Low-Density Parity-Check Codes: Principles and Practice

Fengfan Yang[+] and Tho Le-Ngoc[*]

[+]College of Information Science and Technology

Nanjing University of Aeronautics and Astronautics (NUAA)

Yu Dao St. No. 20, Nanjing, 210016, Jiangsu province

People's Republic of China

E-mail: yffee@nuaa.edu.cn, Tel: 86 25 84895951

[*]Department of Electrical and Computer Engineering

McGill University, 3480 University Street

Montreal, Quebec, Canada, H3A 2A7

E-mail: tho.le-ngoc@ece.mcgill.ca, Tel: 1 514 398 5252

Chapter 1 Introduction

1.1 Modern Digital Communications with Error-Correcting Codes

With the advent of high-speed logic circuit and very large-scale integration (VLSI), date processing and storage equipment has inexorably moved towards employing digital techniques. Data is encoded into strings of zeros and ones in digital systems, corresponding to the on and off states of semiconductor switches. This has brought about fundamental changes in how information is processed. Real-world data is typically in analog form; this is the only way we can perceive it with our sense. This analog information needs to be encoded into a digital representation, for example, into a string of ones and zeros. The conversion from analog to digital and back ate processes that have becomes ubiquitous, as, for example, in the digital encoding of speech and video, etc.

Digital information is processed differently in communications than analog information. Signal estimation becomes signal detection; that is, a communications receiver need not look for an analog signal and make a “best” estimate, it only needs to make a decision between a finite number of discrete signals, say a one or zero in the most basic case. Digital signals are more reliable in a noisy communications environment. They can usually be detected perfectly, as long as the noise levels are below a certain threshold. This allows us to restore digital data, and, through error-correcting techniques, even correct errors made during transmission. Digital data can easily be encoded in such a way

as to introduce dependency among a large number of symbols. This is called *error control coding*.

The digitization of data is convenient for a number of other reasons too. The design of signal processing algorithms for digital data seems much easier than designing analog signal processing algorithms. The abundance of such digital algorithms, including error control and correction techniques, combined with their ease of implementation in very large-scale integrated (VLSI) circuits, has led to many successful applications of error control coding in practice. Fig. 1.1 shows a communication system for transmitting information for a source to a destination through a channel.

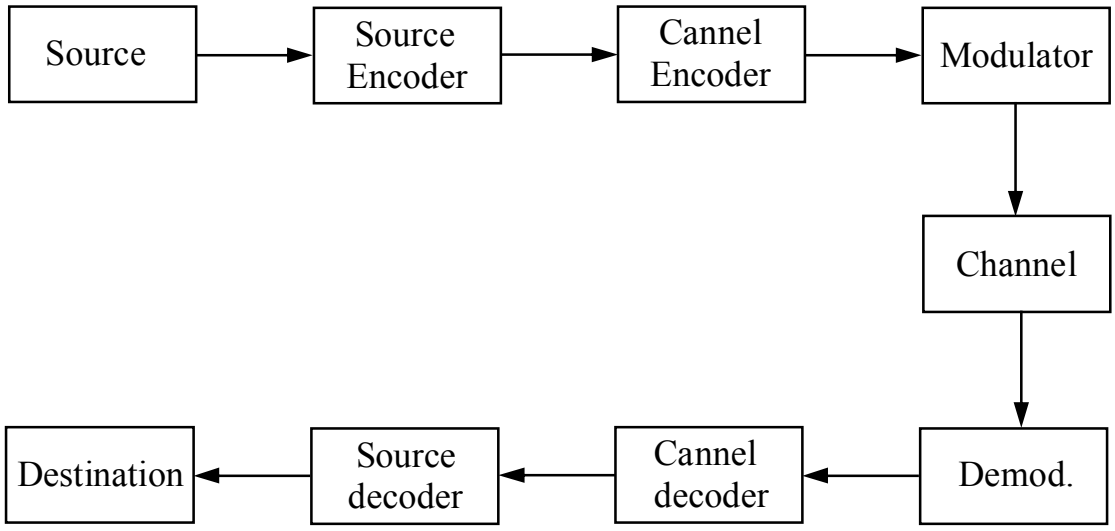

Fig. 1.1 Communication system with error control coding.

Error control coding was first applied in deep-space communications where we are confronted with low-power communications channels with virtually unlimited bandwidth.

On these data links, convolutional codes are employed with sequential and Viterbi decoding, and the future will see the application of turbo coding and low-density parity-check (LDPC) coding that is the main theme of this book. The next successful application of error control coding is to storage devices, most notably the compact disk player, which uses powerful Reed-Solomon codes [89] to handle the raw error probability from the optical readout device that is too large fro high-fidelity sound reproduction without error correction. Another hurdle taken was the successful application of error control to bandwidth-limited telephone channels, where trellis-coded modulation (TCM) [90] was adopted to produce impressive improvements that push transmission rates right up toward the theoretical limit of the channel. Nowadays, coding is routinely applied to satellite communications [91] teletext broadcasting, computer storage devices, logic circuits, semiconductor memory systems, magnetic recording systems, audio/video systems, and modern mobile communications systems like the pan-European TDMA digital telephony standard GSM [92], IS-95 [93], CDMA-2000, and IMT-200, all new digital cellular standards using spread spectrum techniques.

1.2 Theoretical Limits and Potentials of Error Control Coding

The modern approach to error control in digital communications started with the ground-breaking work of Shannon [94], Hamming and Golay [95]. While Shannon established a theory that explained the fundamental limits on the efficiency of communications systems, Hamming and Golay were the first to develop practical error

control schemes. A new paradigm was born, one in which errors are not synonymous with data that is irretrievably lost, but a proper design, errors could be corrected, or avoided altogether. This new thinking was revolutionary. But even through Shannon's theory promised that large improvements in the performance of communication systems could be achieved, practical improvements had to be excavated by laborious work over half a century of intensive research. Indeed, Shannon's theory clearly states fundamental limits on communication efficiency, but it methodology provides no insight on how to actually achieve these limits, because it is largely based on sophisticated averaging arguments that eliminate all detailed system structure. Coding theory, on the other hand, evolved from Hamming and Golay's work into a flourishing branch of applied mathematics [95].

The most of famous formula from Shannon's work is arguably the channel capacity of an ideal band-limited Gaussian channel, which is given by

$$C = W \log_2(1 + S/N) \quad \text{bits/second} \tag{1-1}$$

where C is the channel capacity, that is the maximum number of information bits that can be transmitted through this channel per unit time (second), W is the bandwidth of th channel, and S/N is the signal-to-noise power ratio at the receiver.

The parameter that characterizes how efficiently a system uses its allotted bandwidth is the bandwidth efficiency η, defined as

$$\eta = \frac{\text{Tansmission rate}}{\text{Channel bandwidth W}} \tag{1-2}$$

Therefore, we obtain the maximum bandwidth efficiency for an additive white Gaussian noise channel, the Shannon limit, as

$$\eta_{\max} = \log_2(1 + S/N) \text{ bits/s/HZ} \tag{1-3}$$

The average signal power S can be expressed as

$$S = \frac{kE_b}{T} \tag{1-4}$$

where E_b is the energy per bit, k is the number of bits transmitted per symbol, and T is the duration of a symbol. The parameter $R = k/T$ is the transmission rate of the symbol in bits/s. Rewriting the signal-to-noise power ratio S/N, where $N = WN_0$, i.e., the total noise power equals the one-sided noise power spectrum density N_0 multiplied by the transmission bandwidth. Thus, based on (1-3) we obtain the Shannon limit in terms of the bit energy and noise power spectrum density as

$$\eta_{\max} = \log_2\left(1 + \frac{RE_b}{WN_0}\right) \tag{1-5}$$

Note that $R/W = \eta_{\max}$ is the limiting spectral efficiency when the *k* bits per symbol in (1-4) are all information bits, we obtain a bound from (1-5) on the minimum ratio of bit energy to one-sided noise spectrum density for reliable transmission at a given spectrum efficiency, which is also called the *Shannon bound*.

$$\frac{E_b}{N_0} \geq \frac{2^{\eta_{\max}} - 1}{\eta_{\max}} \tag{1-6}$$

In the limit as the signal is allowed to occupy an infinite amount of bandwidth, namely $\eta_{\max} \to 0$. We get

$$\frac{E_b}{N_0} \geq \lim_{\eta_{\max}\to 0} \frac{2^{\eta_{\max}}-1}{\eta_{\max}} = \left.\frac{d(2^{\eta_{\max}}-1)}{d\eta_{\max}}\right|_{\eta_{\max}=0}$$

$$= \ln 2 = -1.59\text{dB} \tag{1-7}$$

Shannon theorem, which accompanies (1.1), asserts that error probabilities as small as desired can be achieved as long as the transmission rate R through the channel in bits/second is smaller than the channel capacity C. This can be achieved by using an appropriate encoding and decoding operation. However, Shannon theorem is silent about the structure of these encoders and decoders.

Example 1.1: Consider BPSK modulation scheme, the number of bits per symbol k is 1, and the symbol duration is the same as the bit duration that is denoted as T_b seconds. Hence, the transmission rate $R = 1/T_b$ bits/s. Thus, the bandwidth of the channel is $W = 1/T_b$. The maximum (limiting) spectral efficiency $\eta_{\max}$ is

$$\eta_{\max} = R/W$$

$$= \frac{1/T_b}{1/T_b} = 1 \tag{1-8}$$

By (1-6) the minimum ratio of bit energy to noise power spectral density is

$$\frac{E_b}{N_0} = \frac{2^1-1}{1}$$

$$=1=0\text{dB} \tag{1-9}$$

Example 1.2: Consider QPSK modulation scheme, the number of bits per symbol k is 2, and let the symbol duration be T_s seconds. Hence, the transmission rate $R=2/T_s$ bits/s. If the two carriers $\sin(2\pi f_c t)$ and $\cos(2\pi f_c t)$ are employed to in a quadrature transmission, then the bandwidth in each in-phase (I) and quadrature (Q) channel is the sane, i.e., $W=1/T_s$. The maximum (limiting) spectral efficiency $\eta_{\max}$ is

$$\eta_{\max}=R/W$$

$$=\frac{2/T_s}{1/T_s}=2 \tag{1-10}$$

By (1-6) the minimum ratio of bit energy to noise power spectral density is

$$\frac{E_b}{N_0}=\frac{2^2-1}{2}$$

$$=1.5=1.761\text{dB} \tag{1-11}$$

Therefore, thresholds in terms of E_b/N_0 (dB) for BPSK and QPSK modulation schemes are 0dB and 1.761dB, respectively.

Fig. 1.2 shows the error performance of QPSK scheme [96], which allows data transmission up to 2 bits/symbol. The bit error probability of QPSK is shown as a function of the signal-to-noise ratio (SNR), defined by the ration of bit energy to noise power spectral density, i.e., E_b/N_0 (dB). Evidently, an increased SNR provides a gradual decrease in error probability. This contrasts with the Shannon limited marked in the figure,

which promises zero probability at the maximum spectral efficiency of 2 bits/s/Hz as long as SNR>1.5dB (1.76dB) obtained in Example 1.2.

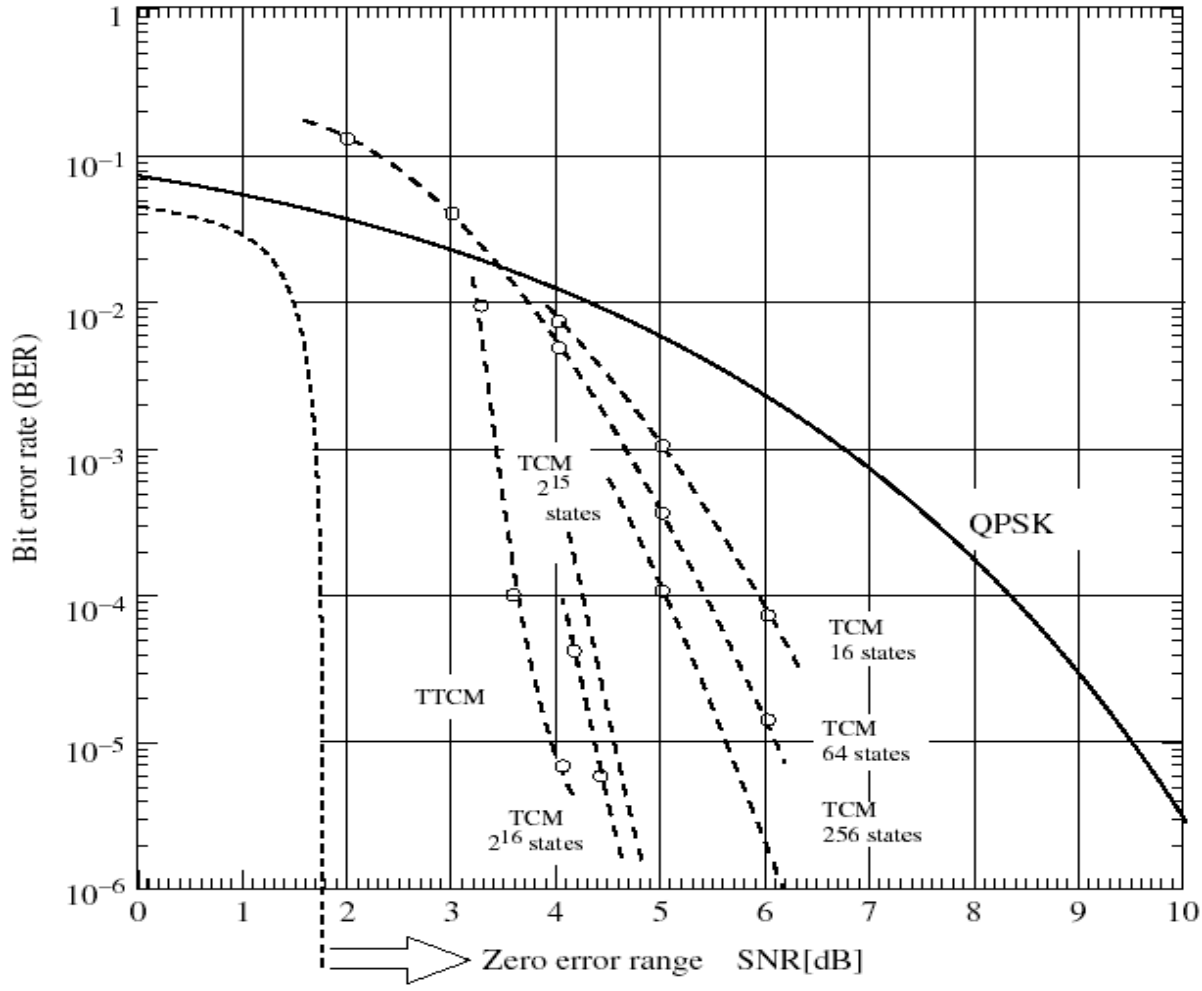

Fig. 1.2 Bit error probability of quadrature phase-shift keying (QPSK) and selected 8-PSK trellis-coded modulation (TCM) and trellis-turbo-coded modulation (TTCM) systems as a function of the normalized signal-to-noise ratio.

Fig. 1.2 also provides the performance of several trellis-code modulation (TCM) and trellis-coded modulation (TCM) and trellis-turbo-coded modulation (TTCM) schemes using 8-ary phase-shift keying (8-PSK), and the improvement made possible by coding

becomes evident. The difference in SNR for an objective target bit error rate between a coded system and an uncoded system is termed the coding gain, which clearly indicates the potentials for the practical coding schemes and the related ultimate theoretical limit. Note that the coding schemes shown in Fig. 1.2 achieve these gains without requiring more bandwidth than the uncoded QPSK system.

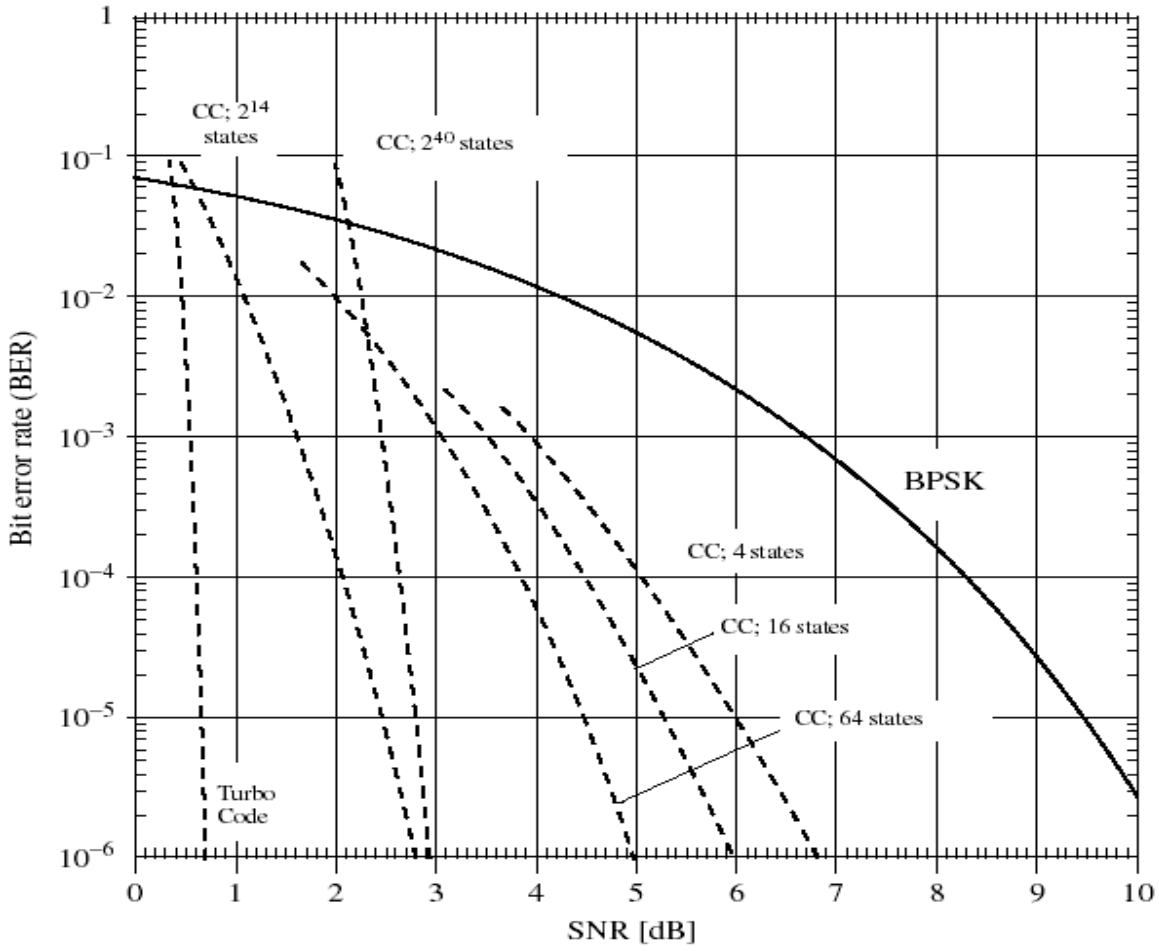

Fig. 1.3 Bit error probability of rate R=1/2 convolutional and turbo codes with BPSK modulation scheme as a function of signal-to-noise ratio (SNR). The large-state space code is decoded sequentially, while the performance of all the other convolutional codes is for maximum likelihood decoding. The Shannon limit is at SNR=0dB.

Fig. 1.3 offers the performance of rate R=1/2 bits/symbol convolutional and turbo codes over an additive white Gaussian channel (AWGN) [96]. Compared with TCM, these codes do not preserve bandwidth and the gains in power efficiency are partly obtained by a power bandwidth tradeoff, i.e., the rate 1/2 convolutional codes require twice as much bandwidth as uncoded transmission. As a consequence, for the same complexity, a higher coding gain can be achieved than with TCM. The Shannon limit for BPSK modulation scheme is at 0dB obtained by the Example 1.1.

Note that both TCM with QPSK modulation and rate-1/2 convolutional codes with BPSK modulation have far away from the corresponding ultimate limits. In other words, we may still have great potentials for approaching the goals by using proper coding strategies. For example, the 256-state 2/3-TCM as shown in Fig. 1.2, which is frequently used over satellite link, is about 4.44dB away from its theoretical limit if we regard bit error rate at 10^{-6} as an approximate error-free transmission, and the rate-1/2 64-state convolutional code as shown in Fig. 1.3 is about 5dB away its limit at a bit error rate of 10^{-6}. However, a rate-1/2 turbo coding in Fig. 1.3 can achieve impressive performance with only 0.7dB [10] away from Shannon limit by the well designed interleaver and sufficient long block. This is one of the important reasons that we are particularly interested in the iteratively decodable codes, such as turbo codes and low-density parity-check (LDPC) codes, especially for the latter one that is the main theme in this book.

1.3 Two Phenomenal Iteratively Decodable Codes

Since the advent of the successful turbo codes [1] that is recognized as parallel concatenated convolutional codes and can be effectively decoded by some high-performance iterative approaches, another class of block codes exhibiting similar characteristics and performance was rediscovered [5][6]. This class of codes, which is called *low-density parity-check* (LDPC) codes was first introduced by Gallager in his early thesis [1]. During the long period between Gallager's thesis and the invention of turbo codes, LDPC codes and their invariants were largely ignored partially due to the reason that other kinds of block codes, e.g., various cyclic codes, and convolutional codes were widely used in many applications by maximum likelihood decoding and other suboptimal approaches.

In fact, there is a close resemblance between LDPC and turbo codes. Both can achieve impressive error performance [10][12][35] that is very close to the ultimate Shannon limit by using similar iterative decoding algorithms. LDPC codes, turbo codes have contributed an significant and also long-standing impact on coding theory by providing the effective capacity-approaching iterative decoding algorithms that have been naturally extended to serially concatenated convolutional/block codes [100], and even more generalized turbo-like codes and processing, repeat-accumulate (AC) codes [97], compound codes [23][24] depicted by graph model, turbo multi-user detection [98] and turbo equalization [99], etc.

1.3.1 Turbo Codes and Low-Density Parity-Check Codes

In order to introduce the fundamentals of LDPC codes, it is necessary to revisit the essentials of turbo codes. A typical example of a rate-1/3 turbo encoder with two identical constituent recursive systematic convolutional (RSC) codes is shown in Fig. 1.4.

(1) Turbo codes:

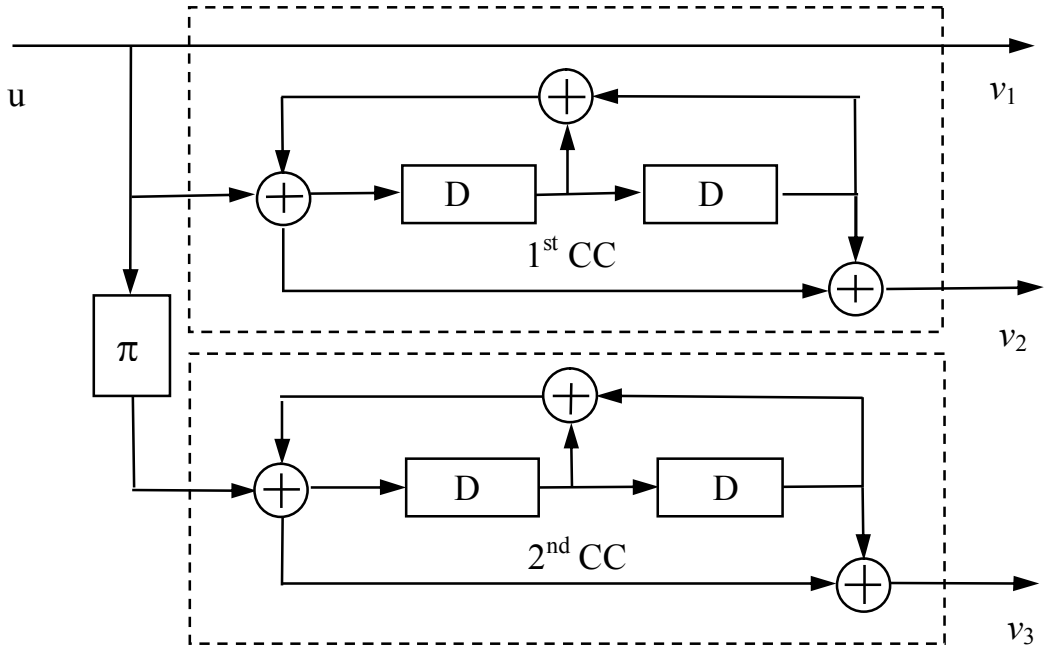

Fig. 1.4 An example of a rate-1/3 turbo encoder with two identical constituent codes (CC's).

Turbo codes are constructed by applying two or more simple-to-decode codes to differently permuted versions of the same information sequence. The corresponding turbo decoding algorithm iterates the outputs of simple decoder acting on each constituent code. Good turbo codes usually use short constraint length, infinite impulse response (IIR) convolutional codes as components instead of the more familiar finite impulse response

(FIR) convolutional codes. These IIR convolutional codes are also referred to as recursive convolutional codes because previously encoded information bits are continuously fed back to the encoder's input. For each of IIR and FIR convolutional encoders, a single isolated information-bit error will produce the same encoded sequence wherever it is permuted within the information sequence. For an IIR encoder, this impulse response has infinite weight, while for a FIR encoder the weight of its response to a single isolated bit error cannot be much larger than the code's free distance. Thus, the IIR property is extremely important for building turbo codes, because it avoids low-weight encodings that are impervious to the action of the interleaver.

Next, we will briefly explain the two fundamentals, i.e., why the turbo codes apply the well-constructed recursive systematic convolutional (RSC) codes and random-like interleaver(s)? How the two key ingredients can jointly work to achieve a low bit error rate (BER) at very low signal-to-noise ratios (SNR's) over an AWGN channel?

Why Recursiveness and Interleaving

Fig. 1.4 presents a rate-1/3 turbo encoder with two simple constituent RSC encoders of constraint length K=2. The first output stream is the uncoded information sequence. The other two output streams are parity-check sequences related to a ratio of generator polynomials with relatively prime, $g_b(D)/g_a(D)$, where $g_a(D)$=1+D+D^2 and $g_b(D)$=1 +D^2 are the feedback and feedforward polynomials with degree 2, respectively. Evidently, the two streams would be identical if no permutation π were used in-between. The trellis

section of a rate-1/2 systematic convolutional code adopted by the first constituent RSC code (with the uncoded stream) is shown in Fig. 1.5 with the generator matrix $G(D)$=[1, $g_b(D)/g_a(D)$]. The minimum Hamming distance of the convolutional code is 5 with a path (or codeword) indicated by the dashed lines. This codeword is produced by a weight-3 input information sequence (111000…) with three consecutive 1's, and weight of the parity sequence is 2.

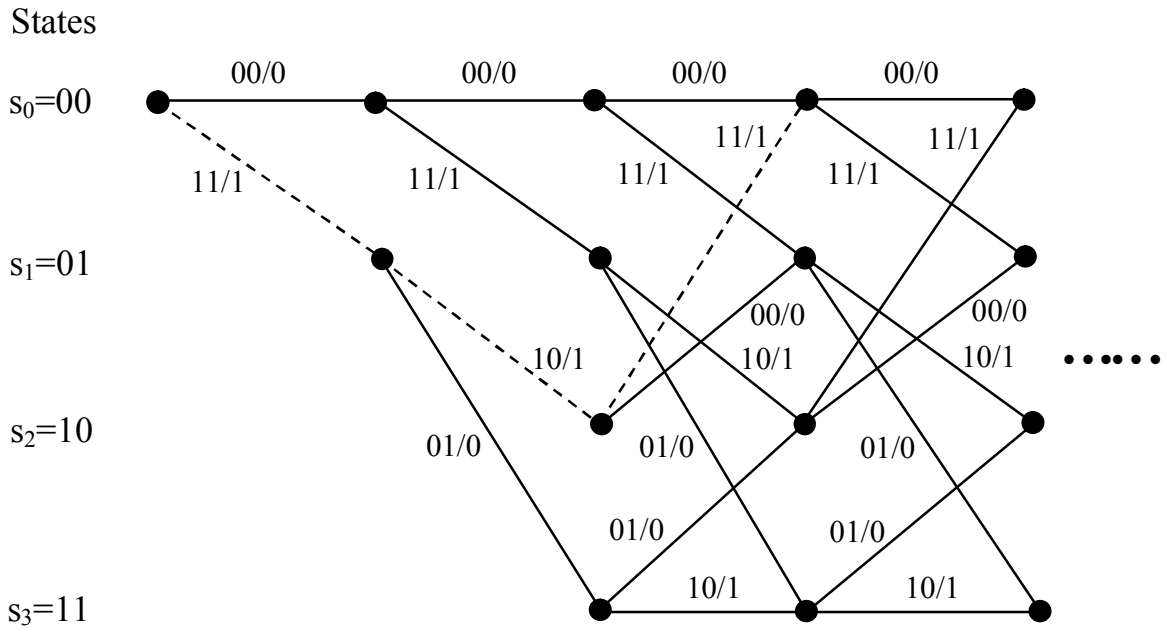

Fig. 1.5 The trellis section of a rate-1/2 RSC code with generator matrix $\boldsymbol{G}(D)$.

The weight distribution for the codewords generated by an overall turbo encoder depends on how the codewords from each of the constituent encoders are teamed with the codewords from its counterpart one.

(a) Why recursiveness?

Let us take a very simple example to show why adopt the convolutional codes of recursive type other than the non-recursive type as the constituent codes for the turbo codes.

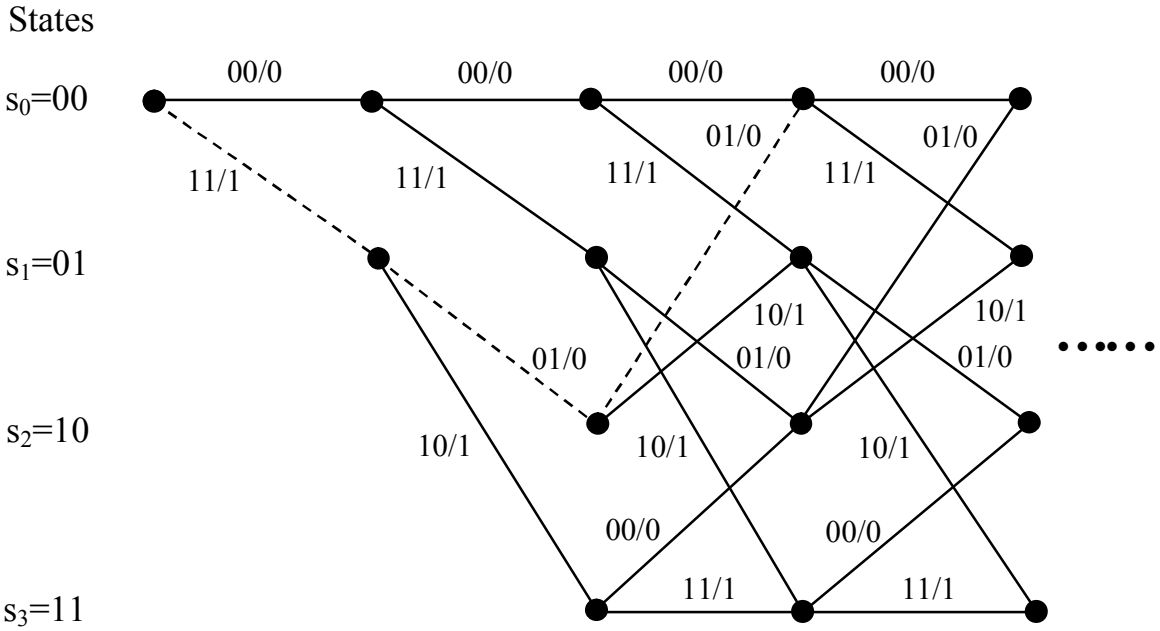

Fig. 1.6 The trellis section of a rate-1/2 non-recursive systematic convolutional code with generator matrix [1, $g_a(D)$].

Example 1.3: Consider a rate-1/2 non-recursive systematic convolutional code whose generator polynomial is $\boldsymbol{G}(D)$=[1, $g_a(D)$]. The path associated with the unit weight information sequence is marked by the dotted lines shown in Fig. 1.6 with the associated codeword weight 4, where the related parity-check sequence weight is 3. If $g_a(D)$ is chosen as the parity-check polynomial for both component convolutional codes of the rate-1/3 turbo code, then the overall weight of the resultant codeword is only 1+2×3=7

for the single "1" input sequence. Similarly, the weight is only 1+2×2=5 while $g_b(D)$ is adopted for the component convolutional codes of the turbo code. The interested reader can easily obtain these results. □

If the component RSC codes are adopted by the same turbo code as illustrated in Fig. 1.4, then the weight-1 input information sequence will create the output parity-check sequence as an IIR for each constituent code and thus result in an overall codeword of the turbo code with infinite weight, which is much larger than that of the corresponding codeword by non-recursive type in the Example 1.3. The output coded weight is kept finite only for the trellis termination at the end of the block.

Similarly, the parity-check sequences generated by non-recursive component convolutional codes are also finite weights for other input patterns, such as weight-2, 3, etc., which more likely generate infinite weights parity-check sequences by recursive convolutional codes.

A simple single encoder, whose low (high) output weights are resulted from high (low) input weights, is generally undesired for conventional use where the ultimate goal is to minimizing the decoded BER of information bits. However, it is precisely this characteristic that enables such an encoder very desirable as a component of a turbo code. This motivates the use of recursive convolutional encodes in the turbo code, where the key ingredient is the recursiveness and not the fact that the constituent codes are systematic.

(b) Why interleaving?

Another kind of important information sequence except the aforementioned weight-1 one is the weight-2 input sequence (0…010…010… 0) because it may produces relatively low weights at the outputs of each of the component encoders.

Example 1.4: A special weight-2 input sequence (10010…0) is sent to the component RSC encoders shown in Fig. 1.4 and also self-terminating if no permutation before the input to the component codes. Let us revisit the turbo code shown in Fig. 1.4, if no permutation is employed, then the polynomials of both parity-check sequences are calculated as

$$v_1(D)=v_2(D)=(1+D^3)\frac{1+D^2}{1+D+D^2}$$

$$=(1+D)(1+D+D^2)\frac{1+D^2}{1+D+D^2}$$

$$=1+D+D^2+D^2 \qquad (1\text{-}12)$$

The weight of each parity-check sequence is only 4, and thus the total codeword weight of the turbo encoder is 2+2×4=10. □

We now consider the scenario where a nontrivial permutation π is used for the turbo encoder.

Example 1.5: Let the weight-2 input sequence for the first component RSC encoder be the same as that in the Example 1.4 and the input sequence for the second RSC encoder

in terms of polynomial be $1+D^{12}$ after a nontrivial interleaver π. Therefore, the polynomial of the second parity-check sequence is

$$v_2(D)=(1+D^{12})\frac{1+D^2}{1+D+D^2}$$

$$=(1+D^3)(1+D^3+D^6+D^9)\frac{1+D^2}{1+D+D^2}$$

$$=1+D+D^2+D^4+D^5+D^7+D^8+D^{10}+D^{11}+D^{12} \quad (1\text{-}13)$$

The weight of each parity-check sequence becomes 10, thus the total codeword weight of the turbo encoder is significantly increased to 2+4+10=16 as compared with the scenario in the Example 1.4. □

Generally, the crucial strategy hidden behind turbo coding is to match low-weight encodings of one input version with high-weight encodings of the other, thus resulting total weight is significantly higher than the both low weights that are inevitably produced from each of the simple component codes. Good distance properties will yield a desirable error performance when maximum likelihood decoding (MLD) or other suboptimal soft-input soft-output (SISO) approaches are employed, such as maximum *a posterior* (MAP) algorithm [10], soft out Viterbi algorithm (SOVA) algorithm, Max-Log-MAP and Log-MAP Algorithms [101], etc. Therefore, recursiveness and interleaving are two crucial ingredients that enable turbo codes to be capacity-approaching error-correction codes.

(**2**) **Low-Density Parity-Check (LDPC) codes**:

Low-density parity-check (LDPC) codes are another kind of iteratively decodable error-correction codes, which are defined by simple sparse parity-check matrix and decoded by the simple and practical belief propagation (BP) algorithm [1][5][6].

Before commencing the introduction of LDPC codes in Chapter 4, we will use the following example to illustrate the essentials of a regular LDPC code.

Example 1.6: Consider a parity-check matrix $\mathbf{H}_{5\times 20}$ in (1-14) that defines a (20, 15) block code. There are four 1's in each row, which implies that four bits participate in a parity-check equation. On the other hand, there is a single 1 in each column, which means that each bit only attends one parity-check equation. Evidently, the number of 1's in the matrix is only 5×4=20 and relatively sparse in contrast to a randomly chosen matrix, in which the numbers of 1's and 0's are the same, namely 50. Hence, it is called a sparse parity-check matrix that defines a (20, 15) low-density parity-check (LDPC) code.

$$H_{5\times 20} = \begin{pmatrix} 1 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 1 \end{pmatrix} \quad (1\text{-}14)$$

Meanwhile, we may depict the check relationships given by (1-14) in a bipartite graph, which will be introduced in the next chapter. Intuitively, a bipartite graph consists of two kinds of nodes, which represent the codeword bits and checks, respectively, and an edge is only linked between a pair of nodes of different kinds as long as a codeword bit

participates in a specified check-sum. The illustrative bipartite graph for (20, 15) block code defined by (1-14) is shown in Fig. 1.7, where the circle nodes, denoted by v_1, v_2, …, v_{20}, represent all the codeword bits and square nodes, denoted by c_1, c_2, …, c_6, represent all the checks. For instance, the four bits, v_1, v_4, v_6, v_8, are connected with the check node c_1 since they attend the check-sum corresponding to the first row in (1-14). □

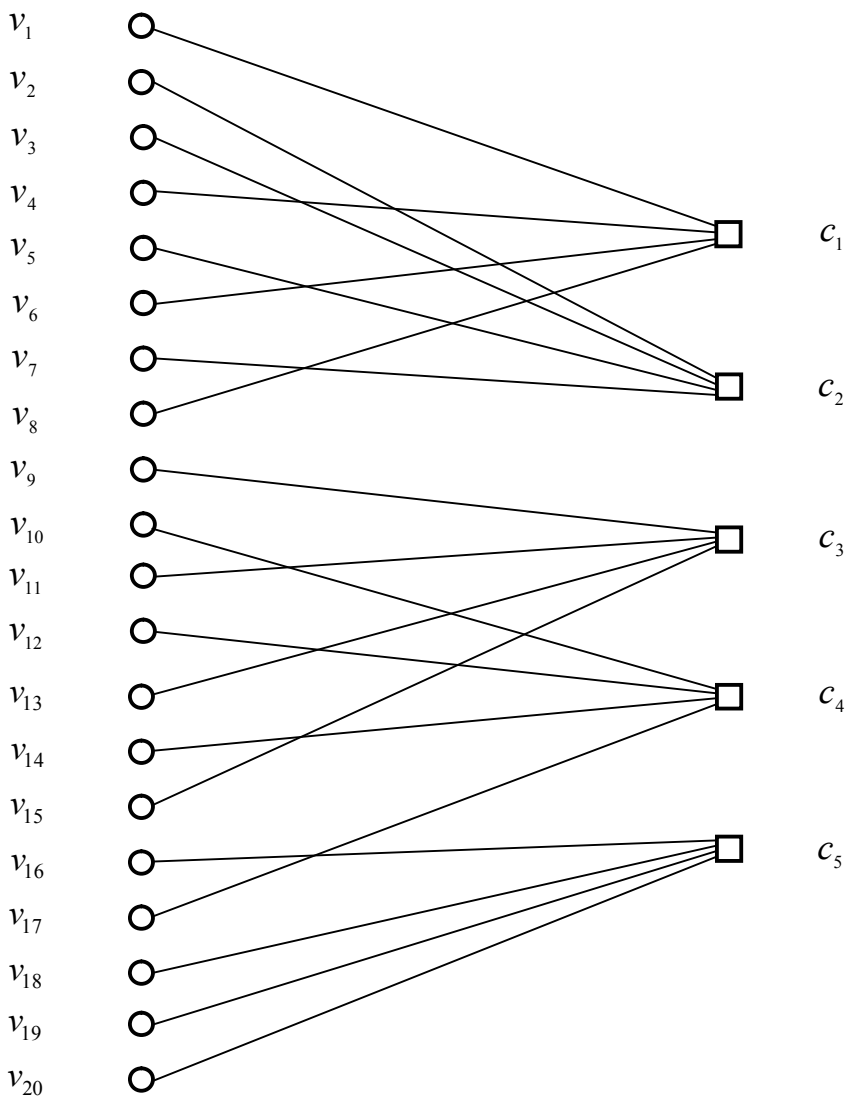

Fig. 1.7 The illustrative bipartite graph for the (20, 15) block codes defined by the sparse matrix in (1-14).

There are two interesting and intuitive interpretations for the design of LDPC codes by parity-check matrix or associated bipartite graph.

(1) **Two different groups of players for a game**: It would be convenient to think of the construction process as an interesting game over a bipartite graph, where the nodes for codeword bits are recognized as one group of players and the nodes for checks are regarded as another group of players. After the game begins, each player in every group is trying to choose the right number of edges. A constraint on the game is that the nodes of both groups must agree on the total number of edges. In view of parity-check matrix, the two groups of players are equivalently regarded as columns and rows of the matrix, respectively, which are trying to get the right number of 1's in the game.

(2) **A random permutation for the edges**: Assume that the total number of edges is e in a bipartite graph, a random permutation π of $\{1, 2, \ldots, e\}$ is chosen. Then, for all $i \in \{1, 2, \ldots, e\}$, the edge with index i out of the left side is identified with the edge with index $\pi(i)$ out of the right side. More generally and systematically, we can arrange e sockets, numbered by 1, 2, …, e, in each side of the graph. Suppose an edge out of the left side be related to the i-th socket on the same side, and associated with a socket with index $\pi(i)$ on the right side. Hence, we establish an on-to-one correspondence of e sockets on both sides by e edges. The nodes are then trying to choose the right number of sockets in each side, which has the similar views as the aforementioned players for a game.

The degree of a socket is defined as the number of edges connected. It is at least one for a left socket and at least two for a right socket. This guarantees that a codeword bit participates in at least one check equation, and any check equation has at least two codeword bits involved.

Generally, we can image that it may exist a reasonable rule for the game, which enables each player with a proper number of edges and eventually results in good error performance for the optimized LDPC code over a noisy channel.

1.3.2 Error Performance Comparison for LDPC Codes and Turbo Codes

The comparison between the two phenomenal iteratively decodable codes, i.e., low-density parity-check codes and turbo codes, in many aspects is always an interesting question. In this chapter we only present a brief comparison with respect to bit error rate (BER) for both codes.

It is reported in [8][9] that the performance of regular LDPC codes over a binary-input additive white Gaussian noise (BIAWGN) channel is only slightly inferior to that of turbo codes. For example, a rate-1/2 LDPC code of block length 10^4 bits requires an E_b/N_0 of roughly 1.4dB to achieve a bit error probability of 10^{-5}, whereas an equivalent turbo code with comparable complexity achieves the same performance at roughly E_b/N_0=0.8dB [10]. In order to achieve reliable transmission (error-free transmission) at a rate of one-half bit per channel symbol using BPSK modulation scheme over the

continuous-input AWGN channel, the maximum spectral efficiency $\eta_{\max}$ =1/2 and by Shannon bound in (1-6) the minimum E_b/N_0 is

$$E_b/N_0 = \frac{2^{1/2}-1}{1/2} = -0.8176\text{dB} \tag{1-15}$$

For convenience, we often use 0dB [8][10] as a reference for rate-1/2 coding scheme with BPSK modulation. The limit value increases to 0.187dB [8][9] as indicated in Fig. 1.8 if we restrict the input to be binary.

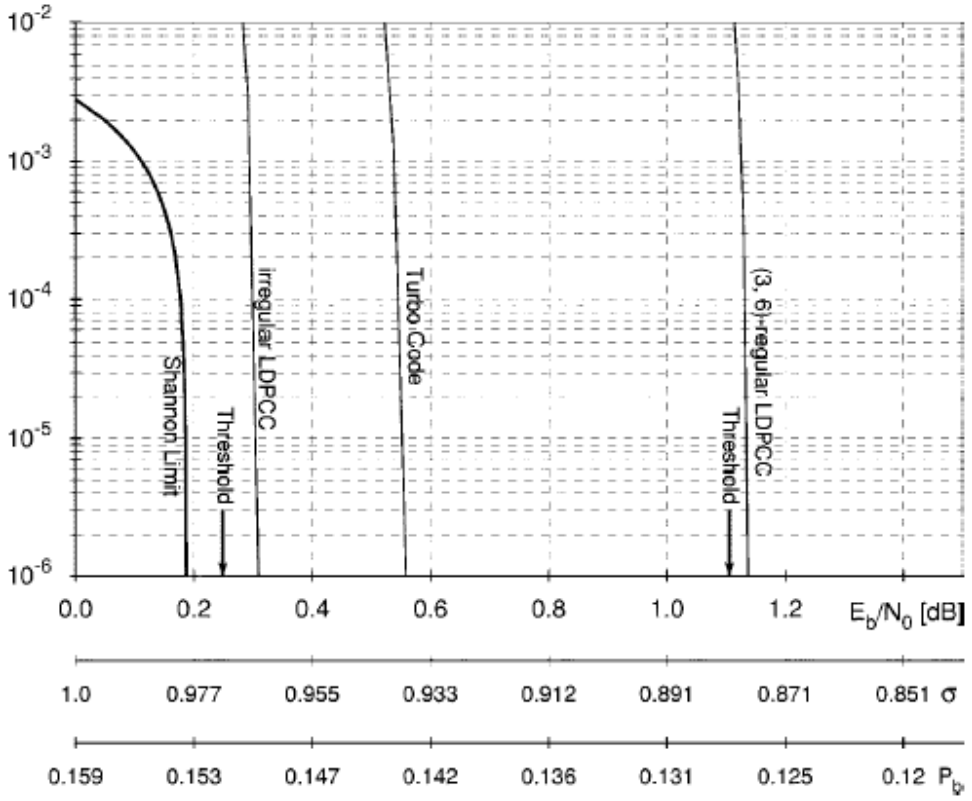

Fig. 1.8 Comparison of bit error rate for a (3, 6) regular LDPC code, turbo code and the optimized irregular LDPC code over a binary-input additive white Gaussian noise (BIAWGN) channel. All codes are of block length 10^6 bits and of rate one-half.

Fig. 1.8 offers the comparison [8] of bit error rates achieved by a (3, 6) regular LDPC code, turbo code and the optimized irregular LDPC code in binary-input AWGN channel. All codes are of rate-1/2 and of block lengths 10^6. The Shannon limit and the thresholds of respective LDPC codes using BP decoding algorithm are also shown in the Figure. The ratio E_b/N_0 (in decibels) is one-to-one corresponding to the standard deviation σ if the input binary signal is ± 1 scheme, and also to the raw input bit error probability P_b, which is the bit error rate after the hard demodulation in the receiver. We can easily observe that turbo code outperforms the regular LDPC code in error performance under the same conditions, but it is inferior to the optimized irregular LDPC code. These simulations essentially provide the similar results as the aforementioned case [8] of block length 10^4 bits.

Fig. 1.9 shows the performance [9] of various irregular LDPC and turbo codes with block lengths 10^3, 10^4, 10^5 and 10^6 bits over a BIAWGN channel, where the error performance of turbo code only outperforms that of irregular LDPC code in case of block length 10^3 bits. At length of one million bits, the given irregular LDPC code is less than 0.13dB [9] away from Shannon limit at a bit error probability of 10^{-6}.

In the following subsequent chapters, we will see that LDPC codes can be fully parallelizable in decoding implementations with slightly lower complexity, and allow many different decoding algorithms with a far ranging tradeoff between performance and

complexity. Generally, LDPC codes can be considered serious competitors to the well known turbo codes.

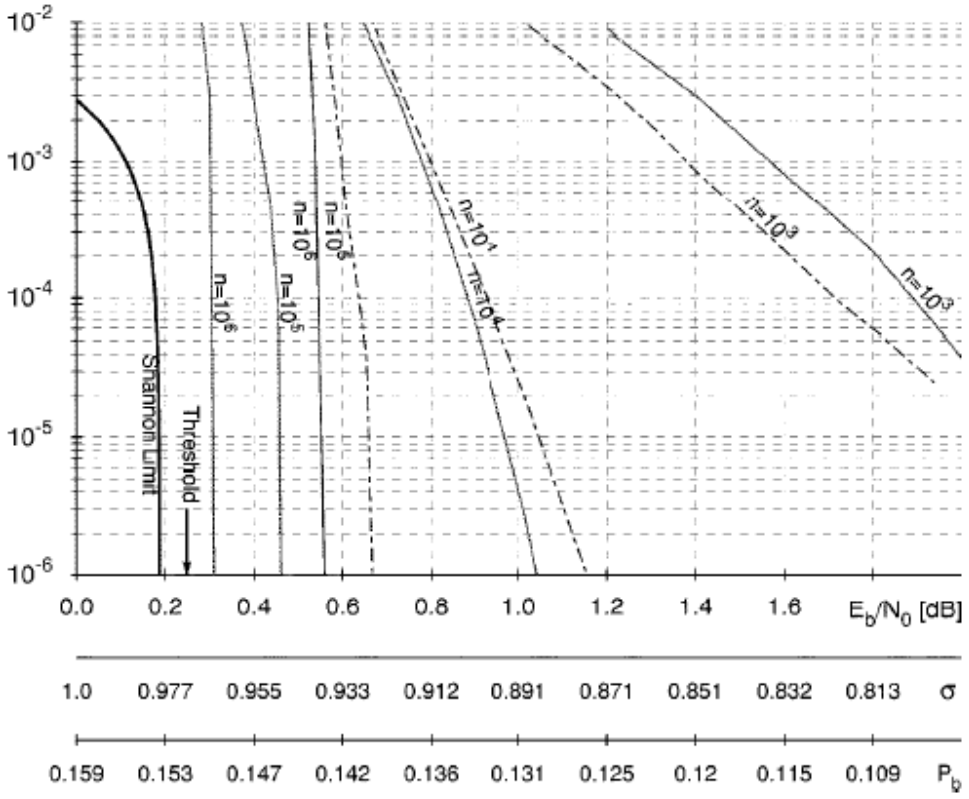

Fig. 1.9 Comparison between bit error rates by turbo codes (dashed curves) and LDPC codes (solid curves) of block lengths n=10^3, 10^4, 10^5 and 10^6 bits with one-half rates over a binary-input additive white Gaussian noise (BIAWGN) channel.

1.4 Outline of the Book

In this chapter, we present the backgrounds of fundamental communication systems and some of basic concepts of information theory and coding theory that we consider useful for the materials covered in the subsequent chapters.

In Chapter 2, some important and necessary notions of directed acyclic graph, directed Markov field and Bayesian belief network are presented through several intuitive but heuristic examples, which are very important to characterize the ordinary linear error-correcting codes, such as block/convolutional codes, and some well known concatenated codes, such as low-density parity-check (LDPC) codes, parallel concatenated (turbo) codes, serially concatenated codes, etc.

In Chapter 3, we introduce the basic subjects of codes on graphs, including the general trellis representation of linear block codes, Markov property and the state space theorem for linear block codes, general linear behavioral realization for linear group codes. These subjects essentially form an intellectual foundation for all known classes of linear error-correcting codes, including the capacity-approaching low-density parity-check codes and turbo codes.

In Chapter 4, the fundamentals of regular low-density parity-check codes of Gallager's type are introduced, including the mathematical definition and encoding process of a regular LDPC code by sparse parity-check matrix. We also analyze the encoding complex of regular LDPC codes and a typical example is given to show the practical encoding process.

In Chapter 5, we discuss a general belief propagation (BP) algorithm for LDPC codes, and its alternative variants by using Tanh rule, Gallager's and Jacobian approaches. In order to analyze the performance of BP algorithm in case of infinite block length and

sufficiently number of decoding iterations, density evolutions and thresholds for various BP algorithms are also presented, including Gallager's decoding algorithms A and B over a binary symmetric channel (BSC), iterative decoder with erasures over BSC, and thresholds for BP algorithms with continuous message alphabets over binary-input AWGN (BIAWGN) and binary-input Laplace (BIL) channels.

In Chapter 6, we present several reduced-complexity BP decoding algorithms for LDPC codes and their related performance in order to fit for the increasing needs in practical implementations, which include BP-based, normalized BP-based, offset BP-based approximations and other memory-efficient decoding algorithms. The density evolution and threshold analysis is also given for unquantized and quantized BP-based approximations, respectively.

In Chapter 7, we extend the conventional regular LDPC codes to irregular LDPC codes. First, the fundamentals of irregular LDPC codes are introduced, including the mathematical definition of an irregular LDPC code and the variable and check nodes degree distributions that can characterize the properties of an irregular LDPC code. Then, the analysis for iterative decoding of irregular LDPC codes is presented in theory, where Gallager's decoding algorithm A over pure binary erasure channel (BEC) and binary symmetric channel (BSC) is particularly emphasized in this chapter.

In Chapter 8, the design of optimized irregular LDPC codes by linear and nonlinear optimizations is presented in both BSC and AWGN channels. First, we discuss the

optimization procedures with respect to left-concentrated and right-concentrated degrees distributions for an irregular LDPC code over a simple BSC. Then, we introduce a useful Gaussian approximation that is widely used in the theoretic analysis of the BP-based decoding algorithms for an LDPC code. Furthermore, we extend our discussions to the optimization of an irregular LDPC code in an AWGN channel by using the Gaussian approximation. Finally, the details of the optimization for check and variable nodes degree distributions are presented, and related some stability issues and asymptotic bit error rate are also concluded.

In Chapter 9, we first discuss the distance distribution and average ensemble distance distribution for arbitrary linear block code. Then we apply this general principle to regular and irregular LDPC codes whose average ensemble distributions are theoretically derived. Finally, the simulation results regarding the average ensemble distance distribution for regular LDPC codes are presented.

In Chapter 10, the cooperative strategy and LDPC codes for relay networks are studied. First, the general idea of cooperative communication as a means of diversity is briefly introduced. Second, we focus on the relay networks that are regarded as a particular case in general cooperative communications. Finally, the performance of LDPC codes in time-division half-duplex one-relay channel with additive Gaussian noise is investigated.

Finally, the appendices and bibliographies of all the chapters are given at the end of this book.