# 7. Forward Error Correction for Optical Transponders

Alexandre Graell i Amat, Laurent Schmalen

Forward error correction (FEC) is an essential technique required in almost all communication systems to guarantee reliable data transmission close to the theoretical limits. In this chapter, we discuss the state-of-the-art FEC schemes for fiber-optic communications. Following a historical overview of the evolution of FEC schemes, we first introduce the fundamental theoretical limits of common communication channel models and show how to compute them. These limits provide the reader with guidelines for comparing different FEC codes under various assumptions. We then provide a brief introduction to the general basic concepts of FEC, followed by an in-depth introduction to the main classes of codes for soft-decision decoding and hard-decision decoding. We include a wide range of performance curves, compare the different schemes, and give the reader guidelines on which FEC scheme to use. We also introduce the main techniques to combine coding and higher-order modulation (coded modulation), including constellation shaping. Finally, we include a guide on how to evaluate the performance of FEC schemes in transmission experiments. We conclude the chapter with an overview of the properties of some state-of-the-art FEC schemes used in optical communications, and an outlook.

7.1 **History of Forward Error Correction in Optical Communications** ..... 179
7.2 **A Glimpse at Fundamental Limits and Achievable Rates** ..... 183
7.2.1 Achievable Rates for Soft-Decision Decoding ..... 185
7.2.2 Achievable Rates for Hard-Decision Decoding ..... 186
7.2.3 Comparison of Achievable Rates for Bit-Wise and Symbol-Wise Decoding ..... 187
7.2.4 Achievable Rates and Actual Code Performance ..... 189
7.3 **Basics of Forward Error Correction** ..... 189
7.3.1 Linear Block Codes ..... 191
7.3.2 Error Detection and Error Correction Capability of a Block Code over the Binary Symmetric Channel ..... 192
7.3.3 Optimal Decoding of Block Codes ..... 193
7.3.4 Coding Gain and Net Coding Gain ..... 195
7.3.5 Bounds on the Performance of Block Codes ..... 196
7.4 **Soft-Decision Forward Error Correction** 197
7.4.1 Low-Density Parity-Check Codes ..... 197
7.4.2 Spatially Coupled Low-Density Parity-Check Codes ..... 206
7.4.3 Polar Codes ..... 211
7.4.4 Other Coding Schemes ..... 211
7.4.5 Performance Curves ..... 212
7.5 **Hard-Decision Forward Error Correction** ..... 218
7.5.1 Classical Block Codes: Bose–Chaudhuri–Hocquenghem and Reed–Solomon Codes ..... 219
7.5.2 Product Codes ..... 220
7.5.3 Generalized Product Codes ..... 220
7.5.4 Generalized Product Codes as Instances of Generalized Low-Density Parity-Check Codes ..... 223
7.5.5 A General Code Construction of Generalized Product Codes ..... 224
7.5.6 Decoding of Product-Like Codes: Iterative Bounded Distance Decoding ..... 225
7.5.7 Analysis and Optimization of Generalized Product Codes ..... 226
7.5.8 Soft-Aided Decoding of Product-Like Codes ..... 227
7.5.9 Performance Curves ..... 228
7.6 **Coded Modulation—An Introduction** ..... 230
7.6.1 Trellis-Coded Modulation ..... 230
7.6.2 Multilevel Coding ..... 231
7.6.3 Bit-Interleaved Coded Modulation ..... 232
7.6.4 Coded Modulation with Nonbinary Codes ..... 232
7.6.5 Signal Shaping ..... 233
7.6.6 Performance Curves ..... 234




B. Mukherjee et al. (Eds.), *Springer Handbook of Optical Networks*, Springer Handbooks, https://doi.org/10.1007/978-3-030-16250-4_7

7.7 **Evaluating Forward Error Correction Performance in Transmission Experiments** ..... 236
7.7.1 Threshold-Based Forward Error Correction Performance Prediction ..... 237
7.7.2 Implementing Forward Error Correction in Experiments ..... 242
7.7.3 Performance Example ..... 244
7.8 **Conclusion and Outlook** ..... 246
**References** ..... 248

The fundamental problem of communication is that of sending information from one point to another (either in space or time) efficiently and reliably. Examples of communication are the data transmission between a ground station and a space probe through the atmosphere and free space, the write and read of information on flash memory, and communication between servers in a data center through optical interconnects. The transmission over the physical medium, e.g., the optical fiber, is prone to noise due to amplification, distortion, and other impairments. As a consequence, the transmitted (digital) message may be received with errors.

To reduce the probability of error, one could (if possible) increase the signal-to-noise ratio (SNR), i.e., transmit at a higher power, or transmit using a larger bandwidth. In his landmark 1948 paper [7.1], *Claude E. Shannon* showed that there is a third parameter one can play with: the system complexity. Shannon proved that, for any given channel (e.g., a given SNR and bandwidth), there is a fundamental limit, the so-called *channel capacity*, at which information can be transmitted reliably, i.e., with diminishing error probability. Aside from determining the channel capacity, he also showed how this limit can be achieved: by the use of coding.

Shannon's contribution marks the birth of the fields of information theory and coding theory. While information theory aims at determining the fundamental limits of communication, coding theory aims at finding *practical* codes that achieve (or approach) these limits. The principle of coding is very simple: to introduce redundancy in the transmitted sequence in a controlled manner, such that it can be exploited by the receiver to correct the errors introduced by the channel. However, while Shannon proved the *existence* of capacity-achieving codes, he did not provide any insight on how to construct *practical codes*. His proof was based on a random-coding argument. Informally, the argument says that a random code will achieve capacity with high probability. Unfortunately, random codes are not practical, as both encoding and decoding require computational complexity and storage that grow exponentially with the block length (and Shannon showed that to achieve capacity, large block lengths—in truth infinitely large—are required!). Therefore, since the late 1940s, finding practical codes that are able to perform close to capacity with reasonable decoding complexity has been the holy grail of coding theory. For decades, researchers worked to develop powerful codes with a rich algebraic structure, among them the celebrated Bose–Chaudhuri–Hocquenghem (BCH) codes and Reed–Solomon codes, which unfortunately still performed far from the channel capacity with feasible decoding algorithms. Two breakthroughs came in the 1990s, the first when two researchers, *Claude Berrou* and *Alain Glavieux*, introduced turbo codes, the first class of codes that was shown to approach capacity with reasonable decoding complexity, and the second with the rediscovery of low-density parity-check (LDPC) codes (originally introduced in 1960 by *Robert Gallager*) by *David MacKay*. The advent of turbo codes and the rediscovery of LDPC codes constituted a cornerstone in coding theory and gave birth to what nowadays is referred to as the field of *modern coding theory*, which is also intimately related to the iterative decoding of these codes. LDPC codes and turbo codes have now been adopted in a plethora of communication standards.

The application of coding goes far beyond the classical communication and storage problem. Coding is a very powerful, fundamental tool that plays a key role in many other applications and research fields, including distributed storage and caching [7.2, 3], uncoordinated multiple access [7.4], to speed up distributed computing tasks [7.5, 6], for quantum key distribution [7.7], and post-quantum cryptography [7.8].

Coding theory is a vast field that is impossible to cover in a single book chapter. This is certainly not our aspiration. The aim of this chapter is simply to provide a brief and accessible introduction to the main concepts underlying the art of coding and a simple but rather complete overview of the main coding schemes that will arguably be used in future optical communications systems. It is intended to be an entrance door to the exciting and important realm of coding for researchers in the field of optical communications that, while perhaps not ambitioning to become experts on coding, would like to become familiar with the field or are simply interested in widening their knowledge. It is also a handy source intended for engineers and practitioners of optical communication systems who are faced with the choice of a coding scheme.

One of the purposes of this chapter is to review the principal coding schemes for optical communications.

Because of the limited space, while we briefly describe classical codes such as BCH and Reed–Solomon codes, the main focus of the chapter is on *modern codes*, in particular LDPC codes and product-like codes such as staircase codes. Modern codes are also commonly referred to as *graph-based* codes or *codes on graphs*, since they can be conveniently described by means of a graph. LDPC and product-like codes are the two main code classes currently being used in fiber-optic communications with soft-decision and hard-decision decoding, and will likely continue being adopted in future systems. We will briefly describe the main building blocks of these codes, which are *rooted* in classical coding theory.

The remainder of the chapter is organized as follows. The chapter starts with a brief overview of the history of coding, with a special focus on the field of optical communications. In Sect. 7.2, an introduction to the fundamental performance limits, to which the performance of practical coding schemes can be compared, is provided. We then introduce the basic definitions and concepts of coding in Sect. 7.3. Sections 7.4 and 7.5 are devoted to soft-decision forward error correction (FEC) and hard-decision FEC, respectively. In Sect. 7.6, the combination of coding and higher-order modulation, dubbed *coded modulation*, is discussed, and some basics of constellation shaping are given. In Sect. 7.7, we discuss how to evaluate the performance of FEC schemes in transmission experiments. Finally, in Sect. 7.8 we draw some conclusions and provide an outlook on open research problems.

Notation: The following notation is used throughout the chapter. We use boldface letters to denote vectors and matrices, e.g., $\boldsymbol{x}$ and $\mathbf{X}$, respectively, and calligraphic letters to denote sets, e.g., $\mathcal{X}$. The cardinality of a set $\mathcal{X}$ is given by $|\mathcal{X}|$. The real and imaginary parts of a complex number are denoted by $\mathrm{Re}\{\cdot\}$ and $\mathrm{Im}\{\cdot\}$, respectively. The probability mass function (PMF) and the probability density function (PDF) of a random variable (RV) $X$ are denoted by $P_X(x)$ and $p_X(x)$, respectively, as shorthand for $P_X(X=x)$ and $p_X(X=x)$. Sometimes we will simply write $P_X$ and $p_X$. Also, we write $p_{XY}(x,y)$ to denote the joint PDF of two RVs $X$ and $Y$, and $p_{Y|X}(y|x)$ to denote the conditional PDF of $Y$ conditioned on $X=x$. We use similar notation for the joint and conditional PMFs. Probability is denoted by $\Pr(\cdot)$ and expectation by $\mathbb{E}$. More precisely, the expectation of a function $g(X)$ with respect to $p_X$ is denoted as $\mathbb{E}_{p_X}[g(x)]$, or, in short, $\mathbb{E}_X[g(x)]$, i.e., $\mathbb{E}_X[g(x)] = \int g(x)p_X(x)\mathrm{d}x$. Likewise, the expectation of a function $g(x)$ with respect to a (discrete) PMF $X$ is given by $\mathbb{E}_{P_X}[g(x)] = \sum_x g(x)P_X(x)$. Vector and matrix transpose are denoted by $(\cdot)^\top$, and $\|x\|$ denotes the $\ell_2$-norm of $x$; $\mathbb{1}_{\{s\}}$ denotes the indicator function returning 1 if the statement s is true and 0 otherwise.

## 7.1 History of Forward Error Correction in Optical Communications

Forward error correction (FEC) in optical communications was first demonstrated in 1988 [7.9]. Since then, it has evolved significantly. This pertains not only to the techniques themselves but also to the encoder and decoder architectures. The history of FEC is, however, much older and has found more widespread application than only optical communications. In this section, we give a historical overview of FEC and link the historical developments to the developments in fiber-optic communications.

The first error-correcting codes, introduced by *Richard Hamming* in 1950 [7.10], were binary linear block codes that were designed to improve the reliability of electromechanical computing machines [7.11, 12]. Hamming codes are based on simple algebraic descriptions and have a straightforward hard-decision decoding (HDD) algorithm to correct single errors. Hamming codes were one of the first codes applied in optical communications. In 1988, *Wayne Grover* [7.9] reported a performance improvement not only with respect to the required received power and the observed residual error floor, but also with respect to chromatic dispersion tolerance and resilience to laser mode hopping.

Following the introduction of Hamming codes and the emergence of the field of coding theory, many new classes of codes were introduced, some with important properties and applications. One very powerful early code was the binary triple-error-correcting Golay code of length 23 bit and its ternary double-error-correcting counterpart (length 11 ternary digits) that were constructed by the Swiss engineer *Marcel Golay* in 1949 [7.13]. We should note that, while Hamming only published his work in 1950 after Golay's code, his work was already known to Shannon in 1948 and highlighted in [7.1], and hence several authors consider Hamming codes to be the first error-correcting codes. It is surprising that the ternary Golay code was in fact first discovered by the Finnish football fan *Juhani Virtakallio*, who published it in the 1947 issue of the Finnish football magazine *Veikkaaja* as a betting system [7.14]. The paper [7.14] gives an excellent overview of the early days of coding theory and its connections with neighboring scientific fields. The Golay

code is an exceptional code, as it is the *only* binary *perfect code* that can correct more than a single error ([7.15, Chap. 6, Sect. 10]). This means that there is no other binary code of smaller size that can correct up to three errors with HDD. The Golay code has found widespread application, and it was famously used in the two 1977 Voyager space missions [7.11]. The introduction of Hamming codes and the Golay code is a landmark in coding theory. For this reason, together with Shannon, Hamming and Golay are considered the fathers of the mathematical discipline of coding theory [7.15].

The 1950s saw the development of many new coding schemes that could correct multiple errors, for example the important Reed–Muller codes, with a construction first presented by *David Muller* in 1954 [7.16], and the decoding algorithm given by *Irving Reed* in the same year [7.17]. Also in 1954, *Peter Elias* introduced product codes (PCs) [7.18], a very relevant class of codes and the first example of codes built from smaller component codes. PCs are adopted nowadays for data storage on magnetic media and flash memory [7.19, 20], in the WiMAX (Worldwide Interoperability for Microwave Access) wireless communication standard [7.21], and for transmission over fiber-optic links [7.22]. Their excellent performance is due to a decoding algorithm that iterates between the decoding of the component codes. This iterative decoding algorithm was not originally envisaged by Elias, but only a few decades later, which explains why PCs were not initially adopted after their invention. A year later, *Elias* himself introduced the concept of convolutional codes [7.23] as an alternative to the then dominant block codes. In contrast to block codes, convolutional codes contain memory, and the output of the decoder depends not only on the input at the current time instant, but also on previous inputs. Because of their excellent performance with relatively low decoding latency, they have found their way into numerous communication standards. They are used in satellite and spacecraft links [7.11], for cellular radio (e.g., Global System for Mobile Communications (GSM)) [7.24], wireless local area networks [7.25], and Ethernet cables [7.26]. A particularly popular class of convolutional codes is based on simple binary linear shift registers. In this case, the encoder is a simple shift register with only a few memory elements. The key enabler for the success of convolutional codes was the Viterbi algorithm, introduced in 1967 [7.27, 28], which provides an efficient means of implementing optimal maximum likelihood (ML) decoding. The Viterbi algorithm enabled—for the first time—low-complexity soft-decision decoding (SDD) exploiting the full output of the channel. This represented a big step forward, as we will later see in Sect. 7.4. Despite their widespread use in many communication standards, convolutional codes have not yet found widespread use in fiber-optic communications. This is mostly due to the fact that parallelization of the decoder is not trivial, and to the availability of stronger codes at the time coding was adopted in fiber-optic communications. However, some notable exceptions are an early study of on-off-keying (OOK) modulation with different types of error-correcting codes, including convolutional codes [7.29], a paper analyzing the combination of convolutional codes and bandwidth-efficient modulation formats for fiber-optic communications [7.30], and a paper describing the use of low-rate convolutional codes to protect some levels in multilevel coding schemes [7.31].

The 1950s ended with the invention of one of the most important coding schemes still in use today, BCH codes, which were independently introduced by two groups [7.32, 33]. BCH codes are a class of cyclic error-correcting codes defined over a finite field that are able to correct multiple bit errors using simple syndrome decoding based on efficient closed-form solutions for computing the error locations [7.15, 34, 35]. After Hamming codes, single-error-correcting BCH codes of length 65 kbit, and two-error- and three-error-correcting BCH codes [7.36] were the next codes adopted in fiber-optic communication systems. These codes have overhead of less than 1%, while allowing the post-FEC bit error rate (BER) to be met with a channel pre-FEC BER of approximately $10^{-6}$.

Shortly after the introduction of BCH codes, *Reed* and *Gustave Solomon* [7.37] introduced nonbinary cyclic error-correcting codes that are able to correct and/or detect a predetermined number of erroneous nonbinary symbols, consisting of groups of bits. The corresponding algebraic decoding algorithm was proposed by *Elwin Berlekamp* [7.38] and *James Massey* [7.39]. A widely used code is the Reed–Solomon code RS(255, 239) code (the notation will be introduced in Sect. 7.3) consisting of 255 nonbinary symbols, with each symbol composed of 8 bits. This code can correct up to eight symbol errors (or a single burst error of up to 57 bits) with 6.7% overhead and achieves a net coding gain (NCG) of 6.2 dB in SNR at a post-FEC BER of $10^{-15}$. This means that we can tolerate an effective net SNR, taking into account the extra bandwidth required for the coding overhead, that is four times lower than without coding. This Reed–Solomon code was the first to find widespread use in fiber-optic communication systems, starting from submarine transmission [7.40], to metro and optical access systems [7.41]. Because of its low encoder and decoder implementation complexity, this code is still in use for applications with low coding gain requirements

and strict constraints on transceiver power consumption. It is included in several standards and described in detail in the ITU-T G.975 standard [7.42]. This code is commonly referred to as *first-generation* coding for light-wave systems [7.43].

In the decades that followed, coding research focused mostly on the construction of new, more powerful codes from the now existing schemes (i.e., following the same principle proposed originally by Elias with his PCs!). One such construction was the concatenation of simple component codes (e.g., convolutional codes, BCH codes, and Reed–Solomon codes), leading to long and powerful codes capable of achieving significant coding gains with low decoding complexity. Concatenated codes were first introduced by *Dave Forney* in 1965 in the form of serially concatenated codes (SCCs), with a cascaded *inner* and *outer* code [7.12, 44]. At the receiver, both component codes are decoded separately, resulting in much lower complexity compared with optimal ML decoding of the entire concatenated code, at the cost of some performance degradation. Typically, the inner code is a convolutional code decoded using SDD, while the outer code is a block code, usually a Reed–Solomon code, decoded using HDD. Concatenated codes soon became a popular code construction for satellite communications. A simple concatenated coding scheme was used already in the 1971 Mariner Mars orbiter mission, but regular use of concatenated codes for deep space communications began with the Voyager program. A concatenated code with an inner convolutional code decoded using Viterbi decoding and an outer Reed–Solomon code was first used in the 1977 NASA space mission Voyager 2, which was launched with the goal of studying the outer planets of our solar system. Concatenated codes are also used in the digital television broadcast standard DVB-S [7.45].

The so-called *second-generation* FEC schemes for fiber-optic communications are based mainly on PCs and concatenated codes, where the component codes are simple algebraic codes such as Hamming, BCH, or Reed–Solomon codes, or convolutional codes. The decoding of concatenated block codes is usually performed iteratively by alternately decoding each of the component codes, improving the overall result after each iteration. Almost all 6.7% overhead codes specified in the ITU-T G.975.1 standard [7.46] are concatenated codes. For instance, the code defined in [7.46, Appendix I.2] is composed of a convolutional inner code and a Reed–Solomon outer code. Similarly, the code defined in [7.46, Appendix I.5] is composed of an inner Hamming code and an outer Reed–Solomon code. All concatenated codes specified in [7.46] are very long codes, with block lengths of around 500 kbit. These codes achieve very good performance, with low error floors, and have moderate implementation complexity. For example, the concatenated code specified in [7.46, Appendix I.9], with inner and outer BCH codes, has an NCG of 9.24 dB and provides an output BER of less than $10^{-16}$ at an input BER of about $4\times10^{-3}$.

Second-generation coding schemes continue to evolve today for low-complexity applications such as metro networks [7.47] or data center interconnects. Three very recent developments in second-generation HDD FEC codes are continuously interleaved codes [7.48], staircase codes [7.49], and braided BCH codes [7.50], which can be considered generalizations of Elias' PCs with an additional convolutional-like structure. The NCG of the staircase code in [7.49] amounts to 9.41 dB at a BER of $10^{-15}$, which is only 0.56 dB from the capacity of the hard-decision channel [7.49].

With the advent of coherent transmission schemes and high-resolution analog-to-digital converters (ADCs), SDD became an attractive means of increasing the NCGs and hence the transmission reach. Although SDD is the natural way to decode a code exploiting the full knowledge of the transmission statistics, HDD was prevalent in early fiber-optic communication systems due to the lack of ADCs and the requirements for simple high-speed receivers that carry out a binary 0/1 decision using analog radio-frequency (RF) circuits.

A breakthrough in the history of coding occurred in 1993, when Berrou and Glavieux introduced turbo codes, together with a simple iterative SDD algorithm, the *turbo principle* (due to its similarities to a turbo engine). Turbo codes are built from the parallel concatenation of two convolutional codes and were the first codes to approach capacity with acceptable decoding complexity. After the invention of turbo codes, a sudden interest in iteratively decodable codes arose in the research community. This led to the rediscovery of LDPC codes by *MacKay* [7.51, 52]. LDPC codes were originally introduced in 1960 by *Gallager* in his landmark PhD thesis [7.53, 54]. In his thesis, Gallager also proposed algorithms for both HDD and SDD. However, LDPC codes were not further investigated for many years because of the perceived higher decoding complexity for long codes. In the years that followed the invention of turbo codes and the rediscovery of LDPC codes, numerous publications from various researchers paved the way for a thorough understanding of these classes of codes, leading to numerous applications in various communication standards, such as WLAN (IEEE 802.11) [7.55], DVB-S2 [7.56], and 10G Ethernet (IEEE 802.3) [7.57].

In the realm of optical communications, FEC schemes with (iterative) SDD are typically classified as

*third-generation* FEC schemes. Today, there are largely two competing classes of codes that allow for SDD and that are attractive for implementation in optical receivers at decoding throughputs of 100 Gbit/s and above. The first class corresponds to the natural extension of the second-generation decoding schemes, i.e., PCs with SDD. Iterative low-complexity SDD of PCs was pioneered by *Ramesh Pyndiah* in 1998 [7.58], following the invention of turbo codes. Because of their similarities with turbo codes, PCs decoded using SDD are also often denoted as turbo product codes (TPCs) or block turbo codes (BTCs). The error floor of these codes tends to be low, with a steep BER slope. However, they require large block lengths to realize a small overhead, leading to a larger decoding latency. A further disadvantage is that flexibility with respect to varying frame sizes and overheads is more difficult to realize. Additionally, the decoder implementation requires the tweaking of many parameters to achieve the best possible performance. Furthermore, with overheads of more than 15% to 20%, these codes no longer perform well. The application of BTCs with iterative SDD in the context of fiber-optic communications was demonstrated in [7.59]. The second popular class of soft-decision decodable codes in optical communication systems is LDPC codes. One instance of an LDPC code has already been standardized in ITU-T G.975.1 ([7.46, Appendix I.6]). However, it was designed mostly for HDD. LDPC codes for SDD in optical communications were originally studied in [7.60], with work proposing complete coding schemes and frame structures shortly thereafter [7.61]. The popularity of LDPC codes is chiefly due to the existence of a suboptimal, conceptually very simple, low-complexity decoder for SDD: the belief propagation (BP) decoder. In addition to binary LDPC codes, nonbinary LDPC codes were promoted for optical communication systems in [7.62]. In short, nonbinary codes operate on symbols (e.g., modulation symbols), rather than bits. They come, however, at a cost of higher decoding complexity.

LDPC codes often suffer from an *error floor* at BERs of $10^{-8}$–$10^{-10}$ if no extra preventive measures are taken. Above a certain SNR, the BER no longer drops rapidly, but follows a curve with a smaller slope [7.63]. This effect is due to some structures in the code that lead to error patterns that cannot be recovered with BP decoding. Some modern high-performance FEC systems are therefore often constructed using an LDPC inner code decoded using SDD, which reduces the BER to a level of $10^{-3}$–$10^{-5}$, and an outer block code (typically a BCH code or a Reed–Solomon code) decoded using HDD, which pushes the system BER to levels below $10^{-15}$ [7.61]. The implementation of a coding system with an outer cleanup code requires a thorough understanding of the LDPC code, the decoding algorithm, and its error mechanisms, and a properly designed interleaver between the LDPC code and the outer code so that errors at the output of the LDPC decoder—which typically occur in bursts—do not cause uncorrectable blocks after outer decoding. With increased computing resources, it is now also feasible to evaluate very low target BERs of LDPC codes and design LDPC codes that have very low error floors below the maximum acceptable BER [7.64].

Over the past decade, the field of coding theory has seen two important milestones: polar coding and spatial coupling. In 2008, *Erdal Arıkan* introduced the concept of channel polarization [7.65, 66], which led to the new concept of *polar codes*. The design of polar codes is based entirely on information-theoretic principles and differs radically from previously known code constructions. Despite the beauty of their construction and the fact that they are capacity-achieving with a simple *successive cancellation decoder*, which successively decodes the single bits of the codeword, polar codes are not immediately suited for practical application in optical communications. While polar codes show competitive performance for short block lengths with an improved *list decoder* [7.67] (which is one of the reasons they were chosen for transmitting control messages in the 5G radio standard [7.68]), for long block lengths, frequently used in optical communications to achieve high coding gains, their performance is not yet comparable to that of state-of-the-art LDPC codes. Furthermore, the decoder is inherently sequential, complicating their application in high-speed communication systems requiring high levels of parallelization. These drawbacks are currently topics of active research [7.69, 70].

The second breakthrough in the past decade was the introduction of the concept of *spatial coupling*. Spatial coupling itself has its own history. In 1999, *Alberto Jiménez Feltström* and *Kamil Zigangirov* investigated LDPC convolutional codes [7.71], a generalization of LDPC codes with a superimposed convolutional structure. It was only later fully realized that these codes, now known as *spatially coupled* LDPC (SC-LDPC) codes, possess outstanding performance. In fact, *Michael Lentmaier* and others realized that the asymptotic decoding performance of a certain class of SC-LDPC codes approaches the performance of optimal decoding [7.72–74] with simple suboptimal BP decoding. This effect, dubbed *threshold saturation*, was later rigorously proven in [7.75, 76]. The main advantage of spatially coupled codes is that it is possible to construct very powerful codes that can be decoded with a simple windowed decoder [7.77]. Spatial coupling is a very general concept that applies to other

classes of codes, such as turbo-like codes [7.78], as well as to other scenarios, including lossy compression [7.79], code-division multiple access [7.80], and compressed sensing [7.81]. Some product-like codes such as staircase codes and braided codes can also be seen as instances of spatially coupled codes.

Spatially coupled codes are now emerging in various forms. Two examples are the staircase code and the braided BCH code presented in [7.49] and [7.50], respectively, which are both codes with 6.7% overhead targeted for low-complexity applications with HDD (Sect. 7.5.3). Both codes can be classified as spatially coupled BCH PCs, and hence enable a natural windowed decoder implementation. Spatially coupled codes and polar codes are also being considered for SDD in optical communications [7.82–84]. However, as of today, the commercial application of polar codes in high-speed fiber-optic communications still seems further away than that of spatially coupled codes. The latter are a very general and broad concept that allows one to update many existing schemes and to reuse existing decoder hardware, at least to some extent. Therefore, we believe that the commercial application of spatially coupled codes is viable as of today.

The evaluation of the different generations of coding schemes for fiber-optic communications is shown in [7.85, Fig. 1.7]. In summary, we can say that within the span of only two decades, FEC in fiber-optic communication systems has evolved from the use of very simple schemes toward the implementation of state-of-the-art coding schemes in conjunction with high-speed decoding engines that operate at low complexity. Today, modern high-speed optical communication systems require high-performing FEC systems featuring throughput of 100 Gbit/s or multiples thereof, low power consumption, coding gains close to the theoretical limits, and adaptation to the peculiarities of the optical channel [7.86]. The stringent complexity requirements have even driven the investigation of new coding schemes, such as staircase codes [7.49], which are probably the first class of codes proposed specifically for optical transmission.

## 7.2 A Glimpse at Fundamental Limits and Achievable Rates

Before moving into the basics of coding, in this section we review the fundamental limits against which the performance of the coding schemes discussed in this chapter can be compared.

A simplified block diagram of a communication system is depicted in Fig. 7.1. The block of $k$ source data bits $\boldsymbol{u} = (u_1, \ldots, u_k)$ is first encoded by an error-correcting code into a codeword $\boldsymbol{c} = (c_1, \ldots, c_n)$ consisting of $n$ code bits, which is then modulated into a sequence of $\bar{n}$ constellation symbols $\boldsymbol{x} = (x_1, \ldots, x_{\bar{n}})$ that is transmitted over the channel. The received sequence $\boldsymbol{y}$ is demodulated and fed to the decoder, which aims at recovering the source data.

If the channel is memoryless, $p_{\boldsymbol{Y}|\boldsymbol{X}}(\boldsymbol{y}|\boldsymbol{x}) = \prod_i p_{Y|X}(y_i|x_i)$. In other words, the channel is completely specified by the conditional probability $p_{Y|X}(y|x)$ of receiving a single symbol $y$ when a single symbol $x$ is transmitted. The probability $p_{Y|X}(y|x)$ is often referred to as the *channel law*. Because of its prevalence and widespread use in modeling, throughout the chapter we will focus mainly on the case of memoryless channels with additive white Gaussian noise (AWGN), for which the channel law per real dimension (i.e., in each I/Q dimension in the case of complex constellation symbols) is expressed as

$$p_{Y|X}(y|x) = \frac{1}{\sqrt{2\pi}\sigma} \mathrm{e}^{-\frac{(y-x)^2}{2\sigma^2}} , \tag{7.1}$$

where $\sigma^2$ is the noise variance. We note that when using complex constellations, we often assume that the noise of the real and imaginary parts are independent. In this case, the circularly symmetric complex channel law is expressed as

$$\begin{aligned} p_{Y|X}(y|x) &= p_{Y_I|X_I}(\mathrm{Re}\{y\}|\,\mathrm{Re}\{x\}) p_{Y_Q|X_Q}(\mathrm{Im}\{y\}|\,\mathrm{Im}\{x\}) \\ &= \frac{1}{\pi N_0} \mathrm{e}^{-\frac{\|y-x\|^2}{N_0}} \end{aligned}$$

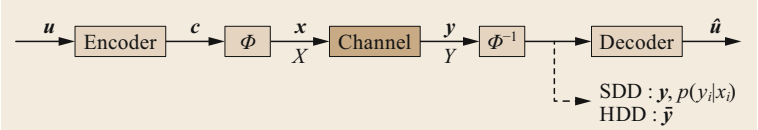


**Fig. 7.1** Simplified block diagram of a communication system. For SDD, the decoder estimates the transmitted codeword based on $\boldsymbol{y}$ (equivalently the transition probabilities $p(y_i|x_i)$), while for HDD the decoder estimates the transmitted codeword based on the hard-detected sequence $\bar{\boldsymbol{y}}$

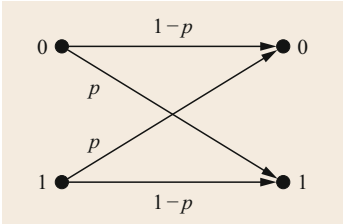


**Fig. 7.2** The BSC. A bit is received correctly with probability $1-p$ and flipped with probability $p$

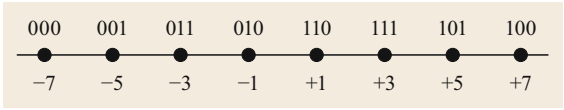


**Fig. 7.3** 8-PAM constellation with binary reflected Gray code labeling. The channel input $X$ takes values on a set of 8 different symbols

The SNR is defined as $\mathrm{SNR} = E_\mathrm{s}/N_0$, where $E_\mathrm{s}$ is the average energy per symbol and $N_0/2 = \sigma^2$ the double-sided noise power spectral density.

We will also consider the binary symmetric channel (BSC), which is an important channel model for HDD. In fact, for an AWGN channel and binary phase-shift keying (BPSK) modulation, if hard detection is performed at the output of the channel, the resulting discrete memoryless channel between the encoder and the decoder can be modeled as a BSC with bits at the input and at the output, where a bit is flipped with probability

$$p = Q\left(\sqrt{\frac{2E_\mathrm{s}}{N_0}}\right) \tag{7.2}$$

and is received correctly with probability $1-p$. In (7.2), $Q(x) = (1/\sqrt{2\pi})\int_x^\infty \exp\left(-\tau^2/2\right)\mathrm{d}\tau$ is the tail probability of the standard normal distribution. The BSC is depicted in Fig. 7.2.

Let $X$ be the channel input that takes values on the alphabet $\mathcal{X} = \{\mathtt{X}_1, \mathtt{X}_2, \ldots, \mathtt{X}_M\} \subset \mathbb{C}$, i.e., $\mathcal{X}$ is the constellation of cardinality $M$ imposed by the modulation (e.g., $M = 16$ for a 16-QAM format) and $Y$ the channel output. Note that $X$ and $Y$ are random variables ($x$ and $y$ are their realizations). We assume that the channel output $Y$ is a continuous random variable. As an example, consider transmission using the eight-pulse-amplitude modulation (PAM) constellation depicted in Fig. 7.3. In this case, $X$ is a random variable that takes values on the set $\mathcal{X} = \{-7, -5, -3, -1, +1, +3, +5, +7\}$ according to a given distribution $P_X$. For conventional constellations, all symbols are equiprobable, i.e., $P_X(\mathtt{X}_i) = 1/M$ for all $\mathtt{X}_i$.

For a given distribution $P_X$ of the channel input, the mutual information (MI) between the channel input $X$ and channel output $Y$,

$$\begin{aligned} I(X;Y) &\triangleq \sum_{x\in\mathcal{X}} P_X(x) \int p_{Y|X}(y|x) \\ &\quad \times \log_2 \frac{p_{Y|X}(y|x)}{\sum_{x'\in\mathcal{X}} p_{Y|X}(Y|x')P_X(x')}\mathrm{d}y \\ &= \mathbb{E}_{XY}\left[\log_2\left(\frac{p_{Y|X}(Y|X)}{\sum_{x'\in\mathcal{X}} p_{Y|X}(Y|x')P_X(x')}\right)\right], \end{aligned} \tag{7.3}$$

determines the upper limit on the achievable rate. Here, by rate we mean the transmission rate, denoted by $R$, in bits per symbol (or bits per channel use). Note that in (7.3), the expectation is with respect to the PDF $P_X p_{Y|X}$. The rate $R$ is sometimes referred to as spectral efficiency in the literature. This implicitly assumes ideal pulse shaping with infinitely extended $\sin(x)/x$ pulses and transmission at the Nyquist rate. In practice, the spectral efficiency (given in (bit/s)/Hz) and the transmission rate (given in bit/symbol or bit/channel use) often differ by a *spectral utilization factor* that depends on the pulse shaping, pre-emphasis, and other system parameters.

A rate $R$ is called achievable if there exists a sequence of coding schemes such that the probability of a decoding error can be made arbitrarily small in the limit of infinitely large block lengths $n$. In other words, it is possible to transmit with vanishing error probability if $R < I(X;Y)$. Conversely, a sequence of coding schemes with vanishing error probability must have $R \le I(X;Y)$, and transmitting at a rate $R > I(X;Y)$ will always result in a probability of error bounded above zero. The highest possible achievable rate is given by the channel capacity, obtained by maximizing $I(X;Y)$ over all possible input distributions,

$$C \triangleq \max_{P_X} I(X;Y)\,. \tag{7.4}$$

Achieving the rate of (7.3) is possible with an optimal decoder or a typical sequence decoder for the channel $p_{Y|X}(Y|X)$ (in the limit of infinitely large block lengths). Furthermore, for some channels, suboptimal decoders may also achieve (7.3). In general, a communication system imposes some constraints on the transmitter and on the receiver (e.g., detection method, decoding strategy such as SDD or HDD), and thus for a specific communication system, the MI in (7.3) is not necessarily achievable. A lower bound on the true MI can be obtained using the mismatched decoding framework [7.87, 88]. The key idea behind mismatched decoding is to design a receiver that is optimal for an auxiliary channel (a good approximation of the true channel) and then computing (7.3) by averaging with respect to the true channel but using in the argument of the logarithm the conditional distribution of the auxil-

iary channel, i.e.,

$$\tilde{I}(X;Y) \triangleq \mathbb{E}_{XY}\left[\log_2\left(\frac{q_{Y|X}(Y|X)}{\sum_{x'\in\mathcal{X}} q_{Y|X}(Y|x')P_X(x')}\right)\right], \tag{7.5}$$

where $q_{Y|X}(Y|X)$ is the conditional distribution of the auxiliary channel and the expectation is taken with respect to $P_X p_{Y|X}$. The rate $\tilde{I}(X;Y)$ is achievable by using the optimal decoder for the auxiliary channel $q_{Y|X}(Y|X)$ over the real channel and is a lower bound of the MI, i.e., $I(X;Y) \geq \tilde{I}(X;Y)$.

Note that, in practice, the channel may not be memoryless. In this case, $X$ and $Y$ should be interpreted as vectors, $\boldsymbol{X}$ and $\boldsymbol{Y}$, and the conditional distribution $p_{\boldsymbol{Y}|\boldsymbol{X}}(\boldsymbol{Y}|\boldsymbol{X})$ should be considered. The fiber-optic channel, indeed, has memory. In this section, to compute achievable rates, we make the common assumption that the memory of the fiber-optic channel can be neglected. In practice, most common receivers neglect the memory of the channel or remove it by sufficient interleaving; hence the memoryless assumption yields meaningful achievable rates for actual receivers.

To achieve high spectral efficiencies and performance close to the theoretical limits, fiber-optic communications employ coding in combination with higher-order constellations, a scheme known as coded modulation (CM). At the receiver side, both SDD and HDD may be used. For SDD, the decoder estimates $\boldsymbol{c}$ based on the full observation $\boldsymbol{y}$, i.e., $\boldsymbol{y}$ (or, equivalently, the transition probabilities $p(y_i|x_i)$) is fed to the decoder. For HDD, the demodulator takes hard decisions at the channel output, and the sequence of hard-detected symbols, denoted by $\bar{\boldsymbol{y}}$, is fed to the decoder (Fig. 7.1). Furthermore, binary codes and nonbinary codes may be employed, which lead to two decoding strategies, namely bit-wise decoding and symbol-wise decoding, respectively. In the following, we briefly discuss achievable rates for SDD and HDD with both bit-wise and symbol-wise decoding. We restrict ourselves to the conventional uniform input distribution, i.e., $p_X(x) = 1/M$ for all $x \in \mathcal{X}$. A brief discussion of CM with probabilistic shaping is provided in Sect. 7.6.

For further reading, achievable rates for fiber-optic communications assuming a memoryless channel have been addressed in [7.89, 90]. Achievable rates for probabilistic shaping are discussed in [7.91]. The capacity of the nonlinear optical channel with finite memory is addressed in [7.92, 93], where the joint effect of nonlinearity and dispersion was modeled as a finite-state machine and the capacity was estimated. Capacity lower bounds for a variety of scenarios were obtained in [7.94]. Achievable rates for a nonlinear wavelength-division-multiplexed (WDM) fiber-optic system using the mismatched decoding framework were obtained in [7.95].

### 7.2.1 Achievable Rates for Soft-Decision Decoding

In the following, we review some achievable rates for SDD. We consider both symbol-wise and bit-wise decoding, suitable for nonbinary codes and binary codes, respectively, which lead to different achievable rates.

#### Symbol-Wise Soft-Decision Decoding

Consider first a CM scheme that employs a nonbinary code to encode the data and symbol-wise SDD (SW-SDD). Using mismatched decoding, an achievable rate is given by

$$R_{\text{SW-SDD}} = \sup_{s\geq 0} \mathbb{E}_{XY}\left[\log_2\left(\frac{q_{Y|X}(Y|X)^s}{\frac{1}{M}\sum_{x\in\mathcal{X}} q_{Y|X}(Y|x)^s}\right)\right], \tag{7.6}$$

where sup stands for supremum (which can be interpreted as a maximization), $s$ is an optimization parameter, and the expectation is taken with respect to $P_X(x)p_{Y|X}(y|x)$, where $P_X(x) = 1/M$ assuming equiprobable symbols. To obtain the results in Sect. 7.2.3 below, we will assume that the auxiliary channel $q_{Y|X}(Y|X)$ is an AWGN channel, which is a good approximation of the fiber-optic channel under certain conditions. Note that for transmission over the AWGN channel, (7.6) boils down to (7.3), i.e., for transmission over the AWGN channel, the MI is indeed an achievable rate using nonbinary coding and SW-SDD.

#### Bit-Wise Soft-Decision Decoding

In fiber-optic communications, CM is typically implemented according to the bit-interleaved coded modulation (BICM) paradigm [7.96], where a binary code is used in combination with a higher-order constellation, and the code bits are interleaved prior to the mapping to constellation symbols (BICM and other CM schemes are discussed in Sect. 7.6.3). Accordingly, bit-wise decoding is performed at the receiver. While the rate (7.6) is achievable by a symbol-wise soft-decision decoder, it is not necessarily achievable by a bit-wise decoder. In the following, we give an achievable rate for bit-wise decoding.

The BICM decoder treats the different bits of a given symbol as independent. Note that the bits are independent only in the case of perfect interleaving. However, the BICM decoder treats bits as though they were truly independent, which simplifies the decoding. We denote by $m = \log_2 M$ the number of bits

used to label each constellation symbol, and by $\mathsf{L}(x) = (b^{(1)}(x), \ldots, b^{(m)}(x))$ the $m$-bit labeling associated with symbol $x \in \mathcal{X}$. An example of 8-PAM with the binary reflected Gray code labeling is shown in Fig. 7.3. In the figure, $\mathcal{X} = \{-7, -5, -3, -1, +1, +3, +5, +7\}$ and $\mathsf{L}(+1) = (1, 1, 0)$. To simplify notation, we will sometimes write $\mathsf{L}(x) = (b_1, \ldots, b_m)$, hence omitting the argument $x$ on the $b^{(i)}$'s. Also, we denote by $\mathcal{X}_b^{(i)}$, $i = 1, \ldots, m$, the subset of constellation points such that $b^{(i)} = b$, i.e., $\mathcal{X}_b^{(i)} \triangleq \{x \in \mathcal{X} : b^{(i)}(x) = b\}$. For the example in Fig. 7.3, $\mathcal{X}_0^{(2)} = \{-7, -5, +5, +7\}$ and $\mathcal{X}_1^{(2)} = \{-3, -1, +1, +3\}$.

Similar to symbol-wise decoding, we can compute an achievable rate using the mismatched decoding framework. In particular, an achievable rate for BW-SDD is given by the generalized mutual information (GMI),

$$R_{\text{BW-SDD}} = \text{GMI}_{\text{BW-SDD}} \triangleq \sup_{s>0} \mathbb{E}\left[\log_2\left(\frac{q(X,Y)^s}{\frac{1}{M}\sum_{x\in\mathcal{X}} q(x,Y)^s}\right)\right]. \tag{7.7}$$

Assuming that bit levels are independent (i.e., assuming that the random variables $B^{(i)}$, $i = 1, \ldots, m$, are independent), the GMI of the BICM mismatched decoder is equal to the sum of the GMI of the independent binary-input parallel channels [7.97, Th. 2],

$$R_{\text{BW-SDD}} = \sup_{s>0} \sum_{i=1}^{m} \mathbb{E}_{B^{(i)}Y}\left[\log_2\left(\frac{q_i(b^{(i)},Y)^s}{\frac{1}{2}\sum_{b'\in\{0,1\}} q_i(b',Y)^s}\right)\right], \tag{7.8}$$

where the expectation is over $P_{B^{(i)}}(b^{(i)})p_{Y|B^{(i)}}(y|b^{(i)})$, with $P_{B^{(i)}}(b^{(i)}) = 1/2$, and where the $i$-th bit decoding metric $q_i(b, y)$, $i = 1, \ldots, m$, is given by

$$q_i(b,y) \triangleq q_i(b^{(i)}(x) = b, y) = \sum_{x'\in\mathcal{X}_b^{(i)}} q_{Y|X}(y|x'), \tag{7.9}$$

with the auxiliary channel $q_{Y|X}(Y|X)$ being the AWGN channel.

### 7.2.2 Achievable Rates for Hard-Decision Decoding

HDD consists of two steps. First, hard decisions on the received symbols are performed, and then the decoder employs the Hamming distance metric to decode. By doing so, the discrete-input, continuous-output channel with discrete input $x \in \mathcal{X}$ and continuous output $y$ is turned into an $M$-ary input, $M$-ary output discrete memoryless channel with input $x \in \mathcal{X}$ and output $\bar{y} \in \mathcal{X}$ (the hard-detected symbols) with transition probabilities $P_{\bar{Y}|X}(\bar{Y} = \bar{y}|X = x)$. We denote by $\boldsymbol{x} = (x_1, \ldots, x_n)$, $x_i \in \mathcal{X}$ the codeword of modulated symbols and by $\bar{\boldsymbol{y}} = (\bar{y}_1, \ldots \bar{y}_n)$, $\bar{y}_i \in \mathcal{X}$, the sequence at the output of the hard detector.

#### Symbol-Wise Hard-Decision Decoding

A possible strategy for computing an achievable rate is to resort to the mismatched decoding framework [7.87, 88]. With optimal Hamming-metric decoding, the decoding rule is

$$\hat{\boldsymbol{x}}_{\text{SW-HDD}} = \arg\min_{\boldsymbol{x}\in C} d_{\text{H}}(\boldsymbol{x}, \bar{\boldsymbol{y}}), \tag{7.10}$$

where $d_{\text{H}}(\boldsymbol{x}, \bar{\boldsymbol{y}})$ is the Hamming distance between vectors $\boldsymbol{x}$ and $\bar{\boldsymbol{y}}$, i.e., the number of positions in which the two vectors differ (see Definition 7.4 in Sect. 7.3 for a formal definition).

For symbol-wise decoding, assuming that the channel is memoryless, employing HDD based on the Hamming decoding metric is equivalent to maximizing the codeword mismatched decoding metric [7.90]

$$q(\boldsymbol{x}, \bar{\boldsymbol{y}}) = \prod_{i=1}^{n} q(x_i, \bar{y}_i) \tag{7.11}$$

with

$$q(x, \bar{y}) = \begin{cases} 1-\xi & \text{if } \bar{y} = x \\ \dfrac{\xi}{M-1} & \text{otherwise} \end{cases}, \tag{7.12}$$

where $\xi$ is an arbitrary value in $(0, (M-1)/M)$. In fact (see [7.90] for details),

$$\begin{aligned} \hat{\boldsymbol{x}}_{\text{SW-HDD}} &= \arg\max_{\boldsymbol{x}\in C} \prod_{i=1}^{n} q(x_i, \bar{y}_i) \\ &= \arg\max_{\boldsymbol{x}\in C} \left(\frac{\xi}{(1-\xi)(M-1)}\right)^{d_{\text{H}}(\boldsymbol{x},\bar{\boldsymbol{y}})} \\ &\overset{\text{(a)}}{=} \arg\min_{\boldsymbol{x}\in C} d_{\text{H}}(\boldsymbol{x}, \bar{\boldsymbol{y}}), \end{aligned} \tag{7.13}$$

where (a) holds if and only if $0 < \xi < (M-1)/M$.

The metric in (7.12) corresponds to the optimal (i.e., ML) metric for an $M$-ary symmetric channel with symbols $x \in \mathcal{X}$ at its input and symbols $\bar{y} \in \mathcal{X}$ at its output and error probability $\xi$ (i.e., the probability that symbol $x \in \mathcal{X}$ is received erroneously is $\Pr(\bar{y} \neq x|x) = \xi$). Implicitly, a Hamming distance metric decoder treats the channel $P_{\bar{Y}|X}$ as a symmetric channel, ignoring the

information provided by the actual channel transition probabilities.

An achievable rate for symbol-wise HDD is given by the GMI [7.87, 98]

$$R_{\text{SW-HDD}} = \text{GMI}_{\text{SW-HDD}} \triangleq \sup_{s>0} \mathbb{E}_{X\bar{Y}} \left[ \log_2 \left( \frac{q(X,\bar{Y})^s}{\frac{1}{M}\sum_{x\in\mathcal{X}} q(x,\bar{Y})^s} \right) \right], \tag{7.14}$$

where the expectation is over $P_{X\bar{Y}}$, with $P_X(x) = 1/M$, and $q(x,\bar{y})$ is the (symbol) mismatched decoding metric. Using (7.12) as the mismatched metric in (7.14), after some simple derivations, the achievable rate for SW-HDD is obtained as [7.90]

$$R_{\text{SW-HDD}} = \log_2 M - h_{\text{b}}(\delta) - \delta \log_2(M-1), \tag{7.15}$$

where

$$h_{\text{b}}(\delta) = -\delta \log_2 \delta - (1-\delta)\log_2(1-\delta) \tag{7.16}$$

is the binary entropy function evaluated in $\delta$, with $\delta$ being the pre-FEC channel symbol error probability, i.e., $\delta = \Pr(\bar{Y} \neq X)$.

### Bit-Wise Hard-Decision Decoding

As before, let $\mathsf{L}(x) = (b^{(1)}, \ldots, b^{(m)})$ be the $m$-bit labeling associated with symbol $x \in \mathcal{X}$. Also, denote by $\mathsf{L}(\bar{y})$ the binary labeling associated with the hard-detected symbol $\bar{y}$. Assuming BICM, we can model the $m$-bit level channels as $m$ parallel independent BSCs. Let $\epsilon_i$ be the bit error probability of the $i$-th channel, i.e., $\epsilon_i = \Pr(\hat{B}^{(i)} \neq B^{(i)})$. Analogous to SW-HDD, an achievable rate for BW-HDD is given by the GMI (7.14). For bit-wise decoding employing the Hamming distance metric decoding, the mismatched metric is

$$q(x,\bar{y}) = \xi^{d_{\text{H}}(\mathsf{L}(x),\mathsf{L}(\bar{y}))}, \tag{7.17}$$

where $\xi$ is an arbitrary value in $(0,1)$, and $d_{\text{H}}(\mathsf{L}(x),\mathsf{L}(\bar{y}))$ is the Hamming distance between the binary vectors $\mathsf{L}(x)$ and $\mathsf{L}(\bar{y})$.

Using (7.17) in (7.14), after some simple derivations the achievable rate for BW-HDD is obtained as [7.90]

$$R_{\text{BW-HDD}} = m[1 - h_{\text{b}}(\bar{\epsilon})], \tag{7.18}$$

where $h_{\text{b}}(\bar{\epsilon})$ is the binary entropy function (7.16) evaluated in $\bar{\epsilon}$ and

$$\bar{\epsilon} = \frac{1}{m}\sum_{i=1}^{m} \epsilon_i . \tag{7.19}$$

We note that, rather than taking hard decisions over the received symbols, one may first compute bit-wise log-likelihood ratios (LLRs) and then take hard decisions at the bit level. This leads to an achievable rate different from the one in (7.18). However, the two achievable rates are very similar.

## 7.2.3 Comparison of Achievable Rates for Bit-Wise and Symbol-Wise Decoding

In Fig. 7.4, we plot the achievable rates (7.6), (7.8), (7.15), and (7.18) for transmission over the AWGN channel using 16-QAM and 64-QAM modulation as a function of the SNR. The achievable rate curves have to be interpreted as follows: for each SNR, they determine the maximum spectral efficiency that *can be achieved* (with vanishing probability of error for infinite code length and ideal pulse shaping) by a given decoding strategy (SW-SDD, BW-SDD, SW-HDD, or BW-HDD). Alternatively, for a given spectral efficiency, a given decoding strategy can yield vanishing error probability if operating at an SNR of at least the value provided by the curve. It is observed that for SDD, the achievable rates for bit-wise and symbol-wise decoding are similar. More precisely, the achievable rate for bit-wise decoding is slightly smaller, especially for low spectral efficiencies. However, bit-wise decoding entails virtually no loss in the region of interest (i.e., for spectral efficiencies where the curve starts bending toward its maximum

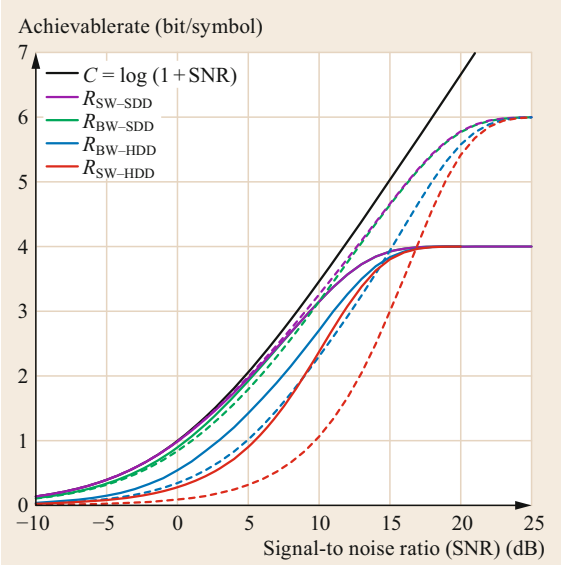


**Fig. 7.4** Achievable rates for transmission over the AWGN channel with 16-QAM modulation (*solid curves*) and 64-QAM modulation (*dashed curves*) for SDD and HDD with both bit-wise and symbol-wise decoding

**Table 7.1** Fiber and simulation parameters for the SSFM

| | |
|---|---|
| Attenuation ($\alpha$) | 0.2 dB/km |
| Dispersion ($D$) | 17 ps/nm/km |
| Nonlinear coefficient ($\gamma$) | 1.3 $(\text{Wkm})^{-1}$ |
| Wavelength $\lambda$ | 1550 nm |
| Symbol rate | 32 Gbaud |
| Span length | 80 km |
| EDFA noise figure | 4.5 dB |
| SSFM step size | 0.1 km |

value $m$) with respect to symbol-wise decoding. On the other hand, as shown in [7.90], for HDD the achievable rates for bit-wise decoding (using BICM) are higher than those for symbol-wise decoding. As can be seen in the figure, the penalty incurred by using symbol-wise decoding yields a significantly lower achievable rate for low-to-medium SNRs. Equivalently, to achieve a given rate, the symbol-wise decoder requires a significantly higher SNR. According to these results, for HDD, binary FEC codes and bit-wise decoding are preferable to nonbinary FEC codes and symbol-wise decoding [7.90]. Furthermore, the decoding complexity of binary codes is lower than that of nonbinary codes. In the figure, we also plot the channel capacity. It is very important to realize that for high spectral efficiencies, there is a gap between the achievable rate $R_{\text{SW-SDD}}$, which coincides with the MI for the AWGN channel as explained in Sect. 7.2.1, *Symbol-Wise Soft-Decision Decoding*, and the capacity. This gap, referred to as the *shaping loss*, occurs because QAM constellations with equiprobable symbols are not capacity-achieving. The shaping loss induced by standard constellations is discussed in Sect. 7.6.5, together with methods to reduce it.

In Fig. 7.5, we plot the achievable rates as a function of the transmission distance for a polarization-multiplexed (PM) single-channel transmission system with electronic dispersion compensation (EDC) to compensate for the chromatic dispersion for 64-QAM and 256-QAM. Here, we neglect the polarization mode dispersion and the state of polarization (SOP) drift. Therefore, the two polarizations are independent of each other. The parameters of the optical fiber are summarized in Table 7.1. The results are for optimal launch power. The span loss is compensated for using erbium-doped fiber amplifiers (EDFAs), and a root-raised cosine pulse with a roll-off factor of 0.25 is used. The split-step Fourier method is used to simulate the fiber-optic channel and estimate $p_{Y|X}(y|x)$ for SDD and the transition probabilities $P_{\bar{Y}|X}(\bar{y}|x)$, the error probability $\bar{\epsilon}$ in (7.19), and the pre-FEC channel symbol error probability $\delta$ for HDD, and correspondingly calculate the achievable rates.

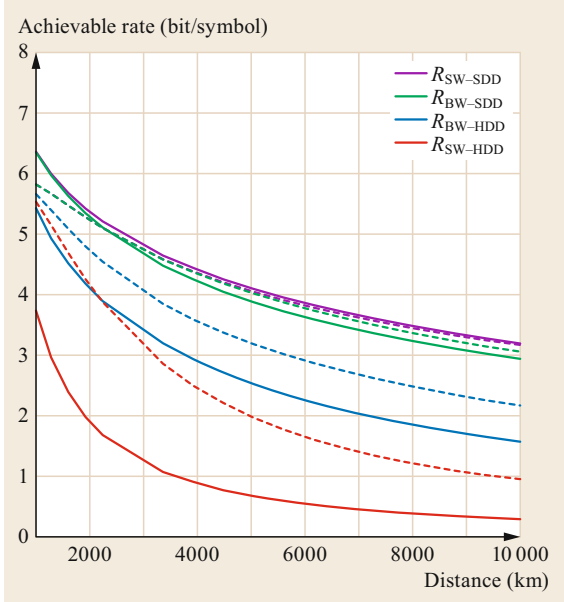


**Fig. 7.5** Achievable rates for a PM single-channel transmission system with EDC as a function of the transmission distance for SDD and HDD with both bit-wise and symbol-wise decoding and optimal launch power. *Dashed curves* correspond to 64-QAM modulation and *solid curves* to 256-QAM modulation

In Fig. 7.6, we plot the achievable rates for a WDM transmission system with 81 channels and the parameters of Table 7.1. The optical channel for such a system is well approximated by the AWGN channel, a model known as the GN model in the literature [7.99, 100]. The achievable rates for the middle channel with 64-QAM and 256-QAM are shown in Fig. 7.6.

The results in Figs. 7.5 and 7.6 for the PM single-channel and WDM systems lead to similar conclusions as those of the results for the AWGN channel. We observe in Fig. 7.6 that for SDD, the bit-wise and symbol-wise decoding achievable rate curves for 16-QAM are indistinguishable, while for 64-QAM, bit-wise decoding incurs only a small penalty compared with symbol-wise decoding for 64-QAM for low spectral efficiencies. For 256-QAM, the achievable transmission reach for symbol-wise decoding is higher than that for bit-wise decoding. However, for the spectral efficiencies of interest, bit-wise SDD and symbol-wise SDD yield almost identical transmission reach. Similar results are observed in Fig. 7.5. In contrast, for HDD, bit-wise decoding yields larger achievable rates than symbol-wise decoding, which leads to a significant optical reach enhancement.

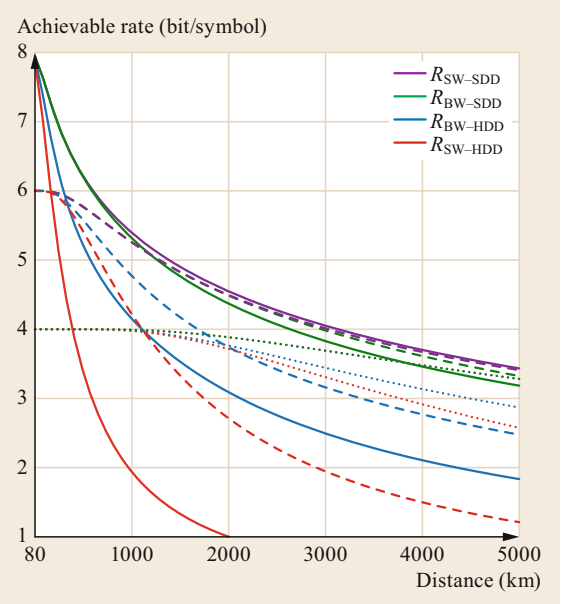


**Fig. 7.6** Achievable rates for a WDM transmission system using the GN model as a function of the transmission distance for SDD and HDD with both bit-wise and symbol-wise decoding and optimal launch power. *Dotted curves* correspond to 16-QAM modulation, *dashed curves* to 64-QAM modulation, and *solid curves* to 256-QAM modulation

### 7.2.4 Achievable Rates and Actual Code Performance

It is important to stress that the achievable rates provided by information theory, as those discussed above, determine a rate that can be achieved, i.e., *there exists* a code that, with infinite block length, achieves that particular transmission rate. However, achievable rates tell little about the actual performance of particular FEC codes in general. A given code, decoded using a certain (suboptimal) decoding algorithm, does not necessarily achieve them. For example, regular LDPC codes (with growing degree) are capacity-achieving for binary transmission over the AWGN channel under maximum a posteriori (MAP) decoding, but not under practical iterative decoding. Similarly, staircase codes decoded with (suboptimal) iterative bounded distance decoding (Sect. 7.5.6) are not capacity-achieving. Therefore, in general, practical codes (in the sense of codes decoded using suboptimal decoding algorithms) do not achieve the achievable rates discussed above, and a performance gap will be observed. Furthermore, the gap with respect to the achievable rate may depend on the code rate (see (7.22) in Sect. 7.3 for the formal definition of code rate). For example, for staircase codes, the gap relative to the achievable rate increases for low rates. Nonetheless, achievable rates provide a very useful benchmark for comparing the performance of practical codes and are still informative.

Note that above, by capacity-achieving we mean a code that achieves the Shannon limit. For instance, for binary transmission using BPSK and SDD, the ultimate limit, also called the Shannon limit, is the corresponding MI. A code that achieves the MI for binary transmission over the AWGN channel does not achieve the capacity of the AWGN channel (since this requires Gaussian input). However, with some abuse of language, in coding theory jargon, such a code is called capacity-achieving.

As we will discuss in Sects. 7.4 and 7.5, for LDPC codes, generalized product codes, and modern codes in general, the performance limits (in the limit of infinite block length) for a certain code ensemble are given by density evolution. The so-called decoding threshold obtained from density evolution can be seen as the *effective capacity* (i.e., the effective performance limit) for a given code ensemble. One should then compare the threshold of a given code ensemble with the corresponding achievable rate to see how much is lost because of the choice of a particular code and decoding algorithm. Designing codes that approach the fundamental limits given by information theory is the ultimate goal of coding theory.

## 7.3 Basics of Forward Error Correction

Forward error correction (FEC), also known as *error-correcting coding*, *channel coding*, or simply *coding*, is an indispensable technique for achieving reliable communication and storage that has become ubiquitous in any communication and storage system. The role of FEC is to protect the data to be transmitted (or stored) from the channel impairments. It does so by following a basic principle: adding redundancy.

In this section, we introduce the basic concepts of FEC. Our focus is on binary codes, since their lower decoding complexity with respect to nonbinary codes makes them by far the most popular and widely used (except, perhaps, Reed–Solomon codes). However, the concepts introduced in this section can be extended to nonbinary codes in a straightforward manner.

We first give a layman's definition of an error-correcting code.

*Definition 7.1 Error-correcting code*
An error-correcting code is a device that adds redundancy to a block of data bits and uses this redundancy to correct potential transmission errors.

In this chapter, we focus on the most important and ubiquitous class of codes, *block codes*, and give the most basic definition of a block code.

*Definition 7.2*
A binary block code $C$ of code length $n$ and dimension $k$, denoted by $C(n,k)$, is a collection of $2^k$ binary vectors of length $n$ bits, called $n$-vectors, i.e.,

$$C(n,k) \triangleq \left\{\boldsymbol{c}^{[1]}, \boldsymbol{c}^{[2]}, \ldots, \boldsymbol{c}^{[2^k]}\right\}, \quad \boldsymbol{c}^{[m]} \in \{0,1\}^n \,. \tag{7.20}$$

The $n$-vectors $\boldsymbol{c}^{[m]}$ are called *codewords*, and $k$ is also referred to as the *information block length* of the code. The set of binary numbers is also often denoted as the *Galois field* of two elements and notated as $\mathrm{GF}(2) \triangleq \{0,1\}$. Note that the notion of Galois field also defines the operations of addition and multiplication of its elements. Loosely speaking, $\mathrm{GF}(q)$ is a finite set of $q$ objects over which addition and multiplication are defined with similar properties as addition and multiplication of real numbers. For details and rigorous definitions, see [7.101, Chap. 2].

*Example 7.1*
Consider the 3-repetition code with parameters $n = 3$ and $k = 1$, consisting of $2^k = 2$ codewords of length 3 bits,

$$C_{\mathrm{rep}}(n=3, k=1) = \{(0,0,0), (1,1,1)\} \,. \tag{7.21}$$

One may consider an *encoder* that assigns to the information bit $u = 0$ the codeword $(0,0,0)$ and to the information bit $u = 1$ the codeword $(1,1,1)$. The assignment of codewords to information words (in this case to a single bit) is usually referred to as *encoding*.

An important parameter is the *code rate* $R$, or simply rate, defined as

$$R \triangleq \frac{k}{n} < 1 \,. \tag{7.22}$$

The rate is a measure of the redundancy of the code. Since $R < 1$, then $n > k$, i.e., the information words are encoded into longer sequences, thereby introducing redundancy. In contrast to the field of coding theory, in the optical communications literature the code rate is rarely used. Therein, the *code overhead*, or simply overhead, defined as

$$\mathrm{OH} \triangleq \frac{1}{R} - 1 = \frac{n-k}{k} \,, \tag{7.23}$$

is frequently used instead. The overhead denotes the relative portion of redundant bits that are added to the information bits. Note that the rate of the code is expressed in information bits per transmitted bit and is thus a measure of the *bandwidth efficiency* of the code. For example, the rate of the 3-repetition code of Example 7.1 is $R = 1/3$, which corresponds to overhead of $\mathrm{OH} = 200\%$. For fiber-optic communications, practical codes with much lower overhead are desired.

We now define one of the most important parameters of error-correcting codes, the minimum Hamming distance. We will first need the definition of Hamming weight and Hamming distance.

*Definition 7.3 Hamming weight*
For a binary vector $\boldsymbol{c} = (c_1, \ldots, c_n)$ of length $n$, the Hamming weight, denoted by $w_{\mathrm{H}}(\boldsymbol{c})$, is the number of entries in which $c_i = 1$, i.e.,

$$w_{\mathrm{H}}(\boldsymbol{c}) \triangleq |\{i \,:\, c_i = 1\}| \,. \tag{7.24}$$

*Definition 7.4 Hamming distance*
For any two binary vectors $\boldsymbol{c}$ and $\tilde{\boldsymbol{c}}$ of length $n$, the Hamming distance, $d_{\mathrm{H}}(\boldsymbol{c}, \tilde{\boldsymbol{c}})$, is the number of entries in which $\boldsymbol{c}$ and $\tilde{\boldsymbol{c}}$ differ, i.e.,

$$d_{\mathrm{H}}(\boldsymbol{c}, \tilde{\boldsymbol{c}}) \triangleq |\{i \,:\, c_i \neq \tilde{c}_i\}| \,. \tag{7.25}$$

*Definition 7.5 Minimum Hamming distance*
The minimum Hamming distance of a code $C$, denoted by $d_{\min}(C)$, is the smallest Hamming distance between any two distinct codewords of the code and is defined as

$$d_{\min}(C) \triangleq \min_{\substack{\boldsymbol{c}, \tilde{\boldsymbol{c}} \in C \\ \boldsymbol{c} \neq \tilde{\boldsymbol{c}}}} d_{\mathrm{H}}(\boldsymbol{c}, \tilde{\boldsymbol{c}}) \,. \tag{7.26}$$

As we will later see, the minimum Hamming distance of a code is related to its *error-correcting capability*. Since the notion of minimum Hamming distance is so important in coding, an $(n,k)$ block code with minimum Hamming distance $d_{\min}$ is sometimes denoted as an $(n,k,d_{\min})$ block code.

The following simple but powerful upper bound on the minimum Hamming distance holds.

*Theorem 7.1 Singleton bound*
The minimum Hamming distance of an $(n, k, d_{\min})$ code $C$ over GF($q$) is upper bounded by $d_{\min} \leq n - k + 1$.

Codes whose minimum Hamming distance attains the Singleton bound achieve the best possible error-correcting capability for a given overhead and are referred to as maximum distance separable (MDS) codes. A very important class of MDS codes are Reed–Solomon codes, which are briefly discussed in Sect. 7.5.1. In contrast, there are no binary MDS codes, except trivial codes—the repetition code and the parity-check code.

### 7.3.1 Linear Block Codes

Linear block codes are a very important class of codes. Indeed, almost all practical codes in use today are linear codes. The reason is that linear codes are sufficient to achieve capacity [7.23, 102], and linearity allows for efficient encoding and decoding circuits. All codes considered in this chapter are linear codes.

*Definition 7.6 Linear block code*
A binary block code $C(n, k)$ is a linear block code if and only if its codewords $\boldsymbol{c}^{[1]}, \ldots, \boldsymbol{c}^{[2^k]}$ form a $k$-dimensional subspace of the $n$-dimensional vector space $\{0, 1\}^n$.

For a linear block code, the *all-zero* codeword $(0, 0, \ldots, 0)$ is always a codeword, i.e., $(0, 0, \ldots, 0) \in C$, and the (binary) addition of two codewords is also a codeword, i.e., for $\boldsymbol{c} \in C$ and $\tilde{\boldsymbol{c}} \in C$, then $\boldsymbol{c} + \tilde{\boldsymbol{c}} \in C$, with $+$ denoting the binary (modulo-2) addition. In other words, the set of codewords is closed under component-wise binary addition, where binary addition corresponds to the logical exclusive OR (XOR) operation, and binary multiplication corresponds to the logical AND operation. For any two binary vectors $\boldsymbol{c}$ and $\tilde{\boldsymbol{c}}$,

$$d_{\mathrm{H}}(\boldsymbol{c}, \tilde{\boldsymbol{c}}) = w_{\mathrm{H}}(\boldsymbol{c} + \tilde{\boldsymbol{c}}) \,, \tag{7.27}$$

and the minimum Hamming distance of a linear block code is equal to the minimum codeword weight, i.e.,

$$d_{\min}(C) = \min_{\substack{\boldsymbol{c} \in C \\ \boldsymbol{c} \neq \boldsymbol{0}}} w_{\mathrm{H}}(\boldsymbol{c}) \,. \tag{7.28}$$

Linear codes have appealing properties for analysis and implementation. It is easy to see that the 3-repetition code of Example 7.1 is a linear code. For this code, the three-dimensional subspace representation is shown in Fig. 7.7.

Since a linear block code is a $k$-dimensional subspace, we can find $k$ linearly independent basis vectors $\boldsymbol{g}_1, \ldots, \boldsymbol{g}_k$ in $\{0, 1\}^n$ that span $C$, i.e., every codeword in $C$ is a linear combination of these $k$ basis vectors. Then, the codeword $\boldsymbol{c} = (c_1, \ldots, c_n)$ for the message $\boldsymbol{u} = (u_1, \ldots, u_k)$ is obtained as the product of $\boldsymbol{u}$ and $\mathbf{G}$ as

$$\boldsymbol{c} = \boldsymbol{u}\mathbf{G} \,, \tag{7.29}$$

where $\mathbf{G}$ is a $k \times n$ binary matrix with rows $\boldsymbol{g}_1, \ldots, \boldsymbol{g}_k$. Since $\mathbf{G}$ spans (i.e., generates) the code $C$, it is usually referred to as the *generator matrix* of the code. Note that the basis vectors $\boldsymbol{g}_1, \ldots, \boldsymbol{g}_k$ are also codewords.

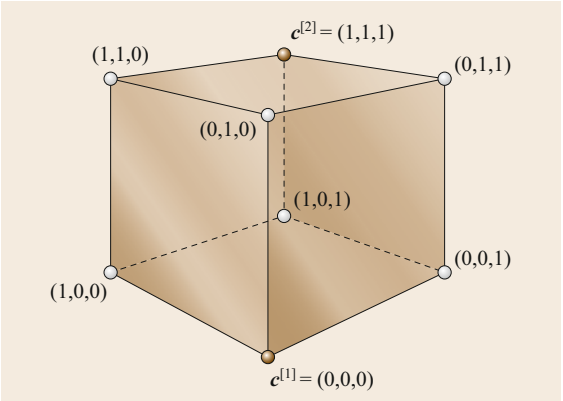


**Fig. 7.7** Subspace representation of the $(3, 1)$-repetition code

The code, as well as the encoder (i.e., the mapping of information words to codewords), is completely specified by the generator matrix $\mathbf{G}$. Alternatively, a linear block code can be defined through its $(n-k) \times n$ parity-check matrix $\mathbf{H}$. Since a binary $(n, k)$ linear block code $C$ is a $k$-dimensional subspace of $\{0, 1\}^n$, its *null* (or *dual*) space, i.e., the set of all binary words of length $n$ that are orthogonal to every element in $C$, is an $(n-k)$-dimensional subspace of $\{0, 1\}^n$. Two vectors $\boldsymbol{a}$ and $\boldsymbol{b}$ are orthogonal if their inner product is zero, i.e., $\langle \boldsymbol{a}, \boldsymbol{b} \rangle = \boldsymbol{a}\boldsymbol{b}^{\top} = 0$. We denote the null space of $C$ by $C_{\perp}$ and write

$$C_{\perp} = \{\tilde{\boldsymbol{c}} : \langle \tilde{\boldsymbol{c}}, \boldsymbol{c} \rangle = 0 \quad \text{for all } \boldsymbol{c} \in C\} \,. \tag{7.30}$$

Note that $C_{\perp}$ is a binary $(n, n-k)$ linear block code, usually referred to as the *dual code* of $C$. Let $\mathbf{H}$ be a generator matrix of the code $C_{\perp}$, of dimensions $(n-k) \times n$. The rows of $\mathbf{H}$, $\boldsymbol{h}_1, \ldots, \boldsymbol{h}_{n-k}$, are linearly independent codewords of $C_{\perp}$, i.e., $\boldsymbol{h}_1, \ldots, \boldsymbol{h}_{n-k} \in C_{\perp}$, and every codeword in $C_{\perp}$ is a linear combination of these $n-k$ codewords. Since every codeword $\boldsymbol{c} \in C$ is orthogonal to every codeword $\tilde{\boldsymbol{c}} \in C_{\perp}$, it follows that

$$\boldsymbol{c}\mathbf{H}^{\top} = \mathbf{0} \,. \tag{7.31}$$

Note that (7.31) is satisfied *if and only if* $\boldsymbol{c} \in C$. Therefore, we can define a code $C$ through the generator matrix of its dual code $C_{\perp}$, $\mathbf{H}$.

*Definition 7.7*

A binary vector $\boldsymbol{c}$ is a codeword of $C$ if and only if $\boldsymbol{c}\mathbf{H}^\top = \mathbf{0}$, i.e., the code $C$ is defined as the null space of $\mathbf{H}$,

$$C = \left\{\boldsymbol{c} : \boldsymbol{c}\mathbf{H}^\top = \mathbf{0}\right\}. \tag{7.32}$$

Therefore, a linear block code is uniquely specified by $\mathbf{G}$ or $\mathbf{H}$. Usually, the generator matrix is used for encoding, while the decoding is based on $\mathbf{H}$.

A way to interpret (7.31) is by noting that $\boldsymbol{c}\mathbf{H}^\top = \mathbf{0}$ is a set of $n-k$ linearly independent equations

$$\begin{aligned} h_{1,1}c_1 &+ h_{1,2}c_2 + \ldots + h_{1,n}c_n = 0 \\ h_{2,1}c_1 &+ h_{2,2}c_2 + \ldots + h_{2,n}c_n = 0 \\ &\vdots \qquad\quad \vdots \qquad\quad \ddots \qquad\quad \vdots \qquad\quad \vdots \\ h_{(n-k),1}c_1 &+ h_{(n-k),2}c_2 + \ldots + h_{(n-k),n}c_n = 0 \end{aligned} \tag{7.33}$$

where $h_{i,j}$ is the entry of the matrix $\mathbf{H}$ at row $i$ and column $j$. These equations are known as *parity-check equations*, and are the equations that each codeword must satisfy. As a result, the matrix $\mathbf{H}$ is commonly referred to as the *parity-check matrix* of the code $C$.

A practically important class of (linear) codes is the class of *systematic* codes.

*Definition 7.8 Systematic code*

A *systematic code* is a code with an encoder that generates codewords $\boldsymbol{c}$ containing the information word $\boldsymbol{u}$ as verbatim copy in $\boldsymbol{c}$.

The generator matrix of a *systematic linear code* (assuming that the $k$ information bits appear in the first $k$ positions of the codeword) can always be written as $\mathbf{G} = (\mathbf{I}_k \ \mathbf{P})$, where $\mathbf{I}_k$ is a $k \times k$ identity matrix. The codewords of a systematic code can then be written as

$$\begin{aligned} \boldsymbol{c} &= (c_1 \ldots, c_n) = (u_1, , \ldots, u_k, p_1, \ldots p_{n-k}) \\ &= (\boldsymbol{u} \ \boldsymbol{p}), \end{aligned} \tag{7.34}$$

i.e., the first $k$ bits of the codeword are a replica of the information bits. The code bits $p_1, p_2, \ldots, p_{n-k}$ are referred to as the *parity bits* of the code.

*Example 7.2*

Consider the $(7, 4)$ binary block code defined by the $4 \times 7$ generator matrix

$$\mathbf{G} = \begin{pmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 0 & 1 \end{pmatrix}. \tag{7.35}$$

**Table 7.2** Codewords of the $(7, 4)$ Hamming code

| $u_1$ | $u_2$ | $u_3$ | $u_4$ | $p_1$ | $p_2$ | $p_3$ |
|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 1 | 1 | 0 | 1 |
| 0 | 0 | 1 | 0 | 1 | 1 | 1 |
| 0 | 0 | 1 | 1 | 0 | 1 | 0 |
| 0 | 1 | 0 | 0 | 0 | 1 | 1 |
| 0 | 1 | 0 | 1 | 1 | 1 | 0 |
| 0 | 1 | 1 | 0 | 1 | 0 | 0 |
| 0 | 1 | 1 | 1 | 0 | 0 | 1 |
| 1 | 0 | 0 | 0 | 1 | 1 | 0 |
| 1 | 0 | 0 | 1 | 0 | 1 | 1 |
| 1 | 0 | 1 | 0 | 0 | 0 | 1 |
| 1 | 0 | 1 | 1 | 1 | 0 | 0 |
| 1 | 1 | 0 | 0 | 1 | 0 | 1 |
| 1 | 1 | 0 | 1 | 0 | 0 | 0 |
| 1 | 1 | 1 | 0 | 0 | 1 | 0 |
| 1 | 1 | 1 | 1 | 1 | 1 | 1 |

Table 7.2 lists all codewords of this code, which is the famous $(7, 4)$ Hamming code. We observe that the first four bits of each codeword correspond to the information bits, i.e., the code is systematic.

### 7.3.2 Error Detection and Error Correction Capability of a Block Code over the Binary Symmetric Channel

Consider an $(n, k, d_{\min})$ block code that is used for transmission over the AWGN channel using BPSK modulation. As discussed in Sect. 7.2, if hard decisions on the received vector are taken, the discrete channel between the encoder and the decoder can be modeled as a binary symmetric channel (BSC).

For transmission over the BSC, a block code with minimum Hamming distance $d_{\min}$ can *correct* all error patterns with

$$t = \left\lfloor \frac{d_{\min} - 1}{2} \right\rfloor \tag{7.36}$$

or fewer errors, where $\lfloor \tau \rfloor$ denotes the largest integer smaller than or equal to $\tau$. The parameter $t$ is usually referred to as the *error correction capability* of the code, and a code that can correct up to $t$ errors is called a $t$-error-correcting code. Furthermore, a block code with minimum Hamming distance $d_{\min}$ can *detect* all error patterns with

$$d = d_{\min} - 1 \tag{7.37}$$

or fewer errors.

*Example 7.3*

The minimum Hamming distance of the 3-repetition code (Example 7.1) and the $(7,4)$ Hamming code in Example 7.2 is $d_{\min} = 3$; hence both codes can correct up to $t = 1$ error and detect up to $d = 2$ errors. The repetition code has rate $R = 1/3$, while the rate of the Hamming code is $4/7$; hence the latter is spectrally more efficient.

The minimum Hamming distance of a code is a very relevant parameter that determines how many errors the code can correct (or detect) for transmission over the BSC. For other channels, the minimum Hamming distance does not completely determine the code performance, but it dominates the code performance (under ML decoding) for high SNRs. Thus, to achieve low error rates, one should construct codes with good minimum Hamming distance.

### 7.3.3 Optimal Decoding of Block Codes

We have seen that we can characterize a block code in terms of its minimum Hamming distance. We now derive the optimal decoder for block codes for both SDD and HDD. We consider binary transmission using BPSK modulation (i.e., $\mathcal{X} = \{\pm 1\}$) by modulating every code bit $c_i$, $i = 1, \ldots, n$, of the codeword as $x_i = (-1)^{c_i}$, i.e., without loss of generality we assume the mapping $0 \mapsto +1$ and $1 \mapsto -1$. For example, for $\boldsymbol{c} = (0,0,1,1,0)$, we get $\boldsymbol{x} = (+1,+1,-1,-1,+1)$. Instead of BPSK, we can also consider quadrature phase-shift keying (QPSK) by independently modulating the I and Q components of the constellation. We assume transmission over a memoryless AWGN channel with zero mean and noise variance $\sigma^2$. At the output of the channel, we observe a (noisy) vector $\boldsymbol{y} = \boldsymbol{x} + \boldsymbol{n}$, where $\boldsymbol{x} = (x_1, \ldots, x_n)$ is the transmitted (modulated) codeword and $\boldsymbol{n}$ is the noise vector.

Based on the noisy observation $\boldsymbol{y}$, the role of the decoder is to optimally *estimate* the transmitted codeword $\boldsymbol{c}$. Decoding is a probabilistic *inference* problem: the decoder estimates the *most likely* transmitted codeword $\boldsymbol{c}$ (from which it can reverse-lookup the information bit sequence) given the noisy channel output $\boldsymbol{y}$. The most natural decoding criterion is that of minimizing the probability of error. If the decoding criterion is to minimize the probability that the decoder fails to decode to the correct codeword, i.e., minimize the *codeword error probability*, it can be shown that this is equivalent to maximizing the a posteriori probability $P_{C|Y}(\boldsymbol{c}|\boldsymbol{y})$. The *maximum a posteriori* (MAP) decoding rule is therefore

$$\hat{\boldsymbol{c}} = \arg\max_{\boldsymbol{c}\in C} P_{C|Y}(\boldsymbol{c}|\boldsymbol{y}) = \arg\max_{\boldsymbol{c}\in C} \frac{p_{Y|C}(\boldsymbol{y}|\boldsymbol{c}) P_C(\boldsymbol{c})}{p_Y(\boldsymbol{y})} , \tag{7.38}$$

where in the last equality we used Bayes' law. Typically, we can assume that all codewords are equiprobable, i.e., $P_C(\boldsymbol{c}_i) = 1/|C|$, for all $i$. In this case, and since $p_Y(\boldsymbol{y})$ is independent of $\boldsymbol{c}$, (7.38) can be rewritten as

$$\hat{\boldsymbol{c}} = \arg\max_{\boldsymbol{c}\in C} p_{Y|C}(\boldsymbol{y}|\boldsymbol{c}) , \tag{7.39}$$

which is referred to as the ML decoding rule. Note that the situation changes if probabilistic amplitude shaping (PAS), as described in Sect. 7.6.5, is used. In this case, not all codewords are equiprobable. For a detailed exposure, see [7.91].

In the next two sections, we expand the ML decoding rule (7.39) for SDD and HDD.

#### Soft-Decision Decoding

Starting from the ML decoding rule (7.39) and assuming a memoryless channel, i.e., $p_{Y|C}(\boldsymbol{y}|\boldsymbol{c}) = \prod_{i=1}^{n} p_{Y|C}(y_i|c_i)$, we can proceed as

$$\begin{aligned}
\hat{\boldsymbol{c}} &= \arg\max_{\boldsymbol{c}\in C} \prod_{i=1}^{n} p_{Y|C}(y_i|c_i) \\
&\overset{(a)}{=} \arg\max_{\boldsymbol{c}\in C} \ln \prod_{i=1}^{n} p_{Y|C}(y_i|c_i) \\
&= \arg\max_{\boldsymbol{c}\in C} \sum_{i=1}^{n} \ln p_{Y|C}(y_i|c_i) ,
\end{aligned} \tag{7.40}$$

where (a) follows because $\ln(\cdot)$ is a monotonically increasing function. To perform ML decoding, we need to have access to the channel law $p_{Y|C}(y_i|c_i)$, which is the conditional probability (density) that we have received $y_i$ given that $c_i$ was transmitted. If known, the true channel law can be used in (7.40). However, we usually resort to models. In the case of coherent optical communications, the channel is reasonably well modeled by an AWGN channel, as predicted by the GN models and their extensions [7.99, 103]. Another example is the case of amplitude-modulated solitons, where the channel law can be modeled as a noncentral chi-squared distribution with four degrees of freedom [7.104]. Furthermore, when higher-order modulation formats with bit-wise decoding (often called BICM) are used (see also Sect. 7.6.3), the equivalent noise that the FEC decoder observes does not necessarily follow a Gaussian distribution [7.105] but can be closely approximated by one.

We will now focus on the case of AWGN, where the channel law is given in (7.1). Continuing the derivation, we have with $x_i = (-1)^{c_i}$,

$$\begin{aligned}
\hat{\boldsymbol{c}} &= \arg\max_{\boldsymbol{c}\in C} \sum_{i=1}^{n} \ln\left( \frac{1}{\sqrt{2\pi}\sigma} \mathrm{e}^{-\frac{(y_i-(-1)^{c_i})^2}{2\sigma^2}} \right) \\
&= \arg\max_{\boldsymbol{c}\in C} \sum_{i=1}^{n} \left[ \ln\left( \frac{1}{\sqrt{2\pi}\sigma} \right) - \frac{[y_i-(-1)^{c_i}]^2}{2\sigma^2} \right]
\end{aligned}$$

$$\stackrel{(b)}{=} \arg\max_{\boldsymbol{c}\in C} \sum_{i=1}^{n} \frac{-[y_i-(-1)^{c_i}]^2}{2\sigma^2}$$
$$\stackrel{(c)}{=} \arg\min_{\boldsymbol{c}\in C} \sum_{i=1}^{n} \frac{[y_i-(-1)^{c_i}]^2}{2\sigma^2}$$
$$\stackrel{(d)}{=} \arg\min_{\boldsymbol{c}\in C} \sum_{i=1}^{n} [y_i-(-1)^{c_i}]^2 \,, \tag{7.41}$$

where in (b) we exploit the fact that addition of a constant does not change the maximization, in (c) we use the fact that maximizing is equivalent to minimizing the negation, and finally in (d) dividing by the constant $2\sigma^2$ does not change the minimization. Equation (7.41) is the famous Euclidean distance metric, as we take the squared Euclidean distance between (modulated) codewords and the received values. Therefore, for SDD and transmission over the AWGN channel, the ML decoder chooses among all possible transmitted codewords the codeword $\boldsymbol{c}$ that minimizes the Euclidean distance $d_{\mathrm{E}}(\boldsymbol{x},\boldsymbol{y}) \triangleq \|\boldsymbol{x}-\boldsymbol{y}\|$ between $\boldsymbol{y}$ and the BPSK-modulated codeword $\boldsymbol{x} = (x_1,\ldots,x_n) = ((-1)^{c_1},\ldots,(-1)^{c_n})$, i.e.,

$$\hat{\boldsymbol{c}} = \arg\min_{\boldsymbol{c}\in C} d_{\mathrm{E}}(\boldsymbol{x},\boldsymbol{y}) \,. \tag{7.42}$$

Note that the Euclidean distance follows solely from the assumption of Gaussian noise.

We can further simplify the expression (7.41) yielding

$$\hat{\boldsymbol{c}} = \arg\min_{\boldsymbol{c}\in C} \sum_{i=1}^{n} \left[y_i^2 + (-1)^{2c_i} - 2y_i(-1)^{c_i}\right]$$
$$= \arg\min_{\boldsymbol{c}\in C} \sum_{i=1}^{n} \left[y_i^2 + 1 - 2y_i(-1)^{c_i}\right]$$
$$\stackrel{(e)}{=} \arg\min_{\boldsymbol{c}\in C} \sum_{i=1}^{n} \left[-2y_i(-1)^{c_i}\right]$$
$$\stackrel{(f)}{=} \arg\max_{\boldsymbol{c}\in C} \sum_{i=1}^{n} y_i(-1)^{c_i} \,, \tag{7.43}$$

where in (e) we used the fact that addition of $y_i^2+1$ does not change the maximization and in (f) we replaced minimization by maximization of the negation.

### Hard-Decision Decoding

We now derive the optimal decoder for HDD. We start again from the ML decoding rule, assuming a memoryless channel,

$$\hat{\boldsymbol{c}} = \arg\max_{\boldsymbol{c}\in C} p_{Y|C}(\boldsymbol{y}|\boldsymbol{c})$$
$$= \arg\max_{\boldsymbol{c}\in C} \prod_{i=1}^{n} p_{Y|C}(y_i|c_i) \,. \tag{7.44}$$

Note that for HDD, the output sequence $\boldsymbol{y} = (y_1,\ldots,y_n)$ is also a binary sequence, and the channel can be modeled as a BSC with crossover probability $p$ (see (7.2)), i.e.,

$$p_{Y|X}(y_i|x_i) = p_{Y|C}(y_i|c_i) = \begin{cases} p & y_i \neq c_i \\ 1-p & y_i = c_i \,. \end{cases} \tag{7.45}$$

Using this in (7.44),

$$\hat{\boldsymbol{c}} = \arg\max_{\boldsymbol{c}\in C} p^{d_{\mathrm{H}}(\boldsymbol{c},\boldsymbol{y})}(1-p)^{n-d_{\mathrm{H}}(\boldsymbol{c},\boldsymbol{y})}$$
$$= \arg\max_{\boldsymbol{c}\in C} \ln\left[p^{d_{\mathrm{H}}(\boldsymbol{c},\boldsymbol{y})}(1-p)^{n-d_{\mathrm{H}}(\boldsymbol{c},\boldsymbol{y})}\right]$$
$$= \arg\max_{\boldsymbol{c}\in C} d_{\mathrm{H}}(\boldsymbol{c},\boldsymbol{y}) \ln p + [n-d_{\mathrm{H}}(\boldsymbol{c},\boldsymbol{y})] \ln(1-p)$$
$$= \arg\max_{\boldsymbol{c}\in C} d_{\mathrm{H}}(\boldsymbol{c},\boldsymbol{y}) \ln\left(\frac{p}{1-p}\right)$$
$$= \arg\min_{\boldsymbol{c}\in C} d_{\mathrm{H}}(\boldsymbol{c},\boldsymbol{y}) \,, \tag{7.46}$$

where in the last equality we assume that $p < (1/2)$.

We see that for HDD, the ML decoder must choose among all possible transmitted codewords the codeword $\boldsymbol{c}$ that minimizes the Hamming distance between $\boldsymbol{x}$ and $\boldsymbol{y}$.

We illustrate the ML decoder using a simple example.

*Example 7.4*

Assume that we want to transmit the data bit $u = 0$ using the 3-repetition code of Example 7.1. Information bit $u = 0$ is encoded onto codeword $\boldsymbol{c} = (0,0,0)$, which is modulated to $\boldsymbol{x} = (+1,+1,+1)$. After transmission over the AWGN channel, we receive $\boldsymbol{y} = (-0.2,+1.1,-0.7)$. We can see that two sign changes occurred during transmission, so a HDD looking purely at the sign (and hence minimizing the Hamming distance) will erroneously assume that the codeword $(1,1,1)$ was transmitted. However, when we carry out the SDD ML decoding rule, we observe that for codewords $(0,0,0)$ and $(1,1,1)$ we get

$$(0,0,0): \quad \sum_{t=1}^{3} y_t(-1)^0 = +0.2 \,, \tag{7.47}$$
$$(1,1,1): \quad \sum_{t=1}^{3} y_t(-1)^1 = -0.2 \,. \tag{7.48}$$

We can see that the sum is maximized for $(0,0,0)$; thus the decoder (correctly) decides for $\hat{\boldsymbol{c}} = (0,0,0)$ and hence $\hat{u} = 0$.

The example above highlights that HDD incurs a performance penalty with respect to SDD. On the other hand, HDD is usually significantly less complex than SDD.

#### Decoding Complexity

The complexity of the ML decoder scales exponentially with the number of data bits $k$. Therefore, ML decoding is only feasible for simple (short) codes and quickly becomes prohibitive as $k$ increases. For practical codes, $k$ is easily above 10 000, which means that the code can contain more than $2 \times 10^{3010}$ codewords. If we assume that we have access to the 500 largest supercomputers worldwide, which have an aggregated computational capacity of 748 PFlop/s [7.106], and that we only require a single floating point operation to compute the sum required in (7.43) or the Hamming distance in (7.44), we would still require $2.7 \times 10^{2992}$ seconds to evaluate all codewords. As the universe has existed for roughly $4.3 \times 10^{17}$ seconds, we see that all efforts to carry out ML decoding for such codes are futile. Therefore, suboptimal decoding strategies which yield an acceptable decoding complexity are required.

The quest for long, powerful codes with low decoding complexity has given rise to so-called *modern codes*, such as LDPC codes and product-like codes, which build upon simpler component codes and are decoded using suboptimal (iterative) decoding algorithms, with reasonable decoding complexity and yet astonishing performance. Suboptimal decoding strategies are discussed in Sects. 7.4 and 7.5.

### 7.3.4 Coding Gain and Net Coding Gain

An important measure for comparing different coding schemes is the NCG. To determine the NCG, bipolar signaling and an AWGN channel are frequently used. However, the definition can be easily extended to other types of channels. Consider an AWGN channel with SNR $E_s/N_0$. Since the modulated symbol $x$ is obtained as $x = (-1)^{c_i}$, all modulation symbols have the same energy, and hence $E_s$ also denotes the *energy per coded bit* in this setting. We therefore have $E_s = \mathbb{E}[X^2] = 1$. In a transmission system, $E_s$ is usually fixed (for example, to the value corresponding to the laser transmit power). Here, we assume a normalized transmit energy $E_s = 1$. Instead of the SNR $E_s/N_0$, for a fair comparison of coding schemes it is helpful to use the ratio $E_b/N_0$, where $E_b$ denotes the average *energy per information bit*. For example, if a code of rate $R = 4/5$ is used, corresponding to overhead OH = 25%, the ratio of code bits $n$ versus information bits $k$ amounts to $n/k = 5/4 = 1.25$, i.e., 1.25 code bits are transmitted for each information bit. This means that if the code bits are transmitted each with energy $E_s$, the equivalent amount of energy conveyed by each information bit amounts to $E_b \triangleq E_s/R \geq E_s$. As $E_b$ is normalized to the information bits of the transmission system, it allows us to evaluate the NCG, i.e., the gain that is actually achievable by taking into account the coding. The NCG is defined as the gain (in dB) of the coded transmission in terms of $E_b/N_0$ relative to uncoded transmission for a given output BER.

Figure 7.8 illustrates the NCG and the influence of $E_b/N_0$ versus $E_s/N_0$. First, we consider uncoded transmission ($R = 1$), where $E_b = E_s = 1$. In this case, the BER is obtained as $Q(\sqrt{2E_b/N_0}) = Q(\sqrt{2E_s/N_0})$. If we now apply coding with a rate $R = 4/5$, we get

$$\frac{E_b}{N_0} = \frac{E_s}{RN_0} = \frac{1}{R}\frac{E_s}{N_0}, \tag{7.49}$$

i.e., $E_b/N_0$ is scaled by a factor $1/R$, corresponding to $-10\log_{10}(R) = 0.969$ dB. Hence, by applying coding, the energy per information bit is scaled by this factor, i.e., each information bit carries 0.969 dB additional energy compared with an uncoded transmission. We may also take on a different perspective and fix the energy per information bit $E_b$ (e.g., to $E_b = 1$). From this perspective, as coding increases the effective number of transmitted bits per information bit, the effective energy per transmitted symbol is decreased. In this case, when the receiver cannot apply decoding but needs to recover only the (systematic) information bits from the sequence of transmitted bits, it requires an SNR that is increased by 0.969 dB to achieve the same BER. By carrying out decoding, we can now significantly improve the post-decoding BER starting from this shifted curve, as can be seen in Fig. 7.8. This increase corresponds to the coding gain (sometimes also called *gross* coding gain), indicated by the bottom arrow in Fig. 7.8. The NCG is reduced by the 0.969 dB required for transmitting the extra parity bits. In this example, we see that the coding scheme employed here yields an NCG of 11.9 dB at a BER of $10^{-15}$.

In optical communications, the optical signal-to-noise ratio (OSNR) is also frequently employed. The OSNR is the SNR measured in a reference optical bandwidth, where frequently a bandwidth $B_{ref}$ of 12.5 GHz is used, corresponding to 0.1 nm-resolution wavelength at a carrier wavelength of 1550 nm. The OSNR relates to the $E_s/N_0$ as follows

$$\begin{aligned} 10\log_{10}\mathrm{OSNR} &= 10\log_{10}\frac{E_s}{N_0} + 10\log_{10}\frac{R_s}{B_{ref}} \\ &= 10\log_{10}\frac{E_b}{N_0} + 10\log_{10}\frac{mRR_s}{B_{ref}}, \end{aligned} \tag{7.50}$$

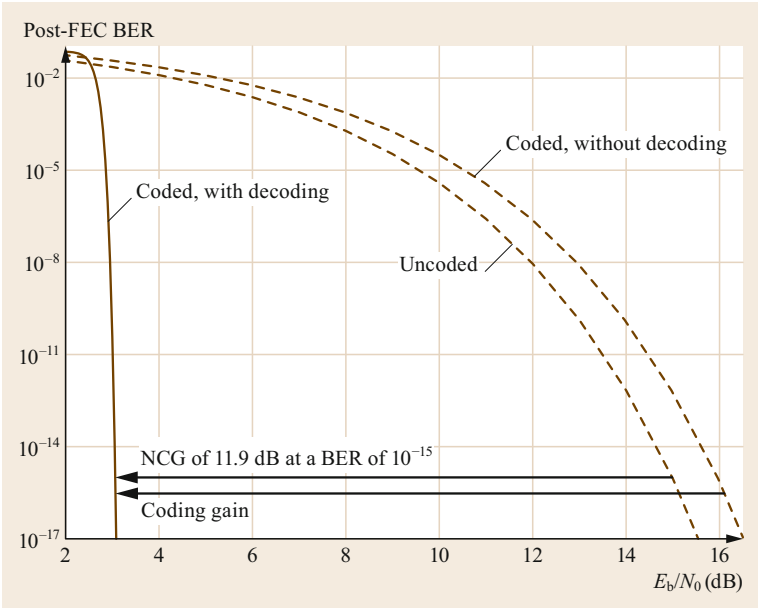


**Fig. 7.8** Net coding gain definition

where $B_{ref}$ is the reference bandwidth, $R_s$ corresponds to the symbol rate of the transmission, $R$ is the code rate, and $m$ is the number of bits mapped to each modulation symbol.

### 7.3.5 Bounds on the Performance of Block Codes

From information theory (Sect. 7.2), we know that the required SNR for successful transmission, based on Shannon's capacity results, assumes infinite block lengths. This is usually not a valid assumption in practical communication systems. Unfortunately, selecting a finite (moderate) code length does not necessarily lead to good performance. Much work has been dedicated to obtaining precise bounds on the performance of transmission systems employing codes of finite block length $n$. A good overview of results can be found in [7.107] and [7.108], where the latter provides refined finite-length performance bounds. Shannon already provided a lower bound on the codeword error probability $P_w$, i.e., the probability that the decoder decodes an incorrect codeword $\hat{\boldsymbol{c}} \neq \boldsymbol{c}$, based on geometric arguments [7.107, 109].

In the following we use the normal approximation of the finite-block-length bound for the binary-input AWGN channel given in [7.110]. This bound allows us to compute an approximation of the gap relative to capacity for the binary-input AWGN channel (which describes BPSK and QPSK modulation with AWGN noise, see also at the beginning of Sect. 7.3.3). We assume that the decoder targets a codeword error probability of $P_w = 10^{-13}$, which means that one in $10^{13}$ codewords can be decoded erroneously. With this, the bit error probability $P_b$ can be bounded as

$$\frac{1}{Rn} P_w \leq P_b \leq P_w \,, \tag{7.51}$$

where the upper bound assumes that all bits inside an erroneously decoded codeword are in error, and the lower bound assumes a single bit error only. Figure 7.9 shows

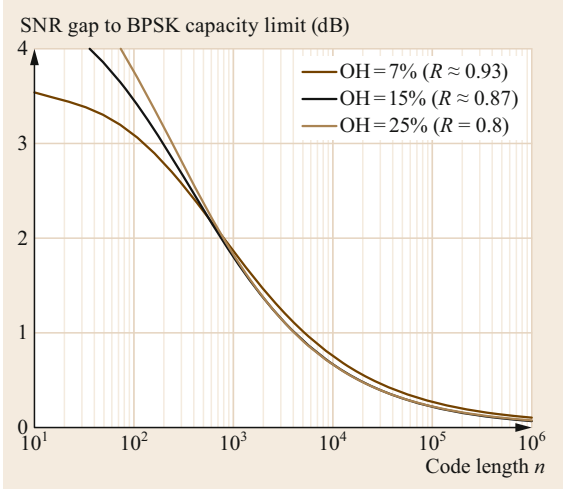


**Fig. 7.9** Gap to the BPSK capacity as a function of the block size $n$ for codes of different rates $R$ and a codeword error probability $P_w = 10^{-13}$

the normal approximation for OH $\in \{7\%, 15\%, 25\%\}$ and a target codeword error rate of $P_w = 10^{-13}$, which is a value commonly used in fiber-optical communications (when the target $P_b$ is around $10^{-15}$). We often use Monte Carlo simulations to estimate the probabilities $P_w$ and $P_b$. In this case, the estimated quantities are referred to as frame error rate (FER) and BER, respectively. In the remainder of this chapter, with some abuse of language, we will often use FER to refer to $P_w$ and BER to refer to $P_b$.

We see in Fig. 7.9 that we need block sizes $n >$ 5 kbit to be able to operate within 1 dB from the channel capacity. Unfortunately, most classical coding schemes are only defined for small to medium $n$ (usually less than 1 kbit) or do not perform well for large $n$. Code concatenation was a first step toward the realization of longer codes that proved to be very successful with HDD. In the next section, we will introduce different coding schemes that naturally allow the construction of long codes.

## 7.4 Soft-Decision Forward Error Correction

Contrary to the historical development of FEC in fiber-optic communications, we start by introducing SDD, and later discuss HDD in Sect. 7.5. The reason for doing so is that SDD immediately follows when optimally decoding many channels that are encountered in optical communications, such as the AWGN channel (Sect. 7.3.3). Furthermore, recently proposed hard-decision FEC schemes have strong ties with soft-decision FEC schemes (such as LDPC codes) and their analysis tools; hence it is somewhat natural to discuss these first. In this section, we introduce some popular coding schemes for SDD in fiber-optic communications, including LDPC codes, SC-LDPC codes, and polar codes. We discuss these codes and their variants in detail, and give detailed descriptions of the most popular decoding algorithms and construction methods. We further show the strengths and weaknesses of the different schemes by means of various simulation examples.

### 7.4.1 Low-Density Parity-Check Codes

We have seen in Sect. 7.3.5 that codes with large block sizes $n$ are required to operate close to the achievable rate or capacity of the channel. For example, from Fig. 7.9, $n$ should be $\geq 2\times10^4$ if we want to operate within 0.5 dB of the capacity of the binary-input AWGN channel with a code of overhead OH = 25%. As seen in Sect. 7.3.1, a code is completely specified by its parity-check matrix $\mathbf{H}$, of dimensions $m \times n$, with $m < n$. Most of classical coding theory focused on finding algebraic constructions for $\mathbf{H}$ (e.g., BCH codes), avoiding the need to store $\mathbf{H}$. However, in the general case, when $\mathbf{H}$ does not contain any *structure* and is random-like, storing it is difficult. The storage requirements for $\mathbf{H}$ scale as $O(n^2)$, and hence it may already not be feasible to store $\mathbf{H}$ for relatively small values of $n$.

We thus need practical codes that have enough structure to enable storage of $\mathbf{H}$ using data structures with low memory requirements and that allow operation close to the achievable rate. Inspired by the fact that random parity-check codes, i.e., codes where the entries $h_{i,j}$ of $\mathbf{H}$ are chosen to be *0* or *1* with probability 1/2, are capacity-achieving, to construct codes with high coding gains and efficient storage of $\mathbf{H}$, *Gallager* introduced LDPC codes in the 1960s [7.54].

An LDPC code is generally defined by a *sparse* binary parity-check matrix $\mathbf{H}$. Recall from Sect. 7.3 that each column of the parity-check matrix $\mathbf{H}$ corresponds to a code bit. Likewise, each row of $\mathbf{H}$ corresponds to a parity-check equation and ideally defines a single parity bit (if $\mathbf{H}$ has full rank). Sparse means that the number of *1*s in $\mathbf{H}$ is small compared with the number of zero entries. Practical codes usually have a fraction of *1*s that is below 1% by several orders of magnitude. Usually, provided that the parity-check matrix has full row rank, i.e., rank($\mathbf{H}$) $= m$, the number of information bits equals $k = n - m$. If the parity-check matrix $\mathbf{H}$ is rank-deficient, the number of information bits $k$ can be larger, i.e., $k \geq n - m$. In the remainder of this chapter, we fix $k = n - m$, i.e., we do not actively use any possible additional information bits. In this case, the rate is given by $R = (n - m)/n$ and the overhead equals OH $= m/(n - m)$.

LDPC codes are often subdivided into two major classes, *regular* and *irregular* LDPC codes. We describe both classes in what follows.

#### Regular Low-Density Parity-Check Codes

*Definition 7.9 Regular $(d_v, d_c)$ LDPC Code*
A regular $(d_v, d_c)$ LDPC code is an LDPC code where each column of $\mathbf{H}$ is of weight $d_v$ (i.e., it contains exactly $d_v$ *1*s), and each row of $\mathbf{H}$ is of weight $d_c$. We assume $d_v \geq 2$ and $d_c \geq 2$.

The storage complexity of $(d_v, d_c)$ LDPC codes is only $O(n)$, as the column weight is constant (and indepen-

dent of $n$), which enables efficient implementation of LDPC codes with low memory requirements.

As an example, consider the $(3, 6)$ LDPC code of rate $R = 1/2$, with $n = 18$, $m = 9$ defined by the parity-check matrix

$$\mathbf{H} = \begin{pmatrix} 0&1&0&0&1&0&0&0&1&0&1&0&0&0&1&0&1&0 \\ 1&0&0&0&0&1&1&0&0&0&0&1&0&1&0&1&0&0 \\ 0&0&1&1&0&0&0&1&0&1&0&0&1&0&0&0&0&1 \\ 0&1&0&0&0&1&0&0&1&0&1&0&0&0&1&1&0&0 \\ 0&0&1&0&1&0&0&1&0&0&0&1&0&1&0&0&1&0 \\ 1&0&0&1&0&0&1&0&0&1&0&0&1&0&0&0&0&1 \\ 0&1&0&0&0&1&0&0&1&1&0&0&1&0&0&1&0&0 \\ 1&0&0&0&1&0&1&0&0&0&0&1&0&1&0&0&1&0 \\ 0&0&1&1&0&0&0&1&0&0&1&0&0&0&1&0&0&1 \end{pmatrix} . \tag{7.52}$$

We summarize in the following some of the properties of regular $(d_\mathrm{v}, d_\mathrm{c})$ LDPC codes:

- $n\,d_\mathrm{v} = m\,d_\mathrm{c}$ has to hold, as the total number of *1*s of $\mathbf{H}$ is identical regardless of the way we count them. Therefore, the rate $R$ is given by

$$R = 1 - \frac{m}{n} = 1 - \frac{n\,d_\mathrm{v}}{n\,d_\mathrm{c}} = 1 - \frac{d_\mathrm{v}}{d_\mathrm{c}} \,. \tag{7.53}$$

- If $d_\mathrm{v}$ is even, then the summation of all rows of $\mathbf{H}$ is the all-zero row, i.e., there is at least one linearly dependent row, i.e., $\mathrm{rank}(\mathbf{H}) \leq m - 1$.
- With high probability, the parity-check matrix of a regular LDPC code has asymptotically full rank [7.111], i.e.,

$$\lim_{n\to\infty} P(\mathrm{rank}(\mathbf{H}) = m - \mathbb{1}_{\{d_\mathrm{v}\ \mathrm{even}\}}) = 1 \,. \tag{7.54}$$

Already for small values of $n$, the limit in (7.54) is approached.

### Irregular Low-Density Parity-Check Codes

In some cases it can be advantageous to relax the regularity and allow codes where the *distribution* of *1*s in the parity-check matrix is *irregular* [7.112]. This gives rise to *irregular* LDPC codes, which are formally defined as follows.

*Definition 7.10 Irregular $(\boldsymbol{\delta}_\mathrm{v}, \boldsymbol{\delta}_\mathrm{c})$ LDPC Code*

An irregular $(\boldsymbol{\delta}_\mathrm{v}, \boldsymbol{\delta}_\mathrm{c})$ LDPC code, with $\boldsymbol{\delta}_\mathrm{v} = (\delta_{\mathrm{v},1}, \delta_{\mathrm{v},2}, \ldots, \delta_{\mathrm{v},d_{\mathrm{v,max}}})$ and $\boldsymbol{\delta}_\mathrm{c} = (\delta_{\mathrm{c},1}, \delta_{\mathrm{c},2}, \ldots, \delta_{\mathrm{c},d_{\mathrm{c,max}}})$ is an LDPC code with $\delta_{\mathrm{v},i}\,n$ columns of $\mathbf{H}$ of weight $i$ and $\delta_{\mathrm{c},j}\,m$ rows of $\mathbf{H}$ of weight $j$. We assume that $\delta_{\mathrm{v},1} = \delta_{\mathrm{c},1} = 0$ and that $\delta_{\mathrm{v},i}\,n$ and $\delta_{\mathrm{c},j}\,m$ are integers for all $i \in \{2, \ldots, d_{\mathrm{v,max}}\}$ and $j \in \{2, \ldots, d_{\mathrm{c,max}}\}$, respectively. Note that $\sum_{i=1}^{d_{\mathrm{v,max}}} \delta_{\mathrm{v},i} = \sum_{j=1}^{d_{\mathrm{c,max}}} \delta_{\mathrm{c},j} = 1$. The vectors $\boldsymbol{\delta}_\mathrm{v}$ and $\boldsymbol{\delta}_\mathrm{c}$ are called the *degree distribution* (from a node perspective) of the code.

Note that irregular codes comprise the class of regular LDPC codes. Indeed, a regular $(d_\mathrm{v}, d_\mathrm{c})$ LDPC code is described by the degree distribution vectors $\boldsymbol{\delta}_\mathrm{v}$ and $\boldsymbol{\delta}_\mathrm{c}$ with $\delta_{\mathrm{v},d_\mathrm{v}} = 1$ and $\delta_{\mathrm{v},i} = 0$, $\forall i \neq d_\mathrm{v}$, and $\delta_{\mathrm{c},d_\mathrm{c}} = 1$ and $\delta_{\mathrm{c},i} = 0$, $\forall i \neq d_\mathrm{c}$. The rate of an irregular LDPC code can be expressed as a function of the vectors $\boldsymbol{\delta}_\mathrm{v}$ and $\boldsymbol{\delta}_\mathrm{c}$, namely

$$R = 1 - \frac{\sum_{i=1}^{d_{\mathrm{v,max}}} i\,\delta_{\mathrm{v},i}}{\sum_{j=1}^{d_{\mathrm{c,max}}} j\,\delta_{\mathrm{c},j}} \,. \tag{7.55}$$

An important generalization of regular and irregular LDPC codes are multi-edge type (MET) LDPC codes [7.111, Sect. 7.1]. In this class of codes, the parity-check matrix is divided into different regions, called types. Each such region shares some common properties. Some high-performing codes with low error floors have been constructed using this approach. One example is the so-called AR4JA codes [7.113] which have been standardized for deep-space communications [7.114]. These codes, which often require punctured state variables, have not yet found widespread application in fiber-optic communications, as their convergence is relatively slow and a larger number of decoding iterations is required to reach the desired performance.

### Graphical Representation of Low-Density Parity-Check Codes

A widespread representation of LDPC codes is based on the so-called *Tanner graphs*, introduced in 1981 by *Michael Tanner* [7.115] as a tool for building large codes from small component codes. Tanner graphs are useful for the design and analysis of LDPC codes and can also serve as a tool for deriving and understanding the most basic decoding algorithm. In what follows, we introduce Tanner graphs by means of an example.

Let $C$ be a binary (LDPC) code with parity-check matrix $\mathbf{H}$ of size $m \times n$. The Tanner graph of the code $C$ is a *bipartite graph*, i.e., it has two types of nodes, and every edge connects a node of the first type with a node of the second type (i.e., there are no connections between nodes of the same type):

- The first type of nodes are called variable nodes (VNs) (they are also sometimes called *bit nodes*), and there are $n$ of them, each one corresponding to a code bit $c_i$ and thus to *column* $i$ of $\mathbf{H}$. We draw the VNs by open circles ○.

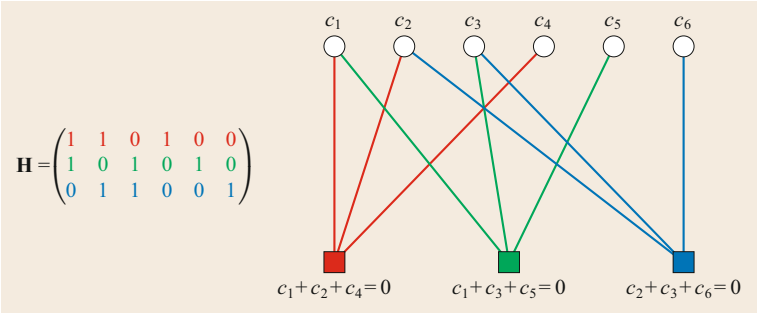


**Fig. 7.10** Tanner graph representation of the $(6, 3)$ code given by the parity-check matrix $\mathbf{H}$ in (7.56)

- The second type of nodes are called check nodes (CNs), and there are $m$ of them. Each CN corresponds to a parity-check constraint of the code and thus to a *row* of $\mathbf{H}$. We draw the CNs by filled squares ■.

CN $j$ is connected by an edge to VN $c_i$ if and only if $h_{j,i} = 1$, i.e., if $\mathbf{H}$ contains a $1$ in row $j$ and column $i$. We then say that code bit $c_i$ participates in the $j$-th parity-check constraint.

Let us illustrate the construction of a Tanner graph using the following $(6, 3)$ code with parity-check matrix

$$\mathbf{H} = \begin{pmatrix} 1 & 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 0 & 1 \end{pmatrix} . \tag{7.56}$$

The Tanner graph of this code is shown in Fig. 7.10. We use different colors to illustrate the relation between CNs and rows of $\mathbf{H}$ (which use the same coloring).

Consider as a second example the parity-check matrix in (7.52). The Tanner graph corresponding to this matrix is shown in Fig. 7.11. Each of the nine CNs corresponds to a row of the parity-check matrix $\mathbf{H}$. Every code bit has $d_v = 3$ adjacent edges, and every CN has $d_c = 6$ adjacent edges.

The Tanner graph can be described by the connection sets $\mathcal{M}(j)$ and $\mathcal{N}(i)$. The set $\mathcal{M}(j) = \{i : h_{j,i} \neq 0\}$ contains the positions (columns) of nonzero entries at row $j$ of the parity-check matrix $\mathbf{H}$. For the exemplary matrix in (7.52) we have $\mathcal{M}(1) = \{2, 5, 9, 11, 15, 17\}$ and $\mathcal{M}(4) = \{2, 6, 9, 11, 15, 16\}$. Additionally, the set $\mathcal{N}(i) = \{j : h_{j,i} \neq 0\}$ contains the positions (rows) of nonzero entries at column $i$ of the parity-check matrix $\mathbf{H}$. For the matrix in (7.52) we have $\mathcal{N}(1) = \{2, 6, 8\}$ and $\mathcal{N}(2) = \{1, 4, 7\}$.

### Decoding Behavior of Low-Density Parity-Check Codes

Before discussing the decoder implementation in detail, we give a high-level overview of the post-FEC BER behavior of LDPC codes as a function of the channel quality (e.g., $E_b/N_0$). The decoding behavior of LDPC codes is visualized in Fig. 7.12 for an example code. We can distinguish three regions of the BER curve. Up to a specific $E_b/N_0$ value, the post-FEC BER does not change significantly. For $E_b/N_0$ values above this value, in the so-called *waterfall region* (highlighted in gray color in the figure), the post-FEC BER decreases rapidly. The slope of this rapidly decreasing BER curve depends mainly on the length of the code $n$, with longer codes leading to narrower waterfall regions. Finally, at a certain $E_b/N_0$, the waterfall region ends, the slope of the BER curve abruptly changes, and we observe an *error floor*. In the error floor region, the post-FEC BER only slowly decreases with increasing $E_b/N_0$. This effect is due mainly to the presence of some structures in

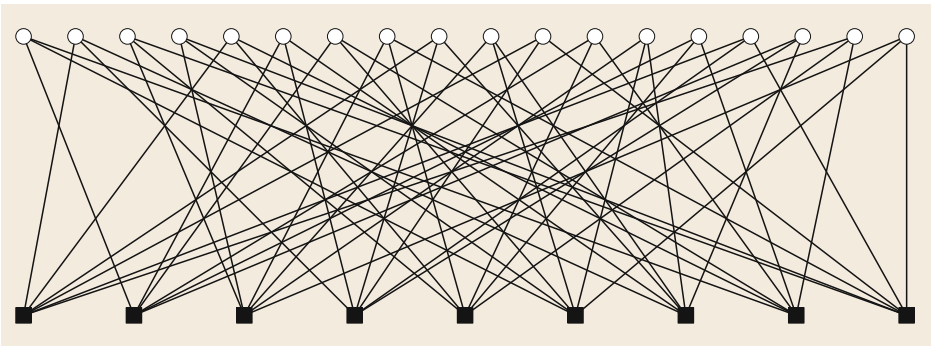

**Fig. 7.11** Tanner graph representation of the code with parity-check matrix given by (7.52)

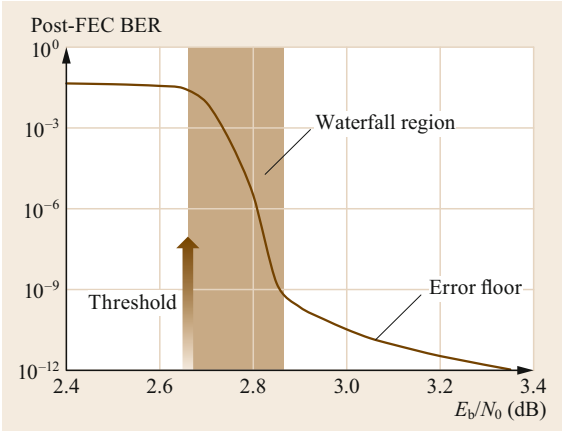


**Fig. 7.12** Description of the post-FEC BER behavior for LDPC codes with waterfall region and error floor

the parity-check matrix and Tanner graph of the code that cause the decoder to stall for some specific error patterns [7.63, 116].

The value of $E_b/N_0$ where the LDPC code performance curve bends into the characteristic waterfall behavior, to which with some abuse of language we refer here as *threshold* (Fig. 7.12), depends on the parameters of the code (i.e., $\boldsymbol{\delta}_v$ and $\boldsymbol{\delta}_c$) and the channel. It can be determined using a technique called *density evolution* [7.117] or approximations thereof [7.118]. Formally speaking, the threshold denotes the $E_b/N_0$ value at which decoding will be successful (the BER tends to zero), with probability approaching 1 in the asymptotic limit of large block length ($n \to \infty$), provided that the decoder complexity is not constrained. For a detailed discussion, we refer the interested reader to [7.117]. Thresholds of some codes for the binary-input AWGN channel are given in Table 7.3, together with the channel capacity.

From Table 7.3, we see that the thresholds of some irregular LDPC codes are very close to the capacity limits of the binary-input AWGN channel. Indeed, irregular LDPC codes are a very important class of capacity-approaching codes [7.119]. Unfortunately, capacity-approaching irregular LDPC codes typically require parity-check matrices with many weight-2 columns (i.e., $\delta_{v,2} > 0$). Such codes usually have error floors in the range of $10^{-4}$ to $10^{-8}$, depending also on $n$, that are unacceptable in fiber-optic communications. Hence, there is a trade-off between regular LDPC codes, which usually have low error floors but thresholds bounded away from capacity, and irregular codes, which have thresholds closer to capacity, but often entail unacceptably high error floors. Irregular codes have found important application in fiber-optic communications, especially when designed jointly with iterative decoding and demodulation schemes [7.120, 121]. Later, in Sect. 7.4.5, we will show a simulation example illustrating irregular LDPC codes.

Due to the rapid decrease in BER in the waterfall region, and because of other outstanding properties, LDPC codes for SDD in fiber-optic communications have been studied in multiple publications [7.60, 61, 86, 121, 122]. Due to the error floor, some modern high-performance FEC systems are constructed using a soft-decision LDPC inner code, which reduces the BER to a level of $10^{-3}$ to $10^{-5}$, and a hard-decision outer code which pushes the system BER to levels below $10^{-12}$ [7.61]. The implementation of a coding system with an outer cleanup code requires a thorough understanding of the LDPC code and a properly designed interleaver between the LDPC and outer code so that the errors at the output of the LDPC decoder—which typically occur in bursts—do not cause uncorrectable blocks after outer decoding. With increasing computing resources, and in particular field-programmable gate array (FPGA)-based decoders, it has now also become feasible to evaluate very low target BERs of LDPC codes and optimize the codes to have very low error floors, below the target BER of the system [7.64, 86].

### Construction of Low-Density Parity-Check Matrices

Unlike classical codes such as BCH codes or Reed–Solomon codes, there is no commonly accepted design rule for LDPC codes. Instead, a plethora of LDPC code design methodologies and heuristics exist, each with its own advantages and disadvantages. The goal of an

**Table 7.3** Decoding thresholds of some regular and irregular LDPC codes on the AWGN channel

| Code | Rate $R$ | $E_b/N_0$ threshold (dB) | Capacity ($E_b/N_0$) |
|---|---|---|---|
| Regular (3, 6) | 1/2 | 1.113 dB | 0.187 dB |
| Regular (4, 8) | 1/2 | 1.618 dB | 0.187 dB |
| Irregular code from [7.119] | 1/2 | 0.192 dB | 0.187 dB |
| Regular (3, 15) | 4/5 | 2.586 dB | 2.039 dB |
| Regular (4, 20) | 4/5 | 2.669 dB | 2.039 dB |
| Irregular $\boldsymbol{\delta}_v = (0, 0, (9/10), 0, 0, 0, 0, 0, (1/10))$, $d_c = 18$ | 4/5 | 2.441 dB | 2.039 dB |
| Irregular $\boldsymbol{\delta}_v = (0, 0.4, 0.36, 0, 0, 0, 0, 0.24)$, $d_c = 19$ | 4/5 | 2.232 dB | 2.039 dB |

LDPC code designer is to find a code that yields high coding gains and possesses some structure facilitating the implementation of the encoder and decoder. We point the interested reader to numerous articles published on this topic [7.111, 123, 124] and references therein. We provide in the following two ways of constructing parity-check matrices that can serve as a starting point for developing own coding schemes based on LDPC codes.

Regular $(d_\mathrm{v}, d_\mathrm{c})$ LDPC codes can be constructed using Gallager's method [7.54]. This simple method leads to codes that are sufficient for many purposes. In this method, we define the $m \times n$ matrix $\mathbf{H}$ by a *banded* structure. The rows in $\mathbf{H}$ are divided into $d_\mathrm{v}$ sets with $m/d_\mathrm{v}$ rows in each set (i.e., $m$ must be an integer multiple of $d_\mathrm{v}$). The first set of rows contains $d_\mathrm{c}$ consecutive *1*s ordered from left to right across columns, such that each row in the set contains exactly $d_\mathrm{c}$ *1*s. Note that this implies that $n$ must be an integer multiple of $d_\mathrm{c}$. Every other set is given by a *randomly chosen column permutation* of the first set. For example, a parity-check matrix of a $(3, 4)$ LDPC code of size $9 \times 12$ is constructed as

$$\mathbf{H} = \left(\begin{array}{cccccccccccc}
1&1&1&1&0&0&0&0&0&0&0&0\\
0&0&0&0&1&1&1&1&0&0&0&0\\
0&0&0&0&0&0&0&0&1&1&1&1\\
\hline
1&0&1&0&0&1&0&0&0&1&0&0\\
0&1&0&0&0&0&1&1&0&0&0&1\\
0&0&0&1&1&0&0&0&1&0&1&0\\
\hline
1&0&0&1&0&0&1&0&0&1&0&0\\
0&1&0&0&0&1&0&1&0&0&1&0\\
0&0&1&0&1&0&0&0&1&0&0&1
\end{array}\right) . \quad (7.57)$$

In practice, the permutations should be chosen such that any sub-matrix of the form $\left(\begin{smallmatrix}1&1\\1&1\end{smallmatrix}\right)$, i.e., two columns sharing *1*s in two common rows, is avoided. This is the so-called *row-column (RC) constraint*, which can improve the decoding performance significantly. Note that the RC constraint cannot be fulfilled for the matrix in (7.57), as, by the pigeonhole principle, a necessary condition to fulfill the RC constraint with Gallager's construction is that $m \geq d_\mathrm{v} d_\mathrm{c}$. Further note that due to the constraints of the banded Gallager design method, the rank of $\mathbf{H}$ is upper bounded as $\mathrm{rank}(\mathbf{H}) \leq m - d_\mathrm{v} + 1$.

Parity-check matrices can also be obtained via the Tanner graph. This method is particularly useful for irregular LDPC codes. We start with an empty graph and first place the $n$ VNs and the $m$ CNs. Note that each VN of degree $i$ is equipped with $i$ (empty) sockets that connect to an edge in the final graph. Similarly, each CN of degree $j$ is equipped with $j$ empty sockets. In the next step, we place the $n \sum_{i=1}^{d_{\mathrm{v,max}}} i\delta_{\mathrm{v},i}$ edges of the graph as follows: we connect each empty VN socket with a randomly chosen, empty CN socket by an edge such that no VN is connected to the same CN more than once (i.e., parallel edges in the Tanner graph are avoided). As there is a one-to-one correspondence between the Tanner graph and the parity-check matrix, we can construct $\mathbf{H}$ from the graph. Such a matrix will most likely not fulfill the RC constraint. In that case, we may either add a post-processing step that reshuffles some of the edges (i.e., entries of $\mathbf{H}$) such that the RC constraint is fulfilled without changing the degree distribution, or try to fulfill the RC constraint directly while picking the sockets.

These constructions can serve only as a starting point for good designs of parity-check matrices. A popular improvement to the graph-based construction described above is the progressive edge growth (PEG) algorithm [7.125], which adds edges sequentially such that the length of the shortest cycle in the graph is kept small. The ACE algorithm [7.126] also attempts to design LDPC codes with large minimum distance. It can be combined with the PEG algorithm [7.127] as well. In this chapter, we are able to just scratch the surface of all the possibilities and point to the vast body of literature describing different construction methods. For a deep treatment of LDPC codes together with an overview of construction methods, we refer the interested reader to [7.101, Chaps. 6, 10–12].

### Quasi-Cyclic Low-Density Parity-Check Codes

The parity-check matrices $\mathbf{H}$ constructed using the methods highlighted in the previous section will be random-like due to the random permutations used in the construction. Unfortunately, when trying to construct a decoder application-specific integrated circuit (ASIC) or implement a decoder on an FPGA, the wiring and routing complexity will usually be prohibitively high even for moderate values of $n$. Hence, we need constructions with structures that can be leveraged to simplify the wiring and routing in the decoder. For this reason, most LDPC codes that are implemented in practice today are so-called quasi-cyclic (QC) LDPC codes. They have a parity-check matrix with a structure that allows for inherent parallelization of the decoder, reduced wiring complexity, and an efficient encoder realization. QC LDPC codes are constructed using a so-called lifting matrix $\mathbf{A}$. The lifting matrix has integer entries $a_{i,j} \in \{-1, 0, 1, \ldots, S-1\}$, where $S$ denotes the *lifting factor*. The parity-check matrix $\mathbf{H}$ is constructed from $\mathbf{A}$ by replacing each element $a_{i,j}$ with either an all-zero matrix of size $S \times S$ or a cyclically permuted identity matrix of size $S \times S$. If $a_{i,j} = -1$, then the all-zero matrix of size $S \times S$ is used, and if $a_{i,j} \geq 0$, then $a_{i,j}$ denotes the number of cyclic right shifts of the identity matrix that are carried out. If $\dim(\mathbf{A}) = m' \times n'$, then $\dim(\mathbf{H}) = m \times n$, with $m = m'S$ and $n = n'S$.

Formally, we construct the parity-check matrix $\mathbf{H}$ as

$$\mathbf{H} = \begin{pmatrix} \mathbf{P}^{a_{1,1}} & \mathbf{P}^{a_{1,2}} & \dots & \mathbf{P}^{a_{1,n'}} \\ \mathbf{P}^{a_{2,1}} & \mathbf{P}^{a_{2,2}} & \dots & \mathbf{P}^{a_{2,n'}} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{P}^{a_{m',1}} & \mathbf{P}^{a_{m',2}} & \dots & \mathbf{P}^{a_{m',n'}} \end{pmatrix}, \tag{7.58}$$

where $\mathbf{P}$ is the matrix of size $S \times S$ with $p_{i,i+1} = 1$, $\forall i \in \{1, \dots, S-1\}$, $p_{S,1} = 1$, and zeros everywhere else. We note that $\mathbf{P}^0 = \mathbf{I}$, the identity matrix, and

$$\mathbf{P}^j = \underbrace{\mathbf{P}\mathbf{P}\cdots\mathbf{P}}_{j \text{ times}} \tag{7.59}$$

equals the identity matrix cyclically shifted to the right by $j$. We further define $\mathbf{P}^{-1} \triangleq \mathbf{0}$, the all-zero matrix.

Typically, in modern FEC schemes, a small code with parity-check matrix $\tilde{\mathbf{H}}$ is first constructed, for example, based on Gallager's method or more advanced schemes such as the PEG and ACE algorithms [7.125, 126, 128], MET analysis and design [7.129], protographs [7.130], or finite geometries [7.101, Chap. 10]. See also [7.131] for an extensive overview and attempts at unifying construction methods. In a second step, we replace the nonzero elements of $\tilde{\mathbf{H}}$ by appropriately selected shift values. As an example, we can take the matrix of (7.57) and convert it to a lifting matrix to obtain, with $S = 16$,

$$\mathbf{A} = \begin{pmatrix} 0 & 0 & 0 & 0 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 \\ -1 & -1 & -1 & -1 & 0 & 0 & 0 & 0 & -1 & -1 & -1 & -1 \\ -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & 0 & 0 & 0 & 0 \\ 3 & -1 & 7 & -1 & -1 & 1 & -1 & -1 & -1 & 3 & -1 & -1 \\ -1 & 5 & -1 & -1 & -1 & -1 & 3 & 6 & -1 & -1 & -1 & 5 \\ -1 & -1 & -1 & 5 & 9 & -1 & -1 & -1 & 9 & -1 & 2 & -1 \\ 7 & -1 & -1 & 3 & -1 & -1 & 12 & -1 & -1 & 10 & -1 & -1 \\ -1 & 4 & -1 & -1 & -1 & 0 & -1 & 4 & -1 & -1 & 7 & -1 \\ -1 & -1 & 14 & -1 & 14 & -1 & -1 & -1 & 8 & -1 & -1 & 9 \end{pmatrix}. \tag{7.60}$$

After replacing the entries of the lifting matrix by $\mathbf{P}^{a_{i,j}}$, we obtain the *lifted*, structured parity-check matrix $\mathbf{H}$ of size $144 \times 192$ shown in Fig. 7.13, where we added dashed lines to serve as visual guides. The *1*s are represented by black dots in Fig. 7.13, and the *0*s are omitted. The main task of the code designer is to find good values $a_{i,j}$. Many design rules for finding these values have been proposed in the literature. Commonly, the values $a_{i,j}$ are selected to maximize the so-called girth of the code, which is indirectly related to the decoding performance [7.132]. In the example above, we have selected the entries $a_{i,j}$ to yield a girth-10 code. Other constructions are aimed at maximizing a bound of the minimum distance [7.133], and use finite geometries [7.101, Chap. 10], matrix dispersions

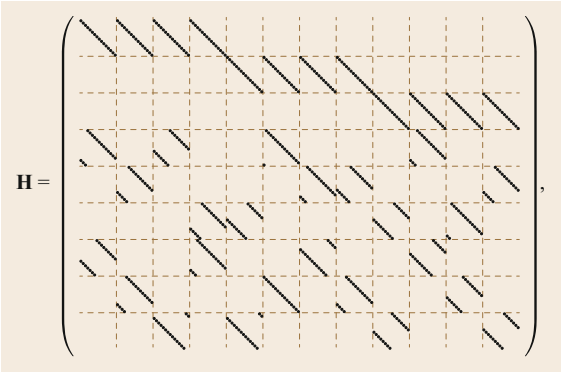


**Fig. 7.13** Lifted structured parity-check matrix of size $144 \times 192$ obtained by lifting the entries of (7.60) with cyclic permutation matrices of size $16 \times 16$

of finite fields [7.101, Chap. 11], [7.131], combinatorial designs [7.101, Chap. 12], the Chinese remainder theorem [7.134], and other methods.

### Encoding Low-Density Parity-Check Codes

In general, LDPC codes can be treated as any other block code when it comes to encoding. The generator matrix $\mathbf{G}$ of the LDPC code must be orthogonal to $\mathbf{H}$, i.e., $\mathbf{G} \cdot \mathbf{H}^\top = \mathbf{0}$. If $\mathbf{H}$ can be rearranged such that $\mathbf{H} = \left(\mathbf{P}^\top \ \mathbf{I}_{n-k}\right)$, then $\mathbf{G} = \left(\mathbf{I}_k \ \mathbf{P}\right)$.

Using Gauss–Jordan elimination, any $\mathbf{H}$ can in principle be converted to $\mathbf{H}' = \left(\bar{\mathbf{H}}^\top \ \mathbf{I}_m\right)$ with $\bar{\mathbf{H}}^\top$ an $m \times k$ binary matrix and $\mathbf{I}_m$ the identity matrix of size $m \times m$. The generator matrix is then obtained as $\mathbf{G} = \left(\mathbf{I}_k \ \bar{\mathbf{H}}\right)$. However, there is a fundamental problem with this simple approach: $\mathbf{G}$ will most likely *not be sparse*; thus the encoding complexity and the storage requirements will be $O(n^2)$, which we initially wanted to avoid when introducing LDPC codes.

Fortunately, there is a method to directly encode LDPC codes from $\mathbf{H}$ [7.111, App. A] that allows for an encoding complexity that scales (almost) linearly with the code length $n$. Instead of finding a generator matrix $\mathbf{G}$ for a given parity-check matrix $\mathbf{H}$, the encoding is carried out directly from $\mathbf{H}$. In a first step, $\mathbf{H}$ is transformed into an approximate triangular form: Using *only row and column permutations*, we construct $\mathbf{H}'$ from $\mathbf{H}$ with

$$\mathbf{H}' = \begin{pmatrix} \mathbf{A} & \mathbf{B} & \mathbf{T} \\ \mathbf{C} & \mathbf{D} & \mathbf{E} \end{pmatrix}, \tag{7.61}$$

where $\mathbf{T}$ is a *lower triangular matrix* of size $(m-g) \times (m-g)$, $\mathbf{B}$ is of size $(m-g) \times g$, and $\mathbf{A}$ is of size $(m-g) \times k$ (if $\mathbf{H}'$ is full rank). The parameter $g$ denotes the number of rows left in $\mathbf{C}$, $\mathbf{D}$, and $\mathbf{E}$ and is called the *gap* of the approximate representation. The smaller $g$ is,

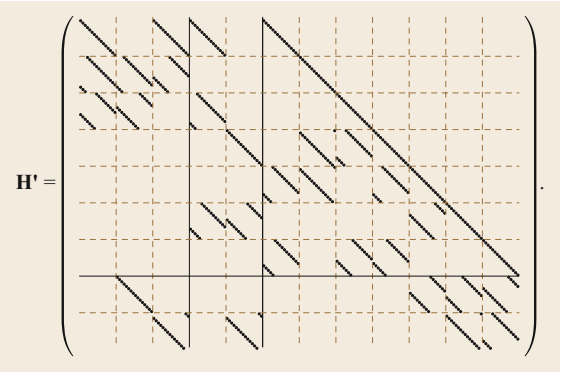


**Fig. 7.14** Parity-check matrix of Fig. 7.13 after transformation into the form of (7.61)

the lower the encoding complexity will be. If the permutation operations are chosen well, we have $g \ll m$. A simple, greedy column- and row-swapping algorithm is presented in [7.111, App. A]. Applying this algorithm yields, for the running example, the matrix $\mathbf{H}'$ shown in Fig. 7.14.

Starting from $\mathbf{H}'$, we can clear $\mathbf{E}$ by Gaussian elimination, or equivalently

$$\begin{pmatrix} \mathbf{I}_{m-g} & \mathbf{0} \\ \mathbf{E}\mathbf{T}^{-1} & \mathbf{I}_g \end{pmatrix} \mathbf{H}' = \begin{pmatrix} \mathbf{A} & \mathbf{B} & \mathbf{T} \\ \widetilde{\mathbf{C}} & \widetilde{\mathbf{D}} & \mathbf{0} \end{pmatrix} \triangleq \widetilde{\mathbf{H}} \,, \tag{7.62}$$

with $\widetilde{\mathbf{C}} = \mathbf{E}\mathbf{T}^{-1}\mathbf{A} + \mathbf{C}$ and $\widetilde{\mathbf{D}} = \mathbf{E}\mathbf{T}^{-1}\mathbf{B} + \mathbf{D}$. Note that addition $+$ and subtraction $-$ are equivalent when operating over GF(2); hence, we can replace all subtractions by additions. Further note that inverting $\mathbf{T}$ is easy, as it is a lower triangular matrix [7.86, App. A]. Also, $\mathbf{T}^{-1}$ will retain its QC structure. Finally, the codeword $\boldsymbol{c}$ is divided into three parts

$$\boldsymbol{c} = (\boldsymbol{u} \;\; \boldsymbol{p}_1 \;\; \boldsymbol{p}_2) \,,$$

where $\boldsymbol{u}$ is the $k$-bit message, $\boldsymbol{p}_1$ holds the first $g$ parity bits, and $\boldsymbol{p}_2$ the remaining $n-k-g$ parity bits. Vector $\boldsymbol{p}_1$ is then obtained from

$$\boldsymbol{p}_1^\top = \widetilde{\mathbf{D}}^{-1}\widetilde{\mathbf{C}}\boldsymbol{u}^\top \,. \tag{7.63}$$

Using back-substitution, $\boldsymbol{p}_2$ can be calculated as

$$\boldsymbol{p}_2^\top = \mathbf{T}^{-1}\left(\mathbf{A}\boldsymbol{u}^\top + \mathbf{B}\boldsymbol{p}_1^\top\right) = \mathbf{T}^{-1}\left(\mathbf{A} + \mathbf{B}\widetilde{\mathbf{D}}^{-1}\widetilde{\mathbf{C}}\right)\boldsymbol{u}^\top \,. \tag{7.64}$$

Although not obvious at first glance, this procedure has some complexity advantages. Many of the intermediate terms needed in the computation are *sparse* and can be precomputed and stored with low complexity. It can be shown that the total encoding complexity is $O(n + g^2)$, which means that if $g$ is very small, the complexity approaches $O(n)$. Note that if the rank of the original matrix $\mathbf{H}$ is not $m$, some parity bits are linearly dependent, and the procedure must be modified slightly to accommodate this fact by either fixing the linearly dependent parity bits to some predetermined value or assigning additional information bits to them. For further details, see also [7.135].

### Decoding Low-Density Parity-Check Codes

As the complexity of optimal MAP decoding grows exponentially with the number of information bits $k$, MAP decoding is usually not feasible for practical codes, and we need to resort to suboptimal decoding algorithms. The success of LDPC codes is largely due to the existence of very powerful suboptimal decoding algorithms that can be implemented with relatively low complexity. These decoding algorithms belong to the class of *message-passing* algorithms, which work by exchanging messages along the edges of the Tanner graph, with very simple operations to compute the messages inside the nodes. The BP algorithm is the most popular message-passing algorithm for decoding LDPC codes. It works by exchanging *beliefs* about the code bits along the edges of the Tanner graph. The BP algorithm is used in many neighboring fields, for instance in machine learning [7.101, 136, 137], but it was described already in the 1960s to decode LDPC codes [7.54].

We describe the decoder starting from the received, noisy codeword $\boldsymbol{y} = (y_1, \ldots, y_n)$. The goal of the decoder is to compute the bit-wise *a posteriori* probability based on the received codeword $\boldsymbol{y}$,

$$P_{C|\boldsymbol{Y}}(c_i = 1|\boldsymbol{y}) = 1 - P_{C|\boldsymbol{Y}}(c_i = 0|\boldsymbol{y}) \,, \tag{7.65}$$

which can be used to estimate the value of the bit $c_i$ as $\hat{c}_i = \arg\max_{c_i \in \{0;1\}} P_{C|\boldsymbol{Y}}(c_i|\boldsymbol{y})$. Note that this rule is in slight contrast to the MAP rule in (7.38), which estimates the most probable codeword instead of the most probable bit. This rule is therefore also called the bit-wise MAP rule, and minimizes the bit error probability.

Most decoders for LDPC codes (which can include simplified versions such as binary message-passing decoders [7.138]) rely on the knowledge of the channel. The communication channel, or an equivalent communication channel comprising the physical channel as well as various inner receiver and signal processing stages, can be characterized by its *conditional probability mass function* $P_{Y|X}$, or equivalently $P_{Y|C}$, as described in Sect. 7.2. Continuous channels are characterized by their *conditional probability density function* $p_{Y|X}$. Here, we assume a binary-input AWGN channel with $x_i = (-1)^{c_i}$. However, the decoder can easily be

extended to other modulation formats using the respective bit-wise decoder (Sects. 7.2.1 and 7.6.3).

The beliefs that are passed by the BP decoder are conditional probabilities, but for numerical reasons, it is much more convenient to work in the logarithmic domain and use LLRs [7.139]. The use of LLRs also facilitates hardware implementation. The channel-related LLR $l_i$ is defined as

$$l_i = \log \frac{p_{Y_i|C_i}(y_i|0)}{p_{Y_i|C_i}(y_i|1)} \,. \tag{7.66}$$

A simple computation reveals that for the AWGN channel with binary, antipodal $\pm 1$ channel input ($x_i = (-1)^{c_i}$) and channel law given by (7.1), we have

$$l_i = \frac{2}{\sigma_n^2} y_i \triangleq L_c^{[\mathrm{AWGN}]}\, y_i \,. \tag{7.67}$$

Equation (7.67) means that the LLR $l_i$ is obtained simply by multiplication of $y_i$ with a constant $L_c^{[\mathrm{AWGN}]} \triangleq 2/\sigma_n^2$, which depends only on the noise variance. Estimating the noise variance at the receiver is an old problem, and many different solutions exist [7.140, 141]. In many cases, the true noise variance is not even necessary. For example, with the min-sum decoder introduced below, we can assume the noise variance to be constant, and hence the constant $L_c^{[\mathrm{AWGN}]}$ is predetermined and set to a value suitable for implementation. If higher-order modulation formats are used with BICM or PAS, we use the bit-wise decoder (see Sects. 7.6.3 and 7.6.5 for details) to compute the LLRs passed to the decoder.

A vast collection of different varieties of the BP message-passing decoding algorithm for LDPC codes exists. We refer the interested reader to [7.101, 142] for an in-depth treatment of these variations. In the following, we describe the two most popular variants: the flooding and the layered decoder.

*The Flooding Sum–Product Decoder.* The flooding sum-product decoder is perhaps the most widely used decoder for LDPC codes. It can be best explained using the Tanner graph representation of the code. The decoding algorithm itself is iterative, and each iteration can be split into two half-iterations. In the first half-iteration, each VN of degree $d_v$ computes $d_v$ outgoing messages, one for each of its outgoing edges. In the second half-iteration, each CN of degree $d_c$ computes $d_c$ messages, one for each of its outgoing edges. We repeat this procedure until a maximum number of iterations has been achieved or the bits have been successfully recovered.

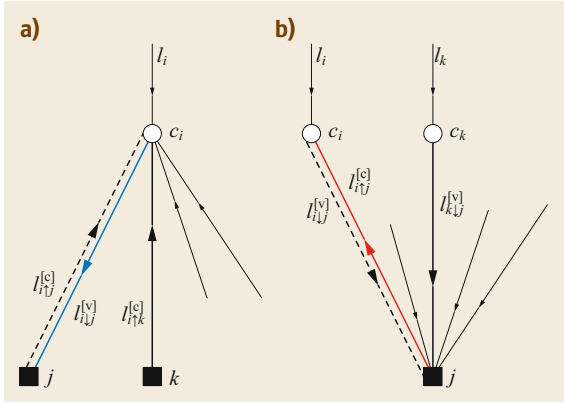


**Fig. 7.15a,b** Illustration of messages of the sum-product flooding decoding inside the Tanner graph. **(a)** VN perspective; **(b)** CN perspective

We denote by $l_{i\downarrow j}^{[v]}$ the message that is computed by VN $i$ (corresponding to code bit $c_i$) and passed to CN $j$ via the edge in the Tanner graph. Similarly, we denote by $l_{i\uparrow j}^{[c]}$ the message computed by CN $j$ and passed (via the edge in the Tanner graph) to VN $c_i$. The superscript [v] or [c] describes the origin of the message (VN or CN), and the direction of the arrow ($\downarrow$ or $\uparrow$) denotes the *direction of the message flow* in the Tanner graph.

Figure 7.15 illustrates the flow of messages along the edges of the Tanner graph, from both a VN perspective (Fig. 7.15a) and a CN perspective (Fig. 7.15b).

We are now ready to state the sum-product algorithm:

1. *(Initialize)* For all $i \in \{1, \ldots, n\}$, set $l_{i\downarrow j}^{[v]} = l_i$, $\forall j \in \mathcal{N}(i)$, where $l_i$ is calculated either using (7.67) (AWGN channel with binary inputs) or using the bit-wise decoder given by (7.93) in Sect. 7.6.3 (if higher-order modulation with BICM or PAS is used).
2. *(CN update)* For every CN $j$, with $j \in \{1, \ldots, m\}$, compute $|\mathcal{M}(j)| = d_c$ outgoing messages

$$l_{i\uparrow j}^{[c]} = 2 \tanh^{-1}\left[\prod_{i' \in \mathcal{M}(j)\setminus\{i\}} \tanh\left(\frac{l_{i'\downarrow j}^{[v]}}{2}\right)\right], \quad \forall i \in \mathcal{M}(j) \,. \tag{7.68}$$

This update equation follows from the probability that a single parity-check equation is fulfilled when the input bits are 0 or 1 with a certain probability and the conversion of probabilities to LLRs. A detailed derivation can be found for instance in [7.101, Sect. 5.4.3].

3. *(VN update)* For every VN $c_i$, with $i \in \{1, \ldots, n\}$, compute $|\mathcal{N}(i)|$ outgoing messages

$$l_{i\downarrow j}^{[\mathrm{v}]} = l_i + \sum_{j' \in \mathcal{N}(i)\backslash\{j\}} l_{i\uparrow j'}^{[\mathrm{c}]}, \qquad \forall j \in \mathcal{N}(i)\,. \quad (7.69)$$

4. *(A posteriori estimate)* For $i \in \{1, \ldots, n\}$, compute

$$\hat{c}_i = \begin{cases} 1 & \text{if } l_i + \sum_{j \in \mathcal{N}(i)} l_{i\uparrow j}^{[\mathrm{c}]} < 0, \\ 0 & \text{otherwise.} \end{cases} \quad (7.70)$$

If $\mathbf{H}\hat{\boldsymbol{c}}^{\top} = \mathbf{0}$, with $\hat{\boldsymbol{c}} = (\hat{c}_1, \ldots, \hat{c}_n)$ or if the maximum number of iterations is reached, then stop; otherwise go to step 2.

The notation $\mathcal{N}(i)\backslash\{j\}$ denotes all the elements of $\mathcal{N}(i)$ except $j$. For the example code of (7.52), we have $\mathcal{N}(2)\backslash\{4\} = \{1, 7\}$.

A straightforward hardware implementation of this algorithm is challenging due to the nature of (7.68). Hence, we will resort to simplified approximations to this update rule that only slightly affects the performance of the decoder. One particularly widely used approximation is the scaled min-sum rule [7.143], which is expressed as

$$l_{i\uparrow j}^{[\mathrm{c}]} = \alpha \left[ \prod_{i' \in \mathcal{M}(j)\backslash\{i\}} \operatorname{sign}\left(l_{i'\downarrow j}^{[\mathrm{v}]}\right) \right] \min_{i' \in \mathcal{M}(j)\backslash\{i\}} \left| l_{i'\downarrow j}^{[\mathrm{v}]} \right|, \quad (7.71)$$

where $\alpha$ is an appropriately chosen scaling factor, which is usually determined offline and which depends on the code. The factor may also be iteration dependent and change during decoding [7.144]. An overview of other approximations and variants is presented in [7.145], [7.101, Sect. 5.5]. The theoretical foundations of this decoding algorithm are beyond the scope of this book. We refer the interested reader to [7.111] for an in-depth treatment of the underlying theory.

*The Layered Decoder.* In what follows, we describe the row-layered variant of the flooding sum-product decoder [7.146]. This variant is optimized for fast convergence and low complexity, and therefore is often used in optical communications. The row-layered decoder can be described directly from the parity-check matrix.

In a first step, the row-layered decoder copies the received LLRs $l_j$ to a memory $\boldsymbol{\mu} = (\mu_1, \mu_2, \ldots, \mu_n)$ of size $n$ with $\mu_j = l_j$, which is continuously updated by the decoder. This memory will be referred to as *a posteriori memory*, as it comprises estimates of the bit-wise a posteriori probabilities. Additionally, the decoder requires an *extrinsic* memory consisting of $m \times d_{\mathrm{c}}$ entries (for a regular code, $m \sum_i i\delta_{\mathrm{c},i}$ for an irregular code), corresponding to the nonzero elements of $\mathbf{H}$. The memory cell $e_{j,i}$ corresponds to row $j$ and column $i$ of $\mathbf{H}$ and is only defined when $h_{j,i} = 1$. We initialize $e_{j,i} = 0, \forall j \in \{1, \ldots, m\}, i \in \mathcal{M}(j)$.

The row-layered decoding operation sequentially carries out three steps for each of the $m$ rows of the parity-check matrix $\mathbf{H}$. After all $m$ rows have been processed, a single decoding iteration has been carried out, and the decoder may restart with the first row to perform an additional iteration. Next we describe the steps for row $j$ of $\mathbf{H}$:

1. *(Computation of input values)* Compute $|\mathcal{M}(j)|$ temporary values $t_{j,i} = \mu_i - e_{j,i}$, $\forall i \in \mathcal{M}(j)$, for all nonzero entries (indexed by $i$) of the $j$-th row of $\mathbf{H}$.
2. *(CN update)* The parity-check update rule is carried out to compute new extrinsic information using the full update rule

$$e_{j,i} = 2 \tanh^{-1}\left[ \prod_{i' \in \mathcal{M}(j)\backslash\{i\}} \tanh\left(\frac{t_{j,i'}}{2}\right) \right], \ \forall i \in \mathcal{M}(j)\,, \quad (7.72)$$

where the product is again taken over all entries in $\mathcal{M}(j)$ except the one under consideration, $i$. Alternatively, e.g., for a hardware implementation, the scaled min-sum update rule

$$e_{j,i} = \alpha \left[ \prod_{i' \in \mathcal{M}(j)\backslash\{i\}} \operatorname{sign}(t_{j,i'}) \right] \min_{i' \in \mathcal{M}(j)\backslash\{i\}} |t_{j,i'}|\,. \quad (7.73)$$

can also be used.
3. *(A posteriori update)* The $|\mathcal{M}(j)|$ a posteriori memory positions that take part in the parity-check equation are updated according to $\mu_i = t_{j,i} + e_{j,i}$, $\forall i \in \mathcal{M}(j)$.

After having carried out the three steps for row $j$, the LDPC decoder continues with decoding row $j+1$ or, if $j = m$ has been reached, it restarts either at $j = 1$ or terminates if the number of iterations has reached a predefined limit. After termination, we can get an estimate of the codeword by evaluating for every code bit

$$\hat{c}_i = \frac{1}{2}\left[1 - \operatorname{sign}(\mu_i)\right]. \quad (7.74)$$

If the code is systematic, the information portion can be recovered by taking the decision at the respective systematic positions.

Because of its sequential updates, this layered decoding algorithm is not immediately suited for massively parallel decoding of arbitrary LDPC codes. However, when used together with QC LDPC codes, we can see that groups of $S$ consecutive rows of $\mathbf{H}$ are always non-interacting. In a decoder, the respective memory accesses are all to different locations, and hence the operations are non-interacting and can be parallelized. The cyclic shifts can be easily realized using barrel shifter circuits. See for instance [7.147] for further details.

### 7.4.2 Spatially Coupled Low-Density Parity-Check Codes

While so far we have focused mostly on block codes (e.g., LDPC codes), a significant number of practical codes were and still are based on simple convolutional codes using simple binary shift registers. About two decades ago, LDPC convolutional codes were introduced [7.71], which superimpose LDPC codes with a convolutional structure. The initial goal was to combine the advantages of convolutional and LDPC codes, and the resulting codes were called *LDPC convolutional codes*. A similar idea had already been pursued in the late 1950s and early 1960s with the concept of recurrent codes ([7.148] and references therein). The full potential of LDPC convolutional codes was not realized until a few years later, when *Lentmaier* et al. noticed that the performance of some LDPC convolutional code constructions asymptotically approached the performance of optimal ML decoding [7.72, 74]. Soon thereafter, this was formally proven for a wide range of channels by *Shrinivas Kudekar* et al. [7.75, 76]. As a proof technique, Kudekar et al. modified the original LDPC convolutional codes slightly and introduced a variation of the construction, denoted as spatially coupled codes. Because of the similarity between the two schemes, we use the term SC-LDPC codes to refer to both constructions for the remainder of this chapter. An alternative proof technique to the one in [7.75, 76] was proposed in [7.149, 150]. The analysis of nonbinary SC-LDPC codes was addressed in [7.151, 152].

#### Definition and Basic Construction

We now extend our description of LDPC codes to SC-LDPC codes. We first take on an encoder perspective. An LDPC encoder is a block encoder, i.e., it encodes a block of information bits $\boldsymbol{u}_t$ into a codeword $\boldsymbol{c}_t$. This is shown in Fig. 7.16a. The encoder of an SC-LDPC code can be interpreted as an encoder of a recurrent code [7.148] and is shown in Fig. 7.16b. The recurrent

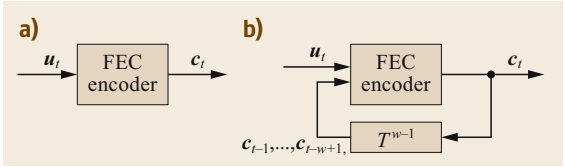


**Fig. 7.16a,b** Comparison of (a) conventional block LDPC encoder and (b) spatially coupled (SC) LDPC encoder

encoder generates *batches* $\boldsymbol{c}_t$ of a codeword which consists of a sequence of such batches. The batch $\boldsymbol{c}_t$ at time $t$ depends not only on $\boldsymbol{u}_t$ but also on $w-1$ previous batches $\boldsymbol{c}_{t-1}, \ldots, \boldsymbol{c}_{t-w+1}$. The parameter $w$ is called the *coupling width* of the code. The term *spatial coupling* highlights the dependence of $\boldsymbol{c}_t$ on previous batches along the *spatial* dimension $t$ (this new, additional dimension was referred to as *spatial* dimension in [7.75]; however, note that it does *not* refer to a physical spatial dimension). Each batch is hence also called a *spatial position*.

At the beginning of the transmission (for $t = 1$), we assume that the contents of the feedback memory are zero, i.e., the previous batches at undefined times $t < 1$ are $\boldsymbol{c}_0 = \boldsymbol{c}_{-1} = \boldsymbol{c}_{-2} = \ldots = \mathbf{0}$. After starting the transmission, we could in principle run the encoding procedure without interrupting it. This is sometimes called *left-termination* or *unterminated* operation. Frequently, however, the encoding is terminated after the transmission of $L$ batches, where $L$ denotes the *replication factor* or *spatial length* of the code. Termination means that $\boldsymbol{c}_t = \mathbf{0}$ for $t > L$, provided that the input $\boldsymbol{u}_t = \mathbf{0}$ for $t > L$. Termination yields a block code again, where the length $n$ of the block code amounts to $Ln_\mathrm{b}$ bits, with $n_\mathrm{b}$ being the size of one batch $\boldsymbol{c}_t$. Termination is realized by slightly reducing the number of information bits to $k'_\mathrm{b} < k_\mathrm{b}$ in the final $w-1$ batches, i.e., $\boldsymbol{u}_{L-w+2}, \ldots, \boldsymbol{u}_L$. The encoder then generates $k_\mathrm{b} - k'_\mathrm{b}$ extra parity bits that are required to enforce the termination condition. These extra parity bits lead to a *rate loss* (i.e., reduce the effective number of information bits). However, note that the effect of the rate loss becomes negligible if $L$ is increased. Termination is often used to fit multiple FEC frames into a larger super-frame (e.g., an optical transport network (OTN) frame [7.153]) or a networking container [7.154], or to prevent error propagation.

Instead of interpreting the terminated encoding procedure as a recurrent procedure of $L$ batches, it is often customary to group the $L$ batches $\boldsymbol{c}_1, \ldots, \boldsymbol{c}_L$ into the codeword $\boldsymbol{c}^{[\mathrm{SC}]} \triangleq \begin{pmatrix} \boldsymbol{c}_1 & \boldsymbol{c}_2 & \ldots & \boldsymbol{c}_L \end{pmatrix}$. We denote the parity-check matrix of the corresponding *terminated* (and non-time-varying) SC-LDPC code with coupling

width $w$ by $\mathbf{H}^{[\mathrm{SC},w]}$. It is given by

$$\mathbf{H}^{[\mathrm{SC},w]} = \begin{pmatrix} \mathbf{H}_1 & & & \\ \vdots & \mathbf{H}_1 & & \\ \mathbf{H}_w & \vdots & \ddots & \\ & \mathbf{H}_w & \ddots & \mathbf{H}_1 \\ & & \ddots & \vdots \\ & & & \mathbf{H}_w \end{pmatrix}, \tag{7.75}$$

with sub-matrices $\mathbf{H}_i$ of dimensions $m_\mathrm{b} \times n_\mathrm{b}$, with $n_\mathrm{b}$ denoting the length of one batch. The simplest case is obtained for $w = 2$ (which we sometimes call *unit-memory* coupling), for which we have

$$\mathbf{H}^{[\mathrm{SC},2]} = \begin{pmatrix} \mathbf{H}_1 & & & \\ \mathbf{H}_2 & \mathbf{H}_1 & & \\ & \mathbf{H}_2 & \ddots & \\ & & \ddots & \mathbf{H}_1 \\ & & & \mathbf{H}_2 \end{pmatrix}. \tag{7.76}$$

The parity-check matrix $\mathbf{H}^{[\mathrm{SC},w]}$ is of dimensions $(L + w - 1)m_\mathrm{b} \times Ln_\mathrm{b}$, and consequently, assuming full rank, the rate of the terminated SC-LDPC code is given by

$$R = 1 - \left(1 + \frac{w-1}{L}\right)\frac{m_\mathrm{b}}{n_\mathrm{b}} \leq 1 - \frac{m_\mathrm{b}}{n_\mathrm{b}}, \tag{7.77}$$

i.e., the rate is decreased by spatial coupling due to the extra rows in $\mathbf{H}^{[\mathrm{SC},w]}$ required for the termination. Note that this *rate loss* vanishes when $L \to \infty$, which is why we usually want to select $L$ that is quite large. Multiple techniques for reducing the rate loss have been investigated [7.155–158].

### Tanner Graph and Construction of Spatially Coupled Low-Density Parity-Check Codes

Due to the structured parity-check matrix $\mathbf{H}^{[\mathrm{SC},w]}$, the Tanner graph of an SC-LDPC code has a particularly simple structure. An SC-LDPC code and the corresponding Tanner graph can be constructed starting from an LDPC code. We show the construction of both the parity-check matrix $\mathbf{H}^{[\mathrm{SC},w]}$ and the Tanner graph by means of an example. We use the *edge-spreading* technique introduced in [7.72], which can be carried out directly in the Tanner graph of an LDPC code.

We start with a Tanner graph of an LDPC code. The first step consists in *coloring* the edges of the graph in $w$ distinct colors or *edge types*. One possibility for coloring the edges is to carry out a random experiment for each edge and assign to it color $w$ with probability $1/w$. This is called *uniform coupling*. The coloring probabilities can also be optimized, for example, to improve the performance of the code [7.158]. The coloring procedure is illustrated in Fig. 7.17. We start with the Tanner graph of a $(3, 6)$ LDPC code which is shown in Fig. 7.17a. We then apply the edge coloring where the two colors are represented by solid and dashed lines, respectively. This is shown in Fig. 7.17b. For each of the two edge colors, we can construct a new Tanner graph by removing all edges of different colors, and from this graph obtain the corresponding parity-check matrices $\mathbf{H}_1$ and $\mathbf{H}_2$. For the example in Fig. 7.17b with $w = 2$, we obtain

$$\mathbf{H}_1 = \begin{pmatrix} 1 & 0 & 1 & 0 & 0 & 0 & 0 & 1 \\ 1 & 1 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 \end{pmatrix} \quad \text{and}$$

$$\mathbf{H}_2 = \begin{pmatrix} 0 & 0 & 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 1 & 1 & 0 & 0 & 0 & 0 & 1 & 0 \end{pmatrix}. \tag{7.78}$$

Hence, by this technique, we have decomposed the matrix $\mathbf{H}$ into $w$ *binary* matrices $\mathbf{H}_i$ of same dimension such that

$$\sum_{i=1}^{w} \mathbf{H}_i = \mathbf{H},$$

where the sum is taken over the natural numbers. We can also describe the edge-spreading technique in the following way: for every column $i$ of $\mathbf{H}$, partition (either randomly or according to some deterministic procedure) the set $\mathcal{N}(i)$ into $w$ pairwise disjoint subsets $\mathcal{N}_j(i)$ such that $\bigcup_{j=1}^{w} \mathcal{N}_j(i) = \mathcal{N}(i)$. Then use $\mathcal{N}_j(i)$ to set the *1* values of $\mathbf{H}_j$.

In the next step, we can construct the Tanner graph of the SC-LDPC code. For this, we start from our basic LDPC graph, and we first generate $w - 1$ additional copies of the VNs. This gives us $w$ groups of VNs, each corresponding to one of the $w$ colors. In the next step, we assign the edges of a single color to the corresponding group of VNs. For the colored graph in Fig. 7.17b, this leads to the graph shown in Fig. 7.18. Note that this graph corresponds to the parity-check matrix $(\mathbf{H}_2 \ \mathbf{H}_1)$. The Tanner graph of the final spatially coupled code with $w = 2$ is obtained by superimposing $L + 1$ of these Tanner graphs with an overlap of one spatial position (and combining the superimposed VNs into one new

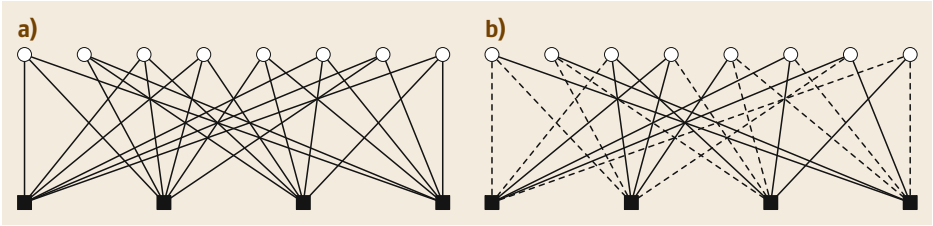


**Fig. 7.17a,b** Illustration of the edge-coloring principle for designing an SC-LDPC code from an LDPC code. **(a)** Tanner graph of a $(3, 6)$ LDPC code; **(b)** Tanner graph with edge coloring

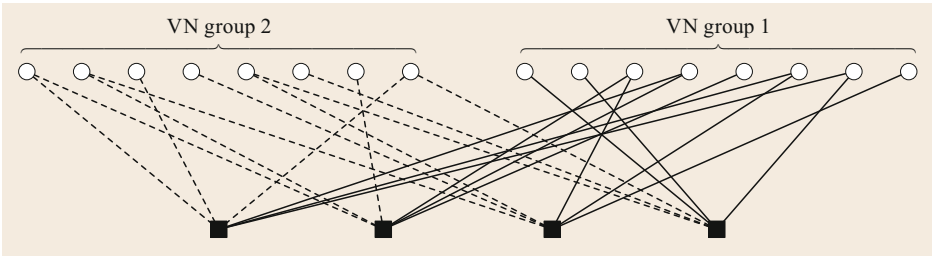


**Fig. 7.18** Tanner graph representation of the example code with parity-check matrix $(\mathbf{H}_2 \ \mathbf{H}_1)$ of (7.78)

VN) and then removing the first and last $n$ VNs together with their associated edges. For $L = 4$, this is illustrated in Fig. 7.19. We can clearly see the repetitive structure of the code. In the general case (i.e., for any $w \geq 2$), we superimpose $L + w - 1$ of the Tanner graphs corresponding to $(\mathbf{H}_w \ \cdots \ \mathbf{H}_2 \ \mathbf{H}_1)$ with an overlap of $w - 1$ spatial positions and then remove the first and last $(w - 1)n$ VNs with their associated edges.

Because of the edge-spreading technique, we have conserved $\boldsymbol{\delta}_\mathrm{v}$, i.e., the VN degree distribution (as well as $\boldsymbol{\delta}_\mathrm{c}$). The number of *1*s in each row of $\mathbf{H}^{[\mathrm{SC},w]}$ is the same everywhere as in the starting matrix $\mathbf{H}$, except in rows 1 to $(w - 1)m$ and $(L + 1)m$ to $(L + w - 1)m$, where we have lower degrees. These correspond to the lower-degree CNs at the boundaries of the Tanner graph (Fig. 7.19). This lack of *1*s, which is due to termination, is crucial for the outstanding performance of SC-LDPC codes [7.75, 159]. This leads to low-degree parity-check equations at the boundaries, which better protect the code bits at the boundaries. These, in turn, after decoding, protect their neighbors in a ripple- or wave-like fashion.

### Encoding Spatially Coupled Low-Density Parity-Check Codes

As an SC-LDPC code is in principle an LDPC code with a structured parity-check matrix, the technique introduced in Sect. 7.4.1, can be directly applied to $\mathbf{H}^{[\mathrm{SC},w]}$. However, as the length of $\mathbf{H}^{[\mathrm{SC},w]}$ is usually relatively long, it can be advantageous to take on the recurrent coding perspective of Fig. 7.16b. In this case, we merely apply the triangularization to $\mathbf{H}_1$ and then encode the different $\boldsymbol{c}_t$s on a batch-by-batch basis. The encoding procedure can be simplified significantly by slightly modifying the construction procedure and enforcing that $\mathbf{H}_1$ has the structure $\mathbf{H}_1 \triangleq (\tilde{\mathbf{H}}_1 \ \mathbf{I}_m)$, where $\mathbf{I}_m$ denotes the $m \times m$ identity matrix. In this case, systematic encoding (see Definition 7.8) of $\boldsymbol{u}_t$ into $\boldsymbol{c}_t = (\boldsymbol{u}_t \ \boldsymbol{p}_t)$ is accomplished by computing

$$\boldsymbol{p}_t^\top = \tilde{\mathbf{H}}_1 \boldsymbol{u}_t^\top + \sum_{i=2}^{w} \mathbf{H}_i \boldsymbol{c}_{t-i+1}^\top \tag{7.79}$$

during the first $L - w + 1$ batches with the initialization $\boldsymbol{c}_0 = \boldsymbol{c}_{-1} = \cdots = \mathbf{0}$. The last batch(es) require special

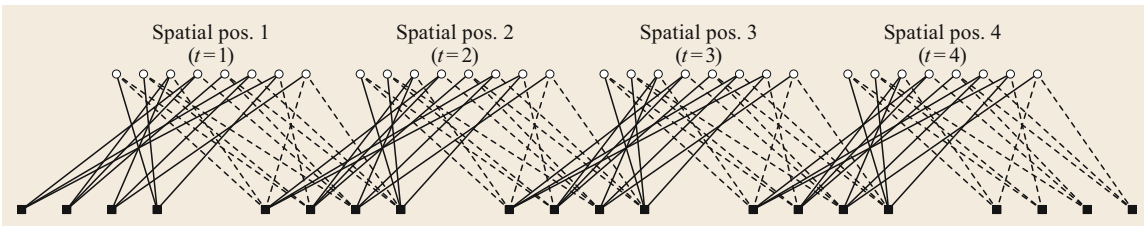


**Fig. 7.19** Tanner graph representation of the example code with parity-check matrix $(\mathbf{H}_2 \ \mathbf{H}_1)$ of (7.78), with $L = 4$

treatment to fulfill the termination constraints, such that $\mathbf{H}^{[\mathrm{SC},w]}(\boldsymbol{c}^{[\mathrm{SC}]})^\top = 0$. We require up to $m(w-1)$ extra parity bits to fulfill these constraints. These are usually accommodated in the last batch or spread over the final $w-1$ batches, at a cost of a smaller number of information bits in these batches. A detailed, in-depth treatment of the encoder and related hardware architecture can be found in [7.71, 160, 161].

### Decoding Spatially Coupled Low-Density Parity-Check Codes

In principle, an SC-LDPC code is merely an LDPC code with a parity-check matrix $\mathbf{H}$ structured in a particular way. Hence, decoding SC-LDPC codes is possible in a straightforward manner using the decoders given in Sect. 7.4.1. However, the decoding can be greatly simplified by exploiting the structure of the parity-check matrix and the decoding mechanics of such codes using a so-called windowed decoder, which operates on only a part of the codeword [7.77]. In the following, we first describe the conventional layered windowed decoder and then an extension that can improve the decoding performance without increasing the decoding latency.

*Conventional Windowed Decoder.* The conventional windowed decoder extends the flooding or layered decoders presented in Sect. 7.4.1 by some extra data flow and message handling to account for the memory of the code. We can define a decoding window size $W_\mathrm{D}$ and extract a portion $\mathbf{H}_\mathrm{W}$ of $m_\mathrm{b}W_\mathrm{D}$ rows of $\mathbf{H}^{[\mathrm{SC},w]}$ (with the corresponding nonzero columns) of size $\dim(\mathbf{H}_\mathrm{W}) = m_\mathrm{b}W_\mathrm{D} \times n_\mathrm{b}(W_\mathrm{D}+w-1)$ that has the form

$$\mathbf{H}_\mathrm{W} \triangleq \begin{pmatrix} \mathbf{H}_w & \mathbf{H}_{w-1} & \cdots & \mathbf{H}_1 & & \\ & \mathbf{H}_w & \mathbf{H}_{w-1} & \cdots & \mathbf{H}_1 & \\ & & \ddots & \ddots & \cdots & \ddots \\ & & & \mathbf{H}_w & \mathbf{H}_{w-1} & \cdots & \mathbf{H}_1 \end{pmatrix}. \tag{7.80}$$

We focus here only on the layered decoder, as it has most practical relevance due to its fast convergence. We initialize an a posteriori memory of size $n_\mathrm{b}(W_\mathrm{D}+w-1)$ as follows: $\mu_1 = \mu_2 = \cdots = \mu_{n_\mathrm{b}(W_\mathrm{D}+w-1)} = +\infty$. In practice, a sufficiently large constant (e.g., the largest value representable with the fixed-point representation) is used instead of $+\infty$. As in the layered decoder, we initialize an extrinsic memory as $e_{j,i} = 0$, $\forall j \in \{1,\ldots,m_\mathrm{b}W_\mathrm{D}\}$, $i \in \mathcal{M}_\mathrm{W}(j)$, where $\mathcal{M}_\mathrm{W}(j) = \{k : h_{\mathrm{W},j,k} \neq 0\}$ is the set of all nonzero entries of row $j$ of $\mathbf{H}_\mathrm{W}$.

The windowed decoder then processes in each step a batch of $n_\mathrm{b}$ received values $\boldsymbol{y}_t$ that correspond to the transmitted batch $\boldsymbol{c}_t$, where $t$ also denotes the spatial position. At the start of a codeword $\boldsymbol{c}^{[\mathrm{SC}]}$, we set $t = 1$. The windowed decoder is described as follows:

1. (*Receive and shift*) Upon reception of a new batch of $n$ received values $\boldsymbol{y}_t = (y_{t,1},\ldots,y_{t,n})$ that correspond to the batch $\boldsymbol{c}_t$ of $\boldsymbol{c}^{[\mathrm{SC}]}$, shift the a posteriori memory by $n$ positions and append the newly received LLRs $l_{t,i}$, i.e.,

$$\begin{aligned} &\mu_i \leftarrow \mu_{i+n_\mathrm{b}}, \quad \text{for } i \text{ from 1 to } (W_\mathrm{D}+w-2)n_\mathrm{b} \\ &\mu_{(W_\mathrm{D}+w-2)n_\mathrm{b}+i} \leftarrow l_{t,i}, \quad \text{for } i \text{ from 1 to } n_\mathrm{b}\,. \end{aligned}$$

   Note that the decoder must run even when $t > L$ until all batches of the codeword have been decoded. If $t > L$, there are no newly received values, and we define $l_{t,i} = +\infty$ for $t > L$ and $i \in \{1,\ldots,n_\mathrm{b}\}$. Similarly, we shift the extrinsic memory and initialize the unused part with zeros,

$$\begin{aligned} &e_{j,i} \leftarrow e_{j+m_\mathrm{b},i}, \quad \text{for } j \text{ from 1 to } (W_\mathrm{D}-1)m_\mathrm{b}, \forall i \\ &e_{(W_\mathrm{D}-1)m_\mathrm{b}+j,i} \leftarrow 0, \quad \text{for } j \text{ from 1 to } m_\mathrm{b}, \forall i\,. \end{aligned}$$

2. (*Decoding*) Run the layered decoder as described in Sect. 7.4.1, *Decoding Low-Density Parity-Check Codes* using $\mathbf{H}_\mathrm{W}$ as parity-check matrix for a predetermined number of iterations $I$ with extrinsic memory $e_{j,i}$ and a posteriori memory $\mu_i$.
3. (*Decision*) Recover the estimated code bits

$$\hat{c}_{t-W_\mathrm{D}-w+2,i} = \frac{1}{2}[1-\mathrm{sign}(\mu_i)], \quad \forall i \in \{1,\ldots,n_\mathrm{b}\} \tag{7.81}$$

   of the batch corresponding to $\boldsymbol{c}_{t-W_\mathrm{D}-w+2}$ of $\boldsymbol{c}^{[\mathrm{SC}]}$. Note the additional delay of $W_\mathrm{D}+w-2$ introduced by the windowed decoder, which adds extra latency to the decoding process.

Note that the number of effective iterations per spatial position is $I(W_\mathrm{D}+w-1)$, i.e., in order to compare SC-LDPC codes with conventional (block) LDPC codes, we need to choose the effective number of iterations accordingly. Also note that the windowed decoder introduces a decoding *delay* of $W_\mathrm{D}+w-2$ batches and hence potentially increases the decoding latency with respect to that of a conventional LDPC code, unless $n_\mathrm{b}$ is chosen to be significantly smaller than the length $n$ of the LDPC code to compensate for this delay. Also note that the decoder will not output meaningful information in the first $W_\mathrm{D}+w-2$ batches and hence must run following the reception of all $L$ transmitted batches for an additional $W_\mathrm{D}+w-2$ batches until all bits have been decoded. These extra executions must be addressed by

the decoder and require attention inside an FPGA or ASIC. However, as the windowed decoder essentially wraps around a conventional LDPC decoder, the effort of a very-large-scale integration VLSI implementation is potentially simple.

*Multi-Engine Windowed Decoder.* Usually $W_D > w$, and hence, as a decoding engine operates on some rows of $\mathbf{H}_W$, only some of the corresponding memory locations are updated, while a significantly large part of $\mathbf{H}_W$ is not considered due to the particular banded structure of $\mathbf{H}_W$. For example, all rows of $\mathbf{H}_W$ that are at least $wm_b$ rows apart do not share common code bits and could hence be treated independently in parallel inside the decoder. For this reason we introduce a multi-engine layered decoder that consists of $E$ engines that operate independently in parallel [7.83]. The only modification is step 2 of the windowed decoder. While the first engine processes the rows from 1 to $m_b W_D$ in that order, the second engine starts with an offset $O_2 \geq wm_b$ and operates on the sequence of rows $(O_2, \ldots m_b W_D, 1, \ldots, O_2 - 1)$, i.e., it wraps around the matrix. In general, engine $i$ operates on the sequence of rows $(O_i, \ldots, m_b W_D, 1, \ldots, O_i - 1)$, and we define $O_1 = 1$. The offsets need to fulfill the following condition: $|O_i - O_j| \geq wm_b$, $\forall (i,j) \in \{1, \ldots, E\}^2$, $i \neq j$. By the pigeonhole principle, the number of engines is hence upper bounded as $E \leq \lfloor W_D / w \rfloor$, where $\lfloor x \rfloor$ denotes the largest integer smaller than or equal to $x$.

### Optimization of Spatially Coupled Low-Density Parity-Check Codes

In contrast to LDPC codes, which are well understood and have now commonly accepted design rules, the design of high-performing SC-LDPC codes is still part of active research efforts. In this section, we highlight some of these activities and recent results that seem promising for the design and evaluation of future close-to-capacity SC-LDPC codes.

Despite their similarity to LDPC codes, SC-LDPC codes have significantly different error correction behavior. This behavior was rigorously analyzed in [7.75, 76, 159], where it was shown that decoding behaves like a *wave* that propagates through the codeword, which is also in line with the windowed decoding mechanism. To optimize codes during construction, knowing the speed of this wave is crucial. An analysis of the speed of this wave together with upper and lower bounds was provided in [7.162–164]. This analysis was used in [7.165] to show that we can design SC-LDPC codes with irregular VN degrees that lead to performance improvements when decoded with a complexity-constrained decoder.

While the asymptotic performance of SC-LDPC codes is now well understood, the influence of the various parameters on the practical, non-asymptotic decoding performance is much less clear. Some attempts to better understand the influence on the different parameters are given in [7.166, 167] which could help to design codes in the future. It turns out that for highly optimized codes, $L$ should be small. In order not to reduce the rate given by (7.77) too much, $w$ should be kept small as well, ideally $w \in \{2, 3\}$, to keep the rate loss and complexity and latency of windowed decoding small. Such codes have been called *weakly coupled* in [7.165]. To construct good codes in this case, *non-uniformly* coupled SC-LDPC codes were constructed in [7.158], which have both superior decoding performance and reduced rate loss.

While selecting a small coupling width $w$ can have its advantages, the choice of a larger $w$ together with very sparse matrices $\mathbf{H}_i$ can also be useful in some applications. In [7.165], the latter case was called *strongly coupled*. Although the windowed decoder needs to cover a potentially longer decoding window, this is compensated for by the fact that we can have a shorter $n_b$ and that the sub-matrices $\mathbf{H}_i$ are very sparse, limiting the effective amount of operations to be done. Constructions for such strongly coupled codes, which closely resemble traditional convolutional codes with large syndrome-former memory, are given for instance in [7.168–170].

As already discussed, one of the main challenges for implementing SC-LDPC codes in practice is the rate loss due to termination. The rate loss, which scales with $w$, can be made arbitrarily small by increasing $L$; however, in some cases we want to avoid a large $L$, as it can worsen the finite-length performance of SC-LDPC codes [7.166]. Nevertheless, in some cases we can let $L$ go to infinity and still achieve gains compared with LDPC codes, as we have demonstrated in [7.171]. For a fixed $L$, there exist multiple proposals for mitigating the rate loss [7.172, 173], which however often introduce extra structure at the boundaries. Such extra structure is usually undesired, as it complicates the decoder data flow. Other approaches use tail-biting codes (without termination) and mimic the termination (summoning the positive effects of termination) by shortening [7.155, 157, 174], energy shaping [7.175], using intelligent interleaving together with the properties of higher-order modulation formats [7.156, 176–178], or iterative demapping and equalization [7.179], just to name a few.

Another line of research is the optimization of the windowed decoder of SC-LDPC codes, which is itself not free of issues. Some of these issues are highlighted for instance in [7.166, 180]. In systems with heavy complexity constraints, the scheduling of the operations inside the decoder can lead to some performance im-

provements [7.181]. The windowed decoder itself can lead to unrecoverable burst errors, significantly degrading the system performance. These burst errors can be addressed as well by intelligent scheduling of the decoder operations [7.182]. A general introduction to SC codes and an overview of some further research problems is given in [7.183].

### 7.4.3 Polar Codes

In addition to spatially coupled codes, another coding scheme has emerged in the past few years that is asymptotically capacity-achieving and has appealing encoding and decoding complexity. This scheme, called *polar coding*, was invented by *Arıkan* in 2008 [7.65, 66] and was the first low-complexity coding scheme that could (provably) achieve the capacity over binary discrete memoryless symmetric channels (DMSCs), nonbinary DMSCs [7.184], and asymmetric channels [7.185]. In fact, polar codes had been proposed already in 2002 by *Norbert Stolte* [7.186] but did not receive much attention until Arıkan proved that they were capacity-achieving. A polar code encodes a vector $\tilde{\boldsymbol{u}}$ of length $n = 2^\eta$, containing the $k$ information bits in specific, predetermined positions and (known) frozen values otherwise, by $\boldsymbol{c} = \tilde{\boldsymbol{u}}\mathbf{F} = \tilde{\boldsymbol{u}} \left(\begin{smallmatrix} 1 & 0 \\ 1 & 1 \end{smallmatrix}\right)^{\otimes\eta}$, where $\otimes\eta$ denotes $\eta$ successive applications of the Kronecker product. If the positions of the $k$ information bits are carefully chosen, then after decoding with a *successive cancellation* decoder or variants thereof, these positions are unaffected by channel noise. More precisely, each component of $\tilde{\boldsymbol{u}}$ *sees* its own equivalent sub-channel, so that in total there are $2^\eta$ sub-channels. As $\eta$ tends to infinity, the error probability of each sub-channel converges to zero (a *good* sub-channel) or one-half (a *bad* sub-channel). This phenomenon is referred to as *channel polarization*. The $k$ information bits can be reliably transmitted over the $k$ good sub-channels. Arıkan showed that the fraction of good sub-channels converges to the channel capacity under successive cancellation decoding. Figure 7.20 shows an example of a rate $R = 3/4$ polar encoder for a short toy code with $n = 8$ ($\eta = 3$ and hence $k = 6$). An offline procedure, as described for instance in [7.187], determines the $k = 6$ *good* sub-channels over which we transmit information, and the frozen bits, which we set to a predetermined value, e.g., *0*. Note that the codeword $\boldsymbol{c}$ is *not* systematic by default. Polar codes can, however, also be rendered systematic following the procedure given in [7.188].

The successive cancellation decoder of polar codes consists of relatively simple binary operations and is of complexity $O(n \log n)$. The elegance of the construction has made polar codes an attractive choice in research with many applications and scenarios, such as 5G. An early application to fiber-optic communications has been given in [7.82, 189]. However, there are multiple areas where polar codes can be further improved. Extremely long block lengths $n$ are required to achieve low error probabilities with polar codes [7.66] and simple successive cancellation decoding. Recently, list decoding [7.67] in conjunction with a concatenated cyclic redundancy check CRC code was proposed to improve the error performance for finite code lengths, albeit at a cost of increased receiver complexity. Furthermore, the original successive cancellation decoder proposed by *Arıkan* [7.66] is not directly suited for parallelization, which is however mandatory if this class of codes is to be used in fiber-optic communications. Recent research efforts are aimed at studying new concepts for building hardware-efficient, parallelizable decoders [7.69, 70, 190, 191], which is still an active area of research. A promising line of research is the use of the sum-product algorithm developed for LDPC for polar codes, enabling efficient parallelization [7.192].

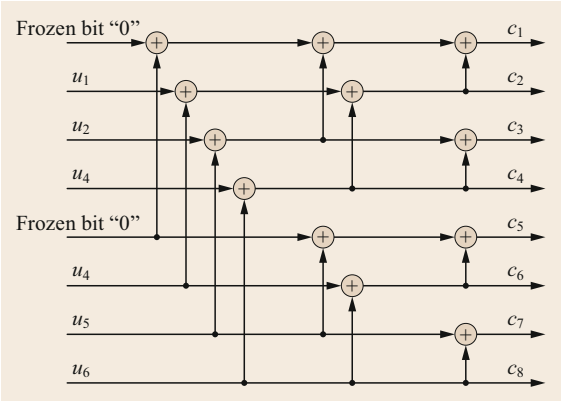


**Fig. 7.20** Threefold application ($\eta = 3$) of the basic channel transform

### 7.4.4 Other Coding Schemes

In addition to LDPC, SC-LDPC, and polar codes, many other coding schemes exist for SDD. However, most of these have not found widespread application in optical communications. Perhaps the most prominent one is SDD of TPCs or concatenated BCH codes, which we will briefly discuss in the next section. In the 3G and 4G mobile wireless communication standards, turbo codes [7.193] (which are parallely concatenated codes based on convolutional component codes) are still the dominant error-correcting codes. However, they have found no application so far in optical communications. They suffer from the disadvantage of a non-negligible error floor and little to no available decoder implementations that can operate at the speeds required in optical

communications. In the upcoming 5G wireless communication standard, turbo codes will be at least partly superseded by LDPC codes (for some data channels) and polar codes (for some control channels) [7.68].

### 7.4.5 Performance Curves

#### Performance of Low-Density Parity-Check Codes

To show the potential of LDPC codes with the simple SDD algorithm described above, we generate two regular QC LDPC codes with parameters $d_v = 3$ and $d_v = 4$, and we fix $d_c$ to obtain codes of rate $4/5$ (OH = 25%). We generate small base matrices of size $n = 300$ using an algorithm similar to Gallager's algorithm and then use lifting with cyclically shifted identity matrices of size $128 \times 128$. We have selected the shifts to maximize the girth parameter of the code by avoiding *Fossorier*'s conditions for low-girth codes [7.132]. In Fig. 7.21, we show simulation results with the non-simplified CN update rule and the normalized min-sum decoding rule (with $\alpha = 3/4$) after transmission over a binary-input AWGN channel. We can see that the code with $d_v = 3$ outperforms the code with $d_v = 4$, and the performance loss due to scaled min-sum decoding is almost negligible. For $d_v = 4$, the performance loss due to scaled min-sum decoding is more pronounced, which is due to a mismatched choice of the scaling factor $\alpha$. The increasing color scales in Fig. 7.21 indicate the number of iterations (1, 2, 3, 5, 10 and 30). We can see that after only 10 decoding iterations, we have achieved a very good performance, and increasing the number of iterations to 30 yields additional gains of only 0.1 dB, which may not be worth the effort.

Figure 7.22 illustrates the advantage of the layered decoder compared with the flooding decoder. We use the $(d_v = 3, d_c = 15)$ regular LDPC code at a channel quality of $E_b/N_0 = 2.9$ dB. We plot the post-FEC BER as a function of the iterations that are carried out inside the decoder. We can see that the BER decreases significantly faster in the case of the layered decoder, and when targeting a specific post-FEC BER, the layered decoder roughly halves the number of required iterations. Especially in high-speed fiber-optic communications, this can lead to dramatic reductions in VLSI complexity, as only a relatively small number of iterations need to be carried out. The convergence speed can be further improved by combining two layered decoders with different schedules, as highlighted in [7.194], at a cost of increased hardware complexity.

We consider as a second simulation example LDPC codes for transmission over the BSC. This case occurs for instance in systems without ADC at the receiver, using a simple thresholding device. Other examples are high-speed communications with different dedicated digital signal-processing (DSP) and FEC chips. In this case, the chip-to-chip interface may not be able to transfer the quasi-continuous decoder input LLRs to the FEC chip, and heavy quantization is necessary. Although the channel output undergoes a decision, we can still carry out SDD by passing *soft* messages inside the decoder and exploiting the channel statistics. Note that for the

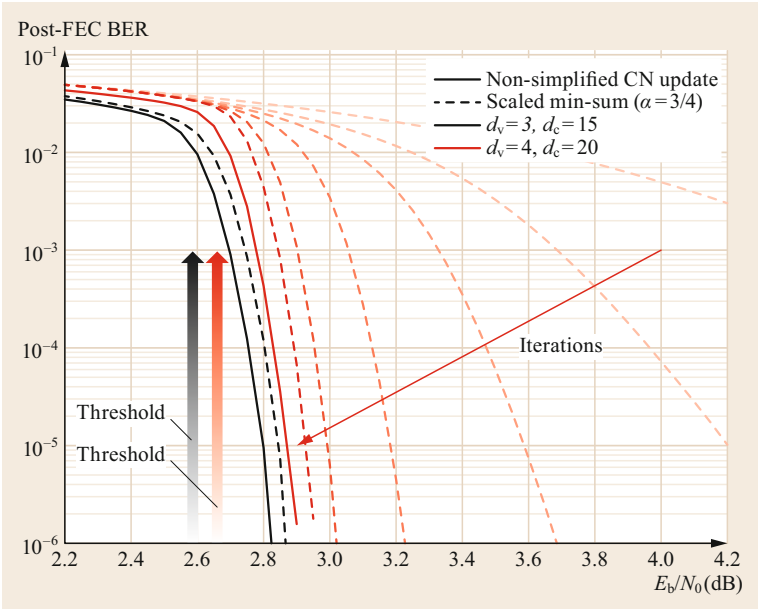


**Fig. 7.21** Simulation of two regular LDPC codes of rate $R = 4/5$ (OH = 25%) with parameters ($d_v = 3$, $d_c = 15$) (*black lines*) and ($d_v = 4$, $d_c = 20$) (*red lines*) with the full update rule (*solid lines*) and scaled min-sum (*dashed lines*) with layered decoding and 30 decoding iterations

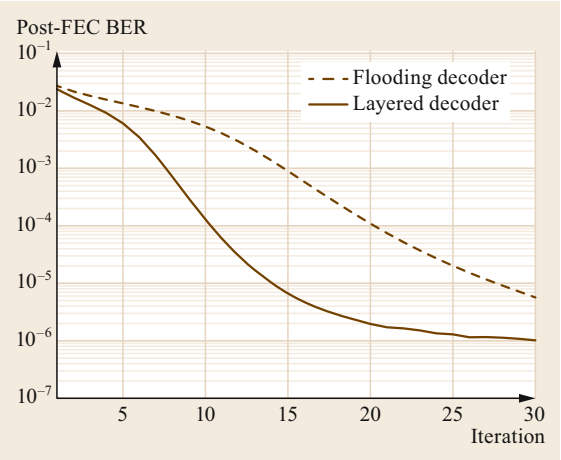


**Fig. 7.22** Comparison of flooding decoder and layered decoder with min-sum update rule ($\alpha = 0.875$) for a regular LDPC code with parameters $d_v = 3$ and $d_c = 15$ ($R = 4/5$, 25% OH) at a channel quality of $E_b/N_0 = 2.9$ dB

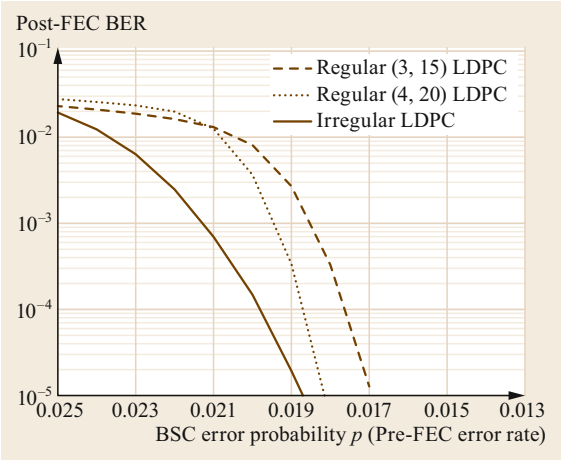


**Fig. 7.23** Simulation of three LDPC codes of rate $R = 4/5$ (25% OH) with scaled min-sum update rule ($\alpha = 3/4$), layered decoding, and 30 decoding iterations

BSC, the achievable rates $R_{\text{SDD-BW}}$ and $R_{\text{HDD-BW}}$ coincide. We compute the channel-related LLRs as

$$l_i = \log \frac{p_{Y_i|X_i}(y_i|+1)}{p_{Y_i|X_i}(y_i|-1)} = y_i \log \frac{1-p}{p} \triangleq y_i L_c^{[\text{BSC}]}, \tag{7.82}$$

where we assume BPSK modulation $x_i = (-1)^{c_i}$ and received values $y_i \in \{\pm 1\}$. The error probability of the BSC is given by $p$ and is known at the receiver. We consider a normalized min-sum decoder with $\alpha = 3/4$. Figure 7.23 shows simulation results for three different codes of rate $R = 4/5$: two regular LDPC codes with parameters $d_v \in \{3, 4\}$ and one irregular LDPC code with $\delta_{v,3} = 4/5$, $\delta_{v,10} = 1/5$, and $d_c = 22$ ($\delta_{c,22} = 1$). We keep $\delta_{v,2} = 0$ in order not to increase the error floor. We can see that in this case, the code with $d_v = 4$ outperforms the code with $d_v = 3$ (contrary to the case of the AWGN channel shown in Fig. 7.21). The irregular code enables further coding gains, especially at post-FEC BERs above $10^{-5}$, but the slope of the BER curve is less steep than that of the BER for the two regular codes.

### Comparison of Low-Density Parity-Check and Polar Codes

In [7.165], LDPC codes, SC-LDPC codes, and polar codes with successive cancellation decoding were compared, and it was concluded that as of today, SC-LDPC codes are the most promising for high-speed optical transponders. In Fig. 7.24, we compare the performance of a ($d_v = 3, d_c = 15$) regular LDPC code of length 38 400 and a polar code of length $n = 2^{15} = 32\,768$, both of rate $R = 4/5$. Note that we use the full update equation (7.72) for both cases to have a fair comparison. We use the successive cancellation list decoder described in [7.67] with list sizes of $L_S = 4$ and $L_S = 32$ and additionally concatenate the polar code with a short CRC (adding 32 parity bits), which is used to improve the selection of candidate codewords kept in the list. We can see that with list-based decoding, polar codes now have comparable performance as LDPC codes. However, the slope of the polar code is significantly less steep than that of the LDPC code. Hence, if low BERs of around $10^{-15}$ are targeted, LDPC codes will still show better performance. Polar codes may have an advantage for low-error-rate applications due to the absence of error floors under successive cancellation decoding [7.195]. In addition, note that the complexity of the list decoder, required for achieving good performance, scales linearly with the list size $L_S$. Furthermore, the baseline complexity of the decoder is significantly larger than that of an LDPC decoder. Hence, polar codes may become attractive in the future when highly parallel and efficient low-complexity decoders become available.

### Performance of Spatially Coupled Low-Density Parity-Check Codes

In this section, we compare a few constructions of SC-LDPC codes. The possibilities for optimizing SC-LDPC codes are vast, and depending on the application, different setups might be useful. We again summarize the parameters of SC-LDPC codes and their influence on performance:

- The degree distribution will affect the performance. Although regular codes already achieve outstand-

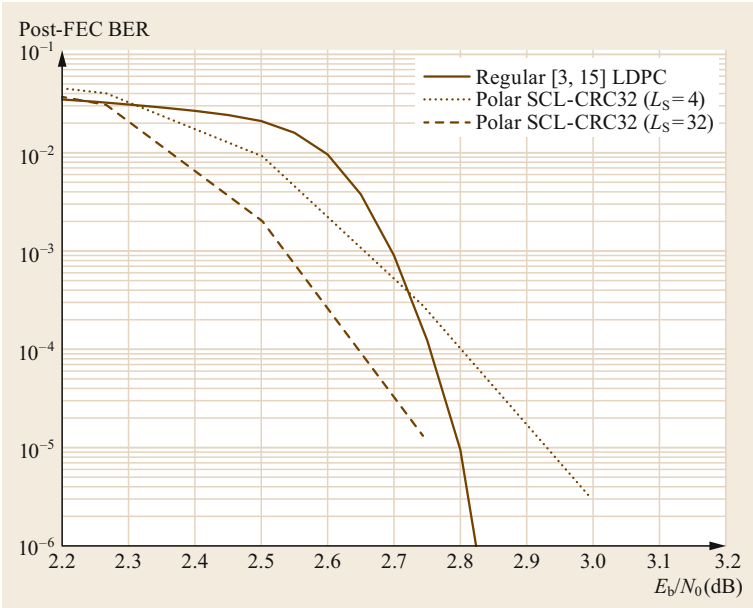


**Fig. 7.24** Comparison of LDPC codes with polar codes with successive cancellation list decoding

ing performance, some improvements are expected with irregular codes, especially if the decoder complexity is constrained, as shown in [7.83].

- The coupling width $w$ has a direct influence on the performance as well. In [7.83], a distinction between weakly coupled codes (small $w$, usually $w < 5$) and strongly coupled codes (large $w$, often $w > 5$) was noted. Both schemes may have their advantages and disadvantages.
- The replication factor $L$ has an influence on the rate loss of the code and also an influence on the error probability of the windowed decoder as shown in [7.166, 180]. However, its influence is not as obvious as the influence of the other parameters.
- The parameters of the windowed decoder have a significant influence on the performance. The complexity of the windowed decoder equals $I\,(W_D + w - 1)$. However, when $W_D$ is chosen that is too small, even a large number of iterations will not lead to good decoding performance. If $W_D$ is too large, we may carry out too many unnecessary operations.

In the following, we will highlight some of these parameter trade-offs with simulation examples. In all simulations we fix $L = 100$. In a first example, we fix the (approximate) decoding complexity to $W_D\,I = 10$, corresponding to an equivalent of roughly 10 layered decoding iterations of conventional LDPC codes. We construct an SC-LDPC of rate $R = 4/5$ using the spreading technique explained in Sect. 7.4.2. We start from a QC-LDPC base matrix $\mathbf{A}$ of size $60 \times 300$ and fix the lifting parameter to $S = 32$. We apply the spreading construction to the base matrix. Hence, the sub-matrices $\mathbf{H}_i$ are of size $1920 \times 9600$. We show the simulation results in Fig. 7.25. For the configuration with $W_D = 5$ and $I = 2$ (Fig. 7.25a.), the code with $w = 2$ yields the best performance. When we switch the configuration to $W_D = 10$ and $I = 1$ (Fig. 7.25b), we can see that the code with $d_v = 4$ and $w = 4$ has the best performance and already outperforms the conventional LDPC code by about 0.1 dB. As the code with $d_v = 4$ gives superior performance, we will no longer consider the scheme with $d_v = 3$ in the following.

In Fig. 7.26, we consider the performance of codes with varying windowed decoder complexity. We consider complexities of $W_D\,I \in \{10, 20, 40, 80\}$ and compare the codes for different coupling factors $w$. We can see that the decoder performance does not saturate early, but keeps improving if we increase the complexity. We picked the window configuration among a few tested such that the coding gains are maximized. Furthermore, we see that, as observed already before, increasing $w$ leads to significant performance improvements. In particular, the code with $w = 6$ improves about 0.5 dB with respect to the LDPC code of Fig. 7.24 at a BER of $10^{-6}$, and approaches the achievable rate of the channel by about 0.3 dB. Considering that the slope of the BER curve is much steeper, the additional coding gain at a low BER of $10^{-15}$ is even larger. Hence, SC-LDPC codes are future-proof in the sense that we can use them today with low complexity, but with only small coding gains. In the future, when high-performing circuits are available, we can spend more complexity to achieve higher NCGs. In the next section, we look at the performance at low BERs.

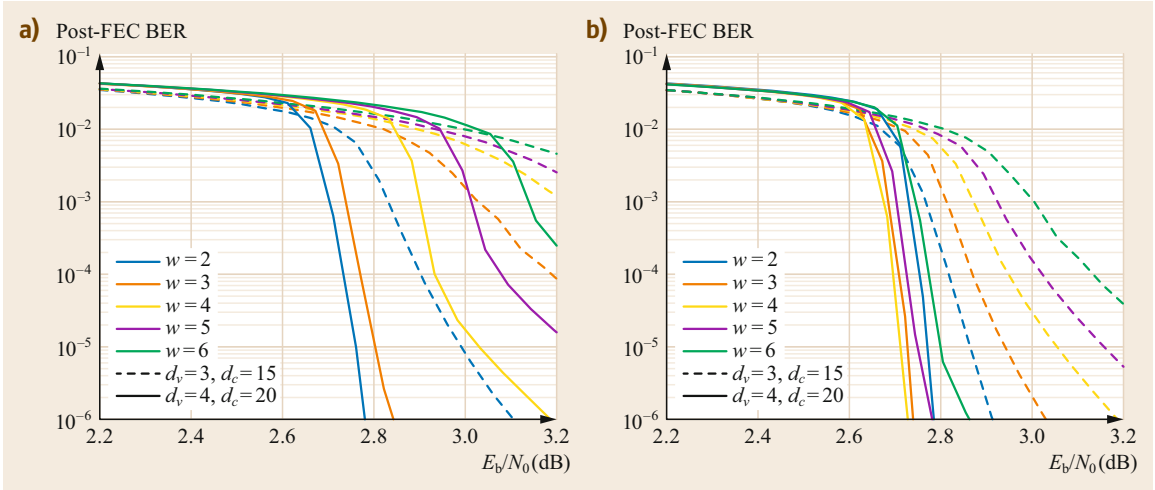


**Fig. 7.25a,b** Performance simulation of regular SC-LDPC codes of rate $R = 4/5$ with windowed decoding (full update rule) for two window configurations. **(a)** $W_D = 5$ and $I = 2$; **(b)** $W_D = 10$ and $I = 1$

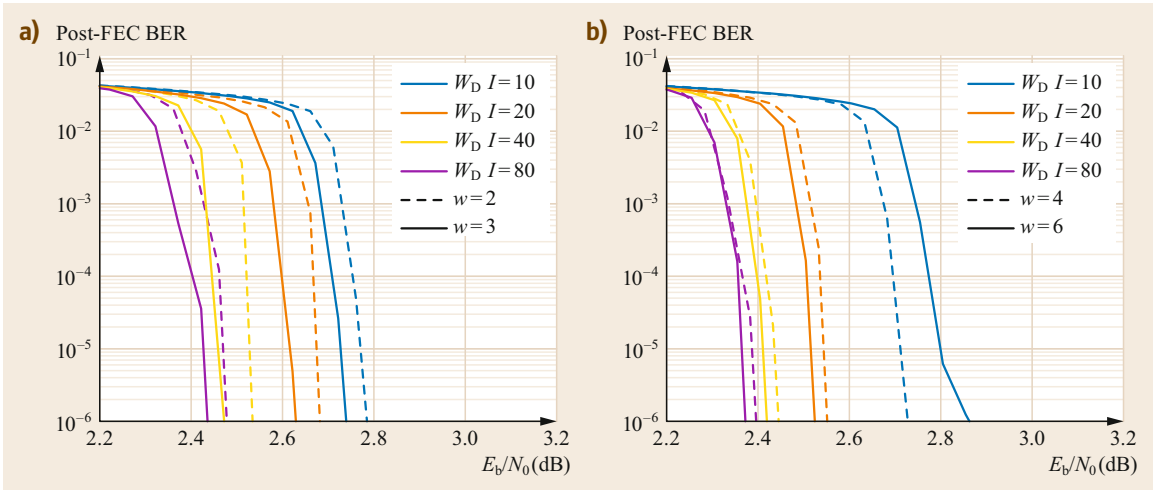


**Fig. 7.26a,b** Performance simulation of regular SC-LDPC codes of rate $R = 4/5$ with windowed decoding (full update rule) for four windowed decoding configurations of increasing complexity: $W_D = 10$ and $I = 1$, $W_D = 20$ and $I = 1$, $W_D = 20$ and $I = 2$, $W_D = 20$ and $I = 4$. **(a)** $w = 2$ and $w = 3$; **(b)** $w = 4$ and $w = 6$

### FPGA-Based Evaluation of Decoding Performance of Spatially Coupled Low-Density Parity-Check Codes

A drawback of LDPC and SC-LDPC codes is that no analytic expression exists for the error performance, and the FEC designer must resort to Monte Carlo simulations. As optical communication systems usually require extremely low post-FEC error rates in the range of $10^{-15}$, we require either extremely large compute clusters or FPGAs.

In order to verify the performance of some LDPC and SC-LDPC codes at low BERs, we use an FPGA platform, whose high-level diagram is illustrated in Fig. 7.27 and which is described in detail in [7.165]. This platform is similar to other platforms reported in the literature [7.84, 196, 197] and consists of three parts: a Gaussian noise generator, an FEC decoder, and an error-detecting circuit. The Gaussian noise generator is built upon a Tausworthe pseudo-random number generator with a sequence length of $\approx 10^{88}$, followed by a Box–Muller circuit transforming two uniformly distributed random numbers into two Gaussian-distributed numbers [7.198]. The output of the Gaussian noise generator is multiplied with a weighting factor in order to adjust $N_0$ and added to the all-zero codeword ($\boldsymbol{x} = (+1, \ldots, +1)$), yielding LLRs, which are then fed to

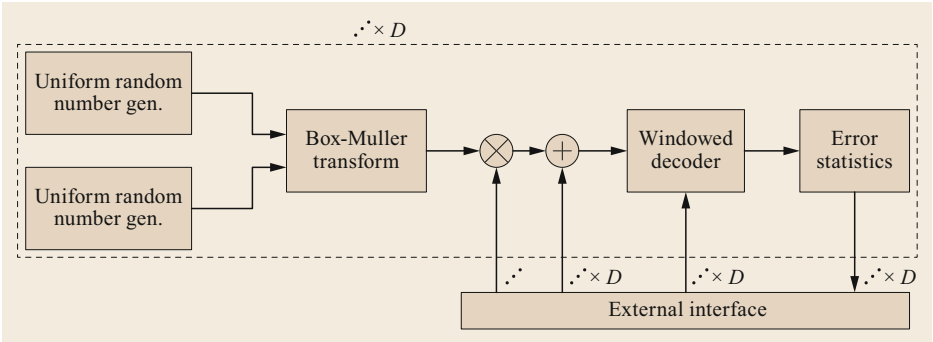


**Fig. 7.27** High-level schematic of the FPGA evaluation platform

the LDPC decoder after quantization to 15 levels. The LDPC decoder is based on the layered decoding algorithm and uses a scaled min-sum CN computation rule.

The windowed decoder carries out $I$ layered decoding iterations as described in Sect. 7.4.2. The effective number of iterations, i.e., the number of updates of each batch, thus corresponds to $I(W_\mathrm{D} + w - 1)$. For simplicity, we set $I = 1$ in most examples such that the window length corresponds roughly to the number of iterations in a conventional decoder. The output of the LDPC decoder is connected to the BER evaluation unit, which counts the bit errors and reports the error positions. A single Virtex-7 FPGA is able to emulate a block LDPC code with 12 decoding iterations at a rate of about 5 Gbit/s and an SC-LDPC code with $w = 3$, $W_\mathrm{D} = 13$, and a single iteration ($I = 1$) at approximately 1.5 Gbit/s. This lower rate is mainly due to increased memory requirements of the windowed decoder compared with the block decoder, which only requires a single a posteriori memory of size $n$, while the windowed decoder requires $(W_\mathrm{D} + w - 1)n_\mathrm{b}$ memory blocks. Note that in an ASIC implementation with an iteration-unrolled decoder, these numbers will be approximately equal.

The decoder we implemented uses a decoding window of size $W_\mathrm{D} = 13$, carries out a configurable number of iterations $I$, and can use either a single engine ($E = 1$) or two decoding engines ($E = 2$), in which case the offset of the second engine is $O_2 = 6m_\mathrm{b} + 1$. The decoder uses a scaled min-sum update rule with either $\alpha = 0.75$ or $\alpha = 0.875$ (depending on the code, different $\alpha$ will be better). We evaluate the performance of several coding schemes of rate $R = 4/5$ (OH = 25%). We consider a $(d_\mathrm{v} = 3, d_\mathrm{c} = 15)$ regular LDPC code as reference. This code is a QC code constructed from cyclic permutation matrices of size $32 \times 32$ and has girth 10 and length $n = 31\,200$.

In addition to the block code, we also compare two SC-LDPC codes, where we select $L \to \infty$ to avoid any termination effects and to simplify the decoder data flow. The performance of such a code is an upper bound of the BER performance of a code with finite $L$, as decreasing $L$ will only improve the BER of the code (note that it may not necessarily improve the NCG as the rate $R$ of the code decreases and hence $E_\mathrm{b}/N_0$ changes). We consider two codes in the simulation:

- □ SC-LDPC code A (marker □) is an irregular code with coupling width $w$, VN degree distribution $\boldsymbol{\delta}_\mathrm{v} = (0, 0, 4/5, 0, 0, 1/5)$, and regular $d_\mathrm{c} = 18$. This degree distribution has been optimized to maximize the decoding velocity at a target BER of $10^{-6}$, as detailed in [7.165]. The batch size is $n_\mathrm{b} = 7500$ ($\dim(\mathbf{H}_i) = 1500 \times 7500$).
- ◊ SC-LDPC code B (marker ◊) is a regular code with $d_\mathrm{v} = 4$, $d_\mathrm{c} = 20$, coupling width $w = 2$, and batch size $n_\mathrm{b} = 7500$, with $m_\mathrm{b} = 1500$.

Both SC-LDPC codes are QC codes constructed from cyclic permutation matrices of size $32 \times 32$. Note that the batch size $n_\mathrm{b}$ of the spatially coupled codes is significantly smaller than the length $n$ of the LDPC code, which may simplify the decoder implementation, especially the routing network.

The simulation results are shown in Fig. 7.28. As expected, the LDPC code performs quite well, albeit slightly worse than the code shown in Fig. 7.21. This is because here, the channel output and the messages inside the decoder have been quantized, which leads to a small performance penalty of $\approx 0.12$ dB. First we look at the case $I = 1$ and $E = 1$. The LDPC code is decoded with $I = 13$ iterations, and hence the number of operations per bit is equivalent for both cases. We can see that both SC-LDPC codes outperform the LDPC code. SC-LDPC code A, which has been optimized for convergence speed, shows a very early start of the waterfall region. However, the slope of the BER curve starts to decrease slowly, such that the performance at low BERs is relatively poor and is expected to cross the BER curve of the block LDPC code. The situation is reversed for SC-LDPC code B, which exhibits a late start of the waterfall curve, which then drops exceptionally

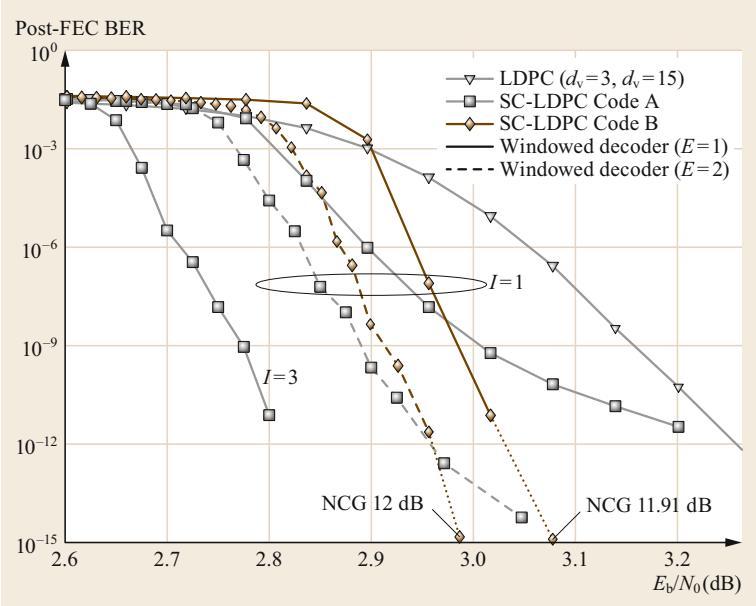


**Fig. 7.28** Decoding results of the FPGA platform for two different codes and two setups of the decoder

quickly. We did not observe any error patterns for the next $E_b/N_0$ point (3.08 dB) of the curve after simulation of $7.9\times10^{14}$ bits, so by extrapolating the BER curve, we may conjecture that SC-LDPC code B has an NCG of 11.91 dB at a post-FEC BER of $10^{-15}$. This NCG is achievable with a modest number of $(W_D + w - 1) = 14$ equivalent iterations.

In order to show the potential of the proposed code, we increase the number of iterations $I$ in step 2 of the windowed decoder of SC-LDPC code A from $I = 1$ to $I = 3$. This result is shown by the leftmost curve in Fig. 7.28. By extrapolating the curve, we conjecture an NCG of approximately 12.14 dB. If an error floor occurs at BERs lower than $10^{-13}$–$10^{-14}$, it can usually be well compensated for by adequate post-processing techniques, as detailed in [7.196].

As increasing the number of iterations to $I = 3$ also increases the decoder latency, we keep $I = 1$ and increase the number of decoding engines to $E = 2$. We can see an improvement in the NCG by around 0.1 dB without an increase in latency (although a doubling of the number of iterations is required). We can also see that the slope of the BER curve of SC-LDPC code A becomes significantly steeper such that it becomes almost comparable in performance to SC-LDPC code B.

To reuse these codes in, for example, transmission experiments, we provide the most common thresholds in terms of MI/GMI and pre-FEC BER in Table 7.4. For a detailed discussion about threshold-based performance evaluation, we refer the reader to Sect. 7.7.

To summarize, we can say that SC-LDPC codes generally outperform conventional (block) LDPC codes. The performance can be further improved if the number of iterations is increased; hence such candidates can be used to build a future-proof communication standard. Although they are being used today with low decoding complexity, the systems could be upgraded in the future by a more complex decoder carrying out a higher number of iterations with increased NCG still guaranteeing backward compatibility.

In order to compare SC-LDPC and conventional LDPC codes, we show the potential of both codes by performing a simulation of regular codes with different rates and compute the extrapolated NCG at a target BER of $10^{-15}$. For the SC-LDPC codes, we consider regular codes with $d_v = 4$ and we vary $d_c$ to achieve different rates. Motivated by the results in Fig. 7.26, we select $w = 6$ and we carry out windowed decoding with different window configurations. We compare these codes with regular LDPC codes with $d_v = 3$,

**Table 7.4** Common thresholds for the two SC-LDPC code B of rate $R = 4/5$ shown in Fig. 7.28

| | GMI | Pre-FEC BER | | | | |
|---|---|---|---|---|---|---|
| | $R_{\text{BW-SDD}}$ | BPSK/QPSK | 8-QAM | 16-QAM | 32-QAM | 64-QAM |
| SC-LDPC Code B, $W_D = 13$, $I = 1$, $E = 1$ | 0.866 | 0.0358 | 0.0362 | 0.0363 | 0.0367 | 0.0372 |
| SC-LDPC Code B, $W_D = 13$, $I = 1$, $E = 2$ | 0.862 | 0.0370 | 0.0375 | 0.0377 | 0.0379 | 0.0385 |
| SC-LDPC Code A, $W_D = 13$, $I = 3$, $E = 1$ | 0.853 | 0.0387 | 0.0395 | 0.0399 | 0.0398 | 0.0405 |

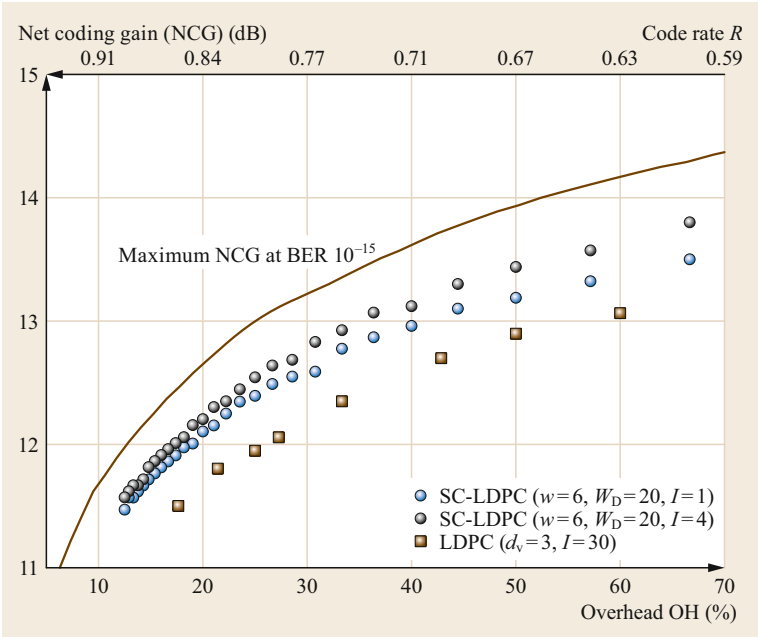


**Fig. 7.29** Achievable NCG at a BER of $10^{-15}$ for several coding schemes

which are frequently used because of their good error floor properties. Figure 7.29 shows the achievable NCGs. We can clearly see that, if we can spend the complexity to run $I = 4$ iterations per windowed decoding step, we can nearly close the gap to the limit, and gains of around 0.8 dB are possible when compared with LDPC codes. However, already with $I = 1$, we can significantly outperform regular LDPC codes.

## 7.5 Hard-Decision Forward Error Correction

The adoption of SDD in fiber-optic communications represented a breakthrough with respect to the classical schemes based on algebraic codes (BCH and Reed–Solomon codes) and HDD. As we have discussed in the previous sections, LDPC codes and SDD provide very large net coding gains and achieve performance close to the Shannon limit. However, the implementation of BP decoding of LDPC codes still presents several challenges at very high data rates. This motivates the use of HDD for applications where complexity and throughput is a concern, since HDD can yield significantly reduced complexity and decoder data flow compared with SDD. HDD-based FEC is currently used in regional/metro optical transport networks [7.22] and for other applications such as optical data center interconnects [7.199]. Powerful code constructions for HDD date back to the 1950s, when *Elias* introduced PCs [7.18]. In recent years, the introduction of new code constructions such as staircase codes [7.49] and braided codes [7.50, 200], which can be seen as generalizations of PCs, and the link between these constructions and codes on graphs has fostered a revival of HDD for optical communications.

In this section, we focus on FEC codes for HDD that belong to the second and third generation of FEC, i.e., staircase codes and braided codes, as well as similar structures. These codes are built from algebraic block codes as component codes, usually BCH codes and Reed–Solomon codes for binary and nonbinary codes, respectively. They can be seen as concatenated codes and also fall within the category of codes on graphs or modern codes.

We start this section by briefly describing BCH and Reed–Solomon codes. We then review Elias' PCs and the underlying iterative decoding algorithm based on bounded distance decoding (BDD) of the component codes. We also discuss some interesting generalizations of PCs, including staircase codes and braided codes, and their connection to LDPC codes and SC-LDPC codes. Finally, we briefly discuss the link between iterative BDD (iBDD) of product-like codes and BP decoding and how to analyze the performance of product-like codes.

### 7.5.1 Classical Block Codes: Bose–Chaudhuri–Hocquenghem and Reed–Solomon Codes

BCH codes and Reed–Solomon codes are two of the most important classes of linear block codes. They are characterized by a rich algebraic structure that enables the use of efficient decoding algorithms that can be implemented with acceptable complexity.

#### Bose–Chaudhuri–Hocquenghem Codes

BCH codes, invented independently by *Alexis Hocquenghem* in 1959 [7.32] and by *Raj Chandra Bose* and *Dijen Ray-Chaudhuri* in 1960 [7.33], form a large class of cyclic codes. For short block lengths (up to a few hundred bits), BCH codes are among the best known codes of the same length and rate. For any pair of positive integers $\nu \geq 3$ and $t < 2^{\nu-1}$, there exists an $(n, k, d_{\min})$ binary BCH code with parameters

$$n = 2^{\nu} - 1, \quad n - k \leq \nu t, \quad d_{\min} \geq 2t + 1 \,, \tag{7.83}$$

where $\nu$ is the Galois field extension. This code can correct all error patterns of $t$ or fewer errors, and it is usually referred to as a $t$-error-correcting BCH code.

BCH codes offer great flexibility in the choice of code length, rate, and error-correcting capability. To achieve further flexibility in terms of code length and rate, BCH codes may be shortened. An $(n', k', d_{\min})$ BCH code of parameters

$$(n', k', d'_{\min}) = (n - s, k - s, d'_{\min} \geq d_{\min}) \tag{7.84}$$

is obtained from an $(n, k, d_{\min})$ BCH code by setting $s$ information bits to zero and not transmitting them. The parameter $s$ is commonly referred to as the shortening parameter. At the receiver, the shortened positions can then be assumed to be transmitted without error. Since the parameters $\nu$, $t$, and $s$ completely determine $(n, k)$, sometimes it is useful to define a BCH code through the triple $(\nu, t, s)$.

The rich structure of BCH codes allows for simple syndrome-based HDD. A very efficient algebraic HDD for BCH codes is the *Berlekamp–Massey* algorithm [7.38, 39], which finds the most probable error pattern introduced by the channel iteratively given the syndrome corresponding to the received binary vector. If the number of redundancy symbols $n-k$ is very large, however, (complete) syndrome decoding may be too complex. To lower the decoding complexity, *incomplete* syndrome decoding can be used. Incomplete syndrome decoding is usually referred to as BDD, which can also be implemented based on the Berlekamp–Massey algorithm. For a block code with error correction capability $t$, BDD corrects all error patterns with Hamming weight less than or equal to $t$ and no others. Note that BDD is not ML; therefore, it incurs a penalty in performance.

For details on the construction of BCH codes and the Berlekamp–Massey algorithm, the interested reader is referred to [7.15, Chap. 9] and [7.101, Chap. 3]. A nice overview of syndrome decoding can be found in [7.201, Chap. 10].

#### Reed–Solomon Codes

Reed–Solomon codes are the most important and widely used class of nonbinary block codes. Reed–Solomon codes have made their way into multiple applications and standards, such as Digital Video Broadcasting (DVB) [7.202] and deep-space communications [7.203], as well as for data storage and distributed data storage systems such as RAID 6 [7.204].

Reed–Solomon codes were introduced in 1960 by *Reed* and *Solomon* [7.37] and can be seen as a subclass of BCH codes generalized to the nonbinary case. For Reed–Solomon codes, both the construction field and the symbol field are the same, i.e., the code symbols are over a nonbinary field. An $(n, k, d_{\min})$ Reed–Solomon code over the Galois field GF$(2^{\nu})$ has parameters

$$n = 2^{\nu} - 1, \quad k = 2^{\nu} - 2t - 1, \quad d_{\min} = 2t + 1 \,, \tag{7.85}$$

where the code symbols are over GF$(2^{\nu})$. The Galois field GF$(2^{\nu})$ is an extension field of the binary numbers $\{0, 1\}$ and contains $2^{\nu}$ nonbinary symbols, which can be represented using $\nu$ bits, together with well-defined addition and multiplication operations. A Reed–Solomon code with these parameters is capable of correcting all error patterns of $t$ or fewer symbol errors.

The minimum Hamming distance of Reed–Solomon codes achieves the Singleton bound, $d_{\min}(C_{\mathrm{RS}}) = n - k + 1$ (this is readily seen from (7.85)), i.e., their minimum Hamming distance is the maximum possible for a linear code of parameters $(n, k)$ (see Theorem 7.1 in Sect. 7.3).

Note that the code symbols of a Reed–Solomon code can also be interpreted as binary vectors by representing each symbol by a binary tuple of $\nu$ bits. In this case, Reed–Solomon codes can be used for binary transmission (e.g., using BPSK) and are well suited for burst-error correction.

Similar to BCH codes, Reed–Solomon codes can also be shortened to yield higher flexibility in terms of block length and code rate. A shortened Reed–Solomon code with shortening parameter $s$ obtained from an

$(n, k, d_{\min})$ Reed–Solomon code over the Galois field GF($2^{\nu}$) has parameters

$$(n', k', d'_{\min}) = (n - s, k - s, d_{\min}) \,. \tag{7.86}$$

Reed–Solomon codes can also be efficiently decoded using the Berlekamp–Massey algorithm.

### 7.5.2 Product Codes

PCs, introduced by *Elias* in 1954 [7.18], are one of the first examples of a construction of long, powerful codes from short component codes. Let $C_1$ be an $(n_1, k_1, d_1)$ linear block code of rate $R_1 = k_1/n_1$ and minimum Hamming distance $d_1$, and $C_2$ an $(n_2, k_2, d_2)$ linear block code of rate $R = k_2/n_2$ and minimum Hamming distance $d_2$. A two-dimensional $(n, k, d_{\min})$ PC $C_{\mathrm{PC}} = C_1 \times C_2$ is defined by a rectangular array $\mathbf{C}$ such that each row of $\mathbf{C}$ is a codeword of $C_1$ and each column is a codeword of $C_2$. The codes $C_1$ and $C_2$ are commonly referred to as *component codes*. Both binary and nonbinary PCs can be constructed. In the first case, the component codes are typically BCH codes, although other (simpler) linear block codes such as single parity-check codes and Hamming codes can also be used. The component codes of nonbinary PCs, on the other hand, are usually Reed–Solomon codes. The code array $\mathbf{C}$ for a PC with component codes with parameters $(n_1, k_1) = (6, 4)$ and $(n_2, k_2) = (6, 3)$ is depicted in Fig. 7.30.

A PC can be conveniently described by the encoding procedure. The $k = k_1 k_2$ information bits are arranged in a $k_2 \times k_1$ array $\mathbf{U}$ and placed in the upper-left quadrant of the code array (in orange in Fig. 7.30). Each row of the array $\mathbf{U}$ is then encoded by the component code $C_1$ to generate $n_1 - k_1$ parity bits, marked in blue in the figure. Each column of the resulting array, of dimensions $k_2 \times n_1$, is then encoded by the code $C_2$ to generate $n_2 - k_2$ parity bits, marked in green. The resulting PC is a systematic code with parameters $k = k_1 k_2$, $n = n_1 n_2$, and rate $R = k_1 k_2/(n_1 n_2) = R_1 R_2$. It is also easy to show that the minimum Hamming distance of a PC is the product of the minimum Hamming distances of the component codes, i.e., $d_{\min} = d_1 d_2$. Thus, PCs are usually characterized by a large minimum Hamming distance. On the other hand, the minimum Hamming distance multiplicity, i.e., the number of codewords of weight $d_{\min}$, of a PC is equal to the product of the minimum Hamming distance multiplicities of the component codes [7.205]. As a result, the minimum Hamming distance multiplicities of PCs are typically high.

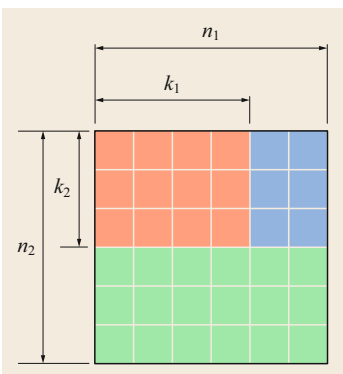


**Fig. 7.30** Code array of a PC with component codes $C_1$ and $C_2$ of parameters $(n_1, k_1) = (6, 4)$ and $(n_2, k_2) = (5, 3)$, respectively. *Orange squares* correspond to data bits; *blue* and *green squares* correspond to parity bits

With the construction above, it is apparent that each row of $\mathbf{C}$ is a codeword of $C_1$ and each column of $C$ is a codeword of $C_2$. Therefore, all code bits of a PC are protected by two component codes, or in other words, each code bit participates in two *code constraints*, a row code constraint and a column code constraint. For example, the second data bit is protected by the second row and second column constraints.

#### Irregular Product Codes and Multidimensional Product Codes

PCs, as described above, are based on a single row and column code, i.e., the same code is used for all rows and the same code is used for all columns. However, all row and column codes do not need to be the same, and codes of different rates and erasure-correcting capabilities can be used as row and column codes. The resulting PC is referred to as an *irregular* PC [7.206, 207]. Like irregular LDPC codes, irregular PCs based on code mixtures may yield performance improvements. It is also important to note that PCs may be built over arrays with more than two dimensions [7.18]. However, two-dimensional PCs are the most common, and therefore in this chapter we restrict ourselves to two-dimensional PCs.

### 7.5.3 Generalized Product Codes

The construction of PCs, with their row-column encoding and the underlying rectangular code array, is a natural construction for constructing long codes from simpler codes. However, more general code arrays, where each row and each column of the array are codewords of given linear block codes, can also be constructed. This gives rise to so-called *product-like codes*, which we will refer to here as generalized product codes (GPCs), since they can be seen as generalizations of PCs. Two of the most popular classes of GPCs are staircase codes and braided codes.

### Staircase Codes

Staircase codes are an important class of error-correcting codes proposed by *Benjamin Smith* et al. [7.49] for optical transport networks. A staircase code is defined by a two-dimensional code array that has the form of a staircase. Formally, given a component code $C$ of length $n_c$, a staircase code is defined as the set of all binary matrix sequences $\mathbf{B}_i$, $i = 1, 2, \ldots$, of dimensions $a \times a$, where $a = n_c/2$, such that each row of $[\mathbf{B}_i^\top, \mathbf{B}_{i+1}]$ is a valid codeword of $C$. The matrix $\mathbf{B}_1$ is assumed to be initialized to all zeros. The code array of a staircase code with component code $C$ of code length $n_c = 12$ and dimension $k_c = 10$ is depicted in Fig. 7.31. Encoding is performed in batches, where each batch corresponds to one of the matrices $\mathbf{B}_i$. Specifically, batch $i$ corresponds to $\mathbf{B}_i^\top$ for $i$ odd and to $\mathbf{B}_i$ for $i$ even. The number of bits per batch, denoted by $B$, is $B = a^2 = n_c^2/4$.

Like PCs, staircase codes are well defined through the encoding procedure. With reference to Fig. 7.31, the first batch (matrix $\mathbf{B}_1^\top$) is initialized to all zeros. Then, $(k_c - n_c/2)(n_c/2) = 4 \times 6$ data bits are placed in the left part of batch 2 (marked with a thick red frame in the figure), and the the rows of batches 1 and 2 are encoded by the component code $C$, thus generating $(n_c - k_c)(n_c/2) = 2 \times 6$ parity bits, i.e., $n_c = 2$ parity bits per row. $(k_c - n_c/2)(n_c/2)$ data bits are then placed in the upper part of batch 3, and the columns of the array formed by batches 2 and 3 are encoded using $C$, generating $n_c - k_c = 2$ parity bits per column. The row/column encoding process continues accordingly for the next batches. The code rate of the staircase code, $R_{SC}$, is the ratio between the number of information bits per batch and $B$, i.e.,

$$R_{SC} = \frac{(k_c - n_c/2)(n_c/2)}{n_c^2/4} = 2\frac{k_c}{n_c} - 1 \,. \tag{7.87}$$

Note that, as for PCs, each bit of a staircase code is protected by two code constraints, a row constraint and a column constraint. Also, it is worth mentioning that staircase codes are stream-oriented. Indeed, the code array in Fig. 7.31 can grow infinitely large.

Good (binary) staircase codes with BCH component codes and overhead ranging from 6% to 33% can be found in [7.208], where the parameters of the component BCH codes were optimized based on computer simulations. An optimization of the staircase code parameters based on density evolution (see Sect. 7.5.7 below) is addressed in [7.209]. An extension of the original code construction with larger staircase block sizes by allowing for multiple code constraints per row/column in the staircase array is also proposed in [7.209], leading to codes with steeper waterfall performance. Nonbinary staircase codes with Reed–Solomon component codes are discussed in [7.90, 210].

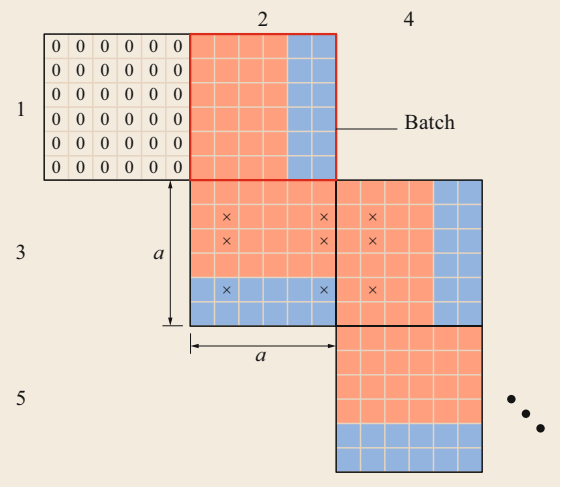


**Fig. 7.31** Code array of a staircase code with component code of code length $n_c = 12$ and dimension $k_c = 10$. The *orange squares* correspond to data bits, while the *blue squares* correspond to parity bits

### Braided Codes

Braided codes were originally proposed by *Jiménez Feltström* et al. [7.200]. Similar to PCs and staircase codes, the code construction is defined by a two-dimensional array where each bit is protected by a row and a column code. Braided codes can be constructed using both convolutional and block codes as component codes. Correspondingly, the resulting codes are referred to as braided convolutional codes and braided block codes, respectively. Here we are interested mainly in block braided codes, as they are more suitable for HDD, and for ease of exposition we will refer to them simply as braided codes.

Braided codes come in many flavors, depending on the structure of the code array. In Fig. 7.32, we depict the code array of a so-called block-oriented braided code with component code $C$ of code length $n_c = 12$ and dimension $k_c = 10$. The encoding is similar to that of product and staircase codes: Data bits are placed in the code array in the parts marked in orange, and parity bits (in blue) are generated by row and column encoding. Encoding is better understood with reference to Fig. 7.32. The code array consists of blocks of dimensions $b \times b$, where $b = n_c/3$. The blocks are grouped in batches comprising three blocks. The number of bits per batch is $B = 3b^2 = n_c^2/3$. The first batch is shown by the thick red frame in the figure. The leftmost and uppermost blocks of the code array are initialized to all zeros. Then, $b^2 + 2(k_c - 2b)b = n_c^2/9 + 2(k_c - 2n_c/3)(n_c/3) =$

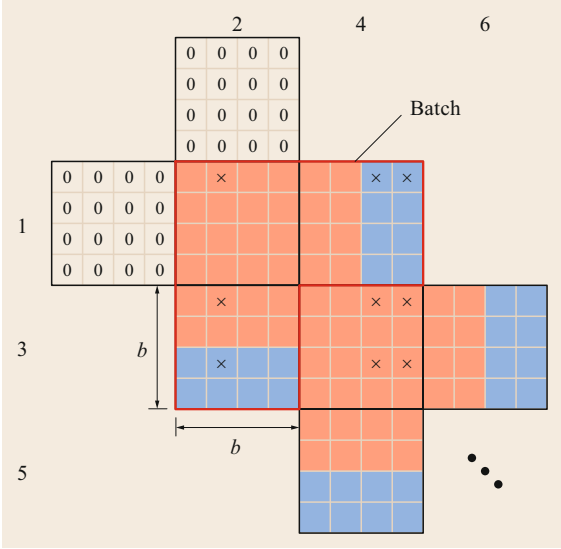


**Fig. 7.32** Code array of a braided code with component code of code length $n_c = 12$ and dimension $k_c = 10$. The *orange squares* correspond to data bits, while the *blue squares* correspond to parity bits

32 data bits are placed in the orange squares of the first batch. The rows of the blocks at (horizontal) position 1 are then encoded by the $(n_c, k_c)$ component code $C$, generating $(n_c - k_c)(n_c/3) = 2 \times 4 = 8$ parity bits, i.e., $n_c = 2$ parity bits per row (marked in blue in the figure). The columns of the array formed by the three blocks in (vertical) position 2 are encoded using $C$, generating $n_c - k_c = 2$ parity bits per column. The row/column encoding process continues accordingly for the next batches. The code rate of the braided code, $R_{BC}$, is the ratio between the number of information bits per batch and $B$, i.e.,

$$R_{BC} = \frac{(n_c^2/9) + 2(k_c - 2n_c/3)(n_c/3)}{n_c^2/3} = 2\frac{k_c}{n_c} - 1 \,. \tag{7.88}$$

Braided codes have been explicitly considered for use in fiber-optic communication systems in [7.50].

### Half-Product Codes and Half-Braided Codes

Consider a PC based on component codes of lengths $n_1 = n_2 = n_c$ (Fig. 7.30). Based on this PC, a new code can be obtained by retaining only codeword arrays with zeros in the main diagonal that are symmetric, in the sense that the array is equal to its transpose. The code bits in the main diagonal of the resulting array do not carry any information (they are known), and hence they can be punctured, i.e., they are not transmitted. Also,

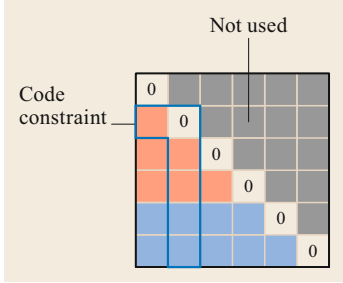


**Fig. 7.33** Code array of an HPC with component code of code length $n_c = 6$ and dimension $k_c = 3$. The *orange squares* correspond to data bits, while the *blue squares* correspond to parity bits. The bits corresponding to the *gray squares* are not transmitted

the bits in the upper triangular part are identical to those in the lower triangular part and can also be punctured. The resulting code has length $n = \binom{n_c}{2}$ and is referred to as a half-product code (HPC) [7.211]. The code array of an HPC with component codes of length $n_c = 6$ is depicted in Fig. 7.33, where one particular code constraint is marked with the thick blue frame. In [7.212], it is shown that HPCs can achieve larger normalized minimum distance than PCs and superior performance in both the waterfall region and the error floor compared with PCs of similar length and rate.

The concept above can be extended to other code arrays. For example, the same symmetry constraint can be applied to braided codes, and the resulting codes are referred to as half-braided codes. The code array of a half-braided code with component codes with block length $n_c = 12$ and dimension $k_c = 10$ is depicted in Fig. 7.34. It is easy to see that the code rate of a half-braided code in the form of Fig. 7.34, $R_{HBC}$, is

$$R_{HBC} = 2\frac{k_c - 1}{n_c - 1} - 1 \,. \tag{7.89}$$

HPCs and half-braided codes belong to a larger class of symmetric GPCs [7.212]. Symmetric GPCs are in turn a subclass of GPCs that use symmetry to reduce the block length of a GPC while using the same component code. In [7.213], it was shown that half-braided codes can outperform both staircase codes and braided codes in waterfall performance, at a lower error floor and decoding delay.

Similar to staircase codes, braided codes, HPCs, and half-braided codes, other generalizations of PCs (leading to other code arrays) can be envisaged. In Sect. 7.5.5, we discuss a general construction of GPCs that encompasses staircase codes and braided codes, as well as other classes of GPCs.

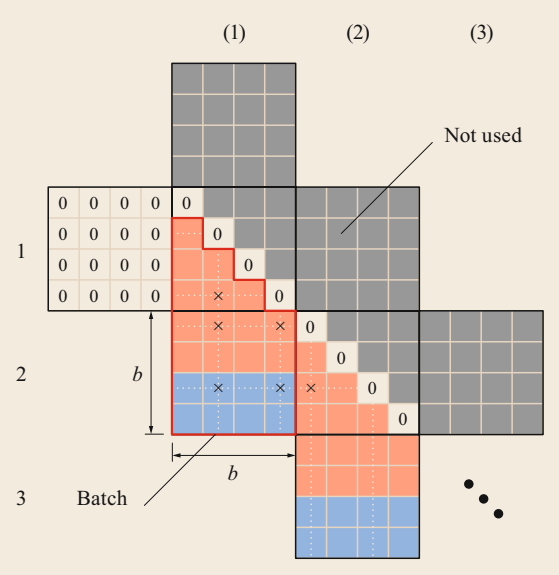


**Fig. 7.34** Code array of a half-braided code with component code of code length $n_c = 12$ and dimension $k_c = 10$. The *orange squares* correspond to data bits, while the *blue squares* correspond to parity bits. The bits corresponding to the *gray squares* are not transmitted

### 7.5.4 Generalized Product Codes as Instances of Generalized Low-Density Parity-Check Codes

It is interesting to observe that GPCs can be interpreted as instances of generalized LDPC (G-LDPC) codes. The link between GPCs and G-LDPCs is relevant because it allows us to borrow tools for the analysis of LDPC codes (and codes on graphs in general) to analyze the behavior of GPCs. G-LDPC codes are a generalization of LDPC codes, where the constraint nodes are not simple single parity-check codes as for LDPC codes, but general linear block codes, e.g., Hamming codes or BCH codes [7.101]. A G-LDPC code is also defined by a bipartite graph where an edge between a VN and a CN indicates that the code bit corresponding to the VN participates in the code constraint enforced by the component code represented by the CN.

To clarify the link between GPCs and G-LDPC codes, let us first consider the Tanner graph representation of a PC. As any block code, PCs can be represented by a Tanner graph where VNs represent code bits and CNs represent row and column codes. For a PC with component codes of code lengths $n_1$ and $n_2$, the corresponding Tanner graph has $n_1 + n_2$ CNs ($n_1$ CNs corresponding to the column codes and $n_2$ CNs corresponding to the row codes, Fig. 7.30) and $n_1 n_2$ VNs corresponding to the $n_1 n_2$ code bits. The (bipartite) Tanner graph of the PC with component codes with parameters $n_1 = n_2 = 6$ in Fig. 7.30 is depicted in Fig. 7.35. Note that each CN has degree 6, since the component codes have code length $n_1 = n_2 = 6$, and each VN has degree 2, since each code bit participates in two code constraints. A simplified version of the Tanner graph of Fig. 7.35 is illustrated in Fig. 7.36. In the simplified Tanner graph, we distinguish between two types of CNs, corresponding to row and column constraints. The VNs (corresponding to code bits) are represented by edges. Since each code bit is protected by a row and a column code, each edge is connected to one CN of each type. The resulting Tanner graph is a complete graph where each CN of one type is connected to all CNs of the other type through an edge. There are $n_1 n_2$ edges, each corresponding to one code bit. Note that the code array in Fig. 7.30, the *standard* Tanner graph in Fig. 7.35, and the simplified Tanner graph in Fig. 7.36 are equivalent representations of the PC. For compactness reasons, it is more convenient to represent PCs with the simplified Tanner graph than with the conventional Tanner graph of Fig. 7.35.

Similar to PCs, GPCs can be represented by a Tanner graph. The standard Tanner graph and the simplified Tanner graph of a staircase code (Fig. 7.31) with component code of code length $n_c = 6$ are depicted in Fig. 7.37 and Fig. 7.38, respectively. The standard Tanner graph of a staircase code is defined by a number of positions, $L$, corresponding to the the total number

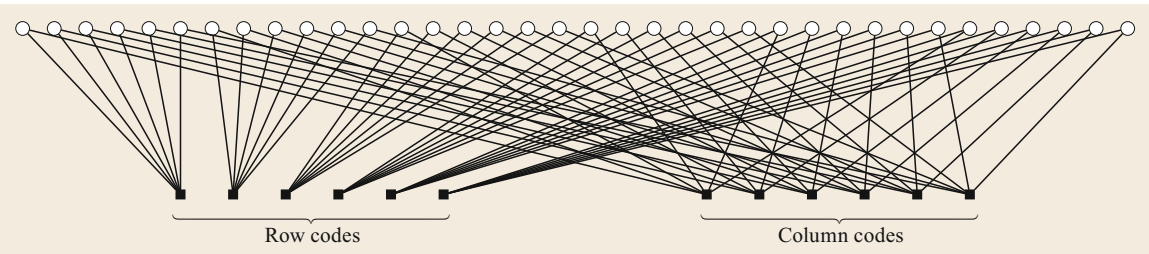


**Fig. 7.35** Bipartite (Tanner) graph of a PC with component codes of lengths $n_1 = n_2 = 6$ of Fig. 7.30

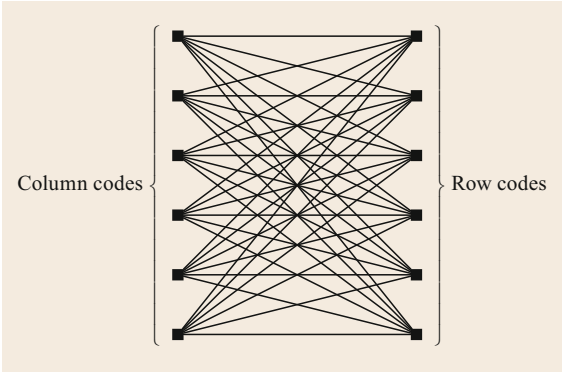


**Fig. 7.36** Simplified Tanner graph of a PC with component codes of lengths $n_1 = n_2 = 6$ of Fig. 7.30

of batches minus one. Each position corresponds to one batch of code bits and correspondingly has $a^2 = (n_c/2)^2$ VNs, corresponding to the $a^2$ code bits in the batch, and $a = n_c/2$ CNs, corresponding to the row (or column) codes defined by the batches $[\mathbf{B}_i^\top, \mathbf{B}_{i+1}]$ for position $i$, $i = 1, 2, \ldots, L$. Accordingly, the simplified Tanner graph is also defined by $L$ positions, with $a$ CNs per position. Each CN in position $i$ is then connected to all CNs in spatial position $i+1$ by an edge (a VN, corresponding to a code bit), Fig. 7.38. Note that all VNs have degree 2, since a code bit participates in two code constraints, and all CNs have degree 6, since $n_c = 6$.

Figures 7.35–7.38 make apparent the link between GPCs and G-LDPC codes. In particular, the bipartite Tanner graph of a staircase code resembles that of an SC-LDPC code (Fig. 7.37 and Fig. 7.19 in Sect. 7.4.2). Indeed, staircase codes can be seen as spatially coupled versions of PCs, and as such, as a subclass of spatially coupled G-LDPC codes with coupling width $w = 2$. Similar to staircase codes, other GPCs such as braided codes and half-braided codes can be classified as SC-GLDPC codes.

The connection of GPCs to G-LDPC codes enables the use of tools for the analysis of codes on graphs, such as density evolution, to analyze GPCs. This is discussed in Sect. 7.5.7.

### 7.5.5 A General Code Construction of Generalized Product Codes

In the previous sections, we described some of the most popular classes of GPCs. In this section, we briefly review a deterministic construction of GPCs proposed in [7.214, 215] that is sufficiently general to encompass several classes of GPCs, including irregular PCs, HPCs, staircase codes, block-wise braided codes, and half-braided codes.

The code construction in [7.214, 215] is defined in terms of three parameters, $\boldsymbol{\eta}$, $\boldsymbol{\gamma}$, and $\boldsymbol{\tau}$, and is specified over the simplified Tanner graph. We denote the corresponding GPC by $C_m(\boldsymbol{\eta}, \boldsymbol{\gamma}, \boldsymbol{\tau})$, where $m$ denotes the total number of CNs in the underlying Tanner graph; $\boldsymbol{\eta}$ is a binary, symmetric $L \times L$ matrix, where $L$ is the number of positions of the Tanner graph, and $\boldsymbol{\gamma} = (\gamma_1, \ldots, \gamma_L)^\top$ is a probability vector of length $L$, i.e., $\sum_{i=1}^{L} \gamma_i = 1$ and $\gamma_i \geq 0$. The parameters $\boldsymbol{\eta}$ and $\boldsymbol{\gamma}$ determine the graph connectivity. Considering the representation of a GPC in terms of the two-dimensional code array (Figs. 7.30–7.34), one may alternatively think about $\boldsymbol{\eta}$ and $\boldsymbol{\gamma}$ as specifying the array shape. Different choices for $\boldsymbol{\eta}$ and $\boldsymbol{\gamma}$ recover well-known code classes. The parameter $\boldsymbol{\tau}$ is used to specify GPCs employing component codes with different erasure-correcting capabilities.

The simplified Tanner graph describing the GPC $C_m(\boldsymbol{\eta}, \boldsymbol{\gamma}, \boldsymbol{\tau})$ is constructed as follows. Assume that the Tanner graph spans $L$ positions. Then, place $m_i \triangleq \gamma_i m$ CNs at each position $i = 1, \ldots L$, where it is assumed that $m_i$ is an integer for all $i$. Then, connect each CN at position $i$ to each CN at position $j$ through a VN (i.e., an edge) if and only if $\eta_{i,j} = 1$.

*Example 7.5*

A PC is obtained by choosing $L = 2$ and $\boldsymbol{\eta} = \left(\begin{smallmatrix} 0 & 1 \\ 1 & 0 \end{smallmatrix}\right)$. The two positions correspond to the row and column codes. Choosing $\boldsymbol{\gamma} = (1/2, 1/2)$ leads to a standard PC (with a square code array) with component codes $C_1$ and $C_2$ of length $n_1 = n_2 = m/2$. Note that by selecting $m = 12$, we recover the Tanner graph of the PC in Fig. 7.36.

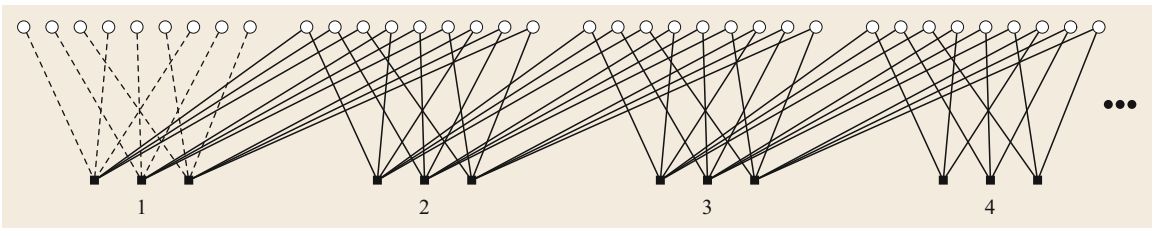


**Fig. 7.37** Bipartite (Tanner) graph of a staircase code (Fig. 7.31) with component code of code length $n_c = 6$

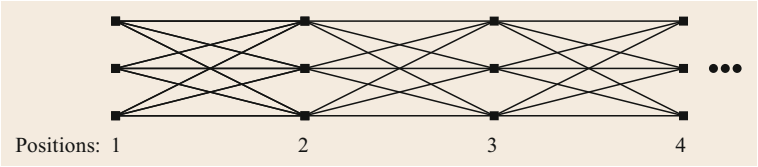


**Fig. 7.38** Simplified Tanner graph of a staircase code (Fig. 7.31) with component code of code length $n_c = 6$

*Example 7.6*

For a fixed $L \geq 2$, the matrix $\boldsymbol{\eta}$ describing a staircase code has entries $\eta_{i,i+1} = \eta_{i+1,i} = 1$ for $i = 1, \ldots, L-1$ and zeros elsewhere. The distribution $\boldsymbol{\gamma}$ is uniform, i.e., $\gamma_i = 1/L$ for all $i \in [L]$. For example, the staircase code corresponding to the code array shown in Fig. 7.31, where $L = 4$ and $m = 24$ (i.e., $m_i = 6$), is defined by

$$\boldsymbol{\eta} = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}, \tag{7.90}$$

and $\gamma_i = 1/6$. The CNs at all positions have the same degree $2n\gamma_i = 12$, except for positions 1 and $L$, where the degrees are $n\gamma_i = 6$. With this choice of $\boldsymbol{\eta}$ we recover the simplified Tanner graph in Fig. 7.38.

For more details on this general code construction, the interested reader is referred to [7.215].

### 7.5.6 Decoding of Product-Like Codes: Iterative Bounded Distance Decoding

PCs may be decoded with SDD using an iterative decoding algorithm based on the iterative exchange of soft information between two soft-input soft-output (SISO) decoders matched to the two component codes. The decoding is similar to that of turbo codes, and for this reason, PCs with iterative SDD are often referred to as turbo product codes (TPCs) or block turbo codes (BTCs). Similarly, GPCs can also be decoded by exchanging soft information between the SISO decoders of the component codes in an iterative fashion. The SISO decoders of the component codes may implement the MAP decoding rule, which can be done based on a trellis representation of the block code. However, MAP decoding of algebraic block codes is complex, and therefore iterative SDD of GPCs based on MAP decoding of the component codes is impractical. An efficient, lower-complexity iterative SDD algorithm for PCs with near-optimal performance was proposed by *Pyndiah* [7.58]. In this algorithm, the SISO decoders of the component codes are based on a modification of the Chase decoding algorithm, a low-complexity suboptimal algorithm for near-ML decoding of linear block codes proposed by *David Chase* in 1972 [7.216]. Since the Chase decoder is a vital component of the iterative SDD algorithm proposed in [7.58], the algorithm used to decode the component codes is usually referred to as the Chase–Pyndiah decoding algorithm. The advantage of this decoding algorithm is that it is highly parallelizable.

The iterative decoding of GPCs employing SISO component decoding is usually referred to as turbo product decoding (TPD) and is typically implemented in practice via the Chase–Pyndiah algorithm. However, it is worth mentioning that TPD can also be based on other component decoders, such as the forward-backward algorithm applied to the component code trellis.

The complexity of the Chase–Pyndiah decoder may still be unacceptable for some applications. Alternatively, one can decode PCs and GPCs using HDD. Indeed, PCs and GPCs are very well suited for HDD thanks to the fact that the component codes (typically BCH or Reed–Solomon codes) have a rich algebraic structure, which enables efficient implementation of BDD.

#### Iterative Bounded Distance Decoding of Product Codes

For large block codes, optimal (ML) HDD is a formidable task. As mentioned in Sect. 7.5.1, to reduce the decoding complexity, incomplete BDD can be used. However, even BDD of powerful codes such as PCs and GPCs is a challenging task. To achieve acceptable complexity, an iterative (suboptimal) decoding algorithm, based on BDD of the component codes, can then be used. Let us first consider HDD of PCs. The decoding of a PC can be accomplished by iteratively applying BDD to the row and column codes, described as follows. The received bits are arranged in an $n_2 \times n_1$ array corresponding to the code array (Fig. 7.30). Decoding is then carried out on the columns of the array, i.e., BDD decoding of the column component codes $C_2$ is performed, and subsequently on the rows of the array, i.e., BDD decoding of the row component codes $C_1$ is performed. Errors remaining after the first column decoding may be corrected by the bounded distance decoders of the row codes. The column-row decoding is then iterated until a maximum number of iterations is reached. We refer to this algorithm as iBDD.

iBDD achieves excellent performance-complexity trade-off. The decoding complexity of iBDD is roughly the sum of the decoding complexity of the BDD of the component codes times the number of iterations. Hence, the complexity of iBDD is much lower than that of ML decoding of the PC. On the other hand, since iBDD is suboptimal, the lower decoding complexity comes at a cost of lower performance. The complexity of iBDD of PCs and GPCs in general is also much lower than that of (suboptimal) BP decoding of LDPC codes. Moreover, the decoder data flow requirements are estimated to be more than one order of magnitude smaller than the requirements for a comparable LDPC code with BP decoding [7.49].

#### Iterative Bounded Distance Decoding of Generalized Product Codes

Like PCs, GPCs can be decoded using iBDD by iterating between the row and column decoders. For simplicity, in this section we describe the decoding of staircase codes. However, the decoding of other GPCs such as braided codes and half-braided codes follows the same principle.

Since staircase codes may be very long (as mentioned earlier, they are stream-oriented in nature), waiting to receive the entire code array prior to the start of decoding would entail a very long delay, which is not acceptable in fiber-optic communications. Fortunately, the batch-oriented structure of staircase codes allows us to perform decoding in a sliding-window fashion, similar to windowed decoding of SC-LDPC codes (Sect. 7.4.2). The windowed decoder operates on part of the array consisting of a window comprising a subset of $W$ received batches. The decoder then iterates between the BDD decoders for all rows and all columns within the window. After a predetermined number of iterations, the window slides to the next position, and iterative decoding is performed within the new window. The decoding delay (in bits) is given by $D = WB$, where $B = a^2$ is the number of code bits per batch (Sect. 7.5.3). The window size provides a trade-off between performance and decoding latency, i.e., one expects a performance improvement by increasing the window size at a cost of higher decoding latency. In general, small values of $W$ (6 to 8) are typically sufficient.

Despite the suboptimality of the iBDD algorithm, staircase codes—and GPCs in general—provide excellent performance for high code rates. For example, the $R = 0.937$ staircase code designed in [7.49] performs only about 0.56 dB from the channel capacity of the BSC.

It is worth mentioning that BDD of the component codes may lead to *miscorrections*, i.e., the component decoders may declare successful decoding, but the decoded codeword is not the correct one. Miscorrections introduce additional bit errors into the iterative decoding process, which result in performance degradation. The impact of miscorrections becomes less severe as the error-correcting capability $t$ increases. In [7.217], a decoding algorithm for staircase codes was proposed, called *anchor decoding*, which reduces the effect of undetected component code miscorrections. The algorithm significantly improves performance, in particular when $t$ is small, while retaining low-complexity implementation.

#### Iterative Bounded Distance Decoding as a Message-Passing Algorithm

With reference to the Tanner graph representation of GPCs, iBDD of GPCs can be interpreted as an iterative message-passing decoding algorithm similar to that of LDPC codes, where *hard* (binary) messages are exchanged between VNs and CNs in the Tanner graph describing the code. This opens the door to the analysis of the performance of iBDD of GPCs using codes-on-graphs tools.

It is worth pointing out that the classical message-passing rule typically used to decode PCs and GPCs violates the principle that only *extrinsic* information should be exchanged during the iterative decoding process [7.218], since the computation of the messages passed from CNs to VNs use the input messages at the CNs. For this reason, we will use the same terminology introduced in [7.50] and refer to the decoding algorithm described above as iterative HDD with intrinsic message passing (IMP). In [7.219], a modified message-passing algorithm that exchanges only extrinsic information, dubbed extrinsic message passing (EMP), was proposed. EMP yields better performance than IMP at a cost of some increase in complexity (mostly memory complexity, which is doubled). A low-complexity EMP HDD algorithm is given in [7.50].

### 7.5.7 Analysis and Optimization of Generalized Product Codes

Similar to LDPC codes and codes on graphs in general, the BER curve of GPCs is characterized by two well-defined regions. In the first one, called the waterfall region, the BER decreases sharply with $E_b/N_0$. The curve then flattens out in the so-called error floor region. The error floor is usually caused by combinations of errors that cannot be corrected by the decoder. For GPCs with iBDD, these patterns are usually referred to as *stall patterns* [7.49]. A stall pattern for the staircase code, the braided code, and the half-braided code in Figs. 7.31, 7.32, and 7.34, respectively, assum-

ing $t = 2$ error-correcting component codes, is shown by the crosses in the figures. Since every component codeword involved has three errors, decoding will not be able to proceed, and stalls.

Let $s_{\min}$ be the size of the minimum stall pattern, defined as the minimum number of array positions that, when all received in error, cause the decoder to stall. A stall pattern is said to be assigned to a batch if at least one of its array positions belongs to the batch and no positions belong to previous batches. The error floor of staircase codes, braided codes, and half-braided codes can be approximated by

$$\mathrm{BER} \approx \frac{s_{\min} M p^{s_{\min}}}{B}, \tag{7.91}$$

where $M$ denotes the number of minimum stall patterns that can be assigned to a batch. For staircase codes, $M$ is derived in [7.49, Sect. V-B]. Using similar arguments, the values of $M$ for braided codes and half-braided codes were obtained in [7.213].

The performance of GPCs in the waterfall region is more difficult to analyze. However, the asymptotic performance of GPCs in the limit of infinite block lengths can be analyzed exploiting the connection with G-LDPC codes. The analysis of codes on graphs such as LDPC codes and G-LDPC codes is commonly based on an ensemble argument, i.e., rather than analyzing a particular code, which is very cumbersome, it is customary to analyze the average behavior of a code ensemble. A code ensemble can be regarded as a family of codes that share some common characteristics, such as the degree distribution. The asymptotic behavior of a code ensemble can be analyzed via a tool called density evolution [7.117], which can be used to find the so-called decoding threshold. The decoding threshold divides the channel parameter range (e.g., SNR) into a region where reliable decoding is possible and another region where it is not, and accurately predicts the region where the code performance curve *bends* into the characteristic waterfall behavior. There exists a concentration phenomenon that ensures with high probability that a particular code taken uniformly at random from the ensemble will have actual performance close to th predicted by density evolution.

The parameters of GPCs, i.e., the parameters of the component codes, can be optimized to achieve good performance in the waterfall region based on extensive software simulations for predicting the code performance, as was done in [7.208] for staircase codes. Alternatively, one can use the fact that a specific GPC, e.g., a staircase code, is contained in a given ensemble of G-LDPC codes, and can optimize the code parameters based on density evolution to optimize the decoding threshold. The optimization approach based on density evolution offers significantly reduced optimization time (or the possibility of exploring a larger parameter space) with respect to a simulation-based approach. The optimization of staircase code parameters based on density evolution was addressed in [7.209].

It is very important to emphasize that, contrary to LDPC codes, which are drawn from an ensemble, GPCs are deterministic codes, i.e., they have a very regular structure in terms of their Tanner graph and are not at all random-like. Thus, while the optimization of GPCs via ensemble-based density evolution yields good codes, formally speaking, the asymptotic performance of GPCs is not necessarily characterized by density evolution of the corresponding spatially coupled G-LDPC code ensemble. For transmission over the binary erasure channel (BEC), the density evolution equations that characterize the asymptotic decoding performance of a broad class of GPCs with a fixed Tanner graph were derived in [7.213–215], without relying on the definition of a code ensemble. The results obtained can also be used to analyze the code performance over the BSC assuming idealized BDD, i.e., completely ignoring miscorrections (idealized BDD over the BSC is conceptually equivalent to transmission over the BEC). However, the rigorous characterization of the asymptotic performance of GPCs over the BSC, including the effect of decoder miscorrections, is still an open problem. An asymptotic analysis taking into account the impact of miscorrections for a code ensemble related to staircase codes is given in [7.220].

### 7.5.8 Soft-Aided Decoding of Product-Like Codes

In Sect. 7.4 and in this section, we have provided an extensive review of soft-decision FEC and hard-decision FEC, respectively. Roughly speaking, soft-decision FEC yields higher NCGs at a cost of higher decoding complexity and data flow compared with hard-decision FEC. Thus, despite its smaller NCG, for applications where very high throughput and low power consumption are required, hard-decision FEC is still an appealing alternative. An interesting question is whether it is possible to close the gap between the performance of hard-decision and soft-decision FEC while keeping the decoding complexity low. One line of research recently pursued in this direction is to concatenate an inner soft-decision FEC code, e.g., an LDPC code decoded via BP, with an outer staircase code [7.221, 222]. Another alternative, recently investigated in [7.223, 224], is to improve the performance of iBDD of product-like codes by exploiting some level of soft information while keeping the core

algebraic decoding of the component codes, an approach that we refer to here as *soft-aided* decoding of product-like codes. Without going into detail, the algorithm in [7.223], dubbed iBDD-SR, exploits the channel reliabilities while still only exchanging binary (hard-decision) messages between component codes. iBDD improves performance over conventional iBDD, with only a minor increase in complexity. In [7.224], another soft-aided decoding algorithm was proposed, based on generalized minimum distance (GMD) decoding of the component codes. The algorithm, referred to as iterative GMD decoding with scaled reliability (iGMDD-SR) yields more significant coding gains while maintaining significantly lower complexity than SDD. In particular, for a PC, it was shown that iGMDD-SR closes over 50% of the performance gap relative to turbo product decoding, while maintaining significantly lower complexity. We note that while the algorithms in [7.223, 224] were demonstrated for PCs, their extension to staircase codes and other product-like codes is straightforward. The implementation of iBDD for a PC has been considered in [7.225], where it is shown that, using a single bit of soft information, the NCG can be improved by 0.2 dB with respect to iBDD, reaching 10.3−10.4 dB, which is similar to that of more complex hard-decision staircase decoders, but with significantly lower circuit area and energy dissipation.

### 7.5.9 Performance Curves

In Fig. 7.39, we plot the BER performance of two staircase codes of rate $R = 0.75$ (corresponding to a code overhead of 33.33%) for transmission over the BSC. The net coding gain is also shown. In particular, the figure shows the performance of the staircase code with BCH component codes with parameters $(\nu, t, s) = (9, 5, 151)$, i.e., $(n_c, k_c) = (360, 315)$ [7.208, Table II], referred to as $C_1$ in the figure. The parameters of the BCH component code were optimized in [7.208] based on extensive simulations. The performance of the code is shown for both IMP decoding, as originally proposed in [7.49], and EMP decoding, as proposed in [7.50]. The decoding is performed in a sliding-window fashion as explained in Sect. 7.5.6. A window size $W = 7$ and $\ell = 8$ iterations within a window are considered. It is observed that EMP yields better performance than IMP in the waterfall region. In the figure, we also plot density evolution results [7.209]. The density evolution curve predicts the pre-FEC BER region where the simulated curve (with EMP) bends into the characteristic waterfall.

In the figure, we also plot the performance of a staircase code of the same code rate whose parameters have been optimized based on density evolution [7.209], and

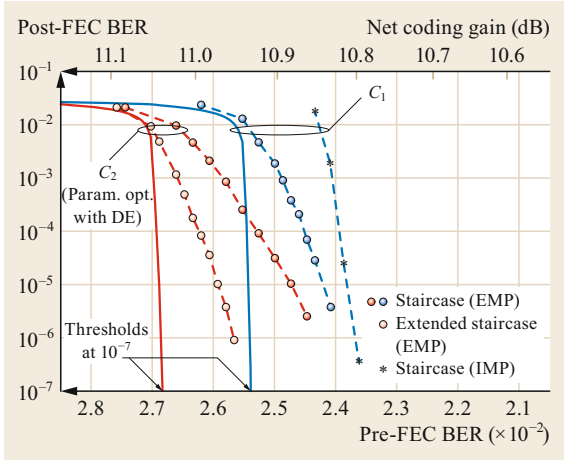


**Fig. 7.39** Simulation results (*dashed curves with markers*) and density evolution results (*solid curves*) for staircase codes and extended staircase codes

which is referred to as $C_2$ in the figure. The component BCH code has parameters $(\nu, t, s) = (8, 3, 63)$, i.e., $(n_c, k_c) = (192, 168)$. As can be seen, the optimization of the code parameters based on density evolution yields better performance in the waterfall region. The poorer slope of the staircase code with component code $C_2$ than with component code $C_1$ is due to the fact that for the former, the size of one staircase block, $a$ (Fig. 7.31), is smaller, namely $a = 96$ versus $a = 180$, corresponding to a decoding delay of $D = 64\,512$ bits and $D = 226\,800$ bits, respectively [7.209, Table 1]. Finally, in the figure we also plot the performance of an extended staircase code with block size $a = 192$ [7.209], which has a decoding delay comparable to that of the staircase code based on $C_1$. The extended staircase code has a steeper curve in the waterfall region.

In Fig. 7.40, we plot simulation results for a staircase code (red curve with markers), a braided code (blue curve with markers), and a half braided code (green curve with markers) with BCH component codes with parameters $(n_c, k_c) = (720, 690)$ and $t = 3$. All three codes have roughly the same code rate, $R \approx 0.917$, corresponding to an FEC overhead of 9.1%. Transmission over a BSC is assumed, and we plot the post-FEC BER as a function of the channel transition probability $p$. The decoding is performed using a windowed decoder with window size $W = 8$ for the staircase code and $W = 6$ for the braided code, such that the two codes have the same delay, $D = 1\,036\,800$ bits, while the decoding delay of the half-braided code with $W = 6$ is roughly half, $D = 517\,680$ bits. The staircase and braided codes are decoded by iterating $\ell = 8$ times between rows and columns within each window.

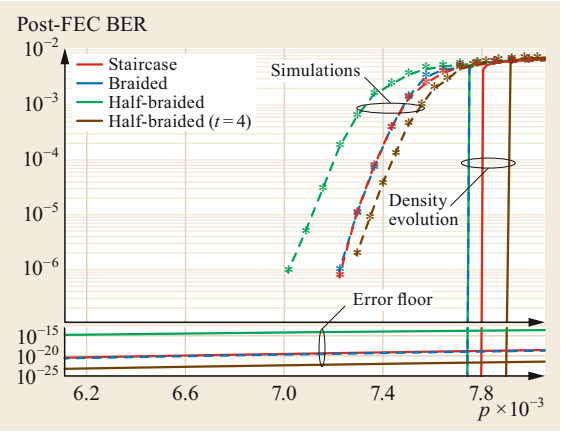


**Fig. 7.40** Simulation results (*dashed curves with markers*), density evolution results, and error floor curves for staircase, braided, and half-braided codes

For the half-braided code, all component codes within each window are decoded simultaneously and $\ell$ is increased to 16 to maintain the same decoding complexity (note that the number of component codes per window is reduced by half for half-braided codes).

In the figure, we also plot density evolution results and the error floor approximation given in (7.91). The density evolution results show a slight performance advantage for the staircase code in the asymptotic regime of large block lengths. However, for the chosen parameters at finite lengths, the simulation curves for the staircase code and the braided code are virtually on top of each other. The density evolution results for the braided code and the half-braided code are roughly the same. However, the simulated half-braided code has worse performance than the staircase and braided codes, caused by different scaling behavior at finite length due to the reduced number of bits within the decoding window. Furthermore, the error floor is increased from $\approx 10^{-20}$ for the staircase and braided codes to $\approx 10^{-14}$ because of the reduction of $s_{\min}$. On the other hand, the half-braided code operates at only half the decoding delay. In the figure, we also plot the performance of a half-braided code with a BCH component code of parameters $(960, 920, 4)$. The code rate is the same, but the decoding delay is now $D = 920\,640$ bits, which is slightly less than that of the staircase and braided codes. The code shows a reduced error floor ($\approx 10^{-23}$) and also improves the waterfall performance, as predicted by density evolution and confirmed by the simulations.

In Fig. 7.41, we plot the BER performance of a PC with double-error-correcting extended BCH (eBCH) codes with parameters $(256, 239, 6)$ as component codes for transmission over the AWGN channel. The code rate of the resulting PC is $R = 239^2/256^2 \approx 0.8716$, which corresponds to overhead OH $\approx 15\%$. In the figure, we plot the performance of both iBDD (red curve with triangle markers) and TPD based on the Chase–Pyndiah algorithm (blue curve with blue markers). Both iBDD and TPD perform about 1 dB from the respective capacity limits at a BER of $10^{-5}$. TPD yields a coding gain of about 1.5 dB with respect to iBDD, at a cost of significantly higher decoding complexity. As a reference, we also show the performance of ideal iBDD, where a genie prevents all miscorrections. Additionally, we plot the performance of the

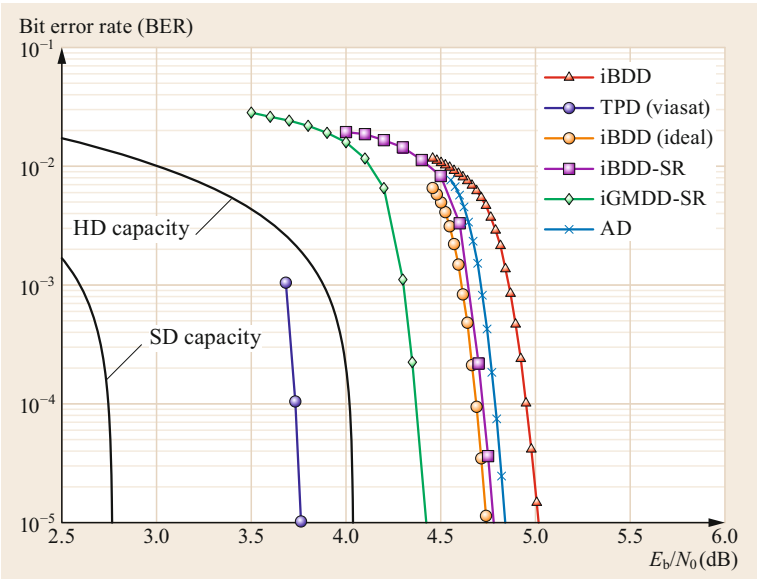


**Fig. 7.41** Simulation results for a PC with $(256, 239, 6)$ double-error-correcting eBCH codes as component codes and different decoding algorithms

anchor decoding algorithm proposed in [7.217] and of iBDD-SR [7.223]. We can observe that both anchor decoding and iBDD-SR are effective algorithms for combating miscorrections. The performance degradation of iBDD-SR compared with ideal iBDD is very small (<0.01 dB). iBDD-SR outperforms the conventional iBDD by 0.25 dB at a BER of $10^{-5}$, with only a very small increase in complexity [7.223, 225]. Finally, we also plot the performance of iGMDD-SR, which outperforms iBDD, anchor decoding, and iBDD-SR. In particular, the performance gain for iGMDD-SR over iBDD is 0.60 dB. Furthermore, iGMDD-SR performs 0.52 dB from TPD, i.e., it closes over 50% of the performance gap between iBDD and TPD [7.224].

## 7.6 Coded Modulation—An Introduction

In the previous sections we have focused mostly on binary codes and binary (BPSK) transmission. As we have seen, to operate close to the Shannon limit, (powerful) error-correcting codes are required. Forward error correction (FEC) increases the power efficiency of the system but introduces redundancy to the transmitted sequence. The added redundancy requires the modulator to operate at a higher data rate and hence requires a larger bandwidth, i.e., the spectral efficiency is decreased. Achieving higher spectral efficiency can be realized by using a higher-order signal constellation (i.e., constellations with cardinality larger than 2). Therefore, to transmit reliably at high spectral efficiencies, as required in fiber-optic communications, error-correcting codes must be combined with the use of a higher-order constellation. The combination of error-correcting coding and higher-order constellations is usually referred to as coded modulation (CM).

CM has been studied since the 1970s and comes in different flavors, depending on the code used (binary or nonbinary) and the way the code is coupled with the higher-order constellation. In this section, we briefly review the most important CM schemes, namely trellis-coded modulation (TCM), multilevel coding (MLC), bit-interleaved coded modulation (BICM), and CM with nonbinary codes. We also briefly discuss constellation shaping as a means to close the fundamental gap relative to capacity that the use of conventional signal constellations with equally spaced signal points and uniform signaling entails.

### 7.6.1 Trellis-Coded Modulation

One of the first CM schemes to appear was TCM, a CM scheme proposed by *Gottfried Ungerboeck* in the late 1970s/early 1980s [7.226, 227] as a way to increase the spectral efficiency of communication systems. It consists of the concatenation of a binary trellis code (a convolutional code) and a higher-order constellation through a careful mapping of the code bits to the constellation symbols based on the set partitioning principle. Due to the correspondence of signal sequences and paths along the trellis of the convolutional code, TCM enables efficient ML decoding using Viterbi decoding.

We have seen in Sect. 7.3.3 that, for transmission over the AWGN channel, the ML rule for SDD is to select from among all possible (modulated) codewords the one at the minimum Euclidean distance to the received vector. However, the mapping of the code bits of a (binary) code optimized in terms of its minimum Hamming distance into constellation points does not guarantee that a good Euclidean distance is obtained. The basic notion of TCM is to optimize the code (now seen as the combination of a binary code and the modulation) in the Euclidean space instead. The signal set (i.e., the constellation) $\mathcal{X}$, of cardinality $M = 2^m$ symbols, is successively partitioned into smaller disjoint subsets of sizes $M/2, M/4, M/8, \ldots$, such that the minimum intra-subset Euclidean distance (the minimum Euclidean distance between signal points within each subset) is maximized. The number of subsets into which the constellation is partitioned depends on the binary code used. The partitioning defines a mapping of binary labels $\mathsf{L}(x) = (b^{(1)}(x), \ldots, b^{(m)}(x))$ to signal points $x \in \mathcal{X}$.

The next step is to assign constellation points to branches of the trellis code. In particular, assume the use of a rate $R = k_1/n_1$, $n_1 \leq m$, convolutional code. In this case, the constellation is successively partitioned into $2^{n_1}$ subsets, each containing $2^{m-n_1}$ constellation points. Each block of $k$ data bits is then split into two sub-blocks of lengths $k_1$ and $k_2 = m - n_1$. The block of $k_1$ data bits is encoded by the convolutional encoder, generating $n_1$ code bits, which are used to select one of the $2^{n_1}$ subsets in which the constellation has been partitioned. The remaining $k_2$ data bits are used to select one of the points within the selected subset. Note that the presence of uncoded bits introduces parallel transitions in the trellis. Ungerboeck showed that by choosing $n_1 = k_1 + 1$ and $k_2 = 1$ (a single uncoded bit), it is possible to design CM schemes with coding gains of between

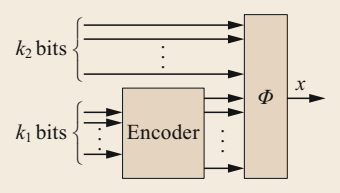


**Fig. 7.42** TCM encoder

3 and 6 dB. A block diagram of the TCM encoder is shown in Fig. 7.42.

Finally, the assignment of constellation points to the branches of the code trellis can be done by means of an exhaustive computer search in order to maximize the minimum Euclidean distance of the coded sequences. Ungerboeck introduced a set of rules that are conjectured to give rise to the best TCM schemes:

1. Members of the same partition are assigned to parallel transitions.
2. Members of the next larger partition are assigned to transitions originating from the same state or converging to the same state.
3. All signal points are used equally often.

After its introduction, TCM became very popular and was the object of intensive research. It was also quickly adopted for modem standards in the early 1990s. A thorough description and analysis of TCM can be found in [7.228]. The concept of TCM was later extended to turbo TCM [7.229] by replacing the convolutional code with a turbo code to decrease the gap relative to the Shannon limit for AWGN channels. TCM with multidimensional constellations was proposed in [7.230]. TCM was first proposed for fiber-optic systems in [7.231]. The concatenation of TCM with an outer Reed–Solomon and BCH code was studied in [7.232], yielding NCGs of 8.4 and 9.7 dB, respectively, at a BER of $10^{-13}$ for the AWGN channel.

### 7.6.2 Multilevel Coding

In parallel to TCM, an alternative CM modulation scheme, MLC, was proposed by *Hideki Imai* and *Shuji Hirakawa* [7.233] in 1977. The key idea underlying MLC is to protect each bit of the constellation symbol by an individual binary error-correcting code $C_i$. In other words, MLC transforms the nonbinary channel into a set of $m$ parallel binary sub-channels, and then uses individual binary error-correcting codes for each sub-channel. Specifically, MLC works as follows: A block $\boldsymbol{u} = (u_1 \ldots, u_k)$ of $k$ data bits is partitioned into $m$ blocks $\boldsymbol{u}^{(i)} = (u_1^{(i)}, \ldots, u_{k_i}^{(i)})$ of length $k_i$ bits, $i = 1, \ldots, m$, with $\sum_{i=1}^{m} k_i = k$. Each data block $\boldsymbol{u}^{(i)}$ is then encoded by an individual binary encoder of rate $R_i = k_i/n$, generating codewords $\boldsymbol{c}^{(i)} = (c_1^{(i)}, \ldots, c_n^{(i)})$, of length $n$ bits, where we assume for simplicity and ease of exposition that all codes have equal code length $n$ (however, in principle, the choice of the component codes is arbitrary). The code rate of the overall coding scheme is equal to the normalized sum of the individual code rates, i.e.,

$$R = \frac{1}{m}\sum_{i=1}^{m} R_i = \frac{1}{m}\sum_{i=1}^{m} \frac{k_i}{n} = \frac{k}{nm} \,. \tag{7.92}$$

At each time instant $t$, the string of $m$ code bits $c_t^{(1)}, \ldots, c_t^{(m)}$ at the output of the $m$ encoders is mapped to a constellation symbol. As for TCM, the mapping is also derived by successively partitioning the signal set into subsets. Note that each component code has a different rate, i.e., each constellation bit is protected differently. A block level of the multilevel encoder is depicted in Fig. 7.43.

ML decoding of MLC requires joint decoding of all component codes, which is unfeasible. Fortunately, decoding of MLC can be efficiently performed using multistage decoding. The multistage decoder operates by decoding the $m$ component codes separately. More precisely, the component codes $C_i$ are decoded sequentially, starting with code $C_1$. We denote by $\mathcal{D}_i$ the decoder corresponding to component code $C_i$. At stage $i$, $i = 1, \ldots, m$, the decoder $\mathcal{D}_i$ is fed the received sequence and also the decisions of the previous decoding stages, i.e., $\hat{\boldsymbol{u}}^{(1)}, \ldots, \hat{\boldsymbol{u}}^{(i-1)}$ (or equivalently $\hat{\boldsymbol{c}}^{(1)}, \ldots, \hat{\boldsymbol{c}}^{(i-1)}$). The block diagram of the multistage decoder is shown in Fig. 7.44. Multistage decoding achieves good performance in practice, with limited complexity, and it has been shown to achieve the channel capacity [7.234].

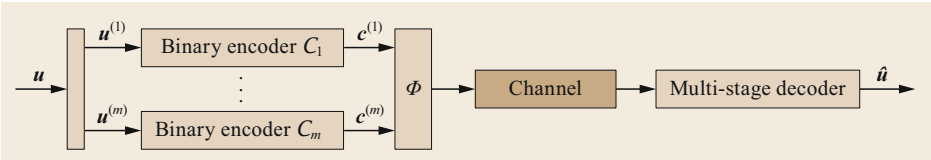


**Fig. 7.43** MLC encoder

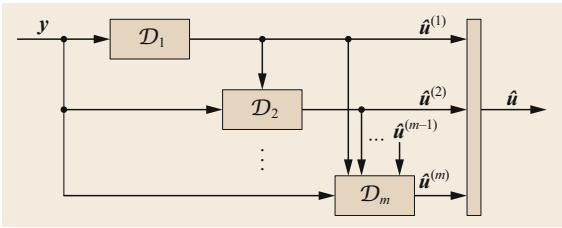


**Fig. 7.44** Multistage decoder for MLC

### 7.6.3 Bit-Interleaved Coded Modulation

BICM was introduced by *Ephraim Zehavi* et al. in 1992 [7.235]. Given its simplicity, flexibility, and capacity-approaching performance, it is nowadays perhaps the most popular CM scheme and has become the de facto standard in many communication standards and modern communication systems.

BICM is a *pragmatic* approach to CM consisting in the concatenation of a (single) binary code with a higher-order constellation through a binary *interleaver* $\pi$. The effect of the interleaver is to uniformly distribute the code bits among the different sub-channels corresponding to the different constellation bits. Therefore, in contrast to TCM and MLC, BICM decouples the code from the modulation, which greatly simplifies the CM scheme design (the code is optimized independently of the modulation).

The block diagram of a BICM encoder and decoder is shown in Fig. 7.45. The information sequence $\boldsymbol{u}$ is first encoded by a binary encoder into codeword $\boldsymbol{c}$, which is interleaved by the interleaver $\pi$. The interleaver permutes the sequence $\boldsymbol{c}$ into another sequence $\tilde{\boldsymbol{c}}$. The interleaved sequence $\tilde{\boldsymbol{c}}$ is then parsed in blocks of $m$ bits each, thus generating $m$ parallel bit streams, denoted by $\boldsymbol{b}^{(1)}, \ldots, \boldsymbol{b}^{(m)}$. At time instant $i$, the modulator $\Phi$ takes $m$ bits $(b_i^{(1)}, \ldots, b_i^{(m)})$, and maps them onto symbols of a constellation $\mathcal{X}$ of cardinality $M = 2^m$ according to the binary labeling of the constellation (usually Gray labeling). The resulting (modulated) sequence $\boldsymbol{x}$ is transmitted over the channel. The sequence at the output of the channel is denoted by $\boldsymbol{y}$.

At the receiver side, for SDD, the demodulator $\Phi^{-1}$ computes soft reliability information about the bits $(b_i^{(1)}, \ldots, b_i^{(m)})$ of the bitstreams $\boldsymbol{b}^{(1)}, \ldots, \boldsymbol{b}^{(m)}$ in the form of LLRs

$$l_i^{(j)} \triangleq \log\left(\frac{p_{Y_i|B_i^{(j)}}(y_i|0)}{p_{Y_i|B_i^{(j)}}(y_i|1)}\right) = \log\left(\frac{\sum_{x\in\mathcal{X}_0^{(j)}} p_{Y_i|X_i}(y_i|x)}{\sum_{x\in\mathcal{X}_1^{(j)}} p_{Y_i|X_i}(y_i|x)}\right), \tag{7.93}$$

where $l_i^{(j)}$ is the LLR for the $j$-th bit at time instant $i$; $X_i$, $Y_i$, and $B_i^{(j)}$ are the RVs corresponding to the channel input $x_i$, channel output $y_i$, and bit $b_i^{(j)}$, respectively; and $\mathcal{X}_0^{(j)} \subset \mathcal{X}$ and $\mathcal{X}_1^{(j)} \subset \mathcal{X}$ are the sub-constellations consisting of all constellation points with binary labeling with a 0 or a 1 in the $j$-th position, respectively, i.e., $\mathcal{X}_z^{(j)} = \{x \in \mathcal{X} : b^{(j)}(x) = z\}$, $z = \{0, 1\}$. The LLRs are then multiplexed, de-interleaved, and fed to a bit-wise soft-decision decoder.

It is also possible to use iterative decoding/demodulation at the receiver. In this case, the decoder and the demodulator exchange extrinsic information in an iterative fashion, and the scheme is referred to as BICM with iterative decoding (BICM-ID) [7.236–238]. Note that, here, *iterative decoding* refers to the iterations between the decoder and the demodulator, and not to the iterative decoding of the binary code. BICM-ID brings performance gains with respect to plain BICM for mappings other than Gray and non-square constellations, at a cost of higher complexity. With Gray labeling and square constellations, there is only a negligible advantage in iterating between the decoder and the demodulator.

For HDD, the demodulator performs minimum distance symbol-by-symbol detection of the received symbols (assuming uniform distribution at the input of the channel) and outputs the binary labeling associated with the detected symbol. The resulting stream of bits (after multiplexing) is de-interleaved and fed to a bit-wise hard-decision decoder.

### 7.6.4 Coded Modulation with Nonbinary Codes

As discussed in Sect. 7.2.1, the MI of nonbinary transmission over the AWGN channel is achievable with

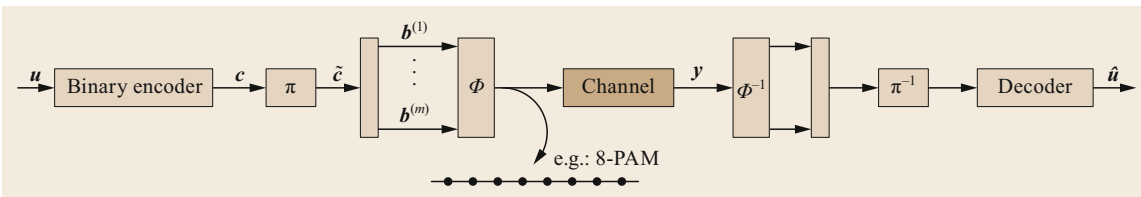


**Fig. 7.45** BICM encoder and decoder

a symbol-wise soft-decision decoder. A natural way to couple an error-correcting code with a higher-order constellation is therefore to consider a nonbinary code that is matched to the constellation size. One can then assign each of the $2^m$ different code symbols to one of the constellation points. Note that in this case, the mapping between code symbols to constellation points becomes trivial. CM with nonbinary LDPC codes is discussed in [7.239].

In practice, the loss with respect to the MI due to the use of a BICM system is very small for most constellations, labeling, and channels in use today. Therefore, since the decoding of nonbinary codes is generally significantly more complex than the decoding of their binary counterparts, it is difficult to justify the use nonbinary codes, and BICM is perhaps the most reasonable CM scheme to consider.

### 7.6.5 Signal Shaping

So far we have assumed nonbinary transmission using a conventional signal constellation with equidistant signal points and uniform signaling, i.e., each signal point is transmitted with the same probability. This is the case for conventional PAM and quadrature amplitude modulation (QAM) constellations used in current fiber-optic systems. However, such constellations exhibit a gap relative to the Shannon limit. The reason is that conventional constellations are not capacity-achieving. For instance, for the AWGN channel, the capacity-achieving distribution is the Gaussian distribution. The use of a discrete signal constellation with equidistant signal points and uniform signaling instead leads to an asymptotic loss of 1.53 dB (asymptotic in the sense of high spectral efficiencies and number of signal points). In Fig. 7.46, the MI curves (see (7.3)) for 4, 8, 16, 32, 64, 128, and 256-QAM constellations are depicted for transmission over the AWGN channel and compared with the channel capacity curve. A gap with respect to the capacity curve is observed.

Thus, to achieve capacity, the use of powerful error-correcting codes is not enough. To reduce the fundamental 1.53 dB gap, constellation-shaping techniques that produce a Gaussian-like distribution, i.e., *shape* the constellation such that it mimics a Gaussian distribution (or the capacity-achieving distribution for a given channel) are required [7.240]. In other words, to achieve capacity, coding and CM techniques must be complemented with signal shaping.

There are two main classes of signal shaping, geometric shaping and probabilistic shaping [7.241, 242]. In geometric shaping, the constellation points are arranged in a non-equally spaced manner to mimic the capacity-achieving distribution. The best constellation

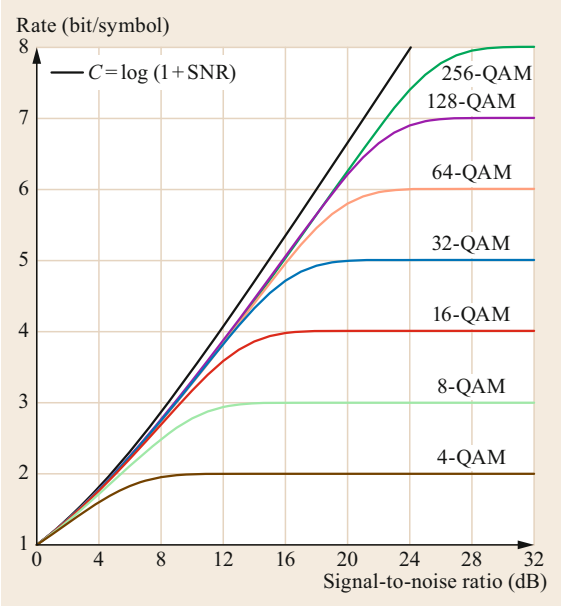


**Fig. 7.46** Mutual information curves for several QAM constellations over the AWGN channel. There is a gap between the MI and the capacity of the channel

depends on the SNR and the decoding metric. Probabilistic shaping, on the other hand, starts with a conventional constellation with equidistant signal points, e.g., PAM or QAM, and assigns different probabilities to different constellation points to induce a Gaussian-like probability distribution. In Fig. 7.47, we plot a probabilistically shaped 8-PAM constellation. The inner signal points have higher probability than the outer signal points, mimicking a Gaussian distribution. A significant advantage of probabilistic shaping is that it builds upon off-the-shelf constellations. In general, geometric shaping requires more complex hardware, such as higher-resolution ADCs, than probabilistic shaping. Furthermore, in [7.243] it was shown that for the AWGN channel and bit-wise decoding, geometric

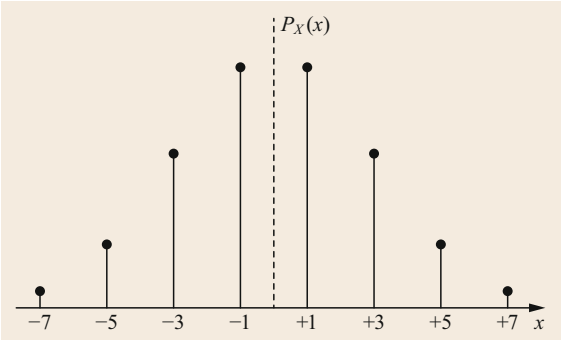


**Fig. 7.47** Probabilistically shaped 8-PAM constellation

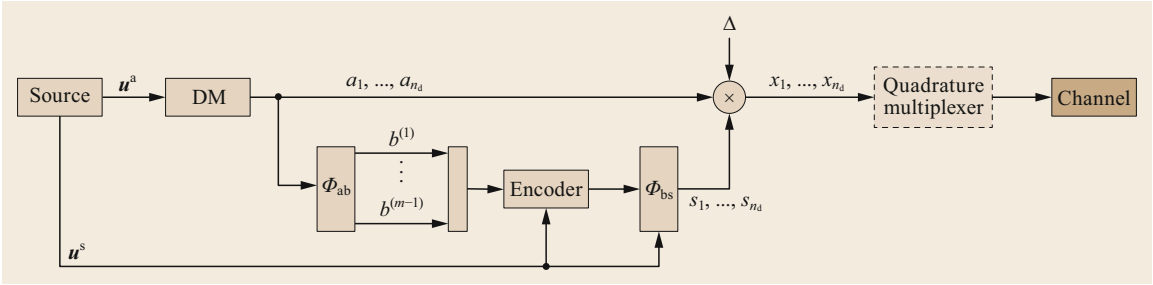


**Fig. 7.48** Block diagram of the PAS scheme

shaping suffers from performance loss compared with probabilistic shaping.

Both geometric and probabilistic shaping have been considered for fiber-optic communications as a means of increasing the spectral efficiency, showing significant gains with respect to classical constellations [7.244–250].

### Probabilistic Amplitude Shaping

A particularly appealing probabilistic shaping scheme, dubbed probabilistic amplitude shaping (PAS), was recently proposed in [7.251]. It achieves performance within 1.1 dB of the capacity of the AWGN channel for a wide range of spectral efficiencies with off-the-shelf LDPC codes [7.251]. More recently, this scheme was considered for fiber-optic communications in [7.246, 248] and is currently receiving a great deal of attention in the optical communications research community. In [7.249, 250], PAS was applied to HDD and adapted for the use of staircase codes.

PAS exploits the fact that, for the AWGN channel (and symmetric channels in general), the capacity-achieving distribution is symmetric. For one-dimensional and two-dimensional constellations, the capacity-achieving distribution is symmetric around zero. This means that the signs of the constellation points are uniformly distributed. A sketch of the capacity-achieving distribution based on a 8-PAM underlying constellation is depicted in Fig. 7.47. The optimal probability mass function is symmetric around zero, i.e., $\Pr(|x|) = \Pr(-|x|)$, and $\Pr(\mathrm{sign}(X) = 1) = \Pr(\mathrm{sign}(X) = -1) = 1/2$. The key idea in PAS is then to move the shaping of the amplitudes *before* the encoding, as opposed to geometric shaping and conventional probabilistic shaping, and use the (uniformly distributed) parity bits at the output of a systematic encoder and some information bits to generate the signs with the desired uniform distribution.

The PAS scheme proposed in [7.251] is depicted in Fig. 7.48. For simplicity, we consider PAM modulation as the underlying modulation, i.e., the channel input alphabet is given by $\mathcal{X} \triangleq \{-2^m + 1, \ldots, -1, 1, \ldots, 2^m - 1\}$, where $m$ is the number of bits per symbol. The information sequence at the output of the source, $\boldsymbol{u} = (u_1, \ldots, u_k)$, where the information bits are uniformly distributed, i.e., $\Pr(U = 0) = \Pr(U = 1) = 1/2$, is split into two sequences $\boldsymbol{u}^s$ and $\boldsymbol{u}^a$. The sequence $\boldsymbol{u}^a$ is used to generate a sequence of amplitudes $\boldsymbol{a} = (a_1, \ldots, a_{n_d})$ with the desired distribution through a so-called distribution matcher (DM) [7.251]. The distribution matcher transforms blocks of uniformly distributed bits into blocks of amplitudes $a_i \in \{1, \ldots, 2^{m-1}\}$ with the desired probability mass function $P_A(a)$. The binary image of the amplitudes $\boldsymbol{b}$ and the remaining information bits, i.e., $\boldsymbol{u}^s$, are then multiplexed and encoded using a binary code (an LDPC code in [7.251]) with a systematic encoder. The parity bits generated by the systematic encoder and $\boldsymbol{u}^s$ are used to generate $n_d$ sign labels $s_1, \ldots, s_{n_d}$. Since the parity bits at the output of the encoder tend to be approximately uniformly distributed, the distribution of the signs is the desired uniform distribution. Finally, the element-wise multiplication of the sequence of amplitudes $(a_1, \ldots, a_{n_d})$ with the sequence of signs $(s_1, \ldots, s_{n_d})$ and with a scaling factor $\Delta$ generates a sequence of symbols $\boldsymbol{x} = (x_1, \ldots, x_{n_d})$ with the desired distribution that is transmitted over the channel. The scaling factor $\Delta$ is chosen to meet the power constraint.

Note that since square QAM constellations can be obtained as the Cartesian product of two PAM constellations, the design of the PAS scheme described above readily extends to QAM constellations. In Fig. 7.48, the dashed block multiplexes two PAM modulated sequences, corresponding to the real and imaginary components of the QAM constellation.

## 7.6.6 Performance Curves

In Fig. 7.49, we give BER results for terminated and tail-biting SC-LDPC codes with SDD and for staircase codes and extended staircase codes with HDD, for an AWGN channel and 64-QAM with BICM. The code rate is $R = 0.75$ for all codes, corresponding to overhead OH = 33%, except for the terminated SC-LDPC

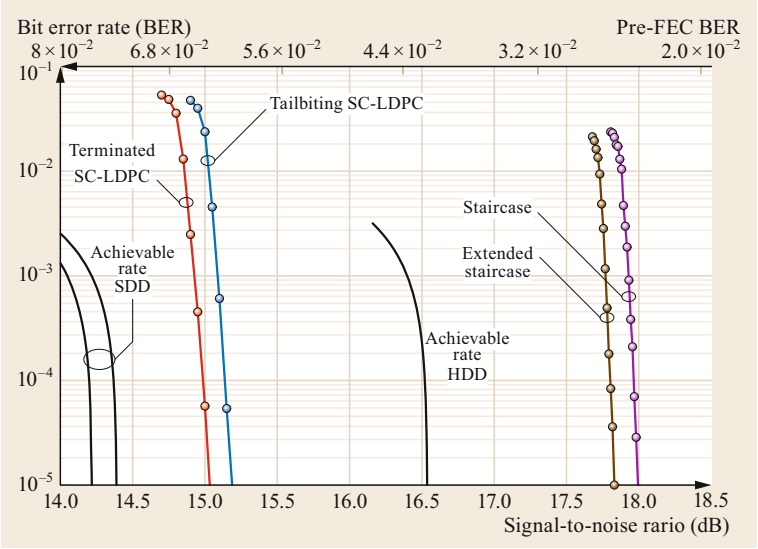


**Fig. 7.49** Simulation results (*solid curves with markers*) and achievable rates (*solid curves*) for a BICM system with SC-LDPC codes, a staircase code, and an extended staircase code for transmission over the AWGN channel with 64-QAM

code, for which the code rate is $R = 0.741$. Staircase codes and extended staircase codes perform at 1.5 dB and 1.3 dB from the achievable rate at a BER of $10^{-5}$. An extra coding gain of 2.6–2.8 dB can be achieved by using SC-LDPC codes and SDD, at a cost of a higher complexity and power consumption. SC-LDPC codes perform around 0.8 dB from capacity. The fact that the staircase code and the extended staircase code perform further from capacity is due to the moderate code rate. As explained in Sect. 7.5, GPCs perform very close to capacity for higher rates.

In Fig. 7.50, we plot the achievable rates for SDD and HDD with a (conventional) uniform 256-QAM constellation and with PAS assuming an underlying 256-QAM for transmission over the AWGN channel, and compare them with the capacity curve $C = \log(1 + \mathrm{SNR})$. Here, we consider a BICM scheme, and the achievable rates are computed as explained in Sects. 7.2.1 and 7.2.2. It is observed that for SDD, transmitting with uniform 256-QAM (dashed blue curve) entails a shaping loss. The solid blue curve is the achievable rate with PAS, which closes the gap relative to the capacity curve significantly, except for the spectral efficiency range, for which the curve bends to the maximum spectral efficiency with 256 signal points, i.e., 8 bits/symbol. The dashed red curve is the achievable rate with HDD and uniform signaling. HDD entails a loss with respect to SDD, and the loss increases for lower spectral efficiencies, as can be seen from the two different slopes of the dashed curves. As for SDD, PAS yields a performance improvement (cf. red dashed and solid curves). Interestingly, the gap with respect to the corresponding achievable rate for SDD (cf. solid red and solid blue curves) is now reduced, and an almost constant gap of about 1.5 dB is observed. Therefore, shaping seems to help in reducing the gap relative to SDD. In the figure, we also plot the performance of practical staircase codes with parameters in [7.249, 250], which are in agreement with the behavior predicted by the achievable rates.

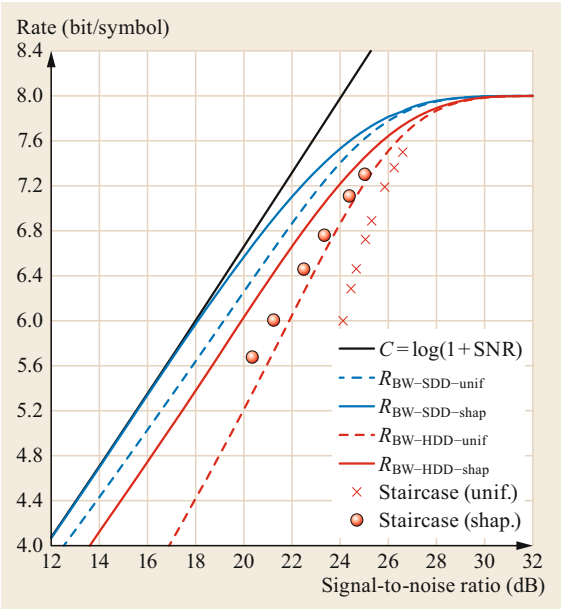


**Fig. 7.50** Achievable rates for SDD and HDD with uniform signaling and shaping and performance of a probabilistically shaped CM scheme using staircase codes and comparison with a BICM system using a staircase code and conventional, uniform signaling. 256-QAM, AWGN channel

## 7.7 Evaluating Forward Error Correction Performance in Transmission Experiments

The design of many optical communication systems often requires the heavy use of transmission experiments to verify models, assumptions, and complete systems. This is largely due to the absence of a rigorous and widely accepted channel model that includes all the effects and impairments of transceivers and optical fibers. Furthermore, in contrast to wireless communications, for example, the fiber-optic communication channel is rather static, enabling relatively easy reproducibility of the experimental results.

Early transmission experiments required specialized hardware such as pattern generators, leading to some constraints on the data to be transmitted. With the absence of ADCs in early optical communication systems, HDD was the dominant decoding method until about 2012 and the advent of commercial systems with SDD [7.252]. Focusing on HDD, a standard solution in system experiments in the 10 and 40 Gbit/s era was to count the bit errors at the receiver (e.g., using a specialized hardware error counter) and to compare this so-called *pre-FEC BER* with the average error-correcting performance of an FEC code with HDD over the BSC. The receiver often consisted of a simple photodiode followed by a high-speed flipflop to sample the photodiode output. This approach assumes that the transmission channel is more or less stationary or is made stationary by a sufficiently large interleaver (to remove, e.g., burst errors). The pre-FEC BER at which the target post-FEC BER was obtained is commonly referred to as the *pre-FEC BER threshold*.

With the advent of ADCs and coherent optical communications, SDD became feasible and quickly became the state-of-the-art decoding scheme in many optical communication systems, especially long-haul and submarine communications. Despite the use of SDD and the absence of a BSC model, the use of a pre-FEC BER threshold was still prevalent in the field for many years. It was first recognized in [7.253] that the pre-FEC BER threshold was not a good performance predictor in systems with differentially detected QPSK followed by an FEC code with SDD. Such systems were dominant in early coherent systems due to a large amount of phase slip from neighboring OOK WDM channels. The authors of [7.253] suggested using a performance prediction threshold that had a better information theoretical foundation and is related to the *achievable rate* of the system (see Sect. 7.2), in that case the MI between code bits and differential detector output.

As coherent optical communication systems matured, and with the introduction of new high-speed digital-to-analog converters (DACs), higher-order modulation formats [7.254], sometimes in combination with probabilistic shaping [7.255], became the state-of-the-art technology for increasing the spectral efficiency of such systems. In transmission experiments, the pre-FEC BER was still often used as threshold despite the fact that the pair $(\mathrm{BER}_{\mathrm{in}}, \mathrm{BER}_{\mathrm{out}})$ does not necessarily fully characterize an FEC code with SDD when the channel, including modulation format, DSP, and quantization, is subject to change. Thus, as previously advocated in [7.253], the authors in [7.256, 257] suggested the use of an information-theoretic measure closely related to the channel and CM scheme utilized. With BICM being the pragmatic choice of CM, this information-theoretic measure is the achievable rate $R_{\mathrm{SDD\text{-}BW}}$ of bit-wise SDD given in (7.8). This quantity is often called GMI, which originates from the theory of mismatched decoding, i.e., the use of a suboptimal decoder [7.87, 88], and is a lower bound of the achievable rate. For BICM, it was shown in [7.97] that the GMI equals the achievable rate $R_{\mathrm{SDD\text{-}BW}}$ ((7.7)–(7.8)). This motivated the authors of [7.256] to refer to $R_{\mathrm{SDD\text{-}BW}}$ as GMI. The term GMI is now ubiquitously used in the field of fiber-optic communications as a proxy for $R_{\mathrm{SDD\text{-}BW}}$. We would like to emphasize here that the GMI is a much broader concept that can be used in a much more general way, and hence we prefer to use the term $R_{\mathrm{SDD\text{-}BW}}$ to denote the achievable rate of BICM.

The authors of [7.256] suggested characterizing a certain FEC code by a pair of achievable rate and output BER $(R_{\mathrm{in}}, \mathrm{BER}_{\mathrm{out}})$ (e.g., $(R_{\mathrm{SDD\text{-}BW}}, \mathrm{BER}_{\mathrm{out}})$), and showed by means of an example based on LDPC codes that this is a fairly good assumption when changing the modulation format but leaving the channel fixed. In [7.258, 259], these results were extended to the case of nonbinary FEC with SDD (i.e., using the pair $(R_{\mathrm{SDD\text{-}SW}}, \mathrm{BER}_{\mathrm{out}})$), another CM scheme that has been promoted for use in fiber-optic communications. At the same time, in [7.259, 260], the use of thresholds as performance predictor in general was questioned, as all threshold-based methods rely on the *universality* of FEC codes, which should be closely checked before employing a threshold-based method.

In this section, we discuss the universality of FEC schemes, introduce the threshold-based performance estimation approach, and discuss the most common thresholds and how to use them in practice. Finally, we show how to best avoid pitfalls of thresholds and how to include FEC in transmission experiments, which gives the most accurate estimates.

### 7.7.1 Threshold-Based Forward Error Correction Performance Prediction

While thresholds are a perfectly fine tool for predicting the performance of some FEC schemes with HDD (under some stationarity assumptions), the use of FEC schemes with SDD together with varying modulation formats and transmission links requires more caution. This is because SDD relies on knowledge of the probabilistic channel model (Fig. 7.1), which is subject to change in the latter scenarios.

#### Forward Error Correction Universality

When assessing and comparing the performance of different modulation formats and transmission scenarios (e.g., fiber types, modulators, converters) based on *thresholds*, it is important to understand the concept of FEC *universality*. An FEC code and decoder pair is said to be *universal* if its performance of the code (measured in terms of post-FEC BER or symbol error rate (SER)) does not depend on the channel, provided that the CM scheme is fixed and the achievable rate of the channel is fixed.

When we refer to *the channel*, we consider the whole transmission chain between the FEC encoder output $\boldsymbol{x}$ and the decoder LLR input $\boldsymbol{l}$, including modulation and demodulation, LLR computation, DSP, ADCs and DACs, optical transmission, filtering, and amplification including noise. We say that the channel changes if any of the components in the chain between $\boldsymbol{x}$ and $\boldsymbol{l}$ changes. This can be for instance the noise spectrum or the OSNR, but also the modulation format or the DSP algorithms. However, we assume that the CM scheme (e.g., BICM) is fixed, as it determines the type of threshold to be used.

Unfortunately, not much is known about the universality of practical coding schemes. It is conjectured that many LDPC codes used in practice are approximately universal [7.261], and some LDPC code ensembles (under some relatively mild conditions) have been shown to be universal in the limit of asymptotically large block lengths [7.262]. Guidelines for designing LDPC codes that show good universality properties are highlighted for instance in [7.263]. The class of SC-LDPC codes has also been shown to be asymptotically universal [7.76]. However, the conditions for achieving this asymptotic universality are relatively stringent, and do not allow much to be said about the universality in the non-asymptotic regime, i.e., for finite batch size $n_\mathrm{b}$, finite coupling width $w$, and finite replication factor $L$. Polar codes (see Sect. 7.4.3) are examples of *nonuniversal* codes: a polar code needs to be redesigned for every different channel. It is worth pointing out, however, that there exists a modification that renders them universal [7.264]. Unfortunately, the required block length to achieve the same performance becomes considerably larger, limiting the practical application of this scheme.

Although most LDPC codes used in practice are asymptotically universal, we wish to emphasize that practical, finite-length realizations of codes may only be approximately universal. For instance, [7.261, Fig. 3] reveals that the performance of some LDPC codes at a BER of $10^{-4}$ differs significantly for different channels. This difference is expected to be even larger at very low BERs because of the different slopes of the curves in the waterfall region.

In the following, we define universality of FEC schemes as in [7.263], with the help of Fig. 7.51.

*Definition 7.11 FEC Universality*

As BICM is ubiquitously used as a CM scheme in fiber-optic communications, we assume BICM and bit-wise SDD. Consider an FEC encoder that generates a codeword consisting of $n$ bits. We transmit these bits over two different communication channels with different (memoryless) channel transition PDFs: channel 1 with PDF $p_{Y_1|X_1}(y_1|x_1)$ and channel 2 with PDF $p_{Y_2|X_2}(y_2|x_2)$. Both channels have *identical* achievable rate $R_{\text{SDD-BW}}$. A fraction $\gamma n$ of the bits is transmitted over channel 1, while the remaining $(1-\gamma)n$ bits are transmitted over channel 2, where $\gamma \in [0, 1]$ such that $\gamma n$ is an integer. We say that a code is *universal* for channels 1 and 2 if the post-FEC BER is independent of $\gamma$.

We can extend this definition to $i$ channels (having the same achievable rate) and say that a code is universal if the post-FEC BER is independent of the fraction of bits transmitted per channel and the channels.

In many practical cases, we generally do not experience any issue with universality, as most often the only change made to the channel is a change in the modulation format, while the underlying fiber and noise model

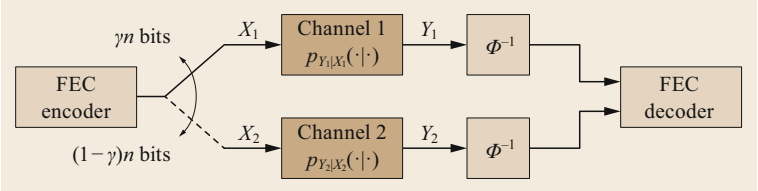


**Fig. 7.51** Definition of universality of FEC schemes according to [7.263]. Channels 1 and 2 have the same achievable rate

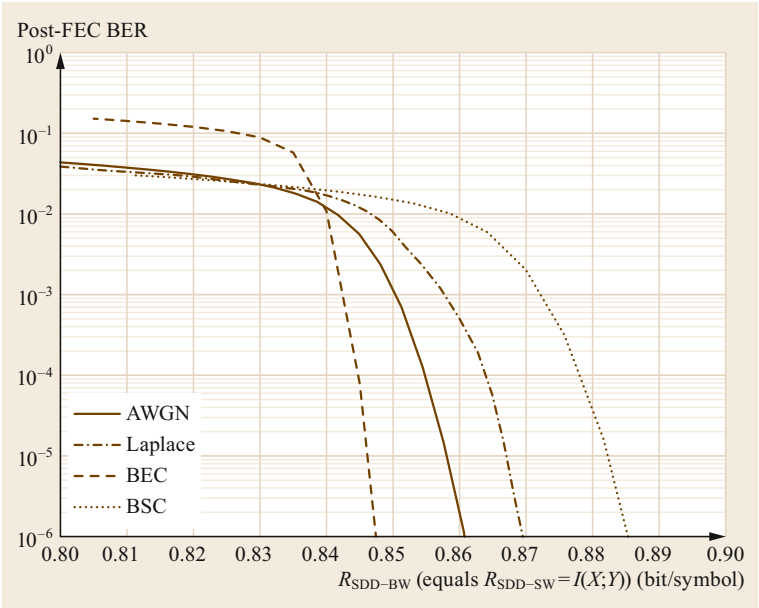


**Fig. 7.52** Illustration of the non-universality of an LDPC code with scaled min-sum decoding

remain relatively constant. The common fiber-optic transmission with coherent reception and no in-line dispersion compensation can be accurately modeled as an AWGN channel. In this case, the codes are approximately universal [7.256, 257, 259], and the achievable rate threshold can serve as an accurate prediction. In some cases, however, the changes to the channel can be more drastic. For example, instead of coherent detection, we could use a simple direct detection (DD) scheme that can lead to significantly different noise distributions. On the other hand, the presence of in-line dispersion compensation might lead to significantly different channel statistics that cannot be easily modeled by AWGN.

In [7.259], the impact of a more drastic change in the channel on code universality was shown for the case of CM with nonbinary LDPC codes: a severe quantization was added at the channel output, and the performance (in terms of gap relative to the achievable rate) of the nonbinary LDPC decoder changed significantly in that case. In the following, we illustrate the concept of non-universality of a common FEC code/decoder by means of two different examples.

In the first example, we consider a regular QC-LDPC code of rate $R = 4/5$ with parameters $d_\mathrm{v} = 3$, $d_\mathrm{c} = 15$, $S = 128$, and $n = 38\,400$. We use this code to transmit codewords over four different channels: the previously introduced binary-input AWGN channel, the BSC, the BEC, and the binary-input Laplace channel. The BEC is a channel with binary input $\mathcal{X} = \{0, 1\}$ and ternary output alphabet $\mathcal{Y} = \{0, ?, 1\}$. The channel transition probabilities are $P_{Y|X}(0|0) = 1-\epsilon$, $P_{Y|X}(?|0) = \epsilon$, $P_{Y|X}(1|0) = 0$, and by symmetry, $P_{Y|X}(1|1) = 1-\epsilon$ and $P_{Y|X}(?|1) = \epsilon$. The channel output is either erased (?) with probability $\epsilon$, or equal to the transmitted bit with probability $1-\epsilon$. The binary-input Laplace channel adds noise (per real dimension) according to the Laplacian distribution

$$p_{Y|X}(y|x) = \frac{1}{2b}\exp\left(-\frac{|y-x|}{b}\right) \quad \text{with } b = \frac{1}{2\sqrt{E_\mathrm{s}/N_0}}\,. \tag{7.94}$$

The channel parameters are configured such that all four channels have the same achievable rate $R_{\mathrm{SDD\text{-}BW}}$ (which for this example is equivalent to $R_{\mathrm{SDD\text{-}SW}}$, as the channel input is binary). After transmission over the channel, we compute LLRs using the true respective channel PDF, i.e., the receiver is matched to the channel. We consider a layered decoder employing the scaled min-sum decoding rule ($\alpha = 0.75$) as described in Sect. 7.4.1 with 10 decoding iterations. The simulation results are shown in Fig. 7.52. We observe not only different behavior of the codes, but also different slopes. Because of the different slopes, the difference will be even larger at low BERs. This offset can be attributed to the fact that the LDPC code utilized is not exactly universal and the code length is relatively small, which is an effect that was also observed in [7.261]. Note that if we are allowed to increase the code length and optimize the degree distribution, as highlighted for instance in [7.263], and use the non-simplified CN update rule (instead of the min-sum simplification),

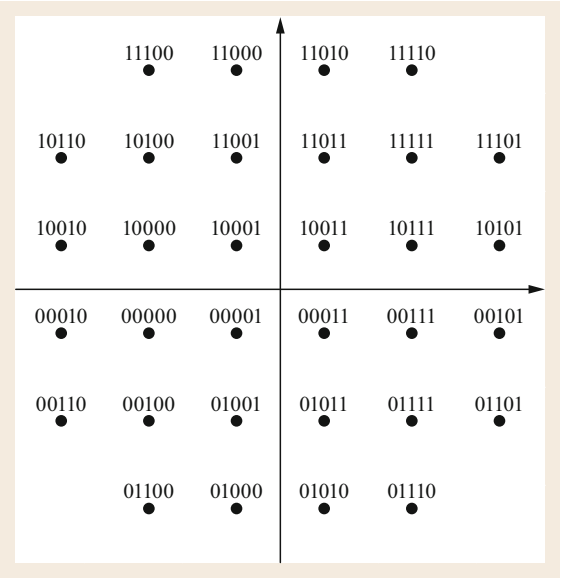


**Fig. 7.53** 32-QAM constellation with bit mapping maximizing $R_{\text{SDD-BW}}$

the performance prediction becomes more accurate again.

Another example is highlighted in [7.253, Fig. 3], where the authors compare a wide range of different WDM systems consisting of either only coherent 100 Gbit/s channels, or of a mix between coherent and OOK channels. Different fiber types are also used. It is shown that the achievable rate (in that case $R_{\text{SDD-SW}}$) serves as a better predictor than the pre-FEC BER, but itself also fails to accurately predict the performance due to changing channel characteristics. We illustrate the lack of universality with another example [7.260]. In this example, we use BICM together with five different constellations: QPSK, 8-QAM (as in [7.259, Fig. 3, $C_3$]), 16-QAM, 32-QAM (with the optimized bit labeling shown in Fig. 7.53), and 64-QAM. We consider transmission over both the AWGN and the Laplace channels using the same code as for the example in Fig. 7.52, and at the receiver employ scaled min-sum decoding ($\alpha = 0.75$) with 10 decoding iterations. The results in terms of normalized $R_{\text{SDD-BW}}/m$ are shown in Fig. 7.54. We can clearly see that even in the case of an AWGN channel, the GMI is only an approximately good threshold of the performance, but if the channel law changes (e.g., a Laplace channel is used), the thresholding effect is compromised, and a significantly higher value of $R_{\text{SDD-BW}}$ is required for decoding. This could lead to significantly misleading conclusions, e.g., in terms of reach prediction.

We hence conclude that performance predictors based on the achievable rate should be used with caution. They can still give rough first-order estimates of the decoding performance, even if we introduce drastic changes into the channel (e.g., a strong quantization, moving from dispersion-uncompensated to dispersion-compensated links, or even from coherent transmission to direct detection systems). Thresholds should therefore be used only to quickly assess the performance and to determine the range of fine measurements. We can improve the accuracy if the channel used to compute the threshold is fairly close to the actual channel of the

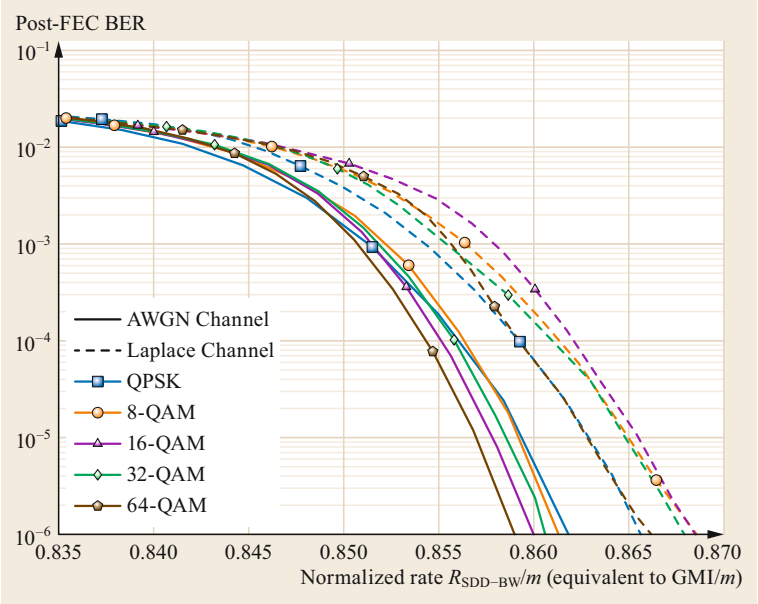


**Fig. 7.54** Illustration of the non-universality with different modulation formats and BICM

system. However, in all cases, actual decoding mimicking as closely as possible the true FEC should be used.

### Summary of Performance Prediction Thresholds and Usage Guidelines

In this section, we present some of the thresholds that are commonly used today, provide guidelines for their usage, and discuss some potential pitfalls. We assume that a transmission experiment has been carried out and that for a particular setup, we have a measurement consisting of $N_M$ data points $(x_\kappa, y_\kappa)$ that are given as the original transmit sequence $x_\kappa \in \mathbb{C}$ and the corresponding received sequence $y_\kappa \in \mathbb{C}$, both consisting of complex modulation symbols. To keep exposition simple, we do not consider 4-D constellation symbols (e.g., dual-polarization modulation formats), but assume an independent treatment of both polarizations. The computation of the achievable rate for 4-D constellations has been addressed in [7.265]. We further assume that both sequences are aligned and that eventual phase jumps have been taken care of (e.g., using pilot symbols). The sequence recovery can be performed either by cross-correlating some or all of the received symbols with the transmit symbols or, if a pseudo-random binary sequence (PRBS) is transmitted, by utilizing a simple PRBS synchronization algorithm.

*Pre-FEC SER.* The pre-FEC SER is perhaps the easiest to compute directly from the experimental measurement database. First, we require a demodulation function. Often, when the channel is Gaussian-like, the nearest-neighbor demodulator (also called Euclidean distance demodulator) is used. The nearest-neighbor demodulator computes estimates of the transmit sequence as

$$\hat{x}_\kappa = \arg\min_{s\in\mathcal{X}} \|y_\kappa - s\|^2 \,. \tag{7.95}$$

Note that the Euclidean distance rule is only optimal for Gaussian noise and should be modified accordingly if the channel behaves differently (e.g., when strong phase noise is present). Using the estimated symbol sequence $\hat{\boldsymbol{x}}$, the pre-FER SER is estimated as

$$\mathrm{SER}_{\mathrm{pre}} = \frac{1}{N_M} \sum_{\kappa=1}^{N_M} \mathbb{1}_{\{\hat{x}_\kappa \neq x_\kappa\}} \,. \tag{7.96}$$

The Pre-FEC SER should be used only as a threshold in particular cases, e.g., when symbol-wise HDD is used and the constellation size is matched to the symbol size of the code. In this case, for a fixed constellation size $M$, the achievable rate $R_{\text{HDD-SW}}$ given by (7.15) is directly linked to the symbol error probability (therein denoted $\delta$), to which the SER converges as $N_M \to \infty$, and has no further dependence on the modulation format (except its size $M$).

*Pre-FEC BER.* The pre-FEC BER was once the most widespread threshold function and still is for FEC schemes with bit-wise HDD. Unfortunately, in many experimental publications, it is not explicitly stated how the pre-FEC BER is computed. Especially if higher-order constellations are used, different computation rules can lead to (slightly) different results.

The computation of the pre-FEC BER is only marginally more complicated than the computation of the pre-FEC SER. Possibly the easiest method is to start from the estimated symbols $\hat{x}_\kappa$, computed using (7.95), and to use these to recover the bit patterns by applying the binary labeling function $\mathsf{L}(\hat{x}_\kappa)$, assuming *implicitly* the use of BICM as CM technique. We denote by $\mathsf{L}(\hat{x}_\kappa) = (\hat{b}_\kappa^{(1)}, \ldots, \hat{b}_\kappa^{(m)})$ the bits that are assigned to the modulation symbol $\hat{x}_\kappa$, and similarly, we denote by $\mathsf{L}(x_\kappa) = (b_\kappa^{(1)}, \ldots, b_\kappa^{(m)})$ the transmit bits associated with $x_\kappa$. The pre-FEC BER $\mathrm{BER}_{\mathrm{pre}}$ is then given by

$$\mathrm{BER}_{\mathrm{pre}} = \frac{1}{mN_M} \sum_{\kappa=1}^{N_M} \sum_{i=1}^{m} \mathbb{1}_{\{b_\kappa^{(i)} \neq \hat{b}_\kappa^{(i)}\}} \,. \tag{7.97}$$

An alternative method for computing the pre-FEC BER is to compare the sign of the LLR $l_\kappa^{(i)}$ given by the bit-wise demodulator $\Phi^{-1}$ with the transmit bit sequence. The two approaches lead to different results. However, for most practical cases and constellations, the differences are negligible.

The pre-FEC BER should be used *only* if bit-wise HDD is utilized and the channel is sufficiently interleaved to remove all burst errors. In this case, the achievable rate $R_{\text{HDD-BW}}$ is directly linked to the average bit error probability (therein denoted $\bar{\epsilon}$), of which the pre-FEC BER is an estimate. The pre-FEC BER threshold can be used in some circumstances with SDD, but only if the FEC that is evaluated has been thoroughly simulated with a model that is sufficiently close to the experimental setup (e.g., using the same modulation format, quantization, fiber model, neighboring channel setup). Only in this case is the use of pre-FEC BER legitimate. Given the effort required to set up such an involved simulation, it may be easier to include the FEC code directly in the experiment.

*Generalized Mutual Information.* In the field of fiber-optic communications, the use of the GMI as decoding threshold has become dominant in the recent years. We would like to point out again that the notion of GMI is a much broader concept, introduced as

a bound on the achievable rate for mismatched decoding [7.87, 88]. In the case of BICM, the GMI equals the achievable rate $R_{\text{SDD-BW}}$. Hence, the term GMI is often used in a somewhat inaccurate way, interchangeably with the achievable rate of BICM.

For computing the GMI, we assume that all constellation symbols are equiprobable. There are multiple ways to compute the GMI [7.266]. We give here the one that we believe is easiest. In a first step, we use the bit-wise demodulator $\Phi^{-1}$ given by (7.93) to compute $m$ LLRs per received symbol, i.e., $\Phi^{-1}(y_\kappa) = (l_\kappa^{(1)}, \ldots, l_\kappa^{(m)})$. Similarly, we compute the corresponding bit patterns of the transmit sequence using $\mathsf{L}(x_\kappa) = (b_\kappa^{(1)}, \ldots, b_\kappa^{(m)})$. The *bit-wise* GMI threshold can then be computed directly from the LLRs as

$$\text{GMI}_{\text{SDD-BW}} = \underbrace{\log_2 |\mathcal{X}|}_{=m} - \frac{1}{N_M} \inf_{s\geq 0} \sum_{i=1}^{m} \sum_{\kappa=1}^{N_m} \log_2\left(1 + \mathrm{e}^{s(-1)^{b_\kappa^{(i)}} l_\kappa^{(i)}}\right). \tag{7.98}$$

The computation includes a minimization (inf) over the variable $s \geq 0$. When the computation of the LLRs (7.93) is carried out using *exactly* the same channel PDF $p_{Y_i|B_i^{(j)}}(y_i|b_i^{(j)})$ that was used for transmission (e.g., in a simulation), the minimum is obtained for $s = 1$, and the minimization does not need to be carried out. In any other case, due to the unimodality of the objective function, the minimization can be easily carried out using, for example, the golden section search or built-in minimization functions of numerical software packages (e.g., `fminbnd` of MATLAB®).

The threshold $\text{GMI}_{\text{SDD-BW}}$ is an estimate of the achievable rate $R_{\text{SDD-BW}}$ and hence should be used *only* for FEC with bit-wise SDD (i.e., in the case of BICM) and if there is sufficient evidence that the code is approximately *universal* or, alternatively, if the GMI threshold of the code has been determined in a simulation mimicking sufficiently closely the transmission experiment, such that the code's lack of universality does not cause a difference.

PAS has recently become an attractive solution for realizing probabilistic constellation shaping and has found widespread use in optical communications. PAS is not a BICM scheme, but still uses a bit-wise demodulator $\Phi^{-1}$ that computes LLRs according to

$$l_i^{(j)} \triangleq \log\left(\frac{\sum_{x\in\mathcal{X}_0^{(j)}} p_{Y_i|\mathsf{X}_i}(y_i|x)P_X(x)}{\sum_{x\in\mathcal{X}_1^{(j)}} p_{Y_i|\mathsf{X}_i}(y_i|x)P_X(x)}\right), \tag{7.99}$$

with $P_X(x)$ being the prior distribution of using constellation symbol $x$. Applying the LLR $l_i^{(j)}$ given in (7.99) in (7.98) yields a value that we denote as $\bar{R}_{\text{SDD-BW}}$ and that was used in [7.267, Eq. (6)] to compute the so-called normalized generalized mutual information (NGMI), which has been shown to serve as a relatively accurate performance threshold. Note, however, that the NGMI should not be confused with the notion of GMI (despite the similarity in the name). For details regarding achievable rates of PAS, we refer the interested reader to [7.91].

*Mutual Information.* The estimation of the MI threshold $\text{MI}_{\text{SDD-SW}}$ can be subdivided into two steps. In a first step, we estimate the actual channel PDF $p_{Y|X}(y|x)$ as closely as possible neglecting potential memory effects. We denote the estimated channel PDF by $q_{Y|X}(y|x)$. The PDF can be modeled using kernel density estimators, histograms, and piecewise approximations, or using Gaussian mixture models (GMMs). In the latter case, using $g$ mixtures, for every constellation symbol $\mathsf{X}_i \in \mathcal{X}$, the estimate is written as

$$q_{Y|X}(y|X = \mathsf{X}_i) = \sum_{j=1}^{g} w_{i,j} \mathcal{CN}(\mu_{i,j}, \sigma_{i,j}^2),$$
$$\text{with } \mathcal{CN}(\mu, \sigma^2) = \frac{1}{\pi\sigma^2} \exp\left(-\frac{\|y-\mu\|^2}{\sigma^2}\right), \tag{7.100}$$

i.e., $\mathcal{CN}(\mu, \sigma^2)$ is the complex, circularly symmetric Gaussian PDF with (complex) mean $\mu$ and variance $\sigma^2$. The PDF $q_{Y|X}$ is parameterized by $Mg$ 4-tuples $(w_{i,j}, \text{Re}\{\mu_{i,j}\}, \text{Im}\{\mu_{i,j}\}, \sigma_{i,j}^2)$ that need to be estimated from the measurement. The $4Mg$ parameters of the model can be estimated for instance using the expectation-maximization (EM) algorithm or built-in estimators of numerical computing environments (e.g., `fitgmdist` of MATLAB®). The MI threshold can then be estimated in a similar way as the GMI

$$\text{MI}_{\text{SDD-SW}} = \frac{1}{N_m} \sup_{s\geq 0} \sum_{\kappa=1}^{N_m} \log_2\left(\frac{M\left[q_{Y|X}(y_\kappa|x_\kappa)\right]^s}{\sum_{x'\in\mathcal{X}}\left[q_{Y|X}(y_\kappa|x')\right]^s}\right). \tag{7.101}$$

The computation includes a maximization (sup) over the variable $s \geq 0$, which can be easily carried out due to the unimodality of the objective function. If $q_{Y|X}(y|x)$ matches the true channel PDF $p_{Y|X}(y|x)$, the maximum is obtained for $s = 1$.

Often, especially in dispersion-uncompensated coherent fiber-optic communications, the GMM is not necessary, and a 2-D Gaussian PDF approximates the

true channel PDF sufficiently well as predicted by the GN model [7.99, 100] and its extensions. Such a 2-D Gaussian PDF yields good estimates of the MI, neglecting any cross-polarization and memory effects. In some circumstances, a 4-D Gaussian PDF can also be used [7.259, 265]. In optical receivers, it is usually not feasible to implement the GMM of the noise, and a complex circularly symmetric Gaussian PDF is usually used as a simple estimate of the true PDF. In this case, we can use a GMI-like argument to compute the threshold

$$\mathrm{MI}_{\text{SDD-SW}} = \frac{1}{N_m} \sup_{s\geq 0} \sum_{\kappa=1}^{N_m} \log_2 \left( \frac{M \exp(-s\|y_\kappa - x_\kappa\|^2)}{\sum_{x'\in\mathcal{X}} \exp(-s\|y_\kappa - x'\|^2)} \right). \tag{7.102}$$

Note that the maximization (sup) over $s$ implicitly estimates the variance of the Gaussian PDF as $\sigma^2 = s_{\max}^{-1}$ [7.259, Example 2].

The MI threshold $\mathrm{MI}_{\text{SDD-SW}}$, which approximates $R_{\text{SDD-SW}}$, should be used only in cases where $R_{\text{SDD-SW}}$ is actually an achievable rate of the transmission system. This includes the use of nonbinary codes, e.g., nonbinary LDPC codes matched to the modulation format as CM scheme, with SDD. The MI threshold can also be used when multilevel coding with multistage decoding is employed as CM scheme. However, the latter is an example of a nonuniversal CM scheme.

Note that in many cases, the values of $\mathrm{MI}_{\text{SDD-SW}}$ and $\mathrm{GMI}_{\text{SDD-BW}}$ are relatively close (e.g., BICM with square QAM constellations and Gray coding, and PAS [7.255]). In these cases, the GMI threshold $\mathrm{GMI}_{\text{SDD-BW}}$ can be replaced by $\mathrm{MI}_{\text{SDD-SW}}$ without losing too much accuracy. This has advantages, as the MI is often easier to compute in the experiment. Computing the MI does not require the implementation of a bit-wise demapper, nor does it need the definition of a bit mapping. The MI can be directly computed from the complex samples at the output of the transmission experiments. In [7.268], MI and GMI thresholds were compared for a field system based on PAS, and were found to be very close.

### 7.7.2 Implementing Forward Error Correction in Experiments

In view of the pitfalls inherent in the use of thresholds, especially when there is uncertainty as to the universality of the FEC and CM scheme used, we suggest including FEC in the transmission experiments. In particular, the use of offline DSP in modern transmission experiments facilitates the inclusion of at least a basic offline FEC decoding in the evaluation phase of the experiments. This is additionally made easy by the availability, for example, of integrated LDPC encoding and decoding routines in common computational software such as MATLAB®. Essentially, there are two main possibilities for including FEC with actual decoding in transmission experiments, which we discuss in the following.

#### Implementing Complete Encoding and Decoding

The closest approximation to the true system performance is achieved by implementing the complete physical layer including FEC encoding and decoding. For this, we use an encoder that generates codewords $\boldsymbol{c}$ (for instance using random information words $\boldsymbol{u}$) which are then fed to the transmitter. After reception, DSP is carried out, and following frame alignment, decoding can be performed. The recovered information words can be compared with the original information words to estimate the post-FEC BER. While this approach should be used whenever possible, it has some drawbacks:

- In some transmission experiments, no programmable DAC is available, and the transmit sequence cannot be freely chosen, but is fixed and limited to some PRBS. In other cases, the memory depth of the DAC may be limited, such that a whole codeword cannot fit into the memory (especially if long SC-LDPC codes with large replication factor $L$ are chosen).
- It is often possible that the code design will not yet be completed when transmission experiments are carried out, and the code designers would like to use the results of the transmission experiment to estimate channel characteristics in order to best adapt their code to the channel. This chicken-and-egg problem requires other approaches.

In both cases, we need to decouple the transmission experiment from the evaluation of the FEC performance. This is highlighted in what follows.

#### Implementing Decoding Using a Database of Measurements

In [7.269], the authors suggested reusing a database of measurements to evaluate the performance of multiple FEC schemes. Essentially the same method was later proposed in [7.270], which however requires a modification of the decoder and introduces some extra interleaving to enlarge the set of sequences, neglecting possible memory effects.

In the following, we describe the method reported in [7.269] that integrates the evaluation of FEC into the

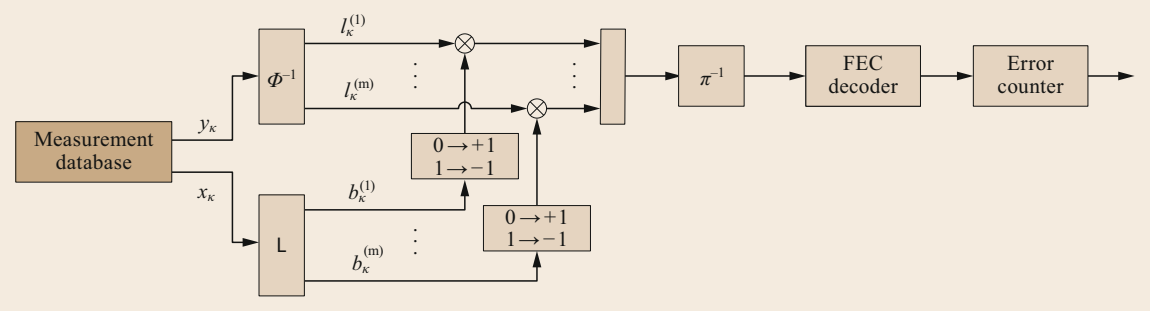


**Fig. 7.55** Evaluating FEC performance from measurements for a BICM-based CM system

receiver processing chain in order to evaluate the performance of different codes. This method is based on the fact that the post-FEC BER performance of most practically employed FEC codes and, most importantly, their decoders, does *not* depend on the transmitted codeword, but only on the noise pattern. This property holds for LDPC and SC-LDPC codes together with the decoders that we introduced in Sect. 7.4 (see [7.111, Chap. 4] for a detailed discussion).

Our method is based on channel adapters introduced in [7.271] for analyzing CM schemes. In order to devise a strategy for performance assessment, we assume that the coding scheme to be tested generates one (or several) valid codewords. For simplicity, if linear codes are used, we can use the all-zero codeword $\boldsymbol{c} = (0, 0, \ldots, 0)$, which belongs to any linear code.

We explain the method for the case with BICM as CM scheme, but we stress that the method can be easily extended to other CM schemes (e.g., PAS) as well. The method is illustrated in Fig. 7.55. The first step of the method consists in generating an equivalent bitstream of length $mN_M$ corresponding to the transmit sequence from the experimental database by

$$\begin{aligned} &\boldsymbol{b} = (\boldsymbol{b}_1, \boldsymbol{b}_2, \ldots, \boldsymbol{b}_{N_M}), \\ &\quad \text{with } \boldsymbol{b}_\kappa = (b_\kappa^{(1)}, \ldots, b_\kappa^{(m)}) = \mathsf{L}(x_\kappa)\,. \end{aligned} \tag{7.103}$$

Similarly, we compute a sequence of LLRs for the received samples $y_k$,

$$\begin{aligned} &\boldsymbol{l} = (\boldsymbol{l}_1, \boldsymbol{l}_2, \ldots, \boldsymbol{l}_{N_M}), \\ &\quad \text{with } \boldsymbol{l}_\kappa = (l_\kappa^{(1)}, \ldots, l_\kappa^{(m)}) = \Phi^{-1}(y_\kappa)\,. \end{aligned} \tag{7.104}$$

Using both sequences, we generate a set of equivalent LLRs, corresponding to the transmission of the all-zero codeword as

$$\begin{aligned} &\tilde{\boldsymbol{l}} = (\tilde{\boldsymbol{l}}_1, \ldots, \tilde{\boldsymbol{l}}_{N_M}) \\ &\quad \text{with } \tilde{\boldsymbol{l}}_\kappa = \left(\tilde{l}_\kappa^{(1)}, \ldots, \tilde{l}_\kappa^{(m)}\right) \\ &\qquad = \left((-1)^{b_\kappa^{(1)}} l_\kappa^{(1)}, \ldots, (-1)^{b_\kappa^{(m)}} \tilde{l}_\kappa^{(m)}\right). \end{aligned} \tag{7.105}$$

This approach corresponds to transmitting the all-zero codeword $\boldsymbol{c}$, which is scrambled using a (time-varying, data-dependent) binary scrambler. This scrambler generates, by modulo-2 addition (i.e., XOR) of a scrambling sequence, the desired transmit bit sequence $\boldsymbol{b}$. This sequence corresponds to the sequence which, after modulation, yields the transmitted symbol sequence used in the experiment.

The sequence $\tilde{\boldsymbol{l}} = (\tilde{l}_1, \ldots, \tilde{l}_{mN_M})$ now consists of $mN_M$ LLRs that can be fed to an FEC decoder. If $mN_M \gg n$, with $n$ being the codeword length, we can partition the sequence $\tilde{\boldsymbol{l}}$ into subsequences of length $n$, by selecting $\tilde{n} \triangleq \lceil n/m \rceil$ consecutive vectors $\tilde{\boldsymbol{l}}_i$ and permuting the resulting sequence of LLRs with a de-interleaver $\pi^{-1}$. Note that in BICM with LDPC codes, interleaving is sometimes not explicitly carried out but assumed to be implicitly part of the code. However, when QC-LDPC codes are used, care must be taken, and we advise the reader to always explicitly carry out interleaving. The de-interleaved vectors are then fed into the FEC decoder, $f_{\mathrm{FEC,dec}}$, which outputs a binary sequence corresponding to the estimated codeword $\hat{\boldsymbol{c}}$. The post-FEC BER $\mathrm{BER}_{\mathrm{post}}$ is then obtained by counting the errors after decoding a sufficiently large number of such frames.

Often, the number of possible measurements is not large enough to obtain sufficiently accurate post-FEC BER estimates. In this case, we can extend the estimation using the interleaver method proposed in [7.270]. However, note that this method assumes that at the receiver, long interleaving across multiple codewords is carried out, breaking all correlation between neighboring symbols that arise due to the interplay of chromatic dispersion and fiber nonlinearities. Such a long interleaving will mask any nonstationary effects and may thus lead to overly optimistic performance estimates. Short burst errors, which can be caused for instance by DSP algorithms that cannot track phase or polarization changes during transmission, and cause uncorrectable blocks (UCBs), will not be captured by such an approach. Our advice is therefore to record sufficiently

long sequences and treat the sequences like a continuous bitstream.

### 7.7.3 Performance Example

To experimentally verify the proposed method, we consider a system using 8-QAM together with BICM. We reuse the measurement database that was used in [7.259] to evaluate nonbinary LDPC codes. We use two 8-QAM constellations $\mathcal{X}_1$ and $\mathcal{X}_2$, shown in Fig. 7.56b and c, together with the bit labeling that maximizes the GMI [7.272]. For illustration purposes, we use three quasi-cyclic regular LDPC codes with rates 0.8, 0.85, and 0.9 (corresponding to FEC overheads of $\approx 25\%$, 18%, and 11%, respectively) with VN degree $d_v = 3$, CN degree $d_c \in \{15, 20, 30\}$, and lifting factor $S = 128$. Each code has length $n = 38\,400$, i.e., 12 800 8-QAM symbols are always mapped to one LDPC codeword. Decoding takes place using $I = 10$ iterations with a layered scaled min-sum decoder ($\alpha = 0.75$). We first test the performance of the three LDPC codes in an AWGN channel. To this end, we first calculate $R_{\text{SDD-BW}}$ for the two constellations $\mathcal{X}_1$ and $\mathcal{X}_2$ as a function of $E_s/N_0$. This is achieved, for example, by applying the estimator given in (7.98) to the results of a simulation over an AWGN channel.

In Fig. 7.56a, we show the post-FEC BER as a function of $R_{\text{SDD-BW}}$. Changing the constellation for a given code can be interpreted as changing the equivalent channel. We see that the codes act approximately universally, and their performance only marginally depends on the chosen constellation. For a post-FEC BER of $10^{-4}$ (horizontal line), we find the equivalent achievable rate thresholds $T_R$ for each rate.

To validate the AWGN results in Fig. 7.56a, we now consider a coherent fiber-optic communication system based on dual-polarization transmission at a symbol rate of 41.6 Gbaud. First, symbol sequences corresponding to constellation $\mathcal{X}_1$ are generated and tested using a high-speed DAC in a back-to-back configuration. A root-raised cosine pulse shaping signal (roll-off factor 0.1) is generated as described in [7.272], and two code rates ($R = 0.8$ and $R = 0.85$) are chosen, giving net data rates of approximately 200 and 212 Gbit/s, respectively.

The empirical achievable rate estimates $\text{GMI}_{\text{SDD-BW}}$, computed using (7.98), are shown as a function of the OSNR in Fig. 7.57a. We also show the GMI thresholds $T_{0.8} = 2.57$ and $T_{0.85} = 2.69$. These thresholds are then used to determine equivalent OSNR thresholds (vertical lines). The measured data are then used to perform LDPC code decoding using the method described in Sect. 7.7.2. The obtained results are shown in Fig. 7.57b by solid markers. Additionally, from the estimated achievable rate values, we interpolate the estimated post-FEC BER values using the AWGN simulation results from Fig. 7.56a, which are given by the thin dotted lines. We observe an agreement between the predicted post-FEC BER and actual post-FEC BER. However, we also note that the performance of the actual decoding deviates from the prediction, especially at low error rates. This is mostly due to some non-stationarity of the measurements, with some frames that are more affected by the noise than others and cannot be reliably decoded.

In order to show that the proposed method also works for a transmission over a link, we apply the method to a transmission experiment using both constellations

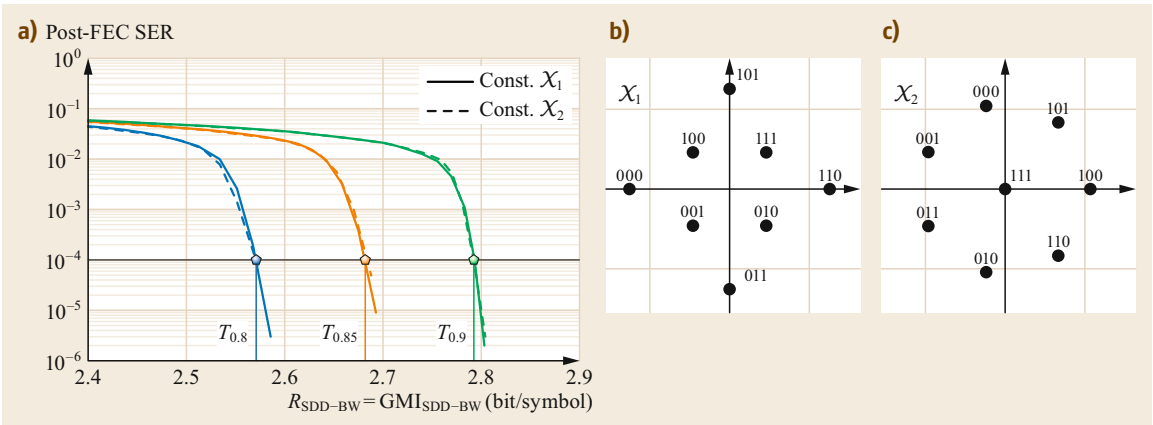


**Fig. 7.56a–c** AWGN simulation results of regular LDPC codes of rates $R \in \{0.8, 0.85, 0.9\}$ with $d_v = 3$ and $d_c \in \{15, 20, 30\}$ with layered normalized min-sum decoding ($\alpha = 0.75$) and $I = 10$ decoding iterations **(a)**. Constellations utilized with GMI-maximizing bit labeling $(b_1, b_2, b_3)$ **(b,c)**. **(a)** Simulation results; **(b)** constellation $\mathcal{X}_1$; **(c)** constellation $\mathcal{X}_2$

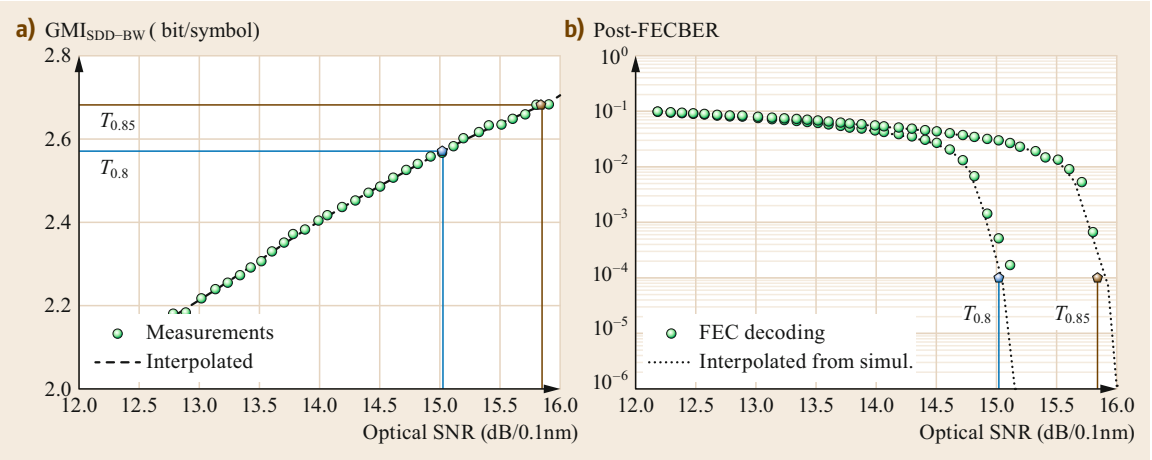


**Fig. 7.57a,b** Back-to-back results. Empirically estimated values of $\mathrm{GMI}_{\mathrm{SDD\text{-}BW}}$ and interpolated values together with the thresholds to achieve a post-FEC BER of $10^{-4}$ with ($d_v = 3$, $d_c \in \{15, 20\}$) regular LDPC codes and layered scaled-min-sum decoding ($\alpha = 0.75$) **(a)** and interpolated post-FEC BER (from AWGN simulations with the same setup as that of Fig. 7.54) together with actual decoding **(b)**

$X_1$ and $X_2$ over a recirculating loop, described in detail in [7.273]. We recapitulate the experimental setup in the following. The transmission testbed consists of one narrow-linewidth laser under test at 1545.72 nm, and additionally 63 loading channels spaced by 50 GHz. The output of the laser under test is sent into a dual-polarization I/Q modulator driven by a pair of DACs operating at 65 Gsamples/s. Multiple delayed-decorrelated sequences of $2^{15}$ bits are used to generate the multilevel drive signals. Pilot symbols and a sequence for frame synchronization are additionally inserted.

The symbol sequences are oversampled by a factor of $\approx 1.56$ and pulse-shaped by a root-raised cosine function with roll-off of 0.1. The load channels are separated into odd and even sets of channels and modulated independently with the same constellation as the channel under test using separate I/Q modulators. Odd and even sets are then polarization-multiplexed by dividing, decorrelating, and recombining through a polarization beam combiner with an approximate delay of 10 ns. The test channel and the loading channels are passed into separate low-speed ($< 10$ Hz) polarization scramblers and spectrally combined through a wavelength-selective switch (WSS). The resulting multiplex is boosted through a single-stage EDFA and sent into the recirculating loop. The loop consists of four 100 km-long dispersion-uncompensated spans of standard single-mode fiber with hybrid Raman–EDFA optical repeaters to compensate for the fiber loss. The Raman pre-amplifier is designed to provide $\approx 10$ dB on–off gain. Loop-synchronous polarization scrambling is used, and power equalization is performed by a 50 GHz-grid WSS inserted at the end of the loop.

At the receiver side, the channel under test is selected by a tunable filter and sent into a dual-polarization coherent receiver feeding four balanced photodiodes. Their electrical signals are sampled at 80 Gsamples/s by a real-time digital oscilloscope with 33 GHz electrical bandwidth. For each measurement, five different sets of 20 µs are stored. The received samples are processed offline by a DSP, including chromatic dispersion compensation, polarization demultiplexing by a 25 tap $T/2$ spaced butterfly equalizer with blind adaptation based on a multi-modulus algorithm, frequency recovery using the fourth and seventh power (depending on the constellation) periodogram, and phase recovery using the blind phase search algorithm. In the latter, equally spaced test phases in the interval $[-\pi/4; \pi/4]$ (constellation $X_1$) or in the interval $[-\pi/7; \pi/7]$ (constellation $X_2$) are used with correspondingly modified phase unwrapping.

We consider transmission over a distance of 3200 km, corresponding to eight round trips in the recirculating loop. Figure 7.58a shows the estimated $\mathrm{GMI}_{\mathrm{SDD\text{-}BW}}$ as a function of the input power $P_{\mathrm{in}}$ per WDM channel [7.273, Fig. 3a]. Additionally, we show the achievable rate thresholds $T_R$ for $R \in \{0.8, 0.85, 0.9\}$. The thresholds give us the region of launch powers at which reliable transmission is possible. To be precise, whenever the estimated achievable rate lies above the threshold $T_R$, it means that transmission with a post-FEC BER below $10^{-4}$ is possible. For example, consider the horizontal line in Fig. 7.58a corresponding to $T_{0.9}$. We can see that with constellation $X_2$, we are just barely above the line for $P_{\mathrm{in}} \in \{-2\,\mathrm{dBm}, -1\,\mathrm{dBm}\}$, which means that decoding is also

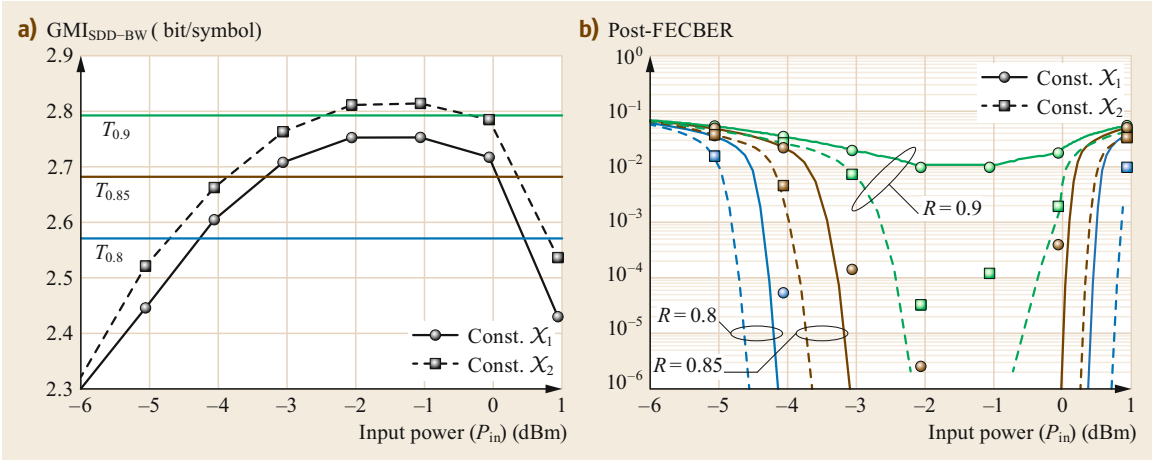


**Fig. 7.58a,b** Transmission results. Estimated $\text{GMI}_{\text{SDD-BW}}$ for two constellations $\mathcal{X}_1$ and $\mathcal{X}_2$ together with thresholds to achieve a post-FEC BER of $10^{-4}$ with LDPC codes ($d_v = 3$, $d_c \in \{15, 20, 30\}$, rates $R \in \{0.8, 0.85, 0.9\}$) with layered scaled-min-sum decoding ($\alpha = 0.75$) **(a)** and interpolated post-FEC BER (from AWGN simulations with the same setup as in Fig. 7.54) together with actual decoding given by markers **(b)**

only barely possible. Contrarily, with constellation $\mathcal{X}_1$, we do not cross the line, and reliable decoding is not possible.

In Fig. 7.58b, we use the simulation results of Fig. 7.56a to estimate the post-FEC performance of the transmission system by interpolation. The interpolated curves are given by the solid (constellation $\mathcal{X}_1$) and dash-dotted (constellation $\mathcal{X}_2$) lines. Additionally, we carried out actual decoding using the LDPC codes introduced before and the method of Sect. 7.7.2. The post-FEC BER results after decoding are given by the solid markers in the figure. We can see that the estimates from interpolation match the actual decoding performance, especially for high BERs. However, we see a deviation at low BERs around $10^{-4}$, which is caused by non-stationarity in the measurements. For instance, at $P_{in}$ of $-1$ dB, a burst caused by the DSP not following the polarization rotation quickly enough occured. The average $\text{GMI}_{\text{SDD-BW}}$ was still below the threshold. However, the burst causes a frame that cannot be decoded by the LDPC decoder and hence yields a post-FEC BER above $10^{-4}$.



## 7.8 Conclusion and Outlook

A vast body of research on FEC exists today, spanning over seven decades. For most of the classical communication channels, coding schemes that asymptotically achieve the theoretical capacity limits are already available. There also exist very powerful coding schemes that yield excellent performance for short block lengths. Surprisingly, however, despite being a relatively old field, and notwithstanding the enormous successes achieved over the past decades, coding is still a very vibrant research topic. One of the main reasons is that it transcends the classical transmission problem. Today, coding plays a crucial role in myriad applications, from distributed storage, caching and computing, to privacy and post-quantum cryptography. We are convinced that coding will continue to be an indispensable tool for future applications, many of which we cannot even envisage, and expertise and frontline research in this area will certainly be necessary.

For the classical data transmission problem, while most of the major theoretical breakthroughs and code constructions are probably behind us, there are still important open challenges to be addressed. This is particularly true in the area of optical communications, which poses severe constraints in terms of complexity, throughput, and power consumption. For example, finding the best scheme yielding the highest coding gains under heavy complexity constraints, and having a highly parallelizable, efficient decoder hardware architecture enabling high data throughput, remains a very challenging engineering task.

Future research on FEC for optical communications will likely focus on low-complexity coding schemes and

**Table 7.5** Comparison of state-of-the-art FEC codes

| Name | Ref. | Dec. | Rate $R$ (OH) | NCG (dB) | $\mathrm{BER_{pre}}$ | $\mathrm{GMI_{SDD\text{-}BW}}$ |
|---|---|---|---|---|---|---|
| LDPC super FEC | [7.46, App. I.6] | HDD | 0.937 (6.7%) | 8.02[d] | 0.00112 | — |
| RS+Hamm. prod. | [7.46, App. I.5] | HDD | 0.937 (6.7%) | 8.5 | 0.0019 | — |
| Concat. BCH | [7.46, App. I.3] | HDD | 0.937 (6.7%) | 8.99 | 0.00310 | — |
| SP-BCH | [7.274] | HDD | 0.937 (6.7%) | 9.4[a] | 0.0045 | — |
| Staircase | [7.49] | HDD | 0.937 (6.7%) | 9.41 | 0.00465 | — |
| Swizzle | [7.275] | HDD | 0.937 (6.7%) | 9.45 | 0.00484 | — |
| Braided BCH | [7.50] | HDD | 0.937 (6.7%) | 9.48[e] | 0.00498 | — |
| Concat. BCH | [7.276] | HDD | 0.936 (6.8%) | 8.91 | 0.0029 | — |
| Orth. concat. BCH | [7.46, App. I.7] | HDD | 0.912 (9.6%) | 9.19[f] | 0.00444 | — |
| CI-BCH | [7.275, 277] | HDD | 0.893 (12%) | 10.0 | 0.00876 | — |
| CI-BCH | [7.275, 277] | HDD | 5/6 (20%) | 10.5 | 0.01524 | — |
| Orth. concat. BCH | [7.46, App. I.7] | HDD | 0.805 (24.3%) | 10.06[f] | 0.01302 | — |
| Algebraic LDPC | [7.278] | SDD | 0.952 (5.05%) | 9.35[c] | 0.0041 | 0.9834 |
| RS+Hamm. prod. | [7.46, App. I.5] | SDD[g] | 0.937 (6.7%) | 9.4 | 0.0045 | 0.9817 |
| Viasat TPC | [7.279] | SDD | 0.935 (7%) | 10.2 | 0.0087 | 0.9654 |
| 400G ZR[h] | [7.280] | SDD | 0.871 (14.8%) | 10.4 | 0.0125 | 0.9510 |
| Viasat TPC | [7.281] | SDD | 0.87 (15%) | 11.0 | 0.0183 | 0.9290 |
| Low-power LDPC | [7.282] | SDD | 5/6 (20%) | 9.5 | 0.0075 | 0.9701 |
| Impr. low-power LDPC | [7.283] | SDD | 5/6 (20%) | 11.0[a] | 0.0204 | 0.9216 |
| LDPC | [7.86] | SDD | 5/6 (20%) | 11.24[b] | 0.0233 | 0.9110 |
| LDPC | [7.64] | SDD | 5/6 (20%) | 11.3 | 0.0241 | 0.9080 |
| Viasat TPC | [7.279] | SDD | 5/6 (20%) | 11.3 | 0.0240 | 0.9080 |
| Convolutional LDPC | [7.84] | SDD | 5/6 (20%) | 11.51 | 0.0267 | 0.8984 |
| Concatenated LDPC+TPC | [7.284] | SDD | 0.830 (20.5%) | 10.8[i] | 0.0183 | 0.9288 |
| SC-LDPC | [7.83] | SDD | 4/5 (25%) | 12.02[c] | 0.0373 | 0.8605 |
| SC-LDPC | [7.165] | SDD | 4/5 (25%) | 12.14[a] | 0.0395 | 0.8525 |

[a] Extrapolated from software verification down to $10^{-12}$
[b] Extrapolated from FPGA verification down to $10^{-13}$
[c] Extrapolated from FPGA verification down to $10^{-14}$
[d] The LDPC super FEC allows lower latency due to lower block length than other codes for HDD defined in [7.46] at a cost of a lower NCG
[e] The extra NCG compared with staircase codes is mostly due to the use of an improved decoding algorithm, which however doubles the memory usage. Without this algorithm, the NCG is similar to that of staircase codes
[f] Extrapolated from software verification. System overhead larger due to extra required stuffing bits
[g] Performance evaluated only for a simplified SDD with 4-level channel output quantization
[h] Concatenated Hamming code and staircase code, optimized for SDD with low power consumption
[i] Outer code not implemented and assuming ideal interleaving between inner and outer code

decoding algorithms that allow us to approach capacity limits, and we believe that the most interesting contributions will lie in this area. Two promising research lines in this direction are soft-aided decoding algorithms for product-like codes [7.223, 224] (see also Sect. 7.5.8) and hybrid HDD/SDD schemes, where an outer hard-decision FEC (a staircase code) is concatenated with an inner soft-decision FEC (typically a low-density parity-check code) [7.221, 222]. The latter schemes are now emerging in standards for low-complexity short-reach optical communications [7.280].

To conclude the chapter, we present to the reader an overview of some state-of-the-art FEC schemes that have been proposed for use in optical communications and whose error rate performance has been evaluated down to a BER of $10^{-15}$ or extrapolated from a reasonably close value. In Table 7.5, we provide a selection of popular and recently proposed FEC schemes for both HDD and SDD. Besides the achievable NCG, we also provide the pre-FEC BER threshold, $\mathrm{BER_{pre}}$, and the GMI threshold, $\mathrm{GMI_{SDD\text{-}BW}}$, if SDD is employed. In the case of SDD, $\mathrm{BER_{pre}}$ is computed assuming BPSK modulation over the AWGN channel. However, we would like to point out (see also Sect. 7.7) that it is not necessarily a good metric for performance estimation, which is why it is given in gray font. The achievable NCGs of the schemes from Table 7.5 are also visualized in Fig. 7.59.

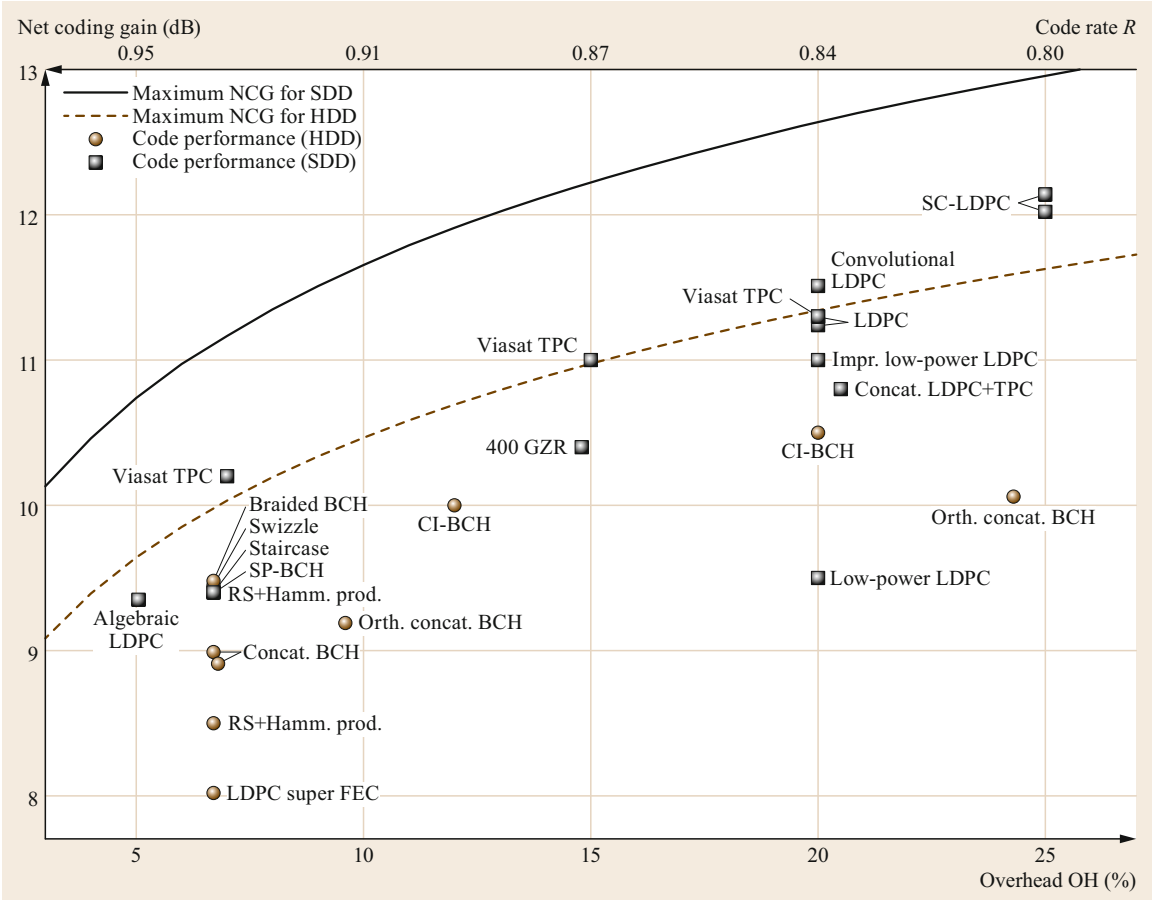


**Fig. 7.59** Illustration of NCGs (at an output BER of $10^{-15}$) obtained by different state-of-the-art FEC schemes with HDD and SDD taken from Tab. 7.5

*Acknowledgments.* Laurent Schmalen would like to acknowledge the support of Sebastian Cammerer (University of Stuttgart) for providing the polar coding simulation results shown in Fig. 7.24, Rafael Rios–Müller (Nokia Bell Labs) for carrying out the experiments used in Sect. 7.7.3, and Detlef Suikat and Detlef Rösener for programming the FPGA platform used in Sect. 7.4.5.

## References

7.1 C.E. Shannon: A mathematical theory of communication, Bell Syst. Tech. J. **27**(379–423), 623–656 (1948)

7.2 A.G. Dimakis, P.B. Godfrey, Y. Wu, M.J. Wainwright, K. Ramchandran: Network coding for distributed storage systems, IEEE Trans. Inf. Theory **56**(9), 4539–4551 (2010)

7.3 M.A. Maddah-Ali, U. Niesen: Fundamental limits of caching, IEEE Trans. Inf. Theory **60**(5), 2856–2867 (2014)

7.4 G. Liva: Graph-based analysis and optimization of contention resolution diversity slotted ALOHA, IEEE Trans. Commun. **59**(2), 477–487 (2011)

7.5 S. Li, M.A. Maddah-Ali, A.S. Avestimehr: A unified coding framework for distributed computing with straggling servers. In: *Proc. IEEE Globecom Work. (GC Wkshps), Washington, DC, Dec* (2016)

7.6 S. Li, M.A. Maddah-Ali, A.S. Avestimehr: Coding for distributed fog computing, IEEE Commun. Mag. **55**(4), 34–40 (2017)

7.7 D. Pearson: High-speed QKD reconciliation using forward error correction. In: *Proc. AIP Int. Conf. Quantum Commun., Measurement Comp*, Vol. 734 (2004) pp. 299–302, Glasgow

7.8 R.J. McEliece: A public-key cryptosystem based on algebraic coding theory, Deep Space Netw. Prog. Rep. **44**, 114–116 (1978)

7.9 W. Grover: Error correction in dispersion-limited lightwave systems, J. Lightwave Technol. **6**, 643–654 (1988)

7.10 R. Hamming: Error detecting and error correcting codes, Bell Syst. Tech. J. **26**, 147–160 (1950)

7.11 D.J. Costello Jr., J. Hagenauer, H. Imai, S.B. Wicker: Applications of error-control coding, IEEE Trans. Inf. Theory **44**(6), 2531–2560 (1998)

7.12 D.J. Costello Jr., G.D. Forney Jr.: Channel coding: the road to capacity, Proceedings IEEE **95**(6), 1150–1177 (2007)

7.13 M.J.E. Golay: Notes on digital coding, Proc. Inst. Radio Eng. **37**(6), 657–657 (1949)

7.14 A. Barg: At the dawn of the theory of codes, Math. Intell. **15**(1), 20–26 (1993)

7.15 F. MacWilliams, N. Sloane: *The Theory of Error-Correcting Codes*, 2nd edn. (North-Holland, Amsterdam 1978)

7.16 D.E. Muller: Application of Boolean algebra to switching circuit design and to error detection, IRE Trans. Electron. Comput. **3**, 6–12 (1954)

7.17 I. Reed: A class of multiple-error-correcting codes and the decoding scheme, IRE Trans. Inf. Theory **4**(4), 38–49 (1954)

7.18 P. Elias: Error-free coding, IRE Trans. Inf. Theory **PGIT-4**, 29–37 (1954)

7.19 R.D. Cideciyan, S. Furrer, M.A. Lantz: Product codes for data storage on magnetic tape, IEEE Trans. Magn. **53**(2), 1–10 (2017)

7.20 S. Emmadi, K.R. Narayanan, H.D. Pfister: Half-product codes for flash memory. In: *Proc. Non-Volatile Memories Workshop* (2015), San Diego

7.21 M. Wang: WiMAX physical layer: Specifications overview and performance evaluation. In: *Proc. Consumer Commun. Networking Conf. (CCNC)* (2011) pp. 10–12

7.22 J. Justesen, K.J. Larsen, L.A. Pedersen: Error correcting coding for OTN, IEEE Commun. Mag. **48**(9), 70–75 (2010)

7.23 P. Elias: Coding for noisy channels. In: *IRE Convention Record, Part IV* (1955) pp. 37–46

7.24 ETSI: *Recommendation GSM 05.03* (ETSI, Sophia Antipolis 1994)

7.25 IEEE: Part 11: Wireless LAN medium access control (MAC) and physical layer (PHY) specifications amendment 5: Enhancements for higher throughput. In: *IEEE Standard for Information technology–Telecommunications and information exchange between systems–Local and metropolitan area networks–Specific requirements* (2005), Std 802.11-2005, Tech. Rep.

7.26 IEEE: Part 3: Carrier sense multiple access with collision detection (CSMA/CD): Access method and physical layer specifications. In: *802.3, Local and Metropolitan Area Networks-specific Requirements* (2006)

7.27 A. Viterbi: Error bounds for convolutional codes and an asymptotically optimum decoding algorithm, IEEE Trans. Inf. Theory **13**(2), 260–269 (1967)

7.28 G.D. Forney Jr.: The Viterbi algorithm: A personal history. In: *Proc. Viterbi Conf.* (2005), University of Southern California, Los Angeles

7.29 T.T. Ha, R.L. Borchardt, E.-S. Hwang: Direct detection optical systems employing OOK modulation and error correction codes, J. Opt. Commun. **17**(6), 212–215 (1996)

7.30 T. Wuth, E. Agrell, M. Karlsson, M. Sköld: Fiber communications using convolutional coding and bandwidth-efficient modulation, Opt. Express **14**(2), 542–555 (2006)

7.31 A. Bisplinghoff, N. Beck, M. Ene, M. Danninger, T. Kupfer: Phase slip tolerant, low power multi-level coding for 64QAM with 12.9 dB NCG. In: *Proc. Opt. Fiber Commun. Conf. (OFC)* (2016) pp. 1–3, Anaheim

7.32 A. Hocquenghem: Codes correcteurs d'erreurs, Chiffres **2**, 147–156 (1959)

7.33 R.C. Bose, D.K. Ray-Chaudhuri: On a class of error correcting binary group codes, Inform. Control **3**(1), 68–79 (1960)

7.34 R. Blahut: *Algebraic Codes for Data Transmission* (Cambridge Univ. Press, New York 2003)

7.35 S. Lin, D.J. Costello Jr.: *Error Control Coding*, 2nd edn. (Prentice Hall, Upper Saddle River 2004)

7.36 S. Quiroga, D. Torres, A. Veloz: In band FEC decoder for SONET/SDH at 2.5 Gbit/s and 10 Gbit/s. In: *Proc. Int. Conf. Elec. Electron. Eng. (ICEEE)* (2004) pp. 70–73

7.37 I.S. Reed, G. Solomon: Polynomial codes over certain finite fields, J. Soc. Ind. Appl. Math. **8**(2), 300–304 (1960)

7.38 E. Berlekamp: *Algebraic Coding Theory* (McGraw-Hill, New York 1968)

7.39 J. Massey: Shift-register synthesis and BCH decoding, IEEE Trans. Inf. Theory **15**(1), 122–127 (1969)

7.40 S. Yamamoto, H. Takahira, E. Shibano, M. Tanaka, Y. Chen: BER performance improvement by forward error correcting code in 5 Gbit/s 9000 km EDFA transmission system, Electron. Lett. **30**, 718–719 (1994)

7.41 L. Schmalen, A. de Lind van Wijngaarden, S. ten Brink: Forward error correction in optical core and access networks, Bell Labs Tech. J. **18**(3), 39–66 (2013)

7.42 ITU-T: *Recommendation G.975: Forward error correction for submarine systems* 1996)

7.43 F. Chang, K. Onohara, T. Mizuochi: Forward error correction for 100 G transport networks, IEEE Commun. Mag. **48**, S48–S55 (2010)

7.44 G.D. Forney: *Concatenated codes, Ph.D. dissertation* (Massachusetts Institute of Technology (MIT), Cambridge 1965)

7.45 ETSI: *DVB-S, ETSI standard EN 300 421 v.1.1.2* (ETSI, Sophia Antipolis 1997)

7.46 ITU-T: *Recommendation G.975.1: Forward error correction for high bit-rate DWDM submarine systems* (ITU, Geneva 2004)

7.47 A. Afshar: Deployment of 100G in the metro network, applications and drivers, Cortina White Paper, Tech. Rep. 20.

7.48 M. Scholten, T. Coe, J. Dillard: Continuously-interleaved BCH (CI-BCH) FEC delivers best in class NECG for 40G and 100G metro applications. In: *Proc. Opt. Fiber Commun. Conf. (OFC)* (2010), San Diego

7.49 B.P. Smith, A. Farhood, A. Hunt, F.R. Kschischang, J. Lodge: Staircase codes: FEC for 100 Gb/s OTN, J. Lightwave Technol. **30**(1), 110–117 (2012)

7.50 Y.-Y. Jian, H.D. Pfister, K.R. Narayanan, R. Rao, R. Mazahreh: Iterative hard-decision decoding of braided BCH codes for high-speed optical communication. In: *Proc. IEEE Global Telecommun. Conf. (GLOBECOM)* (2013), Atlanta

7.51 D.C. MacKay, R. Neal: Near Shannon limit performance of low density parity check codes, Electron. Lett. **32**, 1645–1646 (1996)

7.52 M. Luby, M. Mitzenmacher, A. Shokrollahi, D.A. Spielman, V. Stemann: Practical loss-resilient codes. In: *Proc. ACM Symp. Theory Comput.* (1997) pp. 150–159

7.53 R.G. Gallager: Low-density parity-check codes, IRE Trans. Inf. Theory **8**(1), 21–28 (1962)

7.54 R.G. Gallager: *Low-Density Parity-Check Codes* (M.I.T. Press, Cambridge 1963)

7.55 IEEE: *802.11, wireless LAN medium access control (MAC) and physical layer (PHY) specifications* (IEEE, Piscataway Township 2012)

7.56 ETSI: *DVB-S2, ETSI standard EN 302 307 v.1.2.1* (ETSI, Sophia Antipolis 2009)

7.57 IEEE: *802.3an, local and metropolitan area networks-specific requirements Part 3: Carrier sense multiple access with collision detection (CSMA/CD): Access method and physical layer specifications* (IEEE, Piscataway Township 2006)

7.58 R. Pyndiah: Near optimum decoding of product codes: Block turbo codes, IEEE Trans. Commun. **46**(8), 1003–1010 (1998)

7.59 T. Mizuochi, Y. Miyata, T. Kobayashi, K. Ouchi, K. Kuno, K. Kubo, K. Shimizu, H. Tagami, H. Yoshida, H. Fujita, M. Akita, K. Motoshima: Forward error correction based on block turbo code with 3-bit soft decision for 10-Gb/s optical communication systems, IEEE J. Sel. Top. Quantum Electron. **10**(2), 376–386 (2004)

7.60 B. Vasic, I.B. Djordjevic: Low-density parity check codes for long-haul optical communication systems, IEEE Photon. Technol. Lett. **14**(8), 1208–1210 (2002)

7.61 Y. Miyata, K. Kubo, H. Yoshida, T. Mizuochi: Proposal for frame structure of optical channel transport unit employing LDPC codes for 100 Gb/s FEC. In: *Proc. Opt. Fiber Commun. Conf. (OFC)* (2009)

7.62 I.B. Djordjevic, B. Vasic: Nonbinary LDPC codes for optical communication systems, IEEE Photon. Technol. Lett. **17**(10), 2224–2226 (2005)

7.63 T. Richardson: Error floors of LDPC codes. In: *Proc. Allerton Annu. Conf. Commun. Control Comp.* (2003)

7.64 D.A. Morero, M.A. Castrillon, F.A. Ramos, T.A. Goette, O.E. Agazzi, M.R. Hueda: Non-concatenated FEC codes for ultra-high speed optical transport networks. In: *Proc. IEEE Global Telecommun. Conf. (GLOBECOM)* (2011) pp. 1–5

7.65 E. Arıkan: Channel polarization: A method for constructing capacity-achieving codes. In: *Proc. IEEE Int. Symp. Inform. Theory (ISIT)* (2008) pp. 1173–1177

7.66 E. Arıkan: Channel polarization: A method for constructing capacity-achieving codes for symmetric binary-input memoryless channels, IEEE Trans. Inf. Theory **55**(7), 3051–3073 (2009)

7.67 I. Tal, A. Vardy: List decoding of polar codes, IEEE Trans. Inf. Theory **61**(5), 2213–2226 (2015)

7.68 ETSI: *3GPP TR 38.802 V14.2.0, Technical Specification Group Radio Access Network: Study on new radio access technology physical layer aspects (release 14), 3GPP* (ETSI, Sophia Antipolis 2017), Tech. Rep.

7.69 G. Sarkis, P. Giard, A. Vardy, C. Thibeault, W.J. Gross: Fast list decoders for polar codes, IEEE J. Sel. Areas Commun. **34**(2), 318–328 (2016)

7.70 P. Giard, C. Thibeault, W.J. Gross: *High-Speed Decoders for Polar Codes* (Springer, Berlin, Heidelberg 2016)

7.71 A.J. Felström, K.S. Zigangirov: Time-varying periodic convolutional codes with low-density parity-check matrix, IEEE Trans. Inf. Theory **45**(6), 2181–2191 (1999)

7.72 M. Lentmaier, G. Fettweis, K. Zigangirov, D. Costello: Approaching capacity with asymptotically regular LDPC codes. In: *Proc. Inform. Theory Appl. Workshop (ITA)* (2009)

7.73 M. Lentmaier, A. Sridharan, D.J. Costello, K. Zigangirov: Iterative decoding threshold analysis for LDPC convolutional codes, IEEE Trans. Inf. Theory **56**(10), 5274–5289 (2010)

7.74 M. Lentmaier, D. Mitchell, G. Fettweis, D. Costello: Asymptotically good LDPC convolutional codes with AWGN channel thresholds close to the shannon limit. In: *Proc. Int. Symp. Turbo Codes Iterative Inform. Process. (ISTC)* (2010), Brest

7.75 S. Kudekar, T. Richardson, R. Urbanke: Threshold saturation via spatial coupling: Why convolutional LDPC ensembles perform so well over the BEC, IEEE Trans. Inf. Theory **57**(2), 803–834 (2011)

7.76 S. Kudekar, T. Richardson, R. Urbanke: Spatially coupled ensembles universally achieve capacity under belief propagation, IEEE Trans. Inf. Theory **59**(12), 7761–7813 (2013)

7.77 A. Iyengar, M. Papaleo, P. Siegel, J. Wolf, A. Vanelli-Coralli, G. Corazza: Windowed decoding of protograph-based LDPC convolutional codes over erasure channels, IEEE Trans. Inf. Theory **58**(4), 2303–2320 (2012)

7.78 S. Moloudi, M. Lentmaier, A. Graell i Amat: Spatially coupled turbo-like codes, IEEE Trans. Inf. Theory **63**(10), 6199–6215 (2017)

7.79 V. Aref, N. Macris, R. Urbanke, M. Vuffray: Lossy source coding via spatially coupled LDGM ensembles. In: *Proc. IEEE Int. Symp. Inform. Theory (ISIT)* (2012) pp. 373–377, Cambridge

7.80 K. Takeuchi, T. Tanaka, T. Kawabata: Improvement of BP-based CDMA multiuser detection by spatial coupling. In: *Proc. IEEE Int. Symp. Inform. Theory (ISIT)* (2011) pp. 1489–1493, St. Petersburg

7.81 D.L. Donoho, A. Javanmard, A. Montanari: Information-theoretically optimal compressed sensing via spatial coupling and approximate message passing, IEEE Trans. Inf. Theory **59**(11), 7434–7464 (2013)

7.82 Z. Wu, J.K. Fischer, B. Lankl: Experimental investigation of polar code performance for coherent UDWDM PONs. In: *Proc. Opt. Fiber Commun. Conf. (OFC)* (2015)

7.83 L. Schmalen, D. Suikat, D. Rösener, V. Aref, A. Leven, S. ten Brink: Spatially coupled codes and optical fiber communications: An ideal match? In: *Proc. IEEE Workshop Signal Process. Adv. Wirel. Commun. (SPAWC)* (2015) pp. 460–464

7.84 D. Chang, F. Yu, Z. Xiao, N. Stojanovic, F.N. Hauske, Y. Cai, C. Xie, L. Li, X. Xu, Q. Xiong: LDPC convolutional codes using layered decoding algorithm for high speed coherent optical transmission. In: *Proc. Opt. Fiber Commun. Conf. (OFC)* (2012) p. OW1H.4

7.85 I.B. Djordjevic, W. Ryan, B. Vasic: *Coding for Optical Channels* (Springer, Cham 2010)

7.86 A. Leven, L. Schmalen: Status and recent advances on forward error correction technologies for lightwave systems, J. Lightw. Technol. **32**(16), 2735–2750 (2014)

7.87 N. Merhav, G. Kaplan, A. Lapidoth, S. Shamai Shitz: On information rates for mismatched decoders, IEEE Trans. Inf. Theory **40**(6), 1953–1967 (1994)

7.88 A. Ganti, A. Lapidoth, I.E. Telatar: Mismatched decoding revisited: general alphabets, channels with memory, and the wide-band limit, IEEE Trans. Inf. Theory **46**(7), 2315–2328 (2000)

7.89 T. Fehenberger, A. Alvarado, P. Bayvel, N. Hanik: On achievable rates for long-haul fiber-optic communications, Opt. Express **23**(7), 9183–9191 (2015)

7.90 A. Sheikh, A. Graell i Amat, G. Liva: Achievable information rates for coded modulation with hard decision decoding for coherent fiber-optic systems, J. Lightwave Technol. **35**(23), 5069–5078 (2017)

7.91 G. Böcherer: Achievable rates for probabilistic shaping, http://arxiv.org/abs/arXiv:1707.01134 (2017)

7.92 I.B. Djordjevic, B. Vasic, M. Ivkovic, I. Gabitov: Achievable information rates for high-speed long-haul optical transmission, J. Lightwave Technol. **23**(11), 3755–3763 (2005)

7.93 E. Agrell, A. Alvarado, G. Durisi, M. Karlsson: Capacity of a nonlinear optical channel with finite memory, J. Lightwave Technol. **32**(16), 2862–2876 (2014)

7.94 R.J. Essiambre, G. Kramer, P.J. Winzer, G.J. Foschini, B. Goebel: Capacity limits of optical fiber networks, J. Lightwave Technol. **28**(4), 662–701 (2010)

7.95 M. Secondini, E. Forestieri, G. Prati: Achievable information rate in nonlinear WDM fiber-optic systems with arbitrary modulation formats and dispersion maps, J. Lightwave Technol. **31**(23), 3839–3852 (2013)

7.96 G. Caire, G. Taricco, E. Biglieri: Bit-interleaved coded modulation, IEEE Trans. Inf. Theory **44**(3), 927–946 (1998)

7.97 A. Martinez, A. Guillén i Fàbregas, G. Caire, F.M.J. Willems: Bit-interleaved coded modulation revisited: A mismatched decoding perspective, IEEE Trans. Inf. Theory **55**(6), 2756–2765 (2009)

7.98 G. Kaplan, S. Shamai Shitz: Information rates and error exponents of compound channels with application to antipodal signaling in a fading environment, AEÜ **47**(4), 228–230 (1993)

7.99 P. Poggiolini: The GN model of non-linear propagation in uncompensated coherent optical systems, J. Lightwave Technol. **30**(24), 3857–3879 (2012)

7.100 P. Johannisson, E. Agrell: Modeling of nonlinear signal distortion in fiber-optic networks, J. Lightwave Technol. **32**(23), 3942–3950 (2014)

7.101 W.E. Ryan, S. Lin: *Channel Codes. Classical and Modern* (Cambridge Univ. Press, Cambridge 2009)

7.102 R.L. Dobrushin: Asymptotic optimality of group and systematic codes for some channels, Theory Probab. Appl. **8**(1), 47–60 (1963)

7.103 A. Carena, G. Bosco, V. Curri, Y. Jiang, P. Poggiolini, F. Forghieri: EGN model of non-linear fiber propagation, Opt. Express **22**(13), 16335–16362 (2014)

7.104 N.A. Shevchenko, J.E. Prilepsky, S.A. Derevyanko, A. Alvarado, P. Bayvel, S.K. Turitsyn: A lower bound on the per soliton capacity of the nonlinear optical fibre channel. In: *Proc. IEEE Inform. Theory Workshop (ITW)* (2015) pp. 104–108

7.105 A. Alvarado, L. Szczecinski, R. Feick: On the distribution of extrinsic L-values in gray-mapped 16-QAM. In: *Proc. Int. Conf. Wirel. Commun. Mobile Comp. (IWCMC)* (2007) pp. 329–336

7.106 Top500: The List, https://www.top500.org/ (2019)

7.107 S. Dolinar, D. Divsalar, F. Pollara: *Code performance as a function of block size* (Jet Propulsion Laboratory, Pasadena 1998) pp. 42–133, TMO Prog. Rep.

7.108 Y. Polyanskiy, H. Poor, S. Verdu: Channel coding rate in the finite blocklength regime, IEEE Trans. Inf. Theory **56**(5), 2307–2359 (2010)

7.109 C.E. Shannon: Probability of error for optimal codes in a gaussian channel, Bell Syst. Tech. J. **38**, 611–656 (1959)

7.110 T. Erseghe: Coding in the finite-blocklength regime: Bounds based on Laplace integrals and their asymptotic approximations, IEEE Trans. Inf. Theory **62**(12), 6854–6883 (2016)

7.111 T. Richardson, R. Urbanke: *Modern Coding Theory* (Cambridge Univ. Press, Cambridge 2008)

7.112 T.J. Richardson, M.A. Shokrollahi, R.L. Urbanke: Design of capacity-approaching irregular low-density parity-check codes, IEEE Trans. Inf. Theory **47**(2), 619–637 (2001)

7.113 D. Divsalar, C. Jones, S. Dolinar, J. Thorpe: Protograph based LDPC codes with minimum distance linearly growing with block size. In: *Proc. IEEE Global Telecommun. Conf. (GLOBECOM)*, Vol. 3 (2005) p. 5

7.114 Consultative Committee for Space Data Systems (CCSDS): *TM synchronization and channel coding, recommended standard, CCSDS 131.0-B-2, blue book* (CCSDS, Washington 2011), Tech. Rep.

7.115 R.M. Tanner: A recursive approach to low complexity codes, IEEE Trans. Inf. Theory **27**(5), 533–547 (1981)

7.116 L. Dolecek, P. Lee, Z. Zhang, V. Anantharam, B. Nikolic, M. Wainwright: Predicting error floors of structured LDPC codes: Deterministic bounds and estimates, IEEE J. Sel. Areas Commun. **27**(6), 908–917 (2009)

7.117 T. Richardson, R. Urbanke: The capacity of low-density parity-check codes under belief propagation decoding, IEEE Trans. Inf. Theory **47**(2), 599–618 (2001)

7.118 S. Jayasooriya, M. Shirvanimoghaddam, L. Ong, G. Lechner, S.J. Johnson: A new density evolution approximation for LDPC and multi-edge type LDPC codes, IEEE Trans. Commun. **64**(10), 4044–4056 (2016)

7.119 S.-Y. Chung, G.D. Forney Jr., T. Richardson, R. Urbanke: On the design of low-density parity-check codes within 0.0045 db of the Shannon limit, IEEE Commun. Lett. **5**(2), 58–60 (2001)

7.120 S. ten Brink, G. Kramer, A. Ashikhmin: Design of low-density parity-check codes for modulation and detection, IEEE Trans. Commun. **52**(4), 670–678 (2004)

7.121 L. Schmalen, S. ten Brink, A. Leven: Advances in detection and error correction for coherent optical communications: Regular, irregular, and spatially coupled LDPC code designs. In: *Enabling Technologies for High Spectral-Efficiency Coherent Optical Communication Networks*, ed. by X. Zhou, C. Xie (Wiley, Hoboken 2016), Chap. 3

7.122 B. Vasic, I.B. Djordjevic, R.K. Kostuk: Low-density parity check codes and iterative decoding for long-haul optical communication systems, J. Lightwave Technol. **21**(2), 438 (2003)

7.123 G. Liva, S. Song, L. Lan, Y. Zhang, S. Lin, W. Ryan: Design of LDPC codes: A survey and new results, J. Commun. Software Syst. **2**(3), 191–211 (2006)

7.124 N. Bonello, S. Chen, L. Hanzo: Design of low-density parity-check codes, IEEE Veh. Technol. Mag. **6**(4), 1574–1606 (2011)

7.125 X.-Y. Hu, E. Eleftheriou, D.M. Arnold: Regular and irregular progressive edge-growth Tanner graphs, IEEE Trans. Inf. Theory **51**(1), 386–398 (2005)

7.126 T. Tian, C. Jones, J.D. Villasenor, R.D. Wesel: Construction of irregular LDPC codes with low error floors. In: *Proc. IEEE Int. Conf. Commun. (ICC)* (2003) pp. 3125–3129

7.127 H. Xiao, A.H. Banihashemi: Improved progressive-edge-growth (PEG) construction of irregular LDPC codes, IEEE Commun. Lett. **8**(12), 715–717 (2004)

7.128 T. Tian, C.R. Jones, J.D. Villasenor, R.D. Wesel: Selective avoidance of cycles in irregular LDPC code construction, IEEE Trans. Commun. **52**(8), 1242–1247 (2004)

7.129 T. Richardson, R. Urbanke: Multi-edge type LDPC codes. In: *Workshop honoring Prof. Bob McEliece on his 60th birthday* (California Institute of Technology, Pasadena 2002) pp. 24–25

7.130 J. Thorpe: *Low-density parity-check (LDPC) codes constructed from protographs* (NASA JPL, Pasadena 2003), IPN Progress Report 42–154

7.131 J. Li, S. Lin, K. Abdel-Ghaffar, W.E. Ryan, D.J. Costello Jr.: *LDPC Code Designs, Constructions, and Unification* (Cambridge Univ. Press, Cambridge 2016)

7.132 M.P.C. Fossorier: Quasi-cyclic low-density parity-check codes from circulant permutation matrices, IEEE Trans. Inf. Theory **50**(8), 1788–1793 (2004)

7.133 R. Smarandache, P.O. Vontobel: Quasi-cyclic LDPC codes: Influence of proto- and Tanner-graph structure on minimum Hamming distance upper bounds, IEEE Trans. Inf. Theory **58**(2), 585–607 (2012)

7.134 X. Jiang, M.H. Lee: Large girth quasi-cyclic LDPC codes based on the Chinese remainder theorem, IEEE Commun. Lett. **13**(5), 342–344 (2009)

7.135 T. Richardson, R. Urbanke: Efficient encoding of low-density parity-check codes, IEEE Trans. Inf. Theory **47**(2), 638–656 (2001)

7.136 D.J.C. MacKay: *Information Theory, Inference and Learning Algorithms* (Cambridge Univ. Press, Cambridge 2003)

7.137 J. Pearl: *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference* (Morgan Kaufmann, New York 1988)

7.138 G. Lechner, T. Pedersen, G. Kramer: Analysis and design of binary message passing decoders, IEEE Trans. Commun. **60**(3), 601–607 (2012)

7.139 J. Hagenauer, E. Offer, L. Papke: Iterative decoding of binary block and convolutional codes, IEEE Trans. Inf. Theory **42**(2), 429–445 (1996)

7.140 D.R. Pauluzzi, N.C. Beaulieu: A comparison of SNR estimation techniques for the AWGN channel, IEEE Trans. Commun. **48**(10), 1681–1691 (2000)

7.141 C.F. Mecklenbräuker, S. Paul: On estimating the signal to noise ratio from BPSK signals. In: *Proc. Int. Conf. Acoust. Speech Signal Process. (ICASSP)*, Vol. 4 (2005)

7.142 T.K. Moon: *Error Correction Coding – Mathematical Methods and Algorithms* (Wiley, Hoboken 2005)

7.143 J. Chen, M.P.C. Fossorier: Near optimum universal belief propagation based decoding of low-density parity check codes, IEEE Trans. Commun. **50**(3), 406–414 (2002)

7.144 M. Ardakani, F.R. Kschischang: Gear-shift decoding, IEEE Trans. Commun. **54**(7), 1235–1242 (2006)

7.145 J. Zhao, F. Zarkeshvari, A.H. Banihashemi: On implementation of min-sum algorithm and its modifications for decoding low-density parity-check (LDPC) codes, IEEE Trans. Commun. **53**(4), 549–554 (2005)

7.146 D. Hocevar: A reduced complexity decoder architecture via layered decoding of LDPC codes. In: *Proc. IEEE Workshop Signal Process. Syst. (SiPS)* (2004)

7.147 A. Leven, L. Schmalen: Implementation aspects of coherent transmit and receive functions in application-specific integrated circuits. In: *Optical Fiber Telecommunications VI-A*, ed. by I.P. Kaminow, T. Li, A.E. Willner (Academic, New York 2013), Chap. 15

7.148 A. Wyner, R. Ash: Analysis of recurrent codes, IEEE Trans. Inf. Theory **9**(3), 143–156 (1963)

7.149 A. Yedla, Y.-Y. Jian, P.S. Nguyen, H.D. Pfister: A simple proof of Maxwell saturation for coupled scalar

recursions, IEEE Trans. Inf. Theory **60**(11), 6943–6965 (2014)
7.150 A. Yedla, Y.-Y. Jian, P. Nguyen, H. Pfister: A simple proof of threshold saturation for coupled vector recursions. In: *Proc. IEEE Inform. Theory Workshop (ITW)* (2012) pp. 25–29, Lausanne
7.151 A. Piemontese, A. Graell i Amat, G. Colavolpe: Nonbinary spatially-coupled LDPC codes on the binary erasure channel. In: *Proc. IEEE Int. Conf. Commun. (ICC)* (2013) pp. 3270–3274, Budapest
7.152 I. Andriyanova, A. Graell i Amat: Threshold saturation for nonbinary SC-LDPC codes on the binary erasure channel, IEEE Trans. Inf. Theory **62**(5), 2622–2638 (2016)
7.153 ITU-T: *Recommendation G.709: Interfaces for the optical transport network* (ITU-T, Genf 2016)
7.154 W. Lautenschlaeger, N. Benzaoui, F. Buchali, L. Dembeck, R. Dischler, B. Franz, U. Gebhard, J. Milbrandt, Y. Pointurier, D. Roesener, L. Schmalen, A. Leven: Optical ethernet – flexible optical metro networks, J. Lightwave Technol. **35**(12), 2346–2357 (2017)
7.155 S. Kudekar, C. Méasson, T. Richardsony, R. Urbanke: Threshold saturation on BMS channels via spatial coupling. In: *Proc. Int. Symp. Turbo Codes Iterative Inform. Process. (ISTC)* (2010) pp. 309–313
7.156 C. Häger, A. Graell i Amat, A. Alvarado, F. Brännström, E. Agrell: Optimized bit mappings for spatially coupled LDPC codes over parallel binary erasure channels. In: *Proc. IEEE Int. Conf. Commun. (ICC)* (2014) pp. 2064–2069, Sydney
7.157 S. Cammerer, V. Aref, L. Schmalen, S. ten Brink: Triggering wave-like convergence of tail-biting spatially coupled LDPC codes. In: *Conf. Inform. Sci. Syst. (CISS)* (2016) pp. 93–98
7.158 L. Schmalen, V. Aref, F. Jardel: Non-uniformly coupled LDPC codes: Better thresholds, smaller rate-loss, and less complexity. In: *Proc. IEEE Int. Symp. Inform. Theory (ISIT)* (2017), Aachen
7.159 S. Kudekar, T. Richardson, R. Urbanke: Wave-like solutions of general one-dimensional spatially coupled systems, IEEE Trans. Inf. Theory **61**(8), 4117–4157 (2015)
7.160 M. Tavares: *On low-density parity-check convolutional codes: Constructions, analysis and VLSI implementations, Ph.D. dissertation* (TU Dresden, Dresden 2010)
7.161 A.E. Pusane, A.J. Feltström, A. Sridharan, M. Lentmaier, K.S. Zigangirov, D.J. Costello Jr.: Implementation aspects of LDPC convolutional codes, IEEE Trans. Commun. **56**(7), 1060–1069 (2008)
7.162 V. Aref, L. Schmalen, S. ten Brink: On the convergence speed of spatially coupled LDPC ensembles. In: *Proc. Allerton Annu. Conf. Commun. Control Comp.* (2013) p. 2013, arXiv: 1307.3780
7.163 R. El-Khatib, N. Macris: The velocity of the decoding wave for spatially coupled codes on BMS channels. In: *Proc. IEEE Int. Symp. Inform. Theory (ISIT)* (2016) pp. 2119–2123
7.164 R. El-Khatib, N. Macris: The velocity of the propagating wave for spatially coupled systems with applications to LDPC codes, IEEE Trans. Inf. Theory **64**(11), 7113–7131 (2018)
7.165 L. Schmalen, V. Aref, J. Cho, D. Suikat, D. Rösener, A. Leven: Spatially coupled soft-decision error correction for future lightwave systems, J. Lightwave Technol. **33**(5), 1109–1116 (2015)
7.166 P.M. Olmos, R. Urbanke: A scaling law to predict the finite-length performance of spatially-coupled LDPC codes, IEEE Trans. Inf. Theory **61**(6), 3164–3184 (2015)
7.167 M. Stinner, P.M. Olmos: Analyzing finite-length protograph-based spatially coupled LDPC codes. In: *Proc. IEEE Int. Symp. Inform. Theory (ISIT)* (2014) pp. 891–895
7.168 J. Cho, L. Schmalen: Construction of protographs for large-girth structured LDPC convolutional codes. In: *Proc. IEEE Int. Conf. Commun. (ICC)* (2015) pp. 4412–4417
7.169 M. Battaglioni, A. Tasdighi, G. Cancellieri, F. Chiaraluce, M. Baldi: Design and analysis of time-invariant SC-LDPC codes with small constraint length, IEEE Trans. Commun. **66**(3), 918–931 (2018)
7.170 D. Achlioptas, H. Hassani, W. Liu, R. Urbanke: Time-invariant LDPC convolutional codes. In: *Proc. IEEE Int. Symp. Inform. Theory (ISIT)* (2017), Aachen
7.171 L. Schmalen, D. Suikat, D. Rösener, A. Leven: Evaluation of left-terminated spatially coupled LDPC codes for optical communications. In: *Proc. Eur. Conf. Opt. Commun. (ECOC)* (2014)
7.172 K. Tazoe, K. Kasai, K. Sakaniwa: Efficient termination of spatially-coupled codes. In: *Proc. IEEE Inform. Theory Workshop (ITW)* (2012)
7.173 M.R. Sanatkar, H.D. Pfister: Increasing the rate of spatially-coupled codes via optimized irregular termination. In: *Proc. Int. Symp. Turbo Codes Iterative Inform. Process. (ISTC)* (2016)
7.174 H. Kwak, J. Kim, Jong-Seon: Rate-loss reduction of SC-LDPC codes by optimizing reliable variable nodes via expected graph evolution. In: *Proc. IEEE Int. Symp. Inform. Theory (ISIT)* (2017) pp. 2930–2934
7.175 T. Jerkovits, G. Liva, A. Graell i Amat: Improving the decoding threshold of tailbiting spatially coupled LDPC codes by energy shaping, IEEE Commun. Lett. **22**(4), 660–663 (2018)
7.176 C. Häger, A. Graell i Amat, F. Brännström, A. Alvarado, E. Agrell: Comparison of terminated and tailbiting spatially coupled LDPC codes with optimized bit mapping for PM-64-QAM. In: *Proc. Eur. Conf. Opt. Commun. (ECOC)* (2014)
7.177 C. Häger, A. Graell i Amat, F. Brännström, A. Alvarado, E. Agrell: Improving soft FEC performance for higher-order modulations via optimized bit channel mappings, Opt. Express **22**(12), 14544–14558 (2014)
7.178 C. Häger, A. Graell i Amat, F. Brännström, A. Alvarado, E. Agrell: Terminated and tailbiting spatially coupled codes with optimized bit mappings for spectrally efficient fiber-optical systems, J. Lightwave Technol. **33**(7), 1275–1285 (2015)
7.179 S. Cammerer, L. Schmalen, V. Aref, S. ten Brink: Wave-like decoding of tail-biting spatially coupled LDPC codes through iterative demapping. In: *Proc.*

*Int. Symp. Turbo Codes Iterative Inform. Process. (ISTC)* (2016) pp. 121–125
7.180 L. Schmalen, D. Suikat, V. Aref, D. Rösener: On the design of capacity-approaching unit-memory spatially coupled LDPC codes for optical communications. In: *Proc. Eur. Conf. Opt. Commun. (ECOC)* (2016) pp. 1–3
7.181 T. Koike-Akino, S.C. Draper, Y. Wang, K. Sugihara, W. Matsumoto, D.S. Millar, K. Parsons, V. Arlunno, K. Kojima: Optimal layered scheduling for hardware-efficient windowed decoding of LDPC convolutional codes. In: *Proc. Eur. Conf. Opt. Commun. (ECOC)* (2016) pp. 1–3
7.182 K. Klaiber, S. Cammerer, L. Schmalen, S. ten Brink: Avoiding burst-like error patterns in windowed decoding of spatially coupled LDPC codes. In: *Proc. Int. Symp. Turbo Codes Iterative Inform. Process. (ISTC)* (2018), Hong Kong
7.183 D.J. Costello Jr., L. Dolecek, T. Fuja, J. Kliewer, D. Mitchell, R. Smarandache: Spatially coupled sparse codes on graphs: Theory and practice, IEEE Commun. Mag. **52**(7), 168–176 (2014)
7.184 E. Şaşoğlu, I. Telatar, E. Arıkan: Polarization for arbitrary discrete memoryless channels. In: *Proc. IEEE Inform. Theory Workshop (ITW)* (2009) pp. 144–148
7.185 D. Sutter, J.M. Renes, F. Dupuis, R. Renner: Achieving the capacity of any DMC using only polar codes. In: *Proc. IEEE Inform. Theory Workshop (ITW)* (2012) pp. 114–118
7.186 N. Stolte: *Rekursive Codes mit der Plotkin-Konstruktion und ihre Decodierung, Ph.D. dissertation* (Technische Universität Darmstadt, Darmstadt 2002)
7.187 I. Tal, A. Vardy: How to construct polar codes, IEEE Trans. Inf. Theory **59**(10), 6562–6582 (2013)
7.188 E. Arıkan: Systematic polar coding, IEEE Commun. Lett. **15**(8), 860–862 (2011)
7.189 T. Ahmad: *Polar codes for optical communications, Ph.D. dissertation* (Bilkent University, Ankara 2016)
7.190 B. Li, H. Shen, D. Tse: Parallel decoders of polar codes, https://arxiv.org/abs/1309.1026 (2013)
7.191 G. Sarkis, P. Giard, A. Vardy, C. Thibeault, W.J. Gross: Fast polar decoders: Algorithm and implementation, IEEE J. Sel. Areas Commun. **32**(5), 946–957 (2014)
7.192 S. Cammerer, M. Ebada, A. Elkelesh, S. ten Brink: Sparse graphs for belief propagation decoding of polar codes. In: *Proc. IEEE Int. Symp. Inform. Theory (ISIT)* (2018) pp. 1465–1469, Vail
7.193 C. Berrou, A. Glavieux: Near optimum error correcting coding and decoding: Turbo-codes, IEEE Trans. Commun. **44**(10), 1261–1271 (1996)
7.194 J. Zhang, Y. Wang, M.P.C. Fossorier, J.S. Yedidia: Iterative decoding with replicas, IEEE Trans. Inf. Theory **53**(5), 1644–1663 (2007)
7.195 M. Mondelli, S.H. Hassani, R.L. Urbanke: Unified scaling of polar codes: Error exponent, scaling exponent, moderate deviations, and error floors, IEEE Trans. Inf. Theory **62**(12), 6698–6712 (2016)
7.196 Z. Zhang, L. Dolecek, B. Nikolic, V. Anantharam, M. Wainwright: Investigation of error floors of structured low-density parity-check codes by hardware emulation. In: *Proc. IEEE Global Telecommun. Conf. (GLOBECOM)* (2006)
7.197 M. Arabaci, I.B. Djordjevic: An alternative FPGA implementation of decoders for quasi-cyclic LDPC codes. In: *Proc. Telecommun. Forum (TELFOR)* (2008)
7.198 D.-U. Lee, J.D. Villasenor, W. Luk, P.H.W. Leong: A hardware Gaussian noise generator using the Box-Muller method and its error analysis, IEEE Trans. Comput. **55**(6), 659–671 (2006)
7.199 F. Yu, M. Li, N. Stojanovic, C. Xie, Z. Xiao, L. Li: FPGA demonstration of stretched continuously interleaved BCH code with low error floor for short-range optical transmission. In: *Proc. Opt. Fiber Commun. Conf. (OFC)* (2017), Los Angeles
7.200 A.J. Feltström, D. Truhachev, M. Lentmaier, K.S. Zigangirov: Braided block codes, IEEE Trans. Inf. Theory **55**(6), 2640–2658 (2009)
7.201 S. Benedetto, E. Biglieri: *Principles of Digital Transmission: With Wireless Applications* (Kluwer Academic, Norwell 1999)
7.202 European Telecommunications Standards Institute: *Digital video broadcasting (DVB); upper layer FEC for DVB systems, ETSI TR 102 993, 2007* (ETSI, Sophia Antipolis 2011), Tech. Rep.
7.203 Consultative Committee for Space Data Systems: *CCSDS 131.0-B-2, CCSDS recommended standard for TM synchronization and channel coding* (CCSDS, Washington 2011), Tech. Rep.
7.204 J.S. Plank: A tutorial on Reed-Solomon coding for fault-tolerance in RAID-like systems, Softw. Pract. Exp. **27**(9), 995–1012 (1997)
7.205 L.M. Tolhuizen: More results on the weight enumerator of product codes, IEEE Trans. Inf. Theory **48**(9), 2573–2577 (2002)
7.206 S. Hirasawa, M. Kasahara, Y. Sugiyama, T. Namekawa: Modified product codes, IEEE Trans. Inf. Theory **30**(2), 299–306 (1984)
7.207 M. Alipour, O. Etesami, G. Maatouk, A. Shokrollahi: Irregular product codes. In: *Proc. IEEE Inform. Theory Workshop (ITW)* (2012) pp. 197–201, Lausanne
7.208 L.M. Zhang, F.R. Kschischang: Staircase codes with 6% to 33% overhead, J. Lightwave Technol. **32**(10), 1999–2002 (2014)
7.209 C. Häger, A. Graell i Amat, H.D. Pfister, A. Alvarado, F. Brännström, E. Agrell: On parameter optimization for staircase codes. In: *Proc. Opt. Fiber Commun. Conf. (OFC)* (2015)
7.210 A. Sheikh, A. Graell i Amat, M. Karlsson: Nonbinary staircase codes for spectrally and energy efficient fiber-optic systems. In: *Proc. Opt. Fiber Commun. Conf. (OFC)* (2017) p. W1J.1, Los Angeles
7.211 M. Schwartz, P.H. Siegel, A. Vardy: On the asymptotic performance of iterative decoders for product codes. In: *Proc. IEEE Int. Symp. Inform. Theory (ISIT)* (2005) pp. 1758–1762, Adelaide
7.212 H.D. Pfister, S.K. Emmadi, K. Narayanan: Symmetric product codes. In: *Proc. Inform. Theory Appl. Workshop (ITA)* (2015) pp. 282–290, San Diego
7.213 C. Häger, H.D. Pfister, A. Graell i Amat, F. Brännström: Density evolution and error floor analysis for staircase and braided codes. In:

*Proc. Opt. Fiber Commun. Conf. (OFC)* (2016) pp. 1–3, Los Angeles

7.214 C. Häger, H.D. Pfister, A. Graell i Amat, F. Brännström: Deterministic and ensemble-based spatially-coupled product codes. In: *Proc. IEEE Int. Symp. Inform. Theory (ISIT)* (2016) pp. 2114–2118, Barcelona

7.215 C. Häger, H.D. Pfister, A. Graell i Amat, F. Brännström: Density evolution for deterministic generalized product codes on the binary erasure channel at high rates, IEEE Trans. Inf. Theory **63**(7), 4357–4378 (2017)

7.216 D. Chase: Class of algorithms for decoding block codes with channel measurement information, IEEE Trans. Inf. Theory **18**(1), 170–182 (1972)

7.217 C. Häger, H.D. Pfister: Approaching miscorrection-free performance of product codes with anchor decoding, IEEE Trans. Commun. **66**(7), 2797–2808 (2018)

7.218 J. Hagenauer: The turbo principle: Tutorial introduction and state of the art. In: *Proc. Int. Symp. Turbo Codes Iterative Inform. Process. (ISTC)* (1997), Brest

7.219 Y.Y. Jian, H.D. Pfister, K.R. Narayanan: Approaching capacity at high rates with iterative hard-decision decoding. In: *Proc. IEEE Int. Symp. Inform. Theory (ISIT)* (2012) pp. 2696–2700, Cambridge

7.220 D. Truhachev, A. Karami, L. Zhang, F. Kschischang: Decoding analysis accounting for mis-corrections for spatially-coupled split-component codes. In: *Proc. IEEE Int. Symp. Inform. Theory (ISIT)* (2016) pp. 2124–2128, Barcelona

7.221 L.M. Zhang, F.R. Kschischang: Low-complexity soft-decision concatenated LDGM-staircase FEC for high-bit-rate fiber-optic communication, J. Lightwave Technol. **35**(18), 3991–3999 (2017)

7.222 M. Barakatain, F.R. Kschischang: Low-complexity concatenated LDPC-staircase codes, J. Lightwave Technol. **36**(12), 2443–2449 (2018)

7.223 A. Sheikh, A. Graell i Amat, G. Liva: Iterative bounded distance decoding of product codes with scaled reliability. In: *Proc. Eur. Conf. Opt. Commun. (ECOC)* (2018), Rome

7.224 A. Sheikh, A. Graell i Amat, G. Liva: On low-complexity decoding of product codes for high-throughput fiber-optic systems. In: *Proc. Int. Symp. Turbo Codes Iterative Inform. Process. (ISTC)* (2018), Hong Kong

7.225 C. Fougstedt, A. Sheikh, A. Graell i Amat, G. Liva, P. Larsson-Edefors: Energy-efficient soft-assisted product decoders. In: Proc. Opt. Fiber Commun. Conf. (OFC), p. W3H.6. https://arxiv.org/abs/1810.12054 (2019)

7.226 G. Ungerboeck, I. Csajka: On improving data-link performance by increasing channel alphabet and introducing sequence decoding. In: *Proc. IEEE Int. Symp. Inform. Theory (ISIT)* (1976), Ronneby

7.227 G. Ungerboeck: Channel coding with multilevel/phase signals, IEEE Trans. Inf. Theory **28**(1), 55–67 (1982)

7.228 E. Biglieri, D. Divsalar, P.J. McLane, M.K. Simon: *Introduction to Trellis-Coded Modulation with Applications*, 2nd edn. (Prentice Hall, Englewood Cliffs 1992)

7.229 P. Robertson, T. Worz: Bandwidth-efficient turbo trellis-coded modulation using punctured component codes, IEEE J. Sel. Areas Commun. **16**(2), 206–218 (1998)

7.230 L.-F. Wei: Trellis-coded modulation with multidimensional constellations, IEEE Trans. Inf. Theory **33**(4), 483–501 (1987)

7.231 S. Benedetto, G. Olmo, P. Poggiolini: Trellis coded polarization shift keying modulation for digital optical communications, IEEE Trans. Commun. **43**(234), 1591–1602 (1995)

7.232 M. Magarini, R.J. Essiambre, B.E. Basch, A. Ashikhmin, G. Kramer, A.J. de Lind van Wijngaarden: Concatenated coded modulation for optical communications systems, IEEE Photon. Technol. Lett. **22**(16), 1244–1246 (2010)

7.233 H. Imai, S. Hirakawa: A new multilevel coding method using error-correcting codes, IEEE Trans. Inf. Theory **23**(3), 371–377 (1977)

7.234 U. Wachsmann, R.F.H. Fischer, J.B. Huber: Multilevel codes: theoretical concepts and practical design rules, IEEE Trans. Inf. Theory **45**(5), 1361–1391 (1999)

7.235 E. Zehavi: 8-PSK trellis codes for a Rayleigh channel, IEEE Trans. Commun. **40**(5), 873–884 (1992)

7.236 X. Li, J.A. Ritcey: Bit-interleaved coded modulation with iterative decoding, IEEE Commun. Lett. **1**(6), 169–171 (1997)

7.237 X. Li, J.A. Ritcey: Trellis-coded modulation with bit interleaving and iterative decoding, IEEE J. Sel. Areas Commun. **17**(4), 715–724 (1999)

7.238 S. ten Brink, J. Speidel, R.H. Han: Iterative demapping for QPSK modulation, Electron. Lett. **34**(15), 1459–1460 (1998)

7.239 A. Bennatan, D. Burshtein: Design and analysis of nonbinary LDPC codes for arbitrary discrete-memoryless channels, IEEE Trans. Inf. Theory **52**(2), 549–583 (2006)

7.240 G.D. Forney, F. Ungerboeck: Modulation and coding for linear Gaussian channels, IEEE Trans. Inf. Theory **44**(6), 2384–2415 (1998)

7.241 A.R. Calderbank, L.H. Ozarow: Nonequiprobable signaling on the Gaussian channel, IEEE Trans. Inf. Theory **36**(4), 726–740 (1990)

7.242 G.D. Forney: Trellis shaping, IEEE Trans. Inf. Theory **38**(2), 281–300 (1992)

7.243 F. Steiner, G. Böcherer: Comparison of geometric and probabilistic shaping with application to ATSC 3.0. In: *Proc. Int. ITG Conf. Source Channel Coding (SCC)* (2017), Hamburg

7.244 B.P. Smith, F.R. Kschischang: A pragmatic coded modulation scheme for high-spectral-efficiency fiber-optic communications, J. Lightwave Technol. **30**(13), 2047–2053 (2012)

7.245 L. Beygi, E. Agrell, J.M. Kahn, M. Karlsson: Rate-adaptive coded modulation for fiber-optic communications, J. Lightwave Technol. **32**(2), 333–343 (2014)

7.246 F. Buchali, F. Steiner, G. Böcherer, L. Schmalen, P. Schulte, W. Idler: Rate adaptation and reach

increase by probabilistically shaped 64-QAM: An experimental demonstration, J. Lightw. Technol. **34**(7), 1599–1609 (2016)
7.247 M.P. Yankov, F.D. Ros, E.P. da Silva, S. Forchhammer, K.J. Larsen, L.K. Oxenløwe, M. Galili, D. Zibar: Constellation shaping for WDM systems using 256QAM/1024QAM with probabilistic optimization, J. Lightwave Technol. **34**(22), 5146–5156 (2016)
7.248 A. Ghazisaeidi, I. Fernandez de Jauregui Ruiz, R. Rios-Müller, L. Schmalen, P. Tran, P. Brindel, A.C. Meseguer, Q. Hu, F. Buchali, G. Charlet, J. Renaudier: Advanced C+L-band transoceanic transmission systems based on probabilistically shaped PDM-64QAM, J. Lightwave Technol. **35**(7), 1291–1299 (2017)
7.249 A. Sheikh, A. Graell i Amat, G. Liva: Probabilistically-shaped coded modulation with hard decision decoding for coherent optical systems. In: *Proc. Eur. Conf. Opt. Commun. (ECOC)* (2017), Gothenburg
7.250 A. Sheikh, A. Graell i Amat, G. Liva, F. Steiner: Probabilistic amplitude shaping with hard decision decoding and staircase codes, J. Lightwave Technol. **36**(9), 1689–1697 (2018)
7.251 G. Böcherer, F. Steiner, P. Schulte: Bandwidth efficient and rate-matched low-density parity-check coded modulation, IEEE Trans. Commun. **63**(12), 4651–4665 (2015)
7.252 Alcatel-Lucent: *The 400G photonic service engine – leaping toward a future of faster speeds and higher capacity*, Alcatel-Lucent Strategic White Paper, Tech. Rep. (2012)
7.253 A. Leven, F. Vacondio, L. Schmalen, S. ten Brink, W. Idler: Estimation of soft FEC performance in optical transmission experiments, IEEE Photon. Technol. Lett. **20**(23), 1547–1549 (2011)
7.254 W. Idler, F. Buchali: Higher-order modulation formats – concepts and enabling devices. In: *Fibre Optic Communication*, Springer Ser. Opt. Sci., Vol. 161, ed. by H. Venghaus, N. Grote (Springer, Heidelberg, Berlin, New York 2017) pp. 291–357
7.255 F. Buchali, F. Steiner, G. Böcherer, L. Schmalen, P. Schulte, W. Idler: Rate adaptation and reach increase by probabilistically shaped 64-QAM: An experimental demonstration, J. Lightwave Technol. **34**(7), 1599–1609 (2016)
7.256 A. Alvarado, E. Agrell, D. Lavery, R. Maher, P. Bayvel: Replacing the soft-decision FEC limit paradigm in the design of optical communication systems, J. Lightwave Technol. **33**(20), 4338–4352 (2015)
7.257 A. Alvarado, E. Agrell, D. Lavery, R. Maher, P. Bayvel: Corrections to 'replacing the soft-decision FEC limit paradigm in the design of optical communication systems', J. Lightwave Technol. **34**(2), 722 (2016)
7.258 L. Schmalen, A. Alvarado, R. Rios-Müller: Predicting the performance of nonbinary forward error correction in optical transmission experiments. In: *Proc. Opt. Fiber Commun. Conf. (OFC)* (2016) p. M2A.2
7.259 L. Schmalen, A. Alvarado, R. Rios-Müller: Performance prediction of nonbinary forward error correction in optical transmission experiments, J. Lightwave Technol. **35**(4), 1015–1027 (2017)
7.260 L. Schmalen: Performance metrics for communication systems with forward error correction. In: *Proc. Eur. Conf. Opt. Commun. (ECOC)* (2018), Rome
7.261 M. Franceschini, G. Ferrari, R. Raheli: Does the performance of LDPC codes depend on the channel?, IEEE Trans. Commun. **54**(12), 2129–2132 (2006)
7.262 I. Sason, B. Shuval: On universal LDPC code ensembles over memoryless symmetric channels, IEEE Trans. Inf. Theory **57**(8), 5182–5202 (2011)
7.263 A. Sanaei, M. Ramezani, M. Ardakani: On the design of universal LDPC codes. In: *Proc. IEEE Int. Symp. Inform. Theory (ISIT)* (2008) pp. 802–806, Toronto
7.264 S.H. Hassani, R. Urbanke: Universal polar codes. In: *Proc. IEEE Int. Symp. Inform. Theory (ISIT)* (2014) pp. 1451–1455, Honolulu
7.265 T.A. Eriksson, T. Fehenberger, P.A. Andrekson, M. Karlsson, N. Hanik, E. Agrell: Impact of 4D channel distribution on the achievable rates in coherent optical communication experiments, J. Lightwave Technol. **34**(9), 2256–2266 (2016)
7.266 A. Alvarado, T. Fehenberger, B. Chen, F.M.J. Willems: Achievable information rates for fiber optics: applications and computations, J. Lightwave Technol. **36**(2), 424–439 (2018)
7.267 J. Cho, L. Schmalen, P. Winzer: Normalized generalized mutual information as a forward error correction threshold for probabilistically shaped QAM. In: *Proc. Eur. Conf. Opt. Commun. (ECOC)* (2017)
7.268 W. Idler, F. Buchali, L. Schmalen, E. Lach, R.-P. Braun, G. Böcherer, P. Schulte, F. Steiner: Field trial of a 1 Tb/s super-channel network using probabilistically shaped constellations, J. Lightwave Technol. **35**(8), 1399–1406 (2017)
7.269 L. Schmalen, F. Buchali, A. Leven, S. ten Brink: A generic tool for assessing the soft-FEC performance in optical transmission experiments, IEEE Photon. Technol. Lett. **24**(1), 40–42 (2012)
7.270 N. Stojanovic, Y. Zhao, D. Chang, Z. Xiao, F. Yu: Reusing common uncoded experimental data in performance estimation of different FEC codes, IEEE Photon. Technol. Lett. **25**(24), 2494–2497 (2013)
7.271 J. Hou, P.H. Siegel, L.B. Milstein, H.D. Pfister: Capacity-approaching bandwidth-efficient coded modulation schemes based on low-density parity-check codes, IEEE Trans. Inf. Theory **49**(9), 2141–2155 (2003)
7.272 R. Rios-Müller, J. Renaudier, L. Schmalen, G. Charlet: Joint coding rate and modulation format optimization for 8QAM constellations using BICM mutual information. In: *Proc. Opt. Fiber Commun. Conf. (OFC)* (2015) p. W3K.4
7.273 R. Rios-Müller, J. Renaudier, P. Tran, G. Charlet: Experimental comparison of two 8-QAM constellations at 200 Gb/s over ultra long-haul transmission link. In: *Proc. Eur. Conf. Opt. Commun. (ECOC)* (2014) p. P.5.1, Cannes
7.274 Z. Wang: Super-FEC codes for 40/100 Gbps networking, IEEE Commun. Lett. **16**(12), 2056–2059 (2012)
7.275 G. Tzimpragos, C. Kachris, I.B. Djordjevic, M. Cvijetic, D. Soudris, I. Tomkos: A survey on FEC codes for 100 G and beyond optical networks, IEEE Commun. Surv. Tutor. **18**(1), 209–221 (2016)

7.276 K. Lee, H.-G. Kang, J.-I. Park, H. Lee: 100GB/S two-iteration concatenated BCH decoder architecture for optical communications. In: *Proc. IEEE Workshop Signal Process. Syst. (SiPS)* (2010) pp. 404–409
7.277 M. Scholten, T. Coe, J. Dillard, F. Chang: Enhanced FEC for 40G / 100G. In: *Proc. Eur. Conf. Opt. Commun. (ECOC)* (2009), Vienna, presentation in Workshop WS-1
7.278 J. Li, K. Liu, S. Lin, K. Abdel-Ghaffar: Algebraic quasi-cyclic LDPC codes: Construction, low error-floor, large girth and a reduced-complexity decoding scheme, IEEE Trans. Commun. **62**(8), 2626–2637 (2014)
7.279 Viasat Inc.: Viasat SDFEC 66200 – 200 Gbps SDFEC, https://www.viasat.com/sites/default/files/media/documents/sdfec_66200_datasheet_003_web_0.pdf (2017)
7.280 B. Smith, I. Lyobomirsky, S. Bhoja: Leveraging 400G ZR FEC technology. In: *IEEE 802.3 Beyond 10km Optical PHYs Study Group* (2017), Orlando
7.281 Viasat Inc.: ECC66100 Series SD-FEC Encoder/Decoder Cores, https://www.viasat.com/sites/default/files/media/documents/ecc_66100_datasheet_2_pgr_007_web_0.pdf (2017)
7.282 K. Cushon, P. Larsson-Edefors, P. Andrekson: Low-power 400-Gbps soft-decision LDPC FEC for optical transport networks, J. Lightwave Technol. **34**(18), 4304–4311 (2016)
7.283 K. Cushon, P. Larsson-Edefors, P. Andrekson: Improved low-power LDPC FEC for coherent optical systems. In: *Proc. Eur. Conf. Opt. Commun. (ECOC)* (2017) pp. 1–3, Gothenburg
7.284 K. Onohara, Y. Miyata, K. Sugihara, T. Sugihara, K. Kubo, H. Yoshida, K. Koguchi, T. Mizuochi: Implementation of soft-decision forward error correction for 100G digital coherent system. In: *Proc. Opto-Electron. Commun. Conf. (OECC)* (2011) pp. 423–424

**Alexandre Graell i Amat**

Dept. of Electrical Engineering
Chalmers University of Technology
Gothenburg, Sweden
*alexandre.graell@chalmers.se*

Alexandre Graell i Amat received his PhD from Politecnico di Torino, Italy, in 2004. He is currently a professor at the Department of Electrical Engineering of Chalmers University of Technology, Sweden. His research interests are in the field of coding theory and its application to distributed storage and computing, privacy, and fiber-optic communications.

**Laurent Schmalen**

Communications Engineering Lab (CEL)
Karlsruhe Institute of Technology (KIT)
Karlsruhe, Germany
*laurent.schmalen@kit.edu*

Laurent Schmalen received his PhD from RWTH Aachen University in Germany in 2011. Until 2019, he worked as a researcher and Department Head at Nokia Bell Labs in Stuttgart, Germany. He is now a professor at Karlsruhe Institute of Technology (KIT). His work focuses on forward error correcting codes, digital modulation schemes, and machine learning algorithms for high-speed data transmission.