term $P_{y|x}(\eta|\xi)$ denotes the conditional probability that $y = \eta$ is received if $x = \xi$ was transmitted.

The input x and the output y of the discrete channel are assumed to be random variables, their values are denoted as $\xi \in \mathcal{A}_{\text{in}}$ and $\eta \in \mathcal{A}_{\text{out}}$. The simpler denotation $P(y|x)$ is used, if it is not important to distinguish between the

random variables and their values. For the transition probability,

$$\sum_{\eta \in \mathcal{A}_{\text{out}}} P_{y|x}(\eta|\xi) = 1 \quad \text{for all } \xi \in \mathcal{A}_{\text{in}}. \tag{1.3.1}$$

The term *inner discrete channel* (inner DC) comprises the channel without the pair mapping-demapping, where $\mathcal{A}_{\text{mod}}$ and $\mathcal{A}_{\text{dem}}$ take the place of input and output alphabets. The formal description for the inner DC is the triplet $(\mathcal{A}_{\text{mod}}, \mathcal{A}_{\text{dem}}, P_{\tilde{y}|\tilde{x}})$. If it is not required to distinguish between the input alphabet and the modulation alphabet, we will omit the tilde in $\tilde{x}$ for the modulation alphabet.

It is important to distinguish certain properties of discrete channels. Apart from hard-decision and soft-decision demodulation, the properties of the channel or the transmission system can be *time-invariant* or *time-variant*. The discrete channel can also have *memory* (i.e., the received symbol does not only depend on the last transmitted symbol, but also on previous symbols), or it is *memoryless* (i.e., the received value only depends on the currently transmitted symbol).

Definition 1.1 (Discrete Memoryless Channel). *A* discrete memoryless channel *(DMC) is a memoryless and time-invariant discrete channel. The memoryless property is characterized by the fact that the transition probability for sequences factors into the product of the transition probabilities for the single symbols:*

$$P(y_0, \ldots, y_{n-1}|x_0, \ldots, x_{n-1}) = \prod_{i=0}^{n-1} P(y_i|x_i). \tag{1.3.2}$$

We can actually consider the transition probabilities for sequences or blocks for both the abstract DC and for the inner DC, since they are identical with

$$P(\tilde{y}_0, \ldots, \tilde{y}_{\tilde{n}-1}|\tilde{x}_0, \ldots, \tilde{x}_{\tilde{n}-1}) = P(y_0, \ldots, y_{n-1}|x_0, \ldots, x_{n-1}). \tag{1.3.3}$$

For different lengths of the blocks of the symbols x and $\tilde{x}$, this equation is still valid even if there is a symbol rate conversion in the mapping. If such a symbol rate conversion does not take place, then firstly, the above equation is also valid symbol-for-symbol, i.e., $P(\tilde{y}|\tilde{x}) = P(y|x)$. Secondly, the abstract DC is memoryless if and only if the inner DC is memoryless, since in this case the mapping does not have a memory and simply renames the alphabets. If a symbol rate conversion is performed in the mapping, then the memoryless relations become a little trickier. This case will be discussed in more detail in Subsection 1.3.3.

In Chapters 1 to 5 we only consider discrete memoryless channels. Since errors are statistically independent, these are also called *random single errors* and the DMC is then called a *random-single-error channel*. In later chapters, various other types of channels will be introduced: in Sections 6.6 and 6.7, in

Chapter 8 and in Section 10.6?, channels with errors grouped into bursts will be introduced, also called *burst-error channels*. In Section 10.7?, specific time-variant channels are discussed which occur when decoding convolutional codes, as well as super channels for concatenated coding. Fading channels which are of great importance for modern applications as well as channels with intersymbol interference are introduced in Section 14.2?. Further types of channels are described in Chapter 16?.

1.3.2 Discrete Memoryless Channels (DMC) with Hard Decisions

If the transition probabilities for hard decisions fulfill certain symmetries, only a single parameter is needed to characterize the DMC:

Definition 1.2 (Symmetric DMC with Hard Decisions). *A* q-ary symmetric channel with hard decisions *is a DMC with* $\mathcal{A}_{\text{in}} = \mathcal{A}_{\text{out}}$ *and the transition probability*

$$P(y|x) = \left\{ \begin{array}{ll} 1 - p_e & \text{if } y = x \\ p_e/(q-1) & \text{if } y \neq x \end{array} \right\}. \tag{1.3.4}$$

This channel is uniquely defined by the DMC symbol-error probability p_e*. The binary case with* $q = |\mathcal{A}_{\text{in}}| = 2$ *is called a* binary symmetric channel *(BSC).*

We often use $\mathcal{A}_{\text{in}} = \mathcal{A}_{\text{out}} = \{0, 1\}$ for the alphabets of the BSC, however, it is not the alphabets but the transition probabilities which are important. For $p_e = 0$ the symmetric DMC is error-free, but for $p_e = 1/2$ we will prove in Chapter 3 that a reliable transmission is absolutely impossible. If p_e is variable, the channel is time-variant. This situation will be discussed in detail in Section 10.7?. For the BSC, the equation (1.3.4) turns into

$$\begin{aligned} P_{y|x}(0|0) &= P_{y|x}(1|1) = 1 - p_e \\ P_{y|x}(1|0) &= P_{y|x}(0|1) = p_e. \end{aligned} \tag{1.3.5}$$

During transmission one bit is altered with the probability p_e and is transmitted correctly with the probability $1 - p_e$:

$$\begin{aligned} P(y = x) &= 1 - p_e \\ P(y \neq x) &= p_e. \end{aligned} \tag{1.3.6}$$

As a simple example, assume that 110 was transmitted, then 101 is received with a probability of $P_{y|x}(101|110) = P_{y|x}(1|1)P_{y|x}(0|1)P_{y|x}(1|0) = (1-p_e)\cdot p_e \cdot p_e$. The principle of the BSC is also shown on the left-hand side of Figure 1.4, where the branches from $x \in \mathcal{A}_{\text{in}}$ to $y \in \mathcal{A}_{\text{out}}$ are labeled with the transition probabilities $P(y|x)$.

For the q-ary symmetric hard-decision DMC, we note the following important formulas:

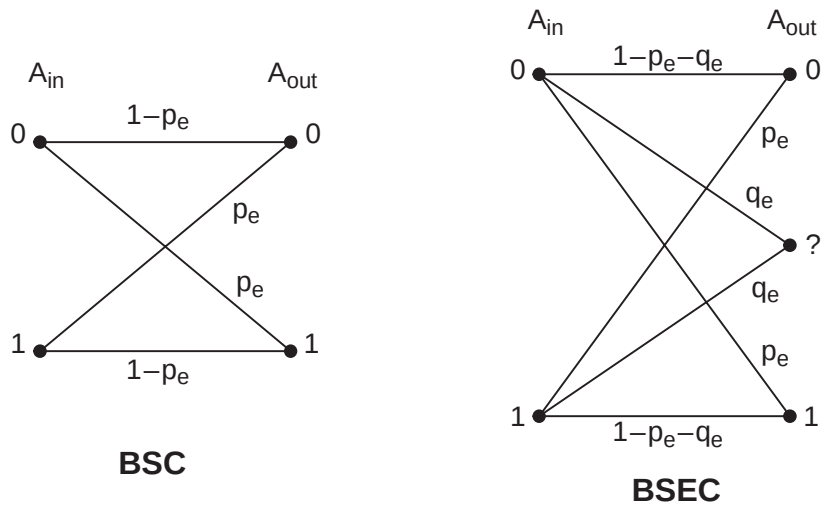

Figure 1.4. Transition probabilities for some discrete memoryless channels

(1) The function P_{ee} (ee = error event) describes the probability of at least one error occuring during the transmission of a sequence $\boldsymbol{x} = (x_0, x_1, \ldots, x_{n-1})$ of length n:

$$\begin{aligned} P_{ee} &= P(\boldsymbol{y} \neq \boldsymbol{x}) \\ &= 1 - P(\boldsymbol{y} = \boldsymbol{x}) \\ &= 1 - P(y_0 = x_0, \ldots, y_{n-1} = x_{n-1}) \\ &= 1 - P(y_0 = x_0) \cdots P(y_{n-1} = x_{n-1}) \\ &= 1 - (1 - p_e)^n \qquad (1.3.7) \\ &\approx np_e, \quad \text{if } np_e \ll 1. \qquad (1.3.8) \end{aligned}$$

The approximation (1.3.8) is derived from the binomial expansion

$$(1 - p_e)^n = \sum_{i=0}^{n} \binom{n}{i} (-p_e)^i, \qquad (1.3.9)$$

see also (A.2.2) for the general binomial formula.

(2) The probability that a sequence of n symbols is altered into another specific sequence, with l specific errors occuring in certain positions, is

$$P(\ l \text{ specific errors in } n \text{ symbols}) = \left(\frac{p_e}{q-1}\right)^l (1 - p_e)^{n-l}, \qquad (1.3.10)$$

since l specific errors each with probability $p_e/(q-1)$ and $n - l$ correct transmissions with probability $1 - p_e$ must occur.

(3) The probability that a sequence of n symbols contains l arbitrary errors in certain positions is

$$P(\ l \text{ specific error positions in } n \text{ symbols}) = p_e^l (1 - p_e)^{n-l}, \qquad (1.3.11)$$

since l errors with probability p_e and $n-l$ correct transmissions with probability $1-p_e$ must occur. This result can also be derived as follows. Since there are $q-1$ possibilities for a symbol error, the result (1.3.11) must be equal to (1.3.10) multiplied by $(q-1)^l$. For the binary case with $q=2$, (1.3.10) and (1.3.11) are identical since an error position also determines the error value.

(4) According to the binomial distribution (see Appendix A.4.1), the probability of l errors in a sequence of n bits is:

$$P(\ l \text{ arbitrary errors in } n \text{ symbols}) = \binom{n}{l} p_e^l (1-p_e)^{n-l} \tag{1.3.12}$$

$$= b(n,l,p_e). \tag{1.3.13}$$

This is also implied by (1.3.11) since there are exactly $\binom{n}{l}$ possibilities for the l arbitrary error positions. Furthermore, note that

$$1 = \sum_{l=0}^{n} \binom{n}{l} p_e^l (1-p_e)^{n-l}, \tag{1.3.14}$$

which is obvious but can also be derived from the binomial formula (A.2.2) or Subsection A.4.1.

A generalization of the BSC is the *binary symmetric erasure channel* (BSEC), also shown in Figure 1.4, where the output is ternary with $\mathcal{A}_{\text{out}} = \{0, ?, 1\}$. Here the demodulator decides on the "value" ?, if the decision to 0 or 1 is vague. For the decoder it is better to have no information on the transmitted bit rather than to have information which is wrong half the time. The BSEC is the simplest case of a discrete channel with soft decisions with

$$P(y|x) = \left\{ \begin{array}{ll} 1-p_e-q_e & \text{for } y = x \\ q_e & \text{for } y = ? \\ p_e & \text{otherwise} \end{array} \right\}. \tag{1.3.15}$$

Of course $P_{y|x}(0|x) + P_{y|x}(?|x) + P_{y|x}(1|x) = 1$ for $x \in \mathcal{A}_{\text{in}} = \{0,1\}$. For $q_e = 0$ a BSEC becomes a BSC, and for $p_e = 0$ a BSEC becomes a pure *binary erasure channel* (BEC).

1.3.3 Memoryless Relations between Abstract and Inner DC

In Subsection 1.3.1, we had already mentioned that without a symbol rate conversion in the mapping the simple equation $P(\tilde{y}|\tilde{x}) = P(y|x)$ is valid for the transition probabilities and that the abstract DC is memoryless if and only if the inner DC is memoryless.

In this subsection we will consider symbol rate conversion in the mapping. As in Figure 1.5, let $p_{e,\text{inner}} = P(\tilde{y} \neq \tilde{x})$ and $p_{e,\text{abstract}} = P(y \neq x)$ be the symbol-error probabilities of the inner DC and the abstract DC, respectively, of course, presupposing hard decisions. Firstly, we prerequisite $\log_2 q$ as an integer multiple of M or vice versa.

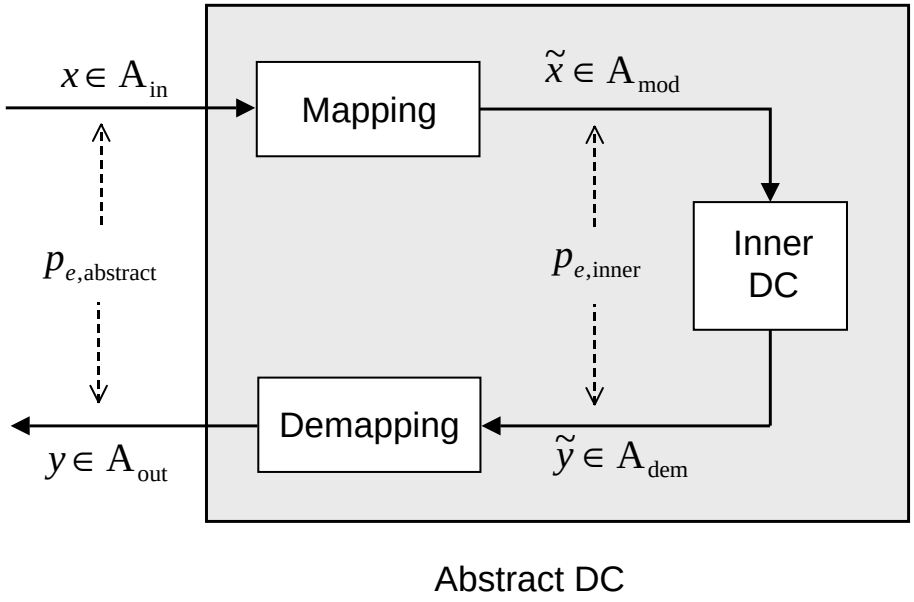

Figure 1.5. Symbol-error probabilities for inner and abstract discrete channel

(a) Suppose $\log_2 q$ as a multiple of M or equal to M. Hence, the mapping increases the symbol rate, and a single encoded symbol requires $(\log_2 q)/M$ subsequent modulation symbols for transmission. So an inner DMC also implies an abstract DMC.

If for instance $q = (2^M)^2$ or equivalently $(\log_2 q)/M = 2$, then $x \in \mathcal{A}_{\text{in}}$ is mapped to $(\tilde{x}_1, \tilde{x}_2) \in \mathcal{A}^2_{\text{mod}}$.

An encoded symbol is correctly received if and only if all of the corresponding $(\log_2 q)/M$ modulation symbols are correctly received. So, using the memorylessness of the inner DC,

$$1 - p_{e,\text{abstract}} = \Big(1 - p_{e,\text{inner}}\Big)^{(\log_2 q)/M}. \tag{1.3.16}$$

For instance, this result is used in (8.1.25) and for the performance curves in Subsections 8.1.5 and 8.2.4 with $q = 32 \ldots 256$ and $M = 1$.

(b) Suppose M as a multiple of $\log_2 q$. Hence, the mapping decreases the symbol rate, and $M/\log_2 q$ subsequent encoded symbols are carried by a single modulation symbol. In this case, an inner DMC only approximately implies the memoryless property after the demapping, depending on the specific design of the mapping.

This becomes intelligible by considering an example: let $2^M = q^2$ or equivalently $M/\log_2 q = 2$, then $(x_1, x_2) \in \mathcal{A}_{\text{in}}^2$ is mapped to $\tilde{x} \in \mathcal{A}_{\text{mod}}$. If $\tilde{y} \in \mathcal{A}_{\text{dem}}$ is received incorrectly, then usually both y_1 and y_2 are wrong. In other words, an inner DMC with random single errors implies burst errors after the demapping and this causes a certain kind of memory for the abstract DC, so the memoryless property does not result *exactly*.

However, for the opposite direction, if the abstract DC is memoryless then so is the inner DC. A modulation symbol is correctly received if and only if all of the corresponding $M/\log_2 q$ encoded symbols are correctly received. So, supposing the abstract DC as memoryless,

$$1 - p_{e,\text{inner}} = \Big(1 - p_{e,\text{abstract}}\Big)^{M/\log_2 q}. \tag{1.3.17}$$

The two equations (1.3.16) and (1.3.17) are equivalent and can be simply rewritten as

$$\Big(1 - p_{e,\text{abstract}}\Big)^{M} = \Big(1 - p_{e,\text{inner}}\Big)^{\log_2 q}. \tag{1.3.18}$$

To make things easier, we can assume for many applications that (1.3.18) is always approximately valid, even if there is no integer relationship between M and $\log_2 q$. A further approximation as in (1.3.8) leads to the simple result

$$p_{e,\text{abstract}} \approx \frac{\log_2 q}{M} \cdot p_{e,\text{inner}}. \tag{1.3.19}$$

In Subsection 2.4.6 we will consider the relation between bit- and symbol error rates and the approximation (2.4.16) will turn out to be a special case of (1.3.19).

However, all other details of the memoryless relations can be generally ignored for most applications unless we explicitly refer to them in the following chapters.

1.3.4 The Additive White Gaussian Noise (AWGN) Channel

In this subsection a model of a memoryless channel with overwhelming importance for digital communication will be introduced. A reasonable simplification and approximation for many applications is to assume that the physical channel attenuates the transmitted signal and introduces thermal noise and also suffers from co-channel and adjacent-channel interference originated from other communication systems. The attenuation and the interference will not be discussed in detail here. The noise is a fundamental barrier to communications, which may be defined as a random disturbance introduced mainly by the transmitter and receiver amplifiers and the propagation medium. The most commonly assumed model for the noisy channel is the additive white Gaussian noise (AWGN) channel, defined in this subsection for baseband signaling (1-dimensional case) as a

discrete-time channel. An extension of the AWGN channel to passband transmission (2-dimensional case) as well as a derivation of the discrete-time models from continuous-time reality is given in Chapter 2. A detailed and comprehensive representation of the statistical properties of the *Gaussian distribution* (also often called *normal distribution*) can be found in Appendix A.4.2 and A.4.3.

Definition 1.3 (Additive White Gaussian Noise). *An* additive white Gaussian noise (AWGN) channel *or simply* Gaussian channel *is a DMC* $(\mathcal{A}_{\rm mod}, \mathcal{A}_{\rm dem}, P_{y|x})$ *where the output* $y \in \mathcal{A}_{\rm dem} = \mathbb{R}$ *is the sum of the input* $x \in \mathcal{A}_{\rm mod}$ *and additive white Gaussian distributed noise denoted* ν*:*

$$y = x + \nu. \tag{1.3.20}$$

The two random variables x *and* ν *are statistically independent with zero means. The value* $E_s = E(x^2) = D^2(x)$ *denotes the* energy per modulation symbol *and* $N_0 = 2\sigma^2$ *denotes the* one-sided noise power spectral density*, where*

$$\sigma^2 = \frac{N_0}{2} = E(\nu^2) = D^2(\nu) \tag{1.3.21}$$

is the variance of the noise. The transition probability for the continuous-valued output is described by the probability density function (PDF):

$$f_{y|x}(\eta|\xi) = \frac{1}{\sqrt{\pi N_0}} \exp\left(-\frac{(\eta-\xi)^2}{N_0}\right). \tag{1.3.22}$$

Hence, the conditional distribution of the output y *for a given input* x *is described by* $N(x, \sigma^2)$*, using the notation of Subsection A.4.2. The noise* ν *is distributed according to* $N(0, \sigma^2)$*.*

The energy per encoded bit *is denoted* $E_c = E_s/M$*. For the* binary AWGN channel*,* $E_c = E_s$*, and the input alphabet is* $\mathcal{A}_{\rm in} = \{-\sqrt{E_c}, +\sqrt{E_c}\}$*.*

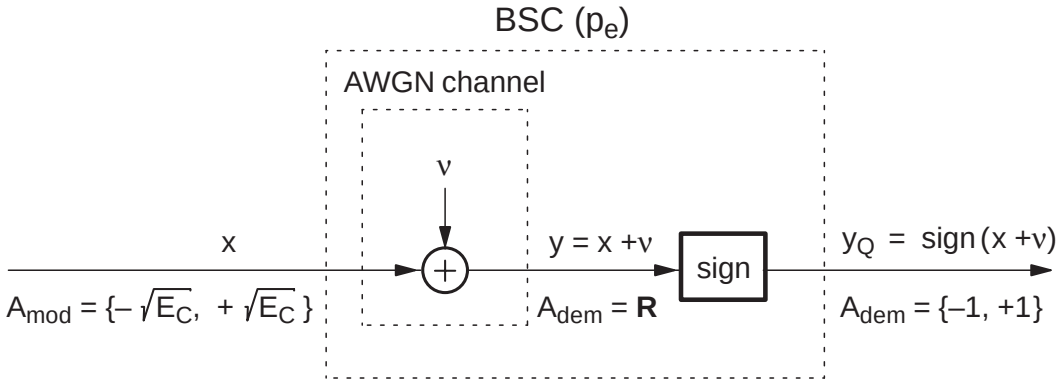

Figure 1.6. The BSC as a binary AWGN channel with binary quantization

The basic principle of the AWGN channel is shown in Figure 1.6. In addition, we suppose that the noise for consecutive AWGN transmissions is statistically

independent to ensure a memoryless channel, this is the reason for the term *white* noise. In case of colored noise, the random variables ν_{t_1} and ν_{t_2}, representing the noise for transmissions at times t_1 and t_2, could be statistically dependent.

Every 2^M-ary encoded modulation symbol of $\mathcal{A}_{\text{mod}}$ with the energy E_s carries M encoded bits. Hence, the corresponding energy per encoded bit is $E_c = E_s/M$.

It should be emphasized that the term *binary* AWGN refers to binary modulation with $M = 1$, so $E_c = E_s$ for this simplest case. Now we consider a demodulator performing binary hard quantization to produce binary output with $\mathcal{A}_{\text{dem}} = \{-1, +1\}$. Together with binary input as shown in Figure 1.6, we again obtain a BSC. The bit-error probability p_e of the BSC is now calculated as

$$\begin{aligned} p_e &= P(y < 0 \mid x = +\sqrt{E_c}) = P(y > 0 \mid x = -\sqrt{E_c}) \\ &= \int_0^{+\infty} \frac{1}{\sqrt{\pi N_0}} \exp\left(-\frac{(\eta + \sqrt{E_c})^2}{N_0}\right) d\eta \\ &= Q\left(\sqrt{\frac{2E_c}{N_0}}\right), \end{aligned} \tag{1.3.23}$$

where

$$Q(\alpha) = \frac{1}{\sqrt{2\pi}} \int_\alpha^\infty e^{-\eta^2/2}\, d\eta = \frac{1}{2}\text{erfc}\left(\frac{\alpha}{\sqrt{2}}\right) \tag{1.3.24}$$

$$= P(\nu > \alpha\sqrt{N_0/2}) \tag{1.3.25}$$

is the *complementary Gaussian error function* (see Appendix A.4.2). The graphical and numerical relation between p_e and E_c/N_0 is shown in Figure 1.7 and Table 1.1. Throughout this book, the graph $p_e = Q(\sqrt{2E_c/N_0})$ is also contained in many other figures with performance results under the term "uncoded" (with $E_c = E_b$, see the equation (1.4.11) with $R = 1$). Figure 1.7 also shows a simple upper bound given by the function $\exp(-E_c/N_0)/2$, which is often used. This function is implied by the inequality (A.4.17), proved in the appendix.

In Definition 1.3, we described the AWGN channel as an inner DMC $(\mathcal{A}_{\text{mod}}, \mathcal{A}_{\text{dem}}, P_{y|x})$ (omitting the tilde in x and y for simplification). Of course, the AWGN channel with soft decisions could only be described by an inner DMC, since the Gaussian PDF can only be defined for the real- or complex-valued alphabet $\mathcal{A}_{\text{dem}}$, but not for $\mathcal{A}_{\text{out}}$. In most applications one of the following cases applies.

(a) Hard-decision decoding with binary ($q = 2$) or non-binary (e.g., $q = 256$) block codes, which will be introduced in Section 1.4.

The memoryless property of the inner DMC is only exactly transferred to the abstract DC, if $\log_2 q$ is a multiple of M or equal to M. This was shown in Subsection 1.3.3.

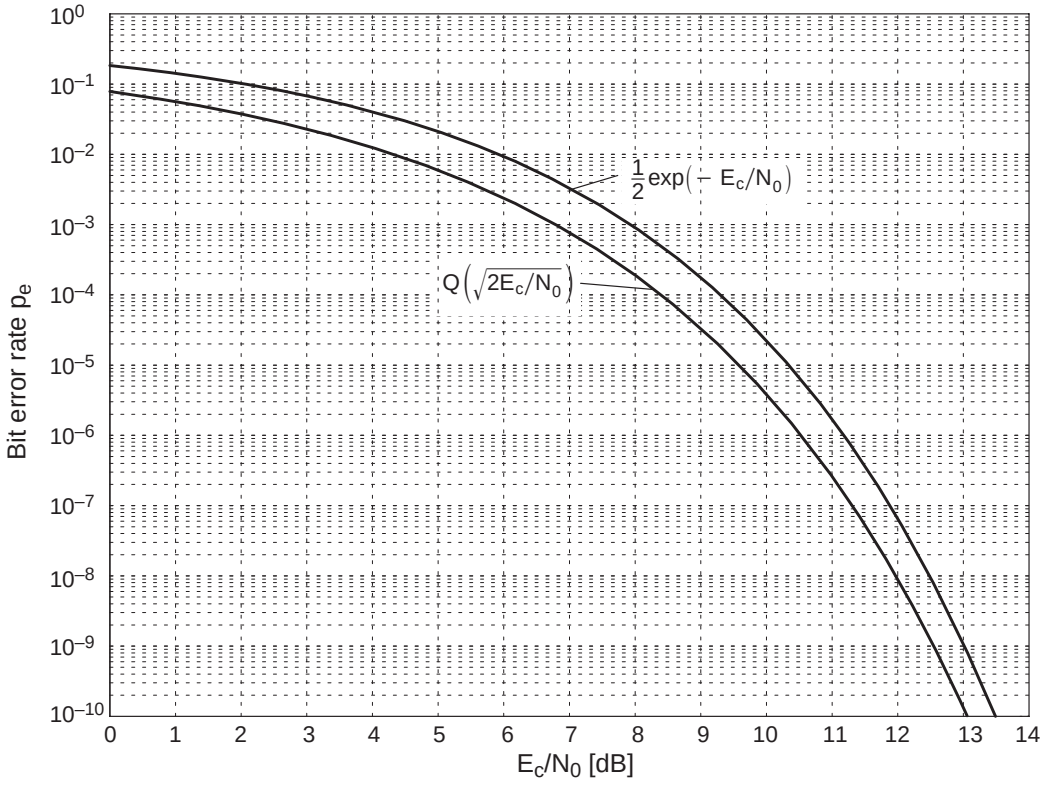

Figure 1.7. Error probability of the BSC as a binary quantized AWGN channel

Table 1.1. BSC bit-error probability
(often used values of p_e expressed as a function of E_c/N_0)

p_e	E_c/N_0 [dB]	p_e	E_c/N_0 [dB]
10^{-1}	-0.86	10^{-11}	13.52
10^{-2}	4.32	10^{-12}	13.93
10^{-3}	6.79	10^{-13}	14.31
10^{-4}	8.40	10^{-14}	14.66
10^{-5}	9.59	10^{-15}	14.99
10^{-6}	10.53	10^{-16}	15.29
10^{-7}	11.31	10^{-17}	15.57
10^{-8}	11.97	10^{-18}	15.84
10^{-9}	12.55	10^{-19}	16.09
10^{-10}	13.06	10^{-20}	16.32

(b) Soft-decision decoding with binary convolutional codes ($q = 2$), which will be introduced in Chapter 9. The combination with high-level modulation schemes ($M > 1$) will be described in Section ?.? as well as in Chapter 11 for TCM.

1.3.5 Octal Quantization of the AWGN Channel

If the quantization in the demodulator is not binary with 1 bit but octal with 3 bits, then the AWGN channel is transformed into a DMC with binary $\mathcal{A}_{\text{mod}} = \{-\sqrt{E_c}, +\sqrt{E_c}\}$ and octal $\mathcal{A}_{\text{out}} = \{-1, -1', -1'', -1''', +1''', +1'', +1', +1\}$. An important feature here is the optimum selection of the 7 quantization thresholds (maybe with quantization intervals of unequal width). A detailed analysis of the optimum thresholds can be found in [95].

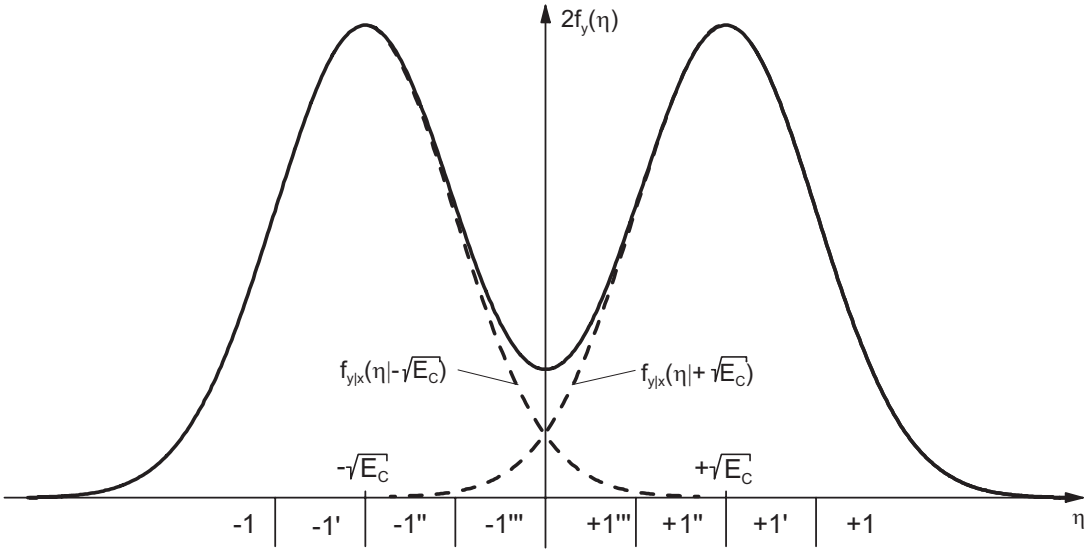

Figure 1.8. Probability distribution functions and octal quantization intervals for the binary AWGN channel

In Figure 1.7, equally spaced quantization thresholds are used. The exact alignment to the values $-\sqrt{E_c}$ and $+\sqrt{E_c}$ of the modulation alphabet obviously requires an automatic gain control in the demodulator. The probability density function (PDF) $f_y(\eta) = \frac{1}{2}\left(f_{y|x}(\eta| - \sqrt{E_c}) + f_{y|x}(\eta| + \sqrt{E_c})\right)$ of the received values results from the superposition of two Gaussian distributions, where $E_c/N_0 = +3$ dB was assumed for the illustration in Figure 1.7. In Figure 1.8, the transition probabilities for the octal channel are not given for the same signal-to-noise ratio, but for $E_c/N_0 = -3$ dB. The transition probabilities from $-\sqrt{E_c}$ to $+1$ or $+1'$ and from $+\sqrt{E_c}$ to -1 or $-1'$ are almost zero. The values of the PDF in Figure 1.8 are calculated, for example, as follows:

$$\begin{aligned} P(-1'''| - \sqrt{E_c}) &= P(-0.5\sqrt{E_c} < y < 0 | x = -\sqrt{E_c}) \\ &= P(0.5\sqrt{E_c} < \nu < \sqrt{E_c}) \\ &= Q(0.5\sqrt{2E_c/N_0}) - Q(\sqrt{2E_c/N_0}) \\ &= Q(0.5) - Q(1) \;=\; 0.3085 - 0.1587 \;=\; 0.1498. \end{aligned}$$

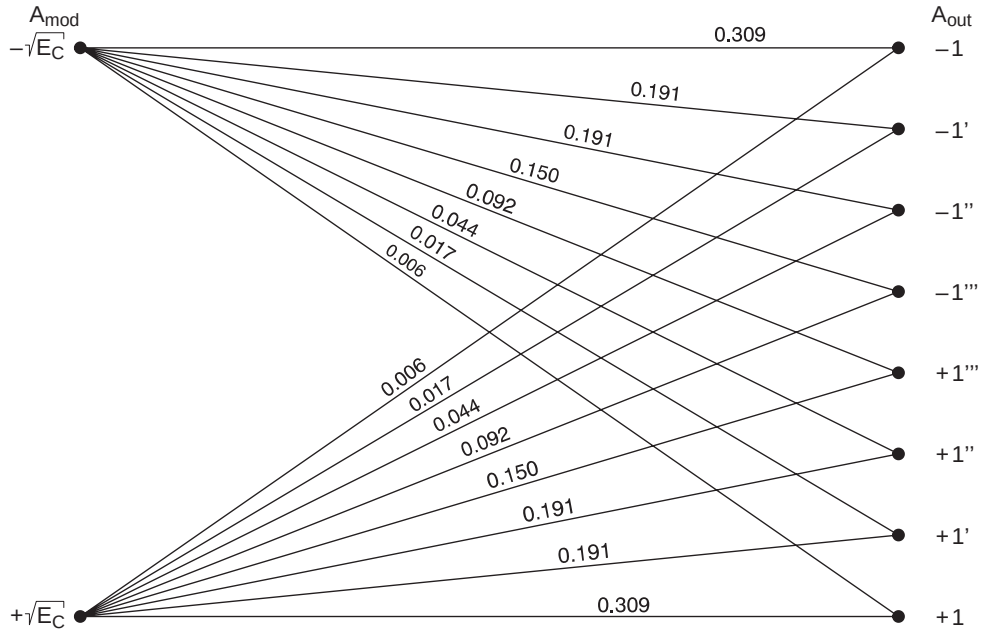

Figure 1.9. Transition probabilities for the binary AWGN channel with octal quantization

1.4 Block Coding

The basic ideas of error-control coding and information theory are best demonstrated by block codes. For the time being, the few basics of block codes given in this section will suffice, however, a detailed presentation will be provided in Chapters 4 to 8. A further class of error-control codes will be introduced in Chapter 9, the convolutional codes.

1.4.1 Basic Principles

The basic principle of block coding is illustrated in Figure 1.9. The data streams of information symbols and encoded symbols are divided into blocks of length k and n, which are called *information words* $\boldsymbol{u} = (u_0, \ldots, u_{k-1})$ and *codewords* $\boldsymbol{a} = (a_0, \ldots, a_{n-1})$. We suppose that $k < n$ with the exception of $k = n$ for uncoded signaling. The encoder assigns a codeword to each information word. At the output of the discrete channel the *received word* $\boldsymbol{y} = (y_0, \ldots, y_{n-1})$ occurs from which the decoder generates an estimation $\hat{\boldsymbol{u}} = (\hat{u}_0, \ldots, \hat{u}_{k-1})$ for the transmitted information word. The assignment of the codewords to the information words in the encoder is required to be

(1) *unique* and *injective*: two different information words are mapped to two different codewords;
(2) *time-invariant*: the assignment scheme does not change over a period of time;

(3) *memoryless*: each information word only influences one codeword, and each codeword is influenced by only one information word.

Due to the memoryless property and time-invariance, sequence numbers are not needed for indexing the information words or codewords. The transmission of a single, isolated codeword is considered instead of a sequence of codewords.

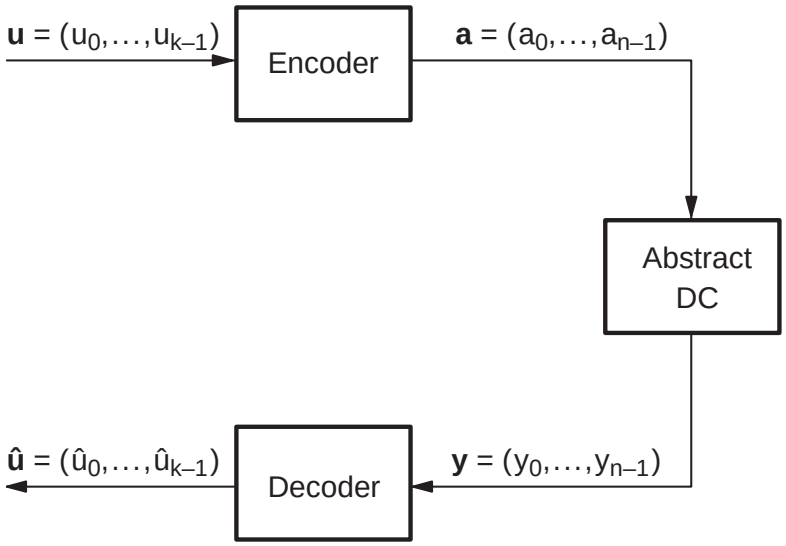

Figure 1.10. The basic principle of (n, k) block coding

In Chapters 1 to 8, we will only discuss block codes for error-control coding, primarily in conjunction with hard-decision decoding. In Chapters 9 and 10, convolutional codes as another major class of error-control codes are introduced, where the codeword is not only a function of the current information word but also of the preceding information words, thereby introducing memory to the encoder. Formally, block codes are a subset of convolutional codes with a memory length of zero, and vice versa there are also truncated convolutional codes which are a specific subset of block codes. However, these relationships between block and convolutional codes are pure formalisms and are not really meaningful.

Definition 1.4. *The aforementioned method defines an* $(n, k, d_{\min})_q$ block code*, or simply an* (n, k) *block code, where* $\mathbb{F}_q$ *denotes the common alphabet of size* q *for the information symbols* u_i*, the encoded symbols* a_i*, the estimated information symbols* $\hat{u}_i$*, and* $d_{\min}$ *denotes the minimum Hamming distance (see Definition 1.8). The length* n *of the encoded blocks is called the* block length *and the ratio of* k *to* n *is called the* code rate

$$R = \frac{k}{n} < 1 \quad \textit{in units of} \quad \left[\frac{\textit{info bit}}{\textit{encoded bit}} = \frac{\textit{info symbol}}{\textit{encoded symbol}} \right]. \tag{1.4.1}$$

The code $\mathcal{C}$ *is simply the set of all codewords. The inverse code rate* $1/R = n/k > 1$ *is called the* bandwidth expansion factor*. Two other forms of the code*

rate for the q-ary abstract DC and the 2^M-ary inner DC are

$$R_q = R\log_2 q = \frac{k}{n}\cdot\log_2 q \quad \textit{in units of} \quad \left[\frac{\textit{information bit}}{q\textit{-ary encoded symbol}}\right], \tag{1.4.2}$$

$$R_M = RM \quad = \frac{k}{n}\cdot M \quad \textit{in units of} \quad \left[\frac{\textit{information bit}}{2^M\textit{-ary modulation symb.}}\right]. \tag{1.4.3}$$

The code rate R can be addressed in units of information bits per encoded bit or equivalently in units of information symbols per encoded symbol, because information and encoded symbols have the same range $\mathbb{F}_q$. The code rate R is actually dimensionless, of course. One 2^M-ary modulation symbol carries M encoded bits, and one encoded bit carries R information bits. Hence, one modulation symbol actually carries $R_M = RM$ information bits. In other words, R_M is the number of information bits transmitted per waveform channel use (and sometimes R_M is referred to as the information transfer rate). For uncoded signaling, $R = 1$, $R_q = \log_2 q$ and $R_M = M$. Obviously $R_q = R$ for a binary abstract DC with $q = 2$, and $R_M = R$ for binary modulation with $M = 1$.

The number of information words of length k with q-ary symbols is obviously q^k, hence the size of the code is q^k, too. However, the total number of possible words of length n is q^n. So, the code $\mathcal{C}$ is a subset with the cardinality q^k of the set of all q^n words:

$$\mathcal{C} \subseteq \mathbb{F}_q^n, \quad |\mathcal{C}| = q^k = q^{nR}. \tag{1.4.4}$$

The block code defined by Definition 1.4 is somewhat restricted. Generally, the cardinality $|\mathcal{C}|$ can be an arbitrary integer, i.e., not necessarily a power of q. In this case, $k = nR = \dfrac{\log_2|\mathcal{C}|}{\log_2 q}$ may not be an integer. However, this generalization does not provide any theoretical advantages and in practice $|\mathcal{C}| = q^k$ is always chosen, with k as an integer.

The *performance of a code* is exclusively determined by how cleverly the q^k codewords are selected from the q^n words. It will turn out that the codewords will have to differ from each other as much as possible.

The encoder only makes an assignment between the q^k information words and the q^k codewords. Except for the requirements of uniqueness, time-invariance and memorylessness, this assignment function can be chosen arbitrarily. So the term *performance of an encoder* is meaningless. However, in practice systematic encoders are used almost exclusively (see Definition 1.5), and to simplify their realization linear codes (see Chapter 4) and cyclic codes (see Chapter 6) are almost always used.

Definition 1.5. *In a* systematic encoder *the mapping of the information words to the codewords is such that the information word is a part of the codeword. Then the other $n-k$ positions of the codeword are called* parity-check symbols.

Example 1.2. The two encoding rules (parity-check symbols first or last)

$$\begin{array}{ll} 00 \mapsto 000 & 00 \mapsto 000 \\ 01 \mapsto 011 & 01 \mapsto 101 \\ 10 \mapsto 101 & 10 \mapsto 110 \\ 11 \mapsto 110 & 11 \mapsto 011 \end{array}$$

create the same $(3,2)_2$ code $\mathcal{C} = \{000, 011, 101, 110\}$. The encoding rule can not be deduced from the code, and both encoders are similar. ■

Definition 1.6. *Two block codes are* identical*, if their sets of codewords are identical. Two block codes are* equivalent *if their sets of codewords are identical after a suitable permutation of the components of the codewords.*

1.4.2 The Role of the Mapping between Coding and Modulation

In addition to the symbol rate (or baud rate) r_s, defined in (1.2.5), we introduce two more data rates, which refer to bits rather than modulation symbols:

$$r_b = \text{information bit rate (info bits per second)} \tag{1.4.5}$$

$$r_c = \frac{r_b}{R} = \text{encoded bit rate (encoded bits per second)} \tag{1.4.6}$$

$$r_s = \frac{r_c}{M} = \frac{r_b}{RM} = \text{symbol rate (modulation symbols per second).} \tag{1.4.7}$$

The information bit rate r_b is also referred to as the *throughput*. Due to $r_c \geq r_b$, error-control coding generally implies an increase of the data rate by the bandwidth expansion factor of $1/R$. Also, note that

$$\frac{r_b}{\log_2 q} = \text{information symbol rate (info symbols per second)} \tag{1.4.8}$$

$$\frac{r_c}{\log_2 q} = \text{encoded symbol rate (encoded symbols per second).} \tag{1.4.9}$$

Do not forget that the stand-alone term *symbol* always refers to the encoded modulation symbols and not to the information symbols or encoded symbols. Obviously, the number of blocks transmitted per second is given by $r_b/(k \cdot \log_2 q) = r_c/(n \cdot \log_2 q)$.

The data rates are summarized in Figure 1.10. Since the mapping causes a symbol rate conversion of the factor $(\log_2 q)/M$ according to (1.2.6), a codeword $\boldsymbol{a}$ with block length n is assigned to a block $\tilde{\boldsymbol{x}}$ of length $\tilde{n} = (n \cdot \log_2 q)/M$ with $\tilde{q} = 2^M$-ary symbols. The pair mapping-demapping can be seen as part of the coding scheme as well as of the abstract discrete channel:

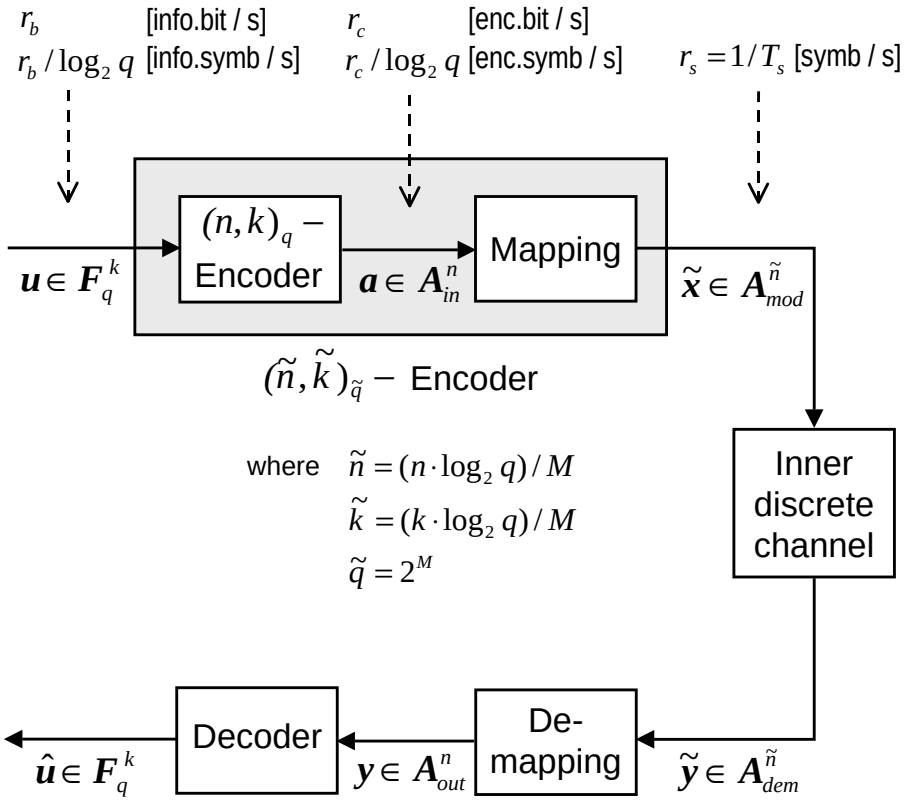

Figure 1.11. Overview of data rates for coded communication

Mapping-demapping as part of the coding scheme. A new $(\tilde{n}, \tilde{k})_{\tilde{q}}$ code is created, where $\tilde{k} = (k \cdot \log_2 q)/M$, because the size of the code remains the same since $\tilde{q}^{\tilde{k}} = q^k$. This new code is directly connected to the inner DC $(\mathcal{A}_{\text{mod}}, \mathcal{A}_{\text{dem}}, P_{\tilde{y}|\tilde{x}})$ with the 2^M-ary modulation alphabet $\mathcal{A}_{\text{mod}}$, which is used with a symbol rate of $r_s = r_b/(RM)$ in units of 2^M-ary modulations symbols per second.

This aspect is important for the AWGN channel with soft decisions, as will become clear with Theorem 1.4, and is the basis particularly for the joint optimization of coding and modulation, with trellis coded modulation (TCM) in Chapter 11 as an important example.

Mapping-demapping as part of the abstract DC. The inner DC with the 2^M-ary modulation alphabet $\mathcal{A}_{\text{mod}}$ is invisible. The original code sees the abstract DC $(\mathcal{A}_{\text{in}}, \mathcal{A}_{\text{out}}, P_{y|x})$ with the q-ary input alphabet $\mathbb{F}_q = \mathcal{A}_{\text{in}}$ and the encoded symbol rate $r_c/\log_2 q$ in units of q-ary encoded symbols per second.

This approach is the basis for the derivation of the general maximum-likelihood decoder in Subsection 1.6.1, for the channel capacity of the abstract DMC in Chapter 3 and for hard-decision decoding of block codes in general.

In Chapter 3, we will introduce the channel capacity as a fundamental concept

of information theory. It will provide us with a boundary for the maximum throughput for an almost error-free transmission. In Subsection 3.2.2, we will realize that the two approaches mentioned above lead to the same results.

For the AWGN channel with 2^M-ary modulation we can define the bit and symbol energies in a similar manner to the bit and symbol rates above:

$$E_b = \text{energy per information bit} \tag{1.4.10}$$
$$E_c = R \cdot E_b = \text{energy per encoded bit} \tag{1.4.11}$$
$$E_s = RM \cdot E_b = \text{energy per } 2^M\text{-ary modulation symbol.} \tag{1.4.12}$$

Corresponding to $r_c = r_b/R > r_b$, the relation $E_c = RE_b < E_b$ indicates that error-control coding generally implies a reduction of the available energy per transmitted bit by a factor of R. Thus the bit-error rate of the received encoded bits before the decoder will be higher than for uncoded transmission. A coding gain is only achieved if the error-correction capability of the code compensates for this negative effect.

1.5 Hamming Distance and Minimum Distance

Let $\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z}$ be words of equal length n, for example codewords with symbols from the alphabet $\mathcal{A}_{\text{in}}$ or received words with symbols from the alphabet $\mathcal{A}_{\text{out}}$.

Definition 1.7. *The* Hamming distance $d_H(\boldsymbol{x}, \boldsymbol{y})$ *between the words* $\boldsymbol{x}$ *and* $\boldsymbol{y}$ *is defined as the number of positions in which the components differ. If zero is an element of the symbol range, the* Hamming weight $w_H(\boldsymbol{x})$ *is the number of components of* $\boldsymbol{x}$ *which are not zero.*

The relation between the Hamming distance and the Hamming weight is as follows:

$$w_H(\boldsymbol{x}) = d_H(\boldsymbol{x}, \boldsymbol{0}) \quad \text{with} \quad \boldsymbol{0} = (0, \ldots, 0). \tag{1.5.1}$$

If a "subtraction" is defined for the symbol range, then

$$d_H(\boldsymbol{x}, \boldsymbol{y}) = w_H(\boldsymbol{x} - \boldsymbol{y}). \tag{1.5.2}$$

Theorem 1.1. *The Hamming distance is a* metric *in the strict mathematical sense, i.e., it has the following properties:*

$$d_H(\boldsymbol{x}, \boldsymbol{y}) = d_H(\boldsymbol{y}, \boldsymbol{x}) \tag{1.5.3}$$
$$0 \le d_H(\boldsymbol{x}, \boldsymbol{y}) \le n \tag{1.5.4}$$
$$d_H(\boldsymbol{x}, \boldsymbol{y}) = 0 \iff \boldsymbol{x} = \boldsymbol{y} \tag{1.5.5}$$
$$d_H(\boldsymbol{x}, \boldsymbol{y}) \le d_H(\boldsymbol{x}, \boldsymbol{z}) + d_H(\boldsymbol{z}, \boldsymbol{y}). \tag{1.5.6}$$

The inequality (1.5.6) is called the triangle inequality *and is illustrated in Figure 1.11. If addition and subtraction are defined for the symbol range, then the Hamming weight has the following properties:*

$$w_H(\boldsymbol{x}) = w_H(-\boldsymbol{x}) \tag{1.5.7}$$

$$0 \le w_H(\boldsymbol{x}) \le n \tag{1.5.8}$$

$$w_H(\boldsymbol{x}) = 0 \quad \Longleftrightarrow \quad \boldsymbol{x} = \boldsymbol{0} \tag{1.5.9}$$

$$w_H(\boldsymbol{x} + \boldsymbol{y}) \le w_H(\boldsymbol{x}) + w_H(\boldsymbol{y}). \tag{1.5.10}$$

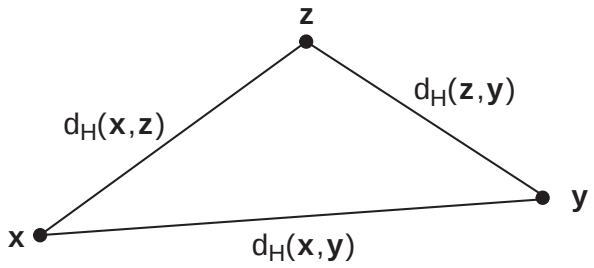

Figure 1.12. Illustration of the triangle inequality for the Hamming distance

Definition 1.8. *The* minimum Hamming distance *(or simply minimum distance)* $d_{\min}$ *of an* $(n, k, d_{\min})$ *block code* $\mathcal{C}$ *is defined as the minimum Hamming distance between all codewords:*

$$d_{\min} = \min\{d_H(\boldsymbol{a}, \boldsymbol{b}) \mid \boldsymbol{a}, \boldsymbol{b} \in \mathcal{C},\ \boldsymbol{a} \neq \boldsymbol{b}\}. \tag{1.5.11}$$

The minimum distance is the most important parameter that determines the performance of a code. Later, the full characterization of a code will be defined by the weight distribution (see Definition 3.7). To compute the minimum distance $d_{\min}$, the distances between all pairs of codewords have to be considered. The larger the minimum distance, i.e., the bigger the differences between the codewords, the better the code. For a given code rate $R = k/n$ and a given block length n the code should be chosen such that $d_{\min}$ is as large as possible. Usually $d_{\min}$ is larger (i.e., a better code), if the code rate is smaller (i.e., more bandwidth is required) or if the block length is larger (i.e., more complex processing is required).

Example 1.3. The $(7, 4)_2$ Hamming code consists of 16 codewords of length 7:

$$\begin{aligned}\mathcal{C} \;=\; \{\, & 0000000, \quad 0100101, \quad 1000011, \quad 1100110, \\ & 0001111, \quad 0101010, \quad 1001100, \quad 1101001, \\ & 0010110, \quad 0110011, \quad 1010101, \quad 1110000, \\ & 0011001, \quad 0111100, \quad 1011010, \quad 1111111 \,\}. \end{aligned}$$

Here, the code is given by a listing of all codewords and not by the (unimportant) encoding rule, which could be systematic with the four information bits in front.

The comparison of the first two codewords delivers an upper bound of $d_{\min} \leq 3$. After comparing all other pairs, no pair with a Hamming distance of 2 or smaller can be found, thus $d_{\min} = 3$. ■

We are obviously going to need better methods than a simple listing of codewords to describe a code and to calculate the minimum distance, therefore in later chapters we will introduce algebraic structures for the code.

1.6 Maximum-Likelihood Decoding (MLD)

1.6.1 Derivation of the General Decoding Rule

The optimum decoding rule should be designed to minimize the word-error probability $P_w = P(\hat{\boldsymbol{u}} \neq \boldsymbol{u})$ after decoding. Assume a stochastic discrete channel which causes errors in the received word that the decoder may not be able to correct, thus leading to errors in the estimated information word. These errors in the transmission of a large number of words should occur as seldom as possible. For most applications it is irrelevant whether there is only one error or several errors in a wrongly estimated information word. A minimization of the bit-error probability $P_b = P(\hat{u}_i \neq u_i)$ is much more difficult.

So the objective is that the estimated information word is equal to the transmitted word as often as possible. This is the requirement that should determine the design of the decoder. We will see that the construction rule for the decoder can be deduced even though the criterion P_w can not be explicitly calculated:

$$P_w = P(\hat{\boldsymbol{u}} \neq \boldsymbol{u}) = P(\hat{\boldsymbol{a}} \neq \boldsymbol{a}) \quad \longrightarrow \quad \text{minimum.} \tag{1.6.1}$$

The unique mapping of information words to codewords allows the codeword to be estimated instead of the information word. The estimate for the information word is correct if and only if the estimate for the codeword is correct. Therefore the word-error probability can be defined by the codewords instead of by the information words.

Figure 1.12 shows the basic principle for deriving the decoding rule. The decoder is divided into a codeword estimator δ and an encoder inverse. This encoder inverse is a direct inversion of the encoder and its only task is to determine the corresponding information word $\hat{\boldsymbol{u}}$ to each estimated codeword $\hat{\boldsymbol{a}}$. This is a trivial operation, for example for systematic encoders this only means omitting the parity-check symbols.

The whole intelligence of the decoder is in the codeword estimator which, in contrast to the encoder inverse, only needs to know the code but not the encoding rule. So the codeword estimator determines the estimated codeword for the received word $\boldsymbol{y}$. The formal function is

$$\delta \; : \; \boldsymbol{y} \in \mathcal{A}_{\text{out}}^n \quad \mapsto \quad \delta(\boldsymbol{y}) = \hat{\boldsymbol{a}} \in \mathcal{C}. \tag{1.6.2}$$

The objective for constructing the function δ is that the word-error probability takes on its minimum:

$$P_w = P(\delta(\boldsymbol{y}) \neq \boldsymbol{a}) \quad \longrightarrow \quad \text{minimum.} \tag{1.6.3}$$

The result of a transmission over a discrete channel with hard decisions, i.e., with $\mathcal{A}_{\text{in}} = \mathcal{A}_{\text{out}} = \mathbb{F}_q$, can be subdivided into the following cases:

$\boldsymbol{y} = \boldsymbol{a}$ Error-free correct transmission.

$\boldsymbol{y} \in \mathcal{C} \backslash \{\boldsymbol{a}\}$ Falsification into a different codeword. This can never be detected or corrected.

$\boldsymbol{y} \notin \mathcal{C}$ The error pattern is generally detectable and could perhaps be corrected by the decoder. For $\delta(\boldsymbol{y}) = \boldsymbol{a}$ the decoding is correct and for $\delta(\boldsymbol{y}) \neq \boldsymbol{a}$ the decoding is wrong. The case of $\delta(\boldsymbol{y}) =$ "undefined" does not occur for an ideal decoder, but for decoders in practice this case is quite sensible (see the following explanation).

In the formal description the function δ assigns one of q^k codewords to each of the q^n possible received words. Later, we will see that this is not always implemented in a real decoder, in other words it could be reasonable not to compute an optimal estimate for each possible received word. Instead, the optimum decoding rule is only realized for received words which occur most often. This method may simplify the realization of the decoder a great deal.

If a word is completely received, a blockwise estimation of the codeword or information word can be made almost immediately, apart from possible processing delays depending on the implementation. So usually, the estimation for the first information symbol in the receiver can only be made as soon as

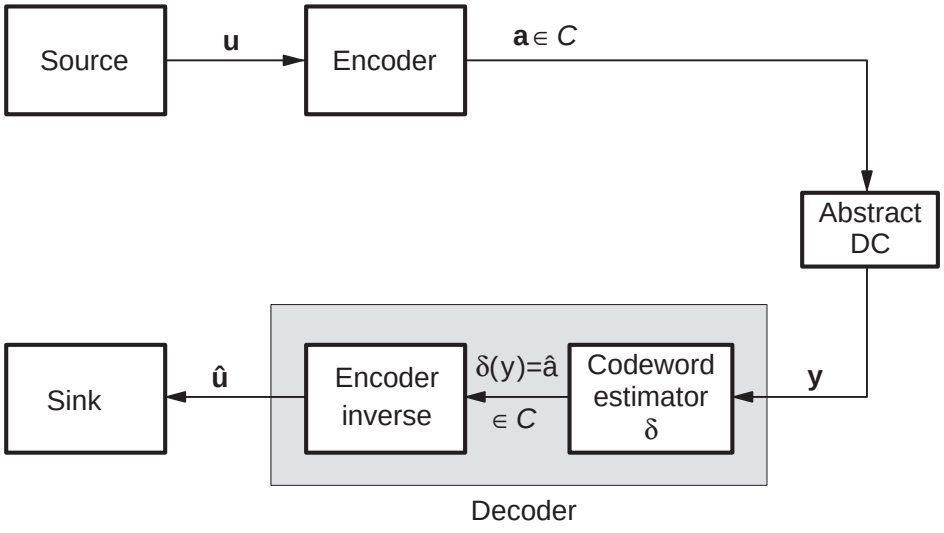

Figure 1.13. Illustration of the decoding rule

the last transmitted encoded symbol has arrived. Therefore the block length n determines a lower bound for the minimum delay for the coded transmission.

Assumption of equal a priori probabilities *for the derivation of the decoding rule: all q^k information words are to occur at the encoder input with the same probability q^{-k}.*

With this precondition all codewords occur with the same probability q^{-k} as well. Firstly, we take a look at the probability that an error occurs during decoding. Assuming that a specific codeword $\boldsymbol{a}$ was transmitted, this is given by the sum over the received words leading to an estimate unequal to $\boldsymbol{a}$:

$$\begin{aligned} P(\delta(\boldsymbol{y}) \neq \boldsymbol{a} \mid \boldsymbol{a} \text{ transmitted}) &= \sum_{\substack{\boldsymbol{y} \\ \delta(\boldsymbol{y}) \neq \boldsymbol{a}}} P(\boldsymbol{y} \text{ received} \mid \boldsymbol{a} \text{ transmitted}) \\ &= \sum_{\substack{\boldsymbol{y} \\ \delta(\boldsymbol{y}) \neq \boldsymbol{a}}} P_{y|x}(\boldsymbol{y}|\boldsymbol{a}). \end{aligned} \tag{1.6.4}$$

Secondly, for the sum over all codewords and all received words,

$$\sum_{\boldsymbol{a} \in \mathcal{C}, \boldsymbol{y}} P_{y|x}(\boldsymbol{y}|\boldsymbol{a}) = \sum_{\boldsymbol{a} \in \mathcal{C}} P_{y|x}(\text{arbitrary } \boldsymbol{y} \text{ received } |\boldsymbol{a}) = \sum_{\boldsymbol{a} \in \mathcal{C}} 1 = q^k. \tag{1.6.5}$$

Thirdly, Bayes' theorem of total probability (A.3.1) implies that

$$\begin{aligned} P_w &= P(\delta(\boldsymbol{y}) \neq \boldsymbol{a}) \\ &= \sum_{\boldsymbol{a} \in \mathcal{C}} P(\delta(\boldsymbol{y}) \neq \boldsymbol{a} \mid \boldsymbol{a} \text{ transmitted}) \cdot P(\boldsymbol{a} \text{ transmitted}) \\ &= \sum_{\boldsymbol{a} \in \mathcal{C}} \sum_{\substack{\boldsymbol{y} \\ \delta(\boldsymbol{y}) \neq \boldsymbol{a}}} P_{y|x}(\boldsymbol{y}|\boldsymbol{a}) \cdot q^{-k} \qquad \text{with (1.6.4)} \\ &= q^{-k} \left(\sum_{\boldsymbol{a} \in \mathcal{C}, \boldsymbol{y}} P_{y|x}(\boldsymbol{y}|\boldsymbol{a}) - \sum_{\substack{\boldsymbol{a} \in \mathcal{C}, \boldsymbol{y} \\ \delta(\boldsymbol{y}) = \boldsymbol{a}}} P_{y|x}(\boldsymbol{y}|\boldsymbol{a}) \right) \\ &= 1 - q^{-k} \cdot \sum_{\substack{\boldsymbol{a} \in \mathcal{C}, \boldsymbol{y} \\ \delta(\boldsymbol{y}) = \boldsymbol{a}}} P_{y|x}(\boldsymbol{y}|\boldsymbol{a}) \qquad \text{with (1.6.5)} \\ &= 1 - q^{-k} \cdot \sum_{\boldsymbol{y}} P_{y|x}(\boldsymbol{y}|\delta(\boldsymbol{y})). \end{aligned}$$

To minimize P_w for each received word $\boldsymbol{y}$, $\delta(\boldsymbol{y}) = \hat{\boldsymbol{a}}$ should be chosen such that the transition probability $P_{y|x}(\boldsymbol{y}|\hat{\boldsymbol{a}})$ takes on its maximum.

Theorem 1.2 (Maximum-Likelihood Decoder MLD). *The word-error probability P_w after decoding takes on its minimum if the decoding is as follows. For the received word $\boldsymbol{y}$ the estimated codeword $\hat{\boldsymbol{a}} \in \mathcal{C}$ is chosen such that*

the transition probability takes on its maximum:

$$P_{y|x}(\boldsymbol{y}|\hat{\boldsymbol{a}}) \geq P_{y|x}(\boldsymbol{y}|\boldsymbol{b}) \quad \textit{for all } \boldsymbol{b} \in \mathcal{C}. \tag{1.6.6}$$

The ML decoding could also be ambiguous, then any codeword with maximum transition probability is chosen.

This result is easily illustrated: for a given received word, the transmitted word or codeword is sought which was most probably transmitted. The calculation of the word-error probability will be explained in Chapter 4.

However, without the precondition of equal a priori probabilities we obtain a different decoder, the *maximum a posteriori (MAP) decoder*. The MAP decoder must be adapted to the source statistics, or more precisely to the probability distribution of the codewords. The MAP approach leads to a smaller word-error probability than the ML decoder if the transmitted words are not equally probable, see Problem 1.11. However, because of the dependency on the source statistics the MAP decoder is only applied in special cases.

1.6.2 ML Decoding for the Hard-Decision DMC

The previous results become even clearer, if the q-ary symmetric DMC is considered. The equations (1.3.2) and (1.3.4) imply that for $\boldsymbol{y} = (y_0, \ldots, y_{n-1})$ and $\hat{\boldsymbol{a}} = (\hat{a}_0, \ldots, \hat{a}_{n-1})$ with $d = d_H(\boldsymbol{y}, \hat{\boldsymbol{a}})$,

$$\begin{aligned} P_{y|x}(\boldsymbol{y}|\hat{\boldsymbol{a}}) &= \prod_{i=0}^{n-1} P_{y|x}(y_i|\hat{a}_i) \\ &= \prod_{i=0}^{n-1} \left\{ \begin{array}{ll} 1-p_e & \text{if } y_i = \hat{a}_i \\ p_e/(q-1) & \text{if } y_i \neq \hat{a}_i \end{array} \right\} \\ &= (1-p_e)^{n-d} \cdot \left(\frac{p_e}{q-1} \right)^d \\ &= (1-p_e)^n \cdot \left(\frac{p_e}{(1-p_e)(q-1)} \right)^d . \end{aligned} \tag{1.6.7}$$

The left-hand factor in the last equation does not depend on $\hat{\boldsymbol{a}}$, thus only the right-hand factor needs to be minimized by choosing a suitable codeword $\hat{\boldsymbol{a}}$. For $p_e < 0.5$ the quotient is smaller than 1, thus we obtain the maximum if $d = d_H(\boldsymbol{y}, \hat{\boldsymbol{a}})$ takes on its minimum:

Theorem 1.3 (MLD for Hard-Decision DMC). *The word-error probability P_w after decoding takes on its minimum if the decoding is as follows. For each received word $\boldsymbol{y}$ the estimated codeword $\hat{\boldsymbol{a}} \in \mathcal{C}$ is chosen such that the Hamming distance takes on its minimum:*

$$d_H(\boldsymbol{y}, \hat{\boldsymbol{a}}) \leq d_H(\boldsymbol{y}, \boldsymbol{b}) \quad \textit{for all } \boldsymbol{b} \in \mathcal{C}. \tag{1.6.8}$$

Example 1.4. Consider the $(5,2)_2$ code $\mathcal{C} = \{\underbrace{00000}_{\boldsymbol{a}_1}, \underbrace{11100}_{\boldsymbol{a}_2}, \underbrace{00111}_{\boldsymbol{a}_3}, \underbrace{11011}_{\boldsymbol{a}_4}\}$ for which obviously $d_{\min} = 3$. In the following table the distances to all codewords for three exemplary received words are given together with the resulting decision of the codeword estimator.

$\boldsymbol{y}$	$d_H(\boldsymbol{y}, \boldsymbol{a}_1)$	$d_H(\boldsymbol{y}, \boldsymbol{a}_2)$	$d_H(\boldsymbol{y}, \boldsymbol{a}_3)$	$d_H(\boldsymbol{y}, \boldsymbol{a}_4)$	$\delta(\boldsymbol{y})$
10000	1	2	4	3	$\boldsymbol{a}_1$
11000	2	1	5	2	$\boldsymbol{a}_2$
10001	2	3	3	2	$\boldsymbol{a}_1$ or $\boldsymbol{a}_4$

For the last received word, ideally the decision should be randomly, either $\boldsymbol{a}_1$ or $\boldsymbol{a}_4$. ■

For the $(7,4)$ Hamming code in Example 1.2 this method already becomes quite elaborate, so that codes used in practice should fulfill two requirements: (1) the minimum distance $d_{\min}$ should be as large as possible; (2) the structure of the code $\mathcal{C}$ should simplify the search for the minimum Hamming distance in the decoder. Both requirements presume an algebraic structure. For (1) the structure is required to be able to find good codes and for (2) the structure is required to facilitate decoders with reasonable implementations.

1.6.3 ML Decoding for the AWGN Channel

We will now derive the MLD for the AWGN channel, similar to Theorem 1.3. According to (1.3.11), the probability density functions (PDF) for words factors into a product of PDFs for symbols and can then be rewritten as follows:

$$\begin{aligned} f_{y|x}(\boldsymbol{y}|\hat{\boldsymbol{a}}) &= \prod_{i=0}^{n-1} f_{y|x}(y_i|\hat{a}_i) \\ &= \prod_{i=0}^{n-1} \frac{1}{\sqrt{\pi N_0}} \exp\left(-\frac{(y_i-\hat{a}_i)^2}{N_0}\right) \\ &= (\pi N_0)^{-n/2} \exp\left(-\frac{1}{N_0}\sum_{i=0}^{n-1}(y_i-\hat{a}_i)^2\right) \\ &= c \cdot \exp\left(-\frac{1}{N_0}\|\boldsymbol{y}-\hat{\boldsymbol{a}}\|^2\right), \end{aligned} \tag{1.6.9}$$

where c is constant and $\|\boldsymbol{y}-\hat{\boldsymbol{a}}\|^2$ is the *Euclidean norm* of $\boldsymbol{y}-\hat{\boldsymbol{a}}$ which is identical to the *Euclidean distance* between $\boldsymbol{y}$ and $\hat{\boldsymbol{a}}$. The right-hand factor is to be maximized by making a suitable choice for $\hat{\boldsymbol{a}}$. This is achieved by minimizing the norm, leading us to the following theorem.

Theorem 1.4 (MLD for Soft-Decision AWGN channel). *The word-error probability P_w after decoding takes on its minimum if the decoding is as follows.*

For the received word $\boldsymbol{y}$ *the estimated codeword* $\hat{\boldsymbol{a}} \in \mathcal{C}$ *is chosen such that the Euclidean distance to the received word takes on its minimum:*

$$\|\boldsymbol{y} - \hat{\boldsymbol{a}}\| \;\leq\; \|\boldsymbol{y} - \boldsymbol{b}\| \quad \textit{for all } \boldsymbol{b} \in \mathcal{C}. \tag{1.6.10}$$

This result is easily comprehensible. Obviously, the difference between hard- and soft-decision ML decoding corresponds to the difference between the Hamming and the Euclidean distance for the search of the nearest codeword.

Now let us formally consider, which consequences result from the fact that the AWGN channel is only defined as an inner DC. The decoder has to search for the codeword $\boldsymbol{b}$ with the minimum Euclidean distance from the received word $\boldsymbol{y}$. However, the codeword $\boldsymbol{b}$ is not taken from $\mathcal{A}_{\text{in}}^n = \mathbb{F}_q^n$ but from $\mathcal{A}_{\text{mod}}^{\tilde{n}} \subset \mathbb{R}^{\tilde{n}}$ or $\mathbb{C}^{\tilde{n}}$ with $\tilde{n} = (n \cdot \log_2 q)/M$ as in Subsection 1.4.2. So the decoder must imitate the mapping and the demapping, and calculate the distance in $\mathbb{R}$ or $\mathbb{C}$ and not in $\mathbb{F}_q$. If the mapping causes a data rate conversion, the block length changes from n to $\tilde{n}$. In conclusion, only the first of the two approaches of Subsection 1.4.2 (mapping as part of the coding scheme) is applicable in this case. So it is irrelevant how the memoryless property is affected by the demapping, because only the inner DC is relevant.

Equation (1.6.10) can be written as $\sum_i (y_i - \hat{a}_i)^2 \leq \sum_i (y_i - b_i)^2$. The terms y_i^2 can be omitted, leading to the simpler form of

$$\sum_{i=0}^{n-1} (\hat{a}_i^2 - 2y_i\hat{a}_i) \;\leq\; \sum_{i=0}^{n-1} (b_i^2 - 2y_i b_i) \quad \text{for all } \boldsymbol{b} \in \mathcal{C},$$

which is equivalent to

$$\sum_{i=0}^{n-1} y_i\hat{a}_i - \frac{1}{2}\sum_{i=0}^{n-1} \hat{a}_i^2 \;\geq\; \sum_{i=0}^{n-1} y_i b_i - \frac{1}{2}\sum_{i=0}^{n-1} b_i^2 \quad \text{for all } \boldsymbol{b} \in \mathcal{C}.$$

Particularly for binary codes we have $\hat{a}_i, b_i \in \{-\sqrt{E_c}, +\sqrt{E_c}\}$, so the quadratic terms disappear, leaving

$$\sum_{i=0}^{n-1} y_i\hat{a}_i \;\geq\; \sum_{i=0}^{n-1} y_i b_i \quad \text{for all } \boldsymbol{b} \in \mathcal{C}. \tag{1.6.11}$$

For a transmitted word, the estimated codeword is chosen such that the *correlation* with the received word takes on its maximum. Yet, 2^k scalar products are to be calculated which is still so time-consuming that block codes are usually only decoded with hard decisions allowing for less complex decoders.

1.7 Asymptotic Coding Gains

The *symbol-error probability* $P_s = P(\hat{a}_i \neq a_i)$ at the decoder output, also called *bit-error probability* P_b or *bit-error rate* (BER) in case of $q = 2$ with $P_b = P_s$, only refers to the information symbols or bits and does not take the parity-check symbols into account. The exact relation between P_s or P_b and the *word-error rate* $P_w = P(\hat{\boldsymbol{a}} \neq \boldsymbol{a})$ at the decoder output is relatively complicated, but can at least be easily lower and upper bounded. Since the number of errors in a wrongly decoded word is between 1 and k,

$$\frac{1}{k} \cdot P_w \leq P_s \leq P_w. \tag{1.7.1}$$

Typically, the approximation

$$P_s \approx \frac{d_{\min}}{k} \cdot P_w \tag{1.7.2}$$

is reasonable which presumes $d_{\min}$ errors per wrongly decoded word. A fairly exact approximation of P_s and a further argument for (1.7.2) will be given in Theorem 4.15. However, this problem is not that important in the practical evaluation of codes.

The comparison of binary codes ($q = 2$) to each other and to uncoded transmission is very often based on the use of binary modulation ($M = 1$) over the AWGN channel. As in Definition 1.3, let N_0 be the one-sided noise power density and E_b the energy per information bit. As already stated in (1.4.11), the energy per encoded bit

$$E_c = R \cdot E_b < E_b \tag{1.7.3}$$

is smaller than E_b by the factor of R. Comparing coded and uncoded transmission with equal signal energy, the signal energy for coded transmission has to be shared among the information bits and the parity-check bits, so that the energy per transmitted bit is reduced. Hence, the bit-error rate of the received bits before the decoder will be higher than for uncoded transmission. This negative effect must be compensated for by the error-correction capability of the code in order to achieve a coding gain in total.

Figures 1.13 and 1.14 present comparisons of coded and uncoded transmissions over the AWGN channel by displaying the word- and bit-error rates P_w and P_b over the signal-to-noise ratio E_b/N_0. For the coding schemes two exemplary *perfect codes* are used, the advantage being that the word-error probability can be exactly calculated (see Theorem 4.15) without any bounding techniques. The graph for the uncoded transmission is identical to the results in Table 1.1.

The $(23, 12)_2$ Golay code (see also Example 4.6) with $2^{12} = 4096$ codewords of length 23 is used in Figure 1.13 as an example to demonstrate the principal behaviour of coded transmission over the AWGN channel. In case of small

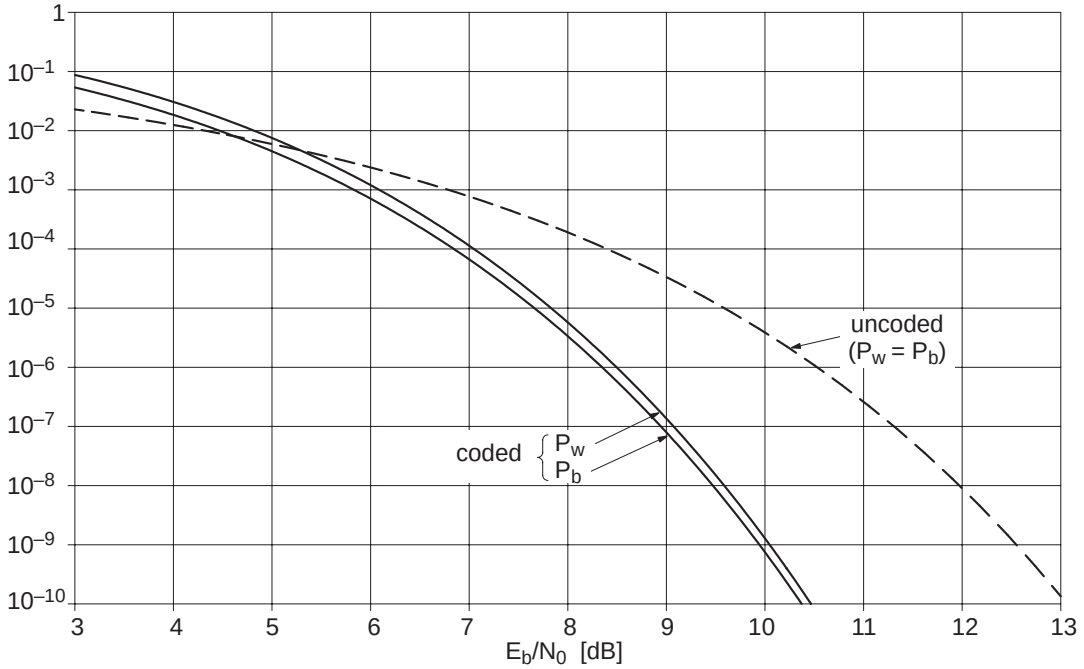

Figure 1.14. Error probability of the $(23, 12)_2$ Golay code with hard decisions

E_b/N_0 (representing poor channel conditions), uncoded transmission is better than coded transmission. However, there is a cross-over point (at about 5 dB for this example) after which a coding gain is achieved. This gain is about 2.0 dB at $P_b = 10^{-6}$ and more than 2.6 dB below $P_b = 10^{-10}$ (representing very good channel conditions). The difference between the error probabilities P_w or P_b is almost negligible.

The term *coding gain* in units of decibels refers exactly to the *horizontal* gap between the coded and uncoded performance curves. In other words this is the achievable reduction of the transmit power for coded transmission with the same error probability after decoding as the error probability of uncoded transmission. Moreover, the *vertical* difference between the curves describes the reduction of the error probability of coded versus uncoded transmission for the same signal-to-noise ratio E_b/N_0.

For excellent channel conditions with $E_b/N_0 \to \infty$ the graphs are almost parallel and they also become steeper and steeper, approaching a gradient of $-\infty$. The horizontal gap between the coded and uncoded transmission is not arbitrarily large as $E_b/N_0 \to \infty$, but converges to a limit, which we shall determine now.

Let E_b denote the energy per information bit for coded transmission and E_b' for uncoded transmission. For uncoded transmission we have P_b as the BSC bit-error probability, hence, according to (1.3.12) and (A.3.18),

$$P_{b,\mathrm{unc}} = p_e = Q\left(\sqrt{\frac{2E_b'}{N_0}}\right) \approx \mathrm{const} \cdot e^{-E_b'/N_0}. \tag{1.7.4}$$

As we will show in Theorem 4.15, for a coded transmission with hard decisions and for large E_b/N_0, we have approximately

$$P_{w,\text{cod}} \approx \text{const} \cdot p_e^{t+1}. \tag{1.7.5}$$

The value const represents a constant which depends on the code, p_e is the BSC bit-error probability of the encoded bits with $E_c = RE_b$, and $t = \lfloor (d_{\min} - 1)/2 \rfloor$ corresponds to about half the minimum distance ($\lfloor \lambda \rfloor$ denotes the biggest integer $\leq \lambda$). Thus, according to (1.7.2) and (A.3.18),

$$\begin{aligned} P_{b,\text{cod}} &\approx \text{const} \cdot P_{w,\text{cod}} \\ &\approx \text{const} \cdot p_e^{t+1} \\ &\approx \text{const} \cdot \left[Q\left(\sqrt{\frac{2RE_b}{N_0}} \right) \right]^{t+1} \\ &\approx \text{const} \cdot e^{-R(t+1) \cdot E_b/N_0}. \end{aligned} \tag{1.7.6}$$

The coding gain refers to equal bit-error rates $P_{b,\text{unc}} = P_{b,\text{cod}}$, implying that

$$\text{const} \cdot e^{-E_b'/N_0} = \text{const} \cdot e^{-R(t+1) \cdot E_b/N_0}. \tag{1.7.7}$$

The constants are, at most, linearly dependent on E_b/N_0 or t and can therefore be neglected while taking the logarithm for large E_b/N_0

$$\frac{E_b'}{N_0} \approx R(t+1) \cdot \frac{E_b}{N_0}. \tag{1.7.8}$$

Thus the *asymptotic coding gain* for hard-decision decoding is

$$G_{\text{a,hard}} = 10 \cdot \log_{10}(R(t+1)) \quad \text{dB}. \tag{1.7.9}$$

For soft-decision decoding we will later show in Theorem 3.17 that

$$P_{w,\text{cod}} \approx \text{const} \cdot e^{-Rd_{\min} \cdot E_b/N_0}. \tag{1.7.10}$$

The comparison to (1.7.4) leads to the following asymptotic coding gain for soft-decision decoding

$$G_{\text{a,soft}} = 10 \cdot \log_{10}(Rd_{\min}) \quad \text{dB}. \tag{1.7.11}$$

For a large E_b/N_0 the error probability decreases exponentially with E_b/N_0:

$$P_b = \text{const} \cdot e^{-E_b/N_0 \cdot \text{const}}. \tag{1.7.12}$$

This result is independent of whether coding was applied or not. For large E_b/N_0 the graphs in Figure 1.13 are parallel with a horizontal gap of $G_{\text{a,hard}}$. The gradients of the graphs converge to $-\infty$ as $E_b/N_0 \to \infty$. This is why the constants in (1.7.7) and also the difference between bit- and word-error

probability as in (1.7.1) or (1.7.2) are not that important because the vertical distance between the P_b and P_w curves remains constant, however, the horizontal distance between the P_b and P_w converges to zero.

The great importance of the minimum distance now becomes immediately clear. The larger $d_{\min}$ is, while keeping the code rate constant (for instance this is achievable with larger block lengths or more generally with better codes), the more E_b (for coded transmission) can be reduced in contrast to E_b' (for uncoded transmission).

Soft decisions generally provide an asymptotic coding gain of about 3 dB, resulting from the approximation $t + 1 \approx d_{\min}/2$:

$$G_{\text{a,soft}} \;\approx\; G_{\text{a,hard}} + 3.01 \text{ dB}. \tag{1.7.13}$$

However, for "realistic" values of E_b/N_0 or "medium" values of P_b the coding gain with soft decisions is usually only in the order of 2 dB.

For the $(23, 12)_2$ Golay code in Figure 1.13 with $t = 3$ the asymptotic coding gain is $G_{\text{a,hard}} = 10 \cdot \log_{10}(12/23 \cdot 4) = 3.2$ dB, however, this result can not be taken from the figure itself, since a bigger E_b/N_0 would have to be considered. For $P_w = 10^{-10}$ or $E_b/N_0 = 10.5$ dB the coding gain is only about 2.6 dB. So, G_{a} is a measure for the comparison of different codes rather than for the calculation of the coding gain for moderate error rates.

Figure 1.14 shows the error probability of the $(7, 4)_2$ Hamming code with $t = 1$. Its coding gain is $G_{\text{a,hard}} = 10 \cdot \log_{10}(4/7 \cdot 2) = 0.6$ dB. The coded transmission is slightly better than the uncoded case, but only if the channel is good or if the error probability is small. High coding gains can only be expected for complex codes, in particular only for big block lengths, which is exactly the statement of the channel coding theorem (see Section 3.2).

Further error probability graphs over E_b/N_0 are given in Section 4.8 when comparing hard- and soft-decision decoding for the Hamming code and in particular in Section 8.1 for RS codes and in Section 8.2 for BCH codes as well as in Section 10.5 for convolutional codes. The analytical calculation of the error probability for block codes can be found in Sections 4.7 and 4.8, for convolutional codes in Section 10.5 and for trellis coded modulation (TCM) in Section 11.7. We will realize that the error probability calculation becomes easier with bigger E_b/N_0. On the other hand, error probabilities can be simulated down to an order of about $10^{-5} \ldots 10^{-6}$ with acceptable processing time. Hence, theory and simulation complement each other almost perfectly.

The channel for a concrete application is usually so complicated that an exact description is impossible to make. Therefore codes are usually designed for the simple BSC or AWGN channel models which also provides a certain amount of robustness against changes of the channel properties. However, models for channels with burst errors (see Section 5.6, 5.7) as well as for fading channels (see Section 11.2, 11.3) are also used.

So it is not surprising that the coding gain in practice is not exactly the same as the theoretical forecast. In addition, there are implementation losses as

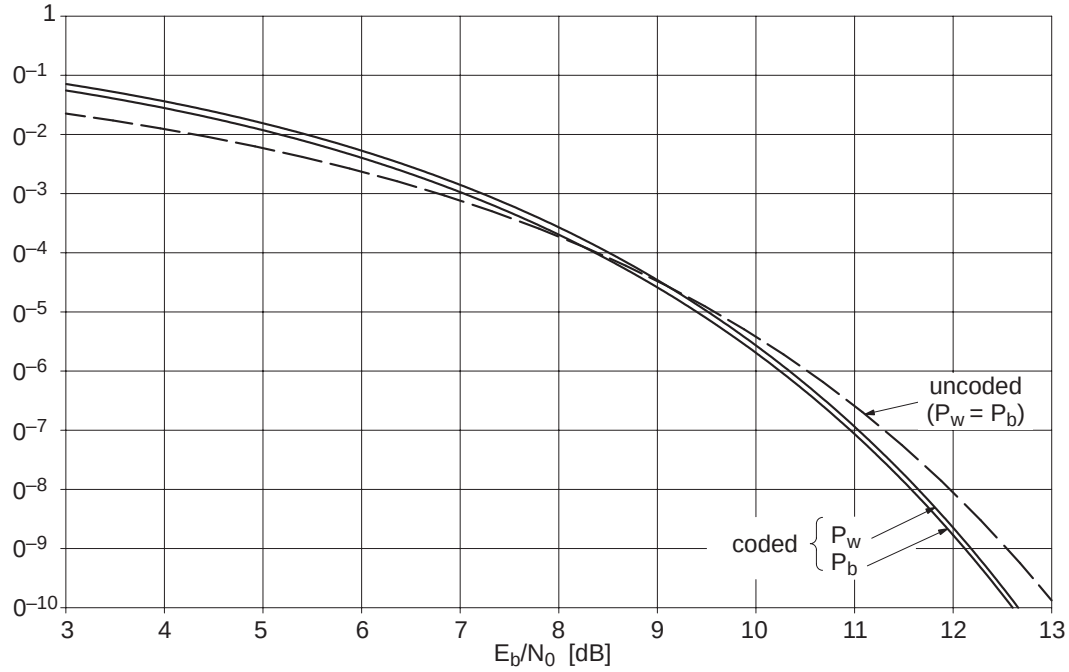

Figure 1.15. Error probability of the $(7, 4)_2$ Hamming code with hard decisions

well as a further effect, since for coded transmission the energy per code bit is smaller than for uncoded transmission. This might make the synchronization of frequency, phase, clock and frame much more difficult than all the operations in the encoder and decoder.

If coding is used when the modulation system and the bandwidth are not changed, while the data rate of the information bits is reduced, the coding gain is $G^*_{\text{a,hard}} = 10 \cdot \log_{10}(t+1)$ dB. In other words, this is the asymptotic difference between coded and uncoded signaling if the graphs are displayed over E_c/N_0 instead of over E_b/N_0 (see also Subsection 2.5.2 and Figure 2.16).

1.8 The Basic Idea of Error-Correction Codes

The concept of block codes helps us to explain the fundamental idea of error-correction coding, since the underlying principle is very similar to the principle of digitization in communications with similar pros and cons:

Digitization is the quantization of the symbols within their range. If Δ is the quantization width, then $\Delta/2$ is the maximum quantization error.

- Advantage: small transmission errors (smaller than $\Delta/2$) are completely eliminated in contrast to analog transmission, where the signal-to-noise ratio gets worse after each amplification.

- Disadvantage: even without transmission errors there always remains an average quantization error of $\sqrt{\Delta^2/12}$. Large transmission errors (greater than $\Delta/2$) are further increased because of the assignment to wrong quantization levels.

- Conclusion: use quantization only if the majority of errors is smaller than $\Delta/2$.

Error-control coding is the quantization of entire long symbol sequences in the time domain. The $n - k$ parity-check symbols are chosen such that the codewords of length n differ in at least $d_{\min}$ positions. This is possible because in the set of q^n possible words only q^k words are actually codewords. As we will prove in detail in Section 4.2:

- Advantage: for less than $d_{\min}/2$ transmission errors the received word is closer to the transmitted word than to any other codeword and can therefore be decoded correctly.

- Disadvantage: for more than $d_{\min}/2$ transmission errors a wrong codeword might be chosen and the number of errors is increased.

- Conclusion: use error-control coding only if there are less than $d_{\min}/2$ errors for the majority of received words. So error-control coding only makes sense for relatively good channels and for high requirements of reliability, whereas it is not recommended for bad channels.

The main idea of error-control coding is to transform long information blocks into even longer code blocks. The amount of redundancy added depends on the expected channel quality and the desired transmission quality.

1.9 Problems

1.1. Assume a BSC with the bit-error probability $p_e = 0.01$. With which probability is the word 0110100 changed to 0010101 during transmission?

1.2. Assume a BSC with $p_e = 0.1$. For words of length 7 determine the probability of the error patterns of all Hamming weights. With which probability do a maximum of 2 or more than 2 errors occur?

1.3. Assume a BSC with $p_e = 0.001$. Determine the approximate probability for more than 2 errors for words of length 31.

1.4. Describe the channel which is formed by the concatenation of two BSC's with the bit-error probabilities $p_{e,1}$ and $p_{e,2}$.

1.5. Prove that $w_H(\boldsymbol{x} + \boldsymbol{y}) \geq w_H(\boldsymbol{x}) - w_H(\boldsymbol{y})$.

1.6. Prove and illustrate

$$d_H(\boldsymbol{x}, \boldsymbol{y}) \geq |d_H(\boldsymbol{x}, \boldsymbol{z}) - d_H(\boldsymbol{z}, \boldsymbol{y})|. \tag{1.9.1}$$

1.7. Prove the *quadrangular inequality*

$$|d_H(\boldsymbol{x}, \boldsymbol{y}) - d_H(\boldsymbol{x}', \boldsymbol{y}')| \leq d_H(\boldsymbol{x}, \boldsymbol{x}') + d_H(\boldsymbol{y}, \boldsymbol{y}'). \tag{1.9.2}$$

1.8. How many different $(3, 2, 2)_2$ block codes are there?

1.9. For the function $f(p_e) = p_e^l(1 - p_e)^{n-l}$, prove and interpret the result of

$$\max_{p_e} f(p_e) = f\left(\frac{l}{n}\right) = 2^{-nH_2(l/n)}, \tag{1.9.3}$$

where $H_2(.)$ is the binary entropy function as given in (A.2.3).

1.10. Assume the $(3, 1, 3)_2$ code $\mathcal{C} = \{(-1; -1; -1), (+1; +1; +1)\}$ and the AWGN channel. For the received word $\boldsymbol{y} = (+0.1; -1.0; +0.1)$ determine the ML estimation with soft and hard decisions.

1.11. Prove without the precondition of equal a priori probabilities that the word-error probability P_w is minimized by the maximum a posteriori decoder which is defined as follows: for the received word $\boldsymbol{y}$ the estimate $\hat{\boldsymbol{a}} \in \mathcal{C}$ is the codeword for which the a posteriori probability $P(x|y) = P(y|x) \cdot P(x)/P(y)$ takes on its maximum, or equivalently expressed:

$$P_x(\hat{\boldsymbol{a}}) \cdot P_{y|x}(\boldsymbol{y}|\hat{\boldsymbol{a}}) \geq P_x(\boldsymbol{b}) \cdot P_{y|x}(\boldsymbol{y}|\boldsymbol{b}) \quad \text{for all } \boldsymbol{b} \in \mathcal{C}. \tag{1.9.4}$$