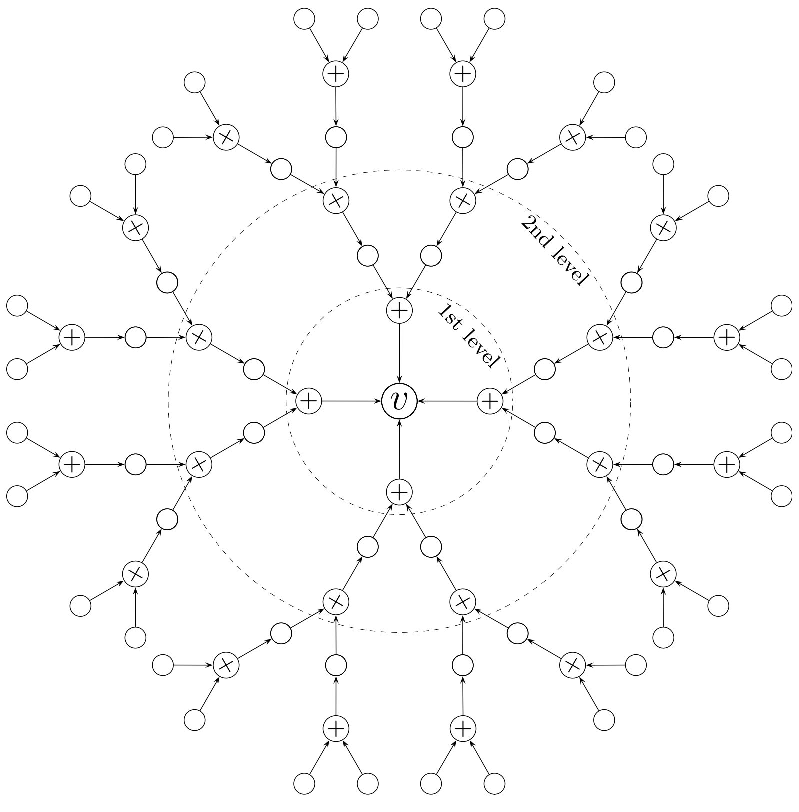
Christian-Albrechts-University of Kiel
Faculty of Engineering

Superposition Mapping & Related Coding Techniques

Tianbin Wo



Kiel 2011

Superposition Mapping
&
Related Coding Techniques

Dissertation

zur Erlangung des akademischen Grades

Doktor der Ingenieurwissenschaften (Dr.-Ing.)

der Technischen Fakultät

der Christian-Albrechts-Universität zu Kiel

vorgelegt von

Tianbin Wo

Kiel 2011

Tag der Einreichung: 21.01.2011
Tag der Disputation: 20.05.2011

Berichterstatter: Prof. Dr.-Ing. Peter Adam Höher
Prof. Dr.-Ing. Martin Bossert
Prof. Li Ping

Preface

This thesis was developed during my time as a research and teaching assistant at the Information and Coding Theory Lab (ICT), Faculty of Engineering, University of Kiel, Germany.

At the time of publishing this thesis, I would first like to thank my advisor Prof. Dr.-Ing. Peter Adam Höher for his inspiring guidance throughout my work in Kiel. The research environment at the ICT is enjoyable and the research team of the ICT is excellent. Additionally, I would like to thank Prof. Höher for enabling me to frequently visit international conferences, which was surely a great help for enriching my academic experience.

I would like to thank Prof. Dr.-Ing. Martin Bossert from Ulm University and Prof. Li Ping from City University of Hong Kong for evaluating this work. Their pertinent suggestions are more than helpful for improving the quality of the thesis and their in-time review is important for setting up a prompt date for the thesis defense.

I would also like to thank my former colleagues of the ICT and the Institute for Circuit and System Theory for many interesting discussions and pleasant activities. I am pretty sure that I will continuously miss my time in Kiel.

Last but not least, I would like to thank my wife Lin Lin, who has accompanied and encouraged me throughout my study in Kiel, and my parents for their enduring support. Without their encouragement and support, this thesis would has been impossible.

[illegible]

Hangzhou, December 2011

Abstract

Since Shannon's landmark paper in 1948, it has been known that the capacity of a Gaussian channel can be achieved if and only if the channel outputs are Gaussian. In the low signal-to-noise ratio (SNR) regime, conventional mapping schemes suffice for approaching the Shannon limit, while in the high SNR regime, these mapping schemes, which produce uniformly distributed symbols, are insufficient to achieve the capacity. To solve this problem, researchers commonly resort to the technique of signal shaping that mends the symbol distribution, which is originally uniform, into a Gaussian-like one.

Superposition mapping (SM) refers to a class of mapping techniques which use linear superposition to load binary digits onto finite-alphabet symbols that are suitable for waveform transmission. Different from conventional mapping schemes, the output symbols of a superposition mapper can easily be made Gaussian-like, which effectively eliminates the necessity of active signal shaping. For this reason, superposition mapping is of great interest for theoretical research as well as for practical implementations. It is an attractive alternative to signal shaping for approaching the channel capacity in the high SNR regime.

This thesis aims to provide a deep insight into the principles of superposition mapping and to derive guidelines for systems adopting it. Particularly, the influence of power allocation to the system performance, both w.r.t. the achievable power efficiency and supportable bandwidth efficiency, is made clear. Considerable effort is spent on finding code structures that are matched to SM. It is shown that currently prevalent code design concepts, which are mostly derived for coded transmission with bijective uniform mapping, do not really fit with superposition mapping, which is often non-bijective and nonuniform. As the main contribution, a novel coding strategy called low-density hybrid-check (LDHC) coding is proposed. LDHC codes are optimal and universally applicable for SM with arbitrary type of power allocation.

Keywords: Digital modulation, signal shaping, Gaussian channel, bit-interleaved coded modulation (BICM), soft-input soft-output (SISO) demapping, channel coding, low-density parity-check (LDPC) code, low-density summation-check (LDSC) code, low-density hybrid-check (LDHC) code, sparse matrix, iterative message passing, belief propagation.

Contents

Chapter 1

Introduction

When considering data transmission over physical channels, a modulator is the interface which maps data sequences onto analog signals that match the characteristics of the channel [1]. By convention, digital modulation consists of three separate steps: mapping binary digits (bits) onto continuous valued symbols, mapping discrete-time symbols onto continuous-time pulses, and loading pulses onto carrier waves. All three steps have significant influence on the system power and bandwidth efficiency. Nevertheless, a discrete-time baseband channel model already comprises the second and third steps, as well as matched filtering and sampling. Therefore, the focus of this thesis will be merely on the first stage, i.e., the mapping from bits to symbols, and the related channel coding techniques. As a matter of fact, channel coding and mapping are the kernel parts of digital communication, from an information theoretical point of view.

1.1 Background

According to information theory, the capacity of a Gaussian channel corresponds to the maximum mutual information between channel inputs and outputs [2]. Given a power constraint, this maximum can be achieved if and only if the channel outputs are Gaussian distributed. Strictly speaking, this is only possible if the channel inputs are Gaussian as well. This issue has been puzzling engineers and researchers for several decades, as it is practically difficult to map bits onto a Gaussian symbol and to separate a Gaussian symbol from Gaussian noise. Nevertheless, at low signal-to-noise ratios (SNRs), binary antipodal symbols often suffice, due to the strong impact of the additive noise on the channel output distribution. This explains why early works focus on the power-limited regime and the proposed coding techniques are exclusively binary.

In the first 50 years since Shannon's landmark paper [3], tremendous effort has been spent on finding good codes for data transmission with binary symbols. As a result, the field of binary coding has already become rather mature, especially after the invention of turbo codes [4] and the rediscovery of low-density parity check (LDPC) codes [5, 6]. Given binary antipodal signalling and sufficiently long block lengths, researchers have been able to approach the Shannon limit of the Gaussian channel as close as just a few thousandths of decibels [7]. Hence, the problem of approaching the Gaussian channel capacity in the power-limited regime has mostly been solved. In contrast, the field of modulation as well as coding for the high-SNR bandwidth-limited regime is still under development. Until the 1970's, the common practical solution for high-rate transmission was nothing more than uncoded higher-order modulation, e.g., phase-shift keying (PSK) and quadrature amplitude modulation (QAM). The success of trellis-coded modulation [8], which is based on the concept of set partitioning, unveiled the importance of channel coding for non-binary data transmission. Also based on set partitioning, multilevel coding [9,10] showed to be an alternative way to improve the performance of higher-order modulation. A breakthrough was finally achieved in the late 1990's when the concept of bit-interleaved coded modulation (BICM) [11,12] was formalized. Placing a bit-level interleaver between the encoder and the mapper and performing iterative demapping and decoding at the receiver, a superior performance was achieved for binary-encoded higher-order modulation systems. Upon this point, it was recognized that available capacity-achieving binary codes are also suitable for non-binary modulation formats, if the structure of BICM is adopted. Hence, there seems to be only one remaining issue for approaching the channel capacity in the high-SNR regime. That is to generate Gaussian-distributed symbols and devise corresponding receiver algorithms.

Mainly due to the desire of easy transmission and detection, most of the traditional mapping schemes produce uniformly distributed symbols. This in the end caused an impassable gap between the Shannon limit and the practically achievable performance, as the capacity of high-SNR Gaussian channels can only be achieved if the inputs are sufficiently Gaussian. To solve this problem, the concept of signal shaping came into being in the late 1980's [13, 14]. Signal shaping is sometimes also called constellation shaping. The basic idea is to construct a high-dimensional uniform constellation which results in low-dimensional nonuniform constituent constellations. Since the high-dimensional constellation is uniform, a one-to-one mapping can be established between a block of bits and a block of symbols. Meanwhile, as the constituent constellation is now nonuniform, often Gaussian-like, a shaping gain can be obtained without any additional effort w.r.t. channel coding. It was shown that signal shaping yields an essential contribution to approach the capacity at high SNRs [13]. In the 1990's, many practical schemes were introduced for an efficient implementation of signal shaping. Among these schemes, the most well-known and successful two are trellis shaping [15] and shell mapping [16–19], both of which are capable of delivering shaping gains about 1 dB. Particularly, shell mapping later became a part of the international telephone-line modem standard ITU recommendation V.34 [20].

With capacity-achieving binary codes, bit-interleaved coded modulation, and signal shaping, it is nowadays a common assumption for researchers that the problems in approaching the Shannon limit of linear Gaussian channels for the high SNR regime have essentially been solved [21]. Nevertheless, this is not completely true.

1.2 Motivation

It is true that available capacity-achieving binary codes can easily be incorporated into a BICM system. However, there will be some tricky issues when one applies signal shaping techniques in a BICM system. Both trellis shaping and shell mapping are block-wise uniform signaling methods, which can attain the ultimate shaping gain [13] only in the limit of infinite dimension, i.e., only in the case of infinite symbol block lengths. Hence, to obtain desirable shaping gain, one needs to take a sufficiently large block length. However, a large block length means a high addressing complexity [22], which also makes the calculation of bit-level soft decisions more difficult. As a matter of fact, given a practical symbol block length, e.g., 16 in ITU V.34, bit-level soft decisions can only be calculated in an approximate way. This certainly incurs performance losses when applying signal shaping in a BICM system, since available high-performance channel codes all demand soft decoding. As an alternative approach, Kschischang *et al.* proposed in [23] a nonuniform signaling scheme that maps simple variable-length prefix codes, particularly Huffman codes, onto a constellation that has been designed according to a Maxwell-Boltzmann distribution [24, 25]. This approach can achieve the ultimate shaping gain in any dimension, i.e., can be performed in a symbol-wise manner, but is unfortunately a data-dependent variable-rate mapping scheme, which brings even more problems for a practical receiver.

To facilitate easy implementation, one needs a fixed-rate symbol-wise mapping scheme that produces Gaussian-like symbols. As mentioned above, trellis shaping, shell mapping, and Huffman decoding all violates this requirement. Nonetheless, a recently evolving technique seems to fulfill this requirement very well. The main idea is to load multiple bits onto a symbol simply via linear superposition. Without loss of generality, we may call the corresponding technique as superposition mapping (SM). For SM, the amount of bits loaded on a symbol is fixed and data-independent, subject to the concrete system design. Besides, superposition mapping generally operates in a symbol-wise manner. According to the central limit theorem, the summation of many i.i.d. variables tends to be Gaussian distributed. Hence, it is very easy to let a superposition mapper to produce Gaussian-like symbols. Given these properties, superposition mapping provides an attractive solution for the applications that demand high bandwidth and power efficiency.

In 1997, Duan *et al.* proposed in [26] a modulation scheme which resembles very much a multiple access system. In this scheme, multi-level coded symbol streams are linearly superimposed before being sent via the channel, and are sequentially separated at the receiver side via successive interference cancellation (SIC). Due to linear superposition, the channel symbols have Gaussian-like nonuniform distributions regardless the fact that the parallel constituent symbol streams are all binary antipodal. For this reason, the authors claimed that large signal constellation and active signal shaping are no longer necessary in such a system. This is the first time that researchers explicitly use linear superposition as a mapping scheme. Although no detailed discussion was presented in [26], this work initiated the research on the topic of superposition mapping.

Ma and Li Ping provided a comprehensive analysis of superposition mapping in [27]. It was shown that single-level coded superposition mapping can achieve similar performance. It was also shown that, with a capacity-achieving binary code, e.g., a Turbo code, superposition mapping is indeed capacity-achieving for linear Gaussian channels. Nevertheless, the bandwidth efficiency of the reported results is limited to 2 bits/symbol per signal dimension. Almost at the same time, Schoeneich and Hoeher proposed in [28] a multi-layer interleave-division multiple access (ML-IDMA) scheme where interleave-division multiplexing (IDM) is done for each user. The kernel part of IDM is in fact a phase-shifted superposition mapper (PSM), which is different from the scheme proposed in [27] by adding a unique phase shift to each antipodal signal before superposition. However, the reported results were limited to 2 bits/symbol per signal dimension as well. It was presumed by the authors of both [27] and [28] that higher bandwidth efficiency can only be supported by means of performance tradeoff via unequal power allocation. Although this presumption is later found to be unprecise [29, 30], it does give an important message about the special property of superposition mapping, that is it is very challenging to support high bandwidth efficiency in case of equal power allocation. Similar issues are also encountered in SM-related modulation techniques, which are non-unique mapping [31], generalized modulation [32], modulation doping [33], etc..

Because of its fixed-rate symbol-wise working style, superposition mapping is undoubtedly an attractive solution for generating near-optimum nonuniform symbols. However, without breaking the tight limit on supportable bandwidth efficiency, the applicable scenarios will be constrained. Certainly, breaking the bandwidth efficiency limit should not be done at the price of degraded power efficiency, or in other words it should not be done at the price of damaging the Gaussian-like symbol distribution. Both [27] and [28] did not provide sufficient theoretical analysis in the concern of the reason of the bandwidth efficiency limit of superposition mapped data transmission. Nevertheless, one thing is clear, that is the commonly applied coding approaches for superposition mapping are not optimal. Since superposition mapping resembles very much a multiple-access system, one may wonder if we can use the techniques available therein to accomplish this task. As a matter of fact, the coding theory for multiple-access systems, or equivalently linear superposition channels, is still far from being mature. Up to the time of writing this thesis, there are indeed no capacity-achieving practical codes for general multiple-access channels with Gaussian noise. Besides, when researchers are designing codes for multiple-access systems, the issue of keeping Gaussian-like symbol distribution is usually not taken into account, as this is rarely an issue in that scenario. For example, the so-called near-optimum uniquely decodable codes [34–36] for multiple-access channels are capacity-achieving only in the case of noiseless channels, as these codes all lead to non-Gaussian-like symbol distributions. Therefore, in order to fully exploit the capacity-achieving potential of SM, suitable channel codes need to be found. This gives the motivation for this thesis work.

1.3 Scope and Aim

This thesis aims to provide a deep insight into the principles of superposition mapping and to develop channel codes that well fit with this type of mapping schemes. It does not try to solve all the open problems, but it does try to solve the most important issues that are critical for practical applications.

The first attempt is to examine the effect of power allocation on the power/bandwidth efficiency of superposition mapping. In many previous works [27, 37–40], it is observed that unequal power allocation can efficiently enhance the supportable bandwidth efficiency of coded superposition mapping, but also causes the symbol distribution non-Gaussian-like. Certainly, this is undesirable from an information theoretical point of view, as this undermines the capacity-achieving potential of superposition mapping. Hence, to find an unequal power allocation strategy that improves the supportable bandwidth efficiency of SM but does not degrade the achievable power efficiency is of particular importance.

Due to linear superposition, superposition mapping is often non-bijective, which raises a problem that ambiguity-free detection is even not possible for a noiseless channel. This fact brings a fundamental challenge for the design of channel codes, as currently available code design techniques are all targeted at bijective mapping schemes. A non-bijective superposition mapper might be treated as a lossy source encoder, or in other words lossy compression happens during the mapping process. To enable ambiguity-free decoding at the receiver side, the channel code has to be designed in a way that the lossy compression (superposition mapping) on the code bits does not incur any information loss on the info bits. This special requirement opens an interesting new research area for channel coding.

For clearness and compactness, the additive white Gaussian noise (AWGN) channel is assumed throughout this thesis, i.e., topics related to fading, intersymbol interference (ISI), and transmission with imperfect channel knowledge are excluded. However, the coding techniques proposed in this thesis can easily be extended to more general applications, such as multi-antenna transmission, multi-carrier transmission, etc..

1.4 Thesis Outline

The contents in the remainder of this thesis develop as follows.

Chapter 2 introduces the discrete-time additive white Gaussian noise channel model and related information theoretical concepts.

Chapter 3 provides a systematic view on superposition mapping. The main attention is given to the effects of power allocation. Given different types of power allocation strategies, the signal properties of superposition mapping are carefully studied, particularly the achievable power efficiency and supportable bandwidth efficiency. A grouped power allocation strategy is proposed to boost the performance potential of SM.

Chapter 4 examines the performance of uncoded SM transmission over the Gaussian channel, given maximum-a-posteriori bit-by-bit detection. Some interesting issues related to source coding are inspected.

Chapter 5 provides a preliminary discussion for coded SM transmission over the Gaussian channel. Several low-complexity SISO demapping algorithms are introduced. The performance of repetition-coded SM and parity-check-coded SM are briefly surveyed.

Chapter 6 makes a thorough investigation for the influence of spreading, scrambling, and interleaving on the performance of repetition-coded SM. A novel concept, called low-density summation-check (LDSC) coding, is proposed to facilitate system optimization.

Chapter 7 discusses the topic of channel coding for superposition mapping in a more general framework. Various theoretical as well as practical issues are investigated. To enable a clever combination of repetition coding and parity check coding for SM, a unified coding concept called low-density hybrid-check (LDHC) coding is proposed.

Chapter 8 summarizes the work and enumerates interesting topics for future research.

Chapter 2

Gaussian Channel

The additive white Gaussian noise (AWGN) channel is probably the most important continuous alphabet channel, and is the prime for algorithm prototyping and system design. Though very simple, it models the fundamental effects of communication in a noisy environment. The AWGN channel is a building block for most practical channel models. Since the channel gain of the AWGN channel is constant, it provides a performance upper bound for many communication channels.

2.1 Channel Model

For modern digital communication systems, the most popular channel model is the discrete-time AWGN channel model described by

$$y = x + z \ , \qquad z \sim \mathcal{N}(0, \sigma_z^2) \ , \tag{2.1}$$

where y is the channel output, x is the channel input, and z is a noise sample drawn i.i.d. from a zero-mean Gaussian distribution with variance σ_z^2 and is assumed to be independent of the channel input. Fig. 2.1 depicts this channel model. In reality, the additive noise may be caused by many different things, such as the circuit noise, the ambient noise, and the quantization noise, etc. However, according to the central limit theorem, the

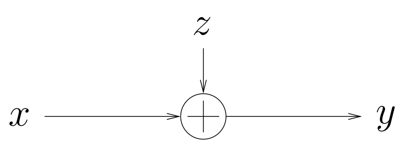


Figure 2.1: Discrete-time additive white Gaussian noise channel.

cumulative effect of a large number of small random effects will be approximately normal distributed. Therefore, the Gaussian assumption of the additive noise is valid in a large number of situations.

To keep consistence with the convention and avoid possible notation confusion, throughout this thesis, we use E_s to represent the average symbol power/energy:

$$E_s \doteq \mathrm{E}\left\{x^2\right\} , \tag{2.2}$$

and N_0 to represent the single-sided noise spectral density in the passband:

$$N_0 \doteq 2\mathrm{E}\left\{z^2\right\} = 2\sigma_z^2 . \tag{2.3}$$

The technical background of this notation convention can be found in [1].

2.2 Mutual Information

Mutual information is a measure of the amount of information that one random variable contains about another variable [2]. For a communication channel, the mutual information between its input and output is the reduction in the uncertainty of the input due to the knowledge of the output. If the channel input distribution is certain, this mutual information indeed gives the maximum rate at which information can pass through this channel with an arbitrarily low error probability.

Given the channel model in (2.1), the mutual information between the channel input and output is

$$I(x;y) = h(y) - h(y|x) = h(y) - h(z) , \tag{2.4}$$

where $h(\cdot)$ denotes the differential entropy of a continuous random variable. By definition, the entropy of the channel output can be calculated as

$$h(y) = -\int p(y)\log p(y)\ \mathrm{d}y . \tag{2.5}$$

$p(y)$ is the probability density function (PDF) of y, which is determined by the distribution of the channel input and the additive noise. For practical systems, the channel input will usually be a finite-alphabet discrete random variable, which leads to

$$p(y) = \sum_{x\in\mathcal{X}} P(x)p(z = y - x) = \sum_{x\in\mathcal{X}} P(x)\frac{1}{\sqrt{2\pi\sigma_z^2}}e^{-\frac{(y-x)^2}{2\sigma_z^2}} , \tag{2.6}$$

where $\mathcal{X}$ is the finite alphabet of x, and $P(x)$ is the probability mass function (PMF) of x. In this scenario, the PDF $p(y)$ is a so-called mixed Gaussian function, which can be evaluated numerically via a computer.

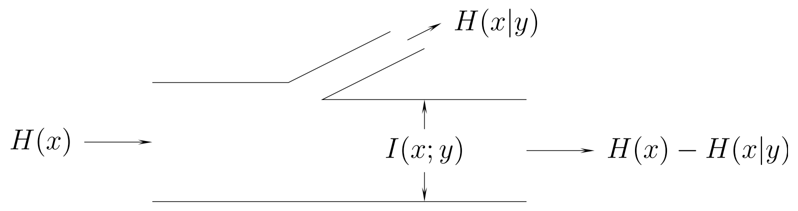


Figure 2.2: Liquid (information) flowing through a pipe (channel).

For Gaussian distributed noise samples z, we have

$$h(z) = \frac{1}{2}\log 2\pi e\sigma_z^2 \ . \tag{2.7}$$

A detailed mathematical derivation of (2.7) is given in Appendix C.2.

Using (2.4) to (2.7), the mutual information given any input distribution can be computed.

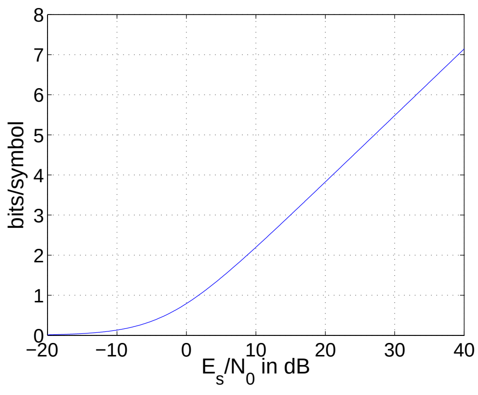
In general we have

$$I(x;y) \leqslant H(x) \ , \tag{2.8}$$

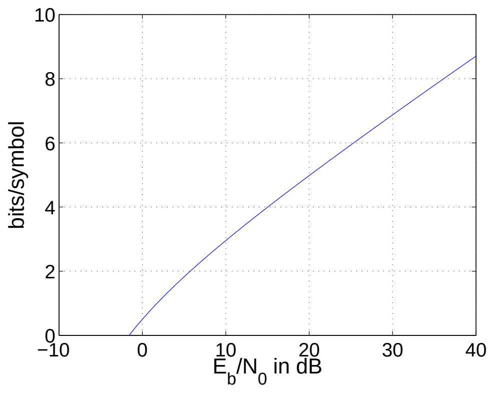
where $H(\cdot)$ denotes the entropy of a discrete random variable. Hence, some information carried in x can not pass through the channel due to the disturbance from the noise. This phenomenon resembles a pipe with one input port, one narrower output port, and one leaking port, as illustrated in Fig. 2.2. The amount of information that is leaked out is given by $H(x|y)$, and the amount of information that finally passes the channel is given by $I(x;y) = H(x) - H(x|y)$. A communication system engineer needs to optimize the distribution of x (without changing the average power) to widen the pipe output port as much as possible so that information can pass through it easily. A natural question would be what is the maximum channel throughput and what is the corresponding distribution for x. This motivates the concept of channel capacity.

2.3 Capacity for Zero Error Probability

The channel capacity is the highest rate in bits per channel use at which information can be transfered with an arbitrarily low error probability [2]. Without any constraint on the channel input, the capacity of an AWGN channel is infinite. This is easy to understand, as one may choose an infinite set of inputs which are arbitrarily far apart from each other, so that they are distinguishable at the output with an arbitrarily small error probability. In practice, however, there will be always certain type of constraints on the channel input, among which the most common limitation is the power/energy constraint. Usually, the average power/energy that can be used to transmit a symbol is limited by physical reasons.



(a) Capacity vs. SNR per symbol.



(b) Capacity vs. SNR per info bit.

Figure 2.3: AWGN channel capacity for a zero error probability.

The channel capacity under input power constraint is given by

$$C = \max_{p(x):\mathrm{E}\{x^2\}=E_s} I(x;y) = \max_{p(x):\mathrm{E}\{x^2\}=E_s} \{h(y) - h(z)\}\ . \tag{2.9}$$

Since $h(z)$ is constant as long as the noise power is certain, the only concern is to maximize $h(y)$ given the power constraint. Noting that the noise z is zero-mean and independent of x, we have

$$\mathrm{E}\{y^2\} = \mathrm{E}\{(x+z)^2\} = \mathrm{E}\{x^2\} + \mathrm{E}\{z^2\} = E_s + N_0/2\ . \tag{2.10}$$

It is known that the Gaussian distribution maximizes the entropy over all distributions with the same variance [2], which brings the following inequality:

$$h(y) \leqslant \frac{1}{2}\log 2\pi e(E_s + N_0/2)\ . \tag{2.11}$$

Hence, the capacity of the AWGN channel is

$$C = \frac{1}{2}\log 2\pi e(E_s + N_0/2) - \frac{1}{2}\log 2\pi e(N_0/2) = \frac{1}{2}\log_2\left(1 + 2\frac{E_s}{N_0}\right)\ \text{bits/symbol}\ , \tag{2.12}$$

which is attained if and only if y is Gaussian, which in turn requires x to be Gaussian. Therefore, to maximize the mutual information for a Gaussian channel, one needs to make the channel input as Gaussian as possible, which gives the motivation for studying superposition mapping.

Fig. 2.3(a) plots the channel capacity versus SNR per symbol. It is often useful to have a plot of channel capacity vs. SNR per info bit. This can be obtained by treating the relationship between channel capacity and SNR in a different way. With a little bit of manipulation on (2.12), we get

$$\frac{E_s}{N_0} = \frac{1}{2}(2^{2C} - 1)\ , \tag{2.13}$$

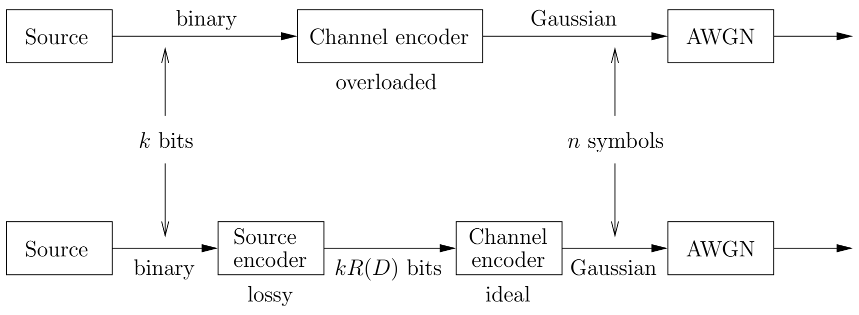


Figure 2.4: A transmission system with a finite error probability.

which states the minimum SNR per symbol required to achieve capacity C in bits/symbol. Noting that $E_b = E_s/C$, we attain the minimum SNR per info bit to achieve the capacity:

$$\frac{E_b}{N_0} = \frac{E_s}{N_0}\Big/C = \frac{1}{2}(2^{2C} - 1)/C \ . \tag{2.14}$$

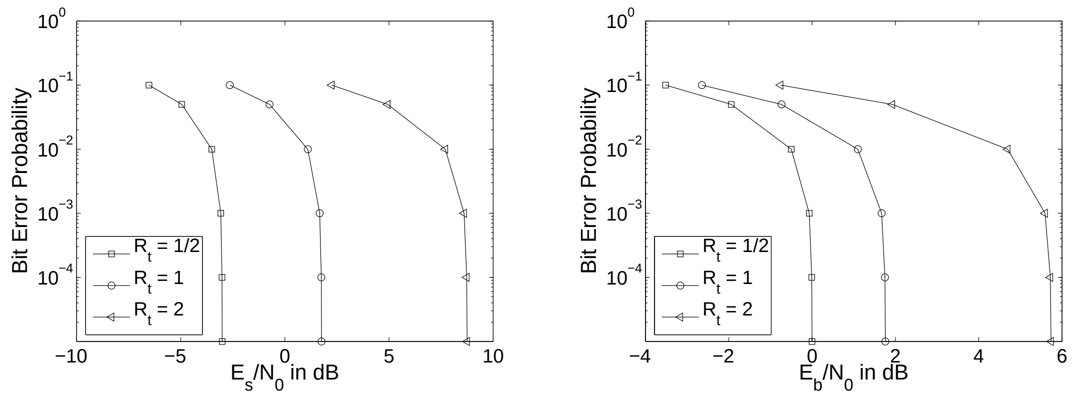
Fig. 2.3(b) gives the plot of channel capacity vs. SNR per info bit.

2.4 Capacity for Finite Error Probability

Fig. 2.3 gives a fundamental guideline for the design of practical communication systems, as it clearly states the maximum achievable information rate for error-free transmission. Equivalently, it states that the channel capacity must be larger than or equal to the transmission rate in order to guarantee a perfect reconstruction of the transmitted info bits. In the AWGN channel scenario, it tells the minimum required SNR for perfect transmission, given a certain transmission rate.

What happens if errors are allowed? In practical applications, the bit error probability is usually required to be lower than a certain threshold but not necessarily to be zero, particularly in voice and video communication. In such cases, one may want to know the minimum required SNR for a finite error probability instead of a zero error probability, which is however not explicitly stated in the theorem of channel capacity. To provide the answer for this question, it is necessary to incorporate the rate distortion theory.

Consider a communication system with a binary source, a channel encoder with Gaussian outputs, and an AWGN channel, as depicted in the upper part of Fig. 2.4. The binary source generates a sequence of k i.i.d. bits with Bernoulli($\frac{1}{2}$) distribution. After channel encoding, n Gaussian symbols are sent over the AWGN channel. Hence, the transmission rate of this system is given by $R_t = k/n$ bits/symbol. The maximum allowed bit error probability is assumed to be P_e. To calculate the minimum required SNR, it is necessary



(a) Capacity vs. SNR per symbol. (b) Capacity vs. SNR per info bit.

Figure 2.5: AWGN channel capacity for finite error probabilities.

to introduce an equivalent transmission system as depicted in the lower part of Fig. 2.4. In this equivalent system, lossy source encoding with Hamming distortion D is applied. The following channel encoder is assumed to be ideal, so that errors are now only introduced by the source encoder. According to the rate distortion theory, the minimum rate of the source code is given by the rate distortion function $R(D)$. Consequently, the minimum information load of each code symbol is given by $kR(D)/n$ bits. To have error-free channel decoding at the receiver side, the following inequality must be fulfilled:

$$C \geqslant kR(D)/n = R(D)R_t \ , \tag{2.15}$$

i.e., the channel capacity must be larger than or equal to the information rate at the channel encoder output. For a Bernoulli($\frac{1}{2}$) source, the rate distortion function with Hamming distortion D is given by

$$R(D) = \begin{cases} 1 - h(D) \ , & 0 \leqslant D \leqslant \frac{1}{2} \ , \\ 0 \ , & D > \frac{1}{2} \ , \end{cases} \tag{2.16}$$

and the bit error probability is exactly the same as the Hamming distortion. The above statements give the following inequality:

$$\frac{1}{2}\log_2(1 + 2\frac{E_s}{N_0}) \geqslant k(1 - h(P_e))/n \quad \Longrightarrow \quad \frac{E_s}{N_0} \geqslant \frac{1}{2}\left(2^{2(1-h(P_e))R_t} - 1\right) \ , \tag{2.17}$$

which tells the minimum required SNR, given a certain bit error probability and a certain transmission rate. If the equality in (2.17) is attained, the system is said to be optimal. Fig. 2.5 gives an example for several transmission rates. These curves serve as tight bounds for the achievable power efficiency of practical systems, i.e., the BER curve of a practical system will always be on the right side of the corresponding bound.

Chapter 3

Superposition Mapping (SM)

According to Shannon's information theory, the capacity of a Gaussian channel can be achieved if and only if the channel outputs are Gaussian distributed. In the low SNR regime, the strong impact from the additive Gaussian noise makes the channel outputs Gaussian even if the inputs are far from Gaussian. For this reason, conventional uniform mapping schemes are already capacity-achieving at low SNRs. Nonetheless, low SNR means low capacity, and low capacity means low bandwidth efficiency. To attain high bandwidth efficiency, one needs to operate in the high SNR regime. For high SNRs, the channel outputs can only be Gaussian if the channel inputs have a distribution with a Gaussian-like envelope. In this scenario, conventional uniform mapping schemes are no longer suitable. Superposition mapping (SM), as a newly evolving modulation technique, seems to fulfill this requirement very well. The characteristic feature of this technique is that the conversion from binary digits to symbols is done by a certain form of linear superposition instead of bijective (one-to-one) mapping. Due to linear superposition, the symbol distribution can be as Gaussian as desired. On the other hand, superimposed components interfere with each other, and the resulting relationship between bit tuples and symbols is often non-bijective. As a result, SM shows many different features w.r.t. conventional uniform mapping schemes. In this chapter, we perform a systematic study on SM and try to understand SM from an information theoretical point of view. Particular focus is on the effects of power allocation. It will be shown that equal power allocation (EPA) provides an excellent power efficiency but comes with a limited bandwidth efficiency for a reasonable superposition order. In contrast, unequal power allocation (UPA) provides a high bandwidth efficiency but a degraded power efficiency. To overcome the drawbacks of EPA and UPA, a novel scheme called grouped power allocation (GPA) is proposed. SM-GPA delivers a significantly improved bandwidth efficiency w.r.t. SM-EPA but does not degrade the achievable power efficiency like SM-UPA does.

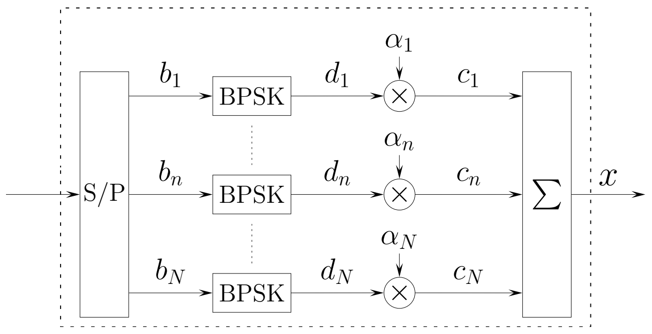


Figure 3.1: General structure of superposition mapping.

3.1 General Description

Linear superposition is a natural phenomenon in the real world, which partly explains the prevalence of the normal distribution, as the cumulative effect of many random effects will be approximately normal distributed. According to the central limit theorem, the mean of a sufficiently large number of i.i.d. random variables will be approximately Gaussian distributed. Equivalently, if one superimpose many i.i.d. variables and fix the total power of these variables, the resulting summation will be approximately Gaussian. Hence, it is a natural idea to use linear superposition to create Gaussian-like symbols.

Fig. 3.1 shows a general structure of superposition mapping (SM) with binary antipodal component symbols. After serial-to-parallel conversion, N code bits are first converted into binary antipodal symbols via binary phase shift keying (BPSK). Then, a certain amplitude is allocated to each of these symbols. Afterwards, these component symbols are linearly superimposed to create a finite-alphabet output symbol. This procedure might be described by the following equation:

$$x = \sum_{n=1}^{N} c_n = \sum_{n=1}^{N} \alpha_n d_n = \sum_{n=1}^{N} \alpha_n (1 - 2b_n) \ , \quad \alpha_n \in \mathbb{R} \ , \quad b_n \in \{0, 1\} \ . \tag{3.1}$$

In the terminology of mapping, the SM mapping rule can be defined as

$$\phi_{\text{SM}}(\mathbf{b}) = \sum_{n=1}^{N} \alpha_n (1 - 2b_n) \ , \quad \mathbf{b} \in \mathbb{F}_2^N \ , \tag{3.2}$$

i.e., a binary N-tuple is mapped onto a symbol with a finite alphabet. The amplitude coefficients α_n, $1 \leqslant n \leqslant N$, should be chosen in a way that $E_{\phi_{\text{SM}}} = E_s$ is fulfilled. That is

$$\mathrm{E}\left\{x^2\right\} = \sum_{n=1}^{N} \mathrm{E}\left\{d_n^2 \alpha_n^2\right\} = \sum_{n=1}^{N} \alpha_n^2 \overset{!}{=} E_s \ , \tag{3.3}$$

assuming that all chips are mutually independent. To fairly compare with other mapping schemes, one should normally take $E_s = 1$.

Certainly, instead of taking $\alpha_n \in \mathbb{R}$, one may also take $\alpha_n \in \mathbb{C}$. By using complex-valued coefficients α_n, one can control not only the power of each chip but also the phase. This brings another degree of freedom for the design of superposition mapping. In [29], it is shown that allocating unique phases to superimposed chips increases the symbol cardinality and consequently improves the supportable bandwidth efficiency of SM given conventional channel coding schemes, e.g., convolutional codes. Nevertheless, allocating unique phases to superimposed chips also makes the SM constellation points non-equispaced and geometrically irregular in the complex signal space. This degrades the quality of SM symbol distribution and consequently deteriorates the performance of SM transmission over the Gaussian channels. Furthermore, unequispaced constellation points may cause a troublesome issue for the frontend circuit of a transmission system. As a matter of fact, the preference of phase-shifted superposition mapping (PSM) in many previous works [29, 38, 41–43] is mainly for the reason that the supportable bandwidth efficiency of SM without phase allocation is very limited, which is actually due to the unavailability of suitable channel codes. However, with the coding approaches provided in Chapter 6 and Chapter 7, this issue will be no longer existing. Given these considerations, we exclude the topic of phase allocation in this thesis and focus our discussion on real-valued superposition mapping, which might be considered as one signal dimension (either in-phase or quadrature) of complex-valued superposition mapping.

Hereafter, N will be called the bit load of the superposition mapper, and the binary antipodal component symbols c_n will be called chips. Obeying the convention of BPSK, the following correspondence holds:

$$\begin{array}{lllll} b_n = 0 & \leftrightarrow & d_n = +1 & \leftrightarrow & c_n = +\alpha_n \\ b_n = 1 & \leftrightarrow & d_n = -1 & \leftrightarrow & c_n = -\alpha_n \end{array} . \tag{3.4}$$

Note that using binary antipodal chips does not cause any limit on the overall bandwidth efficiency. To support different data rates, it is only necessary to adjust the bit load N of SM. Certainly, one may use higher-order modulation formats for the chips, but this merely increases the mapping and demapping complexity without bringing any practical or theoretical benefits. Hence, throughout this thesis, chips will be always binary antipodal.

Different from conventional mapping schemes, the cardinality of an SM symbol is not necessarily 2^N. Let $\mathcal{X}$ denote the alphabet of x. In general, the situation is

$$|\mathcal{X}| \leqslant 2^N , \tag{3.5}$$

i.e., the mapping rule is often non-bijective and subsequently the symbol distribution is often nonuniform. A nonuniform symbol distribution is desirable from an information theoretical point of view. However, a non-bijective mapping rule brings a fundamental challenge for the design of suitable channel codes, as discussed in later chapters.

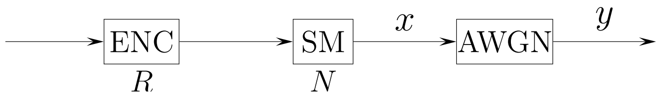


Figure 3.2: Coded SM transmission over Gaussian channel.

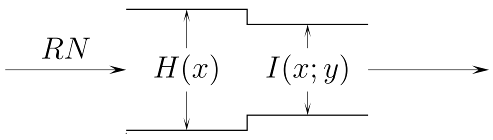


Figure 3.3: A "pipe" interpretation of the information path.

3.2 An Information Theoretical View

Since a superposition mapper is often non-bijective, it takes some special carefulness when using it. In this section, superposition mapping is reexamined from an information theoretic point of view. The discussion herein serves as a guideline for the usage of SM.

When a superposition mapper is non-bijective, one has

$$H(x) < N \text{ bits }, \tag{3.6}$$

where $H(\cdot)$ denotes discrete entropy. By the definition of entropy, $H(x)$ is the average amount of information that an SM symbol can carry. In case of uncoded transmission, the total amount of information carried by N input bits is exactly N bit, assuming an ideal source encoder. If (3.6) holds, one is eventually loading more information than an SM symbol can carry. Consequently, error-free detection will not be possible. Hence, non-bijective superposition mapping is not suitable for uncoded transmission. To enable error-free detection, channel coding is necessary to reduce the information rate of the input bits of a superposition mapper.

Consider coded SM transmission over the AWGN channel, depicted in Fig. 3.2. The channel coding rate is given by R and the bit load of the superposition mapper is given by N. Without loss of generality, one may interpret the above system as a "pipe" with two sections, and the information passing through as certain type of "liquid", as illustrated in Fig. 3.3. The diameter of the first pipe section is given by the information-carrying capability of SM symbols, i.e., the symbol entropy $H(x)$. The diameter of the second pipe section is given by the mutual information between the channel input and output, which is denoted by $I(x; y)$. Due to the disturbance from the additive noise, $I(x; y) \leqslant H(x)$ holds in general. As commonly known, the maximum throughput of a pipe is determined by the diameter of the narrowest section. Similarly, the maximum information rate of the above system is given by $I(x; y)$, which is frequently called channel capacity with constrained input. To enhance the effective system throughput, one should try to maximize $I(x;y)$ instead of $H(x)$.

Each SM symbol carries N code bits, while each code bit carries R bits information. Therefore, the amount of information that one loads onto an SM symbol is RN bits. To have error-free detection in case of noiseless transmission, the following condition has to be fulfilled:

$$RN \leqslant H(x) \quad \Rightarrow \quad R \leqslant H(x)/N \ . \tag{3.7}$$

This gives a tight constraint for systems employing superposition mapping. Note that for a non-bijective superposition mapper, one has $H(x) < N$. In this concern, SM is inferior to conventional uniform mapping schemes for which $H(x) = N$ holds. Nevertheless, if the channel is noisy, an even tighter condition applies:

$$RN \leqslant I(x;y) \ . \tag{3.8}$$

In this scenario, SM with a properly designed power allocation strategy outperforms uniform mapping schemes in the sense of higher mutual information given a certain power constraint, which is the main topic of discussion in the following sections.

3.3 Equal Power Allocation (EPA)

In principle, one can take infinite possibilities in choosing the power allocation scheme for superposition mapping. Nevertheless, the simplest yet the most essential scheme is the equal power allocation (EPA) scheme, which assigns all chips with the same amplitude. This simple mechanism leads to an elegant mathematical description and a well-structured symbol distribution. As a matter of fact, the EPA scheme lies in the heart of the capacity-achieving potential as well as the working philosophy of superposition mapping. This section gives an extensive and in-depth study on SM-EPA and try to reveal its strength as well as its limits.

3.3.1 Overview

For equal power allocation, the chip amplitudes are all identical. That is

$$\alpha_i = \alpha_j \quad \forall \quad 1 \leqslant i,j \leqslant N \ . \tag{3.9}$$

Consequently, one may use a single $\alpha \in \mathbb{R}$ to denote the chip amplitudes. The mapping rule of SM-EPA can be written as

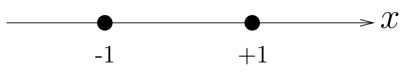
$$\phi_{\text{SM-EPA}}(\mathbf{b}) = \alpha \sum_{n=1}^{N} (1 - 2b_n) \ , \quad \mathbf{b} \in \mathbb{F}_2^N \ , \tag{3.10}$$

where α is chosen to satisfy $E_{\phi_{\text{SM-EPA}}} = E_s$, or equivalently $\alpha^2 N = E_s$. Hence, to have $E_s = 1$, the amplitude coefficient should be chosen as $\alpha = \sqrt{1/N}$. Nevertheless, in order to achieve simple expressions, we often take $\alpha = 1$, i.e., $\mathrm{E}\{x^2\} = N$, for the purpose of illustration, while $\alpha = \sqrt{1/N}$ for the purpose of performance assessment.

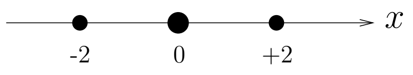
In the simplest case, $N = 1$, SM-EPA is merely a BPSK mapper, with a symbol alphabet consisting of two distinct values, depicted in Fig. 3.4(a). In this scenario, SM-EPA is bijective and all constellation points are equiprobable. Increasing the bit load from 1 to 2, the property of the mapper shows a significant change. Depicted in Fig. 3.4(b), the cardinality of the symbol alphabet at $N = 2$ is 3 instead of $2^N = 4$, and the probability of $x = 0$ is twice that of $x = \pm 2$. The constellation points are equispaced but no longer equiprobable, which gives a big difference to conventional mapping schemes.

As a matter of fact, for $N \geqslant 2$, SM-EPA will be always non-bijective, i.e., having a many-to-one correspondence between the bit tuples and output symbols. For example, given $N = 4$, the mapping rule will be given by Tab. 3.1, which is indeed very interesting. It can be seen that bit tuples with the same amount of 0's or 1's are always mapped onto the same symbol value, or in other words the permutation of 0's and 1's does not make any difference in the mapping result. Particularly, the bit tuples corresponding to $x = 0$ are typical sequences of Bernoulli($\frac{1}{2}$) distribution. Given independently and uniformly distributed (i.u.d.) input bits, this mapping mechanism resembles very much source coding. In source coding, the most frequently occurring message is encoded with the shortest code word, such that the average code word length is minimized, while in superposition mapping, the most frequently occurring set of bit tuples is mapped onto the least-energy symbol value, which minimizes the average energy of symbol transmission. Nevertheless, due to a many-to-one correspondence between bit tuples and symbols, information loss happens during this mapping procedure. Hence, SM-EPA works as if a lossy source encoder.

Back for 20 years, researchers might consider such a mapper as catastrophic, as ambiguity-free detection is not possible even over a noiseless channel. Nevertheless, with modern results on coded modulation, one will find that this is a mapper that Shannon would applaud, since it fulfills the requirement for approaching Gaussian channel capacity.



(a) $N = 1$.



(b) $N = 2$.

Figure 3.4: Symbol alphabets of SM-EPA, $\alpha = 1$.

b_1, b_2, b_3, b_4	c_1, c_2, c_3, c_4	$x = \sum_{n=1}^{4} c_n$
1 1 1 1	$-1\ -1\ -1\ -1$	-4
0 1 1 1	$+1\ -1\ -1\ -1$	-2
1 0 1 1	$-1\ +1\ -1\ -1$	
1 1 0 1	$-1\ -1\ +1\ -1$	
1 1 1 0	$-1\ -1\ -1\ +1$	
0 0 1 1	$+1\ +1\ -1\ -1$	0
0 1 0 1	$+1\ -1\ +1\ -1$	
0 1 1 0	$+1\ -1\ -1\ +1$	
1 0 0 1	$-1\ +1\ +1\ -1$	
1 0 1 0	$-1\ +1\ -1\ +1$	
1 1 0 0	$-1\ -1\ +1\ +1$	
0 0 0 1	$+1\ +1\ +1\ -1$	$+2$
0 0 1 0	$+1\ +1\ -1\ +1$	
0 1 0 0	$+1\ -1\ +1\ +1$	
1 0 0 0	$-1\ +1\ +1\ +1$	
0 0 0 0	$+1\ +1\ +1\ +1$	$+4$

Table 3.1: Mapping rule of SM-EPA, $N = 4$, $\alpha = 1$.

3.3.2 Symbol Distribution

Given equal power allocation (EPA), superimposed chips all have identical distribution. Consequently, an SM-EPA symbol, which is the summation of N i.i.d. chips, will have a Gaussian-like distribution as long as N is sufficiently large. To offer a solid understanding, this section provides a detailed survey of the symbol distribution of SM-EPA.

Revisiting (3.1), it is easy to find that an SM-EPA symbol is formed as

$$x = \alpha \sum_{n=1}^{N} d_n \ , \quad d_n \in \{\pm 1\} \ . \tag{3.11}$$

As a matter of fact, the resulting symbol alphabet $\mathcal{X}$ grows (w.r.t. N) in an interesting way, demonstrated in Tab. 3.2. There is a simple relationship between the bit load and the symbol cardinality:

$$|\mathcal{X}| = N + 1 \ . \tag{3.12}$$

Hence, the symbol cardinality increases linearly with the bit load, instead of exponentially. This property is very helpful in reducing the demapping complexity, as explained in Sec. 5.2.2. Besides, it can be seen from Tab. 3.2 that the symbol alphabet of SM-EPA

N	$\mathcal{X}$	$\lvert\mathcal{X}\rvert$
1	$-1, +1$	2
2	$-2, 0, +2$	3
3	$-3, -1, +1, +3$	4
4	$-4, -2, 0, +2, +4$	5
5	$-5, -3, -1, +1, +3, +5$	6

Table 3.2: Symbol alphabets of SM-EPA, $\alpha = 1$.

can be written as

$$\mathcal{X} = \{-\alpha N, -\alpha(N-2), \ldots, \alpha(N-2), \alpha N\} \ . \tag{3.13}$$

Let χ_i denote these possible symbol values in an ascending order, one attains

$$\chi_i = -\alpha(N - 2i) \ , \quad 0 \leqslant i \leqslant N \ , \tag{3.14}$$

where the subscript i actually also gives the number of 0's in the corresponding bit tuples, cf. Tab. 3.1. Given that the input bits are i.i.d., an SM-EPA symbol conforms to a binomial distribution $\mathcal{B}(N, p)$:

$$P(x = \chi_i) = \binom{N}{i}(1-p)^i p^{N-i} \ , \tag{3.15}$$

where p is the probability of each bit to be 1 and

$$\binom{N}{i} = \frac{N!}{i!(N-i)!} \tag{3.16}$$

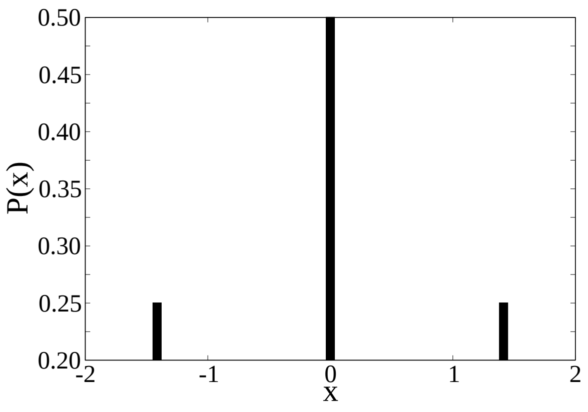
is the binomial coefficient. In the field of digital communication, it is a common practice to make code bits uniformly distributed, e.g., via scrambling. Whenever this is the case, one has $p = 1/2$ and

$$P(x = \chi_i) = \binom{N}{i}(1/2)^i(1/2)^{N-i} = \binom{N}{i}/2^N \ . \tag{3.17}$$

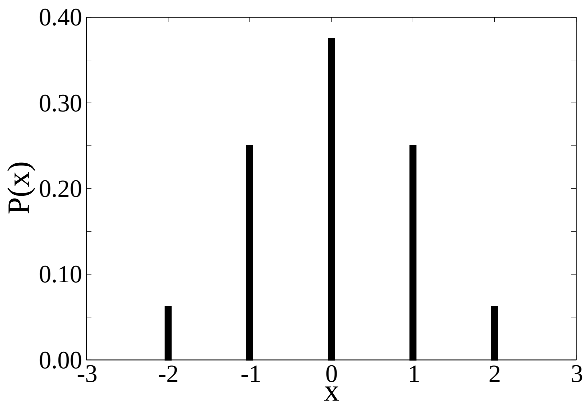
In this case, the skew of the distribution is zero, i.e., the distribution is symmetric w.r.t. the mean of x. If N is large enough, an excellent approximation to $P(x)$ is given by a normal distribution with a suitable continuity correction [44].

Fig. 3.5 plots the symbol distribution of SM-EPA for various bit loads. As N increases, the envelope of $P(x)$ first becomes triangular-like and then Gaussian-like. Note that, all constellation points are equispaced, and the gap between neighboring points is given by 2α. For a fixed energy per symbol, the chip amplitude α is inversely proportional to the square root of the bit load:

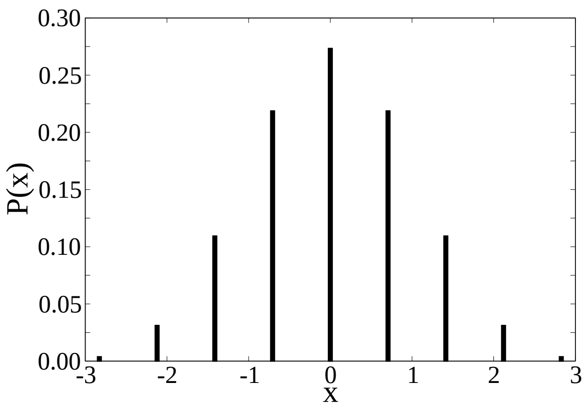
$$\alpha \propto \frac{1}{\sqrt{N}} \ , \tag{3.18}$$



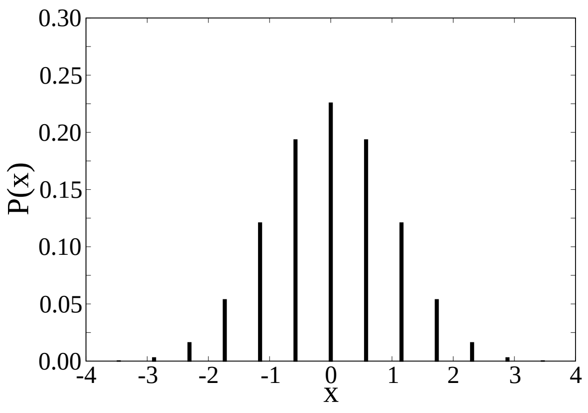
(a) $N = 2$.



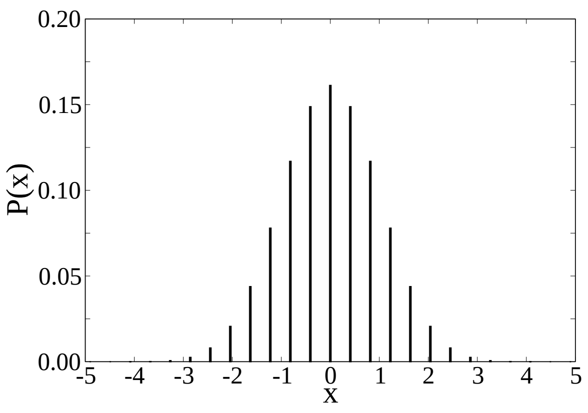
(b) $N = 4$.



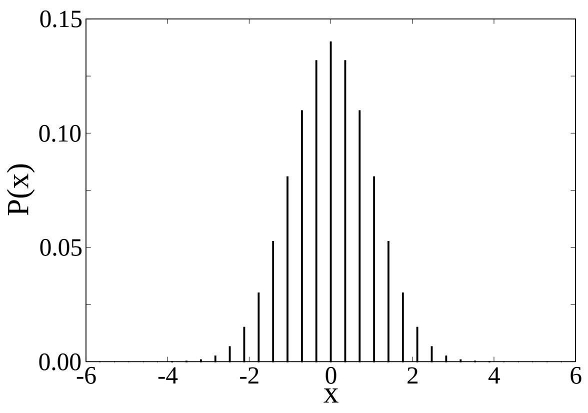
(c) $N = 8$.



(d) $N = 12$.



(e) $N = 24$.



(f) $N = 32$.

Figure 3.5: Symbol distributions of SM-EPA, $E_s = 1$, $\alpha = \sqrt{1/N}$.

N	$\lvert\mathcal{X}\rvert$	$H(x)$	$H(x)/N$
1	2	1.0000 bits	1.0000
2	3	1.5000 bits	0.7500
4	5	2.0306 bits	0.5077
8	9	2.5442 bits	0.3180
12	13	2.8385 bits	0.2365
16	17	3.0465 bits	0.1904
24	25	3.3393 bits	0.1391
32	33	3.5470 bits	0.1108
64	65	4.0471 bits	0.0632

Table 3.3: Symbol entropies and compression rates of SM-EPA.

which leads to

$$\alpha \to 0 \quad \text{for } N \to \infty\,. \tag{3.19}$$

Hence, as N tends to be infinity, $P(x)$ asymptotically approaches a continuous Gaussian distribution, which is very desirable from an information theoretical point of view.

3.3.3 Symbol Entropy

Due to the nonuniform distribution, the symbol entropy of SM-EPA will generally be less than the bit load. Without loss of generality, one may call $H(x)/N$ the compression rate of a superposition mapper, as N independent code bits are compressed into a symbol carrying $H(x)$ bits of information. As introduced in Section 3.2, this compression rate tightly upper bounds the coding rate for error-free transmission over noiseless channels.

Follow the definition of entropy [2] and revisit (3.17), we obtain

$$H(x) = -\sum_{x\in\mathcal{X}} P(x)\log P(x) \;=\; -\sum_{i=0}^{N} \frac{\binom{N}{i}}{2^N}\log_2\frac{\binom{N}{i}}{2^N} \quad \text{bits}\,. \tag{3.20}$$

Tab. 3.3 lists the symbol entropies as well as the compression rates for some N. It can be seen that the symbol entropy grows slower and slower as the bit load increases. As a consequence, the compression rate decreases with N, which also means that the maximum permissible coding rate decreases with N (since $R \leqslant H(x)/N$ according to (3.7)). The reason of this phenomenon is that when N goes up the size of the typical sequence set for each symbol value gets larger, which in turn introduces more information loss or in other words more compression.

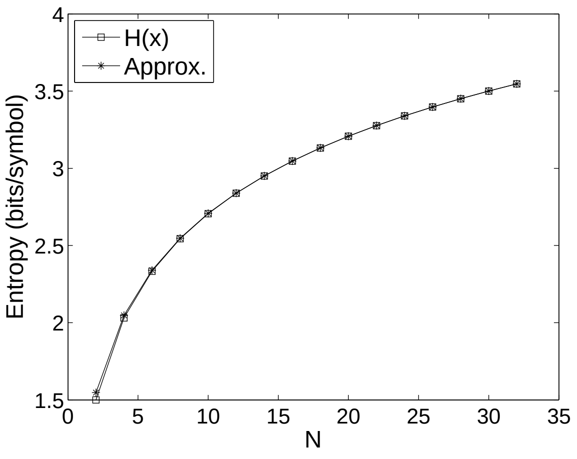


Figure 3.6: Symbol entropy of SM-EPA vs. bit load N.

Equation (3.20) is mathematically strict but does not give any intuition on the relationship between $H(x)$ and N. According to probability theory [45, 46], a binomial distribution $\mathcal{B}(N, \frac{1}{2})$ can be well approximated by a Gaussian distribution $p(u)$ with the same mean and variance for $N \geqslant 5$, given that a proper continuity correction [47] is done. It is clear that x is zero-mean and its variance is given by

$$\mathrm{E}\left\{x^2\right\} = \sum_{n=1}^{N} \alpha^2 \mathrm{E}\left\{d_n^2\right\} = \sum_{n=1}^{N} \alpha^2 = \alpha^2 N \; . \tag{3.21}$$

Note that the distance between neighboring constellation points is always 2α. Then for large N we have the following approximation

$$P(x) \approx \int_{x-\alpha}^{x+\alpha} p(u)\ \mathrm{d}u = \int_{x-\alpha}^{x+\alpha} \frac{1}{\sqrt{2\pi\sigma_u^2}}\ e^{-u^2/(2\sigma_u^2)}\ \mathrm{d}u \tag{3.22}$$

with

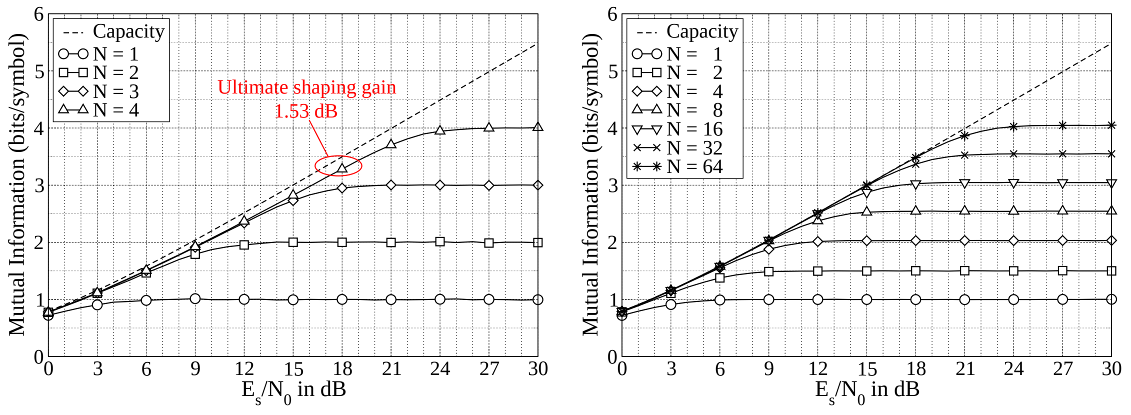
$$\sigma_u^2 = \alpha^2 N \; . \tag{3.23}$$

Without loss of generality, x might be considered as a linear quantization of a Gaussian variable u. Correspondingly, the quantization bin size is given by $\Delta = 2\alpha$. Using formula (C.10) provided in Appendix C.3, we have

$$H(x) \approx \frac{1}{2}\log(2\pi e\sigma_u^2/\Delta^2) = \frac{1}{2}\log(2\pi e\alpha^2 N/(4\alpha^2)) = \frac{1}{2}\log(\frac{\pi}{2}eN) \; , \tag{3.24}$$

i.e., $H(x) \approx h(\mathcal{N}(0, N/4))$. As a matter of fact, $\frac{1}{2}\log(\frac{\pi}{2}eN)$ already gives a very good approximation for $H(x)$ even when N is not so large, depicted in Fig. 3.6, where "Approx." stands for the approximation under concern[1].

[1]This topic already attracts interest in the 1970's [35]. Later on, Hughes *et al.* proposed in [36] a simplified mathematical derivation by utilizing the relationship between the entropy of a continuous variable and the discrete entropy of its quantization [2]. An interesting study on a precise asymptotic approximation of the entropy of binomial distribution can be found in [48].



(a) Uniform ASK mapping. (b) SM-EPA.

Figure 3.7: Mutual information over the AWGN channel.

Given the above derivation, it is now clear that the symbol entropy $H(x)$ of SM-EPA is approximately logarithmic w.r.t. the bit load N. This explains the decrease of compression rate $H(x)/N$ when increasing N. As listed in Tab. 3.3, by choosing $N = 32$ the amount of information that one SM-EPA symbol can carry is merely 3.5470 bits, which indicates that SM-EPA is inefficient in supporting very high bandwidth efficiencies.

3.3.4 Mutual Information

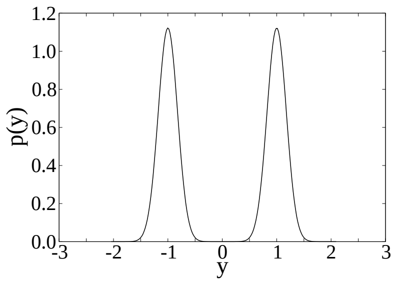
The maximum throughput of a channel is given by the mutual information (MI) between its input and output. For the AWGN channel we have

$$I(x;y) = h(y) - h(y|x) = h(y) - h(z) \ .$$

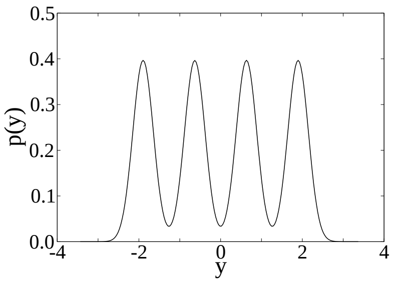
Given a power constraint, $h(y)$ is maximized when y conforms to a Gaussian distribution. Hence, whether a mapping scheme is capacity-achieving or not is determined by whether the corresponding channel output distribution is Gaussian or not.

It is well-known that conventional uniform ASK mapping is not capacity-achieving at high SNRs. As illustrated in Fig. 3.7(a), there is a gap of about 1.53 dB between the MI curves of ASK and the channel capacity curve. In the terminology of signal shaping, this gap is often referred to as the ultimate shaping gain, as this is the maximum possible gain that signal shaping can yield w.r.t. uniform ASK mapping. Fig. 3.8 gives a good explanation for the MI performance of ASK. One sees that the channel output distribution does not become Gaussian for uniform ASK with large bit loads.

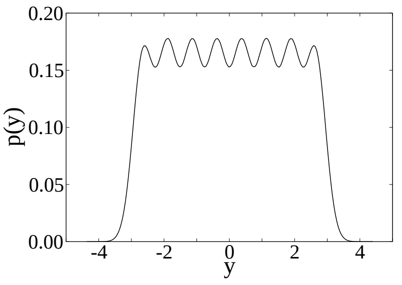
As introduced in Section 3.3.2, the symbol distribution of SM-EPA tends to have a Gaussian-like envelope as the bit load increases. Naturally, one expects a better performance from SM-EPA than that from uniform ASK. Demonstrated by Fig. 3.7(b),



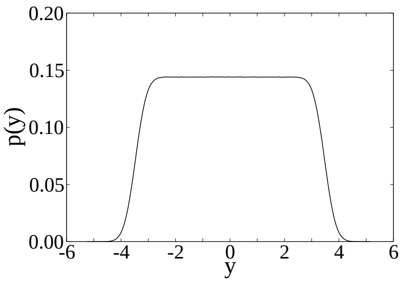
(a) $N = 1$.



(b) $N = 2$.

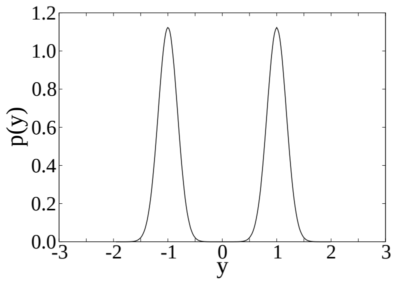


(c) $N = 3$.

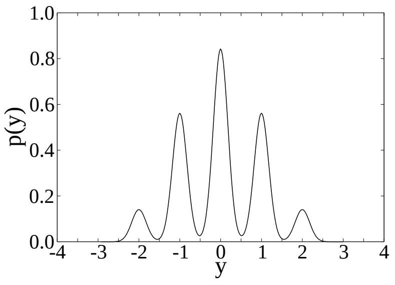


(d) $N = 4$.

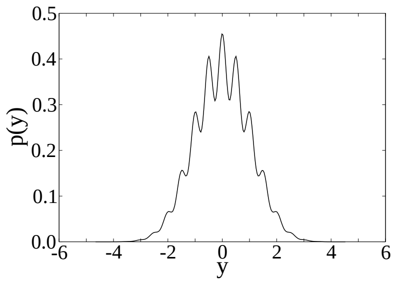
Figure 3.8: AWGN channel output distribution for uniform ASK, $E_s/N_0 = 12$ dB.



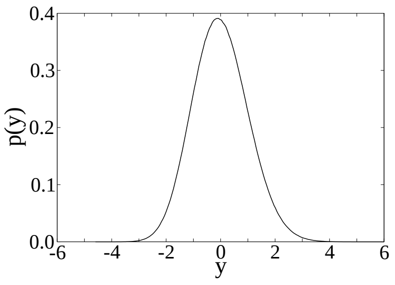
(a) $N = 1$.



(b) $N = 4$.



(c) $N = 16$.



(d) $N = 64$.

Figure 3.9: AWGN channel output distribution for SM-EPA, $E_s/N_0 = 12$ dB.

SM-EPA is indeed capacity-achieving. For example, given $N = 64$, the MI curve of SM-EPA sticks with the capacity curve till $E_s/N_0 \approx 18$ dB. It is not difficult to imagine that SM-EPA can be capacity-achieving at arbitrarily large SNRs as long as the bit load is large enough. Fig. 3.9 illustrates the channel output distribution for SM-EPA. What we see is that for SM-EPA with large bit loads the distribution of the continuous additive noise smoothes out the gaps between the discrete symbol values of SM-EPA.

3.4 Unequal Power Allocation (UPA)

In the previous section, superposition mapping with equal power allocation has been thoroughly studied. It has been shown that the symbol distribution of SM-EPA is Gaussian-like and consequently is capacity-achieving for Gaussian channels. In this section, another important class of power allocation strategy will be investigated, namely unequal power allocation (UPA). This type of power allocation strategy can not really demonstrate the main merits of superposition mapping but deserves interest for academic study. It will be shown that SM-UPA shows a significantly different behaviour compared to SM-EPA, both w.r.t. the symbol property and the achievable power/bandwidth efficiency.

3.4.1 The Exponential Law

Though there are virtually unlimited possibilities for unequal power allocation, the most meaningful choice is the exponential law both for practice and theoretical study. Mathematically, an exponential power allocation strategy can be described as

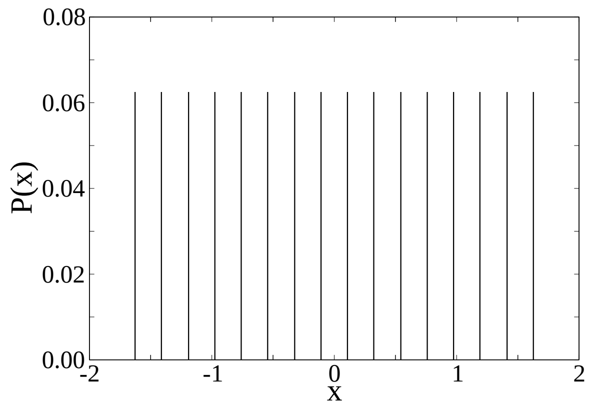
$$x = \sum_{n=1}^{N} c_n = \sum_{n=1}^{N} \alpha_n d_n \ , \qquad d_n \in \{\pm 1\} \ , \tag{3.25}$$

with

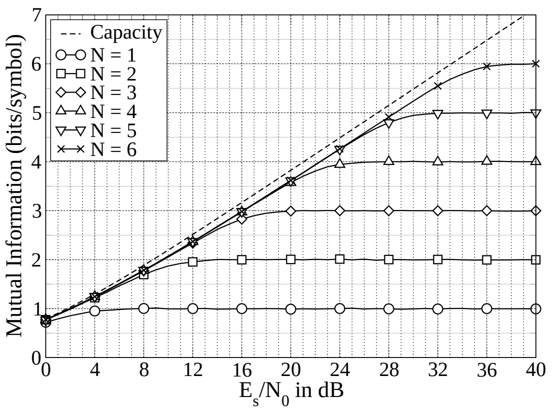
$$\alpha_n = a \cdot \rho^{n-1} \ , \quad 0 < \rho < 1 \ , \tag{3.26}$$

where ρ is the exponential base, and the value of a should be such that $\mathrm{E}\{x^2\} = E_s$ is fulfilled. Note that the power of the $(n+1)$-th chip is always ρ^2 of that of the n-th chip.

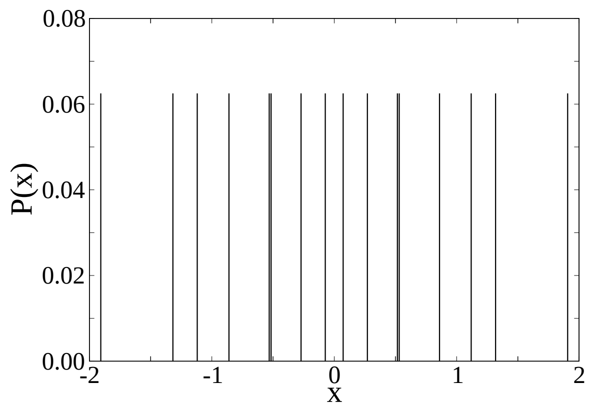
Let us first consider the case that $\rho = 0.5$. Given this exponential base, the power of the $(n+1)$-th chip will be exactly a quarter of the n-th chip. The corresponding SM-UPA is actually bijective and uniform, regardless of the bit load. Shown in Fig. 3.10(a), the symbol cardinality is exactly given by 2^N, and the symbol distribution is uniform. As a result, the respective mutual information (MI) curves are apart from the capacity curve for



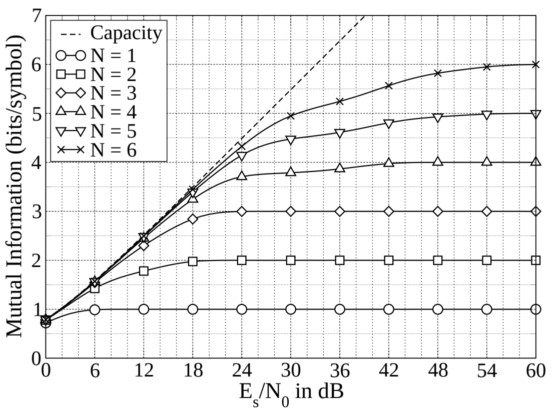
(a) Symbol distribution, $\rho = 0.5$, $N = 4$.



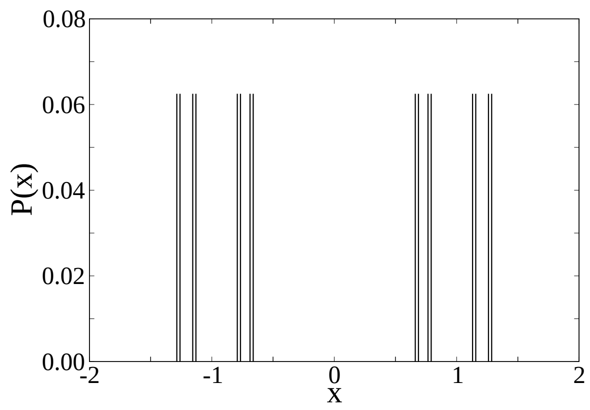
(b) Mutual information, $\rho = 0.5$.



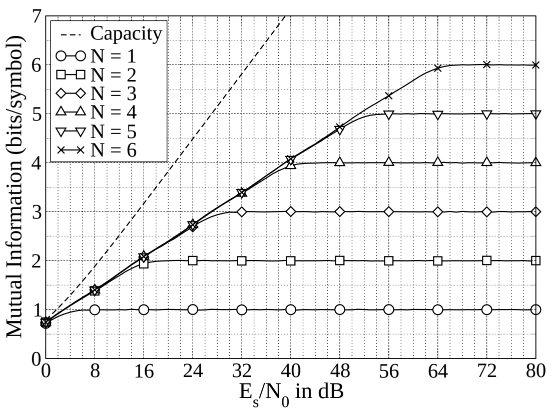
(c) Symbol distribution, $\rho = 0.75$, $N = 4$.



(d) Mutual information, $\rho = 0.75$.



(e) Symbol distribution, $\rho = 0.25$, $N = 4$.



(f) Mutual information, $\rho = 0.25$.

Figure 3.10: Symbol distribution and mutual information of SM-UPA.

about 1.53 dB in the linear section, depicted in Fig. 3.10(b). Comparing Fig. 3.10(b) with Fig. 3.7(a), one recognizes that the MI performance of SM-UPA in this case is identical to that of uniform ASK. A nice feature is that the supportable bandwidth efficiency, given by the top position of the MI curves, is linear in the bit load, while an obvious drawback is that the mapping scheme is not capacity-achieving, just like the case of uniform ASK mapping. Hence, the SM scheme under consideration does not show any advantage over conventional uniform ASK mapping but also no disadvantage, from an information theoretical point of view. To be elaborated in Section 3.4.2, SM-UPA with $\rho = 0.5$ exactly emulates a uniform ASK with natural labeling.

For most bases $\rho \neq 0.5$, the symbol distribution of SM-UPA is still probabilistically uniform but the constellation points will no longer be equispaced. For $0.5 < \rho < 1$, the density of constellation points in the region close to zero is higher than that in the region far from zero, depicted in Fig. 3.10(c). Consequently, the capacity-achieving SNR range is wider than that of $\rho = 0.5$, cf. Fig. 3.10(d). However, for $N \geqslant 4$ the MI curves shows a strange shape in the middle section. Note that in Fig. 3.10(c) there are two pairs of constellation points being very close to each other around ± 0.5, which explains the transition behaviour of the MI curves from low SNR to high SNR. When one chooses $0 < \rho < 0.5$, the symbol property of SM-UPA becomes rather undesirable. As demonstrated by Fig. 3.10(e), the constellation points are level-wise polarized by relatively strong chips, and the minimum distance between constellation points is rather small for high bit loads. This feature is clearly unfavorable for the receiver algorithm. Besides, the symbol distribution is geometrically nonuniform, however, developing in a trend that being more and more non-Gaussian, which is adverse for the MI performance, cf. Fig. 3.10(f).

In general, non-equispaced constellation presents a troublesome issue for the front-end circuits which likely introduce certain nonlinear distortion. To improve the MI property of SM in the linear section, one should choose equal power allocation instead of UPA with $0.5 < \rho < 1$, from a practical point of view. Therefore, among all the possibilities for ρ, 0.5 is a distinguished choice for SM-UPA. In the rest of this thesis, an exponential law with $\rho = 0.5$ will always be assumed for SM-UPA, if not explicitly stated otherwise.

3.4.2 Mapping and Labeling

Given the exponential law with $\rho = 0.5$, the mapping rule of SM-UPA

$$\phi_{\text{SM-UPA}}(\mathbf{b}) = a \sum_{n=1}^{N} \rho^{n-1}(1 - 2b_n) \, , \quad \mathbf{b} \in \mathbb{F}_2^N \, , \tag{3.27}$$

will always be bijective. Hence, no information loss will occur during the mapping proce-

N	$\|\mathcal{X}\|$	$H(x)$	$H(x)/N$
1	2	1.0 bits	1.0
2	4	2.0 bits	1.0
3	8	3.0 bits	1.0
4	16	4.0 bits	1.0
5	32	5.0 bits	1.0
6	64	6.0 bits	1.0

Table 3.4: Symbol entropies and compression rates of SM-UPA, $\rho = 0.5$.

b_0, b_1, b_2, b_3	c_0, c_1, c_2, c_3	$x = \sum_{n=1}^{4} c_n$
0 0 0 0	+1 +0.5 +0.25 +0.125	+1.875
0 0 0 1	+1 +0.5 +0.25 −0.125	+1.625
0 0 1 0	+1 +0.5 −0.25 +0.125	+1.375
0 0 1 1	+1 +0.5 −0.25 −0.125	+1.125
0 1 0 0	+1 −0.5 +0.25 +0.125	+0.875
0 1 0 1	+1 −0.5 +0.25 −0.125	+0.625
0 1 1 0	+1 −0.5 −0.25 +0.125	+0.375
0 1 1 1	+1 −0.5 −0.25 −0.125	+0.125
1 0 0 0	−1 +0.5 +0.25 +0.125	−0.125
1 0 0 1	−1 +0.5 +0.25 −0.125	−0.375
1 0 1 0	−1 +0.5 −0.25 +0.125	−0.625
1 0 1 1	−1 +0.5 −0.25 −0.125	−0.875
1 1 0 0	−1 −0.5 +0.25 +0.125	−1.125
1 1 0 1	−1 −0.5 +0.25 −0.125	−1.375
1 1 1 0	−1 −0.5 −0.25 +0.125	−1.625
1 1 1 1	−1 −0.5 −0.25 −0.125	−1.875

Table 3.5: Mapping rule of SM-UPA, $N = 4$, $a = 1$, $\rho = 0.5$.

dure, and sequentially the compression rate of SM-UPA will always be equal to 1, as listed in Tab. 3.4. For this reason, error-free transmission is also possible for uncoded SM-UPA, which gives a big difference w.r.t. SM-EPA. Tab. 3.5 elaborates the mapping rule of SM-UPA with $N = 4$. One may recognize that this mapping rule is exactly the same as that of uniform 16-ASK mapping with natural labeling. Certainly, the resulting performance will also be identical to that of 16-ASK with natural labeling. Therefore, conventional uniform ASK mapping can easily be emulated by SM-UPA, as long as natural labeling is desired, which brings a practical benefit that one may use a tree-based BCJR [49] algorithm to reduce the detection complexity, as introduced in the next chapter.

3.5 Grouped Power Allocation (GPA)

In Section 3.3, we have seen that superposition mapping with equal power allocation delivers a Gaussian-like symbol distribution, which brings a capacity-achieving power efficiency, but suffers from a logarithmically growing symbol entropy w.r.t. the bit load, which significantly limits the supportable bandwidth efficiency for a reasonable computational complexity. On the other hand, superposition mapping with unequal power allocation offers a linearly growing symbol entropy w.r.t. the bit load, which brings a virtually unlimited supportable bandwidth efficiency, but suffers from a geometrically and probabilistically uniform symbol distribution, which eliminates the possibility to achieve the Gaussian channel capacity, as described in Section 3.4. Therefore, both equal power allocation and unequal power allocation have their pros and cons. In short, equal power allocation is beneficial for power efficiency, while unequal power allocation is beneficial for bandwidth efficiency.

Today's communication systems demand high power efficiency and high bandwidth efficiency simultaneously. Meanwhile, the computational complexity is still a critical concern for the reason of hardware cost and electricity consumption. Therefore, it is of great practical interest to design a power allocation strategy for superposition mapping, such that high power and bandwidth efficiency can be achieved simultaneously at an affordable computational complexity. In this section, a grouped power allocation (GPA) scheme is proposed, which is a hybrid of equal and unequal power allocation strategy. GPA shows the merits of EPA and UPA, while considerably eliminates the problems from both.

3.5.1 Basic Idea

From Fig. 3.5 one observes that equal power allocation helps to build up a Gaussian-like symbol distribution, while from Fig. 3.10 one sees that unequal power allocation helps to increase the symbol cardinality. Following this observation, we may construct a hybrid power allocation strategy such that superimposed chips are divided into several groups with each group assigned a different power level and chips within each group assigned an identical power level. Given a sufficiently large group size, the summation of chips within each individual group will have a Gaussian-like distribution, and consequently the summation of multiple groups will also have a Gaussian-like distribution. Meanwhile, due to the existence of multiple power levels, the symbol cardinality will be enlarged and thus the supportable bandwidth efficiency will be improved, comparing to the case of SM-EPA.

The above idea can be formulated as follows:

$$x = \sum_{l=1}^{L} \alpha_l \sum_{g=1}^{G} d_{l,g} \ , \qquad d_{l,g} \in \{\pm 1\} \ , \tag{3.28}$$

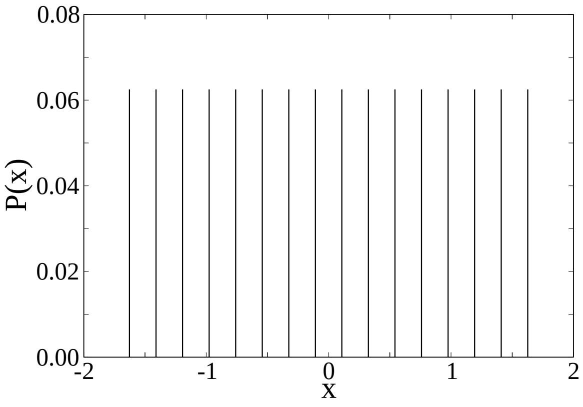
where L gives the number of power levels and G gives the group size. Clearly, $N = L \cdot G$ defines the total amount of chips per symbol, i.e., the bit load. α_l is the amplitude coefficient of the l-th power level, which is defined as

$$\alpha_l \doteq a \ 2^{-(l-1)} \tag{3.29}$$

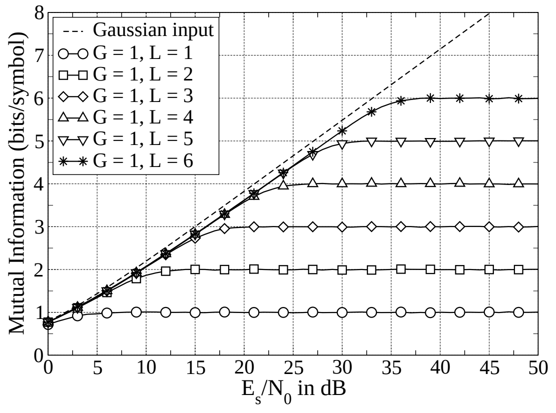
with the value of a chosen to fulfill $\mathrm{E}\{x^2\} = E_s$. Note that the base of exponential has been carefully chosen to be 2. The reason of this choice is obvious when one observes the elegant symbol distribution resulting from this power allocation strategy.

3.5.2 Symbol Distribution & Mutual Information

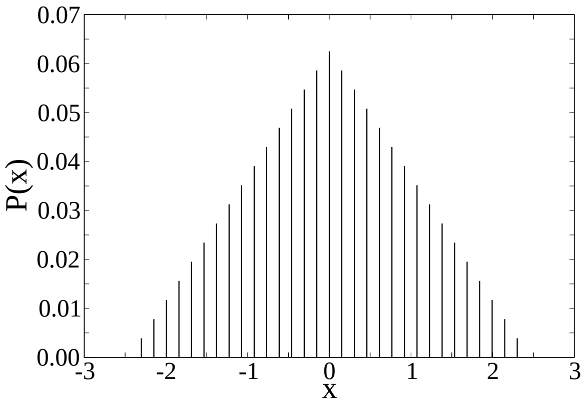
Shown in Section 3.3, as long as the symbol distribution has a Gaussian-like envelope, the potential to achieve the Gaussian channel capacity will be guaranteed. Here we investigate the situation of superposition mapping with grouped power allocation. Fig. 3.11 together with Fig. 3.12 provides an overview on the symbol distribution and mutual information of SM-GPA with various setups. When one chooses $G = 1$, the symbol distribution will be uniform, cf. Fig. 3.11(a). This is easy to understand since SM-GPA with such a setup is equivalent to SM-UPA. Therefore, the resulting mutual information (MI) curves are not capacity-achieving in the linear section, cf. Fig. 3.11(b). Increasing the group size G from 1 to 2, a fundamental change occurs. Now, the symbol distribution has a triangular envelope, demonstrated by Fig. 3.11(c). A triangular distribution envelope is not optimal but much more Gaussian-like than a uniform one. From Fig. 3.11(d), one sees that the corresponding MI curves are almost capacity-achieving in the linear section. If one further increases the group size to $G = 3$, the resulting symbol distribution of SM-GPA exhibits an elegant bell shape and the respective MI curves are indeed capacity achieving in the linear section, cf Fig. 3.11(e) and Fig. 3.11(f). This gives a message that the concept of grouped power allocation is very effective in improving the achievable power efficiency of superposition mapping. One may notice that the shape of the distribution envelope is solely determined by the group size G and in fact a moderate value as $G = 3$ is already sufficient for an optimal power efficiency. By choosing an even larger group size, the symbol distribution becomes more Gaussian, illustrated in Fig. 3.12, but brings no noticeable improvement for the MI performance. Note that the performance improvement is considerable by increasing $G = 1$ to $G = 2$, but is marginal from $G = 2$ to $G = 3$. From an engineering standpoint, $G = 2$ deserves to be a good choice. Later on, in Chapter 7 we will show that $G = 2$ leads to a lower receiver complexity w.r.t. $G = 1$ and $G \geqslant 3$.



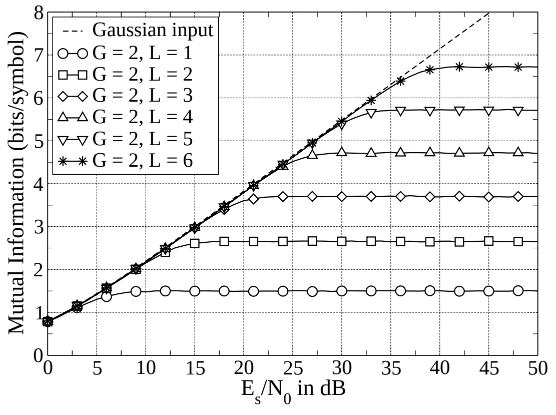
(a) Symbol distribution, $G = 1$, $L = 4$.



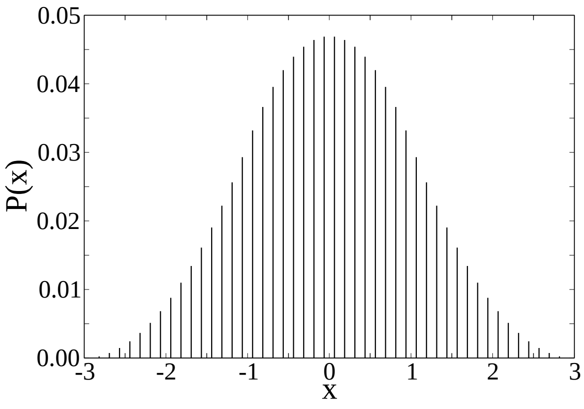
(b) Mutual information vs. SNR, $G = 1$.



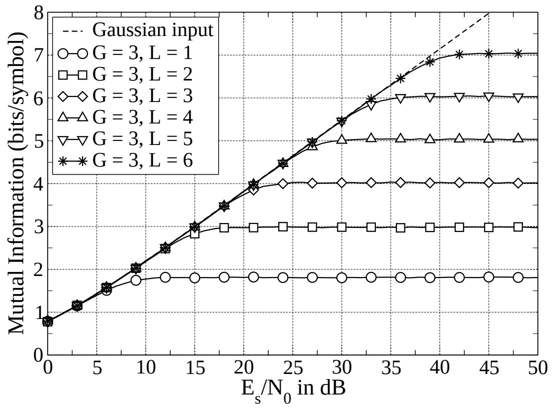
(c) Symbol distribution, $G = 2$, $L = 4$.



(d) Mutual information vs. SNR, $G = 2$.

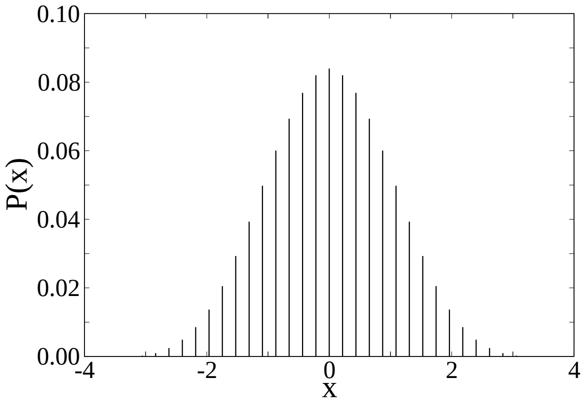


(e) Symbol distribution, $G = 3$, $L = 4$.

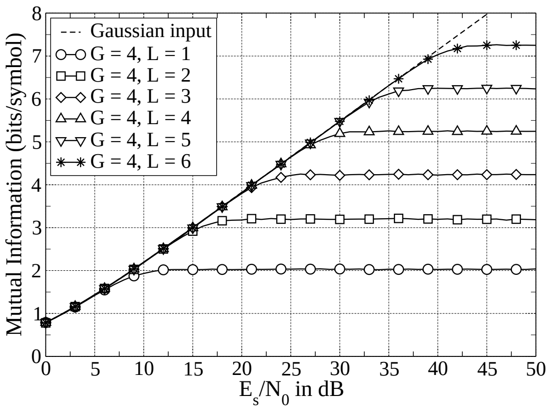


(f) Mutual information vs. SNR, $G = 3$.

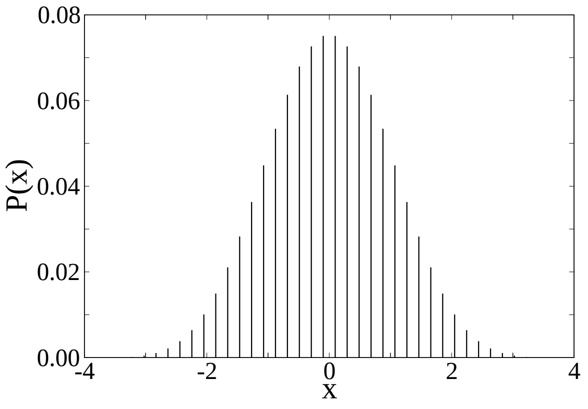
Figure 3.11: Symbol distribution and mutual information of SM-GPA, $E_s = 1$.



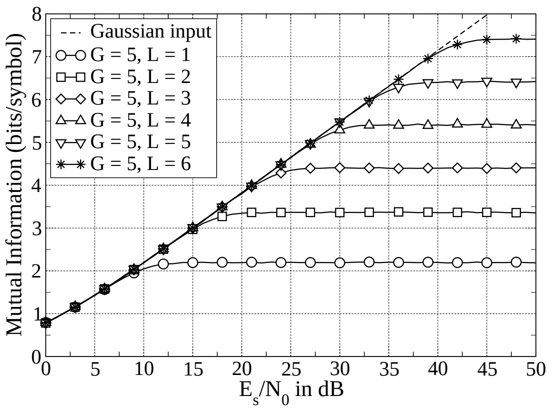
(a) Symbol distribution, $G = 4$, $L = 3$.



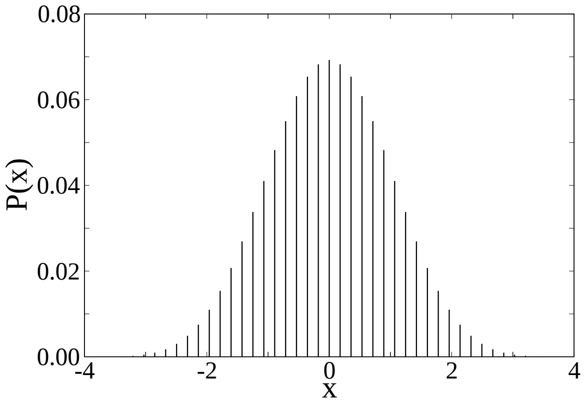
(b) Mutual information vs. SNR, $G = 4$.



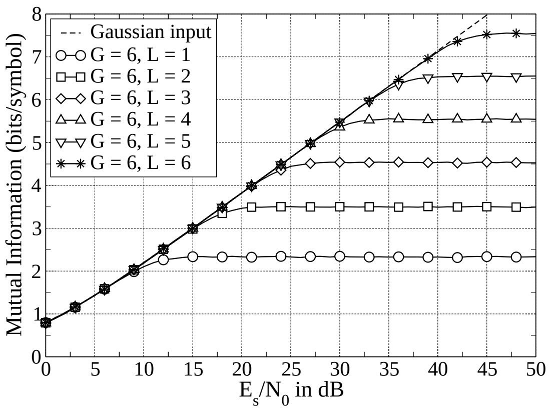
(c) Symbol distribution, $G = 5$, $L = 3$.



(d) Mutual information vs. SNR, $G = 5$.

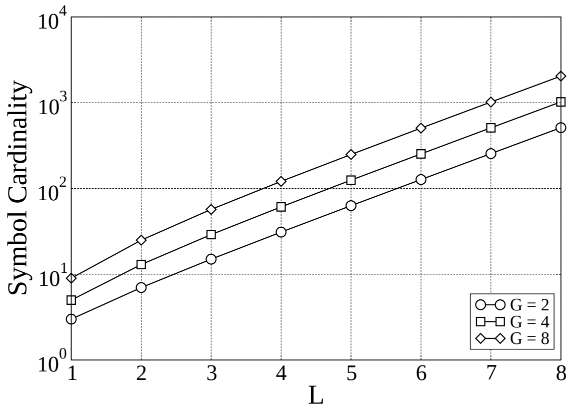


(e) Symbol distribution, $G = 6$, $L = 3$.

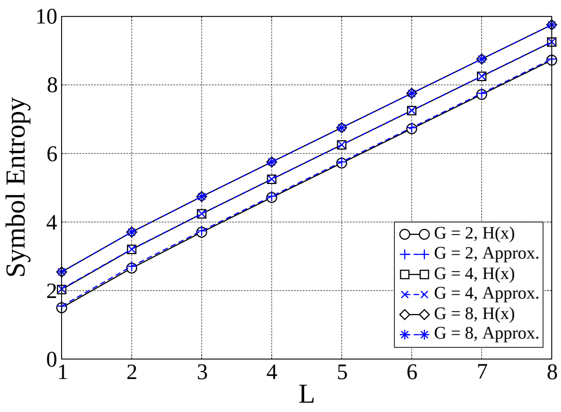


(f) Mutual information vs. SNR, $G = 6$.

Figure 3.12: Symbol distribution and mutual information of SM-GPA, $E_s = 1$.



(a) Symbol cardinality vs. L.



(b) Symbol entropy vs. L.

Figure 3.13: Symbol cardinality and symbol entropy of SM-GPA.

3.5.3 Symbol Cardinality & Symbol Entropy

For SM-GPA, the relationship between the symbol cardinality and the bit load is not so easy to be seen, due to the existence of multiple power levels. Therefore, some additional effort is taken below to reveal the connection between SM-GPA symbol cardinality and the respective parameters.

In general, the value span of SM-GPA symbols is given by

$$\max(x) - \min(x) = 2\sum_{l=1}^{L}\sum_{g=1}^{G} a2^{-(l-1)} = 2G\frac{a(1-2^{-L})}{1-2^{-1}} = 4Ga(1-2^{-L}) \ ,$$

where the second equality follows from the property of geometric series [50]. From Fig. 3.11 and Fig. 3.12 one observes that the constellation points of SM-GPA are always equispaced, which is easy to understand by concerning the specifically chosen exponential base. The distance between two neighboring constellation points is always given by two times the chip magnitude of the weakest power level:

$$\Delta = 2 \cdot a2^{-(L-1)} = 4a2^{-L} \ .$$

Given the above statements, the symbol cardinality of SM-GPA can be determined as

$$|\mathcal{X}| = \frac{\max(x) - \min(x)}{\Delta} + 1 = G(2^L - 1) + 1 \ . \tag{3.30}$$

Equation (3.30) might be interpreted in two ways. Fixing the number of power levels, the symbol cardinality is approximately linear w.r.t. the group size, similar to the case of SM-EPA. However, for a fixed group size, the symbol cardinality is approximately exponential in the number of power levels, cf. Fig. 3.13(a). Correspondingly one sees in Fig. 3.13(b) an almost linear relationship between symbol entropy and L for a given group size, which is to be explained in next step.

G	L	$N = GL$	$\lvert\mathcal{X}\rvert$	$H(x)$	$H(x)/N$
1	1	1	2	1.0000 bits	1.0000
1	2	2	4	2.0000 bits	1.0000
1	3	3	8	3.0000 bits	1.0000
1	4	4	16	4.0000 bits	1.0000
2	1	2	3	1.5000 bits	0.7500
2	2	4	7	2.6556 bits	0.6639
2	3	6	15	3.7023 bits	0.6171
2	4	8	31	4.7159 bits	0.5895
3	1	3	4	1.8113 bits	0.6038
3	2	6	10	2.9843 bits	0.4974
3	3	9	22	4.0247 bits	0.4472
3	4	12	46	5.0345 bits	0.4195
4	1	4	5	2.0306 bits	0.5077
4	2	8	13	3.2014 bits	0.4002
4	3	12	29	4.2386 bits	0.3532
4	4	16	61	5.2475 bits	0.3280

Table 3.6: Symbol cardinalities, symbol entropies, and compression rates of SM-GPA.

Following (3.28), the average symbol energy of SM-GPA comes straightforward:

$$\mathrm{E}\left\{x^2\right\} = \sum_{l=1}^{L}\sum_{g=1}^{G} a^2 2^{-2(l-1)}\mathrm{E}\left\{d_{l,g}^2\right\} = Ga^2\sum_{l=1}^{L} 2^{-2(l-1)} = Ga^2\frac{1-4^{-L}}{1-4^{-1}}\ . \tag{3.31}$$

As long as G is not too small, i.e., the symbol distribution is Gaussian-like, one may utilize formula (C.10) provided in Appendix C.3 and sequentially obtain

$$\begin{aligned} H(x) &\approx \frac{1}{2}\log(2\pi e\sigma_x^2/\Delta^2) = \frac{1}{2}\log(2\pi eGa^2\frac{1-4^{-L}}{1-4^{-1}}\Big/(16a^2 2^{-2L})) \\ &= \frac{1}{2}\log\left(\frac{\pi}{6}eG(2^{2L}-1)\right)\ . \end{aligned} \tag{3.32}$$

Fig. 3.13(b) compares the measured symbol entropy of SM-GPA with the above approximation. Clearly, for $G > 2$ this approximation is rather good. For large L, one may further simplify the expression as

$$H(x) \approx \frac{1}{2}\log\left(\frac{\pi}{6}eG\cdot 2^{2L}\right) = \frac{1}{2}\log_2\left(\frac{\pi}{6}eG\right) + L \quad \text{bits}\ , \tag{3.33}$$

which tells that by using one more power level we obtain about 1 bit of entropy increase for SM-GPA. This can also be seen from Tab. 3.6. Comparing Tab. 3.6 with Tab. 3.3, we will find that SM-GPA is much more efficient than SM-EPA in the sense of supportable bandwidth efficiency, given identical bit loads.

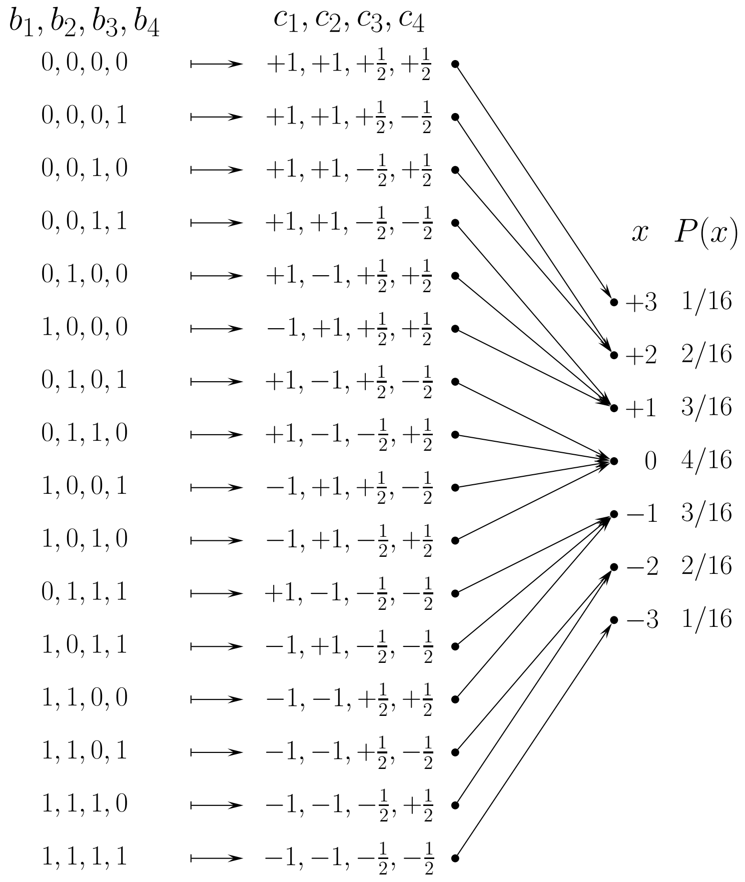

b_1, b_2, b_3, b_4
c_1, c_2, c_3, c_4
$0,0,0,0 \longmapsto +1,+1,+\frac{1}{2},+\frac{1}{2}$
$0,0,0,1 \longmapsto +1,+1,+\frac{1}{2},-\frac{1}{2}$
$0,0,1,0 \longmapsto +1,+1,-\frac{1}{2},+\frac{1}{2}$
$0,0,1,1 \longmapsto +1,+1,-\frac{1}{2},-\frac{1}{2}$
$0,1,0,0 \longmapsto +1,-1,+\frac{1}{2},+\frac{1}{2}$
$1,0,0,0 \longmapsto -1,+1,+\frac{1}{2},+\frac{1}{2}$
$0,1,0,1 \longmapsto +1,-1,+\frac{1}{2},-\frac{1}{2}$
$0,1,1,0 \longmapsto +1,-1,-\frac{1}{2},+\frac{1}{2}$
$1,0,0,1 \longmapsto -1,+1,+\frac{1}{2},-\frac{1}{2}$
$1,0,1,0 \longmapsto -1,+1,-\frac{1}{2},+\frac{1}{2}$
$0,1,1,1 \longmapsto +1,-1,-\frac{1}{2},-\frac{1}{2}$
$1,0,1,1 \longmapsto -1,+1,-\frac{1}{2},-\frac{1}{2}$
$1,1,0,0 \longmapsto -1,-1,+\frac{1}{2},+\frac{1}{2}$
$1,1,0,1 \longmapsto -1,-1,+\frac{1}{2},-\frac{1}{2}$
$1,1,1,0 \longmapsto -1,-1,-\frac{1}{2},+\frac{1}{2}$
$1,1,1,1 \longmapsto -1,-1,-\frac{1}{2},-\frac{1}{2}$
x $P(x)$
$+3$ 1/16
$+2$ 2/16
$+1$ 3/16
0 4/16
-1 3/16
-2 2/16
-3 1/16

Figure 3.14: Mapping rule of SM-GPA with $G = 2$, $L = 2$, and $a = 1$.

3.5.4 Mapping & Labeling

From Tab. 3.1 we see that SM-EPA is a non-bijective mapping scheme for $N \geqslant 2$ and it employs a certain type of typical-sequence labeling. Tab. 3.5 shows that SM-UPA is a bijective mapping scheme with natural labeling. As GPA is a mixture of EPA and UPA, one may wonder the resulting labeling mechanism of SM-GPA. Fig. 3.14 provides an example of the mapping rule of SM-GPA. In general, SM-GPA is non-bijective for $G \geqslant 2$. As for the case of $G = L = 2$, the symbol distribution exhibits a triangular envelope. The corresponding labeling mechanism is neither natural labeling nor typical-sequence labeling. Instead, it is a mixture of natural labeling and typical-sequence labeling, due to the interaction between chips from multiple power levels. Checking Fig. 3.14 carefully one will find that the combination of the first two bits changes the symbol value in a large scale while the combination of the last two bits changes the symbol value in a small scale. Inside each power level, typical-sequence labeling still takes place. As a result, we may term this as a partial-typical-sequence labeling. In the case of $G = L = 2$, for $x = \pm 1$, the corresponding bit combinations are not ϵ-typical with $\epsilon = 0$, but for all $x \neq \pm 1$, the bit combinations are always 0-typical w.r.t. a certain Bernoulli distribution.

Chapter 4

Uncoded SM Transmission

In the previous chapter, three different power allocation strategies were introduced for superposition mapping: equal power allocation (EPA), unequal power allocation (UPA), and grouped power allocation (GPA). Among these three strategies, EPA and GPA provide a Gaussian-like symbol distribution and meanwhile make the mapping scheme non-bijective. In contrast, SM-UPA is uniform and bijective and is eventually equivalent to conventional ASK with natural labeling. In this chapter, the performance of uncoded SM transmission will be studied, given the three power allocation strategies. Easy to imagine, error-free uncoded transmission will not be possible for SM-EPA and SM-GPA, whenever they are non-bijective. It is also easy to imagine that the performance of uncoded SM-UPA will be identical to that of ASK with natural labeling. Nevertheless, the corresponding investigations are very helpful for obtaining a better understanding on the working mechanism of superposition mapping. Particular focus will be put on the performance of maximum a posteriori demapping for non-bijective SM-EPA.

4.1 Maximum-A-Posteriori Demapping

The optimal receiver algorithm for uncoded SM transmission over the Gaussian channel is the maximum a posteriori (MAP) demapping approach, in the sense of minimizing the bit error rate (BER). For easy reference, let us repeat here the basic formula of superposition mapping:

$$x = \sum_{n=1}^{N} c_n = \sum_{n=1}^{N} \alpha_n d_n = \sum_{n=1}^{N} \alpha_n (1 - 2b_n) \tag{4.1}$$

and the AWGN channel model:

$$y = x + z \; , \quad z \sim \mathcal{N}(0, \sigma_z^2) \; . \tag{4.2}$$

Given the above notation and let $\mathbf{b}_{\sim n} \doteq \{b_1, \ldots, b_{n-1}, b_{n+1}, \ldots, b_N\}$ denote the bit set excluding b_n, an MAP demapper computes

$$\begin{aligned}
\hat{b}_n &= \arg\max_{b_n \in \{0,1\}} \{p(b_n|y)\} = \arg\max_{b_n \in \{0,1\}} \{p(y|b_n)P(b_n)/p(y)\} \\
&\overset{(a)}{=} \arg\max_{b_n \in \{0,1\}} \{p(y|b_n)\} \\
&\overset{(b)}{=} \arg\max_{b_n \in \{0,1\}} \left\{ \sum_{\mathbf{b}_{\sim n}} p(y|b_1, \ldots, b_n, \ldots, b_N) \prod_{i=1, i\neq n}^{N} P(b_i) \right\} \\
&\overset{(c)}{=} \arg\max_{b_n \in \{0,1\}} \left\{ \sum_{\mathbf{b}_{\sim n}} p(y|b_1, \ldots, b_n, \ldots, b_N) \right\} \\
&= \arg\max_{b_n \in \{0,1\}} \left\{ \sum_{\mathbf{b}_{\sim n}} \frac{1}{\sqrt{2\pi\sigma_z^2}} \exp\left(-\frac{\left(y - \sum_{n=1}^{N} \alpha_n (1-2b_n)\right)^2}{2\sigma_z^2} \right) \right\}, \qquad (4.3)
\end{aligned}$$

where equality (a), (b), and (c) follow from the assumption that the input bits of the superposition mapper are uniformly distributed and mutually independent. A straightforward evaluation of (4.3) involves a complexity proportional to 2^N. Nevertheless, as the focus of this chapter is merely on theoretical issues, the discussion on reduced-complexity demapping will be excluded here and treated later in Chapter 5.

4.2 Bit Error Probability of SM-EPA

Given equal power allocation, SM is non-bijective for all $N \geqslant 2$, or in other words it is lossy for $N \geqslant 2$. Consequently, error-free reconstruction of the input bits will be only possible at $N = 1$, if no channel coding is applied. The Monte Carlo simulation results in Fig. 4.1 fully agree with this conjecture. There is an interesting phenomenon in Fig. 4.1. The error floor level of $N = 2$ and $N = 3$ are identical, and the same is for $N = 4$ and $N = 5$. In fact, this holds in general for $N = 2n$ and $N = 2n + 1$, $n \in \mathbb{Z}^+$. To clearly explain this phenomenon a deep insight into the MAP demapping procedure is necessary.

Since the focus is on the level of error floor, it is sufficient to investigate the case of noiseless transmission, which largely simplifies the mathematical description. For noiseless SM-EPA transmission, i.e., $y = x$, the MAP demapping formula (4.3) can be rewritten as

$$\begin{aligned}
\hat{b}_n &= \arg\max_{b_n \in \{0,1\}} \{P(x|b_n)\} = \arg\max_{b_n \in \{0,1\}} \left\{ \sum\nolimits_{\mathbf{b}_{\sim n}} P(x|\mathbf{b}) \prod\nolimits_{i=1, i\neq n}^{N} P(b_i) \right\} \\
&= \arg\max_{b_n \in \{0,1\}} \left\{ \sum\nolimits_{\mathbf{b}_{\sim n}} \delta\left(x - \sum\nolimits_{i=1}^{N} \alpha(1-2b_i)\right) \right\}, \qquad (4.4)
\end{aligned}$$

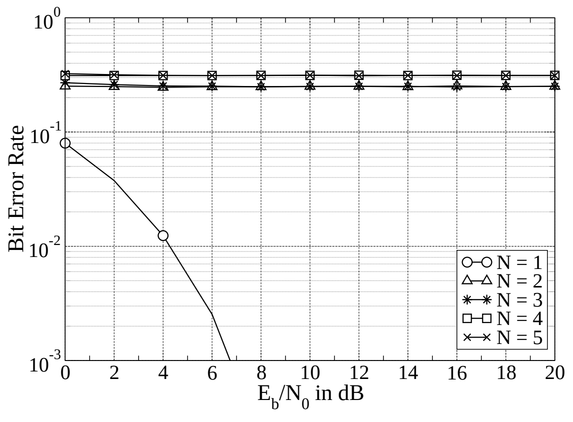


Figure 4.1: BER vs. E_b/N_0, SM-EPA with MAP demapping.

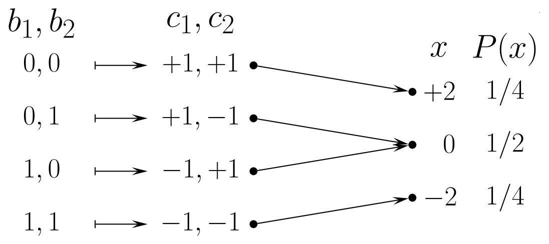


Figure 4.2: Mapping rule of SM-EPA with $N = 2$, $\alpha = 1$.

with the Dirac delta function defined as $\delta(\tau) = 1$ for $\tau = 0$ and $\delta(\tau) = 0$ for $\tau \neq 0$. This is basically to count the amount of bit combinations that would give a summation equal to the received value x, with the value of b_n fixed to 0 and 1 respectively. Note that the amplitude coefficients α_n has been replaced by a single α w.r.t. (4.3), due to EPA.

Let us first consider the case of $N = 2$ and for the sake of simplicity we take $\alpha = 1$. The corresponding mapping rule is illustrated in Fig. 4.2, which also gives the probability distribution of the mapper output with the assumption of i.u.d. input bits. Clearly, the mapping procedure varies with the specific value of received sample x, and so does the demapping procedure. To see this, we define a shorthand notation for the MAP metric:

$$\psi(b_n) \doteq \sum\nolimits_{\mathbf{b}_{\sim n}} \delta\left(x - \sum\nolimits_{i=1}^{N}(1 - 2b_i)\right) . \tag{4.5}$$

Carefully checking Fig. 4.2, one obtains the following logical chain:

$$\begin{array}{rcccccc} x = +2 & \to & \psi(b_n = 0) = 1\,, \;\; \psi(b_n = 1) = 0 & \to & \hat{b}_n = 0 & \to & P_e|_{x=+2} = 0 \\ x = \;\;\;0 & \to & \psi(b_n = 0) = 1\,, \;\; \psi(b_n = 1) = 1 & \to & \hat{b}_n = \,? & \to & P_e|_{x=0} \;= \frac{1}{2} \\ x = -2 & \to & \psi(b_n = 0) = 0\,, \;\; \psi(b_n = 1) = 1 & \to & \hat{b}_n = 1 & \to & P_e|_{x=-2} = 0 \end{array} ,$$

where the question mark "?" stands for the fact that no reliable decision can be made. $P_e|_x$ denotes the error probability of MAP decision given a certain value of x. Due to the symmetry of the mapping rule, the situation is identical for b_1 and b_2. Considering

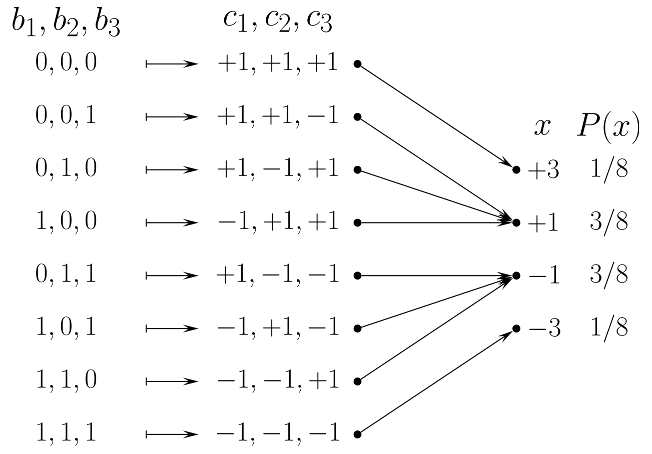

b_1, b_2, b_3
c_1, c_2, c_3
$0,0,0 \longmapsto +1,+1,+1$
$0,0,1 \longmapsto +1,+1,-1$
$0,1,0 \longmapsto +1,-1,+1$
$1,0,0 \longmapsto -1,+1,+1$
$0,1,1 \longmapsto +1,-1,-1$
$1,0,1 \longmapsto -1,+1,-1$
$1,1,0 \longmapsto -1,-1,+1$
$1,1,1 \longmapsto -1,-1,-1$
x $P(x)$
$+3$ $1/8$
$+1$ $3/8$
-1 $3/8$
-3 $1/8$

Figure 4.3: Mapping rule of SM-EPA with $N = 3$, $\alpha = 1$.

the nonuniform distribution of the output symbol, the average error probability of MAP decision is given by

$$P_e = \sum_{x\in\mathcal{X}} P(x) P_e|_x = \frac{1}{4}\cdot 0 + \frac{1}{2}\cdot\frac{1}{2} + \frac{1}{4}\cdot 0 = 0.25 \ ,$$

which well fits with the observation in Fig. 4.1.

In a similar way, one obtains for $N = 3$ the following logical chain:

$$\begin{array}{lllllllll} x=+3 & \rightarrow & \psi(b_n=0)=1\,, & \psi(b_n=1)=0 & \rightarrow & \hat{b}_n=0 & \rightarrow & P_e|_{x=+3}=0 \\ x=+1 & \rightarrow & \psi(b_n=0)=2\,, & \psi(b_n=1)=1 & \rightarrow & \hat{b}_n=0 & \rightarrow & P_e|_{x=+1}=\frac{1}{3} \\ x=-1 & \rightarrow & \psi(b_n=0)=1\,, & \psi(b_n=1)=2 & \rightarrow & \hat{b}_n=1 & \rightarrow & P_e|_{x=-1}=\frac{1}{3} \\ x=-3 & \rightarrow & \psi(b_n=0)=0\,, & \psi(b_n=1)=1 & \rightarrow & \hat{b}_n=1 & \rightarrow & P_e|_{x=-3}=0 \end{array} ,$$

by referring to Fig. 4.3. The bit error probability given $x = +1$ follows from the fact that

$$P_e|_{x=+1} = P(b_n \neq \hat{b}_n | x = +1) = P(b_n \neq 0 | x = +1) = 1/3 \ .$$

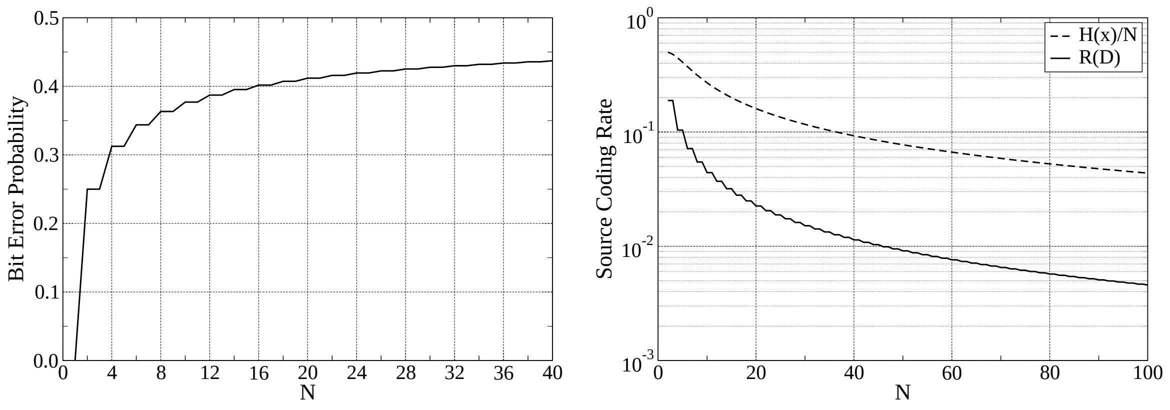
Using the same approach, one obtains $P_e|_{x=-1} = 1/3$ as well. With these derivation, the average bit error probability of MAP demapping for $N = 3$ comes in a straightforward way:

$$P_e = \sum_{x\in\mathcal{X}} P(x) P_e|_x = \frac{1}{8}\cdot 0 + \frac{3}{8}\cdot\frac{1}{3} + \frac{3}{8}\cdot\frac{1}{3} + \frac{1}{8}\cdot 0 = 0.25 \ ,$$

which is identical to the case of $N = 2$ and agrees with the observation in Fig. 4.1 as well.

Summarizing the above two examples, in general one has the bit error probability of MAP demapping for noiseless SM-EPA transmission as

$$P_e = \sum_{i=0}^{N} \frac{\binom{N}{i}}{2^N} \cdot \frac{\min\{i, N-i\}}{N} \ . \tag{4.6}$$



(a) P_e vs. N.   (b) Source coding rate vs. N.

Figure 4.4: SM-EPA with MAP demapping, noiseless channel.

It deserves to be a fortuitous event that $N = 2n$ and $N = 2n + 1$, $n \in \mathbb{Z}^+$, will always give the same error probability. As shown in Fig. 4.4(a), the error probability grows in a stair-wise manner. One may also notice from Fig. 4.4(a) that the error probability monotonically increases as N becomes larger. This tells that the information loss during SM-EPA mapping becomes more and more severe with the rising of N. At this point, an interesting question may arise. How good is SM-EPA as a lossy source encoder? Borrowing the classical definition from information theory, we may call $H(x)/N$ the source coding rate of SM-EPA, as this rate describes the average amount of information that is preserved from each input bit. Assuming that the input bits are from a Bernoulli($\frac{1}{2}$) source, the Hamming distortion (D) will be equivalent to the bit error probability (P_e), and according to the rate distortion theory the minimum source coding rate will be given by

$$R(D) = 1 - h(D) = 1 - h(P_e) \text{ bits/source symbol} , \tag{4.7}$$

where $h(\cdot)$ denotes the binary entropy function. A source encoder achieving the rate limit $R(D)$ is usually termed as an ideal source encoder. Therefore, the distance between $H(x)/N$ and $R(D)$ indicates how well SM-EPA works as a lossy source encoder. To attain a systematic comparison between these two rates, some extra thoughts are still necessary. First, given a certain bit load N, the compression rate $H(x)/N$ of SM-EPA can be easily evaluated, by using the methods introduced in Section 3.3.3. Second, for a given N, the bit error probability P_e of SM-EPA (MAP demapping, noiseless channel) can be measured or simply computed via (4.6), and the corresponding minimum source coding rate follows directly from (4.7). Making these computations for $N = 0, 1, \ldots, 100$, one obtains the pair of curves in Fig. 4.4(b), which shows that SM-EPA is far away from being optimal in the sense of achieving the minimum source coding rate given a certain distortion. What can also be seen is that the distance between $H(x)/N$ and $R(D)$ gets larger with larger N. Hence, SM-EPA is in general not optimal in the sense of lossy source encoding.

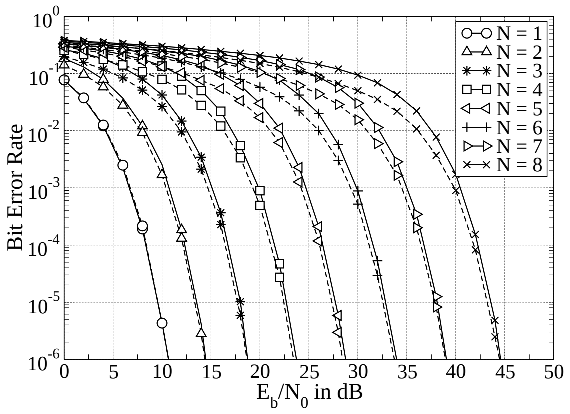


Figure 4.5: SM-UPA with $\rho = 0.5$ (solid line) vs. ASK with Gray labeling (dashed line).

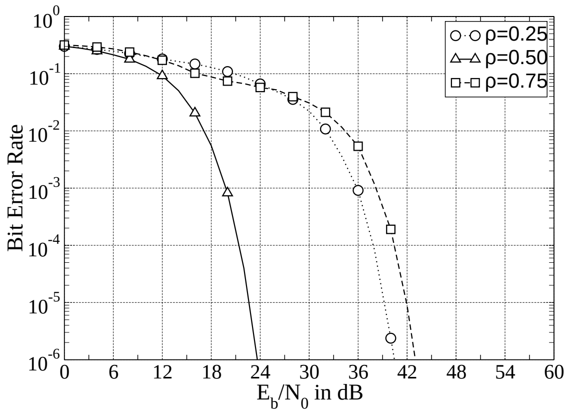
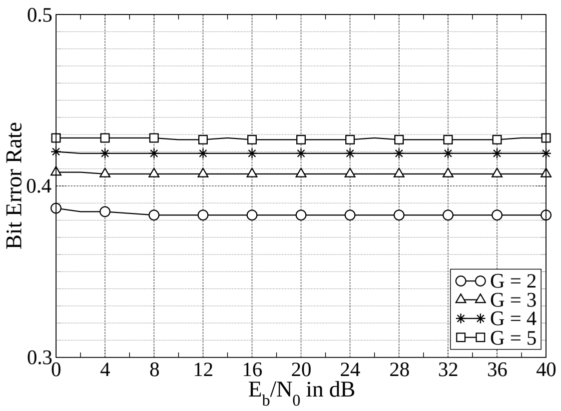


Figure 4.6: The effects of ρ, SM-UPA with MAP demapping, $N = 4$.

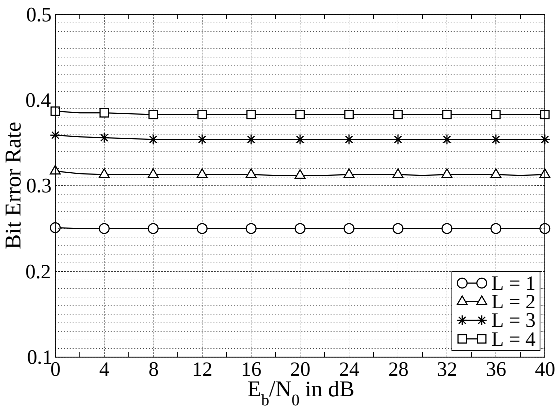
4.3 Bit Error Probability of SM-UPA

Since SM-UPA with $\rho = 0.5$ and conventional uniform ASK with natural labeling are equivalent, their BER performance should also be identical. Consequently, one expects for SM-UPA with $\rho = 0.5$ a BER performance slightly worse than that of ASK with Gray labeling. Fig. 4.5 verifies this conjecture. Because the underlying principle of this performance difference is well-known, we do not provide a detailed discussion on it.

From an information theoretical point of view, it does not make much sense to choose $\rho \neq 0.5$ for SM-UPA, as shown in Section 3.4.1. Nevertheless, it is still interesting to see the performance of SM-UPA given $\rho \neq 0.5$. Demonstrated in Fig. 4.6, the performance given $\rho = 0.25$ and $\rho = 0.75$ are both worse than that given $\rho = 0.5$. This is not difficult to understand by revisiting Fig. 3.10, as the performance of MAP demapping is primarily determined by the symbol distance, while in this concern $\rho = 0.25$ and $\rho = 0.75$ are both inferior to $\rho = 0.5$. Besides, it can be seen from Fig. 3.10 that w.r.t. $\rho = 0.25$, $\rho = 0.75$ provides a larger average symbol distance but a smaller minimum symbol distance, which explains the crossover of the BER curves in Fig. 4.6.



(a) $L = 4$.



(b) $G = 2$.

Figure 4.7: The effects of G and L, SM-GPA with MAP demapping.

4.4 Bit Error Probability of SM-GPA

By its nature, SM-GPA is a hybrid of SM-EPA and SM-UPA. Intuitively, its performance either coded or uncoded should be somewhere in between that of SM-EPA and SM-UPA, depending on the group size G and the number of power levels L. For $G \geqslant 2$, SM-GPA will be always non-bijective. Hence, one expects a considerable error floor from its uncoded performance, similar to the case of SM-EPA. Of particular interest is the relationship between the error floor level and the two parameters G and L.

From Section 4.2, one sees that the bit error probability of a non-bijective mapping scheme in case of uncoded transmission increases when the compression rate $H(x)/N$ becomes smaller. Since for SM-GPA the chips within each group are assigned identical magnitudes, their summation tends to be more and more nonuniform when one increases the group size G. Naturally, a larger G will bring a lower compression rate and subsequently a higher error floor, which is ascertained by the simulation results provided in Fig. 4.7(a).

While the influence of the group size G on the error floor level is more or less straightforward, the influence of the number of power levels L is not so explicit. To clarify the situation, we first have a look at the simulation results provided in Fig. 4.7(b), which shows that the increase of L given a fixed G also raises the error floor level. This observation in turn tells that a larger L brings a lower compression rate as well. Mathematically, we can see this from

$$H(x)/N \approx \left(\frac{1}{2}\log_2\left(\frac{\pi}{6}eG\right) + L\right) \Big/ (G \cdot L) = \frac{1}{L} \cdot \frac{\log_2(\frac{\pi}{6}eG)}{2G} + \frac{1}{G}\,, \tag{4.8}$$

which utilizes the approximation given in (3.33). After all, the reason behind this effect is the inter-group interference due to using base $\rho = 0.5$ for the exponential power decay.

Chapter 5

Coded SM Transmission

For an optimal achievable power efficiency, a Gaussian-like symbol distribution is required. In this concern, equal power allocation (EPA) or grouped power allocation (GPA) should be chosen for superposition mapping. Nevertheless, error-free transmission is strictly not possible for uncoded SM-EPA or SM-GPA, revealed by the discussion in Chapter 4. To enable error-free transmission, a non-bijective superposition mapper must be preceded by a properly designed coding/spreading module such that the compression procedure during superposition mapping does not cause any information loss. In this chapter, we will have some general discussion of coded SM transmission over the Gaussian channel. The main goal is to provide a good overview and illustrate the fundamental concepts that are necessary for later discussions. Hence, code optimization will not be a topic within this chapter. In contrast to uncoded transmission, the superposition demapper of a coded system needs to work in an iterative manner, and consequently needs to accept soft input messages and be able to deliver soft output messages. For this reason, the issue of superposition demapping needs to be re-treated in the framework of iterative processing. As for practical applications, the computational complexity is always a big concern. Therefore, the possibility of low-complexity superposition demapping will also be treated in this chapter. It will be shown that being non-bijective and being nonuniform are both helpful for reducing the demapping complexity. Correspondingly, a tree-based BCJR [49] algorithm and a Gaussian demapping algorithm will be introduced. Given the elaborated demapping algorithms, either optimal or suboptimal, the performance of superposition mapping will be tested with typical conventional coding schemes. Particularly, the advantage of (regular) repetition coding over (regular) parity-check coding in the sense of supportable bandwidth efficiency will be made clear. Several interesting relevant issues are also covered, such as typical system setup with different types of coding and respective factor graph representations.

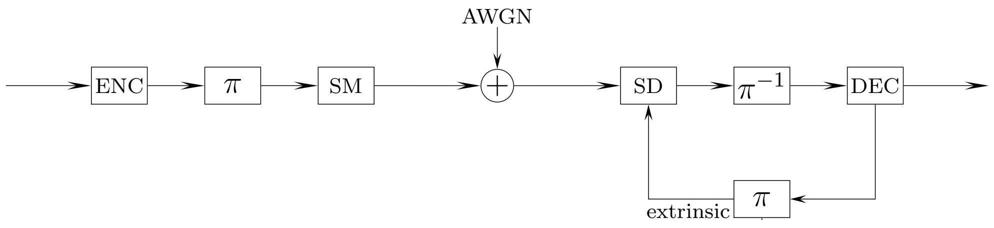


Figure 5.1: Bit-interleaved coded modulation with superposition mapping.

5.1 System Structure

Due to linear superposition, the involved binary antipodal chips of an individual SM symbol interfere with each other, whenever the adopted power allocation strategy leads to a non-bijective mapping rule. Consequently, the receiver of a coded SM transmission system often needs to apply parallel/successive interference cancellation, since a global-level maximum-likelihood detection is computationally prohibitive. In this concern, a coded SM transmission system resembles very much a multiple access system. As a side effect, multi-level coding can easily be applied, if desired, e.g., to achieve unequal error protection or simply use superposition mapping as a multiplexing scheme. In that case, the coding rate or even the code can be different on each level, which yields a great flexibility for system configuration. In this thesis, however, the discussion will be limited to single-level coded SM transmission only. From a theoretical point of view, a multi-level encoding scheme can easily be emulated by a specially designed single-level encoding scheme. The focus of this thesis is on the fundamental properties of SM and finding suitable channel codes to exploit the capacity-achieving potential of SM. For this purpose, a single-level code structure provides more flexibility for code optimization as well as a more convenient mathematical description.

Fig. 5.1 shows the system structure of single-level coded SM transmission over the Gaussian channel, where SM stands for superposition mapping and SD stands for superposition demapping. Eventually, the channel encoding module may comprise a scrambling functionality, and the channel decoding module may comprise a descrambling functionality, respectively. The interleaving module in between the channel encoder and the superposition mapper plays a very important role for the system performance. Without loss of generality, one may call such a system bit-interleaved coded modulation (BICM) with superposition mapping (SM). BICM [11] is known for offering excellent performance in conjunction with conventional uniform ASK mapping. In fact, it is also true for superposition mapping. A special issue for BICM-SM is that iterations between the demapper and the decoder are not only necessary but mandatory, whenever the superposition mapper is non-bijective. This manifests a big difference to BICM with ASK.

5.2 Soft-Input Soft-Output Demapping

For non-bijective superposition mapping, ambiguity-free detection is not possible without utilizing the feedback information from the channel decoder. Hence, non-iterative hard-output demapping introduced in Chapter 4 only makes sense for theoretical analysis, while for practical applications one should always perform soft-input soft-output (SISO) demapping. In this section we will have a systematic study on SISO superposition demapping for the three types of power allocation strategies introduced in Chapter 3. The main attention is paid to the possibility of reduced-complexity demapping. Whenever a superposition mapper is non-bijective, the resulting symbol cardinality will be smaller than that of a bijective one. Using a tree-based BCJR algorithm, a significant complexity reduction can be achieved, without any loss of optimality. The extent of complexity reduction depends on how non-bijective the mapper is. Nevertheless, we will also show that even when the superposition mapper is bijective a non-trivial complexity reduction can still be achieved via a tree-based BCJR algorithm. Whenever a superposition mapper delivers a Gaussian-like symbol distribution, one attains another opportunity to reduce the demapping complexity. Approximating the summation of multiple binary chips and an additive noise sample by a continuous Gaussian variable, a linear-complexity demapping algorithm can be implemented, albeit with a suboptimal performance.

5.2.1 Standard APP Approach

The most straightforward implementation of SISO superposition demapping is the standard a posteriori probability (APP) demapping algorithm. Upon the reception of the AWGN channel output y, a SISO superposition demapper needs to calculate the extrinsic log-likelihood ratios (LLR) of each code bit, taking into account the a priori information feedback from the decoder. Mathematically, this is to compute

$$LLR_e(b_n) \doteq \ln \frac{p(y|b_n = 0)}{p(y|b_n = 1)} \ , \quad n = 1, 2, \ldots, N \ . \tag{5.1}$$

As done previously in Section 4.1, let $\mathbf{b}_{\sim n} \doteq \{b_1, \ldots, b_{n-1}, b_{n+1}, \ldots, b_N\}$ denote the bit set excluding b_n. Similar to (4.3), the likelihood function can generally be computed as

$$\begin{aligned} p(y|b_n) &= \sum_{\mathbf{b}_{\sim n}} p(y, \mathbf{b}_{\sim n}|b_n) \\ &= \sum_{\mathbf{b}_{\sim n}} p(y|\mathbf{b}_{\sim n}, b_n) \ P(\mathbf{b}_{\sim n}) \\ &= \sum_{\mathbf{b}_{\sim n}} p(y|\mathbf{b}_{\sim n}, b_n) \ \prod_{i=1, i\neq n}^{N} P(b_i) \ , \end{aligned} \tag{5.2}$$

where the second and third equality come from the assumption that all bits are mutually independent. Combining (5.1) and (5.2), one obtains the following equation:

$$LLR_e(b_n) = \ln \frac{p(y|b_n = 0)}{p(y|b_n = 1)} = \ln \frac{\sum_{\mathbf{b}_{\sim n}} p(y|\mathbf{b}_{\sim n}, b_n = 0) \ \prod_{i=1,i\neq n}^{N} P(b_i)}{\sum_{\mathbf{b}_{\sim n}} p(y|\mathbf{b}_{\sim n}, b_n = 1) \ \prod_{i=1,i\neq n}^{N} P(b_i)} \tag{5.3}$$

with

$$p(y|\mathbf{b}_{\sim n}, b_n) = \frac{1}{\sqrt{2\pi\sigma_z^2}} \exp\left(- \frac{\left(y - \sum_{i=1}^{N} \alpha_i (1 - 2b_i)\right)^2}{2\sigma_z^2} \right) . \tag{5.4}$$

Since $\mathbf{b}_{\sim n}$ has 2^{N-1} possible value combinations and they need to be considered for both $b_n = 0$ and $b_n = 1$, a literal evaluation of (5.3) involves a complexity proportional to 2^N, which is certainly undesirable for practice but in fact common for conventional uniform mapping schemes. Considering the amount of bits per symbol, the overall demapping complexity is proportional to $N \cdot 2^N$, while the demapping complexity per bit is $\propto 2^N$.

To have a more concrete perception as well as provide reference for later use, we consider a simple example for the APP demapping of SM with equal power allocation (EPA). For simplicity, we take parameters: $N = 2$, $\alpha = 1$, and assume that the channel is noiseless. Then, the channel output will directly be

$$x = c_1 + c_2 = (1 - 2b_1) + (1 - 2b_2) . \tag{5.5}$$

Let us define a shorthand notation $B(b_n) \doteq 1 - 2b_n$ to represent the BPSK mapping operation. The likelihood of the channel output given the input bit pair is given by

$$P(x|b_1, b_2) = \delta(x - B(b_1) - B(b_2)) = \begin{cases} 1 & \text{if } x = B(b_1) + B(b_2) \\ 0 & \text{if } x \neq B(b_1) + B(b_2) \end{cases} . \tag{5.6}$$

To compute the extrinsic LLR for the first bit, one needs the following marginalization:

$$P(x|b_1) = \sum_{b_2 \in \{0,1\}} P(x|b_1, b_2) P(b_2) , \tag{5.7}$$

and for the second bit one needs

$$P(x|b_2) = \sum_{b_1 \in \{0,1\}} P(x|b_1, b_2) P(b_1) . \tag{5.8}$$

Clearly, the results of the above marginalizations depend on the channel output x and the a priori distribution $P(b_1)$ and $P(b_2)$, and so do the computed extrinsic LLRs. In the following, we try to have a deeper insight into this demapping procedure. For easy reference, we provide in Fig. 5.2 a copy of Fig. 4.2, so as to vividly illustrate the mapping scheme under consideration.

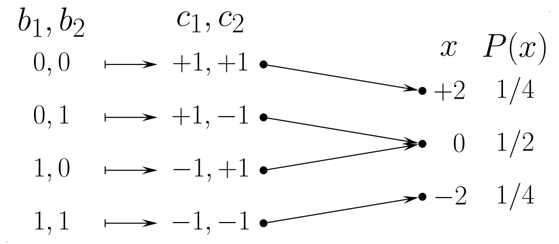


Figure 5.2: Mapping rule of SM-EPA with $N = 2$, $\alpha = 1$.

Substituting (5.6) into (5.7), one obtains the following logical chains:

$$\begin{array}{lcll} x = +2 & \to & P(x|b_1=0) = P(b_2=0)\,, & P(x|b_1=1) = 0 \\ x = \;\;\,0 & \to & P(x|b_1=0) = P(b_2=1)\,, & P(x|b_1=1) = P(b_2=0) \\ x = -2 & \to & P(x|b_1=0) = 0\,, & P(x|b_1=1) = P(b_2=1) \end{array} ,$$

for the three possible values of x, cf. Fig. 5.2. Sequentially, one has

$$\begin{array}{lcll} x = +2 & \to & LLR_e(b_1) = \ln \frac{P(b_2=0)}{0} = +\infty, & \text{if } P(b_2=0) > 0 \\ x = \;\;\,0 & \to & LLR_e(b_1) = \ln \frac{P(b_2=1)}{P(b_2=0)} = -LLR_i(b_2) & \\ x = -2 & \to & LLR_e(b_1) = \ln \frac{0}{P(b_2=1)} = -\infty, & \text{if } P(b_2=1) > 0 \end{array} ,$$

where $LLR_i(b_2) \doteq \ln \frac{P(b_2=0)}{P(b_2=1)}$ denotes the a priori LLR that is intrinsic w.r.t. $LLR_e(b_2)$. Note that in case of $x = 0$, the extrinsic LLR of b_1 will be simply the negative of the intrinsic LLR of b_2. In comparison, for $x = \pm 2$, the extrinsic LLR of b_1 does not have much to do with the a priori distribution of b_2. At the initial iteration, the a priori distribution of b_2 will be uniform, i.e., $P(b_2 = 0) = P(b_2 = 1) = \frac{1}{2}$. In this case, one has for b_1 the following calculation results:

$$\begin{array}{lcl} x = +2 & \to & LLR_e(b_1) = \ln \frac{1/2}{0} = +\infty \\ x = \;\;\,0 & \to & LLR_e(b_1) = \ln \frac{1/2}{1/2} = \;\;\,0 \\ x = -2 & \to & LLR_e(b_1) = \ln \frac{0}{1/2} = -\infty \end{array} .$$

It is easy to imagine that the situation for $LLR_e(b_2)$ at the initial iteration will be exactly the same, due to the symmetry of the mapping rule. Note that the event $x = 0$ has a probability of $\frac{1}{2}$ to occur, which means that at the initial iteration about half of the demapper output LLRs will be zero, or close to zero in case of a noisy channel. Revisiting Fig. 4.3, one may recognize that for SM-EPA with $N > 2$ the situation will be similar. When the symbol magnitude is close to zero, the demapper outputs will be very weak, while for symbol magnitudes close to the maximum, the demapper outputs will be very strong, as long as the additive noise is not so strong. This deserves to be a big difference to the demapping procedure of conventional uniform mapping schemes, for which the demapper outputs do not have such a strong correlation with the observed symbol value.

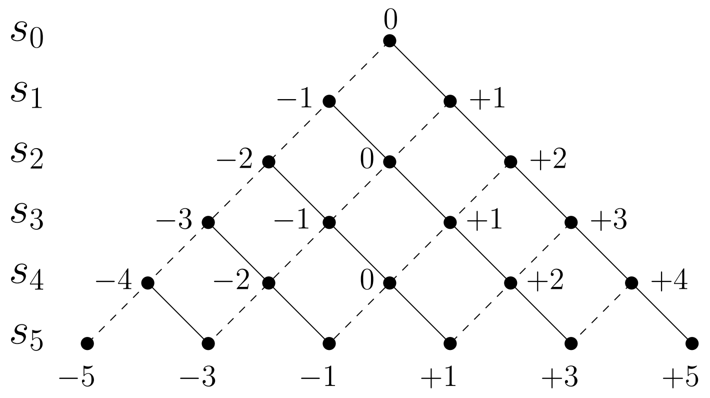


Figure 5.3: A tree diagram showing the growth of SM-EPA symbol alphabet, $\alpha = 1$.

5.2.2 Tree-Based APP Approach

Due to the special symbol formation process of superposition mapping, the task of SISO demapping can in fact be implemented via a tree-based BCJR algorithm. This approach can achieve a dramatic complexity reduction but brings no performance degradation compared to the standard APP approach, whenever the superposition mapper is non-bijective. The idea of using the BCJR algorithm [49] for SISO superposition demapping has first been proposed by Ma and Li Ping in [27], for the scenario of superposition mapping with equal power allocation. In this section, we will generalize this idea to the case of superposition mapping with arbitrary type of power allocation, and show that a non-trivial complexity reduction can be achieved even when the superposition mapper is bijective.

Let us first consider the case of superposition mapping with equal power allocation. Revisiting Tab. 3.2, one will find that the symbol alphabet of SM-EPA grows with the bit load N in an interesting way. The symbol cardinality $|\mathcal{X}|$ is always given by $N + 1$. Vertically stacking the alphabets of linearly increasing bit loads, one obtains something similar to a pyramid[1]. Without loss of generality, one may use a tree diagram to visualize the growing process of the symbol alphabet, as shown in Fig. 5.3. In this tree diagram, each node represents a possible symbol value and each level corresponds to the effect of superimposing one more chip. Since chips are all binary antipodal, there are always two branches emerging from a single node. A solid branch at the nth level corresponds to $c_n = +1$, and a dashed branch corresponds to $c_n = -1$. Due to unified chip magnitudes, a positive branch and a negative branch emerging from two neighboring nodes always merge into a certain node at a new level. This phenomenon is the reason of Gaussian-like symbol distribution, and is also the key for complexity reduction of SISO demapping.

[1]As a matter of fact, if one replaces these symbol values by their frequency of occurrence given independent and uniformly distributed input bits, one obtains exactly a Pascal's triangle.

Since a tree diagram, by treating the levels as the time span, describes a Markov process, the Bahl-Cocke-Jelinek-Raviv (BCJR) algorithm can be used to implement SISO superposition demapping. Let us define the state of the nth level as

$$s_n \doteq \sum_{i=1}^{n} c_i = \sum_{i=1}^{n} \alpha_i (1 - 2b_i) \; . \tag{5.9}$$

Clearly, the relationship between two consecutive states is given by

$$s_n = s_{n-1} + c_n \; , \tag{5.10}$$

which leads to the following conditional probability:

$$P(s_n | s_{n-1}, c_n) = \begin{cases} 1 & \text{for } s_n = s_{n-1} + c_n \\ 0 & \text{for } s_n \neq s_{n-1} + c_n \end{cases} \; . \tag{5.11}$$

As for the BCJR algorithm, the state transition probability is also required for the computation. Following (5.11), the state transition probability of superposition mapping is given by

$$\begin{aligned} P(s_n | s_{n-1}) &= \sum_{c_n} P(s_n, s_{n-1}, c_n) / P(s_{n-1}) \\ &\overset{(a)}{=} \sum_{c_n} P(s_n | s_{n-1}, c_n) P(s_{n-1}) P(c_n) / P(s_{n-1}) \\ &= \sum_{c_n} P(s_n | s_{n-1}, c_n) P(c_n) \\ &= P(c_n = s_n - s_{n-1}) \; , \end{aligned} \tag{5.12}$$

where equality (a) utilizes the fact that s_{n-1} and c_n are independent. Note that

$$P(c_n = s_n - s_{n-1}) = 0 \quad \text{for } s_n - s_{n-1} \neq \pm \alpha_n \; . \tag{5.13}$$

Before one can finally calculate the log-likelihood ratio of each chip, and subsequently the log-likelihood ratio of each bit, there will be two recursive computations necessary to be carried out. The first computation is to obtain the a priori distribution of states, which is determined by the a priori distribution of chips and the conditional probability given in (5.11). Starting from the boundary condition

$$P(s_0 = 0) = 1 \; , \tag{5.14}$$

one evaluates the a priori distribution of states via a forward recursion through the tree:

$$\begin{aligned} P(s_n) &= \sum_{s_{n-1}} \sum_{c_n} P(s_n, s_{n-1}, c_n) \\ &= \sum_{s_{n-1}} \sum_{c_n} P(s_n | s_{n-1}, c_n) P(s_{n-1}) P(c_n) \\ &= \sum_{c_n} P(s_{n-1} = s_n - c_n) P(c_n) \; , \end{aligned} \tag{5.15}$$

where the last equality utilizes the result from (5.11). Since in general we have $|\mathcal{S}_n| > |\mathcal{S}_{n-1}|$, the last expression in (5.15) might contain one or two summands, depending on the value of s_n. The second computation is to determine the likelihood of the received value y given a state s_n. Since $y = x + z = \sum_{i=1}^{N} c_i + z = s_N + z$, we have the following boundary condition:

$$p(y|s_N) = \frac{1}{\sqrt{2\pi\sigma_z^2}} \exp\left(-\frac{(y-s_N)^2}{2\sigma_z^2}\right) . \tag{5.16}$$

Starting from this boundary condition, one can evaluate $p(y|s_n)$ via a backward recursion through the tree:

$$\begin{aligned} p(y|s_n) &= \sum_{s_{n+1}} p(y, s_{n+1}, s_n)/P(s_n) \\ &= \sum_{s_{n+1}} p(y|s_{n+1}, s_n)P(s_{n+1}, s_n)/P(s_n) \\ &= \sum_{s_{n+1}} p(y|s_{n+1})P(s_{n+1}|s_n) , \end{aligned} \tag{5.17}$$

where the last equality follows the fact that the likelihood of y will not be influenced by s_n if s_{n+1} is known. With $P(s_n)$ and $p(y|s_n)$ available for $n = 1, 2, \ldots, N$, the likelihood of the received value y given an arbitrary chip c_n, i.e., $p(y|c_n)$, is right at the hand. Using Bayes' rule and the properties of Markov process, one obtains the following equation chain in a straightforward way:

$$\begin{aligned} p(y|c_n) &= \sum_{s_{n-1}} \sum_{s_n} p(y, s_{n-1}, s_n|c_n) \\ &= \sum_{s_{n-1}} \sum_{s_n} p(y|s_{n-1}, s_n, c_n)P(s_{n-1}, s_n|c_n) \\ &\overset{(a)}{=} \sum_{s_{n-1}} \sum_{s_n} p(y|s_n)P(s_n|s_{n-1}, c_n)P(s_{n-1}|c_n) \\ &\overset{(b)}{=} \sum_{s_{n-1}} \sum_{s_n} p(y|s_n)P(s_n|s_{n-1}, c_n)P(s_{n-1}) \\ &\overset{(c)}{=} \sum_{s_{n-1}} p(y|s_n = s_{n-1} + c_n)P(s_{n-1}) , \end{aligned} \tag{5.18}$$

where equality (a) utilizes the relationship $s_n = s_{n-1} + c_n$ and the philosophy implied by (5.17), that is the likelihood of y will have no dependence on $\{s_0, s_1, \ldots, s_{n-1}\}$ as long as s_n is given. Equality (b) capitalizes on the mutual independence between s_{n-1} and c_n. Finally, equality (c) is a direct application of (5.11). Now, observing the correspondence between the nth bit and the nth chip, the extrinsic LLR of the nth bit is obtained as

$$\begin{aligned} LLR_e(b_n) &= \ln \frac{p(y|b_n = 0)}{p(y|b_n = 1)} \\ &= \ln \frac{p(y|c_n = +\alpha_n)}{p(y|c_n = -\alpha_n)} \\ &= \ln \frac{\sum_{s_{n-1}} p(y|s_n = s_{n-1} + \alpha_n) P(s_{n-1})}{\sum_{s_{n-1}} p(y|s_n = s_{n-1} - \alpha_n) P(s_{n-1})} , \end{aligned} \tag{5.19}$$

which concludes the computation procedure of SISO demapping via the BCJR algorithm.

Computational Complexity for SM-EPA

It is not difficult to find that the computational complexity of the above elaborated algorithm, excluding the calculation of $p(y|s_N)$, is proportional to the amount of branches of the underlying tree diagram. As for the case of SM-EPA, the amount of branches of the tree diagram is generally given by

$$2 \cdot (1 + 2 + \ldots + N) = 2 \cdot \frac{N}{2} \cdot (1 + N) = N(1 + N) \approx N^2 , \tag{5.20}$$

which can clearly be seen from Fig. 5.3. On the other hand, the computational load of obtaining $p(y|s_N)$ is proportional to the number of states of the final level, which is generally given by $1 + N$. For practical systems with finite precision, the computation of (5.16) is often accomplished via a look-up table together with linear interpolation. Hence, we may conclude that the complexity of BCJR demapping for SM-EPA is about quadratic w.r.t. the bit load N. Nevertheless, we should note that this is the overall demapping complexity for the N involved bits. Therefore, we have on average for each bit

$$\text{Demapping complexity of SM-EPA} \quad \propto \quad N^2/N = N , \tag{5.21}$$

which is in fact linear with the bit load. Compared to the standard APP approach, whose complexity per bit is proportional to 2^N, the achieved complexity reduction by using the tree-based approach is significant, particularly when the bit load N is large. For reasonable N values, such a complexity is already acceptable for practical applications, especially when one considers its optimality in performing bit-by-bit SISO demapping. Given an identical system setup, using the standard APP approach or using this tree-based approach will not bring any performance difference.

At this point, a natural question will come up, that is if this tree-based approach is applicable and useful for SM-UPA and SM-GPA as well. As a matter of fact, by checking equations (5.9) to (5.19) once more, one will recognize that they are indeed generally valid for SM with any type of power allocation, given a proper tree diagram. In the next step, we will check the efficiency of this tree-based approach for SM-UPA and SM-GPA.

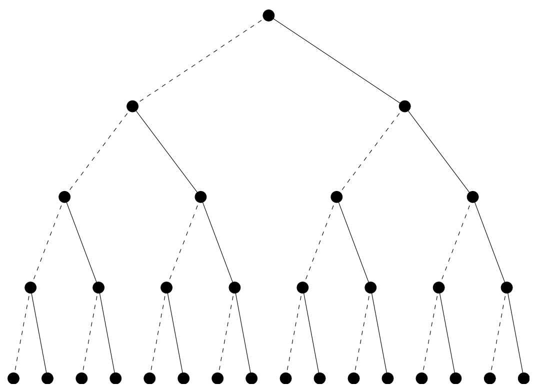


Figure 5.4: A tree diagram showing the growth of SM-UPA symbol alphabet.

Computational Complexity for SM-UPA

Fig. 5.4 illustrates the tree interpretation of the symbol alphabet growth of SM-UPA. Comparing to Fig. 5.3, one finds a big difference. For SM-UPA, the tree branches never merge with each other, and consequently the amount of nodes grows by a factor of two at each new level. This in another way explains the bijectivity of SM-UPA, as the node index can always encode one more bit by adding a new level. Nevertheless, a complexity reduction can still be achieved for SISO demapping by using the BCJR algorithm. From Fig. 5.4, the amount of tree branches for SM-UPA is generally given by

$$2 \cdot (2^0 + 2^1 + \ldots + 2^{N-1}) = 2 \cdot \frac{1-2^N}{1-2} = 2 \cdot (2^N - 1) \approx 2 \cdot 2^N \ . \tag{5.22}$$

Since the final level will have 2^N nodes, the complexity for calculating $p(y|s_N)$ will also be proportional to 2^N. Combining these two observations, we have the average demapping complexity for each bit as

$$\text{Demapping complexity of SM-UPA} \quad \propto \quad 2^N/N \ , \tag{5.23}$$

which is much smaller than 2^N, particularly for a large N. Hence, even for a bijective superposition mapping scheme, using the tree-based BCJR algorithm still offers a non-trivial complexity reduction w.r.t. the standard APP approach.

Computational Complexity for SM-GPA

While the complexity expressions for SM-EPA and SM-UPA are more or less straightforward, it takes some mathematical effort in order to obtain a neat expression for SM-GPA. Fig. 5.5 gives a possible tree diagram for SM-GPA with $G = 2$. This is not the only way to draw the tree for SM-GPA, but it gives the most elegance in the resulting diagram. As

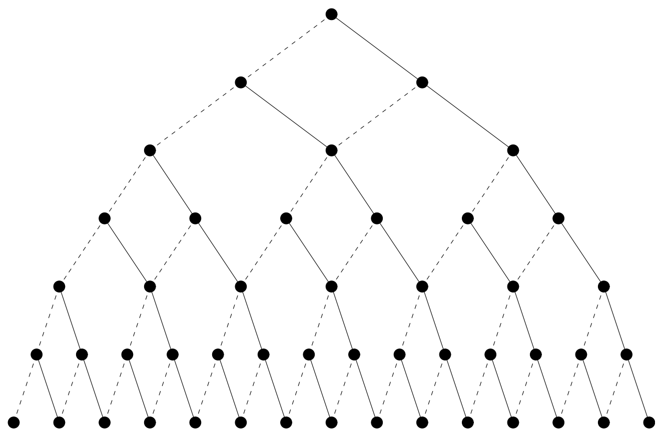


Figure 5.5: A tree diagram showing the growth of SM-GPA symbol alphabet, $G = 2$.

we can see from Fig. 5.5, the branches within each group of levels merge pair by pair as in the case of SM-EPA, while each transition to a new power level always increases the amount of nodes by a factor of 2 as in the case of SM-UPA. Since SM-GPA is a hybrid of SM-EPA and SM-UPA, it is not a surprise that its tree diagram is also a hybrid of that of SM-EPA and SM-UPA. Certainly, the corresponding BCJR demapping complexity will still be proportional to the amount of branches. Since this tree is not as regular as the ones in Fig. 5.3 and Fig. 5.4, its amount of branches has to be counted in a more complicated way. Excluding the root node, the amount of nodes in the first group is given by

$$2 + 3 + \ldots + (G+1) = G(G+3)/2 , \tag{5.24}$$

cf. Fig. 5.5. Following that, the amount of nodes in the second group will be

$$(2G+2) + (2G+3) + \ldots + (3G+1) = G(5G+3)/2 , \tag{5.25}$$

and sequentially the amount of nodes in the third group will be

$$(6G+2) + (6G+3) + \ldots + (7G+1) = G(13G+3)/2 . \tag{5.26}$$

If one continues this procedure for several further groups, one will recognize that in general the amount of nodes in the lth group can be written as

$$G\left((2^{l+1}-3)G+3\right)/2 , \quad l = 1, 2, \ldots, L , \tag{5.27}$$

where L is the number of power levels of the respective SM-GPA scheme. Consequently, the total amount of nodes (excluding the root node) of an SM-GPA tree will be

$$\begin{aligned} \sum_{l=1}^{L} G \cdot \frac{(2^{l+1}-3)G+3}{2} &= \sum_{l=1}^{L} \left\{ 2^l G^2 - \frac{3}{2}G(G-1) \right\} \\ &= 2G^2(2^L - 1) - \frac{3}{2}G(G-1)L . \end{aligned} \tag{5.28}$$

The correctness of the above expression can easily be verified by checking the situation of Fig. 5.5. Checking (5.24) to (5.26) once more, we have the following general expression for the amount of nodes at the final level of the lth group

$$(2^l - 1)G + 1 \,. \tag{5.29}$$

Now, adding the root node but discarding the nodes at the final level of the Lth group, we obtain the amount of branches as

$$\begin{aligned} & 2 \cdot \left(2G^2(2^L - 1) - \frac{3}{2}G(G-1)L + 1 - (2^L - 1)G - 1 \right) \\ = \;& 2G(2G-1)(2^L - 1) - 3G(G-1)L \,. \end{aligned} \tag{5.30}$$

It is also clear from (5.29) that the complexity of computing $p(y|s_N)$ will be about proportional to $(2^L - 1)G$. Summarizing these findings and noting that the bit load of SM-GPA is given by $N = GL$, we acquire the following approximate expression for each bit:

$$\begin{aligned} \text{Demapping complexity of SM-GPA} \;\; & \propto \;\; \big(2G(2G-1)(2^L-1) - 3G(G-1)L\big) \Big/ (GL) \\ & \approx \;\; G(4 \cdot 2^L/L - 3) \quad \text{for large } G \text{ and } L \,, \end{aligned} \tag{5.31}$$

which tells that the complexity is approximately linear w.r.t. the group size G while approximately exponential in the number of power levels L with a mitigation factor of $1/L$. This is again a hybrid of the case of SM-EPA and SM-UPA.

So far, we have calculated the BCJR demapping complexity for SM-EPA, SM-UPA, and SM-GPA. Obviously, this algorithm is also applicable for SM with arbitrary type of power allocation strategies. How much complexity reduction that is achievable depends on the extent of branch merging in the corresponding tree diagram. Last but not least, one may combine the Max-Log APP principle [51,52] with the BCJR demapping algorithm to achieve further complexity reduction. That is to consider only those final-level nodes that are close enough to the channel observation for LLR computation. Doing so can largely reduce the tree size, but will incur a certain degree of performance degradation.

5.2.3 Gaussian-Approximation Approach

When the bit load is not small and the symbol distribution is Gaussian-like, there exists another possibility to reduce the complexity of SISO superposition demapping, but with a non-trivial loss of optimality. Let us reorganize the equation for SM transmission over the Gaussian channel as follows:

$$y = x + z = \sum_{i=1}^{N} c_i + z = c_n + \sum_{i=1, i \neq n}^{N} c_i + z \,. \tag{5.32}$$

Define

$$\eta_n \doteq \sum_{i=1,i\neq n}^{N} c_i + z \tag{5.33}$$

as the effective noise in y w.r.t. c_n. Now, the channel equation can be rewritten as

$$y = c_n + \eta_n \ , \tag{5.34}$$

which leads to the following expression for the likelihood function:

$$p(y|c_n) = p(\eta_n = y - c_n) \ . \tag{5.35}$$

Sequentially, the extrinsic LLR of c_n is now given by

$$LLR_e(c_n) = \ln \frac{p(y|c_n = +\alpha_n)}{p(y|c_n = -\alpha_n)} = \ln \frac{p(\eta_n = y - \alpha_n)}{p(\eta_n = y + \alpha_n)} \ . \tag{5.36}$$

The complexity of the above calculation solely depends on the complexity of $p(\eta_n)$, i.e., the probability density function (PDF) of the effective noise sample, which is an interference-plus-noise mixture. For SM with a large N and a Gaussian-like symbol distribution, one may make the following approximation:

$$p(\eta_n) = \frac{1}{\sqrt{2\pi\sigma_{\eta_n}^2}} \exp\left(-\frac{(\eta_n - \mu_{\eta_n})^2}{2\sigma_{\eta_n}^2}\right) \tag{5.37}$$

with μ_{η_n} and $\sigma_{\eta_n}^2$ being the mean and variance of η_n, respectively. Consequently, (5.36) can now be simplified as

$$LLR_e(c_n) = 2\alpha_n(y - \mu_{\eta_n})/\sigma_{\eta_n}^2 \ . \tag{5.38}$$

By this LLR calculation, the a priori information of chips excluding c_n are implicitly considered via

$$\mu_{\eta_n} = \sum_{i=1,i\neq n}^{N} \mu_{c_i} \ , \qquad \sigma_{\eta_n}^2 = \sum_{i=1,i\neq n}^{N} \sigma_{c_i}^2 + \sigma_z^2 \tag{5.39}$$

and

$$\mu_{c_i} \doteq \mathrm{E}\left\{c_i\right\} = \alpha_i \frac{e^{LLR_i(c_i)} - 1}{e^{LLR_i(c_i)} + 1} \ , \qquad \sigma_{c_i}^2 \doteq \mathrm{E}\left\{(c_i - \mu_{c_i})^2\right\} = \alpha_i^2 - \mu_{c_i}^2 \ , \tag{5.40}$$

where $LLR_i(c_i)$ denotes the intrinsic (a priori) LLR of c_i.

It is not difficult to find that the calculations in (5.38) and (5.40) need to be performed only once for each chip. While for (5.39), if one computes $\sum_{i=1}^{N} \mu_{c_i}$ and $\sum_{i=1}^{N} \sigma_{c_i}^2$ first and derive μ_{η_n} and $\sigma_{\eta_n}^2$ by subtracting μ_{c_n} and $\sigma_{c_n}^2$ from the two sums respectively, the overall complexity of this computation will be linear in N too. Note that $LLR_e(b_n) = LLR_e(c_n)$, i.e., it is equivalent to calculate the LLR for the nth bit or for the nth chip. Therefore, by applying the Gaussian approximation (GA), the overall demapping complexity will be linear w.r.t. the bit load N, and the demapping complexity per bit will in fact be constant w.r.t. the bit load N, which is very attractive for practical applications.

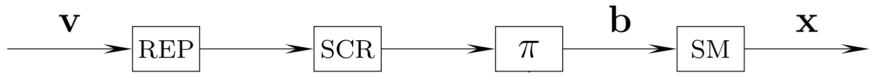


Figure 5.6: Transmitter structure of repetition-coded superposition mapping.

5.3 Repetition-Coded SM

In the framework of BICM, two types of channel codes, repetition codes and parity-check codes, are most frequently adopted due to the ability of their decoders in providing bit-level reliability information. For the case of coded SM transmission, a soft-input soft-output channel decoder is often essential in achieving a desirable performance, particularly when the superposition mapper is non-bijective. Therefore, in this thesis we will also take these two types of codes as building blocks for coded SM systems. Repetition coding is known for its simplicity in encoding and decoding, but is also known as an inefficient coding strategy because of offering no coding gain at all over the AWGN channel. In contrast, parity-check codes often come with a sophisticated encoding and decoding procedure, and can offer a strong coding gain over the AWGN channel if well designed. For this reason, researchers are typically discouraged in using repetition codes, especially when the system under design is aiming to achieve the channel capacity. This way of thinking is proper, however, only with the assumption of using bijective uniform mapping schemes, such as ASK. As for the case of superposition mapping, the situation is in fact substantially different. The simulation results in this section as well as in Section 5.4 will show that a simple repetition code can easily outperform a strong parity-check code, in the sense of offering higher supportable bandwidth efficiency for coded SM transmission.

Fig. 5.6 illustrates the transmitter structure of repetition-coded SM transmission. One may note that there is a scrambler in between the repetition encoder and the interleaver, which is indeed not so common for BICM systems. Certainly, a scrambler does not have any influence on the system data rate, but it does have a big influence on the performance of such a system. By its nature, coded SM transmission shares much similarity with interleave-division multiplexing (IDM) [53] or interleave-division multiple access (IDMA) [54]. In IDM/IDMA systems, the inner code for each data stream is often a repetition code, and each repetition encoder is typically followed by a scrambler as well as an interleaver, which looks very similar to that in Fig. 5.6. This type of system setup does not come occasionally, but actually has good theoretical and practical reasons. We will later show in Chapter 6 that scrambling and interleaving are both indispensable for repetition-coded SM transmission, as long as the adopted power allocation strategy leads to a non-bijective mapping rule. Here in this section our aim is to have a brief overview on the corresponding system as well as its performance.

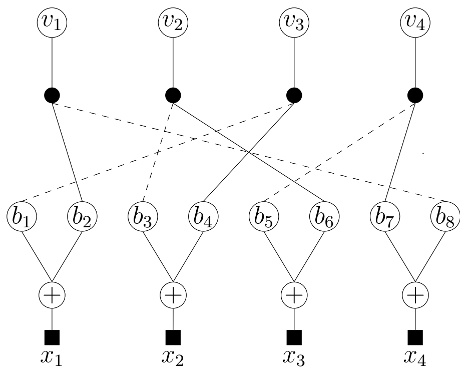


Figure 5.7: A detailed factor graph for repetition-coded SM, $SF = 2$, $N = 2$, $K = 4$.

5.3.1 Factor Graph Representation

In order to illustrate a coded transmission scheme, a global-level factor graph is always useful. A factor graph is also useful for interleaver design and iterative receiver design. For repetition-coded SM as depicted in Fig. 5.6, a detailed factor graph representation can be drawn as in Fig. 5.7, where v_i denotes an input bit to the repetition encoder, and b_i denotes an input bit to the superposition mapper, keeping consistence with the labels in Fig. 5.6. To make the figure easily distinguishable, the system parameters have been chosen to be relatively small. From top to bottom, this graph vividly describes the information flow process from info bits to SM symbols. First, each info bit is repeated into two code bits, one of which gets flipped afterwards, denoted by a dashed edge in the graph. Without loss of generality, we may call a node corresponding to repetition an equality check. For example, the leftmost filled circle "•" states the following constraint:

$$v_1 \quad = \quad b_2 \quad = \quad 1 - b_8 \ ,$$

where the last equality comprises the effects of repetition as well as scrambling. We will show in Chapter 6 that this simple scrambling scheme that flips every second code bit in fact brings a big benefit for the system stability. In Fig. 5.7, each circled plus "$\oplus$" represents a superposition mapping operation. By the convention of factor graphs, we may term such a node as a summation check. For example, assuming SM-EPA with $\alpha = 1$, the leftmost "$\oplus$" would impose the relationship

$$x_1 = B(b_1) + B(b_2) \ , \tag{5.41}$$

where $B(b_i) \doteq 1 - 2b_i$ marks a BPSK mapping operation. Since SM symbols will be transmitted over the channel, each of them will come with a channel observation, which is emphasized in Fig. 5.7 by black squares. From now on, whenever a ■ appears in a factor graph, we mean by it a channel observation node.

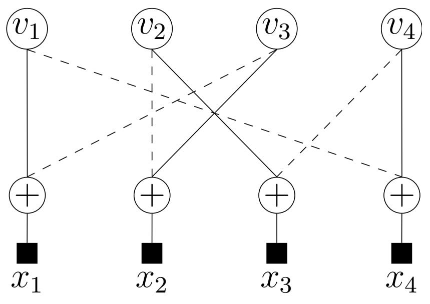


Figure 5.8: A simplified factor graph for repetition-coded SM, $SF = 2$, $N = 2$, $K = 4$.

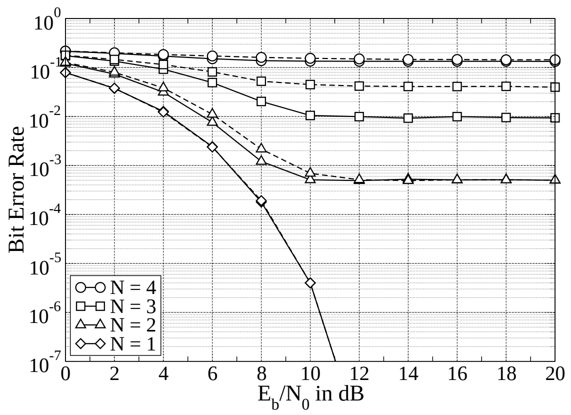
The graph discussed so far is very detailed. While it is nice for a conceptual description, it is in fact not so convenient for practical use. Next, we introduce a simplified but equivalent graph representation for repetition-coded SM. Checking Fig. 5.7 once more, one may build up a direct relationship between an SM symbol and a corresponding pair of info bits. Still assuming SM-EPA with $\alpha = 1$, we may write these relationships as

$$\begin{aligned} x_1 &= B(b_1) + B(b_2) &&= B(1 - v_3) + B(v_1) \\ x_2 &= B(b_3) + B(b_4) &&= B(1 - v_2) + B(v_3) \\ x_3 &= B(b_5) + B(b_6) &&= B(1 - v_4) + B(v_2) \\ x_4 &= B(b_7) + B(b_8) &&= B(v_4) + B(1 - v_1) \,. \end{aligned}$$

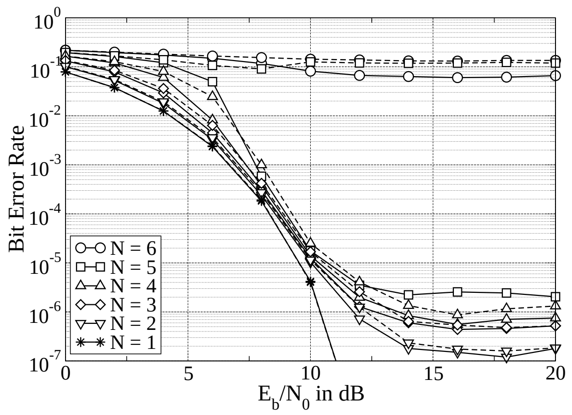
Now, eliminating all the intermediate variable nodes and the equality checks, we obtain a compact factor graph directly emphasizing the connections between info bits and SM symbols, as demonstrated in Fig. 5.8. Though containing much less nodes compared to the one given in Fig. 5.7, this graph loses no functionality for system analysis as well as for code design. The ultimate goal of a receiver is to make correct decisions for the info bits. As the graph in Fig. 5.8 includes all the connections between info bits and channel observations, it covers the complete structural information of the transmission system in an efficient way. Later on for the discussion of repetition-coded SM, we will exclusively utilize this simplified way of graph representation. One will see in Chapter 6 and Chapter 7 that this type of graph is specifically convenient for the sake of interleaver optimization and global-level code optimization, respectively.

5.3.2 Performance Overview

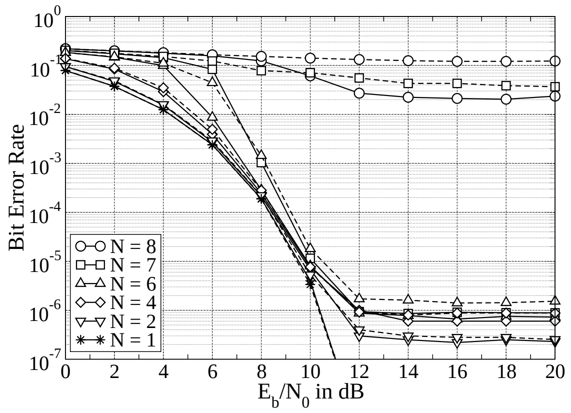
Both Chapter 3 and Chapter 4 show that the property of superposition mapping is essentially determined by the adopted power allocation strategy. Naturally, for the three introduced power allocation strategies, EPA, UPA, and GPA, one expects different coded performance from them as well.



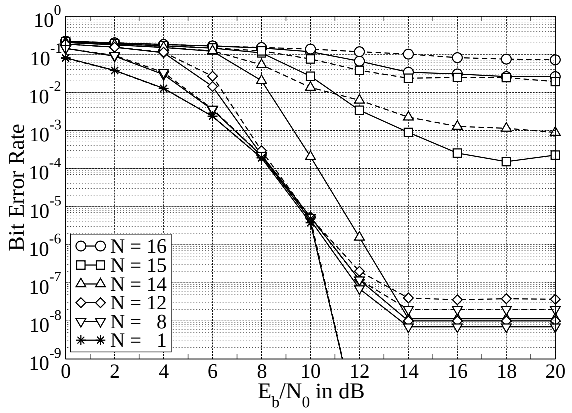
(a) $SF = 2$.



(b) $SF = 3$.



(c) $SF = 4$.



(d) $SF = 8$.

Figure 5.9: Repetition-coded SM-EPA, every second code bit flipped, random interleaving, 1000 symbols per block (999 for $SF = 3$), 20 iterations. Solid lines correspond to BCJR demapping and dashed lines correspond to Gaussian demapping.

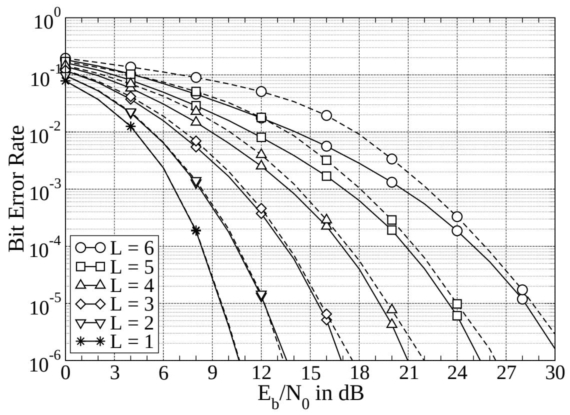
Equal Power Allocation

Fig. 5.9 provides four sets of Monte Carlo simulation results for repetition-coded SM-EPA with scrambling and random interleaving. For each system setup, the performance test has been done with BCJR demapping and Gaussian demapping, respectively. Let us first focus on the cases with BCJR demapping. These four sets of results show a critical issue for coded SM-EPA transmission, that is, the supportable effective bit load (EBL) N/SF is very limited, given the current system configuration. Particularly, for $SF = 2$, the supportable EBL is merely $1/2 = 0.5$ bits/symbol. For $SF \geqslant 3$, the supportable EBL is about 1.7 bits/symbol, up to an acceptable error floor level. Nevertheless, by comparing these results with Tab. 3.3, one recognizes that this supportable EBL limit of less than 2 bits/symbol does not come from the mapping scheme. For example, with $N = 16$, the theoretical supportable EBL is given by $H(x) \approx 3.0465$ bits/symbol, which is significantly higher than that achieved in Fig. 5.9(d). Since the BCJR demapping algorithm is optimal in the sense of delivering accurate extrinsic messages, the imperfectness of the current system should come from other places, including the channel encoder, the scrambler, and the interleaver. This conjecture gives the motivation for the work in Chapter 6.

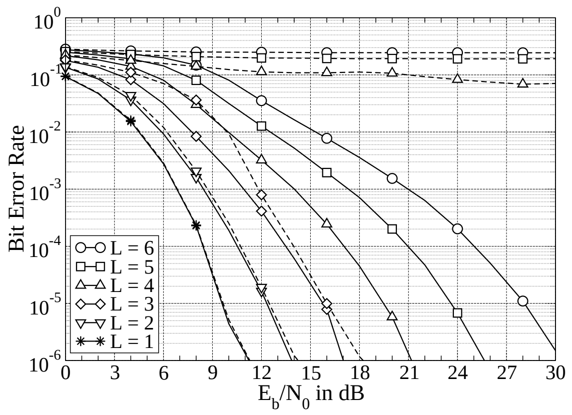
Compared to the case of BCJR demapping, using a Gaussian demapper brings a certain degree of performance degradation for repetition-coded SM-EPA, depending on the system setup. Checking through Fig. 5.9(a) to Fig. 5.9(d), one may find that for $N/SF \leqslant 1.5$ the performance loss due to Gaussian demapping is mostly not significant, except for $SF = 2$. On the other hand, whenever the EBL goes beyond 1.5 bits/symbol, the performance loss becomes very significant, e.g., for $N/SF = 7/4$ and $N/SF = 14/8$. Hence, with Gaussian demapping, the supportable EBL is about 1.5 bits/symbol, given the current system configuration. Similar results are also reported in [38] in the framework of interleave-division multiple access (IDMA). It is also shown in [38] that with a very large spreading factor the performance loss due to Gaussian demapping becomes marginal.

Unequal Power Allocation

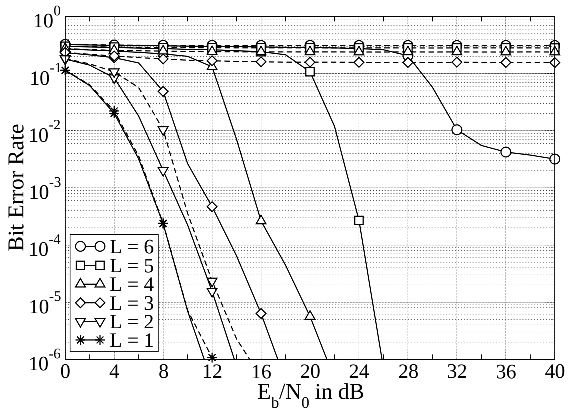
Since SM-UPA is equivalent to SM-GPA with group size $G = 1$, we do not make an individual performance test for it. Instead we may safely utilize the results from Fig. 5.10(a), by a simple parameter conversion: $N = GL = L$. Given unequal power allocation, superposition mapping becomes uniform and bijective. Therefore, the theoretical supportable effective bit load is always given by $H(x) = N$ bits/symbol. Consequently, the iterative receiver will not encounter any problem for convergence no matter how large the bit load N is. This is the advantage of using a bijective uniform mapping scheme. However, one should keep in mind that SM-UPA can not be capacity-achieving due to its non-Gaussian-like symbol distribution. Besides, due to unequal power allocation, code bits are unequally protected. With a rate 1/4 repetition code, which is the case in Fig. 5.10(a), this unequal error protection effect can be partially mitigated if the number of power levels is smaller than 4, as a random interleaver will not always assign the replicas from a single info bit to chips with mutually different power levels. From Fig. 5.10(a), one sees that increasing the bit load N by 1 the BER curve is shifted to the right by about 3 dB, when $N < 4$. When $N \geqslant 4$, increasing the bit load by 1 will bring a performance degradation more than 3 dB, and this loss will asymptotically reach 6 dB as one increases the bit load further. For SM-UPA, a chip of the $(n + 1)$th level will have a power only quarter of that of the nth level, which corresponds to a 6 dB performace drop. After all, an interesting issue in Fig. 5.10(a) is that the performance degradation due to Gaussian demapping is not as large as one would have expected. Though this very-low-complexity demapping algorithm has originally been derived for SM with a Gaussian-like symbol distribution, it actually offers an acceptable performance for repetition-coded SM-UPA.



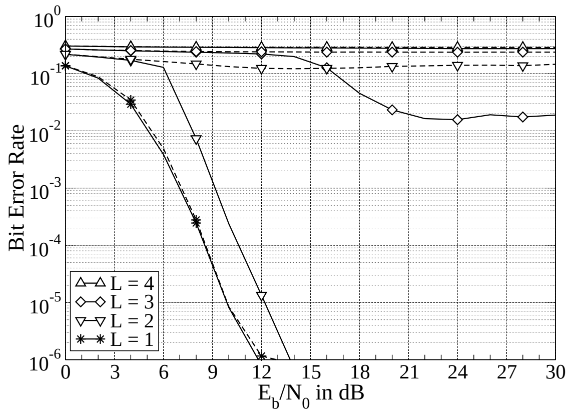
(a) $G = 1$.



(b) $G = 2$.



(c) $G = 3$.



(d) $G = 4$.

Figure 5.10: Repetition-coded SM-GPA, $SF = 4$, every second code bit flipped, random interleaving, 1000 symbols per block, 20 iterations. Solid lines correspond to BCJR demapping and dashed lines correspond to Gaussian demapping.

Grouped Power Allocation

Compared to equal power allocation and unequal power allocation, which are in fact two extremes among all the possibilities, grouped power allocation has a more or less hybrid type of property. Given a suitable group size, SM-GPA can deliver a Gaussian-like symbol distribution as well, but its symbol entropy has a more linear relationship to the bit load, compared to SM-EPA. According to the approximation given in (3.33), the theoretical lower limit for the spreading factor ($SF = 1/R$) is given by

$$\frac{N}{H(x)} \approx \frac{GL}{\frac{1}{2}\log_2(\frac{\pi}{6}eG) + L} < G \, . \tag{5.42}$$

For very large L, we have $N/H(x) \approx G$. Hence, for an ideal channel code, choosing $SF = G$ should already enable error free detection for all L. Now, let us check the realistic performance of repetition-coded SM-GPA, provided in Fig. 5.10. With BCJR demapping, one basically sees no limit on the supportable bit load for $G = 1$ and $G = 2$. While the situation of $G = 1$ is more or less clear, the simulation of $G = 2$ is in fact very meaningful. Given $L = 6$, the receiver still converges well. For this setup, the achieved throughput is $G \cdot L/SF = 2 \cdot 6/4 = 3$ bits/symbol, which is clearly higher than that achieved with SM-EPA (cf. Fig. 5.9). Referring to Fig. 3.11, SM-GPA with $G = 2$ is non-bijective and the corresponding mutual information performance is almost capacity-achieving. Hence, using a grouped power allocation strategy brings benefits for a practical iterative receiver, and more importantly, doing so will not degrade the theoretical optimality of superposition mapping for transmission over the Gaussian channel. Nevertheless, for $G = 3$, we do see a limit on the supportable bit load, which is about $3 \cdot 5/4 = 3.75$ bits/symbol, and for $G = 4$ the limit is even lower at $4 \cdot 2/4 = 2$ bits/symbol. This observation tells that given a fixed spreading factor, increasing the group size G will reduce the supportable bandwidth efficiency of (regular) repetition-coded SM-GPA. On the other hand, (5.42) tells that such a situation should not happen if the channel code is optimal, as for all the simulations in Fig. 5.10 the condition $SF \geqslant G$ is fulfilled. Therefore, (regular) repetition coding is not optimal for SM-GPA, which is of no surprise.

Last but not least, one sees from the dashed curves in Fig. 5.10 that the performance of Gaussian demapping is rather undesirable for $G \geqslant 2$, in the sense of supporting much limited bit load compared to BCJR demapping. If one checks these curves more carefully, one will find that the problem of Gaussian demapping is in the capability of convergence. Whenever it converges, its performance will not be far away from that of BCJR demapping. However, if it does not converge, the corresponding BER curve will be more or less flat. By taking a more suitable channel code, the convergence capability of Gaussian demapping can be improved for SM-GPA, and so is the supportable bit load. Nevertheless, the performance loss w.r.t. BCJR demapping will still be non-trivial. In general, to guarantee the convergence of an iterative receiver which utilizes a certain form of Gaussian approximation, a large spreading factor is necessary [38, 42, 54]. This is not a problem for large-scale systems such as IDMA systems, where a large spreading factor is typically adopted. However, for superposition mapping, a larger spreading factor enforces a higher bit load, if a certain bandwidth efficiency is to be achieved. This leads to a higher computational complexity, which is certainly undesirable for practical applications. From an engineering standpoint, finding a new superposition demapping algorithm that achieves a better performance than the Gaussian demapper and a lower complexity than the tree-based BCJR demapper deserves to be an interesting topic. Since the focus of this thesis is on finding good channel codes that can well exploit the capacity-achieving potential of superposition mapping, issues related to Gaussian demapping will not be covered in more details. From now on, the tree-based BCJR demapping algorithm will always be assumed for the discussion.

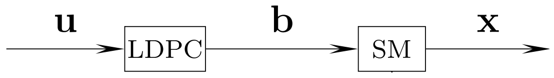


Figure 5.11: Transmitter structure of LDPC-coded superposition mapping.

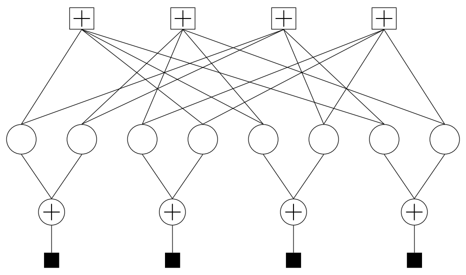


Figure 5.12: A factor graph for LDPC-coded SM, $R = 1/2$, $N = 2$, $K = 4$.

5.4 Parity-Check-Coded SM

In the past two decades, the technological advance in the field of parity-check coding is marvelous. For a binary-input Gaussian channel, the modern random type of parity-check codes can deliver a performance with a negligible distance to the channel capacity. In this section, we will have a brief survey on parity-check-coded SM transmission over the Gaussian channel. As a matter of fact, all well-known powerful parity-check codes, including Turbo codes [4] and repeat-accumulate codes [55], can be interpreted as certain type of low-density parity-check (LDPC) codes. Therefore, for the current discussion, we risk no loss of generality by referring to LDPC codes only.

By means of LDPC coding, interleaving comes as a built-in operation from the channel encoder. This makes an external interleaver no longer necessary. Hence, one may build an LDPC-coded SM transmission system as in Fig. 5.11. Note that we have not placed a scrambler in between the LDPC encoder and the superposition mapper. The reason why we do so becomes clear when the effects of scrambling on repetition-coded SM are discovered in Chapter 6.

5.4.1 Factor Graph Representation

Given the transmitter structure in Fig. 5.11, the corresponding factor graph representation can be plotted as in Fig. 5.12, where each $\boxplus$ stands for a parity check and each $\bigcirc$ stands for a variable node (code bit). Summation checks and channel observations are marked in the same way as in Fig. 5.7. Depending on the specific code design, each parity check will be connected with a certain amount of variable nodes and each variable node will be connected with a certain amount of parity checks. For repetition-coded SM, each variable node is connected with multiple summation checks, cf. Fig. 5.8. Now for LDPC-coded SM, each variable node is connected with a single summation check but with multiple parity checks. In Chapter 7 we will show that there are indeed many commonalities between repetition-coded SM and LDPC coding.

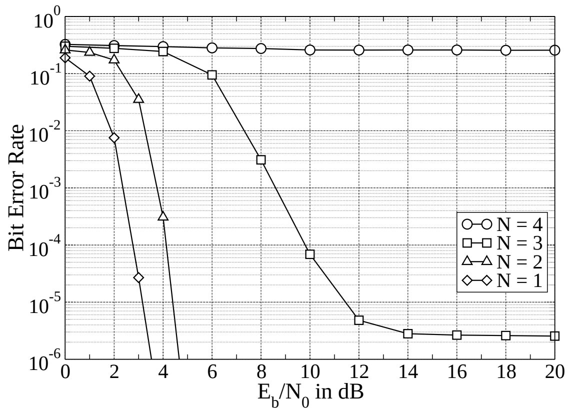
5.4.2 Performance Overview

From Section 5.3.2 we have seen that a regular repetition code works well with superposition mapping, but is not optimal. In a conventional way of thinking, this is of no surprise, because repetition coding is rarely connected to the concept of optimal channel coding. Again in a conventional way of thinking, one expects a significant performance improvement by replacing a zero-gain repetition code by a powerful LDPC code. Nevertheless, the simulation results provided below will show that the true situation is not as simple as one would have expected.

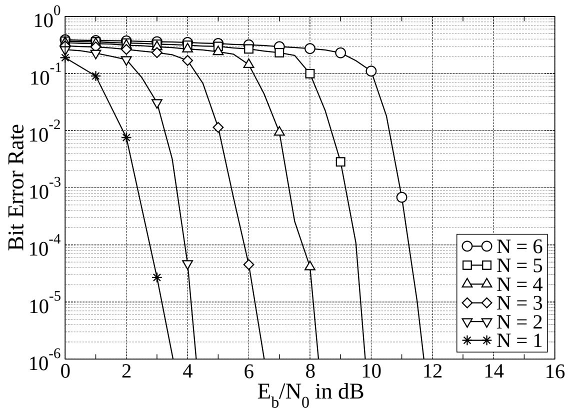
Equal Power Allocation

Fig. 5.13(a) demonstrates the performance of LDPC-coded SM-EPA. For the sake of clearness, we have adopted rate 1/4 regular LDPC codes. The corresponding parity-check matrices always have a unique column weight 3 and a unique row weight 4. The code word length is proportional to the SM bit load, i.e., $1000 \times N$, so that the symbol block length is constantly given by 1000. In order to make a fair comparison with the tests in Section 5.3.2, the number of global receiver iterations is set to be 20. Within each global iteration, no extra LDPC-local iterations are performed[2]. Clearly, given the same number of iterations, the overall receiver computational load of LDPC-coded SM will be higher than that of repetition-coded SM, cf. Fig. 5.12 and Fig. 5.8. Nevertheless, the resulting performances are not necessarily better than that of repetition-coded SM. Comparing Fig. 5.13(a) with Fig. 5.9(c), one observes the following phenomena. First, for $N = 1$, which makes SM-EPA equivalent to BPSK mapping, LDPC coding significantly gains w.r.t. repetition coding. This is because parity-check codes accomplish information spreading in a more efficient way than repetition codes, for the binary-input AWGN channel. Second, for $N = 2$, LDPC coding still gains w.r.t. repetition coding. At this point, the superposition mapper is already non-bijective, but it seems to make no big problem for the LDPC decoder. Third, at $N = 3$, a non-trivial error floor appears and the BER performance is even worse than that of repetition-coded SM-EPA with $N = 4$.

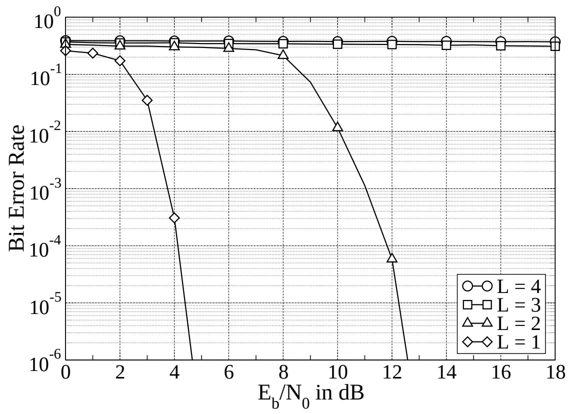
[2]The necessity of LDPC-local iterations will be clarified by the discussion in Chapter 7.



(a) Equal power allocation.



(b) Unequal power allocation.



(c) Grouped power allocation, $G = 2$.

Figure 5.13: LDPC-coded SM, $R = 1/4$, 1000 symbols per block, 20 global iterations, BCJR demapping. The LDPC codes are all $(3, 4)$-regular. Within each global iteration, no extra LDPC-local iterations are performed.

Finally, at $N = 4$, the receiver does not converge at all. Given these observations, one may conclude that regular LDPC-coded SM-EPA only works well for very small bit loads. For example, given $R = 1/4$, one should choose $N \leqslant 2$, which corresponds to a bandwidth efficiency $\leqslant 0.5$ bits/symbol. For SM-EPA with $N \geqslant 3$, regular LDPC coding is not superior but inferior to regular repetition coding.

Unequal Power Allocation

Given unequal power allocation, a superposition mapper will be uniform and bijective. Consequently, a regular LDPC code works well in this case. By checking the SNR range of Fig. 5.13(b) and Fig. 5.10(a), the advantage of LDPC coding over repetition coding is evident for SM-UPA. When one increases the bit load, the BER curve of LDPC-coded SM-UPA shifts rightwards, which is caused by the decreasing minimum symbol distance together with the unequal error protection effect caused by unequal power allocation. After all, we should keep in mind that this type of system setup does not have a potential to approach the Gaussian channel capacity because of the uniform symbol distribution. An interesting comparison may make this issue more clear. Checking the performance of repetition-coded SM-EPA with $SF = 4$ and $N = 6$ in Fig. 5.9(c) and comparing it with the performance of LDPC-coded SM-UPA with $R = 1/4$ and $N = 6$ in Fig. 5.13(b), one may find that the latter is even worse, for BER $\geqslant 10^{-5}$. Clearly, these two system setups give exactly the same bandwidth efficiency. Though by no means the performance under consideration is the best that LDPC-coded modulation can achieve, it does reflect an important principle. Regardless that LDPC codes are excellent for data transmission in the power-limited regime, its advantage over repetition codes in the bandwidth-limited regime is not as much as one would expect from a conventional way of thinking.

Grouped Power Allocation

Given rate 1/4 regular LDPC-coded SM-GPA with $G = 2$, the maximum supportable power level number is $L = 2$, as shown in Fig. 5.13(c). This corresponds to a bandwidth efficiency of $R \cdot G \cdot L = 1$ bit/symbol, which is slightly improved w.r.t. to the case of LDPC-coded SM-EPA. Nevertheless, the resulting power efficiency is in fact undesirable. Comparing the BER curve for $L = 2$ in Fig. 5.13(c) to the BER curve for $L = 2$ in Fig. 5.10(b), one observes that the performance of LDPC-coded SM-GPA is inferior to that of repetition-coded SM-GPA, for BER $\geqslant 10^{-5}$. Combining the observations from Fig. 5.13(a) and Fig. 5.13(c), we may conclude that regular parity-check codes are not suitable for non-bijective nonuniform superposition mapping, whenever the bit load is large. The application of irregular LDPC codes for SM will be discussed in Chapter 7.

Chapter 6

Spreading, Scrambling, and Interleaving

In Chapter 5, the performance of superposition mapping with conventional coding schemes has been surveyed. Surprisingly, repetition coding is advantageous over parity-check coding, particularly in the sense of supportable bandwidth efficiency. Hence, classical capacity-achieving channel codes, which are exclusively designed for bijective uniform mapping, do not work well for non-bijective nonuniform superposition mapping. This observation reveals that designing suitable channel codes for superposition mapping is not as straightforward as one may have expected. Considering the superior performance of repetition-coded SM, it is worthwhile to conduct a careful investigation on this type of system structure. Intuitively, one expects valuable hints for a more complicated code design from this study. This chapter tries to provide a deep insight into the working mechanism of repetition-coded SM via a thorough survey on its three important aspects: spreading, scrambling, and interleaving. As exhibited in Chapter 5, equal power allocation presents the biggest challenge w.r.t. code design. Therefore, the discussion in this chapter will exclusively be focused on SM-EPA. Code design for SM-UPA and SM-GPA will be treated in Chapter 7. It will be shown that spreading, scrambling, and interleaving are indivisible operations for SM-EPA, i.e., all three of them are essential for the system performance and their effects are highly interactive. The way of spreading significantly influences the supportable bandwidth efficiency of SM-EPA, while the quality of interleaving primarily determines the level of error floor. In comparison, scrambling influences the performance in a relatively indirect way, in the sense that its main effect is in improving the stability of iterative detection. When matched to interleaving, scrambling can also help to reduce the error floor level. To achieve the best possible performance, the spreading scheme, the scrambling pattern, and the interleaving pattern, have to be carefully designed.

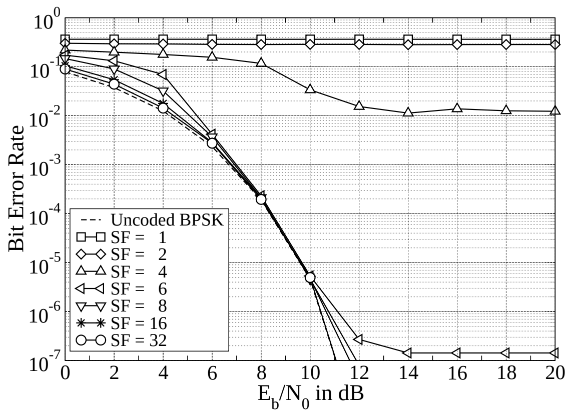


Figure 6.1: Repetition-coded SM-EPA, every second code bit flipped, random interleaving, $N = 8$, 1000 symbols per block (999 for $SF = 6$), 100 iterations.

6.1 Effects of Spreading

According to Chapter 5, a large spreading factor is always beneficial for repetition-coded SM-EPA with iterative detection. Here we provide a more systematic view on this issue. For this discussion, let us still use the simple scrambling scheme that flips every second code bit and let us stay with random interleaving.

6.1.1 Regular Repetition

Fig. 6.1 shows the performance of repetition-coded SM-EPA with bit load $N = 8$ and various spreading factors. To eliminate the influence from insufficient iterations, we always make 100 receiver iterations, regardless of the particular value of the spreading factor. The performance of random interleaving improves with the block length. Hence, to make the comparison fair, we fix the number of symbols per burst, which in turn fixes the number of code bits and consequently the interleaver length. At $SF = 1$, there is no spreading at all. Therefore, a significant error floor is present. As explained by the discussion in Chapter 4, this error floor level will raise monotonically with the bit load N, as long as no spreading is applied before superposition mapping. Increasing the spreading factor to $SF = 2$, the error floor level reduces a little bit but is still significant. This is also easy to understand by checking Tab. 3.3. The compression rate of SM-EPA with $N = 8$ is about 0.3180, which in turn requires a spreading factor larger than or equal to $1/0.3180 \approx 3.1447$. For any spreading factor below this limit, error-free transmission is strictly prohibitive, because one is eventually loading more information than each SM-EPA symbol can carry. By increasing the spreading factor to $SF = 4$, which is already above the theoretical limit, the error floor drops noticeably but is still at a high level. This indicates that the current system configuration is far from being optimal. At the moment, it is not clear whether the imperfectness comes from the spreader, the scrambler, the interleaver, or simply the block length. There is also a chance that these four issues all contribute to the non-optimal performance. We will clarify the situation step-by-step in later discussions. With $SF = 6$, the error floor drops to a level around 10^{-7}, and the BER performance improves further by taking even larger spreading factors. There is an ultimate bound for the BER performance under concern. Due to equal power allocation and regular repetition coding, all info bits are equally protected. Together with the fact that repetition codes provide no coding gain, one should expect no better power efficiency from such a system than uncoded BPSK. Nevertheless, in case of $SF < N$ the achieved bandwidth efficiency will be higher than that of uncoded BPSK.

6.1.2 Irregular Repetition

In the field of LDPC coding, it is a common knowledge that irregular LDPC codes can often provide a performance closer to the capacity than regular ones. By irregular LDPC coding, the code bits are unequally protected, either virtually or truly. During iterative decoding, the strongly protected code bits get correctly estimated first, and then the knowledge of these bits will help the estimation of those weakly protected bits. Given a properly designed degree distribution, an irregular LDPC code can offer a significant improvement w.r.t. the decoding threshold [56–58]. Naturally, in the scenario of repetition-coded SM transmission, one may wonder if an irregular repetition code will also help.

For SM-EPA, using an irregular repetition code creates another type of unequal power allocation. Different info bits are spreaded into a different amount of binary antipodal chips, while these chips all have identical power due to EPA. Consequently, we can no longer use the performance of uncoded BPSK as a bound for the best achievable power efficiency. Instead, for an irregular repetition-coded SM-EPA transmission scheme, the best achievable power efficiency is given by the performance of this scheme at $N = 1$, i.e., when SM-EPA is equivalent to BPSK mapping. That is to reserve the unequal error protection effect from irregular repetition but to eliminate the effect of inter-chip interference due to linear superposition.

From Fig. 6.1 we observe that given the system setup therein a regular repetition code with spreading factor $SF = 4$ cannot bring a good convergence for the iterative receiver. Now, we try two irregular repetition codes both having an average spreading factor of 4. The first code applies degree-3 repetition for 50% of the info bits and degree-5 repetition for the remaining. The second code applies degree-3 repetition for 80% of the info bits and degree-8 repetition for the remaining. Fig. 6.2 illustrates the performance of SM-EPA with

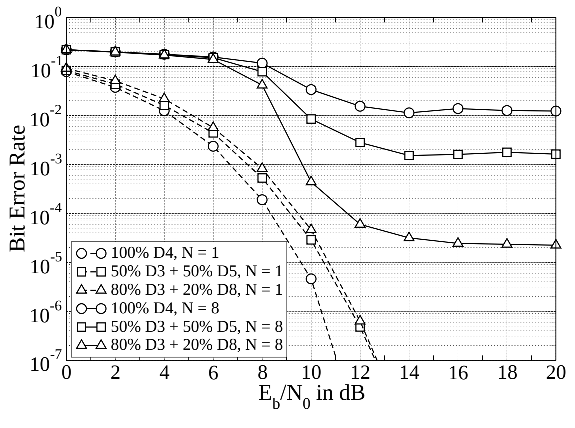


Figure 6.2: Repetition-coded SM-EPA, every second code bit flipped, random interleaving, $N = 8$, 1000 symbols per block, 100 iterations. “D” stands for the degree of repetition.

these two irregular codes as well as the regular $SF = 4$ repetition code. The BER curves corresponding to $N = 1$ are provided as reference curves. Clearly, the irregularity in the repetition code is helpful for the iterative receiver. By introducing degree-5 repetitions, the error floor level is reduced from 10^{-2} to 10^{-3}, and by introducing degree-8 repetitions, the error floor level drops below 10^{-4}. Empirically, an error floor below 10^{-4} is usually caused by short cycles in the corresponding factor graph. We will address this issue later in Section 6.4. Analogous to the mechanism of irregular LDPC coding, the info bits with a large spreading factor get correctly estimated first, and afterwards, the knowledge of these bits helps the estimation of remaining bits with a relatively small spreading factor. In this way, irregular repetition helps the iterative receiver to jump out from many local optima and consequently helps it to converge to the global optimum, if the corresponding factor graph has a nice structure. Another important fact to be noted is the effect of irregular repetition on the decoding threshold. As shown by Fig. 6.2, the distance from the BER curve for $N = 8$ to the BER curve for $N = 1$ decreases as one increases the maximum repetition degree. This means that with the current system setup, the iterative receiver is working more and more close to the optimal level that it can achieve in principle, when one enlarges the irregularity of the repetition code.

6.1.3 Information Aggregation and Information Distribution

By the investigation so far, we have seen that repetition-coded SM-EPA works well and more importantly there is clearly space for further improvements. On the other hand, if we have a quick review on the discussions in Section 5.2.1, we will find that in the initial iteration the LLR messages outputting from an SM-EPA demapper should mostly be very weak, especially when the bit load N is large. As a natural question, one may wonder how the iterative receiver of a repetition-coded SM-EPA system manages to deliver reliable decisions at the end. To clarify this issue, we consider in the following a simple example which visualizes the working procedure of iterative superposition demapping and repetition decoding. This study will also serve as a technical background for the discussion in Section 6.2 and Section 6.3.

Suppose we have a repetition-coded SM-EPA system that transmits 57 info bits per block. The bit load of the superposition mapper is given by $N = 3$. The adopted repetition code is irregular. It repeats one and only one info bit by a factor of 4. It repeats 24 info bits by a factor of 2, and it does not repeat the remaining 32 info bits. Let us denote the info bit with degree-4 repetition by v and focus on the information aggregation and information distribution process on the corresponding variable node. With a good interleaver pattern, one may obtain a graph as shown in Fig. 6.3. Note that this graph is completely cycle-free. Hence, one may also call it a tree diagram, with the node v being the root. Not difficult to find, the amount of summation checks that are reached from v by the 1st iteration is given by 4. This amount increases to $4 + 4 \cdot 2$ by the 2nd iteration, and $4 + 4 \cdot 2 + 4 \cdot 2 \cdot 2$ by the 3rd iteration. Sequentially, if one grows the tree further in this style, by the nth iteration the total amount of reached summation checks from v is given by

$$4 + 4 \cdot 2 + 4 \cdot 2 \cdot 2 + \ldots + 4 \cdot 2^{n-1} \;=\; 4 \cdot (2^n - 1) \;, \tag{6.1}$$

which raises exponentially with the iteration number. Due to iterative message passing, the information contained by these summation checks will all contribute to the estimation of variable v via Bayesian inference. As a result, the estimation of v becomes more and more reliable from iteration to iteration. Easy to imagine, by using a larger spreading factor, the amount of reachable checks will increase even faster. Therefore, for a fixed bit load, the required number of iterations for good convergence should reduce with the spreading factor. On the other hand, a larger bit load typically imposes more iterations.

To have a deeper insight into the iterative message passing process of repetition-coded SM-EPA, let us consider some numerical samples. In case of noiseless transmission, the LLR messages outputting from an $N = 3$ SM-EPA demapper in the initial iteration will have four possible values as shown in the table below, cf. Fig. 4.3 and Section 5.2.1. For

x	-3α	$-\alpha$	$+\alpha$	$+3\alpha$
LLR	$-\infty$	-0.6931	$+0.6931$	$+\infty$

the purpose of detection, it is clearly better to receive more symbol values of $\pm 3\alpha$. Now, assume that the SNR per info bit is given by $E_b/N_0 = 6$ dB, and the channel observations of summation checks are given by the labels with large fonts in Fig. 6.3. For the sake of clearness, we have assumed the amplitude coefficient α to be 1. We start the iterative

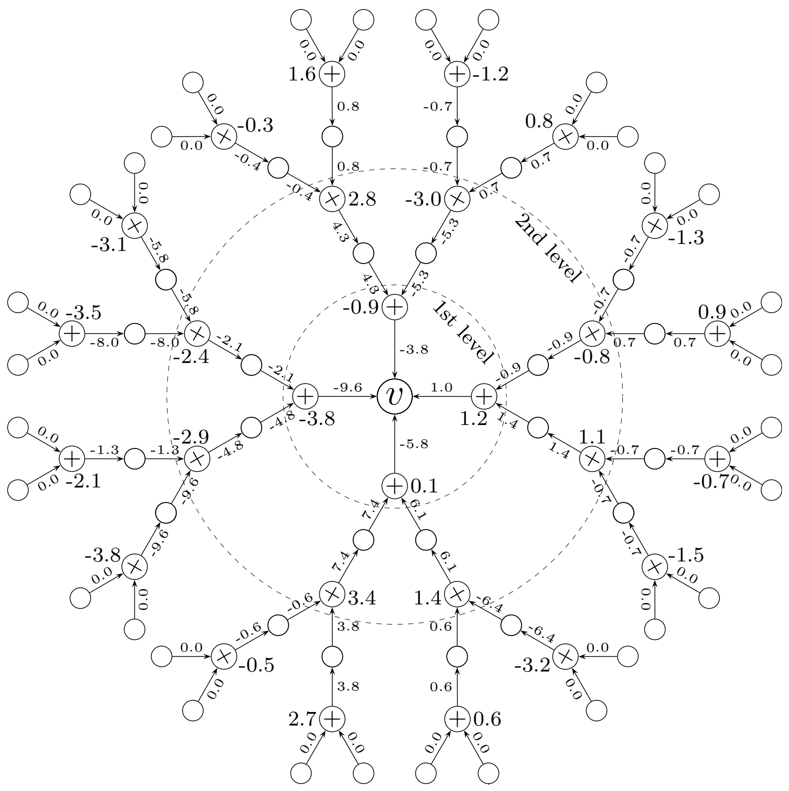


Figure 6.3: A cycle-free graph that visualizes information aggregation during the iterative message passing process of a repetition-coded SM-EPA system, $N = 3$, $E_b/N_0 = 6$ dB.

message passing from the variable nodes in the outermost ring. Since these variable nodes all have degree-1 repetition, their initial messages to the neighboring summation checks will all be zero. Nevertheless, given these all-zero messages, relevant summation checks will deliver messages, either weak or strong, to the variable nodes locating around the second iteration ring. One may find from the above figure that those summation checks with observations close to ±3 will generate very strong LLR messages, while those checks with observations close to ±1 will generate rather weak LLR messages. Since the variable nodes in the middle rings are all with degree-2 repetition, they function as if lossless relays for the message propagation. After three iterations, the information from all summation checks within this graph reaches the variable node v. We see that the four messages entering v have substantially different magnitudes, which is in fact easy to understand. By carefully checking Fig. 6.3, one will find that the summation checks in the left branch are all with observations close to ±3. It is natural that this branch delivers the strongest message to v. In contrast, the summation checks in the right branch come with observations all close to ±1. By the table given on the previous page, this is also natural that this branch delivers the weakest message to v. For the upper and lower branch, the situation is somewhere in the middle, and consequently their messages to v have middle-level magnitudes w.r.t. the other two branches. By this discussion, we get

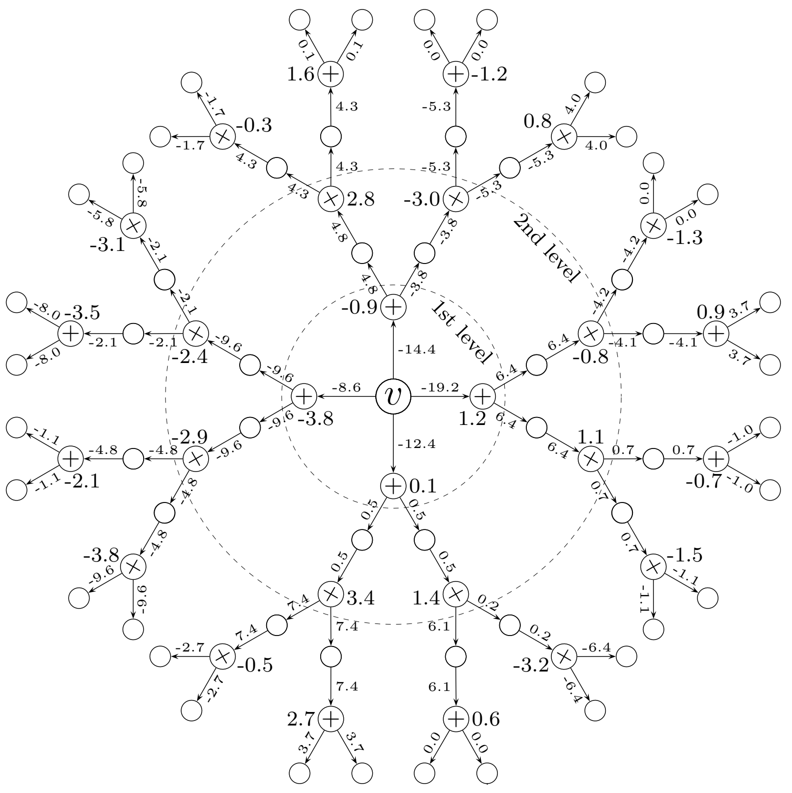
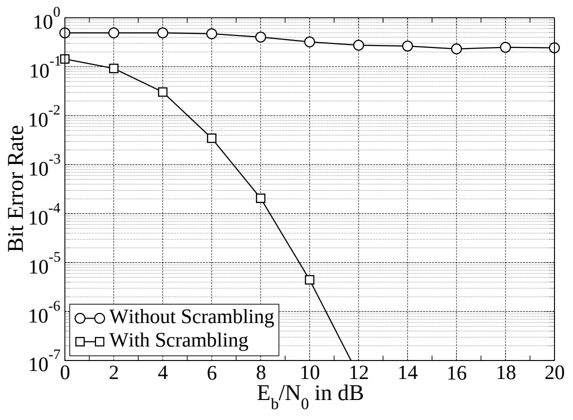


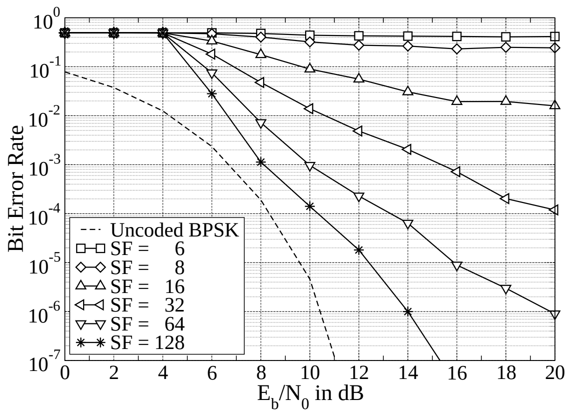
Figure 6.4: A cycle-free graph that visualizes information distribution during the iterative message passing process of a repetition-coded SM-EPA system, $N = 3$, $E_b/N_0 = 6$ dB.

an important hint. That is the quality of soft messages propagating within a factor graph of a repetition-coded SM-EPA system is highly dependent on the amount and density of summation checks with observations close to $\pm N\alpha$. Naturally, with an insufficient amount of strong observations, the iterative receiver is likely to fail.

Next, we check the effect of irregular repetition to message passing. For this purpose, we need to continue the iterations from Fig. 6.3, and distribute the information aggregated at variable node v to the rest of graph, as depicted in Fig. 6.4. Since the variable node v has degree-4 repetition, the messages delivered by this node are very strong. Roughly speaking, a variable node with a high repetition degree works like an amplifier for the LLR messages. As a result, those variable nodes close to v will all benefit from the strong messages being pumped out from it. Certainly, the extent of benefit also relies on the particular observation value of the bridging summation checks. Nevertheless, due to the expansion of node set, the reliable information from variable node v gets diluted from iteration to iteration. Hence, a high-degree variable node can only provide meaningful help for those nodes in a limited distance from it. For this reason, when one builds a graph, those high-degree variable nodes should be evenly distributed over the graph.



(a) $SF = 8$.



(b) Without scrambling.

Figure 6.5: Repetition-coded SM-EPA, random interleaving, $N = 8$, 2000 symbols per block, 20 iterations.

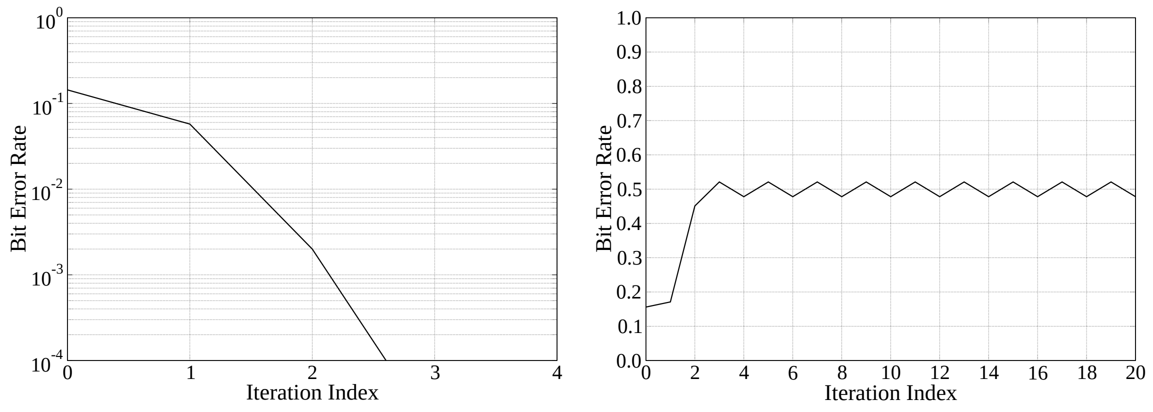
6.2 Effects of Scrambling

In former discussions, we often assumed a fixed-pattern scrambling scheme that flips every second code bit. The motivation as well as the benefit of this practice will become clear in this section.

To start the investigation, let us first have some concrete perception of the scrambling scheme under consideration. Assume that the spreading factor is given by $SF = 4$, i.e., the repetition coding rate is $R = 1/4$. Flipping every second code bit, the association between an info bit and its scrambled code word will be

$$\begin{array}{ccccc} 0 & \overset{\text{spreading}}{\longrightarrow} & [0,0,0,0] & \overset{\text{scrambling}}{\longrightarrow} & [0,1,0,1] \\ 1 & \overset{\text{spreading}}{\longrightarrow} & [1,1,1,1] & \overset{\text{scrambling}}{\longrightarrow} & [1,0,1,0] \end{array} .$$

In case of BPSK mapping, the code bit sequence after scrambling will be DC-free, independent of the info bit sequence. Generally speaking, scrambling is very effective in eliminating a DC signal component for systems employing a symmetric mapping scheme. From a physical transmission standpoint, a DC signal component consumes transmission power but carries no information at all. This explains the wide application of scrambling in practical communication systems. The same argument holds for repetition-coded SM systems. Nevertheless, the importance of scrambling to repetition-coded SM is indeed far beyond this scope. For example, given $SF = 8$ and $N = 8$, with or without scrambling brings a huge performance difference for repetition-coded SM-EPA, cf. Fig. 6.5(a). With scrambling, the receiver converges well, while without scrambling, the receiver does not converge at all. More performance results can be found in Fig. 6.5(b). By comparing Fig. 6.5(b) with Fig. 6.1, one will find that without scrambling a much larger spreading factor is needed to achieve receiver convergence, to be explained in the following.



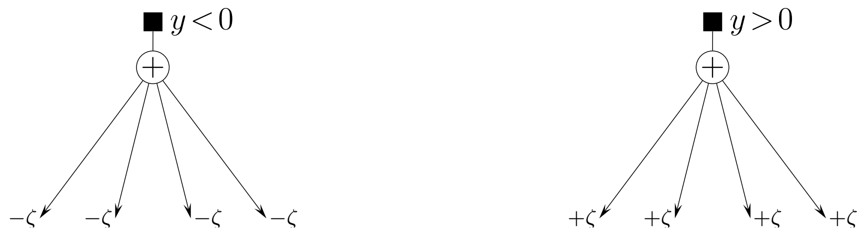
(a) With scrambling. (b) Without scrambling.

Figure 6.6: Variation of the BER w.r.t. the number of iterations. Repetition-coded SM-EPA, random interleaving, $SF = 8$, $N = 8$, 2000 symbols per block, $E_b/N_0 = 10$ dB.

6.2.1 The Trap of Message Oscillation

According to the performance test in Fig. 6.5, scrambling is necessary for repetition-coded SM-EPA and its usefulness in improving the system stability is evident. However, the reasons behind this phenomenon are so far not explicit. Obviously, the “DC component” argument is not convincing enough to explain such a big influence on the performance. For systems similar to repetition-coded SM-EPA, e.g., IDM [28] and IDMA [54], the importance of scrambling is also commonly recognized. Nevertheless, the available knowledge about the effects of scrambling is limited. A popular argument is that an APP demapper, which assumes a uniform a priori distribution ($P(b_n = 0) = P(b_n = 1) = 1/2$) for all code bits in the initial iteration, benefits from the scrambling operation. This argument is true, but does not reveal the most critical effect of scrambling for this type of systems. Note that even for conventional uniform mapping schemes, an APP demapping algorithm will make the same assumption as well. However, a phenomenon like that in Fig. 6.5 has never been reported for coded modulation with a bijective uniform mapping scheme.

To find the truth, we try to monitor the iteration process for repetition-coded SM-EPA. Fig. 6.6(a) shows the change of the BER w.r.t. the number of iterations for a certain transmission block, when scrambling is applied. One observes that the BER drops steadily as the iterative demapping and decoding process proceeds. After a few iterations, the BER is already below 10^{-4}. On the other hand, the situation is catastrophic when scrambling is not applied, as demonstrated in Fig. 6.6(b). Instead of decreasing step-by-step, in a few iterations the BER starts to rapidly alternate between two values at a level around 0.5. Hence, we obtain an important message that an iterative receiver for repetition-coded SM-EPA can easily fall into a periodic oscillating state, if no scrambling is applied. Next, we try to understand how and why such kind of receiver oscillations happen.



(a) With a negative channel observation. (b) With a positive channel observation.

Figure 6.7: The homogeneousness of summation check messages in the initial iteration.

As a matter of fact, an ensemble of summation checks and variable nodes can form a "good" message oscillator, given a "suitable" set of channel observations. For the sake of easy elaboration, we will first investigate the behaviour of summation checks and variable nodes in some extreme cases and later on extend the discussion to more general scenarios.

Given equal power allocation, the initial messages from an individual summation check are homogeneous. Assuming an SM-EPA input, the AWGN channel equation is given by

$$y = x + z = \sum_{n=1}^{N} c_n + z \ , \quad c_n \in \{\pm\alpha\} \ . \tag{6.2}$$

In case that the channel observation y is less than zero, a summation check will assume that the input symbol x is more likely to be negative. Due to equal power allocation, the input symbol x can only be negative if it contains more negative chips than positive ones. For this reason, this summation check (APP demapper) will deliver all-negative LLR messages to the variable nodes, in the initial iteration. Vice versa, if the channel observation is larger than zero, a summation check will deliver all positive-LLR messages to the variable nodes. Moreover, in the initial iteration, the magnitudes of LLR messages from an individual summation check will all be identical. This is easy to understand. Due to equal power allocation, a summation check itself cannot make any distinction between the involved chips for the LLR calculation, when the a priori information from the decoder is still not available. Fig. 6.7 demonstrates this phenomenon, where ζ denotes a certain message magnitude. This often gives the starting point for periodic message oscillations.

Given all-positive message inputs, a variable node will deliver all-positive message outputs, and vice versa. Fig. 6.8 demonstrates this phenomenon. Let $L_n^{(i)}$ and $L_n^{(o)}$ denote the input and output message on the nth edge, respectively. For a degree-D variable node, we have

$$L_n^{(o)} = \sum_{m=1, m\neq n}^{D} L_m^{(i)} \qquad \forall \quad 1 \leqslant n \leqslant D \ . \tag{6.3}$$

Clearly, if the input messages are all-positive, their summations will also be all-positive. Besides, if the input messages are identical, the output messages will be identical as well. This gives an important reason for periodic message oscillations.

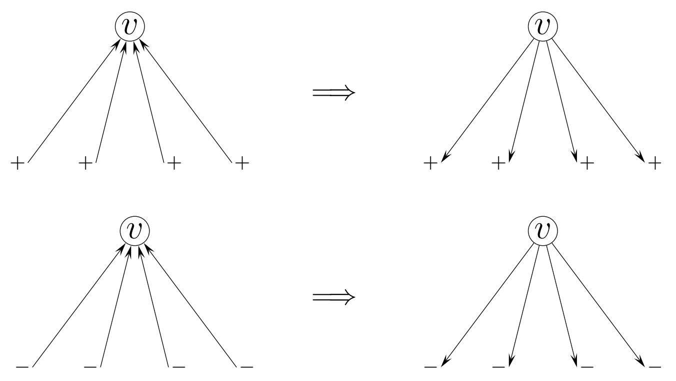


Figure 6.8: Given uni-sign inputs, a variable node produces uni-sign outputs.

Another important reason for periodic message oscillations is the tendency of summation checks in inverting the sign of incoming messages. The SM-EPA symbol alphabet is given by $\mathcal{X} = \{-\alpha N, -\alpha(N-2), \ldots, \alpha(N-2), \alpha N\}$. There will be a threshold γ_0 such that for all $y < +\gamma_0$ the probability $P(x = +\alpha N|y)$ is negligible and for all $y > -\gamma_0$ the probability $P(x = -\alpha N|y)$ is negligible. For a moderate SNR, we have $\alpha(N-2) < \gamma_0 < \alpha N$, as depicted in Fig. 6.9. Given $y < +\gamma_0$, a summation check firmly believes that $x \leqslant \alpha(N-2)$, or in other words, it firmly believes that there must be at least one chip being negative. In this case, given all-positive inputs, this summation check will deliver all-negative outputs, illustrated in the upper part of Fig. 6.10. Hence, all the message signs are simply inversed. Similarly, by observing $y > -\gamma_0$ and receiving all-negative inputs, a summation check will deliver all-positive outputs, illustrated in the lower part of Fig. 6.10. For a large bit load, the probability to have $-\alpha(N-2) \leqslant x \leqslant \alpha(N-2)$ is close to 1, due to the non-uniform symbol distribution. Subsequently, the probability to have $\gamma^- \leqslant y \leqslant \gamma^+$ is also close to 1, given a moderate SNR. That is, within a transmission block, the majority of summation checks can show a behaviour as depicted in Fig. 6.10.

Now, let us check the effects of the above described properties for a small piece of graph. Consider repetition-coded SM-EPA with $N = 3$ and $\alpha = 1$. We assume that the SNR is high such that $\gamma_0 \approx \alpha(N-1) = 2$. Suppose that the observations for the summation checks are all 1, as labelled in Fig. 6.11. Clearly, the initial messages from the summation checks will be all-positive. By receiving all-positive messages, the variable nodes will output all-positive messages. This ends the message passing for the initial iteration. For the second iteration, due to receiving all-positive messages and having $-\gamma_0 < y < +\gamma_0$, the summation checks will output all-negative messages. Following that, the variable nodes will deliver all-negative messages. This ends one cycle of message oscillation. Easy to imagine, this type of message oscillations lead to a dead loop, and there is no chance for the receiver to escape from such a trap. Without loss of generality, we may call such ill-conditioned receiver iterations the trap of message oscillation.

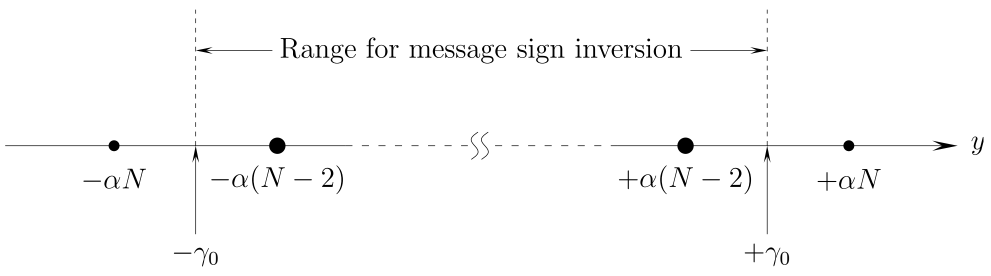


Figure 6.9: The range of channel observation that may lead to message sign inversion.

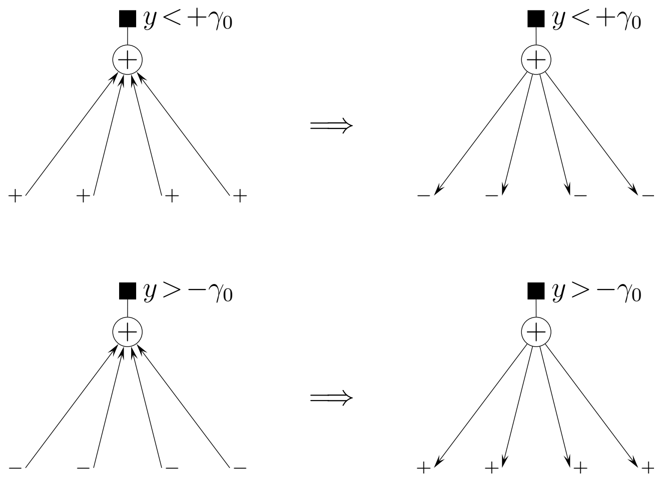


Figure 6.10: The phenomenon of message sign inversion at summation checks.

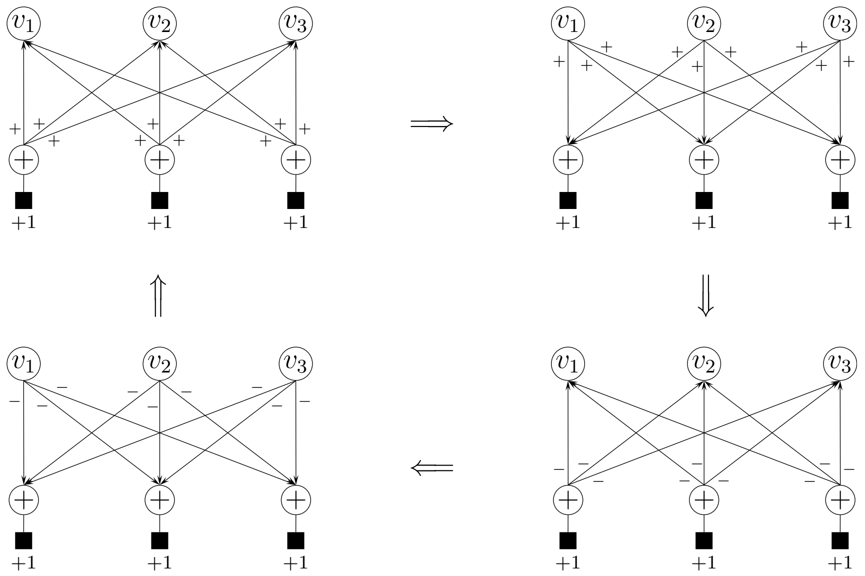
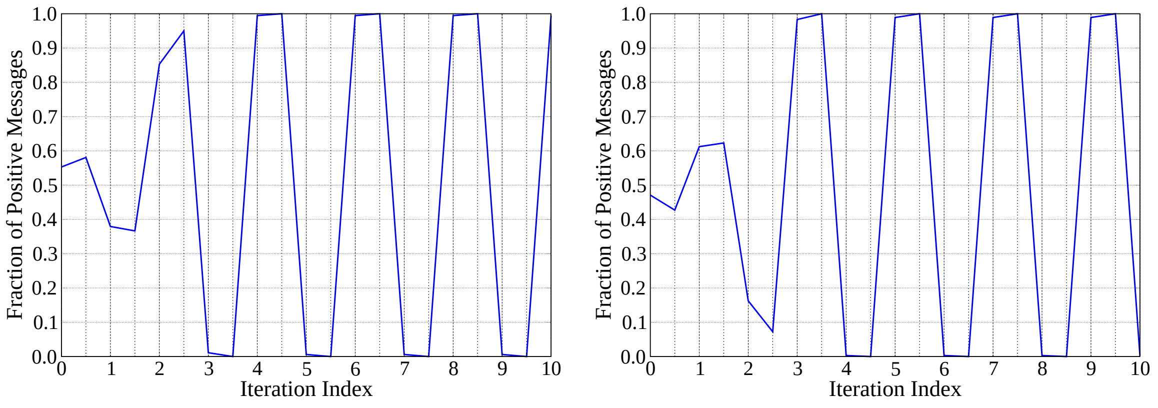


Figure 6.11: A periodic message oscillation within a small piece of graph.



(a) With a starting point higher than 0.5. (b) With a starting point lower than 0.5.

Figure 6.12: Fraction of positive messages at each iteration. Repetition-coded SM-EPA, no scrambling, random interleaving, $SF = 8$, $N = 8$, $K = 2000$, $E_b/N_0 = 10$ dB.

The example considered in Fig. 6.11 is over-simplified, in the sense of using an extremely small block length. Given a realistic block length, the situation is more complicated. To attain a good understanding, we make a new test. For randomly generated data blocks, we measure the fraction of positive messages (FPM) from summation checks as well as variable nodes at each iteration. Note that each iteration starts from message updating at summation checks and ends with that at variable nodes. In Fig. 6.12, the measured values at integer indices correspond to the FPM from summation checks, and those at fractional indices correspond to the FPM from variable nodes. Let us first focus on Fig. 6.12(a). As the starting FPM is slightly larger than 0.5, there is a mild sign imbalance in the initial messages from summation checks. However, this mild imbalance gets amplified in the following message updating process at variable nodes. Illustrated in Fig. 6.13, a variable node tends to unify the message signs whenever the inputs are considerably biased[1]. In the second iteration, this sign imbalance is further amplified by the message updating process at summation checks, albeit towards a reverse direction. Illustrated by Fig. 6.14 and Fig. 6.15, given a channel observation in the range $[-\gamma_1, +\gamma_1]$ a summation check tends to reverse and unify the message signs. Following this procedure, the FPM soon starts to periodically alternate between a value close to 0 and a value close to 1. This means that the messages are either almost all-positve or almost all-negative. Since a large data block will contain roughly equal amount of 0's and 1's, the BER will be alternating between two values close to 0.5, which well explains the result in Fig. 6.6(b). The FPM cannot be really 0 or 1 because those summation checks with $y \ll -\gamma_0$ or $y \gg +\gamma_0$ will insist on outputting messages with a certain sign independent of the inputting messages. Now, the result in Fig. 6.12(b) also becomes easily understandable. The only difference w.r.t. Fig. 6.12(a) is that the oscillation process starts from an opposite direction.

[1]The message updating process at variable node v in Fig. 6.3 and 6.4 gives a good numerical example.

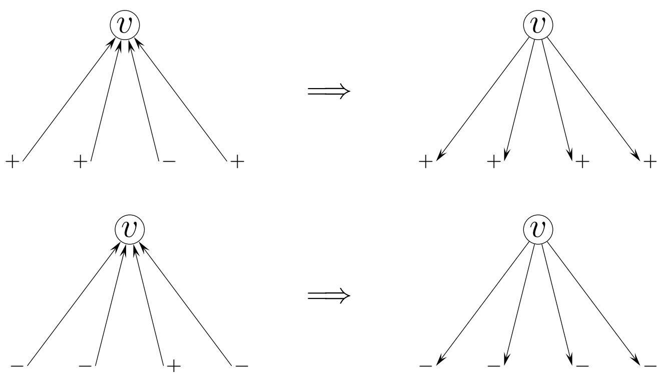


Figure 6.13: The phenomenon of message sign unification at variable nodes.

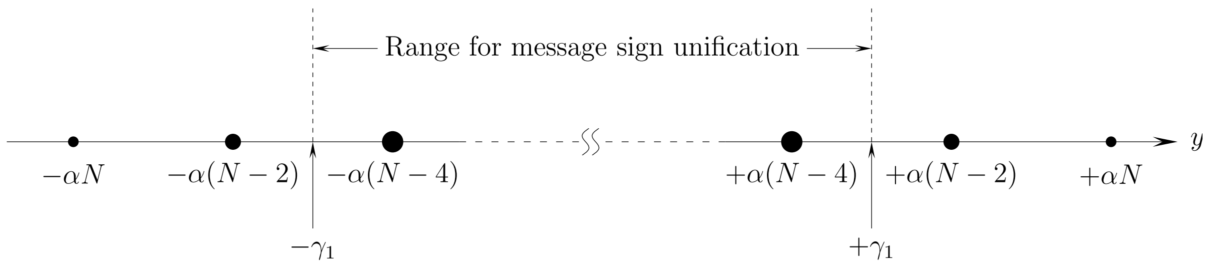


Figure 6.14: The range of channel observation that may lead to message sign unification. Given $y < +\gamma_1$, a summation check firmly believes that at least two chips are negative. Given $y > -\gamma_1$, a summation check firmly believes that at least two chips are positive.

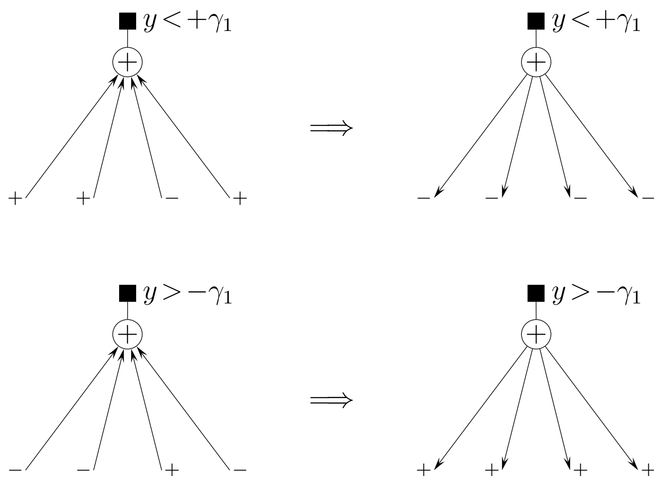
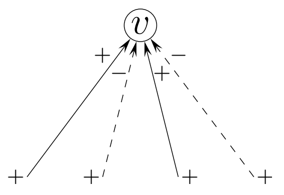


Figure 6.15: The phenomenon of message sign unification at summation checks.

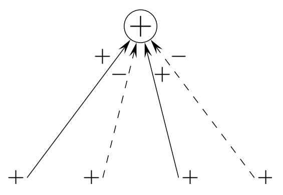
The observations from Fig. 6.12 indicate that a superposition demapper and a repetition decoder tend to "help" each other in entering a periodic osciallating state, which is certainly disastrous for the BER performance. Given a random interleaver, the overall factor graph will often be a collection of quasi-isolated sub-graphs, with each sub-graph having a limited connectivity to the rest of the graph. Assuming i.i.d. info bits, the sign of channel observations within the whole transmission block will be more or less balanced. However, this is likely not the case for a small sub-graph which contains only a limited amount of channel observations. Whenever the channel observations within a certain sub-graph are strongly biased in the sign, as in the case of Fig. 6.11, the message passing process within this sub-graph will fall into the trap of message oscillation with a high probability. Afterwards, with a certain number of sub-graphs falling into the trap of message oscillation, these sub-graphs will soon bring the whole graph into a periodic oscillating state, and consequently prevent the receiver from achieving any performance gain via iterations. To obtain a desirable performance, the trap of message oscillation has to be avoided. There are several possibilities to achieve this, to be elaborated in the following.

The first possibility is to use a large spreading factor, as already implied by Fig. 6.5(b). Given a large spreading factor, each variable node will be connected with a large amount of summation checks. Due to a diversity effect, the sum of messages from these summation checks is reliable already in the initial iteration. A variable node pumping out reliable messages helps the graph to converge to a stable state instead of creates periodic message oscillations. If most of the variable nodes behave like this, it will be difficult for message osciallations to take place. More specifically, channel observations close to $\pm\alpha N$ are particularly helpful for the detection process, since the corresponding summation checks will not create or participate periodic message oscillations. We have $P(x = \pm\alpha N) = 2/2^N$ for SM-EPA symbols. Given a spreading factor that is comparable to 2^{N-1}, the chance that each variable node receives at least one channel observation close to $\pm\alpha N$ becomes considerable, and consequently drops the probability of message oscillations to a low level. For the case of Fig. 6.5(b), we have $2^{N-1} = 2^7 = 128$. Nevertheless, an unnecessarily large spreading factor leads to an unnecessarily low data rate, which is certainly undesirable.

The second possibility is to enlarge the length of cycles. According to former discussions, a necessary condition for message oscillations to happen is that each graph node receives the "echos" of the messages delivered by itself from the rest of graph within a very limited number of iterations. Easy to imagine, it is strictly not possible for a periodic message oscillation to happen over a cycle-free factor graph. Moreover, message "echos" from long cycles are likely to have a diversity due to being extrinsic. Hence, a reasonable conjecture is that a factor graph consisting of purely long cycles will be immune to message oscillations. We will check the correctness of this conjecture via numerical simulations in Section 6.4.3.

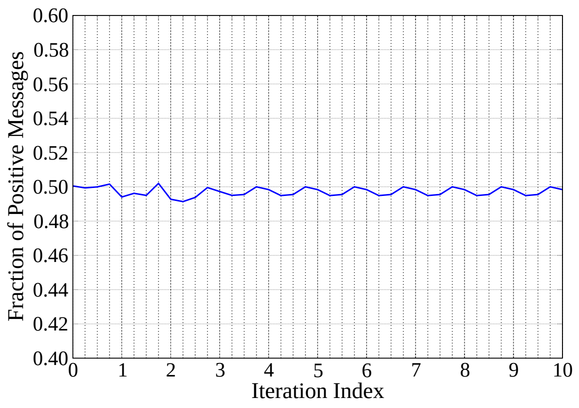


(a) Variable-node-based scrambling (VBS).

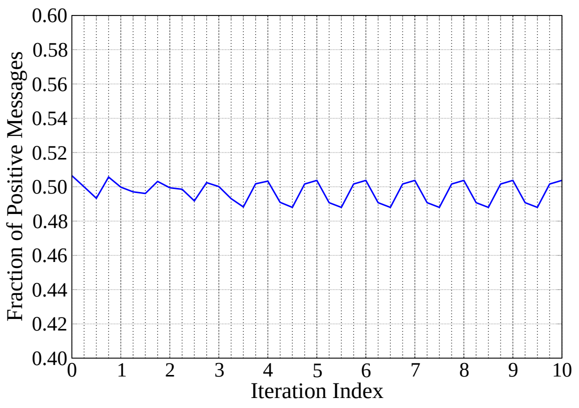


(b) Summation-check-based scrambling (SBS).

Figure 6.16: Two possible strategies for scrambling: VBS and SBS.



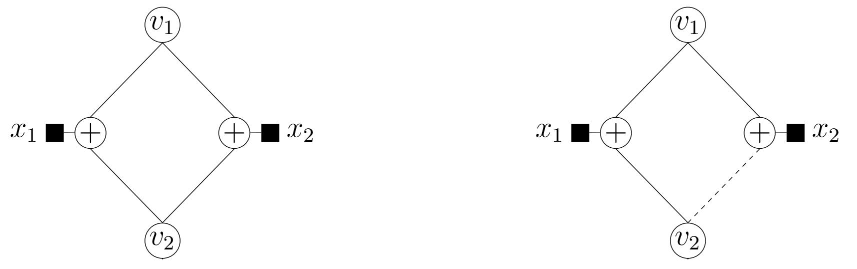
(a) Variable-node-based scrambling.



(b) Summation-check-based scrambling.

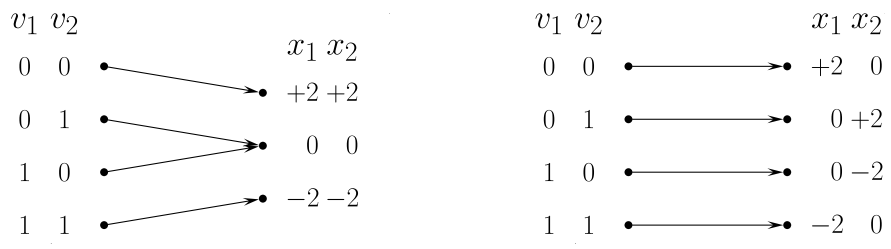
Figure 6.17: The effect of scrambling on the variation of FPM w.r.t. iterations.

The third and the most efficient way to avoid message oscillations is to apply scrambling. Not difficult to find, the primary condition for periodic message oscillations is that by receiving uni-sign incoming messages a graph node will produce stronger but still uni-sign outgoing messages. By means of scrambling after repetition encoding, cf. Fig. 5.6, every second edge from an individual varaible node will contain a message-sign-reversing operation as illustrated in Fig. 6.16(a). Suppose that a message oscillation tends to start, such that a variable node receives uni-sign incoming messages. Due to flipping, half of the messages will have a reversed sign when they finally enter this variable node. In the following message updating process at this variable node, the negative messages will more or less cancel out the positive messages. As a result, the new outgoing messages from this variable node will be weaker and usually non-uni-sign. In this way, the primary condition for message oscillations gets destroyed. We call such a scrambling strategy variable-node-based scrambling (VBS). Similarly, we may apply scrambling after interleaving such that every second edge from an individual summation check contains a message-sign-reversing operation, as depicted in Fig. 6.16(b). We call such a scrambling strategy summation-check-based scrambling (SBS). Fig. 6.17 measures the FPM before and after each message updating operation in a few iterations for VBS and SBS. It shows that VBS and SBS are both effective in stabilizing the iterative detection process, with VBS showing slightly better performance. We will assess the corresponding BER performances in Section 6.4.3.



(a) Without scrambling. (b) With one code bit flipped.

Figure 6.18: Factor graph for repetition-coded SM-EPA, $SF = 2$, $N = 2$, $K = 2$.



(a) Without scrambling. (b) With one code bit flipped.

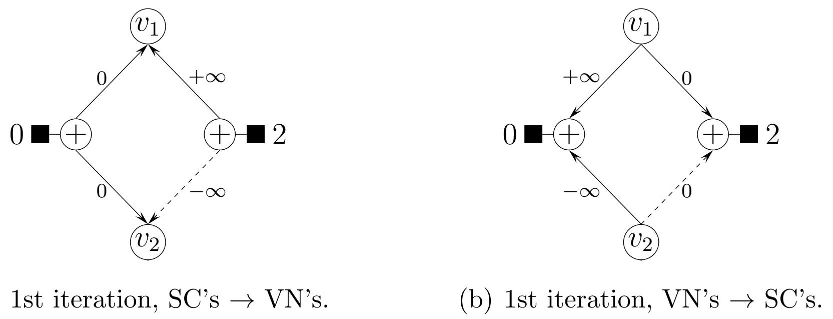
Figure 6.19: Overall mapping rule for repetition-coded SM-EPA, $SF = 2$, $N = 2$, $\alpha = 1$.

6.2.2 Distinguishability of Overlapped Repetition Code Words

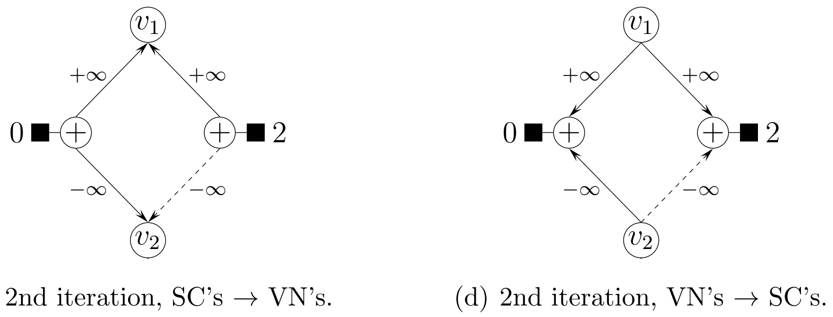
In the previous discussion, we have seen an important effect of scrambling for repetition-coded SM-EPA in preventing an iterative receiver from falling into the trap of message oscillation. In the following, we consider another effect of scrambling, which improves the distinguishability of overlapped repetition code words.

To ease the discussion, we consider repetition-coded SM with $SF = 2$ and $N = 2$ as an illustrative example. As assumed throughout this chapter, equal power allocation is adopted. Suppose that the block length is very short such that each burst contains merely two info bits. Given these assumptions, an optimal interleaver will lead to a factor graph depicted in Fig. 6.18(a). Clearly, this is a single-cycle graph. Besides, the two repetition code words corresponding to v_1 and v_2 overlap with each other. In order to emphasize the effect of scrambling, we further assume that the transmission channel is noiseless.

For the system under consideration, the prerequisite for an error-free detection is that the mapping rule from the info word $\mathbf{v} \doteq [v_1, v_2]$ onto the symbol sequence $\mathbf{x} \doteq [x_1, x_2]$ is bijective. It is easy to derive that, without scrambling, the overall mapping rule will be non-bijective, as shown in Fig. 6.19(a). Since $+1 - 1 = -1 + 1 = 0$, the bit combinations $[0, 1]$ and $[1, 0]$ will both be mapped onto an all-zero symbol sequence. One may call such a system a catastrophic encoder. Hence, error-free detection is not possible for any type of receiver algorithms. Now, we apply scrambling, but only for v_2. The resulting factor



(a) 1st iteration, SC's → VN's. (b) 1st iteration, VN's → SC's.



(c) 2nd iteration, SC's → VN's. (d) 2nd iteration, VN's → SC's.

Figure 6.20: A message passing procedure on a length-4 cycle with one edge flipped.

graph is given in Fig. 6.18(b), where the dashed edge denotes a bit-flipping operation. Correspondingly, one obtains an overall mapping rule as shown in Fig. 6.19(b). What can be seen is that the overall mapping rule is now bijective. Given a maximum-likelihood receiver, error-free detection can be achieved, and actually so does an iterative receiver. Suppose that the received symbol sequence is $[0, +2]$. With a little bit of computation, one obtains a message passing procedure as demonstrated in Fig. 6.20. Since the channel is noiseless, the magnitude of an LLR message is infinite, whenever it is not zero. One may recognize that even in the first iteration correct decisions are already available for v_1 and v_2, while in the second iteration all messages are non-zero and actually stable. Note that a degree-2 variable node simply exchanges the incoming messages without altering the signs or the magnitudes. In contrast, a degree-2 summation check with observation 0 exchanges the incoming messages as well, but with the message signs reversed.

Easy to find, if we apply scrambling for both v_1 and v_2, the overall mapping rule will again be non-bijective. This suggests that applying scrambling for all repetition code words does not always give the best solution. For the sake of easy elaboration, we delay further discussion on this topic till Section 6.4.3, before which we will keep using the scheme that flips every second code bit. The example considered so far is special, as the cycle length is only 4. In reality, length-4 cycles can be eliminated via a proper interleaver design. The effect of scrambling in improving the distinguishability of overlapped repetition code words gets weaker when the cycle length increases. Nevertheless, it is still useful for low-degree variable nodes when the cycle length is moderate, which is studied in Section 6.4.3.

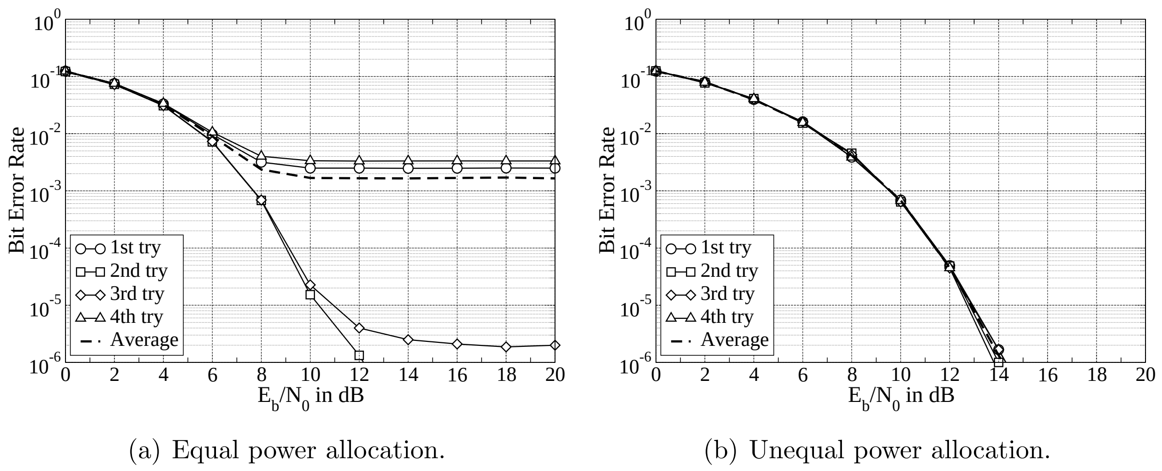


Figure 6.21: BER vs. E_b/N_0, repetition-coded SM-EPA, every second code bit flipped, random interleaving, $SF = 2$, $N = 2$, 300 symbols per block, 20 iterations.

6.3 Effects of Interleaving

So far, we have investigated the effects of spreading and scrambling on repetition-coded SM-EPA. In the relevant discussions we have always assumed random interleaving. However, it is not yet clear whether a random interleaver is already good enough. In this section, the influence of interleaving on the performance of repetition-coded SM-EPA will be investigated. Via a simple example, the impact of the graph structure on the receiver stability of repetition-coded SM-EPA becomes clear. While this section mainly focuses on conceptual studies, the issue of interleaver design will be treated in detail in Section 6.4.

6.3.1 A First Impression

Let us first have a rough impression on the topic under discussion. For simplicity, let us choose $SF = 2$ and $N = 2$. In order to emphasize the effects of interleaving, here we choose a moderate block length of $K = 300$. The corresponding BER performances are demonstrated in Fig. 6.21 for EPA as well as UPA. In both Fig. 6.21(a) and Fig. 6.21(b), the dashed curve gives the average performance corresponding to 50000 randomly generated interleaver patterns and the four solid curves give the performances corresponding to four fixed but randomly generated interleaver patterns. By the first snapshot, one sees that repetition-coded SM-EPA is very sensitive to the interleaver pattern while repetition-coded SM-UPA is insensitive. Since GPA is simply a hybrid of EPA and UPA, it is not difficult to imagine that the situation of SM-GPA will be somewhere in between that of SM-EPA and SM-UPA. This gives the reason why the discussion in this chapter merely focuses on SM-EPA. After all, one observes from Fig. 6.21(a) that different interleaver patterns may lead to dramatically different BER performances for repetition-coded SM-EPA. Hence, for a moderate block length, the interleaver pattern needs to be carefully designed rather than being created completely at random.

According to Fig. 6.21(a), the average performance of random interleaving is not acceptable for repetition-coded SM-EPA, under the current system setup. However, there are some interleaver patterns that bring acceptable performances, e.g., the 2nd and 3rd try. Hence, one may wonder what is the true reason behind this huge performance difference. To answer this question, we introduce in the following a simple yet heuristic interleaver construction example for repetition-coded SM-EPA with $SF = 2$ and $N = 2$.

6.3.2 A Heuristic Example for Interleaver Design

Consider the transmission of four info bits $[v_1, v_2, v_3, v_4]$. Assume that the spreading, scrambling, and interleaving operations finally yield the following code bit sequence:

$$\begin{aligned}
[v_1, v_2, v_3, v_4] &\overset{\text{spreading}}{\longrightarrow} [v_1, v_1, v_2, v_2, v_3, v_3, v_4, v_4] \\
&\overset{\text{scrambling}}{\longrightarrow} [v_1, \overline{v}_1, v_2, \overline{v}_2, v_3, \overline{v}_3, v_4, \overline{v}_4] \\
&\overset{\text{interleaving}}{\longrightarrow} [\overline{v}_1, v_2, \overline{v}_2, v_3, \overline{v}_3, v_4, \overline{v}_4, v_1] ,
\end{aligned}$$

where $\overline{v}_i$ denotes the complement of v_i in $GF(2)$. Note that the interleaver merely performs a one-step cyclic shift on the bit sequence. Now, at the output of the superposition mapper one obtains

$$\begin{aligned}
x_1 &= B(\overline{v}_1) + B(v_2), \\
x_2 &= B(\overline{v}_2) + B(v_3), \\
x_3 &= B(\overline{v}_3) + B(v_4), \\
x_4 &= B(\overline{v}_4) + B(v_1),
\end{aligned}$$

where $B(v) \doteq 1-2v$ stands for the BPSK mapping operation. As a result, the corresponding factor graph will be given by Fig. 6.22, which shows that the entire set of graph nodes are aligned within one unique cycle. Obviously, this simple interleaver indeed maximizes the cycle length. In the next step we will have some interesting study on the performance of such a system setup, which is actually easily predictable.

Revisiting Fig. 5.2 and related discussion therein, one sees that for noiseless SM-EPA transmission with $N = 2$, the LLR messages leaving from a summation check fall into two extremes: of infinite reliability when the observation x is nonzero and of zero reliability when x is zero, as long as there is no a priori information available for the involved code

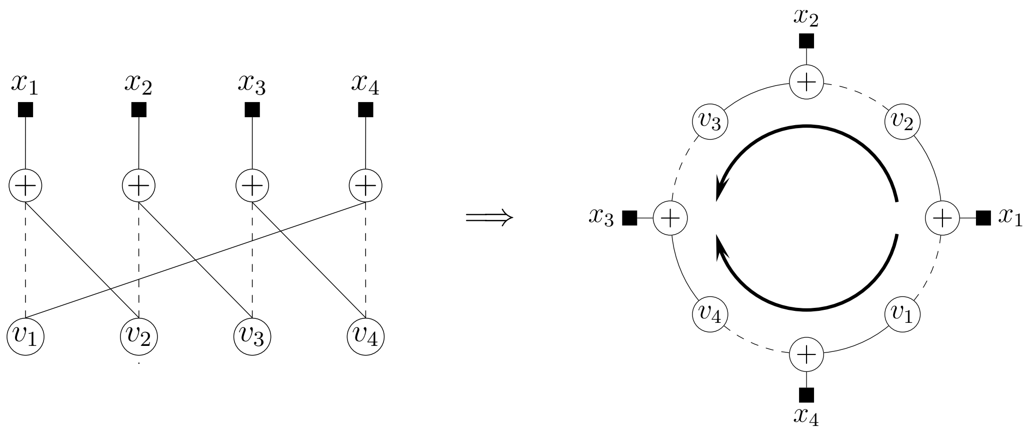


Figure 6.22: Factor graph for repetition-coded SM, $SF = 2$, $N = 2$, $K = 4$, one-step cyclic-shift interleaving. Each dashed edge denotes a bit-flipping operation.

bits. Consequently, for the system under consideration, the results of iterative detection will also fall into two extremes: $P_e = 0.5$ when the observed SM symbols $[x_1, x_2, x_3, x_4]$ are all zero and $P_e = 0$ whenever there is one observed SM symbol being nonzero, assuming a noiseless channel. The respective explanation is given below.

Suppose that the received symbol seqeuence is

$$[x_1, x_2, x_3, x_4] \quad = \quad [0, 0, 0, 0]$$

which happens when the transmitted info bits are all-zero or all-one, i.e.,

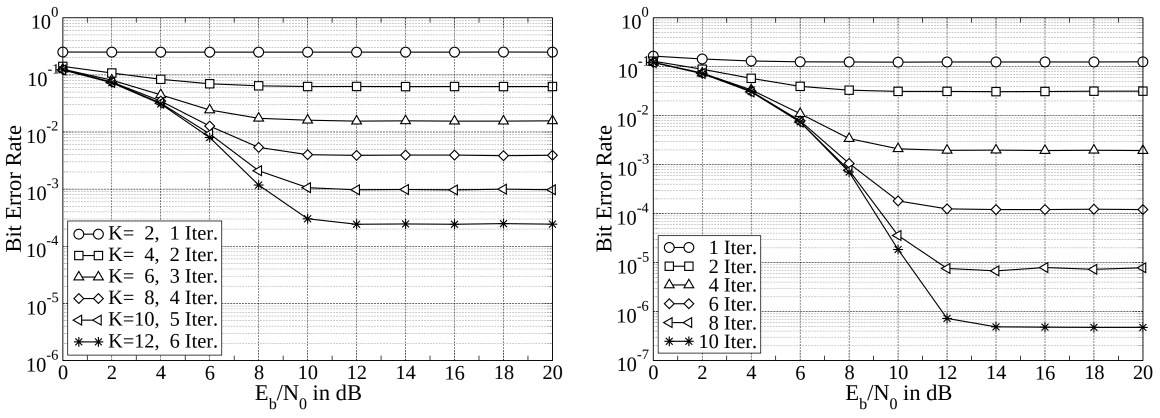
$$[v_1, v_2, v_3, v_4] \;=\; [0, 0, 0, 0] \qquad \text{or} \qquad [v_1, v_2, v_3, v_4] \;=\; [1, 1, 1, 1]\;.$$

Clearly, this will be a catastrophic encoding scenario as two distinct info words are mapped onto the same code word. At the initial iteration, all messages leaving the summation checks will be zero. The exchanging of all-zero messages at the variable nodes will generate all-zero messages as well. At the second iteration, by receiving all-zero messages the summation checks will deliver all-zero messages again. Therefore, nothing will happen through iterative message passing and the resulting bit error probability is $P_e = 0.5$.

Now suppose that one of the observed symbols is non-zero, for example

$$x_1 = +2\;. \tag{6.4}$$

At the initial iteration, the summation check associated with x_1 will deliver reliable messages to variable nodes v_1 and v_2. Then, the information flow procedure goes as depicted in the right part of Fig. 6.22. Receiving the message forwarded by node v_2, summation check x_2 will be able to deliver a reliable message to node v_3, regardless of the particular value of x_2. The same situation holds for the message from x_4 to v_4 as well. Hence, after



(a) Effect of block length. (b) Effect of iterations, $K = 300$.

Figure 6.23: Repetition-coded SM-EPA, $SF = N = 2$, one-step-cyclic-shift interleaving, every second code bit flipped. K denotes the number of symbols per block.

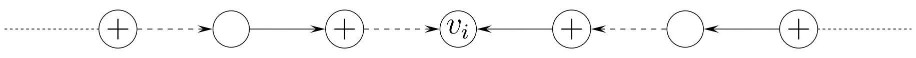


Figure 6.24: An open chain of nodes.

two iterations, the reliable messages from summation check x_1 have already reached all the variable nodes, and subsequently enables an error-free decision for all the bits.

Note that the probability for the info bits to be all-zero or all-one is given by 2×2^{-4}. Therefore, we predict that the average bit error probability is

$$P_e = \frac{2}{2^4} \cdot 0.5 + \left(1 - \frac{2}{2^4}\right) \cdot 0 = 0.0625 \; . \tag{6.5}$$

We can easily generalize this conclusion for an arbitrary block length. Given a noiseless channel and sufficient receiver iterations, the average bit error probability will be

$$P_e = \frac{2}{2^K} \cdot 0.5 + \left(1 - \frac{2}{2^K}\right) \cdot 0 = \frac{1}{2^K} \; , \tag{6.6}$$

which in fact gives the level of error floor in case of transmission over the AWGN channel. (6.6) can be well verified by the Monte Carlo simulation results provided in Fig. 6.23(a), where the number of iterations is chosen to be $K/2$ such that the messages from one summation check will reach all other nodes once and only once. Clearly, by increasing the block length K, one can make this error probability arbitrarily close to zero. In practice, however, an error probability around 10^{-6} is already acceptable for some applications. Therefore, $K/2$ receiver iterations are not always necessary. Easy to imagine, within $T < K/2$ iterations, the graph is seen from a particular node as an open chain, i.e., it looks like cycle-free. Due to bi-directional message passing, the amount of variable nodes reachable from a certain variable node within T iterations is given by $2T$, cf. Fig. 6.24.

The probability for these bits, including the bit at the origin, to be all-zero or all-one is given by $2 \times 2^{-(1+2T)}$. Hence, the average bit error probability will be

$$P_b = 2 \cdot 2^{-(1+2T)} \cdot 0.5 = 2^{-(1+2T)} \; , \tag{6.7}$$

One sees from Fig. 6.23(b) that after 10 iterations the error floor level drops below 10^{-6}. In general, whenever a graph contains a non-trivial amount of degree-2 variable nodes, the corresponding interleaver design needs to be specifically careful. We will come back to this topic in Section 6.4.2.

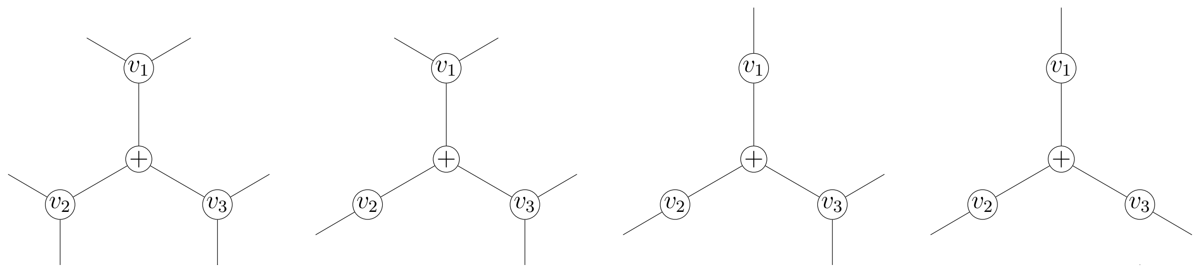
6.3.3 Summation Check Extrinsic Message Degree

In the field of interleaver design for LDPC codes, it is a common knowledge that the minimum cycle length of the graph, often called the graph girth, should be large enough. This is also true for repetition-coded SM-EPA, as demonstrated by the discussion in Section 6.3.2. However, in case of irregular repetition-coded SM-EPA, there will be another critical issue that may cause a significant performance difference.

Let us consider a repetition-coded SM-EPA system with $N = 3$. We apply the following degree distribution

$$\lambda(D) = 0.75D^2 + 0.25D^3 \; . \tag{6.8}$$

That is, 75% of variable nodes have a repetition degree of 2 and 25% of variable nodes have a repetition degree of 3. Since $N = 3$, each summation check is connected with three variable nodes. Given an interleaver pattern that has not been specifically tuned, four different type of segments may exist in the graph, as depicted in Fig. 6.25. Similar to the definition of approximate cycle extrinsic message degree (EMD) in [59], we define the EMD of a summation check as the amount of extrinsic message paths available from the variable nodes connected to it. For example, in Fig. 6.25(a) the summation check has an EMD of 6, and in Fig. 6.25(b) the summation check has an EMD of 5. This definition of EMD is only accurate for a cycle-free graph. When the graph contains cycles, this definition of EMD is approximate, because the message from a variable node to a summation check may be not completely extrinsic. For example, in the scenario depicted in Fig. 6.26, the EMD of the summation check in the middle of the graph segment is only approximately 6, as the messages from node v_1 and node v_3 are not completely extrinsic due to the length-6 cycle. Nevertheless, for a well-designed graph with a large girth, the accurateness of this EMD definition is satisfying. On average, the EMD of a summation check well describes the amount of external helps that this summation check can get from the rest of the graph during iterative message passing. Certainly, the larger the EMD is, the easier it is for a summation check to deliver reliable decisions for the involved code bits.



(a) EMD = 6. (b) EMD = 5. (c) EMD = 4. (d) EMD = 3.

Figure 6.25: Summation checks having various extrinsic message degrees (EMD).

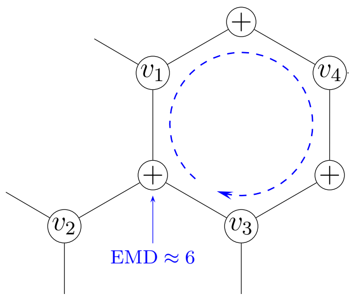


Figure 6.26: A scenario that the definition of summation check EMD is only approximate.

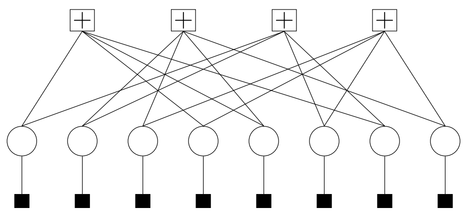
In addition to the degree distribution of variable nodes, the distribution of summation check EMD (SCE) is another important description for the structural property of a graph. When the summation checks in a graph possess various EMD's, the graph attains another type of irregularity. Easy to imagine, those summation checks with a high EMD will first start to output reliable messages, which then help those low-EMD summation checks to output reliable messages. This type of irregularity is often helpful for systems working closely to a certain theoretical limit. However, there is also a price for this irregularity. In case that a certain amount of summation checks with a very low EMD form an isolated sub-graph, a non-trivial error floor typically occurs. Let us presume that the degree distribution in (6.8) can provide a good receiver convergence[2] for SM-EPA with $N = 3$. Suppose that within an isolated sub-graph all the summation checks are connected with purely degree-2 variable nodes, i.e., all with a connectivity shown in Fig. 6.25(d). The iterative receiver is likely to encounter problems, as we observe from Fig. 5.9(a) that it is difficult to achieve a receiver convergence for regular repetition-coded SM-EPA with $SF = 2$ and $N = 3$. Generally speaking, a wide SCE distribution is good for the performance in the waterfall region, while a narrow SCE distribution is good for the performance in the high-SNR region. However, this statement has to be understood in a relative way. In case that a receiver convergence is not achievable given a narrow SCE distribution, a wide SCE distribution is also good for the performance in the high-SNR region.

[2]The investigation in Section 6.4.2 will verify this presumption.

6.4 Low-Density Summation-Check Code

Section 6.1, 6.2, and 6.3 investigate the effects of spreading, scrambling, and interleaving on repetition-coded SM-EPA, from various aspects. Among all, the most important finding is that the performance of repetition-coded SM-EPA can be improved via irregular spreading. It is worthwhile to mention that the unequal error protection effect caused by irregular spreading has an essential difference to what unequal power allocation will cause. In general, the irregularity in spreading has no influence on the symbol distribution of the superposition mapper. Hence, SM with irregular spreading does not lose the potential to achieve the Gaussian channel capacity, as long as the symbol distribution is Gaussian-like. In contrast, SM-UPA brings a uniform symbol distribution and consequently loses the optimality for the Gaussian channel. Compared to spreading, scrambling and interleaving neither vary the system data rate nor change the situation of error protection. Consequently, these two operations have no explicit influence on the theoretical performance limit. However, their strong impacts on the performance of a practical iterative receiver are evident, given a finite block length. For repetition-coded SM-EPA, scrambling together with interleaving dramatically improves the receiver stability, as demonstrated in Section 6.2 and 6.3. Applying scrambling after spreading prevents a receiver from falling into the trap of message oscillation, while applying interleaving after scrambling increases the amount of reachable channel observations from an individual info bit within a finite number of iterations. Furthermore, scrambling together with interleaving helps in distinguishing partially or completely overlapped repetition codewords, if it does happen due to certain reasons. Therefore, one may treat scrambling and interleaving as two necessary companions for spreading. Without these two companions, a good spreading scheme alone can not provide a satisfying performance for SM-EPA.

Summarizing the above statements, to attain a good performance for repetition-coded SM-EPA, one needs a good spreader, a good scrambler, and a good interleaver. One needs some fine tuning in order to obtain a good spreader, and one needs some good methods in order to obtain a good interleaver. A simple scrambler that flips every second code bit does a good job in many cases. However, when the graph contains a large amount of degree-2 variable nodes, some fine tuning is also necessary for the scrambler. Moreover, since these three modules interact with each other, one should do the optimization jointly instead of separately. To facilitate this work, we propose a novel concept, called low-density summation-check (LDSC) coding, for repetition-coded SM. Directly from the name, one already gets an impression that it should have a close relationship to low-density parity-check (LDPC) coding. As a matter of fact, an LDSC code is just a new way of interpreting a complete system of repetition-coded SM, including scrambling and interleaving. Its basic principle shares much commonality with that of an LDPC code.

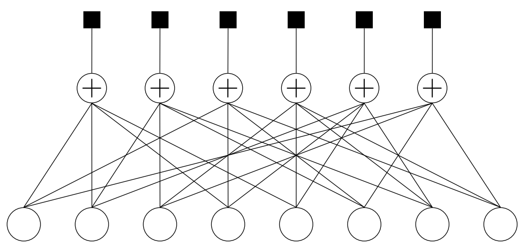


(a) Factor graph.

$$\begin{bmatrix} 1 & 0 & 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 & 0 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 & 0 & 1 & 0 & 1 \end{bmatrix}$$

(b) Incidence matrix.

Figure 6.27: Factor graph and incidence matrix of an LDPC code.



(a) Factor graph.

$$\begin{bmatrix} 1 & 1 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 & 1 & 0 & 1 \end{bmatrix}$$

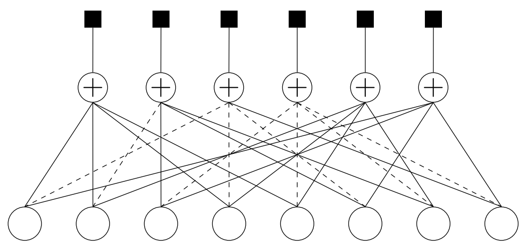
(b) Incidence matrix.

Figure 6.28: Factor graph and incidence matrix of an LDSC code.

6.4.1 Basic Principle

Consider LDPC-coded BPSK transmission over the Gaussian channel. Since one BPSK symbol carries merely one bit, each code bit from the LDPC encoder will be individually associated with a channel observation. Consequently, the overall factor graph of such a system can be drawn as in Fig. 6.27(a), where each ⊞ stands for a parity check, each ◯ stands for a code bit, and each ■ denotes a channel observation. In addition to the graphical representation, there is a more traditional but useful representation for parity-check codes. That is the incidence matrix of code bits, which is also known as the parity-check matrix of the corresponding code. For the graph in Fig. 6.27(a), the equivalent incidence matrix will be as in Fig. 6.27(b). By convention, each column of the matrix corresponds to a code bit and each row corresponds to a parity check, while the associations between code bits and parity checks are marked by nonzero entries in the matrix. Given a large block length, the density of nonzero entries will be very low, which gives the naming reason of LDPC code. As for the design of LDPC codes, both the factor graph representation and the incidence matrix representation are useful, and in fact these two concepts are often interchangeable.

Now, let us consider a repetition-coded SM system with $SF = 3$ and $N = 4$. Suppose that its factor graph is as in Fig. 6.28(a), where each variable node is connected with three summation checks due to the degree-3 repetition operations. Comparing Fig. 6.28(a) with Fig. 6.27(a), one will find that these two graphs have very similar structures. The only



(a) Factor graph.

$$\begin{bmatrix} +1 & +1 & 0 & +1 & +1 & 0 & 0 & 0 \\ 0 & -1 & +1 & 0 & 0 & +1 & +1 & 0 \\ -1 & 0 & 0 & -1 & 0 & -1 & 0 & +1 \\ 0 & 0 & -1 & 0 & -1 & 0 & -1 & -1 \\ 0 & +1 & 0 & +1 & +1 & 0 & +1 & 0 \\ +1 & 0 & +1 & 0 & 0 & +1 & 0 & +1 \end{bmatrix}$$

(b) Incidence matrix.

Figure 6.29: Factor graph and incidence matrix of an LDSC code, with scrambling.

difference is in the association with channel observations. For LDPC code, channel observations are connected with code bits, while for repetition-coded SM, channel observations are connected with summation checks. Consequently, we may follow the fashion of LDPC codes and construct an incidence matrix for the graph in Fig. 6.28(a). The resulting matrix is given in Fig. 6.28(b). Sticking with the LDPC convention, we associate each row of the matrix with a summation check and each column with a variable node. Accordingly, we also use a nonzero entry in the matrix to mark the interconnection between a variable node and a summation check. Given a large block length, this incidence matrix will be of low density as well. Hence, we may safely dub a repetition-coded SM system a low-density summation-check (LDSC) encoding system, without loss of generality.

To include scrambling in the factor graph, we can use dashed edges to represent bit flipping operations, cf. Fig. 6.29(a). In a similar way, we may use two types of nonzero entries in the incidence matrix, as shown in Fig. 6.29(b). Naturally, a "+1" stands for a non-flipped code bit, and a "−1" stands for a flipped code bit. In this way, both the factor graph and the incidence matrix embrace the complete functionality of repetition-coded SM. Note that the applied scrambling scheme in Fig. 6.29 follows the idea of variable-node-based scrambling. For practice, one does not have to align the flipping signs in a well-sorted order as in Fig. 6.29. To avoid the trap of message oscillation, one merely needs to ensure that about half of the nonzero entries in each column are negative. In this concern, the order of flipping signs makes no difference. Often it is more convenient to assign flipping signs in a random order, when the interleaver and the scrambler are designed jointly.

There are many advantages to use the concept of LDSC coding for the system optimization of repetition-coded SM. The first advantage is that length-2 cycles are inherently avoided, since an LDSC matrix cannot have two nonzero entries in the same position. The second advantage lies in the convenience for interleaver design. Borrowing the available methods from LDPC coding, one attains plenty of possibilities to improve the quality of interleaver pattern. The third advantage comes from the full representativeness of the LDSC matrix, which integrates spreading, scrambling, and interleaving. Given a well-designed spreading scheme, the task of system optimization is nothing more than a matrix construction.

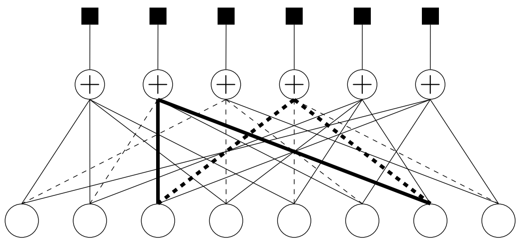
6.4.2 Computer-Based Interleaver Design

By the principle of Bayesian inference [60–64], an iterative belief propagation algorithm can only be guaranteed to converge to the maximum-likelihood solution if the underlying factor graph is cycle-free [65–69]. In reality, however, cycles are often unavoidable given a finite block length. Nevertheless, an iterative belief propagation algorithm will still work well if the cycles are long enough, which is evident from the success of Turbo codes and LDPC codes. Certainly, the larger the cycle length is, the better performance one attains. In general, the maximum possible cycle length is determined by the code structure and the block length, while given a certain code structure and a certain block length the practically achieved cycle length is solely determined by the interleaver pattern. Therefore, interleaver design is always a critical issue w.r.t. the performance of iterative decoding.

A brute-force approach for interleaver design is to try a large amount of randomly generated interleaver patterns and select the one that delivers the best performance. This approach is universal but energy-consuming, particularly when the block length is large. In the past decade, a lot of methods have been proposed for an efficient optimization of interleaver patterns, in the context of LDPC code design. Among these methods, many are computer-based controlled random approaches [70–74], while others try to utilize the available theory from finite mathematics [75–80]. Using the concept of LDSC coding, all relevant methods from LDPC coding can easily be borrowed for the optimization of interleaver patterns for repetition-coded SM. In this section, we consider two computer-based methods: MacKay's method [70] and the progressive edge-growth (PEG) algorithm [81], and check their effectiveness in improving the performance of repetition-coded SM-EPA. Together with these two algorithms, the issue of SCE distribution is investigated. Besides, we derive a randomized graph building (RGB) algorithm from the PEG algorithm.

MacKay's Matrix Construction Method

Taking the previous example given in Fig. 6.29, now we have a close look at the graph structure. Due to the small block length, this graph contains a lot of short cycles. For example, the 3rd and the 7th variable nodes together with the 2nd and 4th summation checks form a length-4 cycle, as emphasized in Fig. 6.30(a). Correspondingly, in the incidence matrix, the 3rd and the 7th columns both have nonzero entries in the 2nd and 4th rows. Alternatively, one may also detect short cycles directly from the incidence matrix. For example, in Fig. 6.30(b), the six entries surrounded by dashed boxes create a length-6 cycle, which can easily be verified by checking Fig. 6.30(a). Length-4 and length-6 short cycles are often harmful for iterative decoding, as within very few iterations the


(a) Factor graph.

$$\begin{bmatrix} +1 & +1 & 0 & +1 & +1 & 0 & 0 & 0 \\ 0 & -1 & \boxed{+1} & 0 & 0 & +1 & \boxed{+1} & 0 \\ -1 & 0 & 0 & -1 & 0 & -1 & 0 & +1 \\ 0 & 0 & -1 & 0 & -1 & 0 & \boxed{-1} & \boxed{-1} \\ 0 & +1 & 0 & +1 & +1 & 0 & +1 & 0 \\ +1 & 0 & \boxed{+1} & 0 & 0 & +1 & 0 & \boxed{+1} \end{bmatrix}$$

(b) Incidence matrix.

Figure 6.30: Cycles in the factor graph and the incidence matrix of an LDSC code.

messages passing in the graph already become non-extrinsic. Therefore, in the code-design procedure of LDPC coding, removing length-4 and length-6 cycles is typically considered as the first task. In [70], MacKay proposed a random matrix construction algorithm that guarantees the non-existence of short cycles, principally length-4 and length-6, in the corresponding factor graph. This method starts with an empty incidence matrix and adds randomly generated columns one-by-one. Before adding a new column, one checks if it will cause cycles shorter than the requirement. If it does, this column is thrown away and one tries another random generation. There is a chance that at a certain stage one fails to obtain a valid column. In this case, the whole matrix is cleaned and the procedure starts again from the very beginning. The complexity of MacKay's method depends on the length of cycles that one wants to eliminate. Certainly, the larger this length is the higher is the computational load for constructing an incidence matrix. Nevertheless, this computation load applies only once, during the design stage. Therefore, concerning the complexity of interleaver design, one only needs to ensure that it is manageable by the available computational power. Last but not least, by means of MacKay's method, the actual graph girth of the obtained matrix is unknown, i.e., it is only guaranteed to be larger than or equal to the targetted value. In most cases, however, the actual graph girth will be equal to the targetted value, since the chance to get short cycles is rather high by random matrix construction without specific controlling.

Interleaver patterns designed via MacKay's method offer desirable performances for LDPC codes, as long as the block length is not so small. However, when it comes for LDSC codes, this is not always the case, particularly when an irregular degree distribution is applied and the amount of degree-2 variable nodes is non-trivial. To see this, let us consider the system setup used in Section 6.3.3, i.e., $N = 3$ and degree distribution

$$\lambda(D) = 0.75D^2 + 0.25D^3 \; . \tag{6.9}$$

Clearly, the average spreading factor of the corresponding repetition code is given by $SF = 0.75 \times 2 + 0.25 \times 3 = 2.25$. We take a moderate block length of $K = 1500$, and we stay with the variable-node-based scrambling strategy that flips every second code bit.

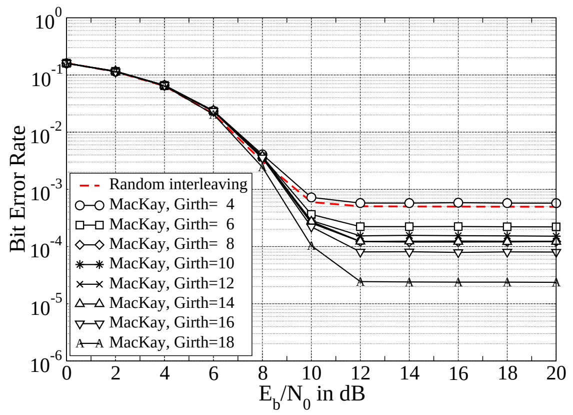
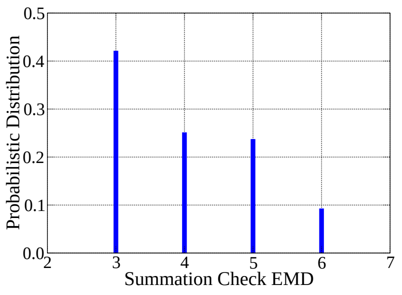
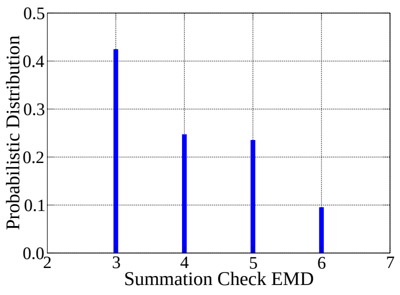


Figure 6.31: LDSC-EPA, every second code bit flipped, $N = 3$, 1500 symbols per block, 40 iterations. Variable node degree distribution: $0.75D^2 + 0.25D^3$.

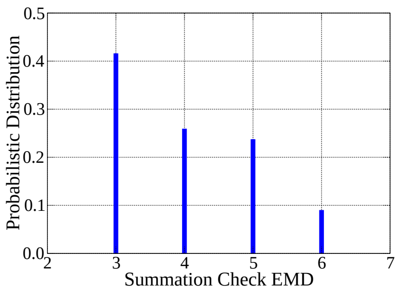
To have a reasonable expectation for the achievable performance, let us revisit Fig. 5.9(a) and Fig. 5.9(b), where the performance of regular repetition-coded SM-EPA with random interleaving is investigated. The scrambling strategy adopted therein is identical to what adopted here. One observes an error floor at 1×10^{-2} for $SF = 2$ and an error floor at 5×10^{-7} for $SF = 3$. By using an irregular repetition code with $2 < SF < 3$, one expects a performance somewhere in between that of $SF = 2$ and that of $SF = 3$. The corresponding result in Fig. 6.31 confirms this conjecture. Given random interleaving, the BER curve of the current system shows an error floor at about 5×10^{-4}, which is lower than 1×10^{-2} but higher than 5×10^{-7}. Now, by using LDSC matrices constructed via MacKay's method, we try to reduce the error floor level. With a targetted girth of 4, the interleaver designed by MacKay's method happens to perform worse than the random interleaver. This is of no surprise, because a random interleaver can from time to time acquire some nice patterns. Increasing the targetted girth step-by-step, the error floor level reduces, but not always. For example, the curves of girth 8, 12, and 14 overlap with each other, and the curve of girth 10 is even higher than that of girth 8. For the tests in Fig. 6.31, we have just made one time of random matrix construction for each targetted girth. That is we have not tried to select out a relatively good matrix from multiple random matrix constructions. We intentionally do so in order to attain an unadorned picture for the performance of MacKay's method when applied in LDSC codes. The most important message we obtain from this set of tests is that enlarging the graph girth only is not sufficient for LDSC codes. Given an irregular degree distribution, cycles can behave very differently, even with identical lengths. Note that the error floor level is still non-trivial for a girth as large as 18. For the current system, 18 is almost the maximum achievable girth. In fact, one already needs to adopt a randomized column-construction order for achieving a girth of 18. Therefore, to improve the performance further, a more advanced governing rule is necessary for the random matrix construction procedure.



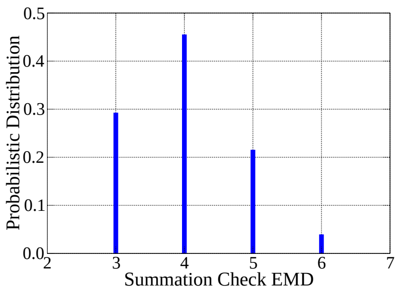
(a) Girth = 4.



(b) Girth = 6.



(c) Girth = 12.



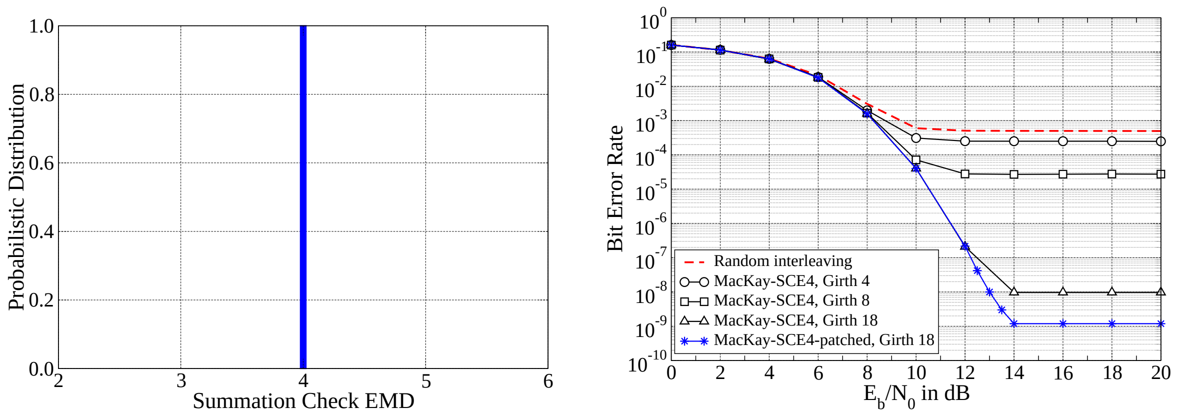
(d) Girth = 18.

Figure 6.32: SCE distribution of the LDSC matrices constructed by MacKay's method.

SCE Distribution and Degree-2 Variable Nodes

To have a good performance, an irregular LDSC code requires the graph to not only have a large girth but also good micro structures. The results in Fig. 6.31 are in fact reasonable, since MacKay's matrix construction method merely takes care of the graph girth but nothing else. In the following discussion, we will investigate the influence from the SCE distribution as well as the local graph structure of degree-2 variable nodes. Let us first check the SCE distribution of the LDSC matrices used in the tests for Fig. 6.31. The corresponding results are provided in Fig. 6.32. In the case of girth 4, more than 40 percent of summation checks are connected with pure degree-2 variable nodes, i.e., with an EMD of 3. As already mentioned in Section 6.3.3, if some isolated sub-graphs are formed among these summation checks, the iterative decoder may easily get stuck at a certain stage of message passing, because an $SF = 2$ regular repetition code can not[3] provide a good convergence for SM-EPA with $N = 3$. By raising the graph girth till 12, the SCE distribution does not change very much. The SCE distribution for girth 18 is much more concentrated on the EMD of 4, which corresponds to a connectivity given in Fig. 6.25(c). Nevertheless, this is due to a randomized column-construction order instead of a large girth. These observations reveal that MacKay's method does not introduce any explicit control on the SCE distribution. On the other hand, a concentrated SCE distribution is generally beneficial for the system performance, to be shown in the following.

[3]We will soon clarify this via EXIT chart analysis in Section 6.4.4.



(a) SCE distribution. (b) BER vs. E_b/N_0.

Figure 6.33: Irregular LDSC-EPA, every second code bit flipped, $N = 3$, 1500 symbols per block, 40 iterations. Variable node degree distribution: $0.75D^2 + 0.25D^3$.

To check how concentrated the SCE distribution of an LDSC graph can be, one needs to convert a node-perspective degree distribution into an edge-perspective EMD distribution. Note that an edge from a degree-β variable node provides an EMD of $\beta - 1$ for a summation check, and there are in total β edges diverging from a degree-β variable node. Therefore, for the VN degree distribution in (6.9), we have the equivalent edge EMD distribution as

$$\begin{aligned}\tilde{\lambda}(D) &= \frac{0.75 \times 2}{0.75 \times 2 + 0.25 \times 3}\, D^1 + \frac{0.25 \times 3}{0.75 \times 2 + 0.25 \times 3}\, D^2 \\ &= \frac{2}{3}\, D^1 + \frac{1}{3}\, D^2 \,. \end{aligned} \tag{6.10}$$

That is among all edges, a fraction of $2/3$ have an EMD of 1 and a fraction of $1/3$ have an EMD of 2. As a matter of fact, (6.10) gives a check-wise edge EMD distribution that leads to a minimized SCE distribution range. Given $N = 3$, we may let each summation check be plugged with two EMD-1 edges and one EMD-2 edges, which leads to

$$\text{SCE} = 3 \cdot \frac{2}{3} \cdot 1 + 3 \cdot \frac{1}{3} \cdot 2 = 2 \cdot 1 + 1 \cdot 2 = 4 \,, \tag{6.11}$$

i.e., connect every summation check in a way depicted by Fig. 6.25(c). This can easily be achieved during matrix construction, by enforcing each row to have exactly two nonzero entries at weight-2 columns. Adding this new strategy, we obtain a modified MacKay's method, which will be referred as "MacKay-SCE4". Fig. 6.33 provides corresponding results. The SCE distribution now merely contains a Dirac impulse at 4. Comparing Fig. 6.33(b) with Fig. 6.31, one observes that the error floor level is noticeably reduced by using the new method. For example, the error floor level is dropped from 2.5×10^{-5} to 1×10^{-8} for a targetted girth of 18. Therefore, a narrow and concentrated SCE distribution is beneficial for the high-SNR performance. This phenomenon can be explained by the

$$\left[\begin{array}{cccccc:cc}
+1 & 0 & 0 & 0 & 0 & +1 & 0 & +1 \\
-1 & +1 & 0 & 0 & 0 & 0 & +1 & 0 \\
0 & -1 & +1 & 0 & 0 & 0 & 0 & -1 \\
0 & 0 & -1 & +1 & 0 & 0 & -1 & 0 \\
0 & 0 & 0 & -1 & +1 & 0 & 0 & +1 \\
0 & 0 & 0 & 0 & -1 & -1 & +1 & 0
\end{array}\right]$$

Figure 6.34: An LDSC matrix constructed by the MacKay-SCE method with a special treatment on degree-2 VN's. $N = 3$, $\lambda(D) = 0.75D^2 + 0.25D^3$. VN-based scrambling.

statements in Section 6.1.3. By having a narrow and concentrated SCE distribution, high-degree variable nodes get evenly distributed over the whole graph. In this way, the graph achieves a high efficiency for information aggregation/distribution, since each high-degree variable node is able to help a large amount of low-degree variable nodes. Contrarily, if the high-degree variable nodes are all squeezed together, if will be difficult for the rest of the graph to benefit from the strong messages delivered by these nodes.

According to Section 6.3.2, degree-2 variable nodes are particularly sensitive to the cycle length. For the system under current discussion, whenever the degree-3 variable nodes all get correctly estimated, the remaining decoding task is equivalent to that for a regular LDSC code with $SF = 2$ and $N = 2$. In this scenario, the achievable BER performance strongly depends on the length of cycles formed by degree-2 variable nodes. For example, for a length-40 cycle consisting of degree-2 variable nodes only, the error probability will be $2^{-20} \approx 9.5 \times 10^{-7}$, according to (6.6). By using the MacKay-SCE4 method, the graph is likely to contain this type of cycles, which are not really short but harmful for the performance. To remove this type of cycles, we apply a small patch to the MacKay-SCE4 method. We generate all the columns with weight 2 in a deterministic fashion, such that the degree-2 variable nodes form a unique cycle with the largest possible length. Fig. 6.34 gives an illustrative example for this patch. It is easy to identify that we have applied one-step-cyclic-shift interleaving (cf. Section 6.3.2) for the degree-2 variable nodes. As a result, the nine degree-2 variable nodes form a unique cycle with length 12. The last three columns corresponding to degree-3 variable nodes are generated quasi-randomly under the cycle length constraint, so that the graph quality for the degree-3 variable nodes is not degraded. It is also easy to find that the EMD's of summation checks are still identical. With this patched method, we achieve for $K = 1500$ a performance as shown in Fig. 6.33(b), marked by "MacKay-SCE4-patched". The error floor level is reduced from 10^{-8} to 10^{-9}, but still nonzero. This interesting test gives two important messages. First, degree-2 variable nodes are very special. Whenever a graph contains a non-trivial amount of degree-2 variable nodes, a special carefulness should be taken. Second, there are other types of cycles contributing to the error floor, which will be discussed in Section 6.4.3.

The Progressive Edge-Growth Algorithm

In the graph terminology, the length of the shortest cycle in a graph is called the graph girth, while the length of the shortest cycle passing a variable node is called the local girth for this variable node. Concerning the performance of iterative decoding, not only the graph girth but also the local girths are important. Furthermore, the run-time local girths, which are the local girths for certain variable nodes given that the other variable nodes have been correctly estimated and virtually eliminated from the graph, play an important role for the decoding performance in the low-BER region. The observations in the previous discussion clearly support this statement. When degree-2 variable nodes exist, enlarging the cycle length among these low-degree variable nodes noticeably improves the performance of an LDSC code. Note that high-degree variable nodes typically get correct decisions earlier than low-degree variable nodes. Upon this recognition, the progressive edge-growth (PEG) algorithm proposed in [71, 81] tries to maximize the run-time local girths for each variable node and consequently reduce the error floor level. The algorithm starts from a set of non-connected variable nodes and check nodes, and build the graph by adding edges one-by-one, in a way that a low-degree variable node always gets connected earlier than a high-degree variable node. Before adding a new edge, the algorithm searches through the partially connected graph and builds up a tree by treating the variable node under operation as the root, similar to the graph piece given in Fig. 6.3. When this is done, there will be two possible scenarios. The first scenario is that the tree cannot grow further but there are some check nodes unreachable. In this case, the algorithm adds an edge between the variable node and a currently unreachable check node. As a result, the newly added edge does not create a new cycle. The second scenario is that the tree cannot grow further because it already includes all the check nodes of the graph. In this case, the algorithm adds an edge connecting the variable node to a check node locating in the farthest level of the tree. For example, if a new edge is to be added for the variable node v in Fig. 6.3, then the best check node candidates are those locating in the outermost ring, because these check nodes have the largest distance from v in the tree, and consequently will provide the largest cycle length for the new edge to be added. For both scenarios, often there will be multiple check node candidates having the best quality concerning the local girth. If this happens, the algorithm chooses a check node candidate that has the minimum connectivity in the partially connected graph. The subtle reason given by the authors in [71, 81] is that doing so will result in a graph with the check-node degrees as uniform as possible, which brings some practical benefits for LDPC codes. However, we will show that the most important impact from this treatment is in reducing the dynamic range of the check node EMD. In many cases, the check node EMD distribution largely shapes the behaviour of an iterative decoder, and is often critical for the performance.

$$\left[\begin{array}{cccccc|cc}
+1 & 0 & 0 & -1 & 0 & 0 & 0 & +1 \\
0 & -1 & 0 & 0 & -1 & 0 & +1 & 0 \\
0 & 0 & +1 & 0 & +1 & 0 & 0 & +1 \\
0 & +1 & 0 & +1 & 0 & 0 & -1 & 0 \\
-1 & 0 & 0 & 0 & 0 & -1 & +1 & 0 \\
0 & 0 & -1 & 0 & 0 & +1 & 0 & -1
\end{array}\right]$$

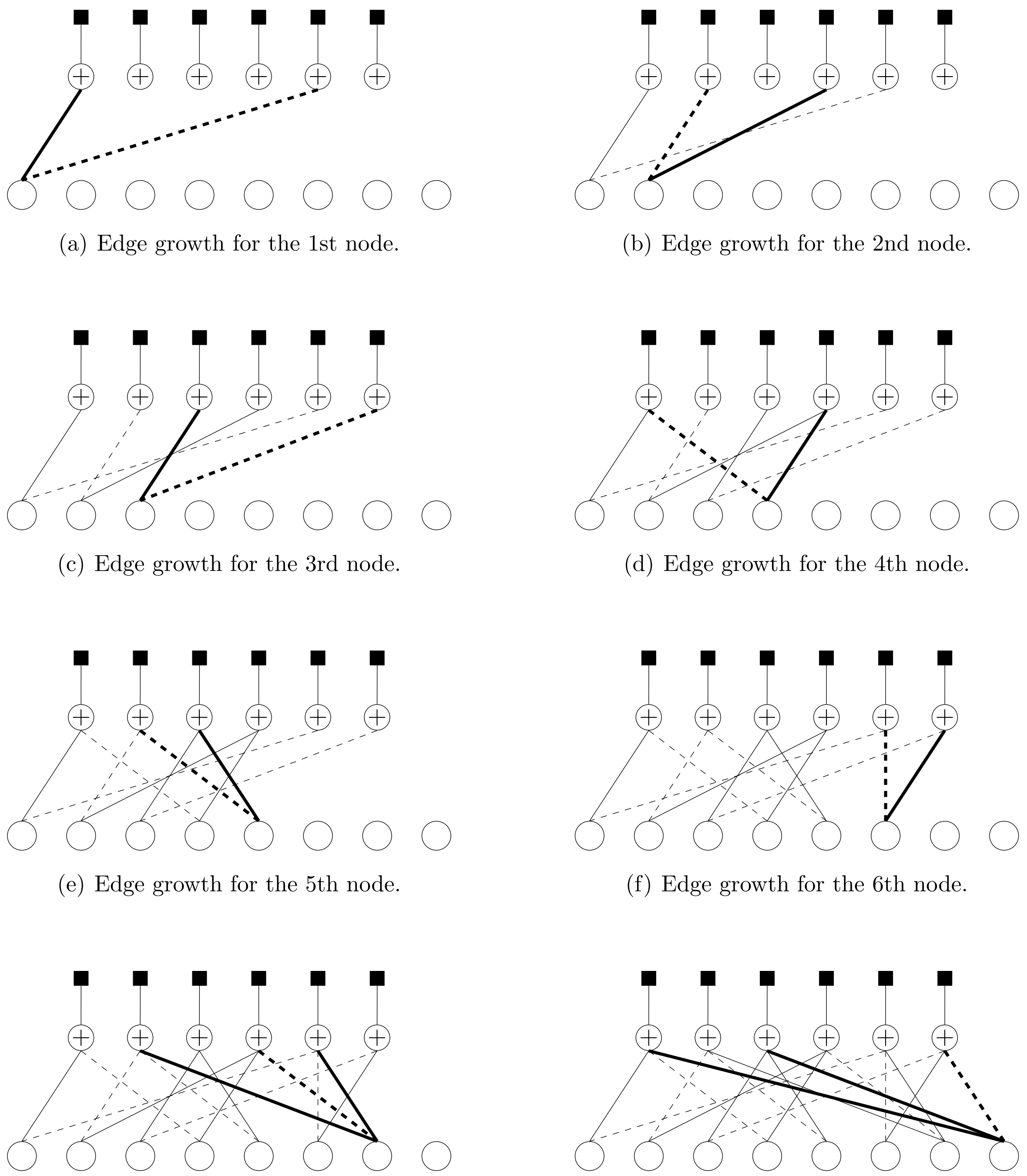
Figure 6.35: An LDSC matrix constructed by the PEG algorithm. The corresponding graph girth is 4. $N = 3$, $K = 6$, VN-based scrambling. $\lambda(D) = 0.75D^2 + 0.25D^3$.

In the early stage of edge-growth, most of the nodes are not yet connected. Hence, there is a high degree of freedom for the graph construction. Since the PEG algorithm starts with the lowest-degree variable nodes, it first tries to maximize the length of cycles among degree-2 variable nodes[4]. Given $N = 3$ and the degree distribution in (6.9), the amount of degree-2 variable nodes is exactly equal to the amount of summation checks. As a result, the PEG algorithm will also let the degree-2 variable nodes form a unique cycle, in a way equivalent to that of one-step-cyclic-shift interleaving. Fig. 6.35 gives an LDSC matrix constructed by the PEG algorithm, with all the parameters identical to that in Fig. 6.34. Though all the columns have been generated in a quasi-random way, one finds that the first six columns form a unique cycle with length 12. Clearly, compared to the patch that we added to MacKay's method, the PEG algorithm offers a more systematic solution.

Fig. 6.36 demonstrates the complete procedure for building the LDSC matrix in Fig. 6.35. One first aligns the variable nodes in a way such that their degrees are non-descending. In Fig. 6.36, degree-2 variable nodes are on the left side of degree-3 variable nodes. Hence, the PEG algorithm proceeds from left to right, in a node-by-node fashion. For adding each new edge to a variable node, one needs to select a check node such that either no new cycle is created or the newly created cycle has the largest possible length, given the partially connected graph. When there are multiple valid check node candidates, the one with the minimum current connectivity is selected. For example, from Fig. 6.36(a) to 6.36(c), a new edge is always connected to a check node which is previously not connected to any variable node. Hence, the minimum-current-connectivity-selection (MCCS) treatment enables an even spread of newly added edges among check nodes. Note that, from Fig. 6.36(d) to 6.36(f) the six newly added edges are again evenly assigned to six summation checks, and from Fig. 6.36(g) to 6.36(h) the situation is the same. As a result, all summation checks get an EMD of 4. Therefore, by using the MCCS treatment, the PEG algorithm typically results in a narrow and concentrated check node EMD distribution[5].

[4]Degree-2 variable nodes are the lowest-degree variable nodes that are relevant for cycle elimination.

[5]Here we assume a regular check node degree distribution. When the check nodes have an irregular degree distribution, the situation is often the opposite, to be discussed in Chapter 7.



(a) Edge growth for the 1st node.

(b) Edge growth for the 2nd node.

(c) Edge growth for the 3rd node.

(d) Edge growth for the 4th node.

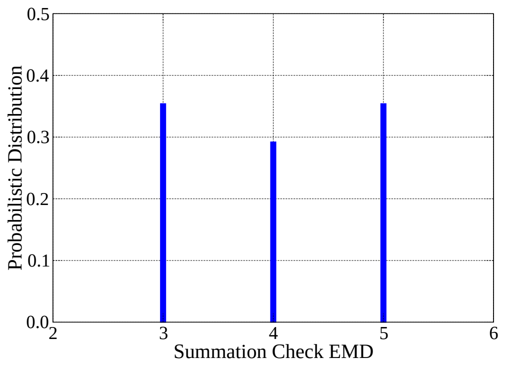
(e) Edge growth for the 5th node.

(f) Edge growth for the 6th node.

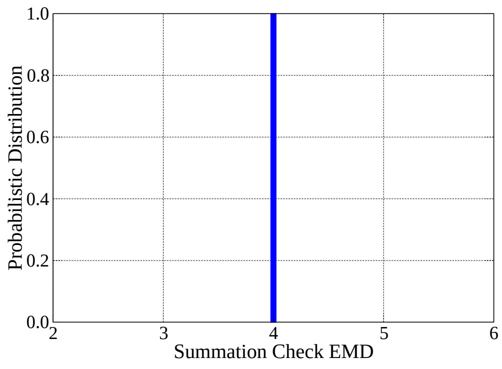
(g) Edge growth for the 7th node.

(h) Edge growth for the 8th node.

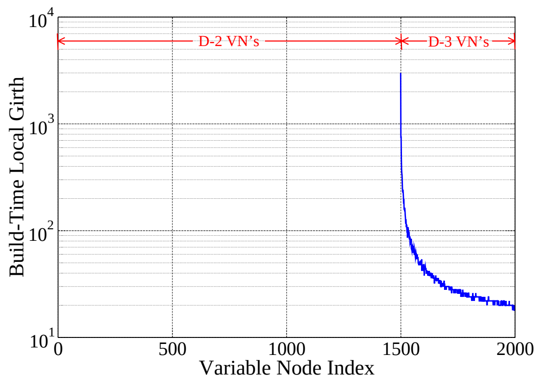
Figure 6.36: A progressive edge-growth procedure for building an LDSC code. The corresponding graph girth is 4. $N = 3$, $K = 6$, VN-based scrambling. $\lambda(D) = 0.75D^2 + 0.25D^3$. Thick lines represent the newly added edges at the current edge-growth operations.



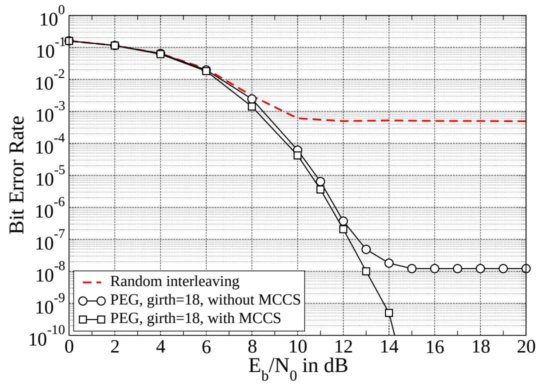
(a) SCE distribution, girth 18, w/o MCCS.



(b) SCE distribution, girth 18, with MCCS.



(c) Build-time local-girth evolution, with MCCS.



(d) BER vs. E_b/N_0.

Figure 6.37: Irregular LDSC-EPA, every second code bit flipped, $N = 3$, 1500 symbols per block, 40 iterations. Variable node degree distribution: $0.75D^2 + 0.25D^3$.

Now we check the effectiveness of the PEG algorithm by taking the same parameters as in Fig. 6.33. First, the influence of the MCCS treatment on the SCE distribution is clearly demonstrated by Fig. 6.37(a) and 6.37(b). It can be seen that using the MCCS treatment makes a big difference to the graph structure. In Fig. 6.37(c), the build-time[6] local-girth evolution for the PEG process with the MCCS treatment is provided. One observes that by adding edges for the first 1499 degree-2 variable nodes, no single cycle is created. By adding edges for the 1500th degree-2 variable node, a cycle of length 3000 is created. Afterwards, the local girths for the degree-3 variable nodes decline exponentially due to the rapidly decreasing connection freedom. By the end of the PEG process, the overall graph girth is given by 18. Fig. 6.37(d) provides the BER performances. The difference between without and with MCCS is evident. By achieving a narrow SCE distribution, the MCCS treatment avoids the overlapping of cycles formed by low-degree variable nodes, which is effective for reducing the error floor level. Applying the MCCS treatment, no error floor has been observed above 10^{-10}, as shown in Fig. 6.37(d). In later discussions, we implicitly assume the MCCS treatment for the PEG algorithm.

[6]The local girths in Fig. 6.37(c) are recorded during the graph building process. They reflect the local girths of the variable node under current operation in the partially connected graph.

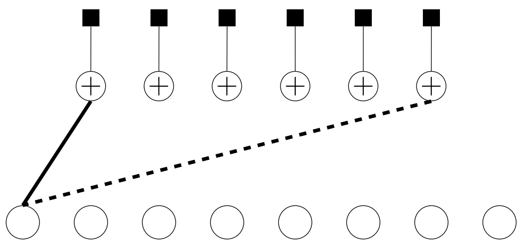
$$\left[\begin{array}{cccccc|cc}
+1 & 0 & 0 & 0 & 0 & 0 & +1 & -1 \\
0 & -1 & -1 & 0 & 0 & 0 & 0 & +1 \\
0 & 0 & 0 & 0 & -1 & -1 & +1 & 0 \\
0 & +1 & 0 & +1 & 0 & +1 & 0 & 0 \\
0 & 0 & +1 & 0 & +1 & 0 & 0 & +1 \\
-1 & 0 & 0 & -1 & 0 & 0 & -1 & 0
\end{array}\right]$$

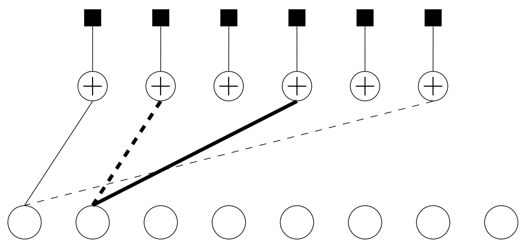
Figure 6.38: An LDSC matrix constructed by the RGB algorithm. The corresponding graph girth is 4. $N = 3$, $K = 6$, VN-based scrambling. $\lambda(D) = 0.75D^2 + 0.25D^3$.

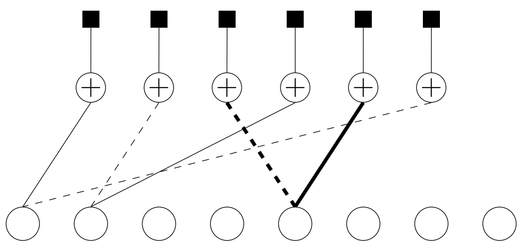
A Randomized Graph Building Algorithm

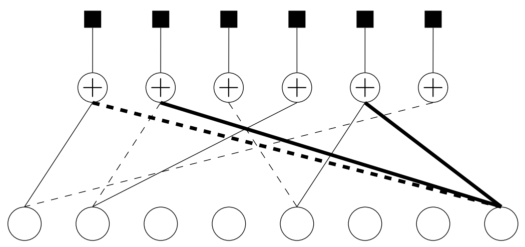
The working principle of the PEG algorithm capitalizes the following facts from iterative decoding of irregular codes. First, low-degree variable nodes are more sensitive to short cycles than high-degree variable nodes. Second, during iterative decoding, high-degree variable nodes get correctly estimated earlier than low-degree variables. Third, given that high-degree variable nodes have been correctly estimated, the decoding performance for the low-degree variable nodes is determined by the structure of the partial graph that excludes those correctly estimated high-degree variable nodes. Therefore, a graph constructed by the PEG algorithm heavily leans toward low-degree variables in the sense of structural quality, cf. Fig. 6.37(c). Such a graph is good for the performance if those high-degree variable nodes do get correctly estimated. However, in some cases, the decoding performance may also be sensitive to the graph quality of high-degree variable nodes. In such cases, a graph built by the PEG algorithm often leads to an undesirable performance.

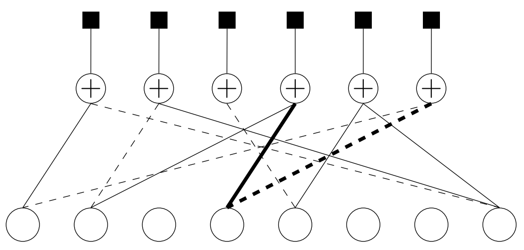
In order to achieve a balanced interleaver design, we propose a randomized graph building (RGB) algorithm, which is derived from the PEG algorithm. The RGB algorithm inherits the edge-growth mechanism from the PEG algorithm, but proceeds in a random order w.r.t. variable nodes with different degrees. Fig. 6.38 gives an LDSC matrix constructed by the RGB algorithm and Fig. 6.39 demonstrates the corresponding graph construction procedure. For each set of edge-growth operations, the RGB algorithm randomly selects a variable node that is not yet connected, regardless of the corresponding degree. To reduce cycle overlapping, the RGB algorithm also adopts the minimum-current-connectivity-selection (MCCS) treatment during edge growth. From the first snapshot, one may think that these two algorithms are rather similar. However, for systems working closely to the theoretical limit, these two algorithms lead to dramatically different performances. Due to the randomized edge-growth procedure, not all high-degree variable nodes will encounter a low-quality local graph structure. As a price, not all low-degree variable nodes will encounter a high-quality local graph structure. Last but not least, the RGB algorithm typically leads to a wide but concentrated check node EMD distribution.

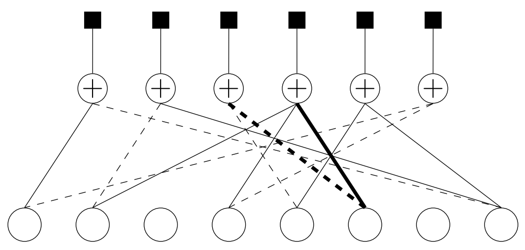

(a) The 1st set of edge-growth operations.

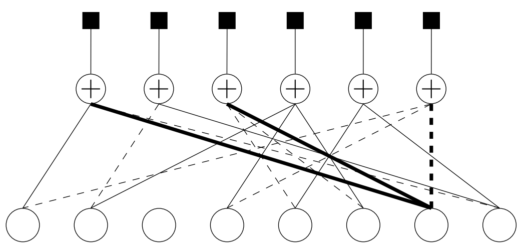

(b) The 2nd set of edge-growth operations.


(c) The 3rd set of edge-growth operations.


(d) The 4th set of edge-growth operations.


(e) The 5th set of edge-growth operations.


(f) The 6th set of edge-growth operations.


(g) The 7th set of edge-growth operations.

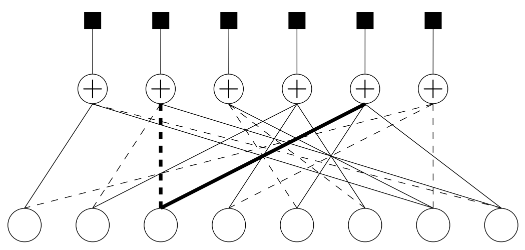

(h) The 8th set of edge-growth operations.

Figure 6.39: A randomized edge-growth procedure for building an LDSC code. The corresponding graph girth is 4. $N = 3$, $K = 6$, VN-based scrambling. $\lambda(D) = 0.75D^2 + 0.25D^3$. Thick lines represent the newly added edges at the current edge-growth operations.

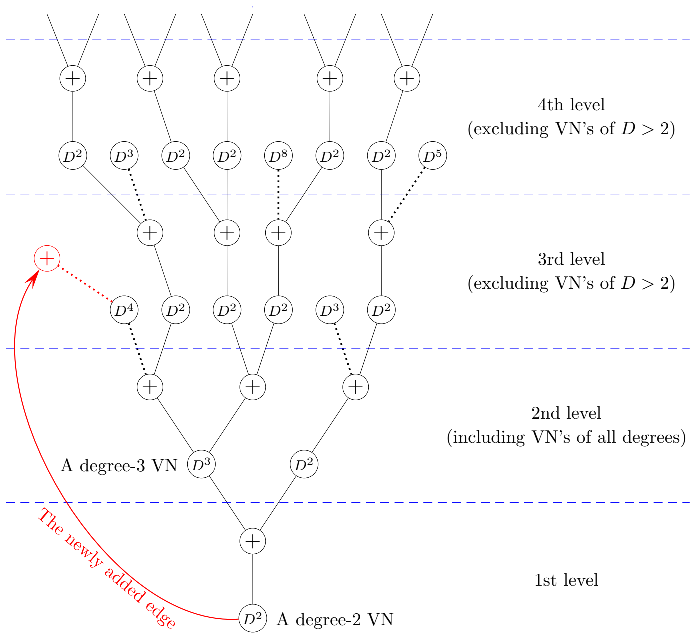
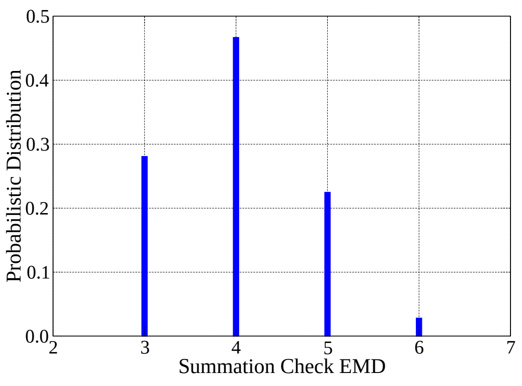
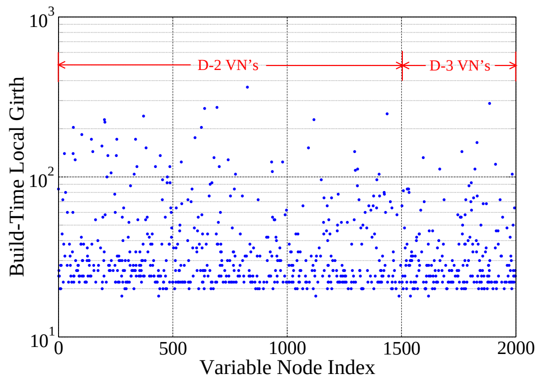


Figure 6.40: A selective tree growth procedure for a degree-2 variable node, $\Upsilon = 2$.

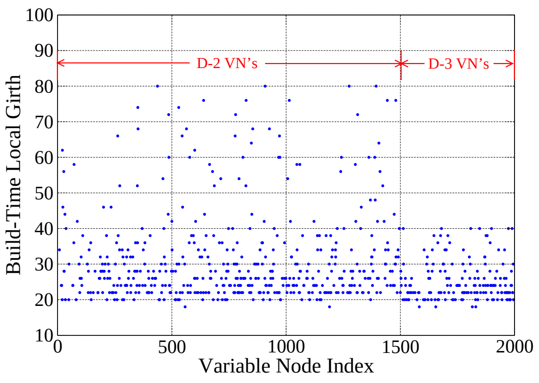
Since the RGB algorithm operates in a random order w.r.t. the VN degree, there is a non-trivial chance to form a short cycle among low-degree variable nodes, particularly by the end of graph construction when the remaining connection freedom is very limited. To solve this problem, we can take two simple but effective approaches. For an easy reference, we may call the tree growth procedure of the PEG algorithm a full tree growth (FTG). The RGB algorithm in fact treats all variable nodes in a fair way. To slightly lean the graph quality towards low-degree variable nodes, we may set finite and different tree searching depths for VN's with different degrees. By allowing a large searching depth for low-degree VN's and a small searching depth for high-degree VN's, the local graph structure of low-degree VN's will be better than that of high-degree VN's, in an average sense. We call this approach a distinct tree growth (DTG). In addition, we may apply a selective tree growth (STG) for the RGB algorithm, as illustrated in Fig. 6.40. Setting a cut-off threshold Υ, we exclude all VN's having a degree larger than that of the VN under current operation, if these variable nodes are not within Υ levels from the root. Doing so we give a higher priority for eliminating short cycles formed by low-degree VN's. For example, as shown in Fig. 6.40, a summation check is connected with a degree-4 VN in the 3rd level. By FTG or DTG, this check node should not be selected for linking the new edge, since there are other check nodes that are farther from the root. By STG, however, this check node can be selected for the new edge, because it is not connected to any degree-2 VN within $\Upsilon = 2$ levels from the root VN. The underlying principle for STG is that short cycles involving high-degree VN's are less risky than those involving low-degree VN's only. For easy illustration, we have used a rather small cut-off threshold in Fig. 6.40. In reality, a larger Υ is necessary to obtain a robust decoding performance.



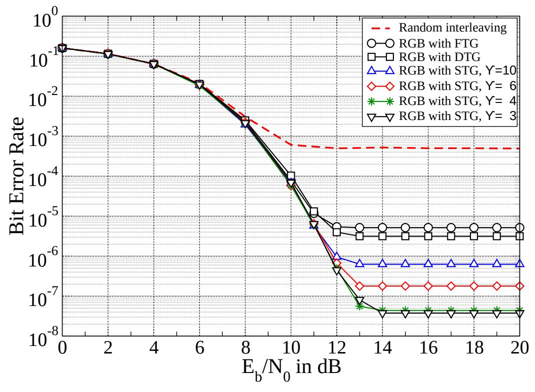
(a) SCE distribution, with MCCS.



(b) Build-time local-girth evolution, FTG.



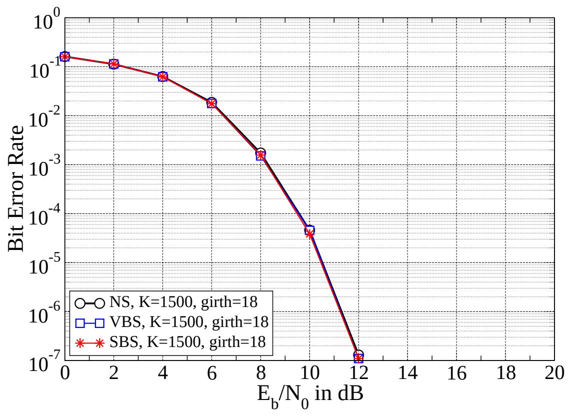
(c) Build-time local-girth evolution, DTG.



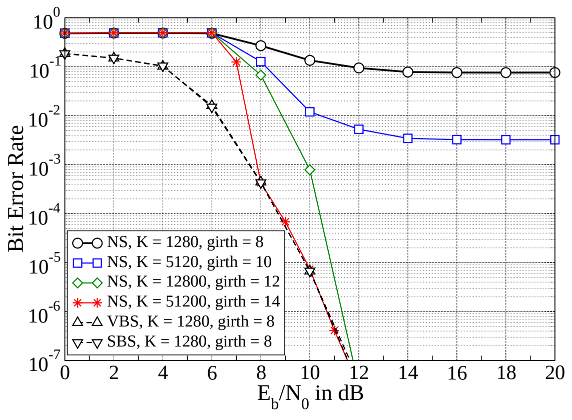
(d) BER vs. E_b/N_0.

Figure 6.41: Irregular LDSC-EPA, every second code bit flipped, $N = 3$, 1500 symbols per block, 40 iterations. Variable node degree distribution: $0.75D^2 + 0.25D^3$.

Using the MCCS treatment, the RGB algorithm typically results in an SCE distribution that is concentrated but not necessarily narrow, as shown in Fig. 6.41(a). Nevertheless, this is an advantage instead of disadvantage, to be discussed in Section 6.4.5. With FTG, low-degree VN's and high-degree VN's are treated fairly concerning cycle lengths, cf. Fig. 6.41(b). With DTG, low-degree VN's have a good chance to have larger local girths. In case of Fig. 6.41(c), the searching depth for degree-2 VN's is 40 and the searching depth for degree-3 VN's is 20. Note that, linking an edge to a check node in the 40th level creates a cycle of length 80. Fig. 6.41(d) shows that applying DTG brings a noticeable reduction in the error floor level, w.r.t. FTG. Now we apply STG, and still use a searching depth of 40 for degree-2 VN's and 20 for degree-3 VN's. One observes from Fig. 6.41(d) that this reduces the error floor level further. Besides, it can be seen that a smaller cut-off threshold leads to a lower error floor level. In the current case, $\Upsilon = 4$ and $\Upsilon = 3$ give the best performances. As a matter of fact, these two threshold values are the best choices for most applications. Since Υ trades off the graph quality for low-degree VN's and that for high-degree VN's, $\Upsilon = 2$ is a relatively risky choice, in the sense of causing a non-trivial probability for burst errors when N is large. Last but not lease, one can easily reduce the error floor of the current code design to a negligible level by increasing the block length.



(a) $N = 3$, $\lambda(D) = 0.75D^2 + 0.25D^3$.



(b) $N = 5$, $\lambda(D) = 0.80D^3 + 0.20D^4$.

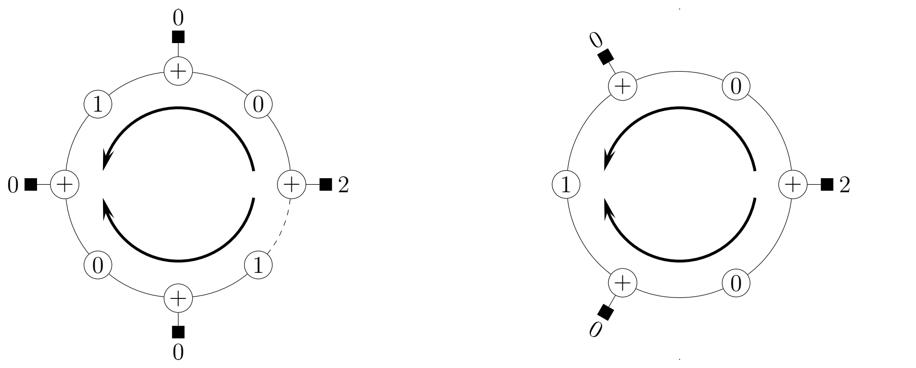
Figure 6.42: Irregular LDSC-EPA, PEG-designed interleavers, 40 iterations.

6.4.3 Computer-Based Scrambler Design

In Section 6.4.2, we have shown that for achieving a low error floor level the run-time local girths of low-degree variable nodes should be large enough. However, this is not always achievable, due to many possible reasons. In this section, we devise a scrambling strategy that can reduce residual errors when the decoding process is almost accomplished. For a systematic study, the discussion starts with some numerical assessments on scrambling.

The Influence of Scrambling w.r.t. Graph Girth

In Section 6.2.1, we conjecture that a graph with a very large girth will be immune to the trap of message oscillation. Now we try to verify this conjecture via numerical tests. We consider LDSC-EPA with $N = 3$ and $N = 5$. Typically, a higher bit load leads to a smaller graph girth, given a fixed block length. Fig. 6.42(a) provides the BER performances for the case of $N = 3$, given no scrambling (NS), VN-based scrambling (VBS), and SC-based scrambling (SBS). In fact, no performance difference is observed w.r.t. different scrambling schemes. Given a small bit load and a large girth as 18, the iterative decoder is indeed insensitive to scrambling. Fig. 6.42(b) provides the BER performances for the case of $N = 5$. Due to a higher bit load, the iterative decoder becomes sensitive to scrambling. Given no scrambling and a girth $\leqslant 10$, the error floor level is considerable, which is caused by burst errors resulting from message oscillations. One also observes that the error floor level can be reduced by increasing the graph girth. This verifies the conjecture. After all, Fig. 6.42(b) shows that scrambling is much more efficient than a large girth in the sense of avoiding the trap of message oscillation. Note that even for a girth of 14, there is a non-trivial performance penalty in the low-SNR region when scrambling is not applied.



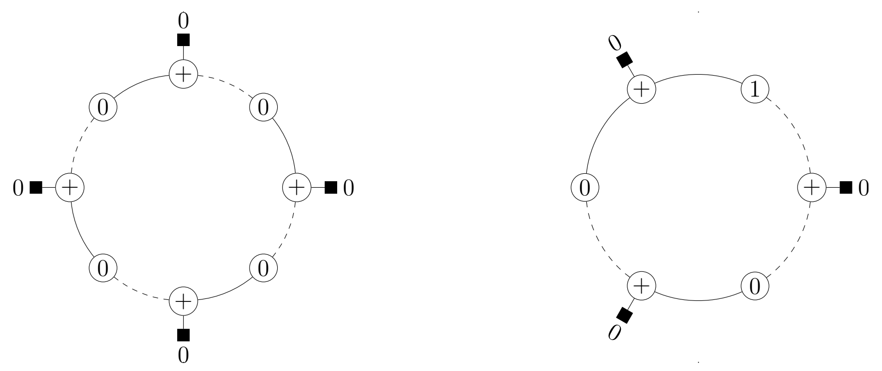
(a) Flip an odd number of edges. (b) Flip an even number of edges.

Figure 6.43: Scrambling strategies for cycles containing an even or odd number of SC's.

A Heuristic Example for Scrambler Design

In most of the previous discussions, we have used a VN-based scrambling strategy, which flips every second edge diverging from a VN. This approach is appropriate and good, but not necessarily the best that we can do, particularly for degree-2 VN's. Fig. 6.42 actually gives two meaningful hints for scrambler design. First, the primary effect of scrambling for high-degree VN's is in avoiding the trap of message oscillation. In this concern, a VN-based scrambler does the job very well. Second, degree-2 VN's are usually irrelevant to message oscillations. This has two reasons. By an appropriate interleaver design, the local girths of degree-2 VN's are often much larger than that of high-degree VN's. Besides, a degree-2 VN does not have a message-sign-unification functionality (cf. Section 6.2.1), as it simply exchanges the two messages that it receives. Hence, scrambling is in fact unnecessary for degree-2 VN's if it is for avoiding the trap of message oscillation.

During iterative LDSC decoding, VN's with the highest degree get correctly estimated first and degree-2 VN's get correctly estimated last. For this reason, cycles formed by pure degree-2 VN's are usually the major cause for the residual errors. As scrambling is not mandatory for degree-2 VN's, we may use it in a new way. Illustrated in Fig. 6.43, we flip one and only one edge for a cycle containing an even number of summation checks (SC's), and we do not flip any edge if the cycle contains an odd number of SC's. Assuming a noiseless channel, a cycle formed by pure degree-2 VN's will produce decision errors if and only if all SC's come with a zero channel observation. Without much difficulty, one finds that the two cycles given in Fig. 6.43 are free of such scenarios, i.e., there will be at least one SC coming with a non-zero observation. Based on this observation, we can easily generalize the scrambling strategy. For the case of Fig. 6.43(a), the error probability will be zero as long as the number of flipped edges is odd. For the case of Fig. 6.43(b), the error probability will be zero as long as the number of flipped edges is even. In fact, such a scrambling strategy tries to eliminate those risky stopping sets from an LDSC graph.



(a) An even number of edges are flipped. (b) An odd number of edges are flipped.

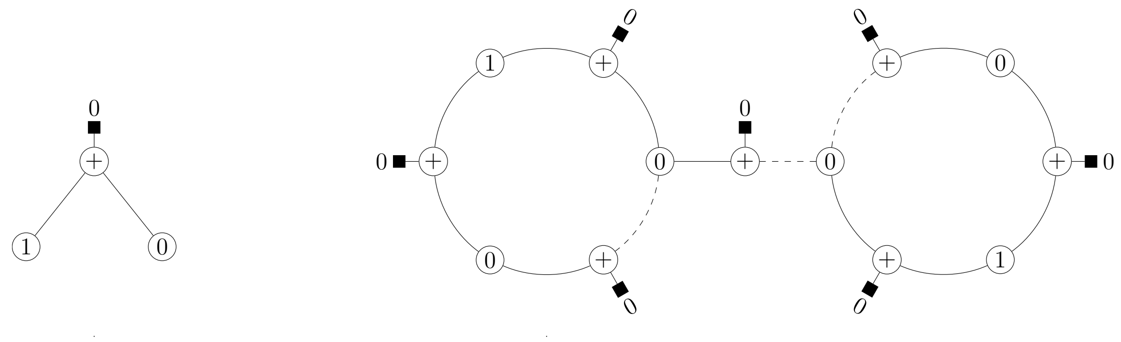
Figure 6.44: Two stopping sets formed by degree-2 variable nodes.

Stopping Sets for LDSC-EPA

The issue of stopping sets is first discovered in the field of LDPC-coded transmission over the binary erasure channel. When the channel erasures include a group of variable nodes that form a stopping set, iterative decoding fails [82]. For LDPC-coded transmission over the AWGN channel, the related concept is trapping sets [83]. Now for LDSC-EPA, there exists stopping sets both over the noiseless channel and the AWGN channel. We define a stopping set for LDSC-EPA as an ensemble of variable nodes and summation checks that fulfill the following conditions:

1. Every summation check is connected to an even number of variable nodes, with this number being larger than or equal to 2;
2. Every variable node is connected with at least one summation checks within this ensemble and no summation check outside this ensemble;
3. For every cycle formed within this ensemble, the number of flipped edges is even if this cycle contains an even number of summation checks, and the number of flipped edges is odd if this cycle contains an odd number of summation checks.

By applying some changes on the scrambling patterns in Fig. 6.43, we obtain two stopping sets formed by degree-2 variable nodes, as illustrated in Fig. 6.44. According to Section 6.3.2, the stopping set in Fig. 6.44(a) has an error probability of 2^{-4}, and the one in Fig. 6.44(b) has an error probability of 2^{-3}, assuming a noiseless channel. A stopping set is characterized by a non-zero probability to have the relevant chip summations being all-zero. For example, given the specific bit values marked in Fig. 6.44, the summation checks in these two stopping sets all come with a zero channel observation. Alternatively, this is to say that a stopping set is characterized by a non-bijective mapping rule between the info bits and the chip summations encompassed by this set. Note that the two cycles in Fig. 6.44 are always stopping sets, regardless of the degree of the summation checks.



(a) A $(2,1)$ stopping set.

(b) A $(6,7)$ stopping set.

Figure 6.45: Two stopping sets formed by degree-1, degree-2, and degree-3 variable nodes.

For shorthand notations, we say a stopping set is a (m, n) stopping set if it contains m variable nodes and n summation checks. A stopping set does not have to contain cycles. For example, the $(2,1)$ stopping set in Fig. 6.45(a) actually gives the smallest stopping set for LDSC-EPA, which occurs when the spreading factor is 1. The error probability of this stopping set is 1/4, cf. Fig. 4.1. Roughly speaking, the smaller a stopping set is, the higher is the error probability. Hence, when an iterative LDSC-EPA decoder produces an error floor, this error floor is mostly contributed by those small stopping sets. As shown in Fig. 6.43, a clever scrambling scheme can help to eliminate some stopping sets. Hence, by enumerating those risky stopping sets and sequentially eliminating them via scrambling, a significant reduction can be expected for the error floor level. Let us still consider the system setup: $N = 3$ and $\lambda(D) = 0.75D^2 + 0.25D^3$, which is used as an illustrative example in Section 6.4.2. Suppose the graph girth is 6, such that no cycles with a length smaller than 6 exist. Given this assumption, the most risky stopping set is actually the one in Fig. 6.44(b), and the stopping set in Fig. 6.44(a) actually gives the secondary risky one. By applying the scrambling strategy given in Fig. 6.43, these stopping sets formed by pure degree-2 variable nodes can effectively be eliminated. In general, a stopping set containing high-degree variable nodes has a larger size than a stopping set containing low-degree variable nodes only, given the same minimum cycle length. For example, the $(6,7)$ stopping set in Fig. 6.45(b) contains two length-6 cycles but only has an error probability of 2^{-6}, which is much lower than that of Fig. 6.44(a) and Fig. 6.44(b). Easy to imagine, by replacing one degree-2 variable node in Fig. 6.45(b) by a degree-3 variable node, the stopping set will include more summation checks and consequently lead to a lower error probability. This explains why a stopping set containing many high-degree variable nodes is not relevant to the error floor. Hence, for reducing the error floor level, those stopping sets that need to be eliminated are the ones formed mainly by degree-2 variable nodes. Following this observation, we devise in the next step a cycle-based scrambling strategy.

A Cycle-Based Scrambling Strategy

A cycle-based scrambling strategy aims to eliminate those risky stopping sets in an LDSC graph via avoiding a non-bijective mapping rule in certain short cycles. We define the weight of a cycle as the summation of the degree of the variable nodes involved in this cycle. By this definition, the weight of a cycle is equivalent to the maximum Hamming distance between the repetition codewords corresponding to this cycle. For example, the cycle in Fig. 6.44(a) has a weight of 8, the cycle in Fig. 6.44(b) has a weight of 6, and the two cycles in Fig. 6.45(b) both have a weight of 7. In general, a low-weight cycle is more risky than a high-weight cycle, because on average the size of the stopping set that a low-weight cycle belongs to is smaller than the size of the stopping set that a high-weight cycle belongs to. Fig. 6.44(b) and Fig. 6.45(b) give a good example for such a situation. A cycle-based scrambling (CBS) strategy decides the scrambling pattern as follows.

1. Construct an LDSC matrix via a certain algorithm, and meanwhile apply VN-based scrambling for all VN's (optionally one may exclude all degree-2 VN's).
2. Perform cycle searching for a certain VN. During the cycle searching, only those VN's that have a degree smaller than or equal to the degree of the current VN are considered. Record the smallest-weight (usually not the shortest) cycle that the current VN is involved. For this cycle, change the flipping sign for one of the edges diverging from the current VN if the cycle contains an even number of summation checks and an even number of flipped edges or if the cycle contains an odd number of summation checks and an odd number of flipped edges. Lock all edges involved in this cycle and forbid any further change on the flipping sign of these edges. If desired, one may search for a cycle whose weight is larger than the previous one but smaller than all the others, and so on. Nevertheless, flipping the edges diverging from a degree-D VN can at most eliminate $D-1$ stopping sets.
3. Execute the above operation for all VN's, in an order that a low-degree VN always gets treated earlier than a high-degree VN.

Note that the above scheme actually makes some adjustments on a VN-based scrambling pattern, whenever it is necessary. Statistically, the amount of flipped edges is still approximately equal to the amount of non-flipped edges in this graph, after these adjustments. Hence, doing so will not increase the risk of message oscillation. The effectiveness of this scrambling strategy will be verified by the numerical results provided in Section 6.4.5. In practice, it is often sufficient to apply CBS for VN's with a degree smaller than or equal to 3. Certainly, it does not harm if one applies CBS for all VN's, but it will not provide any noticeable performance improvement. Moreover, it is usually not feasible to fully rely on scrambling to enable a perfect data separation, to be explained in the following.

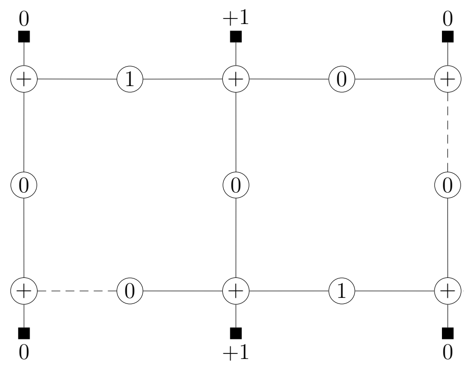


Figure 6.46: A graph piece that contains a $(6,6)$ stopping set.

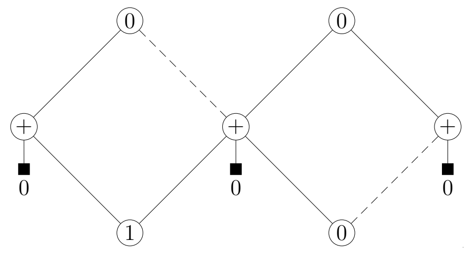


Figure 6.47: A $(4,3)$ stopping set.

As a matter of fact, there are many types of stopping sets that are not removable by means of clever scrambling. To illustrate this situation, let us consider two examples. Fig. 6.46 gives a graph piece that contains a $(6,6)$ stopping set. Checking this graph piece carefully, one may find that the single flipped edge in the left cycle prevents this cycle from forming a $(4,4)$ stopping set, and so does the single flipped edge in the right cycle. Since these two cycles are the smallest-weight cycles within this graph piece, the current scrambling pattern is in fact optimal. Nevertheless, it is easy to find that the nodes in the outer ring form a $(6,6)$ stopping set. It is not possible to remove this stopping set without changing one of the two length-8 cycles into a stopping set. This observation actually implies that a regular $SF = 2$ repetition code can not support SM-EPA with $N = 3$. Fig. 6.47 provides a $(4,3)$ stopping set as the second example. Easy to find, this stopping set contains two length-4 cycles, none of which forms a stopping set due to clever scrambling. However, as these two cycles join each other at a summation check, a $(4,3)$ stopping set is formed. It can be seen from Fig. 6.47 that the channel observations are all zeros given the bit values as marked in the graph. With some simple derivations, one finds that the error probability of this stopping set is given by 2^{-4}, assuming a noiseless channel. Note that it is strictly not possible to avoid such a stopping set by means of scrambling. There are in general two possible approaches to reduce the errors from the above two types of stopping sets. The first approach is to forbid a summation check to be connected with more than two degree-2 variable nodes, so that graph pieces as in Fig. 6.46 and Fig. 6.47 will not exist. The second approach is to enlarge the cycle length by using a larger block length. Both approaches are useful and effective, to be demonstrated in Section 6.4.5.

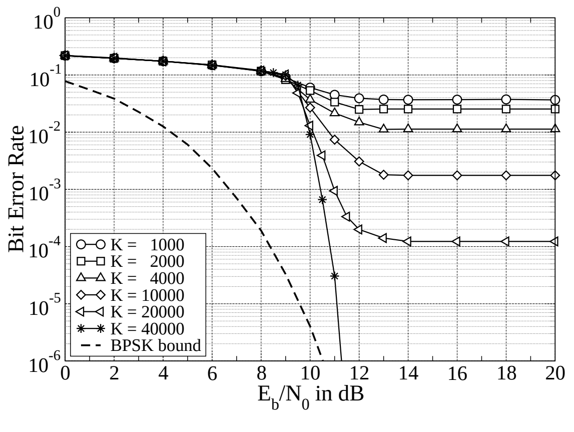


Figure 6.48: Regular LDSC-EPA, $SF = 3$, $N = 6$, PEG-designed interleavers, 100 iterations. K denotes the number of symbols per block.

6.4.4 Supportable Rate with Regular Repetition

Given random interleaving, the supportable rate of regular repetition-coded SM-EPA is slightly less than 2 bits/symbol, cf. Section 5.3.2. It is interesting to check whether this situation can be improved by using the framework of LDSC coding and the interleaver design method, namely the PEG algorithm, described in Section 6.4.2. Note that the RGB algorithm is equivalent to the PEG algorithm for regular LDSC coding.

Fig. 6.48 shows the performance of regular LDSC codes with equal power allocation, for a spreading factor of $SF = 3$ and a bit load of $N = 6$. One observes that, the error floor levels for moderate block lengths are still significant, even for interleavers designed via the PEG algorithm. By increasing the block length to $K = 40000$, the error floor finally diminishes. Note that for regular LDSC codes a graph built by the PEG algorithm is almost optimal, if not exactly. Still, a very large block length is necessary to achieve a decoding convergence. This scenario reminds one that the data rate of 2 bits/symbol might be closely related to a certain type of theoretical limit for regular LDSC-EPA.

For an iterative LDSC decoder, a prerequisite for convergence is that the variable nodes and the summation checks interact with each other in a mutually beneficial way, during iterations. In detail, this is to guarantee that given the extrinsic information delivered by the variable nodes in the current iteration, the summation checks will be able to produce a stronger extrinsic information to the variable nodes w.r.t. the extrinsic information produced in the previous iteration. Only then, will iterations bring potential performance gain. S. ten Brink proposed in [84–86] a novel method, called extrinsic information transfer (EXIT) chart, to visualize the fitness between a pair of iterative decoding modules. This method is nowadays very popular for the design and analysis of iteratively decodable channel codes. In the context of LDSC coding, an EXIT chart analysis is to check the

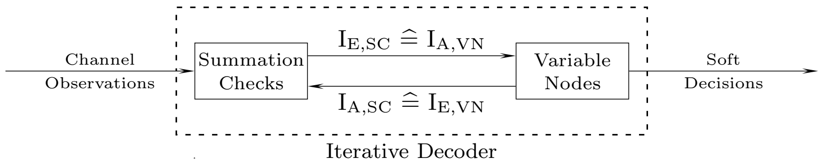


Figure 6.49: The interaction between summation checks (SC) and variable nodes (VN).

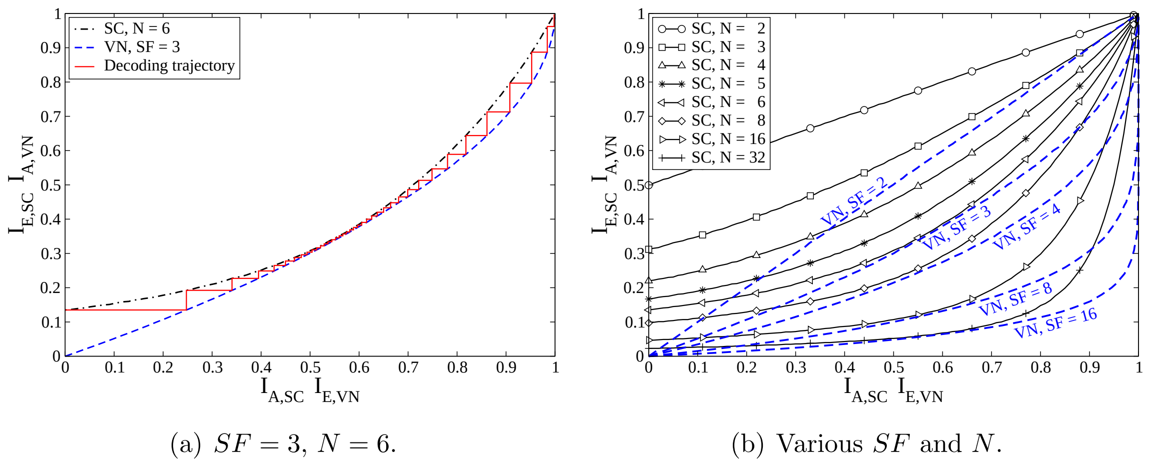
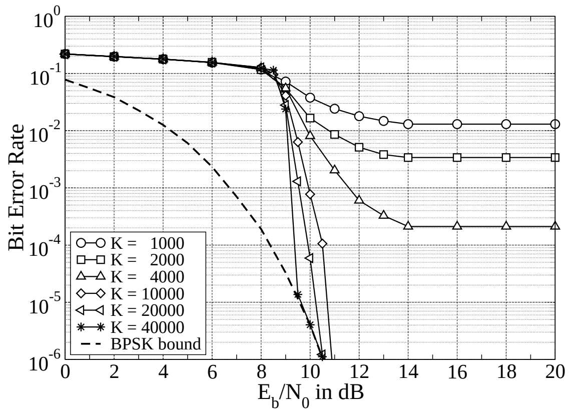


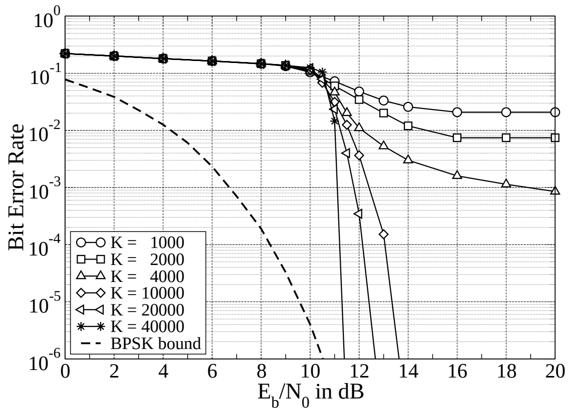
Figure 6.50: EXIT charts for regular LDSC-EPA transmission over a noiseless channel.

extrinsic information transfer property of the summation check ensemble and that of the variable node ensemble. As shown in Fig. 6.49, the extrinsic information ($I_{E,SC}$) outputting from the summation checks correspond to the a priori information ($I_{A,VN}$) inputting to the variable nodes, and vice versa. The major contribution of ten Brink in [84] is in proposing the mutual information between LLR messages and code bits as a metric to measure the strength of extrinsic information exchanging between the iterative decoding modules. This metric proves to be stable and practical for many types of iterative decoders and many types of channels. Fig. 6.50 gives two EXIT charts[7] for regular LDSC-EPA. We have assumed a noiseless transmission channel so as to check the ultimate performance limit. To achieve a decoding convergence, the EXIT curve of the VN ensemble must be below that of the SC ensemble. Only then will a tunnel for iterative message enhancing be open. Fig. 6.50(a) shows that the convergence tunnel is open for $SF = 3$ and $N = 6$ but almost closing in the middle region. Hence, it will take many iterations for an iterative decoder to make a way through the middle section of the tunnel. Since an EXIT chart analysis assumes a cycle-free factor graph, the performance predicted by it is strictly achievable only for an infinite block length. This explains why one needs a block length as large as $K = 40000$ in order to achieve a decoding convergence in the previous test, cf. Fig. 6.48. More EXIT curves are provided in Fig. 6.50(b). It can be seen that given $SF = 2$, the

[7]The EXIT curves in this figure have been obtained by using the simplified mutual information calculation method proposed by Land *et al.* [87–89]. This method is equivalent to the one proposed by S. ten Brink in [84] for optimal APP demapping and decoding, but more convenient.



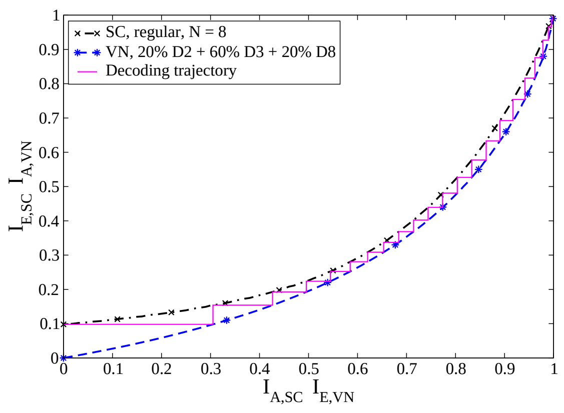
(a) $SF = 4$, $N = 8$.



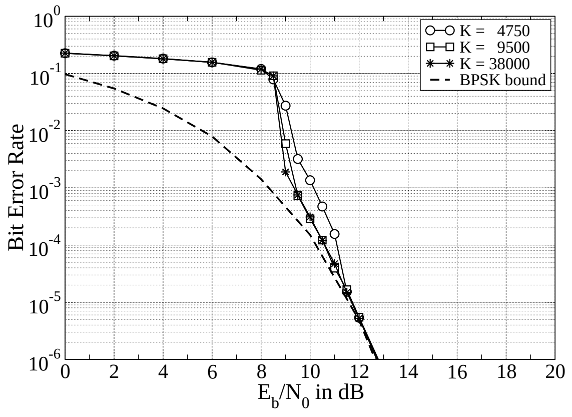
(b) $SF = 8$, $N = 16$.

Figure 6.51: Regular LDSC-EPA, PEG-designed interleavers, 100 iterations. K denotes the number of symbols per block.

convergence tunnel is only open for $N \leqslant 2$. Hence, the maximum supportable rate for $SF = 2$ is merely 1 bit/symbol, which complies with the results in Section 5.3.2 and Section 6.3.2. On the other hand, for $SF \geqslant 3$, the convergence tunnel will be open for all $N \leqslant 2 \cdot SF$. This well explains the 2 bits/symbol rate limit for regular LDSC-EPA. It can also be seen from Fig. 6.50(b) that the middle section of the convergence tunnel for $SF = 4$ and $N = 8$ is wider than that for $SF = 3$ and $N = 6$. Besides, the situation of $SF = 8$ and $N = 16$ is just somewhere in between the former two cases. This phenomenon is clearly reflected in the corresponding BER performances provided in Fig. 6.51, in the sense of minimum required block length for decoding convergence. Besides, from Fig. 6.48 and Fig. 6.51 one also observes that the BER curve of $N = 8$ approaches the asymptotic BPSK bound earlier than that of $N = 6$ and $N = 16$, given $K = 40000$. This is also due to the difference of the tunnel width in the middle section. So far, we may conclude that 2 bits/symbol is achievable for regular LDSC-EPA, given a large enough block length and a well-designed interleaver.



(a) EXIT chart, noiseless channel.



(b) BER vs. E_b/N_0.

Figure 6.52: Irregular LDSC-EPA, $SF = 3.8$, $N = 8$, PEG, 100 iterations.

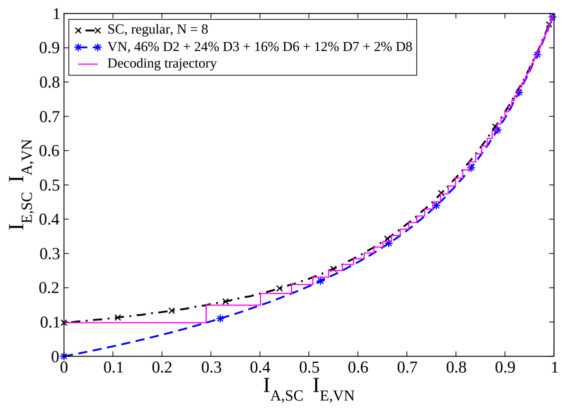
6.4.5 Supportable Rate with Irregular Repetition

In many related works [27,37–39,41,42], a 2 bits/symbol rate limit always exists for coded SM-EPA transmission, either explicitly or implicitly mentioned therein. It is commonly assumed by relevant researchers that a higher bandwidth efficiency is only possible by using unequal power allocation for the superposition mapper. In this section we will show that this presumption is in fact not precise. With a carefully designed irregular repetition code, the supportable rate of SM-EPA can easily go beyond 2 bits/symbol.

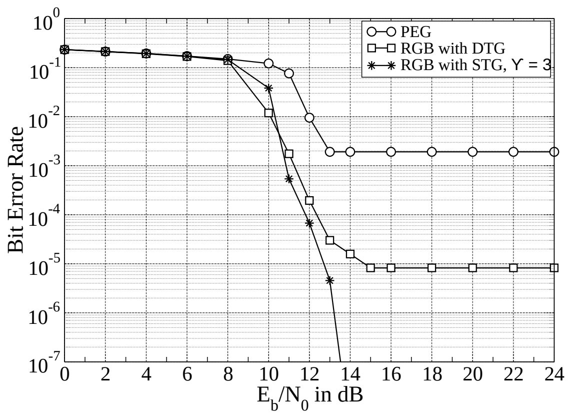
From Fig. 6.50(b), we see that the convergence tunnel is very narrow in the middle region but unnecessarily wide in the right region, for $N = 2 \cdot SF$ with $SF \geqslant 3$. In other words, the EXIT curve of an SM-EPA demapper and that of a regular repetition decoder do not really fit. According to the area property of EXIT charts [90–92], any area between two EXIT curves leads to a rate loss relative to the capacity. In the context of Fig. 6.50(b), where a noiseless channel is assumed, the capacity is given by the SM-EPA symbol entropy. For example, the capacity is about 2.54 bits/symbol for $N = 8$, cf. Tab. 3.3. By a careful code design, rates more than 2 bits/symbol should be achievable. A natural solution is to reduce the surplus of the convergence tunnel in the right region. Fig. 6.52 provides the performance for LDSC-EPA with a well-tuned degree distribution. The average spreading factor is $SF = 3.8$, which leads to a rate of 2.1 bits/symbol. One observes that the decoder has no problem to converge. Hence, the rate limit of 2 bits/symbol is successfully broken. Besides, as the convergence tunnel is widely open for the whole region, the decoder also becomes less sensitive to the block length, cf. Fig. 6.52(b). Now let us revisit Fig. 6.2. The two irregular degree distributions considered therein both have an average spreading factor of 4, which is in fact higher than that in Fig. 6.52. Hence, the corresponding error floors are caused by the imperfectness of random interleaving given a finite block length.

Revisiting Fig. 6.52(a), one finds that there is still space to reduce the spreading factor. Motivated by this observation, we apply a VN degree distribution with $SF = 3.6$, which leads to a rate of $8/3.6 \approx 2.22$ bits/symbol. Fig. 6.53(a) gives the corresponding EXIT chart. Since the convergence tunnel becomes very tight in the rightmost region, it takes 309 iterations for the decoding trajectory to reach the point at $(1, 1)$. To avoid the risk of insufficient iterations, we increase the iteration number to 400 for the BER performance test. Let us first focus on the performance resulting from the PEG-designed interleaver. One observes from Fig. 6.53(b) that a significant error floor exists even for a block length as large as $K = 45000$. By a careful investigation on the error pattern, we found that at high SNR's the majority of the data blocks comes with completely error-free decisions while a few data blocks come with decisions full of errors. Hence, the error floor is caused by burst errors. In Fig. 6.53(c) to 6.53(e), the LLR density evolution processes for three selected data blocks are demonstrated. These three data blocks all attain error-free decisions by the end of iterative decoding. The measured probability density function (PDF) shows the evolution of the distribution of the LLR messages (for info bits) w.r.t. iterations, given a noiseless channel. Easy to understand, the PDF is always bi-Gaussian in the initial iteration. As soon as a good decoding convergence is achieved, the PDF becomes a multiple[8]-Dirac function. Vividly shown by Fig. 6.53(c) to 6.53(e), the actually required number of iterations varies dramatically from block to block. For block 1, the decoding convergence is achieved after about 60 iterations. For block 2, this is after about 100 iterations. For block 3, more than 250 iterations are necessary to achieve the decoding convergence. The reason behind this phenomenon is that the amount of information that those moderate-size stopping sets obtain from the channel vary dramatically from block to block. We will give an in-depth discussion on this issue in Section 7.1.3. With a non-trivial chance, certain stopping sets receive all zeros for the relevant channel observations and fail to deliver reliable messages. If these stopping sets involve many high-degree variable nodes, a situation as in Fig. 6.53(f) occurs, which leads to a burst of decision errors. This observation raises a critical question for the PEG algorithm. By starting from low-degree variable nodes and ending at high-degree variable nodes, the PEG algorithm maximizes the size of stopping sets involving low-degree variable nodes only but sacrifices the size of stopping sets involving high-degree variable nodes. In other words, it reduces the probability of residual errors but increases the probability of burst errors. For LDPC-coded BPSK transmission over the AWGN channel, this is rarely an issue, because each code bit is associated with a channel observation free of interference. For LDSC codes, however, this becomes a problem whenever the targetted rate is close to the capacity.

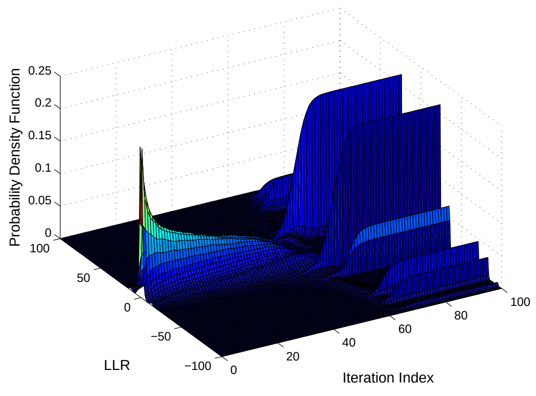
[8]For the sake of numerical stability, the summation check output messages are limited to be in $[-12, 12]$. As a result, the strongest LLR messages have a magnitude of 96, delivered by those degree-8 variable nodes, and the weakest messages have a magnitude of 24, delivered by those degree-2 variable nodes.



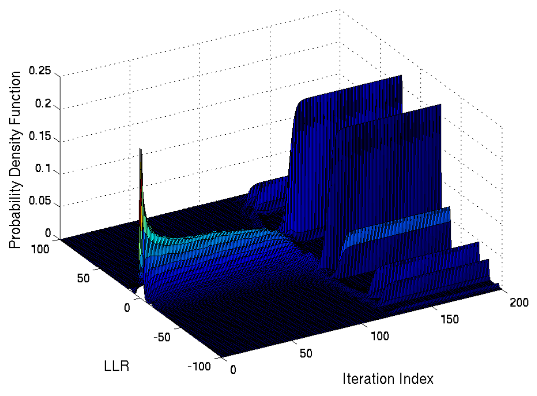
(a) EXIT chart, noiseless channel.



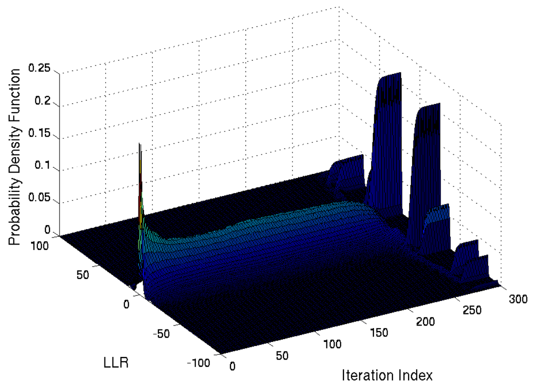
(b) BER vs. E_b/N_0.



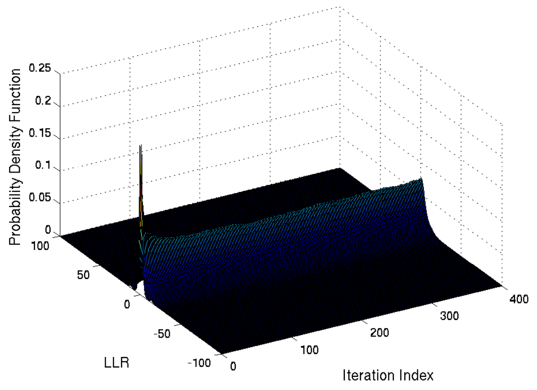
(c) LLR density evolution, PEG, block 1.



(d) LLR density evolution, PEG, block 2.



(e) LLR density evolution, PEG, block 3.

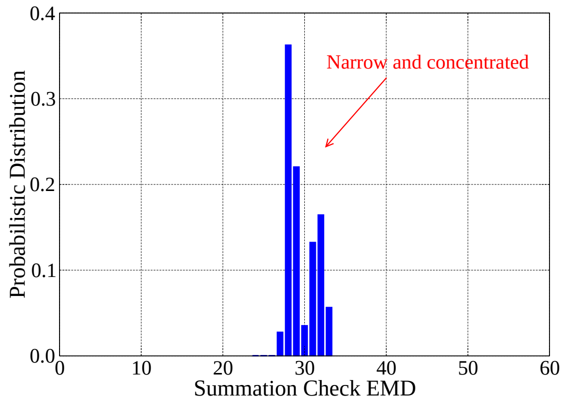


(f) LLR density evolution, PEG, block 4.

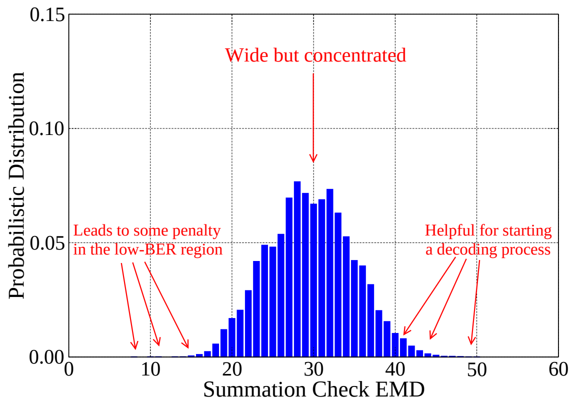
Figure 6.53: Irregular LDSC-EPA, $SF = 3.6$, $N = 8$, $K = 45000$, VBS, 400 iterations.

The respective BER curve in Fig. 6.53(b) verifies this assertion. In contrast, the RGB algorithm proposed in Section 6.4.2 constructs the graph in a randomized order, which achieves a balance between the size of the stopping sets involving only low-degree variable nodes and the size of the stopping sets involving high-degree variable nodes as well. In other words, the RGB algorithm achieves a trade-off between the probability of burst errors and that of residual errors. Shown by Fig. 6.53(b), using the RGB algorithm with a distinct tree growth (DTG) procedure enables a good decoding convergence, albeit leaving a non-trivial error floor. To reduce the error floor level, we apply a selective tree growth (STG) procedure for the RGB algorithm. Taking a cut-off threshold of $\Upsilon = 3$, no error floor is observed above 10^{-7}, cf. Fig. 6.53(b). In fact, for the high SNR region more than 200000000 info bits have been transmitted during the simulation and no single decision error has been detected. Nevertheless, one observes that there is a cross-over between the curve for DTG and the curve for STG, in the high-BER region. This is because an STG procedure improves the graph quality w.r.t. the low-degree variable nodes but degrades the graph quality w.r.t. the high-degree variable nodes. The performance difference between a PEG-designed interleaver and an RGB-designed interleaver can also be explained via the respective summation check EMD distributions. As shown in Fig. 6.54, the SCE distribution resulting from the PEG algorithm is very narrow, which is good for achieving a low error floor but challenging for eliminating burst errors. Eventually, if one increases the block length to an extremely large number, a PEG-designed interleaver should be able to offer a good performance as well. By using the RGB algorithm with DTG, the SCE distribution is concentrated and rich of diversity. This SCE diversity is very helpful in starting a successful decoding process but leads to some penalty for the performance in the low-BER region. In comparison, the SCE distribution resulting from RGB-STG is more diverse than that from PEG and narrower than that from RGB-DTG. Consequently, the resulting BER performance is better than that from PEG, in the high-BER region, and better than that from RGB-DTG, in the low-BER region. After all, a finite-length LDSC code has a strictly non-zero error probability, as long as the corresponding graph contains some stopping sets. Certainly, if the smallest stopping set has a large enough size, the error floor level will be negligible. The primary task of interleaver design for an LDSC code is not to pursue a zero error probability but to achieve a good balance between the burst error probability and the residual error probability.

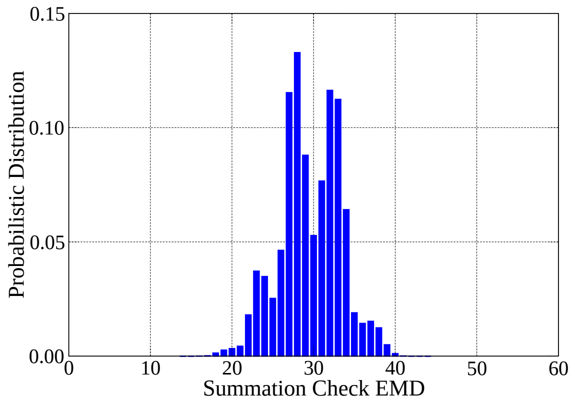
For the simulations in Fig. 6.53(b), the tree searching depth regulations applied for the RGB-DTG curve and the RGB-STG curve are identical. The searching depth for degree-2 variable nodes is limited to 40, and the searching depth for degree-3 variable nodes is limited to 20. For all the other variable nodes, the searching depth is limited to $20-(D-3)$, where D is the degree of the corresponding variable node. This searching depth regulation proves to be robust for LDSC matrix construction with various configurations. In the



(a) PEG.

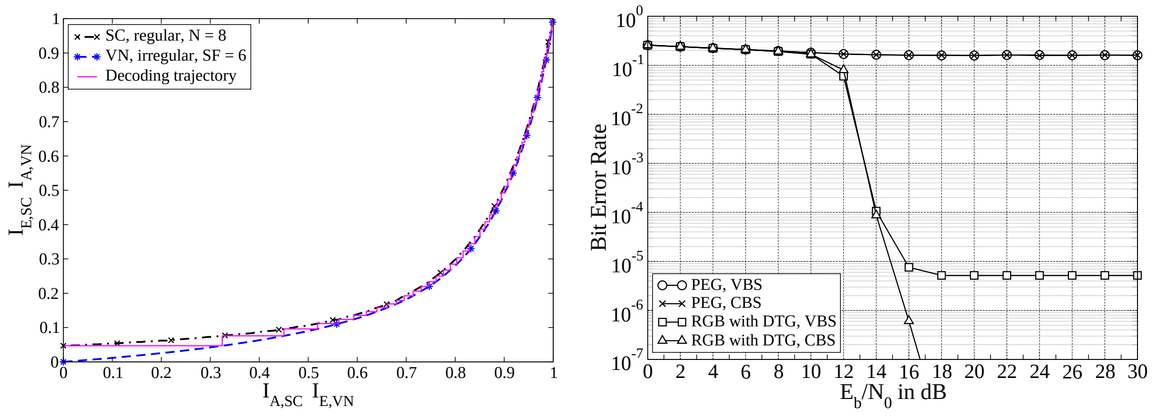


(b) RGB with DTG.



(c) RGB with STG, $\Upsilon = 3$.

Figure 6.54: SCE distributions for irregular LDSC-EPA, $SF = 3.6$, $N = 8$, $K = 45000$.



(a) EXIT chart, noiseless channel. (b) BER vs. E_b/N_0.

Figure 6.55: Irregular LDSC-EPA, $SF = 6$, $N = 16$, $K = 60000$, 400 iterations.

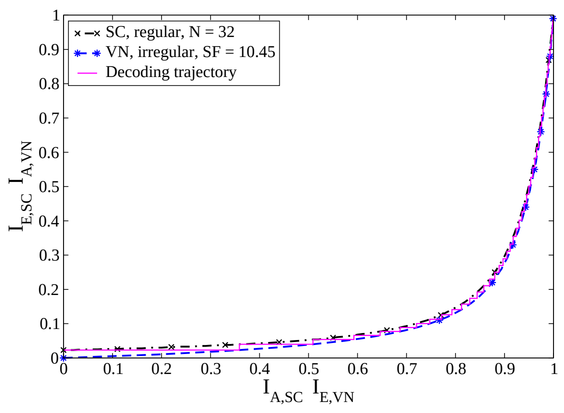
following discussion, we will apply this regulation to LDSC code design for $N = 16$ and $N = 32$. For a variable node with the degree $D > 21$, we have $20 - (D - 3) < 2$. In this case, we set the searching depth limit to 2, so that at least length 4 cycles can be avoided.

According to Tab. 3.3, the supportable bandwidth efficiency of SM-EPA with $N = 16$ is limited by 3.0465 bits/symbol. We apply the following VN degree distribution:

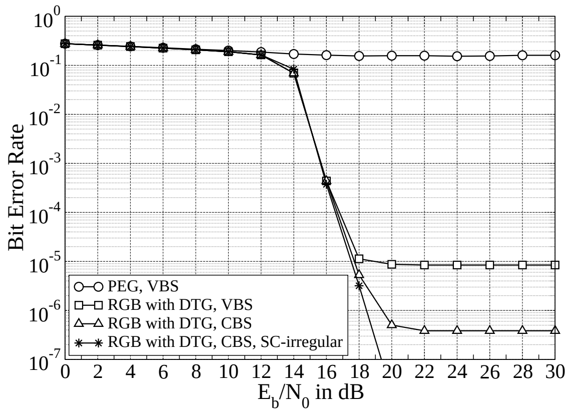
$$\lambda(D) = 0.29D^2 + 0.28D^3 + 0.09D^4 + 0.06D^7 + 0.08D^{10} + 0.09D^{12} + 0.07D^{16} + 0.04D^{20} \ ,$$

which leads to an average spreading factor of 6. The corresponding bandwidth efficiency is $16/6 \approx 2.606$ bits/symbol, which is close to the theoretical limit. Fig. 6.55 provides the EXIT chart as well as the resulting BER performances with different interleavers and different scramblers. Compared to the case of $N = 8$, a PEG-designed interleaver offers an even worse performance, in the sense that the probability of encountering burst errors is almost 1. As an interesting test, we apply cycle-based scrambling (CBS) to the PEG-designed LDSC matrix, and the resulting performance is nearly identical, cf. Fig. 6.55. Hence, the effect of scrambling in enabling a successful data separation is marginal, given a large N. The two stopping sets in Fig. 6.46 and Fig. 6.47 serve as a good explanation for this situation. Nevertheless, the effect of scrambling in eliminating residual errors is evident. With variable-node-based scrambling, an RGB-designed interleaver shows an error floor at about 5×10^{-5}. Applying CBS, no error floor has been observed. Comparing Fig. 6.55 with Fig. 6.53, one finds that selective tree growth and cycle-based scrambling are both effective in reducing the error floor level from an LDSC code.

Following the above treatments, now we check the supportable rate of irregular LDSC-EPA with $N = 32$. For the simulations in Fig. 6.53, we have used a block length of 45000. For the simulations in Fig. 6.55, we have used a block length of 60000. Now for $N = 32$, we will use a block length larger than 100000. The reason for doing so is that the density



(a) EXIT chart, noiseless channel.



(b) BER vs. E_b/N_0.

Figure 6.56: Irregular LDSC-EPA, $SF = 10.45$, $N = 32$, $K = 104500$, 400 iterations.

of an LDSC matrix is given by SF/K. When the spreading factor increases, a larger block length is necessary in order to keep the matrix density in a reasonable level. After some fine tuning, we obtain for SM-EPA with $N = 32$ the following VN degree distribution:

$$\begin{aligned}\lambda(D) &= 0.29D^2 + 0.08D^3 + 0.20D^4 + 0.09D^7 + 0.06D^{10} \\ &\quad +0.08D^{16} + 0.09D^{24} + 0.07D^{32} + 0.04D^{48}\ , \end{aligned} \tag{6.12}$$

which leads to an average spreading factor of 10.45. The corresponding EXIT chart is given in Fig. 6.56(a). One observes that the EXIT curve of the SC ensemble and that of the VN ensemble well match with each other. In case that a decoding convergence is attainable given a practical block length, we achieve a rate of $32/10.45 \approx 3.0622$ bits/symbol, which is far beyond the previously known limit of 2 bits/symbol. Typically, the larger the bit load N is, the more sensitive the decoder is to the graph structure, since a high-degree summation check considerably increases the chance of forming stopping sets. As expected, a PEG-designed interleaver does not provide a desirable performance, and using the RGB algorithm with a distinct tree growth (DTG) procedure eliminates the probability of burst errors but results in a non-trivial error floor, cf. Fig. 6.56(b). Applying cycle-based scrambling drops the error floor from 8×10^{-5} to 3.5×10^{-7}, which is still non-negligible. Due to the existence of very-high-degree variable nodes, short cycles can easily be formed among low-degree variable nodes, given a randomized graph construction order. The main causes for the error floor after applying CBS are graph pieces similar to that in Fig. 6.46 and Fig. 6.47. As already mentioned in Section 6.4.3, there is an simple solution to avoid these types of graph pieces. Limiting the number of degree-2 variable nodes connected to every summation check below or equal to 2 effectively eliminates the choice to encounter such stopping sets. However, this simple solution is not practical, unless one relaxes the regulation on the summation check degree distribution. Due to randomized edge-growth operations, the RGB algorithm has a negligible chance to fulfill such a requirement, as

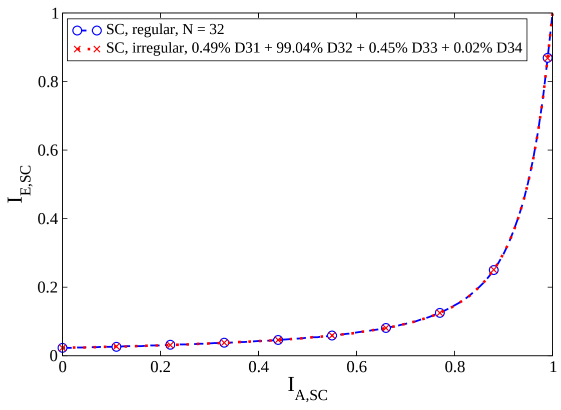


Figure 6.57: EXIT chart for two SC ensembles, EPA, $N = 32$, noiseless channel.

long as all summation checks are required to have a designated degree. Note that the VN degree distribution in (6.12) is a valid design not only for a regular SC degree distribution with $N = 32$ but also for those quasi-regular SC degree distributions with an average bit load of 32. Hence, it is in fact safe to relax the SC degree regulation during the graph construction. Given this motivation, we perform a new graph construction via RGB-DTG, taking into account the connection constraint on degree-2 variable nodes. The resulting LDSC matrix has the following SC degree distribution:

$$\eta(D) = 0.0049D^{31} + 0.9904D^{32} + 0.0045D^{33} + 0.0002D^{34} , \qquad (6.13)$$

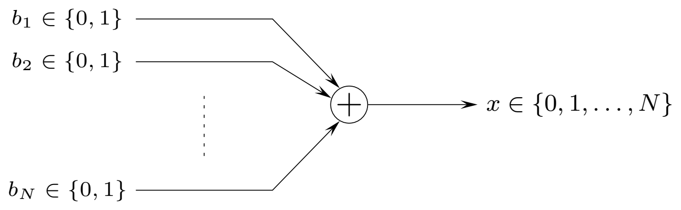
which leads to an average bit load of 32. Note that this is not from a code design but is a natural result from the graph construction procedure without applying an SC degree regulation. Nevertheless, one finds that this degree distribution is almost regular, in the sense that the fraction for $D \neq 32$ is marginal. This is achieved largely because of the MCCS treatment adopted by the RGB algorithm. Fig. 6.57 compares the EXIT function of a regular SC ensemble with $N = 32$ and an irregular SC ensemble obeying (6.13). One can hardly see any difference between the two EXIT curves. Hence, the degree distribution in (6.12) fits that in (6.13) as well. Given the new LDSC matrix and applying cycle-based scrambling, no error floor is observed any more, as shown in Fig. 6.56(b).

Up to this point, it is evident that the supportable rate of SM-EPA is virtually unlimited, given a well-designed irregular repetition code. Certainly, to achieve a very high data rate one needs to take an extremely large bit load, because the entropy of an SM-EPA symbol grows logarithmically w.r.t. the bit load. Equivalently, this is to say that SM-EPA is not really suitable for very-high-rate transmission, in the concern of computational complexity. The discussions within this chapter have not included the issue of power efficiency, in order to attain an easy elaboration. Nevertheless, the discussions in the next chapter treat code design in a more general framework, with the bandwidth efficiency and power efficiency both carefully considered.

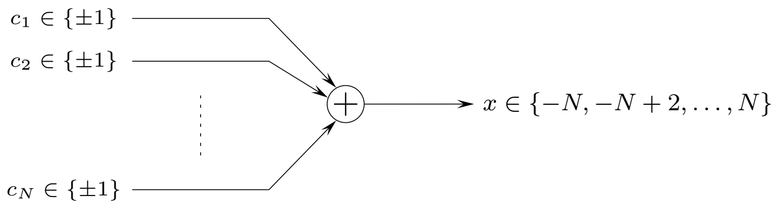
Chapter 7

Channel Coding for Superposition Mapping

Superposition mapping offers many advantages including Gaussian-like symbol distribution, low-complexity SISO demapping, and configuration flexibility. On the other hand, superposition mapping also necessitates a completely new way of thinking for code design. Shown in Chapter 5, using classical powerful parity-check codes leads to a very unsatisfying performance for superposition mapping. In contrast, the results from repetition-coded SM are more desirable, in the sense of supportable bandwidth efficiency. For this reason, Chapter 6 carries out a thorough study on the working mechanism of repetition-coded SM. This study yielded many useful hints for a more systematic code design. For coded SM transmission, the primary task of channel coding is to enable the separation of superimposed chips. If and only if this primary task can be successfully accomplished, will a channel code get the chance to combat the additive noise in an efficient way. For this primary task, repetition coding deserves to be a good choice because it enables an efficient information aggregation and distribution process. Besides, having zero coding gain is not a big issue in the stage of data separation. In fact, superposition mapping itself can offer a considerable power gain w.r.t. uniform bijective mapping. As long as the task of data separation can be successfully accomplished, the necessary coding gain for a superposition mapper is much less than that for a uniform bijective mapper. Hence, an optimal coding scheme for SM should first ensure the possibility of data separation and then provide a moderate but sufficient coding gain. Clearly, the question is how to design such a code. This is what the present chapter will be dedicated to. Various aspects, either theoretical or practical, will also be discussed in this chapter. As a topic of great practical importance, the proper way of combining repetition coding and parity-check coding will be investigated in detail. The discussion will mostly start with SM-EPA and then extend to the case of SM-UPA and SM-GPA when necessary.



(a) An N-user binary adder channel.



(b) SM-EPA with bit load N, $\alpha = 1$.

Figure 7.1: Duality between the binary adder channel and SM-EPA.

7.1 Some Theoretical Aspects

Before we proceed with code design for superposition mapping, there are several interesting issues necessary to be mentioned, including the duality between superposition mapping and the binary adder channel, rate limits for coded SM-EPA transmission, and the effect of finite block lengths. The discussion in this section gives a good starting point for the work in the remainder of the chapter.

7.1.1 Duality to Binary Adder Channel

Although one may apply arbitrary type of power allocation to superposition mapping, the equal power allocation (EPA) strategy should always be used as a basic building block, since only then a Gaussian-like symbol distribution will be attainable. As a matter of fact, SM-EPA itself is equivalent to the binary adder channel (BAC) well-known from multi-user information theory [34, 93–96]. The binary adder channel describes a scenario that multiple users communicate with a single receiver. Typically the inputs to the binary adder channel are defined over $GF(2)$, as depicted in Fig. 7.1(a). In comparison, superposition mapping with equal power allocation shares exactly the same structure with the BAC, except that its inputs are drawn from $\{\pm1\}$, illustrated in Fig. 7.1(b). From an information theory point of view, SM-EPA and BAC are completely equivalent. Consequently, all the channel codes developed for the BAC can easily be applied for SM-EPA, and more importantly, all coding techniques developed within this thesis are also applicable for the BAC. In the following, we will have a brief review on the available coding approaches for the BAC.

The capacity of a binary adder channel is given by the entropy of the channel output. Due to nonuniform distribution, we have

$$H(x) < N \ . \tag{7.1}$$

Therefore, channel coding is mandatory in order to achieve an error-free transmission, and there is generally a rate limit for the channel code according to

$$R \leqslant H(x)/N \ , \tag{7.2}$$

which is identical to the case of SM-EPA, as discussed in Section 3.2. Without loss of generality, we may categorize the available coding approaches for BAC into three classes. The first class of approaches are in fact well-known as code-division multiplexing (CDM) or code-division multiple access (CDMA) [97]. By means of orthogonal spreading, a perfect data reconstruction is possible via a bank of matched filters. This type of approaches are simple and stable. However, when applied to a binary adder channel, the maximum supportable data rate will be one bit per channel use regardless of the particular channel capacity, due to the request for a strict orthogonality. Nowadays, the non-optimality of orthogonal multiplexing has been commonly recognized [98, 99]. The second class of approaches have a history dating back to the late 1970's. In the concern of enhancing the supportable data rate for the BAC, many researchers have been resorting to the concept of uniquely decodable codes [35, 36, 94]. We will briefly elaborate the basic principle of uniquely decodable codes in the next step, so as to show their advantages as well as disadvantages for superposition mapping. The third class of approaches emerge after the invention of turbo codes and the reinvention of LDPC codes. In recent years, researchers have been particularly enthusiastic on applying iteratively decodable channel codes to various types of multiple access channels [100–102], mainly encouraged by the superior performance of sparse-graph codes on the binary-input AWGN channel. With the help from orthogonal spreading or spatial receiver diversity, iteratively decodable parity-check codes work very well for multiple access channels. Nevertheless, successful applications of parity-check codes for a pure BAC (equal power allocation) with $N > 3$ have rarely been reported. There are in fact many reasons behind such a situation. In this chapter, we will clarify most of the relevant issues and provide practical solutions correspondingly.

By uniquely decodable codes, each user is assigned a unique spreading sequence that serves as a user-specific identification code. The set of N spreading sequences are not mutually orthogonal but uniquely distinguishable. For example, the following matrix gives a uniquely decodable code for the BAC with $N = 3$ users:

$$\begin{bmatrix} 1 & 1 \\ 1 & -1 \\ 1 & 0 \end{bmatrix} , \tag{7.3}$$

$$\begin{tikzcd}
b_1, b_2, b_3 & x_1, x_2 \\
0,0,0 \arrow[r] & 0,1 \\
0,0,1 \arrow[r] & 1,1 \\
0,1,0 \arrow[r] & 1,0 \\
1,0,0 \arrow[r] & 1,2 \\
0,1,1 \arrow[r] & 2,0 \\
1,0,1 \arrow[r] & 2,2 \\
1,1,0 \arrow[r] & 2,1 \\
1,1,1 \arrow[r] & 3,1
\end{tikzcd}$$

Figure 7.2: The mapping rule of a 3-user uniquely decodable code.

where -1 denotes a bit flipping operation and 0 denotes a constant zero. Given this code matrix, the spreading operation for each user will be defined as

$$\begin{aligned}
b_1 = 0 \quad &\mapsto \quad [0,0]\ , & b_1 = 1 \quad &\mapsto \quad [1,1] \\
b_2 = 0 \quad &\mapsto \quad [0,1]\ , & b_2 = 1 \quad &\mapsto \quad [1,0] \\
b_3 = 0 \quad &\mapsto \quad [0,0]\ , & b_3 = 1 \quad &\mapsto \quad [1,0]\ .
\end{aligned}$$

After linear superposition, one gets a mapping rule as depicted in Fig. 7.2. Though the spreading sequences are non-orthogonal, they do guarantee a bijective mapping rule between the input binary triple and the channel output couple. The achieved throughput is $3/2 = 1.5$ bits/symbol. As proposed by Chang in [34], one may recursively construct a larger code matrix so as to support more users. Let $\mathbf{D}_0 = 1$ denote the 0th order code matrix, the construction procedure is

$$\mathbf{D}_{i+1} = \begin{bmatrix} \mathbf{D}_i & \mathbf{D}_i \\ \mathbf{D}_i & -\mathbf{D}_i \\ \mathbf{I}_i & \mathbf{0}_i \end{bmatrix}\ , \tag{7.4}$$

with $\mathbf{I}_i$ and $\mathbf{0}_i$ denoting an identity matrix and an all-zeros matrix of order 2^i, respectively. An important feature of this recursive construction method is that the bijectivity of the corresponding mapping rule is always ensured. With a little bit of mathematical maneuver, one finds that by the ith recursive construction the dimension of the code matrix is given by $(2^{i-1}(i+2)) \times 2^i$ and the corresponding throughput is

$$\frac{2^{i-1}(i+2)}{2^i} = \frac{i}{2} + 1 \quad \text{bits/symbol}\ . \tag{7.5}$$

For example, to achieve a data rate of 3 bits/symbol, one needs a code matrix of dimension 48×16. By abandoning strict orthogonality, uniquely decodable codes can achieve significantly higher throughputs than orthogonal spreading. However, these codes come with some non-trivial drawbacks, and are generally non-optimal.

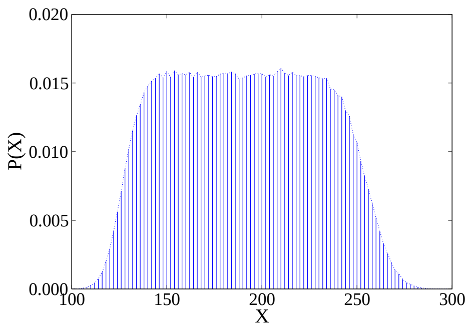


Figure 7.3: Symbol distribution resulting from the 8th recursive code matrix construction.

According to (3.24), the capacity of the BAC with $N = 48$ is about 3.8396 bits/symbol. Hence, a 48×16 uniquely decodable code is far from being optimal. Chang showed in [34] that this code construction will be close to (but with a strictly nonzero gap) optimal only when i reaches the infinity, which is certainly undesirable. Second, the available designs of uniquely decodable codes [35, 36, 103] typically overlook one issue, that is by using a code matrix that ensures bijective mapping the distribution of code symbols will no longer be Gaussian, and consequently incurs a loss of optimality for BACs with an additive Gaussian noise. For example, Fig. 7.3 shows the measured symbol distribution for the code matrix obtained from the 8th recursive construction. One observes that the symbol distribution is actually quasi-uniform within a wide range and is also slightly asymmetric.

Revisiting (7.3), one finds that a '0' in the code matrix actually leads to an absence of a certain bit in a certain symbol. Therefore, a uniquely decodable code is in fact a special irregular repetition code with a deterministic scrambler and a deterministic interleaver. The code matrix construction procedure is not as controlled as that of orthogonal spreading, but it still does not include any randomness. The guiding idea of uniquely decodable codes is clearly to ensure a zero error probability for the noiseless BAC. However, this comes with a high price both in the symbol distribution and the achievable rate. Well-known in the community, the success of Shannon in deriving channel capacity is largely because of his way in treating the error probability. Instead of pursuing a zero error probability, Shannon tries to push the error probability arbitrarily small, but not necessarily zero. An LDSC code can achieve an arbitrarily small error probability, but not necessarily zero as long as a stopping set of certain size exists. From a practical point view, this is already sufficient for most applications. Besides, in the sense of keeping a Gaussian-like symbol distribution, LDSC codes are clearly more elegant than uniquely decodable codes. Moreover, by means of controlled-random interleaving, LDSC codes can achieve higher rates than uniquely decodable codes. For example, a uniquely decodable code with $N = 8$ has a code matrix of dimension 8×4, which leads to a rate of 2 bits/symbol. Given $N = 8$, the supportable rate of LDSC codes is more than 2.2 bits/symbol, cf. Section 6.4.5. Last but not least, code designs to be given in this chapter perform even closer to the capacity.

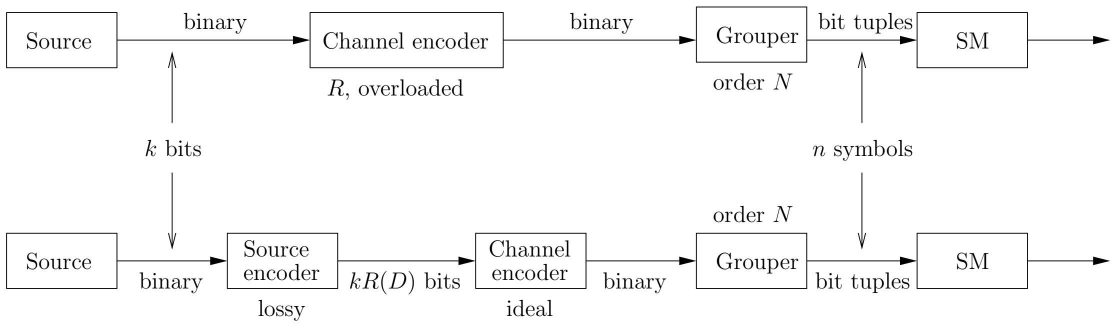


Figure 7.4: Coded SM transmission over a noiseless channel.

7.1.2 Finite-Error Capacity for Coded SM Transmission

For superposition mapping with equal power allocation or grouped power allocation, the symbol distribution is Gaussian-like. Meanwhile, the mapping rule is non-bijective. Hence, there will be a theoretical bound even for noiseless transmission.

Following the same procedure as in Section 2.4, we may draw an equivalent transmission system for coded SM transmission over a noiseless channel, depicted in Fig. 7.4. To enable error-free channel decoding at the receiver side, the following inequality must be fulfilled:

$$kR(D)/n \leqslant H(x) \ , \tag{7.6}$$

where $H(x)$ is the symbol entropy, which can be approximated as

$$H(x) \approx \frac{1}{2}\log_2(2\pi eN/4) \quad \text{bits} \tag{7.7}$$

for SM-EPA and

$$H(x) \approx \frac{1}{2}\log_2(\frac{\pi}{6}eG) + L \quad \text{bits} \tag{7.8}$$

for SM-GPA, cf. Section 3.3.3 and Section 3.5.3. Since

$$R(D) = 1 - h(P_e) \tag{7.9}$$

for a Bernoulli($\frac{1}{2}$) source, we get

$$k(1 - h(P_e))/n \leqslant H(x) \ . \tag{7.10}$$

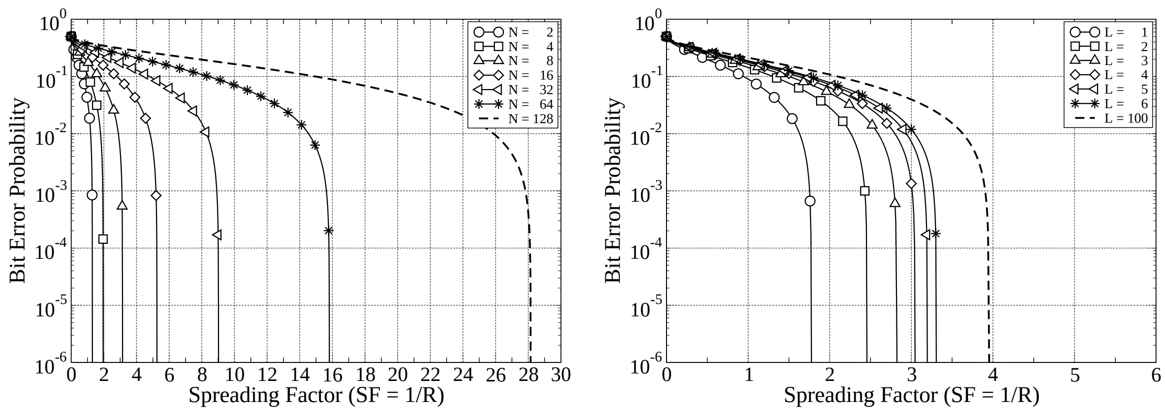
Substituting $k/n = R \cdot N$ into (7.10), we obtain

$$R \cdot N(1 - h(P_e)) \leqslant H(x) \ , \tag{7.11}$$

and subsequently

$$P_e \geqslant h^{-1}\left(1 - \frac{H(x)}{R \cdot N}\right) \tag{7.12}$$

for $P_e \in [0, 0.5]$.



(a) SM-EPA. (b) SM-GPA, $G = 4$.

Figure 7.5: Finite-error capacity for coded SM transmission over a noiseless channel.

Fig. 7.5(a) illustrates the capacity curves for SM-EPA. With an increasing bit load, the capacity curve shifts rapidly towards the right. This means that a larger spreading factor is strictly necessary to enable a receiver convergence when N becomes larger. Moreover, due to the logarithmic relationship between $H(x)$ and N in (7.7), the increase of the minimum required spreading factor is about proportional to the bit load. This causes a serious problem for practical systems. Given a very large spreading factor, a very large block length will be necessary in order to make the incidence matrix being low-density, while a good performance is achievable only if the incidence matrix is low-density. Roughly speaking, SM-EPA with $N > 32$ is not suitable for being used as a mapping scheme.

In contrast, the situation for SM-GPA is different, as shown in Fig. 7.5(b). The distance between the capacity curves becomes smaller and smaller when one increases the number of power levels, given a fixed group size. Furthermore, there is an ultimate limit for the required spreading factor. According to (7.8), the minimum required spreading factor for error-free SM-GPA transmission is given by

$$SF \geqslant \frac{N}{H(x)} \;=\; \frac{G \cdot L}{\frac{1}{2}\log_2(\frac{\pi}{6}eG) + L} \;. \tag{7.13}$$

In general, we have

$$\frac{G \cdot L}{\frac{1}{2}\log_2(\frac{\pi}{6}eG) + L} \;<\; G \;, \tag{7.14}$$

while for very large L, we have

$$\frac{G \cdot L}{\frac{1}{2}\log_2(\frac{\pi}{6}eG) + L} \;\approx\; G \;. \tag{7.15}$$

Hence, given ideal channel codes, taking $SF = G$ should already ensure a perfect data separation for all L. This result is indeed important for practice. That is, by using SM-GPA and choosing $R = 1/G$ one will not encounter any limit on the supportable rate, which is a bottleneck for coded SM-EPA transmission.

7.1.3 Typicality of Finite-Length Symbol Sequences

Employing a nonuniform mapping scheme not only brings a new challenge for the code design but also brings a new issue that is never concerned in conventional communication systems employing uniform mapping schemes. In the following, we investigate the effect of block length to the information-carrying capability of SM-EPA symbol sequences.

As a starting example, let us consider the case of SM-EPA with $N = 2$ and $\alpha = 1$. Due to the nonuniform symbol distribution

$$P(x = -2) = \frac{1}{4}, \qquad P(x = 0) = \frac{1}{2}, \qquad P(x = +2) = \frac{1}{4} \; ,$$

the event $\{x = 0\}$ carries one bit of information, and the event $\{x = \pm 2\}$ carries two bits of information. From a transmitter standpoint, it is nice to send more $\{x = 0\}$ than $\{x = \pm 2\}$, because this saves transmission power. However, from a receiver standpoint, it is desirable to receive more $\{x = \pm 2\}$ than $\{x = 0\}$, because this makes the detection easier. Referring to Fig. 5.2 and the relevant mathematical derivations therein, we see that by observing $\{x = \pm 2\}$ an APP demapper can already deliver reliable LLR messages in the initial iteration, while upon observing $\{x = 0\}$ the produced LLR messages will be all zero in the initial iteration. In an extreme case, if the channel is noiseless and the received symbols are all zero, then an iterative receiver will completely fail regardless of the strength of the channel code, since all messages from the demapper will be zero at the initial iteration and naturally nothing will happen by doing further iterations. Regardless of the particular block length, the occurrence of symbol values will be asymptotically typical, by transmitting an infinite amount of blocks. Therefore, if a certain block contains too many events $\{x = \pm 2\}$, there must be another block that will contain an insufficient amount of events $\{x = \pm 2\}$. Given this concern, the best situation is that each symbol block is typical, which in turn requires the block length to be infinite. Equivalently, this is to say that the practically supportable rate given a finite block length will be smaller than what promised by the symbol entropy.

In a more formal way, the above consideration is to ensure

$$-\log P(x_1, x_2, \ldots, x_K) \geqslant -\log P(v_1, v_2, \ldots, v_Q) \; , \tag{7.16}$$

where K denotes the number of symbols per transmission block and Q denotes the number of info bits per transmission block. The left term of (7.16) gives the amount of information that a particular symbol block actually carries, and the right term of (7.16) gives the amount of information that one tries to load onto this symbol block. As long as a certain transmission block violates (7.16), error-free detection will be strictly prohibitive, even over a noiseless channel. Assume that K is moderate and the applied interleaver pattern is good. Then, the symbols within a transmission block will be approximately independent, which leads to

$$-\sum_{i=1}^{K} \log P(x_i) \gtrapprox -\log P(x_1, x_2, \ldots, x_K) \; . \tag{7.17}$$

Note that $-\log_2 P(v_1, v_2, \ldots, v_Q) = Q$ for i.u.d. info bits. Hence, (7.16) can be rewritten as

$$-\sum_{i=1}^{K} \log_2 P(x_i)/K \geqslant Q/K = R \cdot N \; . \tag{7.18}$$

Without loss of generality, we may call $-\sum_{i=1}^{K} \log P(x_i)/K$ the block-wise information rate. Certainly, $R \cdot N$ is the effective transmission rate of a particular system. In fact, the above requirement is never an issue for systems employing bijective uniform mapping. Given a bijective mapping scheme with bit load N, we have $-\log_2 P(x_i) \equiv \log_2 |\mathcal{X}| \equiv N$. Consequently, (7.18) always holds, regardless of the block length K. Now, for the case of nonuniform superposition mapping, the situation is different. The condition in (7.18) can constantly be fulfilled only if $R \cdot N$ is small enough and K is large enough. According to the asymptotic equipartition property (AEP) theorem [2], the small set of typical sequences with probability

$$P(x_1, x_2, \ldots, x_K) = 2^{-K \cdot H(x)} \tag{7.19}$$

will occupy almost all the probability for $K \to \infty$. Let $\mathcal{A}^K$ represent the set of sequences fulfilling (7.19), we have

$$\lim_{K \to \infty} \Pr\left\{\mathcal{A}^K\right\} = 1 \; , \tag{7.20}$$

and

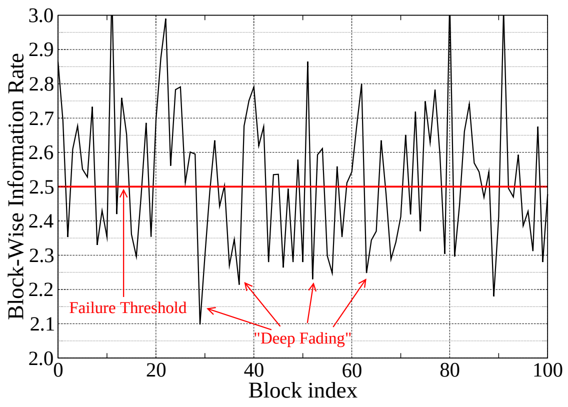
$$\lim_{K \to \infty} |\mathcal{A}^K| = 2^{K \cdot H(x)} \; . \tag{7.21}$$

As a result, we have

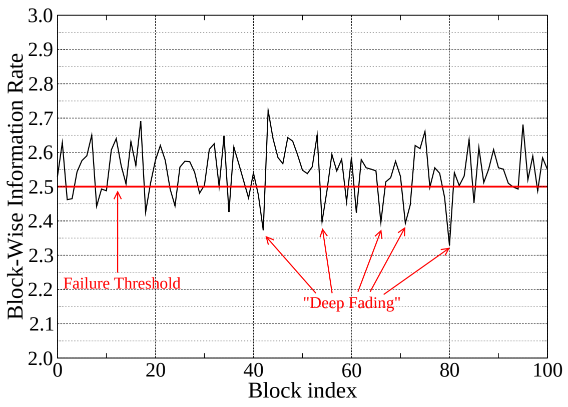
$$\lim_{K \to \infty} -\log_2 P(x_1, x_2, \ldots, x_K) \; = \; -\log_2 2^{-K \cdot H(x)} \; = \; K \cdot H(x) \; . \tag{7.22}$$

Therefore, by taking $R \cdot N = H(x)$, the condition in (7.18) can constantly be fulfilled if and only if the block length K is infinite, in case of coded SM-EPA transmission. Consequently, if the block length K is finite, one has to take a small enough $R \cdot N$ such that the probability that (7.18) gets violated will be sufficiently close to zero. In the following, we illustrate this principle via some numerical tests.

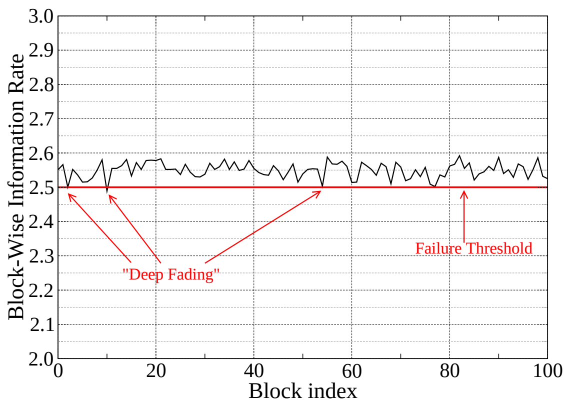
Suppose that one wants to attain a rate of 2.5 bits/symbol via a coded SM-EPA system with $N = 8$. According to Tab. 3.3, this is theoretically possible. In practice, however, this demands a huge block length. Fig. 7.6 shows the measured block-wise information rates given different block lengths. For the test, we assume that all code bits are mutually independent, which is in fact the best case concerning the block-wise information rate.



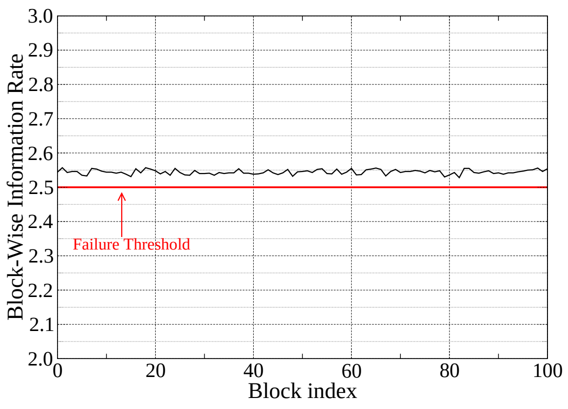
(a) $N = 8$, $K = 20$.



(b) $N = 8$, $K = 200$.



(c) $N = 8$, $K = 2000$.



(d) $N = 8$, $K = 20000$.

Figure 7.6: Block-wise information rate for randomly generated SM-EPA symbol blocks.

Since for the region below 2.5 bits/symbol a decoding failure will definitely occur, we call the corresponding horizontal line as the failure threshold. Any block-wise information rate below this threshold corresponds to a violation of (7.18). As a matter of fact, given a small block length, the situation looks very similar to that of time-varying radio channels. The block-wise information rate varies dramatically from block to block, and there is a big probability for it to drop below the failure threshold and sequentially causes a "deep fading" effect for the decoder. By increasing the block length, the dynamic range of the block-wise information rate reduces steadily. For $K = 20000$, no failure is observed within the 100 tested blocks. Nevertheless, this does not mean that in practice a rate of 2.5 bits/symbol is achievable with $K = 20000$. Given a non-cycle-free graph, the actually required block length is considerably larger than this value. One has to ensure that there exits no quasi-isolated subgraph which contains a less amount of channel observations. On the other hand, a good graph structure is only attainable given a sufficiently large block length. Hence, to achieve a near-capacity rate for SM-EPA, a large block length is mandatory, theoretically and practically. Fig. 7.6(a) shows that the block-wise information rate frequently drops below 2.2 bits/symbol, for $K = 20$. This well explains why an LDSC decoder becomes so sensitive to the interleaver pattern at this rate, cf. Fig. 6.53.

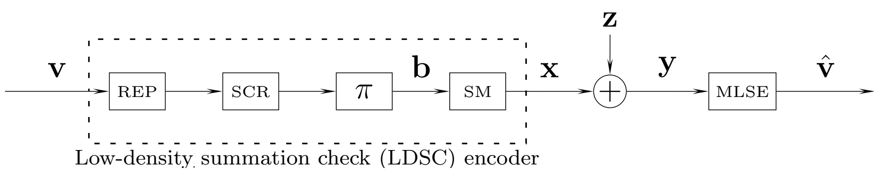


Figure 7.7: Repetition-coded SM with maximum-likelihood sequence estimator.

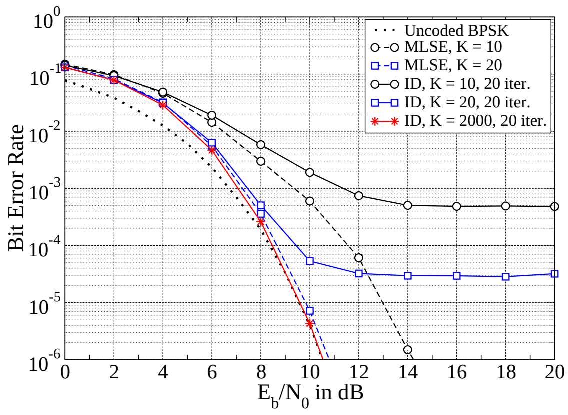


Figure 7.8: Regular repetition-coded SM-EPA, $SF = 3$, $N = 3$, PEG, VBS.

7.1.4 Maximum-Likelihood Decoding vs. Iterative Decoding

For repetition-coded SM transmission, the optimal receiver in the sense of minimizing the word error rate is the maximum-likelihood sequence estimator (MLSE), which computes

$$\hat{\mathbf{v}} = \arg\max_{\tilde{\mathbf{v}}} \{\mathbf{y}|\tilde{\mathbf{v}}\} \ , \tag{7.23}$$

cf. Fig. 7.7. This is an all-in-once brute-force approach. In former discussions on LDSC coding, we always assume iterative decoding (ID), since for a typical block length the complexity of the MLSE is prohibitive. Nevertheless, it is interesting to check the performance difference of these two types of receivers, given short block lengths. From Fig. 7.8 one observes that there is a non-trivial performance difference for a block length of 10 and 20. Given the MLSE, no error floors are observed, which means that the mapping from $\mathbf{v}$ to $\mathbf{x}$ is bijective. Therefore, the error floors resulting from iterative decoding are caused by the inaccuracy of message passing given a small graph full of short cycles. Note that the asymptotic performance of the MLSE is uniquely determined by the minimum code word distance. By comparing the case of 10 and 20 in Fig. 7.8, it is evident that the minimum code word distance of an LDSC code improves with the block length. For $K = 20$, there is a distance to the BPSK bound, even with an MLSE receiver. However, if we increase the block length to 2000, the asymptotic performance of LDSC approaches the BPSK bound, even with iterative decoding, cf. Fig. 7.8. Certainly, in this case the performance difference between the MLSE and iterative decoding becomes negligible. Hence, a large block length not only improves the minimum code word distance but also reduces the performance difference between the MLSE and iterative decoding.

7.2 Some Practical Aspects

In this section, we investigate some practical aspects of coded SM transmission. The goal is to provide some valuable hints for a realistic system design, particularly in the sense of choosing appropriate parameters when the performance requirement is certain.

7.2.1 Information-to-Complexity Ratio

For a practical system, one of the most critical concern is the computational complexity of the transceiver, and usually the major computational load is at the receiver side. Hence, given that the performance requirement is fulfilled, selecting a mapping format that leads to the lowest demapping complexity is of significant importance. It has been shown in Chapter 3 that SM-EPA and SM-GPA are both able to produce symbols with a Gaussian-like probabilistic distribution. From an information theoretical point of view, SM-EPA and SM-GPA are both optimal for data transmission over a Gaussian channel. However, in the concern of the demapping complexity, SM-EPA and SM-GPA have substantially different characteristics. Therefore, it is meaningful to check which mapping strategy is the best for certain data rates. In the following, we compare the information-to-complexity ratio, which describes the figure-of-merit of a mapping scheme, for SM-EPA and SM-GPA. To provide a systematic study, SM-UPA is also considered.

Given a targeted bandwidth efficiency, the first step for designing a coded SM transmission system is to choose an appropriate bit load N and a suitable coding rate R. For the bit load N, one can resort to a mutual information analysis and select a moderate N that ensures a capacity-achieving symbol distribution for the targeted bandwidth efficiency. For example, to achieve a bandwidth efficiency of 2 bits/symbol, SM-EPA with $N = 8$ is a reasonable choice, according to Fig. 3.7(b). Naturally, the corresponding coding rate should be $R = 1/4$. Note that choosing SM-EPA with $N = 16$ instead of $N = 8$ will not bring any benefit but leads to an unnecessary complexity increase. Alternatively, one may also choose SM-GPA with $G \geqslant 2$ and $L = 2$, according to Fig. 3.11 and Fig. 3.12. The corresponding coding rate should be $R = 1/G$, which complies with the observation in Section 7.1.2. More generally, we may derive a rule of thumb from the mutual information analyses provided in Fig. 3.7(b), 3.11, and 3.12. Given SM-EPA or SM-GPA with $G \geqslant 2$, the symbol distribution will be about capacity-achieving for rates fulfilling

$$R \cdot N \leqslant H(x) - 0.5 \quad \text{bits/symbol}\,, \tag{7.24}$$

where $H(x)$ denotes the corresponding symbol entropy. Without loss of generality, we may call $H(x) - 0.5$ the cut-off rate for superposition mapping. Coded SM transmission operating below this rate has a good potential to approach the Gaussian channel capacity.

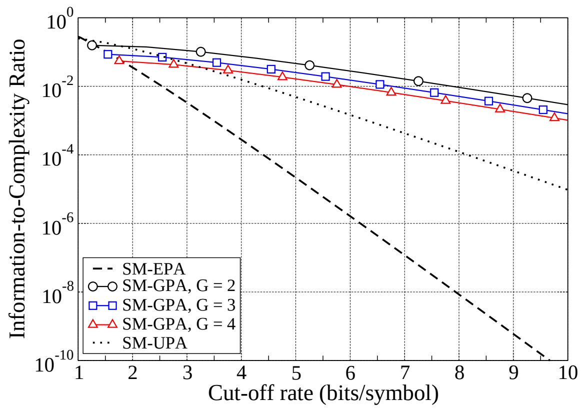


Figure 7.9: Information-to-complexity ratio for superposition mapping.

We define the information-to-complexity ratio (ICR) of SM as the cut-off rate divided by the demapping complexity. According to (3.24) and (5.21), we have

$$\mathrm{ICR}_{\text{SM-EPA}} \doteq \frac{\frac{1}{2}\log_2(\pi e N/2) - 0.5}{N^2} \tag{7.25}$$

for SM with equal power allocation. According to (3.33) and (5.31), we have

$$\mathrm{ICR}_{\text{SM-GPA}} \doteq \frac{\frac{1}{2}\log_2(\pi e G/6) + L - 0.5}{G^2 2^L} \tag{7.26}$$

for SM with grouped power allocation. Similarly, we may define

$$\mathrm{ICR}_{\text{SM-UPA}} \doteq \frac{1}{2}N\Big/2^N \tag{7.27}$$

for SM with unequal power allocation, by observing that for bijective uniform mapping the most successful channel codes are of rate 1/2. Fig. 7.9 compares the ICR for SM with three types of power allocation schemes. The first result shown by the figure is that SM-EPA is not practical for high-rate transmission. Clearly, the reason is the logarithmic growth of the symbol entropy w.r.t. the bit load N. The second result shown by Fig. 7.9 is that SM-GPA is generally better than SM-EPA, which is due to the linear growth of the symbol entropy w.r.t. the number of power levels. Hence, using SM-GPA instead of SM-EPA can effectively reduce the computational complexity per info bit. Besides, the ICR of SM-GPA degrades when the group size becomes larger, which is due to the logarithmic growth of the symbol entropy w.r.t. the group size. Concerning the demapping complexity, SM-GPA with $G = 2$ is the best choice. On the other hand, the symbol distribution for $G = 2$ has a triangular envelope, i.e., not really Gaussian-like, cf. Fig. 3.11. Nevertheless, the theoretically achievable performance gain by using a larger group size is in fact marginal, as shown by Fig. 3.11. The third result shown by Fig. 7.9 is that SM-GPA is more complexity-efficient than SM-UPA for high-rate transmission. This is due to the fact that the symbol entropy of SM-GPA is comparable to SM-UPA while its demapping complexity is exponential in the number of power levels L but only quadratic in the group size G.

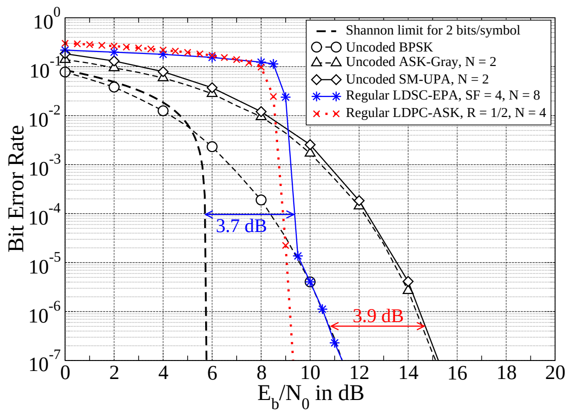


Figure 7.10: Regular LDSC-EPA vs. uncoded SM-UPA for 2 bits/symbol.

7.2.2 Compression Gain and Irregularity Loss

Well-known in the coding community, the asymptotic coding gain of a channel code is determined by the minimum distance between two code words. Assuming a binary-input AWGN channel, the asymptotic coding gain is given by

$$\Delta = 10\log_{10}(R \cdot d_{\min}) \quad \text{dB} \,, \tag{7.28}$$

where R denotes the coding rate and $d_{\min}$ denotes the minimum code word distance. For a linear binary block code, the minimum code word distance is identical to the minimum Hamming weight of all valid code words excluding the all-zero one. In case of convolutional coding, this minimum code word distance is often referred to as the free distance, which corresponds to the minimum Hamming weight of error paths on a trellis diagram [104]. In case of repetition coding, we have $d_{\min} = 1/R$ and subsequently $\Delta = 0$ dB for binary-input AWGN channels. This is exactly the reason why researchers are often discouraged to use repetition codes. Nevertheless, for non-binary-input AWGN channels, this understanding is no longer appropriate. LDSC coding, i.e., repetition-coded superposition mapping, can in fact provide a non-zero power gain w.r.t. uncoded uniform bijective mapping. As a good example, Fig. 7.10 compares the performance of regular LDSC-EPA with that of uncoded SM-UPA. Given $SF = 4$ and $N = 8$, the high-SNR performance of regular LDSC-EPA is identical to that of uncoded BPSK, which is about 3.9 dB better than uncoded SM-UPA with $N = 2$. Compared to uncoded ASK with Gray labeling, this power gain is slightly less. Moreover, Fig. 7.10 shows that the performance of regular LDSC-EPA is comparable to that of regular LDPC-coded ASK (Gray) till a BER of 10^{-5}. Therefore, the conventional understanding of coding gain is not really suitable for coded transmission systems employing a non-bijective mapping scheme. The power gain offered by LDSC-EPA comes from the "compression" procedure of superposition mapping. For this reason, we dub such a power gain a "compression gain".

Given an infinite block length and a cycle-free graph, i.e., assuming that no variable nodes form a stopping set, the minimum distance between the code words of an LDSC-EPA code is determined by the chip magnitude and the minimum VN degree. Mathematically, this can be written as

$$d_{\min,\text{LDSC-EPA}} = (2\alpha)^2 D_{\min} = 4\alpha^2 D_{\min} \ . \tag{7.29}$$

For LDSC-EPA, we have $E_b = SF \cdot \alpha^2$. Hence, the above formula can be rewritten as

$$d_{\min,\text{LDSC-EPA}} = 4 \, \frac{D_{\min}}{SF} \, E_b \ . \tag{7.30}$$

Note that (7.30) is independent of the bit load N. In other words, the minimum code word distance of LDSC-EPA does not change with N, assuming a fixed VN degree distribution. Now, let us check the situation of uncoded SM-UPA transmission. For uncoded SM-UPA, we have $E_s = N \cdot E_b$. According to Section 3.4.2, the average symbol energy for SM-UPA can be written as

$$E_{s,\text{SM-UPA}} = \sum_{n=1}^{N} \left(a 2^{-(n-1)}\right)^2 = \frac{4}{3} a^2 (1 - 2^{-2N}) \ . \tag{7.31}$$

Enforcing $E_{s,\text{SM-UPA}} = N \cdot E_b$, we obtain

$$a^2 = \frac{3N}{4(1 - 2^{-2N})} \, E_b \ . \tag{7.32}$$

Since the minimum symbol distance for SM-UPA is determined by the smallest magnitude of chips, we have

$$d_{\min,\text{SM-UPA}} = \left(2a 2^{-(N-1)}\right)^2 = 16 a^2 2^{-2N} = 12 \, \frac{N}{2^{2N} - 1} \, E_b \ , \tag{7.33}$$

which decreases rapidly with the bit load N. Given the above derivations, the asymptotic coding gain of LDSC-EPA w.r.t. uncoded SM-UPA with bit load N can be written as

$$\Delta = 10 \log_{10} \left(\frac{d_{\min,\text{LDSC-EPA}}}{d_{\min,\text{SM-UPA}}} \right) = 10 \log_{10} \left(\frac{(2^{2N} - 1) D_{\min}}{3N \cdot SF} \right) \quad \text{dB} \ . \tag{7.34}$$

For the parameter set used in Fig. 7.10, i.e., $D_{\min} = SF = 4$ for LDSC-EPA and $N = 2$ for uncoded SM-UPA, the asymptotic coding gain is about 3.98 dB, which agrees with the numerical results in Fig. 7.10. After all, the ultimate reason for this coding gain is that the minimum symbol distance of SM-EPA is constantly given by $4\alpha^2$, independent of the bit load N. This is achieved by reducing the symbol cardinality from 2^N to $N + 1$ via non-bijective mapping. Therefore, a non-bijective nonuniform SM-EPA mapping scheme leads to a smaller symbol entropy but a larger minimum symbol distance, compared to a bijective uniform mapping scheme. Given an appropriate channel code, the "compression" procedure of SM-EPA does not cause an information loss but provides a considerable power gain which reduces the necessary coding gain for approaching the capacity.

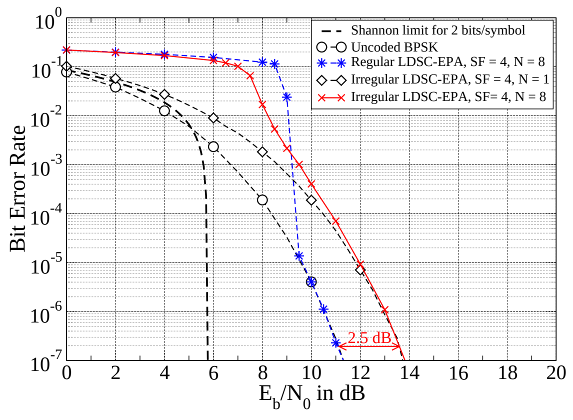


Figure 7.11: Regular LDSC-EPA vs. irregular LDSC-EPA for 2 bits/symbol.

In general, an irregular variable node degree distribution leads to a lower decoding threshold but meanwhile degrades the performance in the high-SNR region. This phenomenon can clearly be observed from Fig. 7.11, where the irregular LDSC-EPA code adopts the following VN degree distribution:

$$\lambda(D) = 0.20D^2 + 0.60D^3 + 0.20D^9 \ . \tag{7.35}$$

The high-SNR performance of an irregular LDSC-EPA code is tightly bounded by that of irregular repetition-coded BPSK with the same degree distribution. According to (7.30), the asymptotic power loss of an irregular LDSC-EPA code w.r.t. a regular one is given by

$$\tilde{\Delta} = 10 \log_{10} \left(\frac{SF}{D_{\min}} \right) \quad \text{dB} \ . \tag{7.36}$$

For the degree distribution in (7.35), we have $D_{\min} = 2$ and $SF = 4$. Correspondingly, the asymptotic power loss will be $\tilde{\Delta} \approx 3$ dB. In Fig. 7.11, the measured power loss is about 2.5 dB at a BER of 10^{-7}. If one measures at even lower BERs, this loss will finally reach 3 dB. Without loss of generality, we may call such a power loss an “irregularity loss”. An important message delivered by Fig. 7.11 is that one can not fully rely on an irregular LDSC code for achieving the channel capacity.

Given grouped power allocation, a superposition mapper offers a compression gain as well, but not for a repetition code with a spreading factor of $SF \geqslant G$. This can be explained via a simple example. Suppose that one wants to achieve a rate of 4 bits/symbol over the AWGN channel, and a 1/2 LDPC code is to be applied. Two possible choices would be to use ASK with $N = 8$ or SM-GPA with $G = 2$ and $L = 4$. By the previous derivation, it is evident that the minimum symbol distance of SM-GPA will be much larger than that of ASK, for such a setup. Consequently, the amount of coding gain necessary for the LDPC code to offer can be noticeably reduced, by using SM-GPA instead of ASK. On the other hand, if a 1/4 regular repetition code is applied, the performance of SM-GPA will be identical to that of ASK with $N = 4$. No loss occurs, but also no gain is achieved.

7.3 Suitable Redundancy for Superposition Mapping

The task of channel coding is to introduce a certain type of redundancy so as to protect the originally mutually independent info bits. Given a bijective uniform mapping scheme, the subject of error protection is merely the additive noise. In this case, the goal of code design is to achieve a coding gain as much as possible. Given a non-bijective nonuniform mapping scheme, the goal of code design becomes two-fold. The channel code has to first ensure the possibility of perfect data separation and then provide a necessary coding gain for combatting the additive noise. Therefore, code design for superposition mapping has to be treated in a way essentially different to that for conventional mapping schemes. In this section, we will have a brief discussion on the suitable type of redundancy for SM.

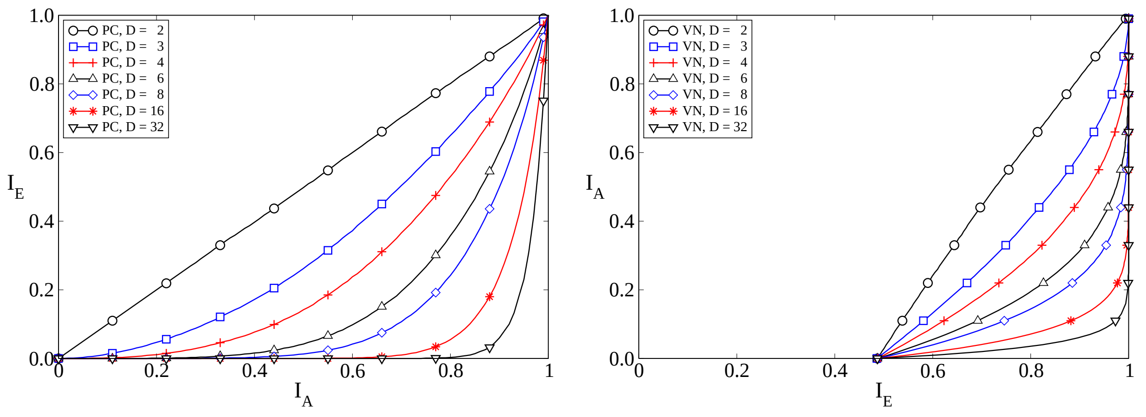
7.3.1 Repetitions vs. Parity Bits

In Chapter 5, we observe that regular parity-check codes are inferior to regular repetition codes for SM-EPA in the sense of supportable bandwidth efficiency. In Chapter 6, the high potential of irregular repetition-coded SM-EPA is revealed in the framework of LDSC coding. Hence, it deserves to be an interesting work to check the effectiveness of irregular parity-check codes for SM-EPA. It is also interesting to compare irregular repetition codes with irregular parity-check codes, so as to find their pros and cons respectively.

Without loss of generality, an LDPC code can be interpreted as a serial concatenation of a collection of repetition codes and a collection of single parity-check codes. Similarly, an LDSC code can be deemed a serial concatenation of a collection of repetition codes and a collection of single summation-check codes. LDPC codes offer superior performances for the binary-input AWGN channel. On the other hand, LDSC codes offer superior performances for the noiseless binary adder channel. Hence, there is indeed much commonality between LDPC coding and LDSC coding. Let us first have a comparison on two types of code constraints: parity checks (PC) and summation checks (SC). As a matter of fact, one may consider a parity check as an XOR summation check. Typically, the result of XOR summation is enforced to be zero, so as to impose a constraint among the involved code bits. In general, parity checks are "thirsty" for the inputting messages. Let $L_i^{(\mathrm{i})}$ and $L_i^{(\mathrm{o})}$ denote the ith input and output of a degree-D parity check. We have

$$L_i^{(\mathrm{o})} = \underset{1\leqslant j\leqslant D, j\neq i}{\boxplus} L_j^{(\mathrm{i})} \approx \operatorname{sign}\{L_i^{(\mathrm{o})}\} \min_{1\leqslant j\leqslant D, j\neq i}\{|L_j^{(\mathrm{i})}|\}\,, \qquad i = 1, 2, \ldots, D\,,$$

where $\boxplus$ is the box-plus operator [58]. With all-zero inputs, a parity check delivers all-zero outputs. Hence, the EXIT curve of parity checks always starts from $(0, 0)$, cf. Fig. 7.12(a). Besides, the strength of the outputting messages from a parity check is mainly determined



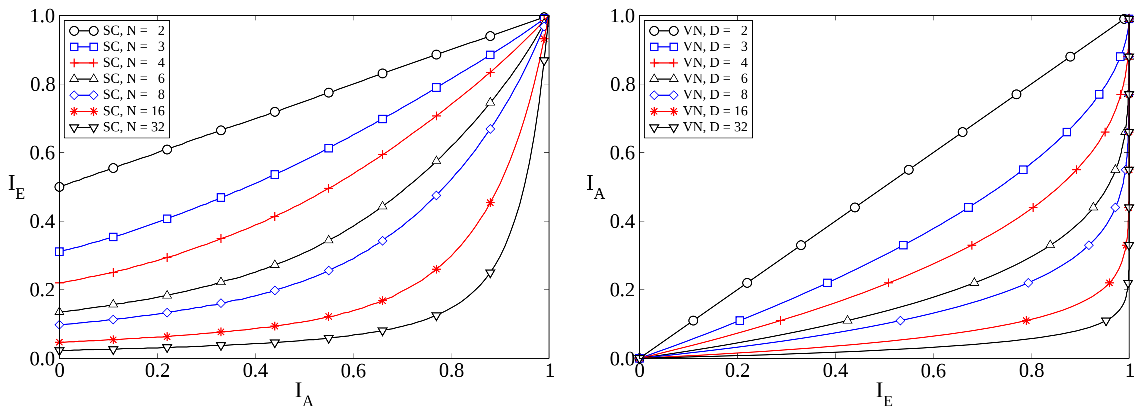
(a) Parity checks. (b) Variable nodes, $E_b/N_0 = 0$ dB.

Figure 7.12: EXIT charts for regular 1/2 LDPC codes, binary-input AWGN channel.

by the strength of the weakest inputting message. As long as one input is close to zero, the majority of the outputs will be close to zero as well. Furthermore, if more than one inputs are close to zero, all the outputs will be close to zero. Consequently, a parity check will deliver meaningful messages only if the a priori inputs are strong enough, particularly when the check degree is high. For example, a degree-32 parity check needs the a priori information to be as strong as 0.8 in order to deliver a nonzero extrinsic information. On the other hand, real summation checks have a very similar property in the sense of being "thirsty" for the inputting messages. Given equal power allocation, the EXIT curve of a summation check becomes more and more convex when one increases the bit load N, as shown in Fig. 7.13(a). The major difference between parity checks and summation checks is that a summation check is able to deliver a nonzero extrinsic information given a zero a priori information. This is because the result of an XOR addition is always binary while the result of a real addition is $(N+1)$-ary, assuming equal power allocation. According to the area property of EXIT charts [90–92], the EXIT curves of two concatenated decoders must fit with each other in order to obtain a good performance. For a collection of single parity-check codes, the best choice for concatenation is a collection of repetition codes. Given a degree-D repetition decoder, the extrinsic L-value of the ith bit is produced as

$$L_i^{(\mathrm{o})} = \sum_{1\leqslant j\leqslant D, j\neq i} L_j^{(\mathrm{i})} .$$

As long as one input is strong, the decoder will generate meaningful messages, which is in a sharp contrast to a single parity-check (SPC) decoder. For this reason, the EXIT curve of a variable node (repetition decoder) is also convex, when plotted on an EXIT chart with the abscissa and the ordinate swapped. From Fig. 7.12, one observes that the EXIT curves of variable nodes are in a good fit with the EXIT curves of parity checks. This actually gives the reason for the superior performance of LDPC codes for the binary-input AWGN channel. For LDPC decoding, the EXIT curve of variable nodes always starts from

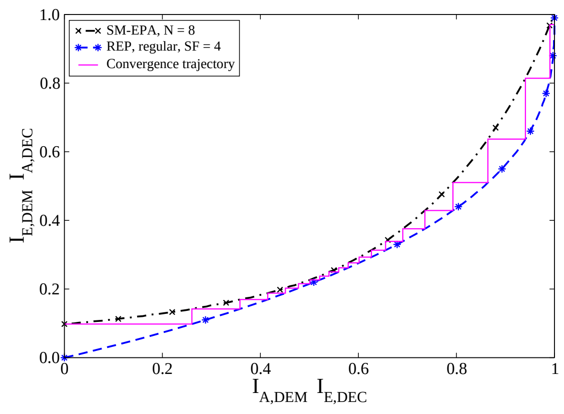


(a) Summation checks, EPA. (b) Variable nodes.

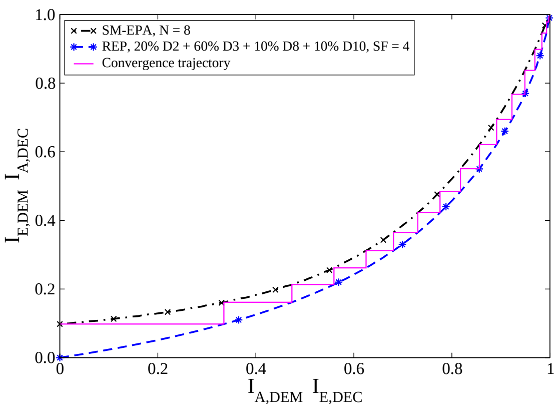
Figure 7.13: EXIT charts for regular LDSC codes, noiseless channel.

a nonzero position, since each variable node is associated with a channel observation. In comparison, for LDSC decoding, the EXIT curve of variable nodes always starts from $(0, 0)$, since the channel observations are not associated with the variable nodes. Similar to the case of LDPC decoding, the EXIT curves of variable nodes are in a good fit with the EXIT curves of summation checks, cf. Fig. 7.13, but not perfectly. When N is moderate, there will be a non-trivial distance between the starting point of the EXIT curve of a summation check, always given by $(\mathrm{I_A} = 0, \mathrm{I_E} > 0)$, and that of a variable node, always given by $(\mathrm{I_E} = 0, \mathrm{I_A} = 0)$. Since any area between two EXIT curves leads to a rate loss relative to the capacity, a pure repetition code is good for SM-EPA but not optimal, as long as the bit load N is moderate. When N is rather large, however, a pure repetition code is near-optimum for SM-EPA transmission over a noiseless channel, given a carefully designed irregular degree distribution. For example, in Section 6.4.5 we have devised an irregular repetition code that has an EXIT curve well matching to the EXIT curve of SM-EPA with $N = 32$. Easy to imagine, for N approaching the infinity, a pure repetition code will be optimal for SM-EPA, assuming a noiseless channel. Note that the uniquely decodable code introduced in Section 7.1.1 is indeed a special irregular repetition code with a deterministic scrambler and a deterministic interleaver, and it is already close to be optimal when N approaches the infinity. Hence, it is of no surprise that an irregular repetition code together with random interleaving will be capacity-achieving for SM-EPA transmission over a noiseless channel, with N approaching the infinity.

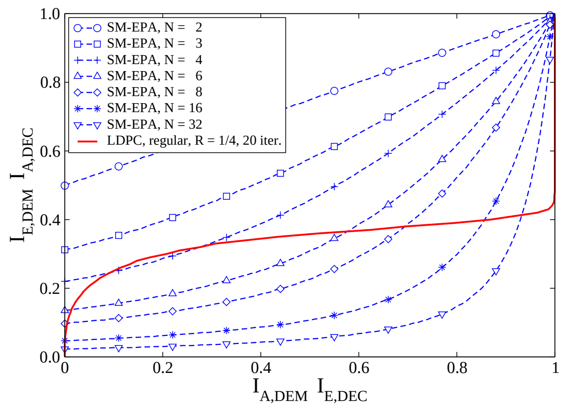
Now, let us come back to the main topic of this discussion, that is what type of redundancy is suitable for superposition mapping. To ease the discussion, we consider coded SM-EPA with $R = 1/4$ and $N = 8$ as an example. We assume a noiseless channel. As known from Chapter 6, a regular repetition code can already achieve a good performance in this case. Nevertheless, a large block length is necessary in order to achieve a decoding convergence,



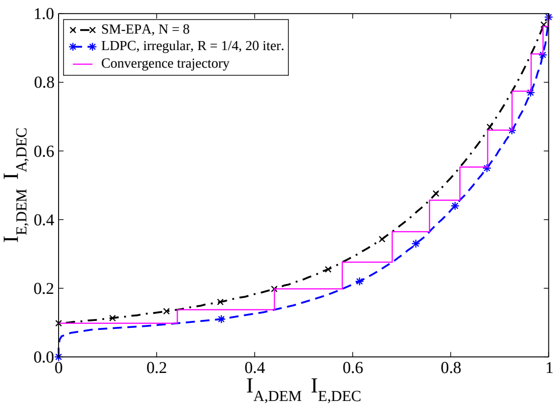
(a) Regular repetition-coded SM-EPA.



(b) Irregular repetition-coded SM-EPA.



(c) Regular LDPC-coded SM-EPA.



(d) Irregular LDPC-coded SM-EPA.

Figure 7.14: EXIT charts for coded SM-EPA transmission over a noiseless channel.

since the middle section of the convergence tunnel is very narrow, cf. Fig. 7.14(a). Using a carefully tuned irregular repetition code, the convergence tunnel gets widely open for the whole region, cf. Fig. 7.14(b). Hence, given the current system setup, redundancy in the form of simple repetitions deserves to be a good choice. One may also consider adding redundancy in the form of parity bits, i.e., apply a parity-check code for SM-EPA. The results in Section 5.4.2 imply that the supportable bit load given a regular rate 1/4 LDPC code is less than 4. This observation is confirmed by the EXIT chart analysis in Fig. 7.14(c), as the convergence tunnel is only open for $N \leqslant 3$. A regular LDPC decoder reaches $\mathrm{I_E} = 1$ when $\mathrm{I_A} \ll 1$, which corresponds to a considerable coding gain for transmission over a noisy channel. However, there is a big penalty in the left region of the EXIT curve. Compared to a repetition decoder, a regular LDPC decoder needs a much stronger a priori information before it can provide any meaningful extrinsic information, which is largely due to the special property of parity checks, cf. Fig. 7.12(a). Clearly, regular LDPC codes are not suitable for SM-EPA transmission, particularly when N is large. The situation can be improved by employing an irregular variable node degree distribution, albeit with a considerable degradation on the achievable coding gain. For example, consider a rate 1/4 irregular LDPC code with the following degree distributions

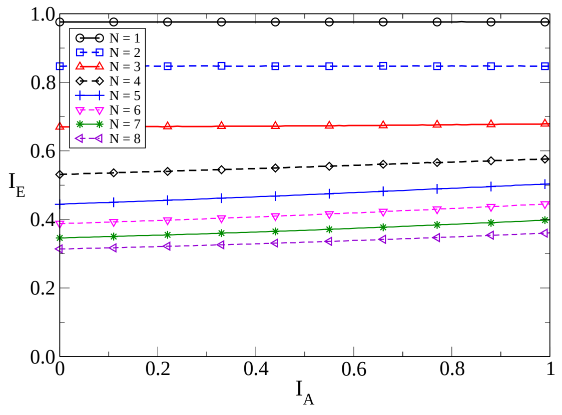
$$\begin{aligned}\lambda(D) &= 0.76D^1 + 0.18D^2 + 0.02D^{10} + 0.02D^{24} + 0.01D^{40} + 0.01D^{80} \\ \eta(D) &= 1.0D^6 ,\end{aligned}$$

where $\eta(D)$ stands for the parity check degree distribution. As shown in Fig. 7.14(d), this code is able to open the convergence tunnel for SM-EPA with $N = 8$. Nevertheless, in order to open the tunnel in the leftmost region, the highest VN degree is set to be 80. As a result, a dominating number of variable nodes are now with degree 1. Consequently, the decoder is no longer able to provide any meaningful coding gain. With such a code design, there is indeed no practical benefit to use an LDPC code instead of a repetition code, given the current system setup. Note that the encoding and decoding complexity of LDPC codes is significantly higher than that of repetition codes. Later on, we will show that in order to let an LDPC code offer an optimal performance for SM-EPA and SM-GPA one has to introduce considerable irregularity into the parity check degree distribution. Last but not least, by comparing Fig. 7.14(b) and Fig. 7.14(d), we find that a repetition code is more effective in the left region, i.e., in the early stage of iterative decoding, while an LDPC code is more effective in the right region, i.e., in the late stage of iterative decoding. This observation indicates that to achieve a satisfying performance, either in the sense of power efficiency or bandwidth efficiency, a good practice is to apply a serial concatenation of LDPC code and repetition code, so as to combine the strength from both codes and compensate the weakness for each, which gives the topic for Section 7.3.2.

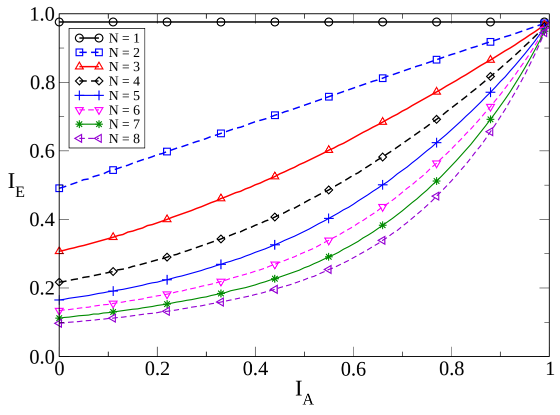
7.3.2 Repetitions plus Parity Bits

For the sake of simplicity, we have been always assuming a noiseless channel in the previous discussion. Moreover, we have merely focused on superposition mapping with equal power allocation. In the following discussion, we will treat the issue of coded SM transmission over the AWGN channel, given three power allocation strategies: EPA, UPA, and GPA.

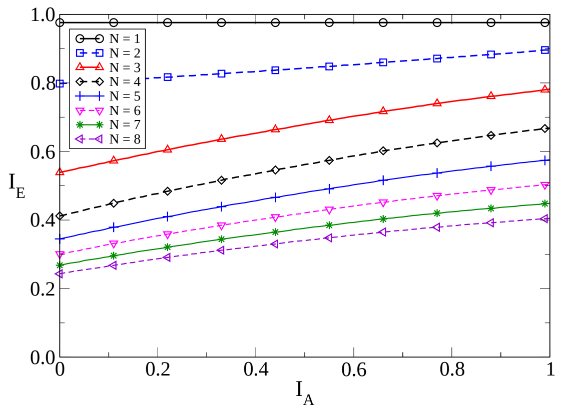
As a preliminary remark, let us first make a comparison between the conventional ASK and SM, for the sake of iterative decoding and demapping. Fig. 7.15(a) provides a set of EXIT charts for the relevant discussion. For a fair comparison, we have fixed the SNR per code bit to be always 5 dB regardless of the mapping format. It is a common knowledge that, given a receiver that does not perform iterative decoding and demapping, the preferable mapping format for the AWGN channel is ASK with Gray labelling. The reason becomes evident by checking the EXIT curves in Fig. 7.15(a). These “curves” are almost straight and more or less horizontal. This means that the extrinsic information from an ASK demapper does not really become stronger by receiving a strong a priori input. This also



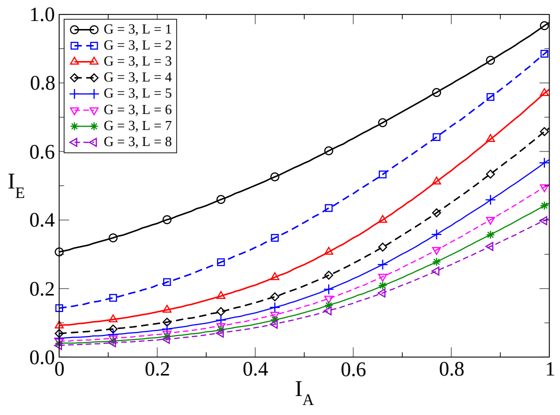
(a) ASK with Gray labelling.



(b) SM-EPA.



(c) SM-UPA.



(d) SM-GPA.

Figure 7.15: EXIT charts for SISO demapping, AWGN channel, $E_c/N_0 = 5$ dB.

means that the initial output message from an ASK demapper is almost at the strongest possible level. Consequently, a non-iterative receiver is able to achieve a near-optimum performance for ASK. However, one should also not expect any meaningful performance gain via iterative decoding and demapping. The starting and the ending position of the EXIT curve drops when one increases the bit load N, because the minimum symbol distance decreases w.r.t. N. As a non-trivial issue, ASK mapping is strictly non-capacity-achieving for the AWGN channel due to a uniform symbol distribution. Now, let us check the situation of SM-EPA, given the same SNR per code bit. One observes from Fig. 7.15(b) that an SM-EPA demapper has an essentially different behaviour from an ASK demapper. Compared to ASK demapping, the EXIT curve of SM-EPA demapping starts from a considerably lower point, which is due to the inter-chip interference. On the other hand, given a fixed E_c/N_0 the EXIT curve of SM-EPA demapping always ends at a fixed position, regardless of the bit load N. The reason behind this phenomenon is clear. That is for SM-EPA the minimum symbol distance does not decrease w.r.t. the bit load. It is also easy to find that the ending position of SM-EPA demapping is significantly higher than that of ASK demapping, for all $N >= 2$. Note that such a difference in the height of the ending position leads to a compression gain, assuming a well-matched channel code. Besides, as the starting point and the ending point of SM-EPA demapping significantly differ in the height, iterations between the decoder and the demapper are mandatory for obtaining a good performance. When an unequal power allocation strategy is applied, the situation of superposition demapping is similar to that of ASK demapping, as shown in Fig. 7.15(c). This is because SM-UPA is equivalent to ASK with natural labelling. Nevertheless, as the EXIT curves have a noticeable slope, iterations between the decoder and the demapper are also necessary for systems employing SM-UPA. Finally, one observes from Fig. 7.15(d) that the extrinsic-information-transfer behaviour of an SM-GPA demapper is simply a hybrid of that of SM-EPA and SM-UPA. Important to be mentioned, the ending position of SM-GPA demapping is merely influenced by the number of power levels L but not the group size G. Given $G > 1$, SM-GPA is also able to provide a compression gain w.r.t. ASK and SM-UPA.

For a coded transmission system applying iterative decoding and demapping, the ultimate criterion for the channel code is not weak or strong but matched or unmatched. Fig. 7.15 shows that a superposition demapper behaves differently given different power allocation strategies. Therefore, the optimal coding strategy for SM will also be dependent on the adopted power allocation scheme. It is relatively easy to imagine that an irregular Turbo code or an irregular LDPC code will be optimal for SM-UPA, as it has a very similar property from ASK. For SM-EPA and SM-GPA, it is so far unclear which kind of channel codes are optimal. A repetition code can offer a good performance for SM-EPA, but often not the best, particularly when the transmission channel is not noise-free. On the other hand, a typical design of irregular LDPC codes looks not optimal as well, in the sense of providing no meaningful coding gain, cf. Fig. 7.14(d). Compared to conventional ASK, both SM-EPA and SM-GPA need less coding gain for achieving the capacity. Meanwhile, SM-EPA and SM-GPA both require the channel code to ensure the separability of superimposed chips. Hence, a suitable channel code for SM-EPA or SM-GPA should behave similarly to a repetition code that enables an efficient data separation, in the early stage of iterative decoding and demapping, and behave similarly to a parity-check code that offers a moderate coding gain, in the late stage of iterative decoding and demapping. Then, a natural question would be if a serial concatenation of parity-check code and repetition code can do a desirable work in this concern. Considering the low complexity of a repetition codec, this question is also of great practical interest.

Suppose that we want to devise a transmission system that achieves 2 bits/symbol over the AWGN channel. According to Fig. 3.7(b), SM-EPA with $N = 8$ is capacity-achieving at this rate. The Shannon limit for 2 bits/symbol is at about 5.8 dB. To give some space for practical imperfectness, let us target at a decoding threshold of 8 dB. An intuitive

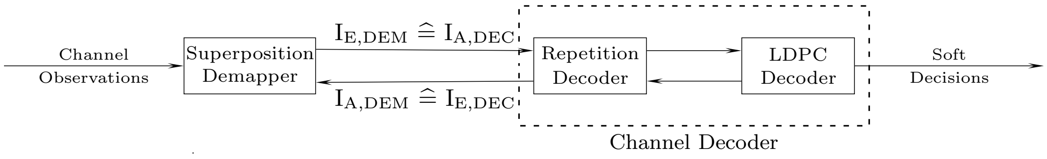
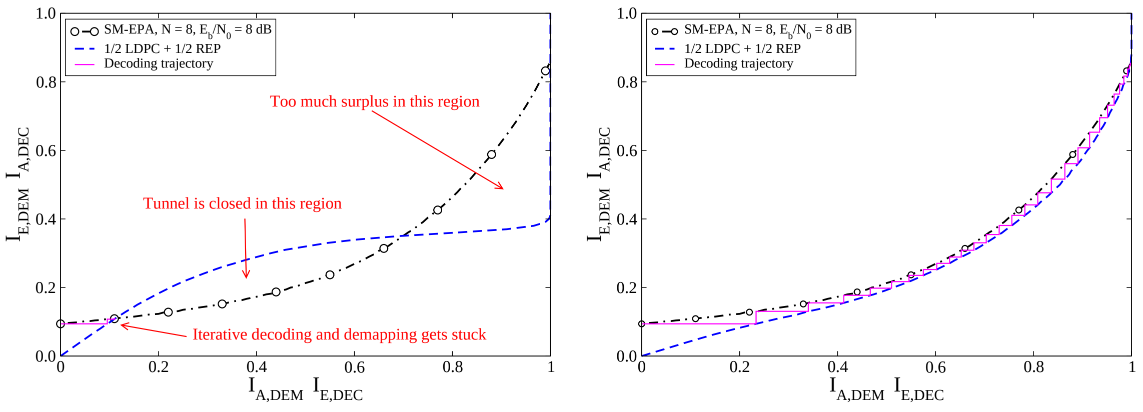


Figure 7.16: The interaction between the superposition demapper (DEM) and the channel decoder (DEC). The LDPC decoder needs to perform some local iterations.



(a) Regular LDPC + regular REP.

(b) Irregular LDPC + irregular REP.

Figure 7.17: SM-EPA with serially concatenated LDPC code and repetition (REP) code. $R_p = 1/2$, $R_r = 1/2$, $R = R_p \cdot R_r = 1/4$. The LDPC decoder performs 20 local iterations.

coding strategy is to apply a serial concatenation of a rate $R_p = 1/2$ LDPC code and a rate $R_r = 1/2$ repetition code. The corresponding EXIT chart testing model is provided in Fig. 7.16. Note that this is the conventional way of EXIT chart analysis for coded transmission system applying iterative decoding and demapping. A tricky issue here is that one has to perform some LDPC-local iterations in order to obtain a good performance prediction. For a systematic study, we first investigate the situation when the LDPC code and the repetition code are both regular. Fig. 7.17(a) gives the resulting EXIT chart. It can be seen that serially concatenating an LDPC code and a repetition code does combine the nice features from both codes and largely mitigate the weaknesses from both codes. In the early stage of iterations, the decoder acts similarly to a repetition decoder, as it starts to deliver meaningful extrinsic information given a very weak a priori input. In the late stage of iterations, the decoder acts similarly to an LDPC decoder, as it reaches $I_{E,DEC} = 1$ for $I_{A,DEC} \ll 1$, i.e., it provides a considerable coding gain. Nevertheless, the convergence tunnel is closed in the middle region. Hence, a decoding convergence is not achievable given such a code design. However, one observes that there is a lot of surplus in the right region, which means that a regular 1/2 LDPC code provides too much coding gain for SM-EPA with $N = 8$. By using irregular degree distributions, we should be able to utilize the surplus in the right region to open the tunnel in the middle region. We apply

$$\lambda(D) = 0.700D^1 + 0.080D^3 + 0.120D^4 + 0.020D^5 + 0.080D^6 \qquad (7.37)$$

for the repetition code, and we apply

$$\begin{aligned}\lambda_(D) &= 0.900D^2 + 0.010D^6 + 0.040D^9 + 0.010D^{12} + 0.020D^{15} + 0.020D^{18}\\ \eta(D) &= 0.610D^4 + 0.200D^5 + 0.040D^8 + 0.020D^{10} + 0.080D^{11} + 0.030D^{22} + 0.020D^{25}\end{aligned}$$

for the LDPC code[1]. The EXIT chart in Fig. 7.17(b) shows that the above code design successfully opens the tunnel for $E_b/N_0 = 8$ dB. Now, the LDPC decoder only provides a moderate but sufficient coding gain. Meanwhile, it helps the repetition decoder in opening the tunnel in the middle region. One observes from Fig. 7.17(b) that the convergence tunnel is widely open for the whole region. Hence, to achieve a good performance, the component LDPC code and the component repetition code should both be irregular. From the EXIT chart in Fig. 7.17(b), the exact decoding threshold for this code design is even lower than 8 dB. Nevertheless, it is so far unclear whether the promised decoding threshold is easily achievable given a finite block length. We will provide a detailed performance analysis for this code design in Section 7.4.4.

The above tests demonstrate that serially concatenating an LDPC code and a repetition code gives a good practical coding approach for SM-EPA. By comparing Fig. 7.15(b) with Fig. 7.15(d), we may safely conjecture that this statement also holds for SM-GPA. However, there are several problems when using the conventional EXIT chart analysis for designing this type of codes. First, as the LDPC decoder takes some local iterations, the EXIT chart analysis is indeed interleaver-dependent. In other words, the resulting performance prediction is only accurate for a specific system with a specific interleaver, and certainly for a finite block length only. This leads to a problem when one tries to design a capacity-achieving system, which is only possible by assuming an infinite block length. Second, by every small adjustment on the degree distributions, a new simulation is necessary to measure the EXIT curve of the decoder. A typical code optimization commonly involves many times of degree distribution adjustments. Besides, an LDPC code gives a superior performance only if the block length is large enough. In the end, such a simulation-based EXIT chart analysis turns out to be not less time-consuming than a direct BER performance test. Note that, for obtaining each EXIT curve, one needs to perform decoding for a big number of $\mathrm{I_A}$'s. In comparison, a typical BER performance test only involves a few SNR points. Therefore, to facilitate an efficient and accurate code design for SM-EPA and SM-GPA, the EXIT chart analysis method has to be improved. We will provide a competent solution in Section 7.4.3.

[1]The advantages as well as disadvantages of applying an irregular parity check degree distribution will be discussed in Section 7.4.

7.4 Low-Density Hybrid-Check Code

For a long time, the optimal coding strategy for SM has been uncertain. Summarizing all the previous discussions in this thesis, we are now ready to clarify fundamental issues on channel coding for SM, and sequentially provide effective solutions. Chapter 6 shows that the previously known rate limit of SM-EPA of about 2 bits/symbol can easily be broken by using an irregular repetition code. Furthermore, with a carefully designed interleaver and a scrambler, irregular repetition-coded SM-EPA is able to work at a rate close to the capacity, cf. Section 6.4.5. Hence, an optimal channel code for SM-EPA should possess most of the properties that an irregular repetition code has, such that an efficient data separation can be achieved. For transmission over the AWGN channel, an important issue is to combat the additive noise in an efficient way. The investigation in Section 7.3.2 demonstrates that a serial concatenation of LDPC code and repetition code can offer a desirable performance, when both component codes adopt carefully designed irregular degree distributions. Due to offering a considerable compression gain, an SM-EPA or an SM-GPA demapper requires a much smaller coding gain w.r.t. an ASK demapper. Hence, an optimal channel code for SM-EPA or SM-GPA should possess some properties that a "weak" parity-check code has, such that a small but enough coding gain is provided. In this section, we propose a universal coding framework, called low-density hybrid-check (LDHC) coding, to facilitate the optimization of channel codes for SM.

7.4.1 Basic Principle

When interpreting a coded modulation system via a factor graph, researchers typically encapsulate the signal demapping operation by a channel observation node, e.g., the case in Fig. 6.27(a). This approach is correct when BPSK mapping is applied. In fact, this approach is generally appropriate for systems employing ASK mapping, since the output messages from an ASK demapper have a very weak dependence on the a priori input, cf. Fig. 7.15(a). However, for a coded superposition modulation system, this approach is no longer appropriate. As shown in Fig. 7.15, the extrinsic output from a superposition demapper is strongly dependent on the a priori input. Therefore, one should encapsulate the superposition demapping operation by a check node and treat it as a special type of code constraints. In this sense, superposition mapping is merely an integral part of the overall coding scheme. The successful applications of LDSC coding in Chapter 6 clearly supports this way of thinking. The relevant discussions show that it is beneficial to treat the serial concatenation of a repetition encoder (optionally including a scrambler), an interleaver, and a superposition mapper as an LDSC encoder. Following a

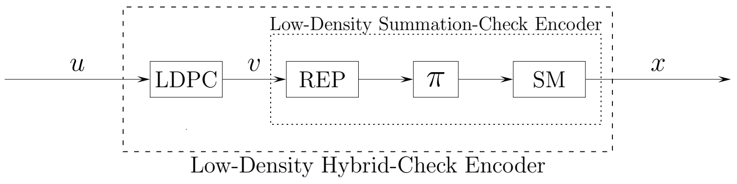
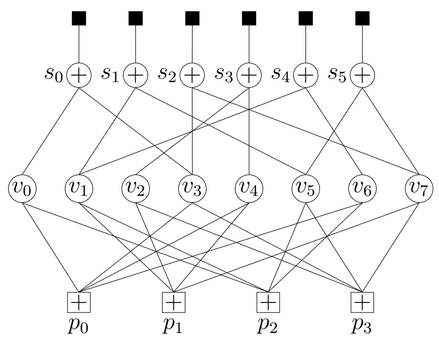

Low-Density Summation-Check Encoder
u
LDPC
v
REP
π
SM
x
Low-Density Hybrid-Check Encoder

Figure 7.18: SM with a serial concatenation of LDPC code and repetition code.



(a) Factor graph.

$$
\begin{array}{c|cccccccc|c}
 & v_0 & v_1 & v_2 & v_3 & v_4 & v_5 & v_6 & v_7 & \\
\hline
\text{LDSC} & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & \leftarrow s_0 \\
 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & \leftarrow s_1 \\
 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & \leftarrow s_2 \\
 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & \leftarrow s_3 \\
 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & \leftarrow s_4 \\
 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & \leftarrow s_5 \\
\hline
\text{LDPC} & 1 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & \leftarrow p_0 \\
 & 0 & 1 & 1 & 0 & 1 & 0 & 0 & 1 & \leftarrow p_1 \\
 & 1 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & \leftarrow p_2 \\
 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 1 & \leftarrow p_3 \\
\hline
 & u_0 & u_1 & u_2 & u_3 & & & & &
\end{array}
$$

(b) Incidence matrix.

Figure 7.19: Low-density hybrid-check code.

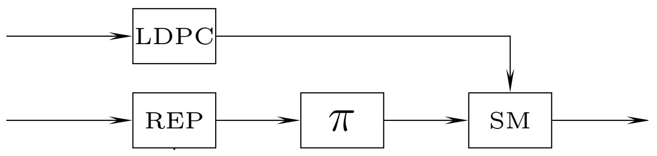
similar track, we may treat the serial concatenation of an LDPC encoder and an LDSC encoder as an LDHC encoder, illustrated in Fig. 7.18. The basic principle of LDHC coding can be understood in two steps. The first step is to construct a factor graph for the complete transmission system as in Fig. 7.19(a). One takes the output bits of the LDPC encoder, i.e., the input bits to the repetition encoder, as variable nodes. Due to the LDPC encoder, each variable node is connected with a certain amount of parity checks, which are represented by $\boxplus$ in Fig. 7.19(a). Due to the LDSC encoder, each variable node is connected with a certain amount of summation checks, which are represented by $\oplus$ in Fig. 7.19(a). Naturally, the result of each summation check corresponds to an output symbol and is therefore associated with a channel observation, which is marked by ■ in Fig. 7.19(a). The second step is to construct an overall incidence matrix for the variable nodes. For example, given the graph in Fig. 7.19(a), the corresponding incidence matrix is shown in Fig. 7.19(b). Each column of the matrix is associated with a variable node, i.e., a code bit. Each row in the upper part of the matrix represents one summation check, while each row in the lower part of the matrix stands for one parity check. In case that the LDPC encoder is systematic, some columns are also directly associated with info bits. Given a reasonable block length, this incidence matrix will be of low density. Noting that there are two types of code checks, we dub this matrix a low-density hybrid-check matrix. The corresponding code we call a low-density hybrid-check code. The upper part of the matrix gives a low-density summation-check matrix, and the lower part of the matrix gives a low-density parity-check matrix. For clearness and compactness, in Fig. 7.19 we have not explicitly described the scrambling operation often necessary in an LDSC code.

Adopting the concept of LDHC coding brings many benefits. Constructing the LDHC matrix via a well-controlled random procedure, e.g., the PEG algorithm, short cycles can be removed in a global level, i.e., no summation checks will form short cycles, no parity checks will form short cycles, and no summation checks will form short cycles with parity checks. Hence, an LDHC matrix enables a global-level interleaver optimization. Later on, we will show that this is rather beneficial for superposition mapping. Besides, the degree distribution of the LDPC code and the degree distribution of the LDSC code can be optimized jointly. During an iterative decoding process, all elements in the factor graph function in an interactive way. Hence, a joint code optimization often brings a significant performance gain. Furthermore, the all-in-one LDHC matrix provides a versatile platform for tuning the degree combination of variable nodes, i.e., the way of combining the summation-check-side repetition degrees and the parity-check-side repetition degrees, which significantly influences the achievable performance as well.

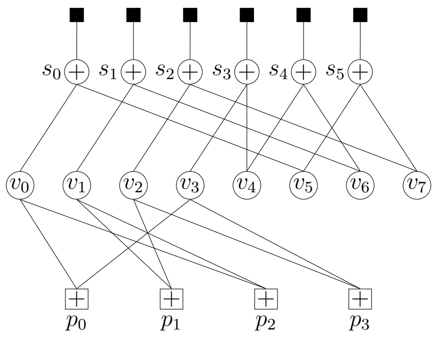
7.4.2 Compatible Code Structures

In the previous section, we have derived the concept of LDHC coding from an SM system employing a serial concatenation of LDPC code and repetition code. In fact, the LDHC coding architecture is also compatible with a parallel concatenation of LDPC code and repetition code. Moreover, it supports purely LDPC-coded SM as well.

Fig. 7.20(a) gives a scenario that the LDPC encoder and the repetition encoder are parallel concatenated. In this case, some code bits are protected only by parity bits and some code bits are protected only by repetitions. The corresponding factor graph will look like Fig. 7.20(b). One observes that some variable nodes are connected with multiple parity checks but a single summation check, and some are connected with multiple summation checks but no parity check at all. This code structure can also easily be identified from the corresponding incidence matrix given in Fig. 7.20(c). One observes that some columns of the LDHC matrix have a single nonzero entry in the upper part but multiple nonzero entries in the lower part, and some have multiple nonzero entries in the upper part but no nonzero entry in the lower part. Fig. 7.21 describes a scenario that a hybrid type of code concatenation is applied. In this case, some code bits are protected only by parity bits, e.g., v_0 and v_4, some are protected only by repetitions, e.g., v_5 and v_7, and some are protected both by parity bits and repetitions, e.g., v_1 and v_3. After all, the important message is that the architecture of LDHC coding is rather versatile and widely applicable. As a matter of fact, when designing an LDHC code, one does not need to specifically make a distinction between the underlying code structures. The only important thing is to find valid variable/check node degree distributions that will offer a desired performance.



(a) Transmitter structure.

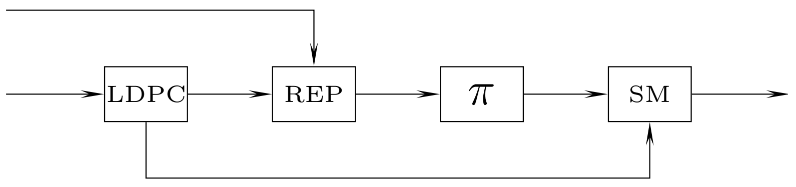

s_0 s_1 s_2 s_3 s_4 s_5
v_0 v_1 v_2 v_3 v_4 v_5 v_6 v_7
p_0 p_1 p_2 p_3

(b) Factor graph.

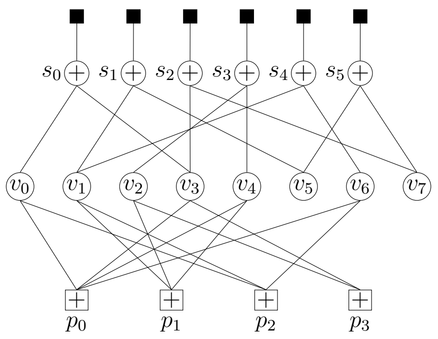
$$\begin{array}{c|cccccccc|c}
 & v_0 & v_1 & v_2 & v_3 & v_4 & v_5 & v_6 & v_7 & \\
 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & \leftarrow s_0 \\
 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & \leftarrow s_1 \\
\text{LDSC} & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & \leftarrow s_2 \\
 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & \leftarrow s_3 \\
 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & \leftarrow s_4 \\
 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & \leftarrow s_5 \\
\hline
 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & \leftarrow p_0 \\
\text{LDPC} & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & \leftarrow p_1 \\
 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & \leftarrow p_2 \\
 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & \leftarrow p_3
\end{array}$$

(c) LDHC matrix.

Figure 7.20: SM with a parallel concatenation of LDPC code and repetition code.



(a) Transmitter structure.

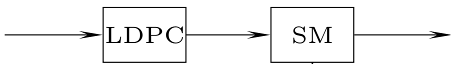

s_0 s_1 s_2 s_3 s_4 s_5
v_0 v_1 v_2 v_3 v_4 v_5 v_6 v_7
p_0 p_1 p_2 p_3

(b) Factor graph.

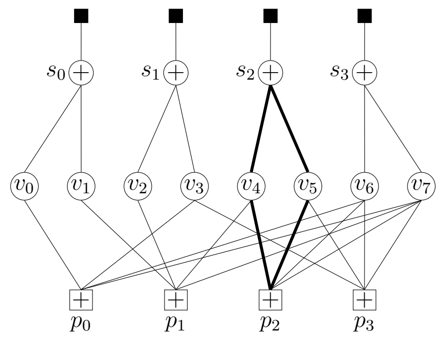
$$\begin{array}{c|cccccccc|c}
 & v_0 & v_1 & v_2 & v_3 & v_4 & v_5 & v_6 & v_7 & \\
 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & \leftarrow s_0 \\
 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & \leftarrow s_1 \\
\text{LDSC} & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & \leftarrow s_2 \\
 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & \leftarrow s_3 \\
 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & \leftarrow s_4 \\
 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & \leftarrow s_5 \\
\hline
 & 1 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & \leftarrow p_0 \\
\text{LDPC} & 0 & 1 & 1 & 0 & 1 & 0 & 0 & 0 & \leftarrow p_1 \\
 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & \leftarrow p_2 \\
 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & \leftarrow p_3
\end{array}$$

(c) LDHC matrix.

Figure 7.21: SM with a hybrid-type concatenation of LDPC code and repetition code.



(a) Transmitter structure.



(b) Factor graph.

$$\begin{array}{c|cccccccc|c}
 & v_0 & v_1 & v_2 & v_3 & v_4 & v_5 & v_6 & v_7 & \\
\hline
\text{LDSC} & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & \leftarrow s_0 \\
 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & \leftarrow s_1 \\
 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & \leftarrow s_2 \\
 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & \leftarrow s_3 \\
\hline
\text{LDPC} & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & \leftarrow p_0 \\
 & 0 & 1 & 1 & 0 & 1 & 0 & 0 & 1 & \leftarrow p_1 \\
 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & \leftarrow p_2 \\
 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 1 & \leftarrow p_3
\end{array}$$

(c) LDHC matrix.

Figure 7.22: SM transmission with a pure LDPC code, sorted VN degree alignment.

In addition to the previously described cases, there is another important application for the concept of LDHC coding. For SM transmission with pure LDPC coding, the LDHC matrix is still useful. As shown in Fig. 7.22, for purely LDPC-coded SM, all variable nodes have a unique edge connected to a summation check, and consequently all columns of the LDHC matrix have a unique nonzero entry in the upper part. It is true that these single 1's in the upper part of the LDHC matrix play no role for the optimization of degree distributions. However, they are important for the optimization of interleaver patterns. Depicted in Fig. 7.22(c), by leaving a single 1 in each column of the upper part of the incidence matrix, we can easily detect and sequentially remove short cycles formed between summation checks and parity checks. These hybrid type of cycles are often harmful for the stability of the iterative decoder. One may easily find from Fig. 7.22(c) that the LDSC sub-matrix has a Toeplitz structure. This eliminates the necessity of an extra interleaver between the LDPC encoder and the superposition mapper. As a result, only the LDPC sub-matrix needs to be constructed by using a certain interleaver design method. For LDPC-coded modulation, the issue of variable node (VN) degree alignment is commonly ignored, mainly due to the horizontal EXIT curve of conventional mapping schemes. For LDPC-coded SM, however, this issue should be carefully taken into account, as the VN degree alignment has a big impact on the summation check EMD distribution. Fig. 7.22(b) gives an example that the VN degrees are sorted in a non-descending order in the graph. From the left to the right, the variable nodes have parity-check-side repetition degrees: $[1, 1, 1, 2, 2, 2, 3, 4]$. Consequently, from the left to the right, the summation checks have EMD's: $[2, 3, 4, 7]$, which has a wide and non-concentrated distribution. Given certain degree distributions, a sorted VN degree alignment is advantageous for the left region of the convergence tunnel but disadvantageous for the right region of the convergence

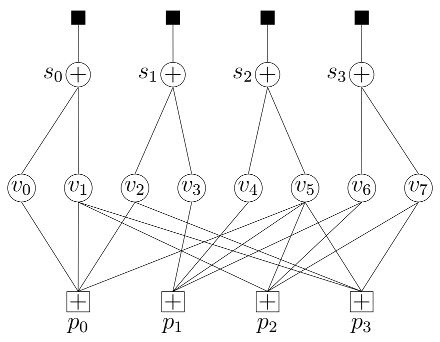
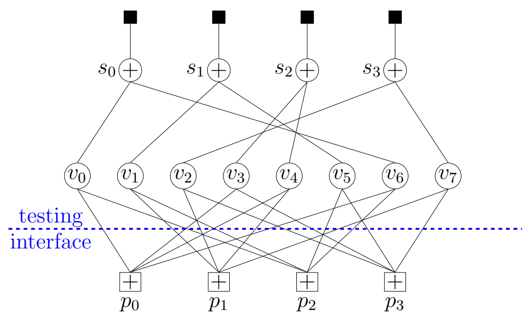


Figure 7.23: Factor graph for LDPC-coded SM, unsorted VN degree alignment.

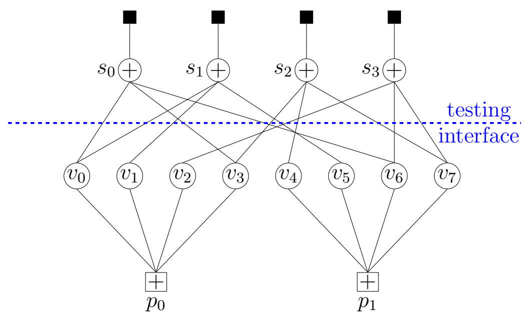
tunnel. In contrast, one may apply an unsorted (randomized) VN degree alignment, as shown in Fig. 7.23. In this case, the variable nodes have parity-check-side repetition degrees: $[1, 3, 2, 1, 1, 4, 2, 2]$, from the left to the right. Correspondingly, the summation checks have EMD's: $[4, 3, 5, 4]$, which has a narrower and more concentrated distribution. Therefore, an unsorted VN degree alignment is often advantageous for the right region of the convergence tunnel but leads to a degradation in the left region of the convergence tunnel. Note that the above statements are given under the assumption of an infinite block length. In practice, however, an unsorted VN degree alignment is more desirable, because it reduces the probability of residual errors caused by those summation checks with an unnecessarily low EMD. We will come back to this topic in later discussions.

7.4.3 Degree Distribution & Degree Combination

Similar to the design of LDSC codes and LDPC codes, the first step for LDHC code optimization is to identify an optimal or near-optimal degree distribution for the variable nodes. A new issue for LDHC codes is that the global degree distribution of the variable nodes in fact consists of two local degree distributions, one for the component LDSC code and one for the component LDPC code. To achieve the best performance, the two local degree distributions need to be optimized jointly instead of separately. Whenever the degree distribution of the LDSC code is adjusted, the degree distribution of the LDPC code needs to be adjusted as well, and vice versa. Inherently, an optimized global degree distribution of an LDHC code also contains an optimized degree combination scheme, which determines the way of combining SC-side repetition degrees and PC-side repetition degrees on each variable node. A brute-force approach for optimizing the global degree distribution, including the degree combination scheme, is the try-and-test method. One tries a large collection of parameter sets and test the corresponding performances by means of Monte Carlo simulations. This approach is valid always but is often impractical whenever the parameter space has a high dimension. Density evolution [56, 57] and



(a) The SC-side repetition degrees are all 1.



(b) The PC-side repetition degrees are all 1.

Figure 7.24: Two special cases of LDHC coding for which a two-part EXIT chart analysis is applicable. "SC" stands for summation checks, and "PC" stands for parity checks.

EXIT chart analysis [84–86] are two popular semi-analytical methods for predicting the performance of LDPC codes with iterative decoding. These two methods can be applied for LDHC codes as well, though some modifications are necessary. Generally speaking, density evolution is more accurate in predicting the decoding threshold. However, a big drawback from density evolution is that it does not provide any intuition on how a degree distribution can be improved further. This feature from density evolution is particularly undesirable for the optimization of LDHC codes. Since we basically need to optimize two local degree distributions in once, an indication on the direction of further improvement is rather beneficial. For this reason, EXIT chart analysis deserves to be the preferable choice for LDHC code optimization. In the following, we will introduce the usage of EXIT charts in identifying optimal or near-optimal degree distributions for LDHC codes.

Conventional EXIT Chart Analysis

An important assumption from an EXIT chart analysis is that the involved factor graph is cycle-free, which is also the primary assumption for density evolution. Nevertheless, given a realistic parameter setup and a finite block length, the factor graph will usually be non-cycle-free. By a conventional EXIT chart analysis, one divides a non-cycle-free factor graph into two parts, with each part being cycle-free. Checking the extrinsic-information-transfer property of these two parts, a good prediction can be obtained for the theoretically achievable performance assuming a cycle-free graph. Certainly, the prerequisite for applying such a two-part EXIT chart analysis is that the graph can indeed be divided into two cycle-free parts. This is actually the case for many coding techniques, e.g., LDPC coding and turbo coding. Not difficult to find, the EXIT chart analyses in the former discussions all follow such a two-part treatment. However, for LDHC coding, a two-part EXIT chart analysis is only suitable for two special cases, as depicted in Fig. 7.24. For LDPC-coded SM transmission, the factor graph will be as in Fig. 7.24(a),

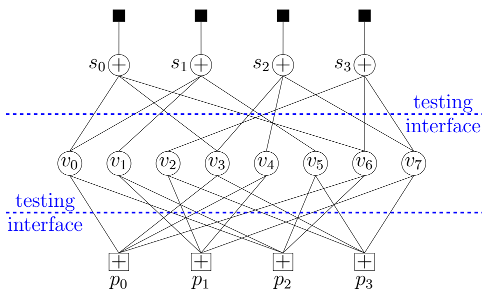


Figure 7.25: The way of graph division for a three-part EXIT chart analysis.

where all variable nodes have degree-1 SC-side repetitions. In this case, cycles can only be formed between variable nodes and parity checks. Consequently, one may set the testing interface between the ensemble of variable nodes and the ensemble of parity checks, cf. Fig. 7.24(a). Doing so results in two sub-graphs free of cycles. Note that the upper sub-graph consists of two types of nodes, i.e., summation checks and variable nodes, but is cycle-free due to degree-1 repetitions. A two-part EXIT chart analysis based on a graph division given in Fig. 7.24(a) is independent of the interleaving pattern. There is another special case of LDHC coding that a two-part EXIT chart analysis is suitable. That is the component LDPC code is simply a collection of single-parity-check (SPC) codes. In this scenario, all variable nodes will have degree-1 PC-side repetitions, as shown in Fig. 7.24(b). Consequently, one may set the testing interface between the ensemble of summation checks and the ensemble of variable nodes. Dividing the graph along this testing interface breaks all the cycles formed between summations and variable nodes. Clearly, the resulting EXIT chart analysis is also independent of the interleaving pattern. In more general cases, when larger-than-1 repetition degrees exist both in the SC-side and the PC-side, the corresponding graph can no longer be divided into two cycle-free parts. To obtain an accurate prediction for the theoretically achievable performance, we need to divide the graph into three parts that are all cycle-free, which necessitates a three-part EXIT chart analysis method, to be proposed in the following.

An EXIT Emulation Technique for LDHC Decoding

Instead of using a single testing interface, we may use two testing interfaces for the EXIT chart analysis. As illustrated in Fig. 7.25, we divide the graph of an LDHC code into three ensembles: summation checks, variable nodes, and parity checks. Clearly, the resulting sub-graphs are all cycle-free. The question is how to perform an EXIT chart analysis given such a graph division. Without loss of generality, we may interpret the interaction between these three ensembles as in Fig. 7.26. During an iterative LDHC decoding process, the messages originate from the summation checks, i.e., from channel observations. Following

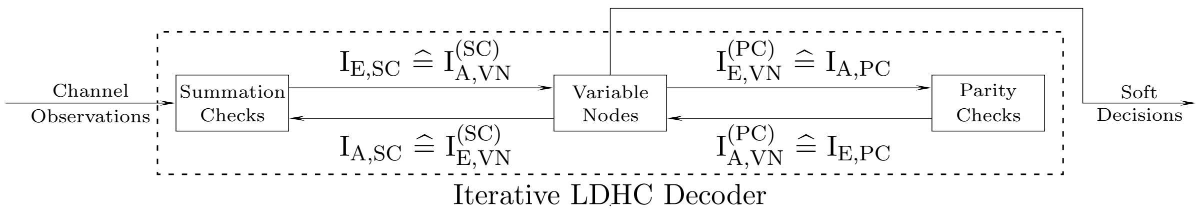

Channel Observations
Summation Checks
$\mathrm{I}_{\mathrm{E,SC}} \widehat{=} \mathrm{I}_{\mathrm{A,VN}}^{(\mathrm{SC})}$
$\mathrm{I}_{\mathrm{A,SC}} \widehat{=} \mathrm{I}_{\mathrm{E,VN}}^{(\mathrm{SC})}$
Variable Nodes
$\mathrm{I}_{\mathrm{E,VN}}^{(\mathrm{PC})} \widehat{=} \mathrm{I}_{\mathrm{A,PC}}$
$\mathrm{I}_{\mathrm{A,VN}}^{(\mathrm{PC})} \widehat{=} \mathrm{I}_{\mathrm{E,PC}}$
Parity Checks
Soft Decisions
Iterative LDHC Decoder

Figure 7.26: The message flow during an iterative LDHC decoding process.

that, these messages are refined and strengthened by the variables nodes and sequentially forwarded to the parity checks. Upon receiving a priori messages from the variable nodes, the parity checks deliver extrinsic messages to the variable nodes. Then, these messages are processed by the variable nodes and passed to the summation checks. This ends one cycle of message passing. The purpose of an EXIT chart analysis is to verify the fitness between iterative decoding modules. In the current scenario, the iterative decoder consists of three interactive modules. As a result, the task of the EXIT chart analysis becomes two-fold. It needs to verify the fitness between the SC ensemble and the VN-plus-PC ensemble, and it needs to verify the fitness between the PC ensemble and the VN-plus-SC ensemble. The extrinsic-information-transfer function of summation checks can easily be measured via numerical simulations. For parity checks, the extrinsic-information-transfer function can be obtained either numerically or analytically [105, 106]. However, it is not straightforward to obtain the extrinsic-information-transfer function for the VN-plus-PC ensemble and the VN-plus-SC ensemble. As stated previously, we should not do this work via simulations. Otherwise, the resulting EXIT chart analysis will again be dependent on the interleaver. To solve the problem, we propose a semi-analytical emulation technique.

Revisiting Fig. 7.26, one may recognize that the ensemble of variable nodes actually acts as an amplify-and-relay unit for the messages between the ensemble of summation checks and the ensemble of parity checks. In case that we are able to characterize the extrinsic-information-transfer function for the VN ensemble, the iterative decoding process can in fact be emulated via a computer-based analytical derivation procedure. To enable such an approach, a Gaussian approximation is necessary for the distribution of LLR messages. Given an AWGN channel with a BPSK input:

$$y = x + z \,, \quad x \in \{\pm 1\} \,, z \sim \mathcal{N}(0, \sigma_z^2) \,, \tag{7.38}$$

we have the LLR of the symbol x as

$$LLR(x) \doteq \ln \frac{p(y|x=+1)}{p(y|x=-1)} = \ln \frac{e^{-\frac{(y-1)^2}{2\sigma_z^2}}}{e^{-\frac{(y+1)^2}{2\sigma_z^2}}} = \frac{2}{\sigma_z^2} y = \frac{2}{\sigma_z^2} x + \frac{2}{\sigma_z^2} z \,. \tag{7.39}$$

Hence, the LLR message itself is a Gaussian variable with mean $\mu = 2x/\sigma_z^2$ and variance $\sigma^2 = (2/\sigma_z^2)^2 \sigma_z^2 = 4/\sigma_z^2$. Note that $\sigma^2 = 2|\mu|$. Since $|x| \equiv 1$, the distribution of the LLR message can be written as

$$LLR(x) \;\sim\; \mathcal{N}(\mathrm{sign}\,(x) \cdot \sigma^2/2, \sigma^2)\ . \tag{7.40}$$

A distribution as in (7.40) is called a consistent Gaussian distribution. Now, the essential assumption for the EXIT emulation method is that all messages passing in the graph have a consistent Gaussian distribution[2]. Given this assumption, a one-to-one correspondence exists between the metric σ and the mutual information $I(x, LLR(x))$, which is well-known as the J function for EXIT chart analysis. For a shorthand notation, we define $\Lambda \doteq LLR(x)$. The J function is obtained as

$$\begin{aligned}
J(\sigma) \doteq I(x;\Lambda) &= H(x) - H(x|\Lambda) = 1 - \mathrm{E}\left\{\log_2 \frac{1}{P(x|\Lambda)}\right\} \\
&= 1 - \sum_{x=\pm1} \int_{-\infty}^{+\infty} p(x,\Lambda)\ \log_2 \frac{1}{P(x|\Lambda)}\ \mathrm{d}\Lambda \\
&= 1 - \sum_{x=\pm1} \int_{-\infty}^{+\infty} P(x)p(\Lambda|x)\ \log_2 \frac{p(\Lambda)}{p(x,\Lambda)}\ \mathrm{d}\Lambda \\
&= 1 - \sum_{x=\pm1} \frac{1}{2} \int_{-\infty}^{+\infty} p(\Lambda|x)\ \log_2 \frac{\sum_x p(\Lambda|x)P(x)}{p(\Lambda|x)P(x)}\ \mathrm{d}\Lambda \\
&= 1 - \sum_{x=\pm1} \frac{1}{2} \int_{-\infty}^{+\infty} p(\Lambda|x)\ \log_2 \frac{p(\Lambda|x=+1)+p(\Lambda|x=-1)}{p(\Lambda|x)}\ \mathrm{d}\Lambda \\
&= 1 - \int_{-\infty}^{+\infty} p(\Lambda|x=+1)\ \log_2 \left(1 + \frac{p(\Lambda|x=-1)}{p(\Lambda|x=+1)}\right)\ \mathrm{d}\Lambda \\
&= 1 - \int_{-\infty}^{+\infty} \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(\Lambda-\sigma^2/2)^2}{2\sigma^2}}\ \log_2(1+e^{-\Lambda})\ \mathrm{d}\Lambda\ ,
\end{aligned} \tag{7.41}$$

where the last equality utilizes (7.40) in the form:

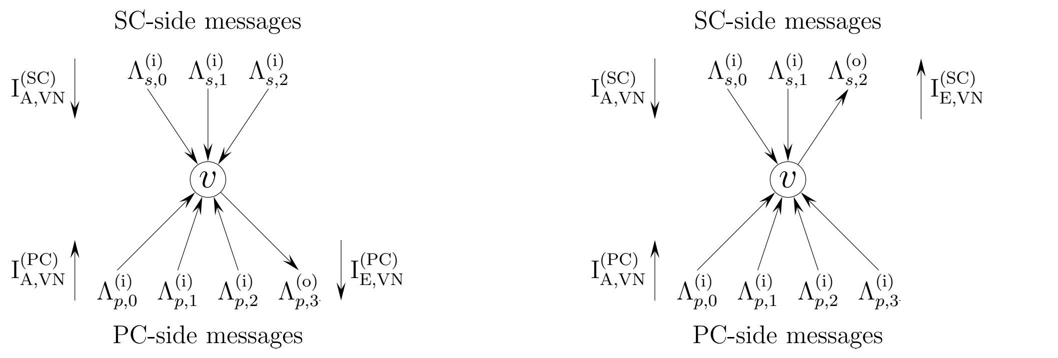
$$p(\Lambda|x=\pm1) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(\Lambda\mp\sigma^2/2)^2}{2\sigma^2}}\ . \tag{7.42}$$

In practice, we can make an extensive pre-calculation for (7.41) and save it into a look-up table. Afterwards, a fast J function can be implemented by using the look-up table. In case that a high-precision is required, one may apply a linear or polynomial interpolation. Now, from $J(\cdot)$, we can numerically derive its inverse function

$$\sigma = J^{-1}(I(x;\Lambda))\ . \tag{7.43}$$

Similarly, we can implement a fast version for this function via a large look-up table and an appropriate interpolator.

[2]In fact, this assumption is commonly applied for the conventional EXIT analysis as well, whenever an EXIT curve needs to be obtained via numerical simulations.


SC-side messages
$I^{(SC)}_{A,VN}$
$\Lambda^{(i)}_{s,0}$ $\Lambda^{(i)}_{s,1}$ $\Lambda^{(i)}_{s,2}$
v
$I^{(PC)}_{A,VN}$
$\Lambda^{(i)}_{p,0}$ $\Lambda^{(i)}_{p,1}$ $\Lambda^{(i)}_{p,2}$ $\Lambda^{(o)}_{p,3}$
$I^{(PC)}_{E,VN}$
PC-side messages
SC-side messages
$I^{(SC)}_{A,VN}$
$\Lambda^{(i)}_{s,0}$ $\Lambda^{(i)}_{s,1}$ $\Lambda^{(o)}_{s,2}$
$I^{(SC)}_{E,VN}$
v
$I^{(PC)}_{A,VN}$
$\Lambda^{(i)}_{p,0}$ $\Lambda^{(i)}_{p,1}$ $\Lambda^{(i)}_{p,2}$ $\Lambda^{(i)}_{p,3}$
PC-side messages

(a) Create a message to a parity check. (b) Create a message to a summation check.

Figure 7.27: Extrinsic information transfer processes at a variable node.

With the consistent-Gaussian assumption as well as the J and J^{-1} functions, we are now ready to emulate the extrinsic information transfer process for a VN ensemble. Consider a variable node has an SC-side repetition degree d_s and a PC-side repetition degree d_p. In each iteration, this variable node receives d_s messages from the summation checks and d_p messages from the parity checks. The generation of an extrinsic message at a variable node is simply a linear addition. The extrinsic message to a parity check associated with the ith edge should be created as

$$\Lambda^{(o)}_{p,i} = \sum_{j=0}^{d_s-1} \Lambda^{(i)}_{s,j} + \sum_{j=0,j\neq i}^{d_p-1} \Lambda^{(i)}_{p,j} \ , \quad 0 \leqslant i < d_p \ , \tag{7.44}$$

cf. Fig. 7.27(a). The extrinsic message to a summation check associated with the ith edge should be created as

$$\Lambda^{(o)}_{s,i} = \sum_{j=0,j\neq i}^{d_s-1} \Lambda^{(i)}_{s,j} + \sum_{j=0}^{d_p-1} \Lambda^{(i)}_{p,j} \ , \quad 0 \leqslant i < d_s \ , \tag{7.45}$$

cf. Fig. 7.27(b). Staying with the notations in Fig. 7.26, we let $I^{(SC)}_{A,VN}$ denote the mutual information of LLR messages from the summation checks to the variable node, etc.. We let

$$\sigma_{s,\mathrm{in}} \doteq J^{-1}\left(I^{(SC)}_{A,VN}\right) \tag{7.46}$$

represent the distribution metric of the corresponding messages. Similarly, we define

$$\sigma_{p,\mathrm{in}} \doteq J^{-1}\left(I^{(PC)}_{A,VN}\right) \ . \tag{7.47}$$

By the consistent-Gaussian assumption and assuming that the code bit has a value 0 (i.e., $x = +1$ for BPSK), the extrinsic messages to the parity checks will have a distribution as

$$\Lambda^{(o)}_{p,i} \quad \sim \quad \mathcal{N}\left(d_s\frac{\sigma^2_{s,\mathrm{in}}}{2} + (d_p-1)\frac{\sigma^2_{p,\mathrm{in}}}{2} \ , d_s\sigma^2_{s,\mathrm{in}} + (d_p-1)\sigma^2_{p,\mathrm{in}}\right) \ , \quad 0 \leqslant i < d_p \ , \tag{7.48}$$

which leads to a distribution metric as

$$\sigma_{p,\text{out}} = \sqrt{d_s\sigma_{s,\text{in}}^2 + (d_p - 1)\sigma_{p,\text{in}}^2}\ . \tag{7.49}$$

Consequently, the mutual information of the LLR messages from the variable node to the parity checks is given by

$$\begin{aligned} \mathrm{I}_{\mathrm{E,VN}}^{(\mathrm{PC})} &= J(\sigma_{p,\text{out}}) = J\left(\sqrt{d_s\sigma_{s,\text{in}}^2 + (d_p - 1)\sigma_{p,\text{in}}^2}\right) \\ &= J\left(\sqrt{d_s\left[J^{-1}\left(\mathrm{I}_{\mathrm{A,VN}}^{(\mathrm{SC})}\right)\right]^2 + (d_p - 1)\left[J^{-1}\left(\mathrm{I}_{\mathrm{A,VN}}^{(\mathrm{PC})}\right)\right]^2}\right)\ . \end{aligned} \tag{7.50}$$

Following the same treatment for the messages to the summation checks, we obtain

$$\begin{aligned} \mathrm{I}_{\mathrm{E,VN}}^{(\mathrm{SC})} &= J(\sigma_{s,\text{out}}) = J\left(\sqrt{(d_s - 1)\sigma_{s,\text{in}}^2 + d_p\sigma_{p,\text{in}}^2}\right) \\ &= J\left(\sqrt{(d_s - 1)\left[J^{-1}\left(\mathrm{I}_{\mathrm{A,VN}}^{(\mathrm{SC})}\right)\right]^2 + d_p\left[J^{-1}\left(\mathrm{I}_{\mathrm{A,VN}}^{(\mathrm{PC})}\right)\right]^2}\right)\ . \end{aligned} \tag{7.51}$$

(7.50) and (7.51) fully describe the behaviour of a variable node for message passing in an iterative LDHC decoding process. The important feature of these two computations is that they are independent of the interleaver. Nevertheless, as (7.50) and (7.51) are only for a single variable node, an averaging operation is necessary to describe the behaviour of an irregular VN ensemble. Certainly, the mutual information should be weighted by the respective VN degree during the averaging operation. Given a fixed degree distribution, the EXIT functions of a VN ensemble are fixed, which can be written as

$$\mathrm{I}_{\mathrm{A,PC}} \ \widehat{=}\ \mathrm{I}_{\mathrm{E,VN}}^{(\mathrm{PC})} = f_{v,p}\left(\mathrm{I}_{\mathrm{A,VN}}^{(\mathrm{SC})}, \mathrm{I}_{\mathrm{A,VN}}^{(\mathrm{PC})}\right) = f_{v,p}\left(\mathrm{I}_{\mathrm{E,SC}}, \mathrm{I}_{\mathrm{E,PC}}\right) \tag{7.52}$$

$$\mathrm{I}_{\mathrm{A,SC}} \ \widehat{=}\ \mathrm{I}_{\mathrm{E,VN}}^{(\mathrm{SC})} = f_{v,s}\left(\mathrm{I}_{\mathrm{A,VN}}^{(\mathrm{SC})}, \mathrm{I}_{\mathrm{A,VN}}^{(\mathrm{PC})}\right) = f_{v,s}\left(\mathrm{I}_{\mathrm{E,SC}}, \mathrm{I}_{\mathrm{E,PC}}\right)\ . \tag{7.53}$$

To characterize the behaviour of an SC ensemble, we can use a look-up table containing pre-measured results obtained from numerical simulations. For an easy reference, let us write the EXIT function of an SC ensemble as

$$\mathrm{I}_{\mathrm{E,SC}} = f_s\left(\mathrm{I}_{\mathrm{A,SC}}, \mathrm{SNR}\right)\ . \tag{7.54}$$

The EXIT function of a PC ensemble is independent of the SNR. We may apply a look-up table as well or simply utilize the duality between the EXIT function of parity checks and that of variable nodes. According to the duality, for a degree d_c parity check the relationship between the input and output mutual information is given by

$$\mathrm{I}_{\mathrm{E}} = 1 - J\left(\sqrt{(d_c - 1)\left[J^{-1}(1 - \mathrm{I}_{\mathrm{A}})\right]^2}\right)\ , \tag{7.55}$$

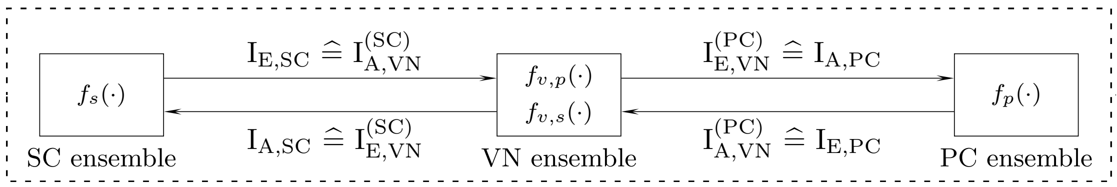

$f_s(\cdot)$
SC ensemble
$\mathrm{I}_{\mathrm{E,SC}} \mathrel{\widehat{=}} \mathrm{I}^{(\mathrm{SC})}_{\mathrm{A,VN}}$
$\mathrm{I}_{\mathrm{A,SC}} \mathrel{\widehat{=}} \mathrm{I}^{(\mathrm{SC})}_{\mathrm{E,VN}}$
$f_{v,p}(\cdot)$
$f_{v,s}(\cdot)$
VN ensemble
$\mathrm{I}^{(\mathrm{PC})}_{\mathrm{E,VN}} \mathrel{\widehat{=}} \mathrm{I}_{\mathrm{A,PC}}$
$\mathrm{I}^{(\mathrm{PC})}_{\mathrm{A,VN}} \mathrel{\widehat{=}} \mathrm{I}_{\mathrm{E,PC}}$
$f_p(\cdot)$
PC ensemble

Figure 7.28: A semi-analytical EXIT emulator for iterative LDHC decoding.

which is exact for the binary erasure channel [90] and very accurate for the binary-input AWGN channel [105, 106]. Given a certain degree distribution, the EXIT function of a PC ensemble can be obtained via a weighted averaging process on the EXIT functions of single parity checks. Let us write the EXIT function of a PC ensemble as

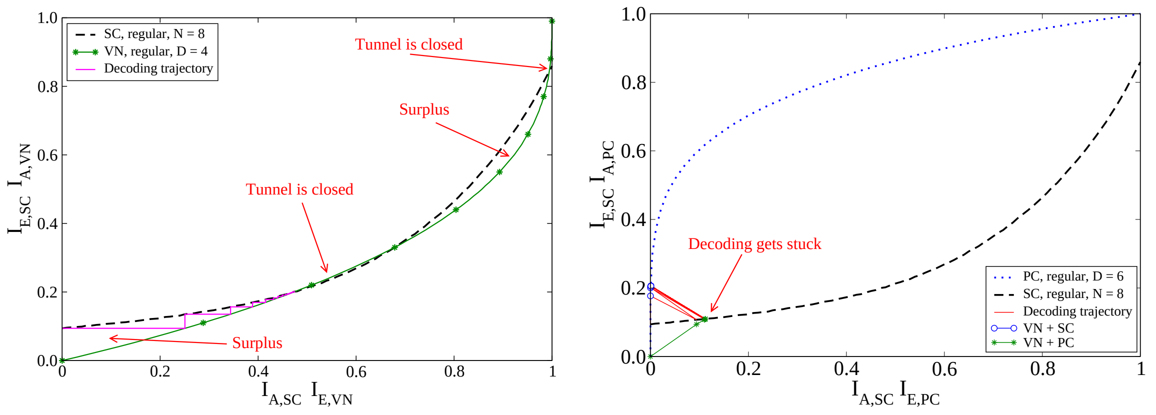
$$\mathrm{I}_{\mathrm{E,PC}} = f_p\left(\mathrm{I}_{\mathrm{A,PC}}\right) \ . \tag{7.56}$$

Given (7.52), (7.53), (7.54), and (7.56), we can emulate an iterative LDHC decoding process without the need of any numerical simulations. As the first step, the emulator is initialized with the following starting values:

$$\mathrm{I}_{\mathrm{A,SC}} = 0 \ , \quad \mathrm{I}_{\mathrm{E,SC}} = f_s\left(0, \mathrm{SNR}\right) \ , \quad \mathrm{I}_{\mathrm{A,PC}} = 0 \ , \quad \mathrm{I}_{\mathrm{E,PC}} = 0 \ . \tag{7.57}$$

Following that, the emulator performs iterations as illustrated in Fig. 7.28. Each concrete node ensemble is replaced by its EXIT function, which is tailored for the respective degree distribution. Local iterations between the SC ensemble and the VN ensemble are optional. So are the local iterations between the VN ensemble and the PC ensemble. By recording the value pair $(\mathrm{I}^{(\mathrm{PC})}_{\mathrm{A,VN}}, \mathrm{I}^{(\mathrm{PC})}_{\mathrm{E,VN}})$ throughout the emulation process, we obtain an interleaver-independent EXIT curve for the VN-plus-SC ensemble. Accordingly, by recording the value pair $(\mathrm{I}^{(\mathrm{SC})}_{\mathrm{A,VN}}, \mathrm{I}^{(\mathrm{SC})}_{\mathrm{E,VN}})$ throughout the emulation process, we obtain an interleaver-independent EXIT curve for the VN-plus-PC ensemble. Moreover, an overall decoding trajectory can be obtained by tracking the value pair $(\mathrm{I}_{\mathrm{A,SC}}, \mathrm{I}_{\mathrm{E,SC}})$ and the value pair $(\mathrm{I}_{\mathrm{A,PC}}, \mathrm{I}_{\mathrm{E,PC}})$ throughout the emulation process. The emulation process stops when all the mutual information values have reached 1 or it stops when all the mutual information values stay unchanged from the last iteration.

There are many advantages to apply such an EXIT emulator instead of a simulation-based EXIT chart analysis. The first advantage is that it truly gives a performance prediction for an infinite block length and a cycle-free graph. Second, it is much more efficient than a conventional EXIT chart analysis. In practice, one basically needs to perform a numerical simulation only for the SC-ensemble, but only once for one SNR value. Afterwards, no more simulation is needed at all, no matter how one adjusts the degree distribution of the variable nodes or the check nodes. This greatly speeds up the code design process. Third, the resulting EXIT chart provides a much better visual aid for the code design. These advantages will soon become manifest via various code design examples.



(a) LDSC-EPA, $SF = 4$. (b) LDHC-EPA, $R_r = 1/2$, $R_p = 1/2$.

Figure 7.29: LDSC-EPA vs. LDHC-EPA, $R = 1/4$, $N = 8$, $E_b/N_0 = 8$ dB.

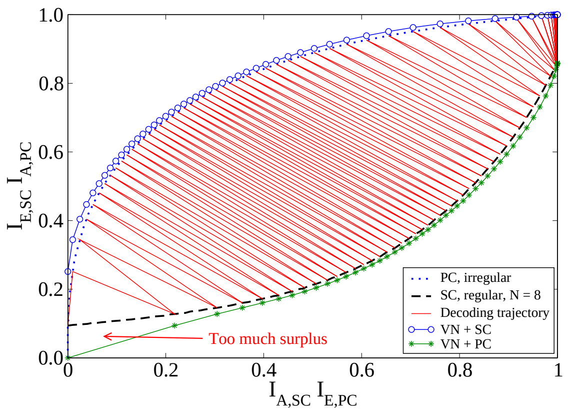
A Code Design Example for SM-EPA with $N = 8$

To demonstrate the usage of the previously proposed EXIT emulation technique, we consider a code design example for SM-EPA with $N = 8$, which has been briefly discussed in Section 7.3.2. We target at a bandwidth efficiency of 2 bits/symbol, for which the Shannon limit is at about 5.8 dB. For a unified notation, we use R_r to denote the coding rate of the component repetition code of an LDHC code, and R_p that of the component parity-check code. Consequently, the overall coding rate of an LDHC code is given by $R = R_r \cdot R_p$. For the current discussion, we require $R = 1/4$.

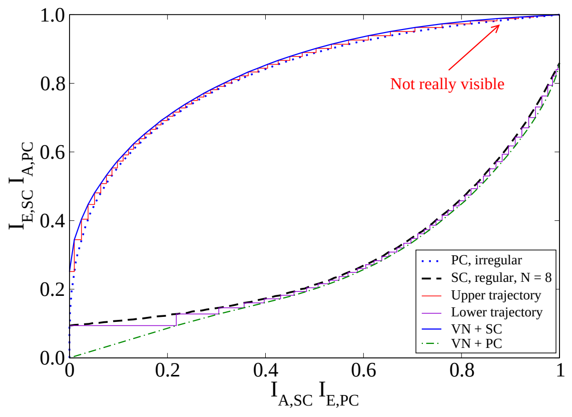
To leave some room for easy elaboration, we first target at a decoding threshold of 8 dB. Let us start with a regular LDSC-EPA code for $SF = 1/R = 4$. Fig. 7.29(a) provides the corresponding EXIT chart. Since the graph of an LDSC code merely consists of two types of nodes, a conventional two-part EXIT chart suffices. One observes that the convergence tunnel is closed in the middle region and the rightmost region, but has surplus in the left region and the right region. According to the area property of EXIT charts [90–92], the area above a decoder EXIT curve is given by $\mathcal{A} = 1 - R$ when plotted on swapped axes, regardless of the code type. This means that the area above the EXIT curve is fixed given a certain coding rate. Hence, in order to open the convergence tunnel, we need to transfer the surplus in the left region and the right region to the middle region and the rightmost region. One should not try to use an irregular LDSC code to solve this problem, as this will open the tunnel in the middle region but make the tunnel gets closed even earlier in the rightmost region. The discussion in Section 7.2.2 gives a firm support for this statement. To open the tunnel in the rightmost region, some parity bits are necessary to be added. By serially concatenating a 1/2 regular LDPC code with a 1/2 repetition code, the EXIT emulator gives an EXIT chart as in Fig. 7.29(b), which shows that the decoding process gets stuck in an even earlier stage w.r.t. the LDSC case. The important message here is the effectiveness of the EXIT emulator. Comparing Fig. 7.29(b) with Fig. 7.17(a), one finds that the VN-plus-PC curve well predicts the EXIT function of the concatenated decoder. Certainly, it is more convenient to obtain this EXIT curve via the emulation approach w.r.t. the simulation approach. As soon as the convergence tunnel gets closed in a certain position, the decoding trajectory gets stopped therein. Note that in a conventional two-part EXIT chart, the decoding trajectory is bouncing between the EXIT curve of a check node ensemble and that of a variable node ensemble, cf. Fig. 7.29(a). In a three-part EXIT chart, the most preferable choice is to plot the decoding trajectory between the summation check ensemble and the parity check ensemble, cf. Fig. 7.29(b), as doing so brings the best visual aid for code design. The following elaboration will make this reason evident. We rewrite the code design corresponding to Fig. 7.17(b) in new notations:

$$
\begin{aligned}
\lambda_s(D) &= 0.700D^1 + 0.080D^3 + 0.120D^4 + 0.020D^5 + 0.080D^6 \\
\lambda_p(D) &= 0.900D^2 + 0.010D^6 + 0.040D^9 + 0.010D^{12} + 0.020D^{15} + 0.020D^{18} \qquad (7.58) \\
\eta_p(D) &= 0.610D^4 + 0.200D^5 + 0.040D^8 + 0.020D^{10} + 0.080D^{11} + 0.030D^{22} + 0.020D^{25} ,
\end{aligned}
$$

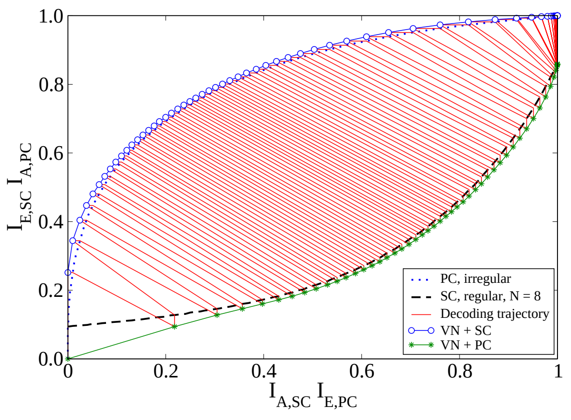
where $\lambda_s(D)$ stands for the SC-side VN degree distribution, $\lambda_p(D)$ the PC-side VN degree distribution, and $\eta_p(D)$ the parity check degree distribution. The average parity check degree is 6 and the component coding rates are $R_r = 1/2$ and $R_p = 1/2$. Fig. 7.17(b) shows that such a code design opens the tunnel given 20 LDPC-local iterations. This is also the case by means of EXIT emulation, as shown in Fig. 7.30(a), though no LDPC-local iterations are performed. It can be seen that the single decoding trajectory vividly demonstrates the iterative decoding process. As a matter of fact, the density of the decoding trajectory in a certain area gives a good judgement for the fitness of the code design in the respective decoding stage. Whenever the convergence tunnel is tight in a certain region, it will take the emulator many iterations to pass this region, and consequently make the decoding trajectory very dense in this region. Vice versa, the lower the density is, the easier it is for the decoder to travel through that area. Nonetheless, for an iterative LDHC decoding process, there are in fact two convergence tunnels, one between the EXIT curve of the VN-plus-SC ensemble and that of the PC ensemble, one between the EXIT curve of the SC ensemble and that of the VN-plus-PC ensemble. To achieve a decoding convergence, both tunnels must be open for the whole region. Specifically, this is to ensure that the VN-plus-SC curve is above the PC curve and the SC curve is above the VN-plus-PC curve. If desired, we may let the emulator to record the decoding trajectories inside these two tunnels, as demonstrated in Fig. 7.30(b). However, the visibility is much worse than the case of plotting a single decoding trajectory. When the tunnel becomes rather narrow, the decoding trajectory is difficult to distinguish. Alternatively, we may also let the emulator to record the trajectory for the complete decoding process, which



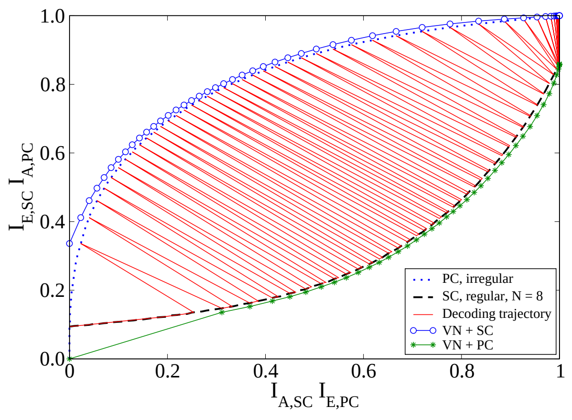
(a) With a single decoding trajectory.



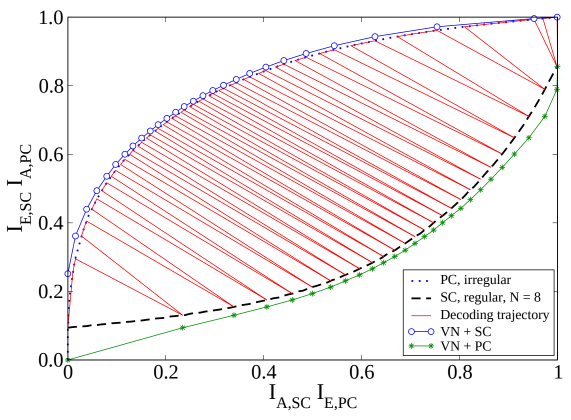
(b) With double decoding trajectories.



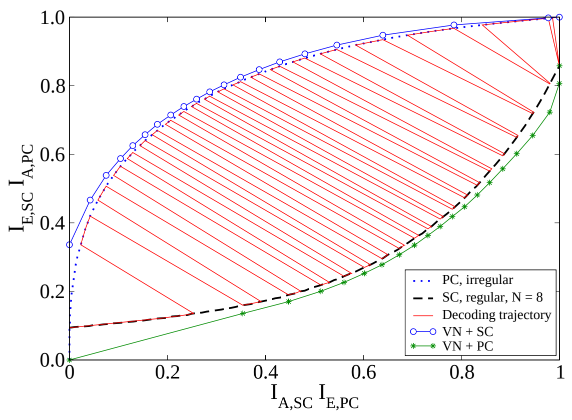
(c) With a complete decoding trajectory.



(d) With 10 LDSC-local iterations.



(e) With 10 LDPC-local iterations.



(f) With 10 LDSC/LDPC-local iterations.

Figure 7.30: LDHC-EPA, $R_r = 1/2$, $R_p = 1/2$, $N = 8$, $E_b/N_0 = 8$ dB.

yields an EXIT chart as in Fig. 7.30(c). This visualizes the trace of the decoding process for every SC-to-VN-to-PC-to-VN-to-SC message passing cycle. Nevertheless, compared to Fig. 7.30(a), doing so does not improve the visibility of the EXIT chart but causes some unnecessary confusions. Therefore, in the rest of the thesis, we will exclusively adopt a single decoding trajectory that bouncing inbetween the EXIT curves of two check node ensembles. The proposed EXIT emulation technique is also useful in checking the necessity as well as the effectiveness of making LDSC-local iterations or LDPC-local iterations. For example, if one makes 10 LDSC-local iterations in each global iteration, the resulting EXIT chart will be as shown in Fig. 7.30(d). It can be seen that, before jumping to the PC curve, the decoding trajectory travels for a certain distance along the SC curve, in each global iteration. This distance decreases as the overall decoding process proceeds. In the late stage of iterative decoding, the benefit of LDSC-local iterations almost diminishes. Hence, it only makes sense to apply LDSC-local iterations in the early decoding stage. In the late decoding stage, the main task of the decoder is to offer some coding gain, which is relatively irrelevant to the repetition decoder. If one makes 10 LDPC-local iterations in each global iteration, the resulting EXIT chart will be as shown in Fig. 7.30(e). Now, the decoding trajectory travels along the PC curve for a certain distance, before it jumps to the SC curve. One observes that the effectiveness of LDPC-local iterations is most evident in the early stage as well as the late stage of iterative decoding. In the early stage, by means of LDPC-local iterations, those variable nodes with a high PC-side repetition degree can deliver a strong information to the rest of the graph and consequently help the overall decoding process. In the late stage, those variable nodes with a low PC-side repetition degree start to deliver strong messages given a certain amount of LDPC-local iterations. Certainly, we may apply LDSC-local iterations together with LDPC-local iterations, which leads to an EXIT chart as in Fig. 7.30(f). Comparing Fig. 7.30(f) with Fig. 7.30(a), one observes that the number of global iterations is reduced by means of LDSC/LDPC-local iterations. Nevertheless, this does not necessarily reduce the overall decoding complexity, and often the situation is the opposite if the local iterations are not well scheduled. From a theoretical point of view, local iterations can in no chance improve the ultimately achievable performance. For the sake of compactness, we exclude the topic of local iterations in the rest of the thesis.

In the above discussion, a sorted VN degree combination is assumed. That is, we combine the SC-side repetition degrees and the PC-side repetition degrees exactly as indicated by (7.58). A variable node with the lowest SC-side repetition degree is assigned with the lowest PC-side repetition degree, and a variable node with the highest SC-side repetition degree is assigned with the highest PC-side repetition degree. This can easily be achieved by sorting the degree alignment of variable nodes in a graph. For example, one sorts the SC-side repetition degrees into a non-descending order and so for the PC-side repetition

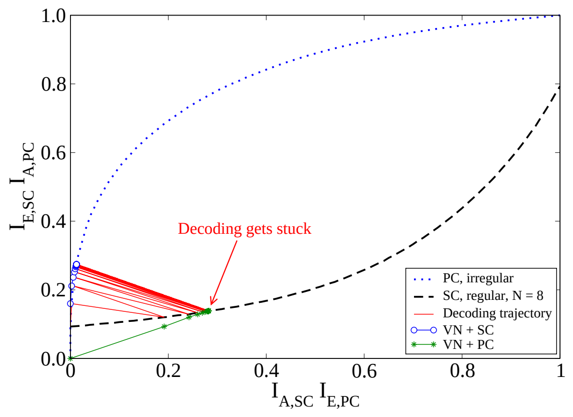


Figure 7.31: $R_r = 1/2$, $R_p = 1/2$, $N = 8$, $E_b/N_0 = 8$ dB, unsorted degree combination.

degrees. Although there are uncountable possibilities for doing the degree combination, such a sorted scheme is in fact the most efficient one in most cases. The underlying idea is that to let high-degree variable nodes open the tunnel in the left region and low-degree variable nodes offer a moderate coding gain. One may wonder what happens if we perform an unsorted degree combination. Given the same degree distribution as in (7.58), an unsorted (randomized) degree combination gives an EXIT chart as in Fig. 7.31. Clearly, the code design is no longer valid in this case, as a decoding convergence is no longer achievable. In some special situations, it makes sense to apply a degree combination that is sorted not exactly in a non-descending order. However, for the sake of clearness, we will exclusively assume a sorted degree combination that always combine high SC-side repetition degrees with high PC-side repetition degrees.

Since the Shannon limit for 2 bits/symbol is at about 5.8 dB, the code design given in (7.58) is still not optimal, at least theoretically. Revisiting Fig. 7.30(a), we find that there is too much surplus between the SC curve and the VN-plus-PC curve in the leftmost region. This is a typical problem for repetition coding, as it tends to be over-qualified in the early stage of iterative decoding. Nevertheless, as the area above the decoder curve is constant given a fixed coding rate, an unnecessary surplus in one region must leads to a tight or closed tunnel in some other region. To reduce the surplus in the leftmost region, a straightforward solution is to reduce the amount of SC-side repetitions and meanwhile increase the amount of PC-side repetitions. Let us target at a decoding threshold of $E_b/N_0 = 7$ dB and consider the following degree distributions:

$$\begin{aligned}\lambda_s(D) &= 0.866D^1 + 0.054D^2 + 0.020D^3 + 0.020D^4 + 0.020D^5 + 0.020D^6 \\ \lambda_p(D) &= 0.840D^2 + 0.020D^5 + 0.020D^6 + 0.020D^7 + 0.020D^9 + 0.020D^{10} \\ &\quad +0.020D^{14} + 0.020D^{20} + 0.020D^{45} \\ \eta_p(D) &= 0.100D^4 + 0.500D^5 + 0.360D^6 + 0.020D^{17} + 0.020D^{30} \qquad (7.59)\end{aligned}$$

with $R_r = 3/4$ and $R_p = 1/3$. Fig. 7.32(a) gives the corresponding EXIT chart, which shows that this is a valid code design for a decoding threshold of 7 dB. Compared to Fig. 7.30(a), the decoding trajectory becomes more dense, which means that it gets more difficult for the decoder to converge with a finite block length. Nevertheless, there is still some surplus in the leftmost region. Hence, as long as the SC curve starts from a position with a non-trivial height, a repetition code introduces a certain extent of suboptimality. Certainly, for SM-EPA with a very large N, this suboptimality becomes negligible. At this point, a natural question would be what will be the achievable decoding threshold if we increase the repetition coding rate further. The following degree distributions:

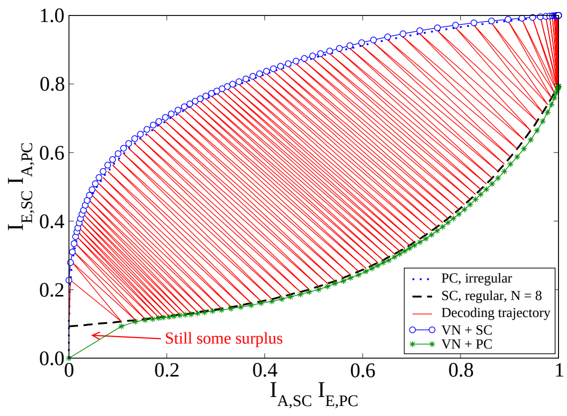
$$\begin{aligned}
\lambda_s(D) &= 0.900D^1 + 0.100D^2 \\
\lambda_p(D) &= 0.590D^2 + 0.110D^3 + 0.130D^4 + 0.050D^5 + 0.020D^7 \\
&\quad +0.050D^{11} + 0.010D^{12} + 0.020D^{16} + 0.010D^{21} + 0.010D^{73} \\
\eta_p(D) &= 0.150D^3 + 0.330D^4 + 0.400D^5 + 0.020D^8 + 0.040D^{14} \\
&\quad +0.040D^{15} + 0.010D^{40} + 0.010D^{51} \qquad (7.60)
\end{aligned}$$

with $R_r = 0.909$ and $R_p = 0.275$ achieves a decoding threshold of 6 dB, which is only 0.2 dB away from the Shannon limit. The resulting EXIT chart is given in Fig. 7.32(b). It is reasonable that the decoding trajectory becomes extremely dense, as the channel capacity is only achievable given an infinite block length and an infinite number of iterations. Typically, for a rate below the capacity, the valid code designs will not be unique. By setting the SC-side repetition degrees to be all 1's, we find another code design that achieves a decoding threshold of 6 dB as well. Fig. 7.32(c) gives the EXIT chart for the degree distributions:

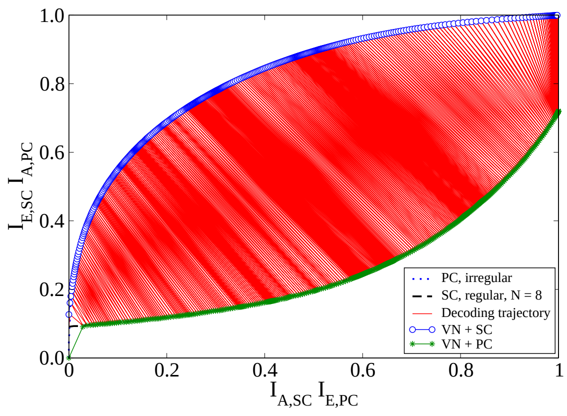
$$\begin{aligned}
\lambda_s(D) &= 1.000D^1 \\
\lambda_p(D) &= 0.600D^2 + 0.080D^3 + 0.130D^4 + 0.080D^5 + 0.010D^6 \\
&\quad +0.050D^{11} + 0.010D^{12} + 0.020D^{16} + 0.010D^{17} + 0.010D^{92} \\
\eta_p(D) &= 0.160D^3 + 0.320D^4 + 0.400D^5 + 0.020D^8 + 0.040D^{14} \\
&\quad +0.040D^{15} + 0.010D^{40} + 0.010D^{52} \qquad (7.61)
\end{aligned}$$

with $R_r = 1$ and $R_p = 1/4$. Checking Fig. 7.32(c) carefully, one finds that for some regions the decoding trajectory is still not extremely dense. This marginal remaining surplus is from the 0.2 dB distance to the Shannon limit.

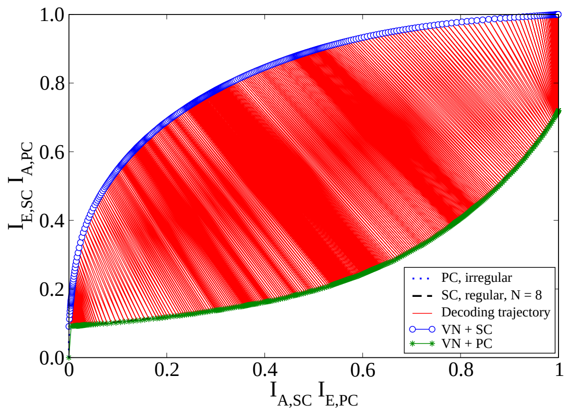
According to the above code designs, superposition mapping with equal power allocation is indeed capacity-achieving. Nevertheless, given the code designs corresponding to Fig. 7.32(b) and Fig. 7.32(c), the promised decoding threshold at 6 dB is only achievable given an infinite block length and a cycle-free graph. We will provide BER performance analysis for these code designs in Section 7.4.4, in conjunction with interleaver design.



(a) $R_r = 3/4$, $R_p = 1/3$, $E_b/N_0 = 7$ dB.

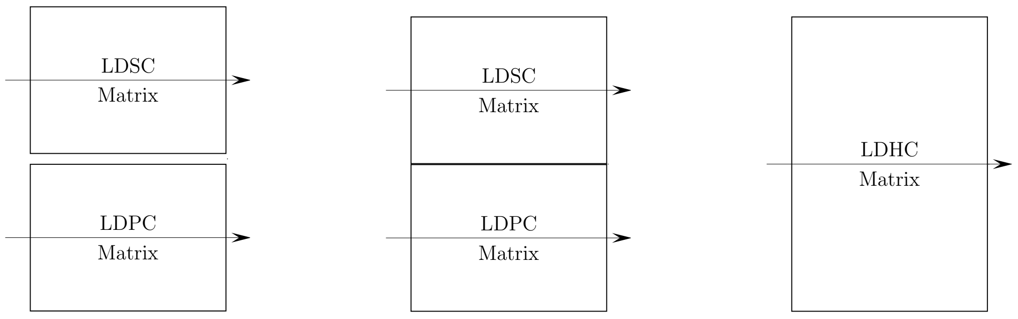


(b) $R_r = 0.909$, $R_p = 0.275$, $E_b/N_0 = 6$ dB.



(c) $R_r = 1$, $R_p = 1/4$, $E_b/N_0 = 6$ dB.

Figure 7.32: LDHC-EPA, $R = 1/4$, $N = 8$, sorted degree combination.

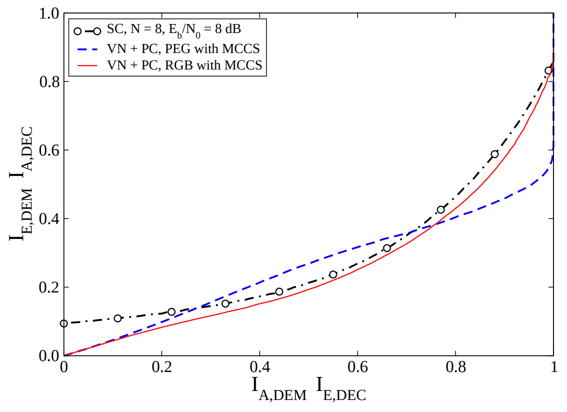


(a) Separate construction. (b) Block-wise construction. (c) All-in-one construction.

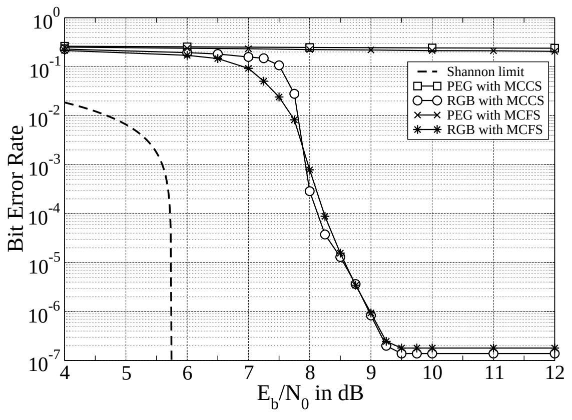
Figure 7.33: Possible ways for constructing an LDHC matrix.

7.4.4 Possible Ways for Interleaver Design

Given an infinite block length and a cycle-free graph, the achievable performance is determined by the degree distribution of variable nodes. On the other hand, the practically achievable performance is significantly influenced by the interleaver pattern, given a finite block length and a non-cycle-free graph. Hence, after one obtains an optimized degree distribution for LDHC coding, the second step for code optimization is to design an interleaver pattern that can approach as closely as possible to the performance promised by the degree distribution. We have provided in Section 6.4.2 an extensive discussion on the topic of interleaver design for LDSC code optimization. The interleaver design methods introduced therein are easily applicable for LDHC codes as well. Nevertheless, since an LDHC matrix is basically a vertical concatenation of an LDSC matrix and an LDPC matrix, there are in fact more options available for the matrix construction. As shown in Fig. 7.33, there are generally three possible approaches to construct the incidence matrix of an LDHC code. The first approach is to construct the constituent LDSC matrix and the constituent LDPC matrix separately, cf. Fig. 7.33(a). The second approach is to construct the LDHC matrix block-wise, cf. Fig. 7.33(b). The third approach is to construct the LDHC matrix all-in-once, cf. Fig. 7.33(c). By the first approach, hybrid-type of cycles between summation checks and parity checks are completely ignored. Though this approach is by no means optimal, it is in fact the common practice in state-of-the-art systems applying a serial concatenation of repetition code and parity-check code. By the second approach, hybrid-type of cycles can be taken into account in the interleaver design. Besides, one gets an option for designing which sub-matrix first. Certainly, the sub-matrix that gets constructed first will have a better graph quality than the sub-matrix that gets constructed second, as the freedom in a graph steadily drops with the construction process. Moreover, one may apply different methods for the two sub-matrices, e.g., the PEG algorithm for the LDSC matrix and the RGB algorithm for the LDPC matrix. By the third approach, both constituent matrices are constructed simultaneously and jointly.



(a) EXIT chart, 20 LDPC-local iterations.



(b) BER vs. E_b/N_0.

Figure 7.34: LDHC-EPA, $R_r = 1/2$, $R_p = 1/2$, $K = 40000$, VBS, 100 global iterations.

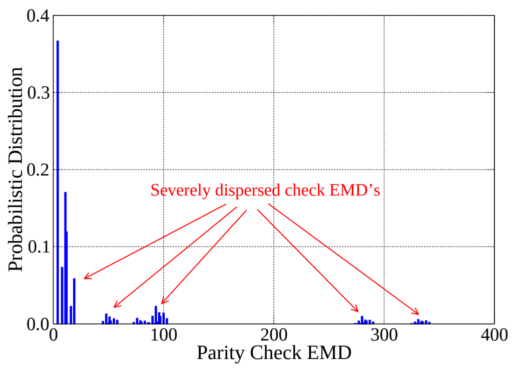
Challenges from Irregular Parity Check Degree Distributions

To check the influence of interleaving on LDHC decoding, we consider the code design in (7.58) as an illustrative example. For the sake of clearness, let us assume separate matrix construction and fix the interleaver design method for the LDSC sub-matrix as the PEG algorithm. Recall that in Fig. 7.17(b) a conventional EXIT chart has been provided for the same code design. For obtaining this EXIT chart, we have set a testing interface in between the SC ensemble and the VN ensemble. Correspondingly, the resulting EXIT chart is independent of the SC-side interleaver but dependent of the PC-side interleaver. Although such an EXIT chart analysis is inefficient for designing capacity-achieving LDHC codes, it is useful for checking the influence of interleaving on the practically achievable performance. Well-known in the community of LDPC coding, a narrow and concentrated parity check degree distribution often provides the best performance for a moderate block length. On the other hand, the code design in (7.58) adopts a wide and dispersed parity check degree distribution. Revisiting (7.59), (7.60), and (7.61), one finds that it is in fact typical for code designs optimized for SM-EPA with $N = 8$. As a matter of fact, an LDPC code design with a wide VN degree distribution and a wide PC degree distribution presents a big challenge for the interleaver design. In Fig. 7.17(b) we have applied the RGB algorithm for constructing the LDPC sub-matrix. The reason for doing so can be found in Fig. 7.34, which demonstrates the noticeable difference between a PEG-constructed LDPC sub-matrix and an RGB-constructed LDPC sub-matrix. One observes that using the PEG algorithm for the LDPC sub-matrix leads to a convergence tunnel closed in the middle region and consequently an almost flat BER curve. In contrast, by using the RGB algorithm for the LDPC sub-matrix, the promised decoding threshold at 8 dB is achieved, albeit with a non-trivial error floor. For the sake of compactness, we delay the discussion on the error floor to the part treating with short cycles.

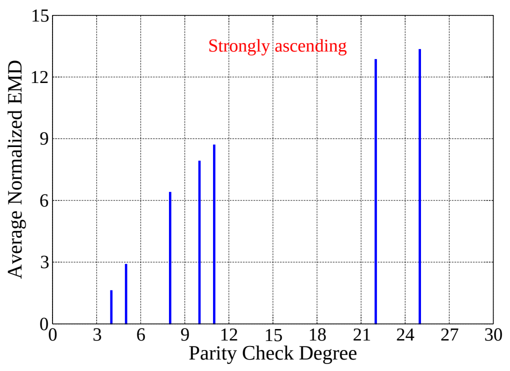
Since popular LDPC code designs all put the irregularity mainly in the VN degree distribution, the PEG algorithm is indeed not devised for a code design as in (7.58). Nonetheless, for the simulations in Fig. 7.34(b) the LDSC sub-matrix has been constructed via the PEG algorithm. This causes no problem since the code design under current consideration applies a regular summation check degree distribution. The large performance difference between a PEG-constructed LDPC sub-matrix and an RGB-constructed LDPC sub-matrix in fact comes from the different parity check EMD distributions. Given a wide VN degree distribution and a wide PC degree distribution, the PEG algorithm tends to produce a severely dispersed parity check EMD distribution, as illustrated in Fig. 7.35(a). Moreover, it tends to fill up the sockets of low-degree parity checks by the edges from low-degree variable nodes. To see this, let us define the normalized EMD of a check node as the EMD of this check divided by the degree of this check. Averaging the normalized EMD over all parity checks with the same degree, we obtain a parity check EMD spectrum as in Fig. 7.35(b). One observes that the average normalized EMD is strongly ascending w.r.t. the parity check degree, which means that the majority of low-degree variable nodes are connected to the low-degree parity checks and vice versa. However, for LDHC-EPA with a large N, only those low-degree parity checks can produce meaningful extrinsic messages in the early decoding stage. In case that low-degree parity checks are fully connected with low-degree variable nodes, these messages cannot be effectively strengthened and disseminated over the graph. This explains the flat BER curve resulting from a PEG-constructed LDPC sub-matrix, since the decoding process gets stuck in a rather early stage. Note that in Fig. 7.34(a) the leftmost section of the tunnel is opened by the repetition decoder instead of the LDPC decoder. In contrast, the parity check EMD distribution resulting from an RGB-constructed LDPC sub-matrix is wide but concentrated, cf. Fig. 7.35(c). Besides, the average normalized EMD is only slightly ascending w.r.t. the parity check degree, cf. Fig. 7.35(d). This means that the edges from low-degree variable nodes and the edges from high-degree variable nodes are more or less evenly spread over the graph. Consequently, a better performance has been achieved.

In reality, a wide check node degree distribution brings another problem. It is relatively difficult to attain a successful graph construction that strictly fulfills the designated check node degree distribution. The reason is that the minimum-current-connectivity-selection (MCCS) treatment can not properly handle graph construction given a wide and dispersed check node degree distribution. With the MCCS treatment, both the PEG algorithm and the RGB algorithm tend to first fill up all the sockets from the low-degree check nodes and leave a situation that a big amount of remaining edges have to be plugged to a small amount of high-degree check nodes. Since one cannot plug multiple edges from a single variable node to the same check node[3], the graph construction process often fails in

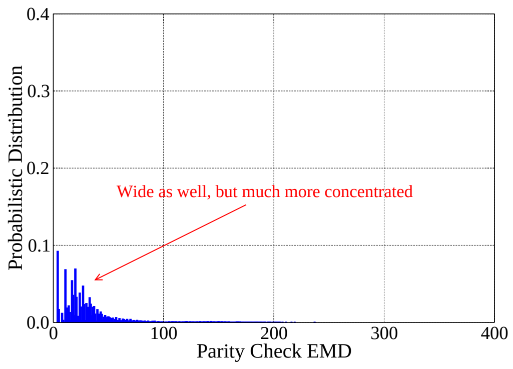
[3]Doing so will create length-2 cycles, which is extremely harmful for the performance.



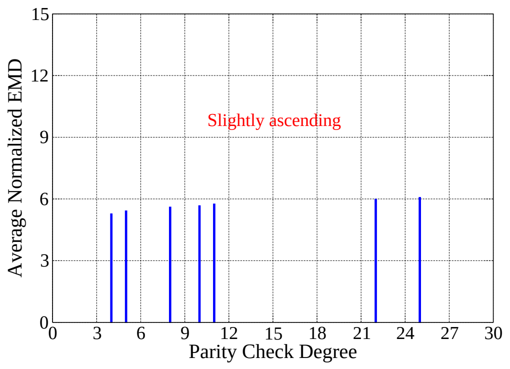
(a) PEG with MCCS.



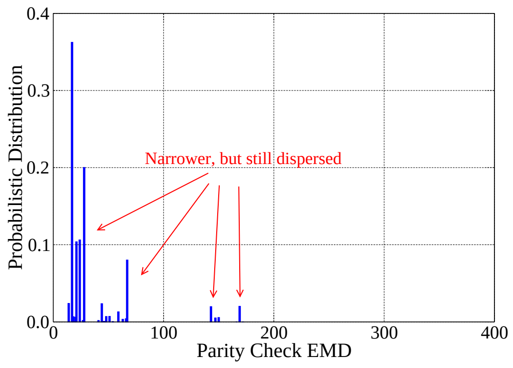
(b) PEG with MCCS.



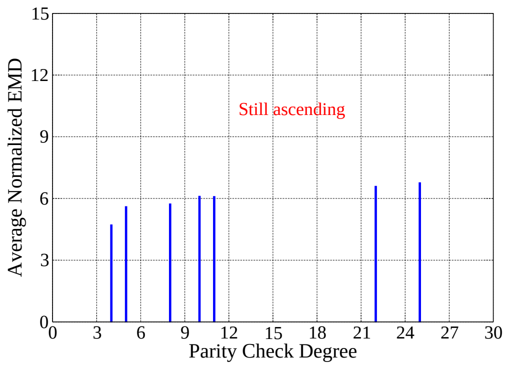
(c) RGB with MCCS.



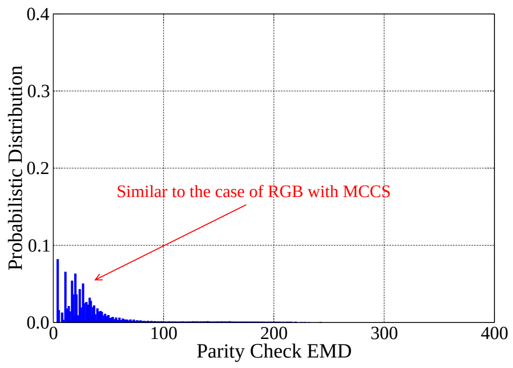
(d) RGB with MCCS.



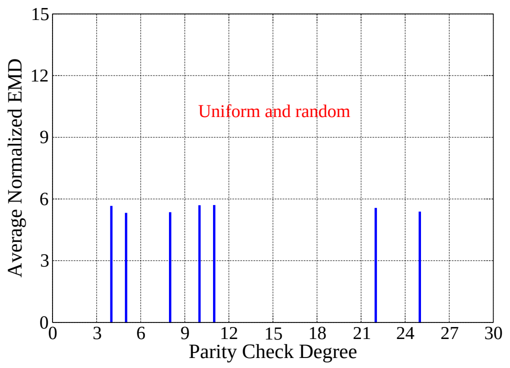
(e) PEG with MCFS.



(f) PEG with MCFS.



(g) RGB with MCFS.



(h) RGB with MCFS.

Figure 7.35: Parity check EMD distributions for the LDPC sub-matrix, $R_p = 1/2$.

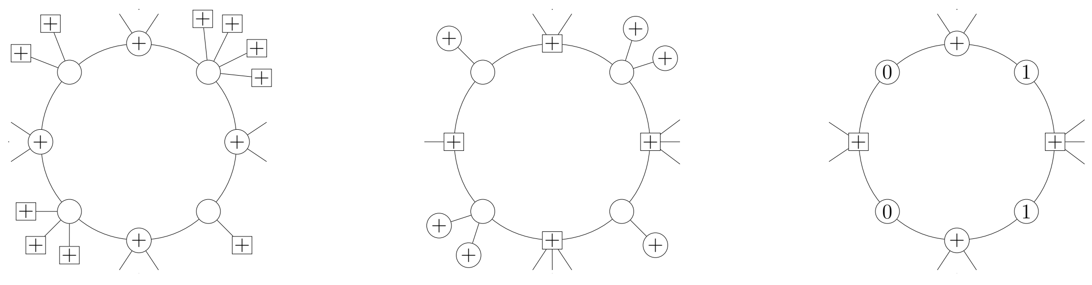
the late stage. To solve this practical problem, we propose a minimum-current-fullness-selection (MCFS) treatment for the PEG algorithm and the RGB algorithm. We define the current fullness of a check node as

$$\text{current fullness of a check node} \doteq \frac{\text{the amount of currently connected edges}}{\text{designated degree of the check node}} . \tag{7.62}$$

During an edge-growth procedure, when there are multiple check nodes fulfilling the cycle-length requirement for the current variable node, we connect the new edge from the current variable node to the check node candidate that has the minimum current fullness among the others. Adopting the MCFS treatment, the success rate of the PEG algorithm and the RGB algorithm can be significantly improved, for code designs that apply a wide and dispersed check node degree distribution. Note that the MCFS treatment is equivalent to the MCCS treatment given a regular check node distribution. The advantage of the MCFS treatment lies in the fact that it treats check nodes with different designated degrees in a fair way. For example, given a degree-6 check node that has 4 currently connected edges and a degree-20 check node that has 5 currently connected edges, the MCCS treatment will select the degree-6 check node for linking the new edge, while the MCFS treatment will select the degree-20 check node. Clearly, the best choice is to take the degree-20 check node as it has much more free sockets available w.r.t. the degree-6 check node. Applying the MCFS treatment also brings changes to the check node EMD distribution, as demonstrated in Fig. 7.35(e) to Fig. 7.35(h). The situation is improved for both the PEG algorithm and the RGB algorithm. Note that an interleaver-independent EXIT chart analysis implicitly assumes an identical normalized EMD for all check nodes. Nevertheless, the check node EMD distribution is still dispersed, given the PEG algorithm. Hence, a good decoding convergence is still not achievable, cf. Fig. 7.34(b). Given the RGB algorithm, the MCFS treatment brings a noticeable performance improvement in the low-SNR region but not in the high-SNR region. The reason is that the edges from certain variable node groups are less congested in the low-degree check nodes but more congested in the high-degree check nodes. Afterall, the non-trivial error floor in Fig. 7.34(b) is in fact closely related to those short cycles among summation checks and parity checks.

Cycles in an LDHC Graph

The graph for an LDHC code contains three different types of cycles, as depicted in Fig. 7.36. The first type of cycles contain only summation checks, cf. Fig. 7.36(a). From the study in Chapter 6, it is clear that short cycles of this type should be avoided by using a proper interleaver pattern, as they are responsible for the decoding problem in the early stage and consequently responsible for burst errors. The second type of cycles contain only parity checks, cf. Fig. 7.36(b). It is well-known in the community

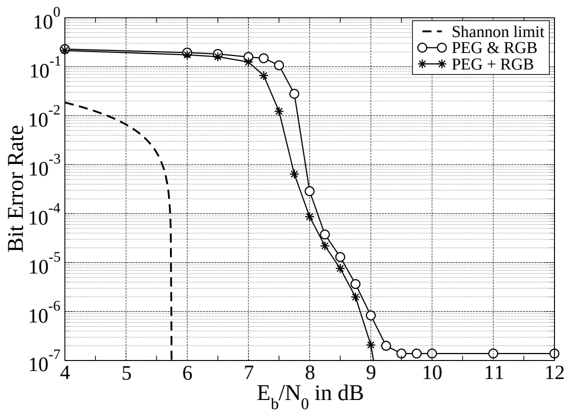


(a) A cycle with pure SC's. (b) A cycle with pure PC's. (c) A cycle with SC's and PC's.

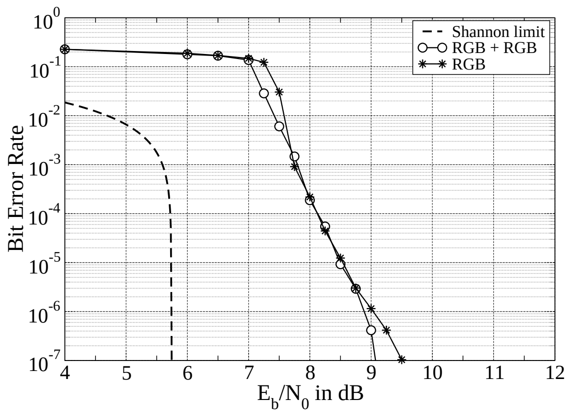
Figure 7.36: Three types of cycles in the factor graph for an LDHC code.

of LDPC coding that short cycles of this type should be avoided, because such cycles degrade the minimum code word distance and sequentially degrade the achievable power efficiency. The third type of cycles contain summation checks as well as parity checks, cf. Fig. 7.36(c). Now, the question is if it is necessary to eliminate short cycles of this type as well. The answer is definitely positive. The three cycles in Fig. 7.36 in fact gives a good reason. In an LDHC graph, a stopping set can be formed if and only if all check nodes in a certain set are connected to the variable nodes of this set at least twice. Therefore, the cycle in Fig. 7.36(a) does not form a stopping set, due to the extrinsic parity checks connected to the variable nodes of this cycle. Similarly, the cycle in Fig. 7.36(b) does not form a stopping set as well, because of the extrinsic summation checks connected to the variable nodes of this cycle. However, the cycle in Fig. 7.36(c) does form a stopping set, in case of equal power allocation. Suppose that in a certain iteration, all the variable nodes connected to the parity checks of this cycle have been correctly estimated, except those variable nodes belonging to this cycle, and all the variable nodes connected to the summation checks of this cycle have been correctly estimated, except those variable nodes belonging to this cycle. In this case, ambiguity-free decisions for the variable nodes within this cycle are still not possible, if their values are as marked in Fig. 7.36(c). Note that a parity check does not distinguish between the bit pairs: $(0, 0)$ and $(1, 1)$. As a matter of fact, for LDHC decoding the third type of cycles are primarily responsible for the error floor, if it exists. For an LDHC code that contains a vast amount of variable nodes that have SC-side repetition degree 1 and PC-side repetition degree 1, this type of cycles are critical for the performance. The numerical results in Section 7.5 will clearly demonstrate this phenomenon, when we try to approach the capacity of the noiseless BAC.

To see the necessity of eliminating short cycles of the type given by Fig. 7.36(c), let us still consider the code design in (7.58) as an example. For an easy reference, now we define a naming convention for the three types of matrix construction methods shown in Fig. 7.33. By constructing the LDSC sub-matrix via the PEG algorithm and the LDPC sub-matrix via the RGB algorithm in a separate way, we refer to the corresponding



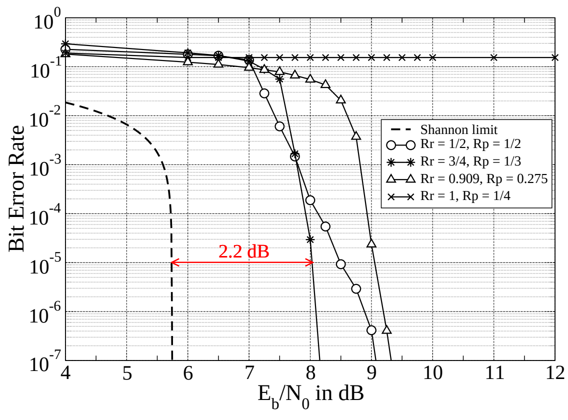
(a) Separate design vs. block-wise design.



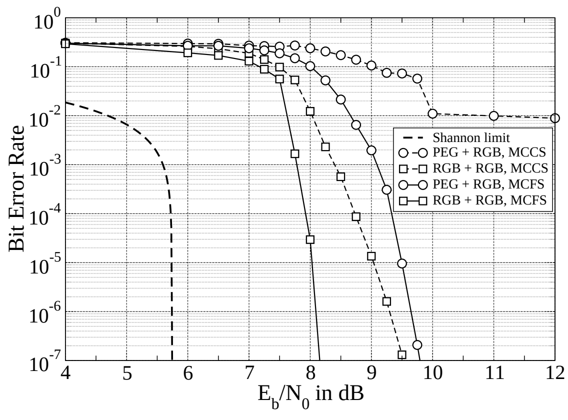
(b) Block-wise design vs. all-in-one design.

Figure 7.37: LDHC-EPA, $R_r = 1/2$, $R_p = 1/2$, $K = 40000$, VBS, 100 global iterations.

interleaver design method as "PEG & RGB". By constructing the LDSC sub-matrix via the PEG algorithm and the LDPC sub-matrix via the RGB algorithm in a block-wise fashion, we refer to the corresponding interleaver design method as "PEG + RGB". Finally, in case that the whole LDHC matrix is constructed all-in-once via the RGB algorithm, we refer to the corresponding interleaver design method simply as "RGB". Fig. 7.37(a) checks the performance difference between a separately constructed interleaver pair and a block-wise constructed interleaver pair. It can be seen that by removing those short cycles containing summation checks as well as parity checks a noticeable performance improvement is achieved. Particularly, no error floor has been observed above 10^{-7}. This tells that the error floor from a separate interleaver design is mainly caused by stopping sets similar to that in Fig. 7.36(c). Besides, as the BER performance also becomes better in the low-SNR region, removing cycles among summation checks and parity checks also improves the overall code distance spectrum. As a block-wise interleaver design is clearly beneficial, one may expect that an all-in-one interleaver design will further improve the performance. Nevertheless, the true situation is in fact the opposite, as demonstrated by Fig. 7.37(b). In order to have a fair comparison, the block-wise interleaver design applies the RGB algorithm for both the LDSC sub-matrix and the LDPC sub-matrix. One observes that an all-in-one interleaver design brings some penalty in the low-SNR region as well as in the high-SNR region. The reason can be found in (7.58). On average, the SC-side repetition degrees are smaller than the PC-side repetition degrees. As a result, short cycles in the LDSC sub-matrix are more critical for the performance. By constructing the LDSC sub-matrix first and the LDPC sub-matrix second, one gives a higher priority to the elimination of short cycles in the LDSC sub-matrix, and consequently achieves a better performance. Furthermore, a block-wise interleaver design gives an additional flexibility in combining the PEG algorithm and the RGB algorithm. Carefully checking Fig. 7.37(a) and Fig. 7.37(b), one finds that a "PEG + RGB" combination gives a better



(a) RGB + RGB.



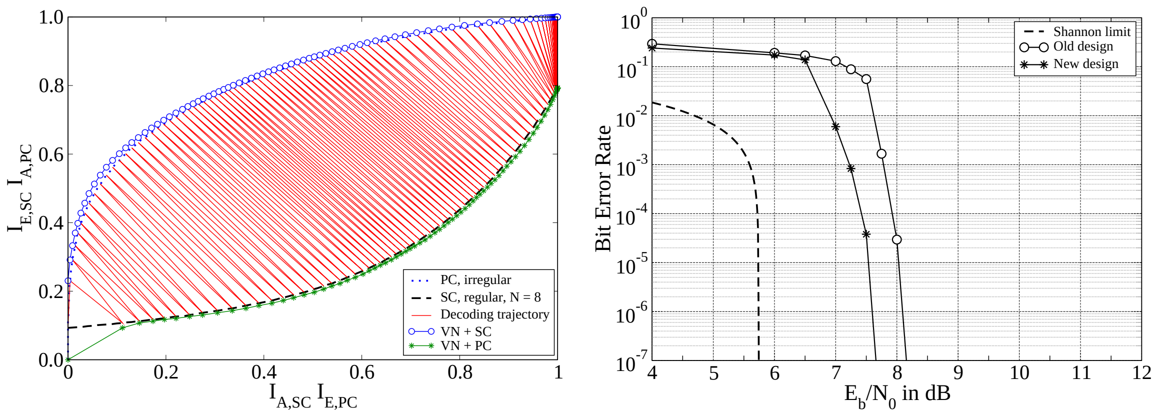
(b) $R_r = 3/4$, $R_p = 1/3$.

Figure 7.38: LDHC-EPA, $R = 1/4$, $K \approx 40000$, VBS, 100 global iterations.

performance than a "RGB + RGB" combination, in the high-SNR region. This is because a PEG-designed interleaver typically leads to a smaller residual error probability. In most cases, a block-wise interleaver design with the LDSC sub-matrix constructed first offers the best performance for a practical block length. For the sake of easy elaboration, we implicitly assume such a matrix construction order in the remaining part of this thesis.

Refining the Code Design for a Practical Block Length

The block-wise interleaver design in Fig. 7.37(a) achieves a decoding threshold even lower than 8 dB. However, the performance in the high-SNR region is not as good as promised by the corresponding code design. Certainly, one may apply an even more sophisticated interleaver design method to improve this situation by trading off the BER performance in the low-SNR region, e.g., by constraining the amount of low-degree variable nodes connected to each check node. Nevertheless, a more systematic solution is to adjust the code design, taking into account the imperfectness of a practical system with a finite block length. In general, a theoretically optimal code design does not necessarily lead to the best practically achievable performance. Moreover, the situation is often the opposite. As a good example, let us check the practically achievable performance of the four code designs provided in Section 7.4.3 for LDHC-EPA with $N = 8$. Fig. 7.38(a) provides the corresponding simulation results. One observes that the 7 dB code design given in (7.59) offers the best performance, while the 6 dB optimal code design given in (7.61) offers the worst performance. Furthermore, the 6 dB code design given in (7.60) enables a decoding convergence but leads to a performance worse than that given by (7.59) as well as (7.58). By comparing the coding rates corresponding to these four code designs, one recognizes that repetition code is important for achieving a good performance given a practical block



(a) EXIT chart. (b) BER vs. E_b/N_0, RGB + RGB.

Figure 7.39: LDHC-EPA, $R_r = 3/4$, $R_p = 1/3$, $K = 40000$, VBS, 100 global iterations.

length. Note that the code design in (7.60) is very similar to that in (7.61), except in using a tiny amount of SC-side repetitions. Nevertheless, this tiny amount of SC-side repetitions in fact bring a huge difference for the practically achievable performance. As a side topic, Fig. 7.38(b) checks the advantage of the MCFS treatment over the MCCS treatment for the 7 dB code design. One sees that the MCFS treatment generally outperforms the MCCS treatment. Since the 7 dB code design offers the best performance so far, we try to derive from it a new code design that offers a better practically achievable performance. Fig. 7.39(a) gives the EXIT chart for the following refined code design:

$$\begin{aligned}
\lambda_s(D) &= 0.866D^1 + 0.054D^2 + 0.020D^3 + 0.020D^4 + 0.020D^5 + 0.020D^6 \\
\lambda_p(D) &= 0.840D^2 + 0.020D^5 + 0.020D^6 + 0.020D^7 + 0.020D^9 \\
&\quad +0.040D^{10} + 0.020D^{17} + 0.020D^{52} \qquad (7.63) \\
\eta_p(D) &= 0.150D^4 + 0.500D^5 + 0.310D^6 + 0.020D^{17} + 0.020D^{35} \ ,
\end{aligned}$$

with $R_r = 3/4$ and $R_p = 1/3$. Compared to Fig. 7.32(a), we utilize the surplus in the right section of the convergence tunnel to widen the left section of the convergence tunnel. This is achieved by enlarging the irregularity both in the VN degree distribution and the PC degree distribution. The BER results in Fig. 7.39(b) shows that this new code design achieves a power gain of about 0.5 dB w.r.t. the old one. The above observation reveals that a sufficient surplus is necessary in the left section of the convergence tunnel in order to enable a successful iterative decoding process. As a matter of fact, such a code refining approach is effective not only for coded SM transmission over the AWGN channel but also for coded SM transmission over the noiseless channel, to be shown by various examples in Section 7.5. After all, a theoretically optimal code design tends to squeeze out all the area between an EXIT curve pair, while a practically optimal code design needs to guarantee an appropriate surplus area in an appropriate region.

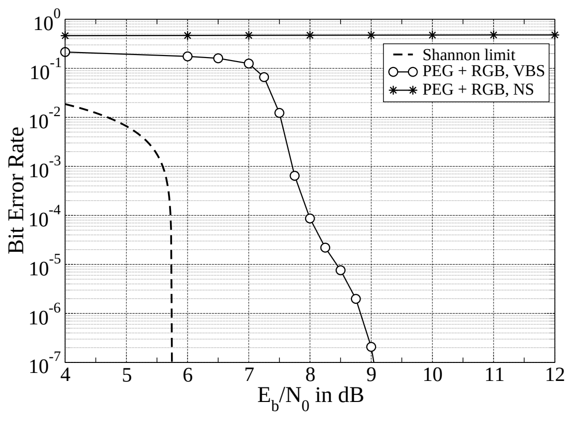


Figure 7.40: LDHC-EPA, $R_r = 1/2$, $R_p = 1/2$, $K = 40000$, 100 global iterations.

7.4.5 Is Scrambling Still Necessary?

Revealed by the study in Chapter 6, the effects of scrambling on repetition-coded SM-EPA are avoiding the trap of message oscillation and improving the separability of directly or indirectly overlapped repetition code words. Given LDHC-EPA, it is unclear if scrambling is still necessary, as each code bit is not only protected by SC-side repetitions but also PC-side repetitions. Nevertheless, a single simulation is sufficient to clarify this issue. In Fig. 7.40, the effect of scrambling for LDHC-EPA is clearly demonstrated. The adopted code design is from (7.58). With no scrambling (NS), the decoder can not converge at all. In contrast, given the same interleaver, the decoder converges well with variable-node based scrambling (VBS). The reason for causing this dramatic performance difference is evidently the periodic message oscillations within the LDSC sub-graph. Hence, scrambling is still necessary, even when the repetition encoder has been concatenated with an LDPC encoder. Certainly, in case that the SC-side repetition degrees are all 1, i.e., the LDHC encoder is a pure LDPC encoder, scrambling is no longer necessary. In such scenarios, the trap of message oscillation does not exist and it is not possible to use scrambling to improve the separability of superimposed chip sequences. Finally, it is also not necessary to apply cycle-based scrambling (CBS) for LDHC-EPA. As indicated by Fig. 7.36, given that the PC-side repetition degrees are all larger than 0, a stopping set can not be formed by pure summation checks. Therefore, the function of CBS in eliminating residual decision errors also vanishes. Easy to imagine, the situation for LDHC-GPA is similar to that for LDHC-EPA, given that the group size is larger than 1. Besides, it is also easy to imagine that scrambling is not necessary for LDHC-UPA. For the sake of compact elaboration, we assume VBS for all code designs that have SC-side repetition degrees larger than 1, and we assume no scrambling for all code designs that have SC-side repetition degrees all equal to 1, in the remaining part of this thesis. Further discussions on the effect of scrambling will be excluded.

7.5 Code Design Examples for the Noiseless BAC

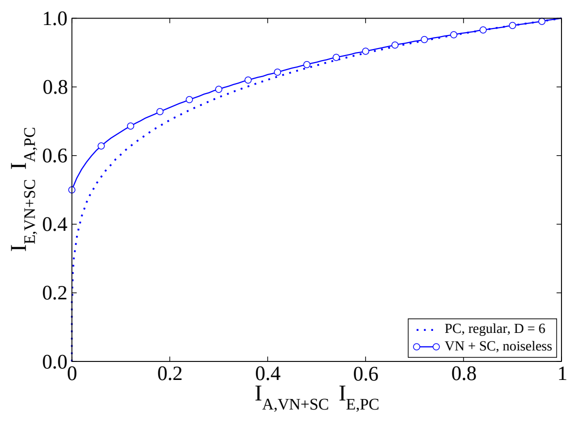
For SM-EPA, the primary task of channel coding is to enable a perfect data separation, since superimposed chips with identical magnitudes severely interfere with each other. As the extent of interference raises with the bit load N, the difficulty in enabling a perfect data separation also increases with N. To highlight this aspect, we provide in this section various code design examples for achieving high rates from SM-EPA transmission over the noiseless channel. Equivalently, this is to design coding schemes that can work closely to the capacity of the noiseless binary adder channel (BAC). Power efficiency will not be concerned within this discussion. Nevertheless, for the sake of easy visualization, all the BER performances will still be tested in the BAC with an additive white Gaussian noise. According to Section 7.4.4, an LDPC code design applying a wide VN degree distribution as well as a wide PC degree distribution presents a big challenge for the interleaver design given a practical block length. Hence, we will stay with regular PC degree distributions throughout this section. Such a code design constraint significantly relaxes the requirement on the interleaver quality, and meanwhile causes no noticeable penalty in the achievable bandwidth efficiency for the noiseless BAC, as long as a properly configured irregular repetition code is applied when necessary.

7.5.1 The Case of $N = 2$

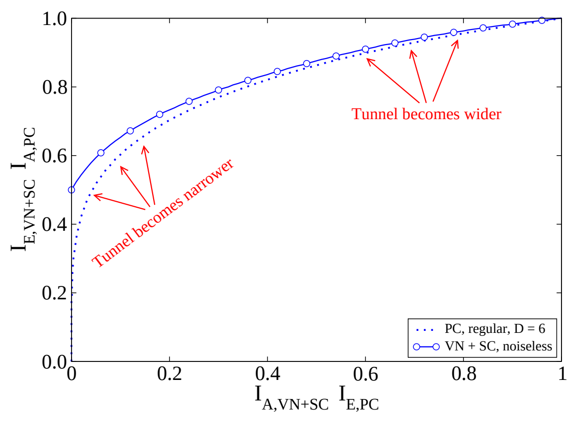
As shown in Section 7.3.1, the EXIT curve of SM-EPA demapping with $N = 2$ starts from a relatively high position. Therefore, a pure parity-check code should provide a desirable performance. According to Tab. 3.3, the symbol entropy of SM-EPA with $N = 2$ is 1.5 bits/symbol. Leaving some room for practical imperfectness, we target at a bandwidth efficiency of 1.45 bits/symbol, which requests the coding rate to be $R = 1.45/2 = 29/40$. After some fine tuning, we identify the following degree distribution set:

$$\begin{aligned} \lambda_s(D) &= 1.0D^1 \\ \lambda_p(D) &= 0.69D^1 + 0.24D^2 + 0.04D^3 + 0.01D^6 + 0.01D^{10} + 0.01D^{20} \\ \eta_p(D) &= 1.0D^6 \,, \end{aligned} \tag{7.64}$$

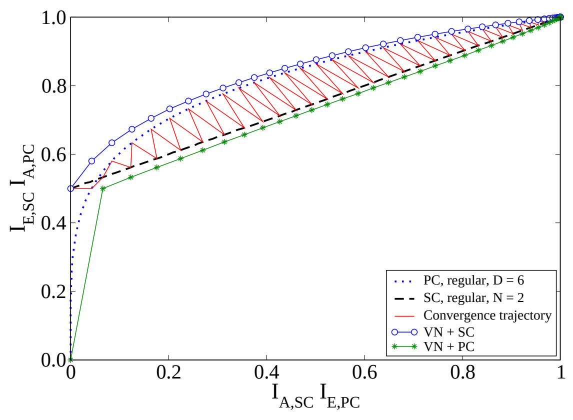
which leads to $R_r = 1$ and $R_p = 29/40$. Since the SC-side repetition degrees are all 1, we may consider a two-part (2-P) EXIT chart analysis as illustrated in Fig. 7.24(a). This helps us to survey the influence of the VN degree alignment on the decoding performance, which has been briefly introduced in Section 7.4.2. Fig. 7.41(a) provides a 2-P EXIT chart for the above degree distribution set given a sorted VN degree alignment, i.e., the variable nodes are aligned in the graph such that the degrees are non-descending, cf. Fig. 7.22(b).



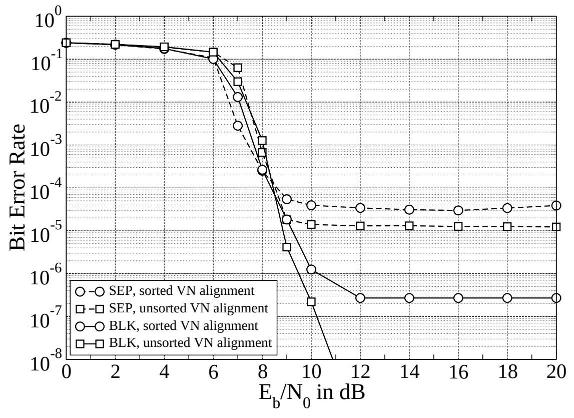
(a) 2-P EXIT chart, sorted VN alignment.



(b) 2-P EXIT chart, unsorted VN alignment.



(c) 3-P EXIT chart, independent of VN alignment.

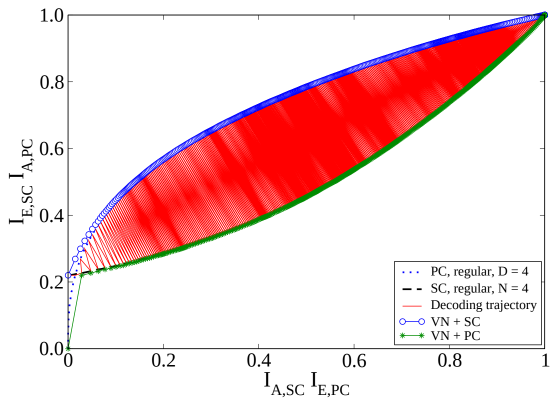


(d) BER vs. E_b/N_0.

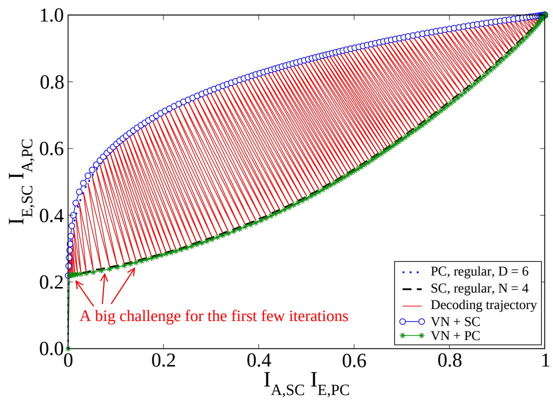
Figure 7.41: LDPC-coded SM-EPA, $R = 29/40$, $N = 2$, $K = 40000$, PEG, 100 iterations.

In this case, high-degree VN's are all connected to neighboring summation checks. The resulting convergence tunnel is wide in the left region but narrow in the right region. Now, we apply an unsorted VN degree alignment such that high-degree VN's get evenly spread over summation checks, cf. Fig. 7.23. The corresponding 2-P EXIT chart in Fig. 7.41(b) shows that this achieves a balance between the left and right region of the convergence tunnel. On the other hand, a three-part (3-P) EXIT chart resulting from the emulation technique proposed in Section 7.4.3 is independent of the VN degree alignment. Virtually, it assumes that all summation checks are linked with an equal amount of VN's of a certain degree. Comparing Fig. 7.41(c) and 7.41(b), one finds that the emulated VN-plus-SC EXIT curve is approximately the same as the one from a 2-P EXIT chart assuming an unsorted VN degree alignment. In the community of LDPC-coded modulation, the issue of VN degree alignment is commonly ignored, mainly because of the weak interaction between a bijective signal demapper and a channel decoder. For LDPC-coded SM-EPA, however, the VN degree alignment makes a big difference in the decoding performance. Fig. 7.41(a) and 7.41(b) show that a sorted VN degree alignment is beneficial for the early decoding stage while a unsorted VN degree alignment is beneficial for the late decoding stage. An early-stage decoding problem typically leads to burst errors, while a late-stage decoding problem usually leads to residual errors. For this reason, a sorted VN degree alignment is better for the high-BER region, while an unsorted VN degree alignment is better for the low-BER region, as shown in Fig. 7.41(d). To have a fair comparison, we have fixed the matrix construction method to be PEG for all simulations in Fig. 7.41(d). Given a separately (SEP) designed interleaver, i.e., an LDPC sub-matrix constructed independently of the LDSC sub-matrix, the error floor level is non-trivial for both VN degree alignments. The main causes for this error floor are cycles containing hybrid checks as in Fig. 7.36(c). Note that the VN degree distribution in (7.64) consists of a dominating fraction of degree-2 VN's, i.e., in the corresponding graph the majority of variable nodes are connected with only a single parity check. For those variable nodes that have a PC-side repetition degree of 1, short cycles similar to that in Fig. 7.36(c) actually form small stopping sets, and consequently lead to a non-trivial error floor. Certainly, one can easily reduce the error floor level by means of a block-wise (BLK) interleaver design. As for the current case, the LDSC sub-matrix does not really need to be designed. One simply constructs a binary Toeplitz matrix as in Fig. 7.22(c), such that no extra interleaving is necessary after LDPC encoding. Afterwards, the LDPC sub-matrix is designed in a way that short cycles between parity checks and summation checks are also eliminated. Doing so brings a considerable enhancement. For a sorted VN degree alignment, residual errors are largely reduced. For an unsorted VN degree alignment, no error floor is observed.

In the currently available literature, the highest rate achieved for a two-user binary adder channel is 1.3178 bits/symbol, via a specifically designed uniquely decodable code [103]. As explained in Section 7.1.1, a uniquely decodable code cannot reach the Shannon limit due to strictly stipulating a zero error probability. An interesting study on this issue can be found in [107]. By pursuing a small error probability instead of a zero error probability, an LDPC code can achieve a significantly higher rate than a uniquely decodable code, as shown in the previous test. Nevertheless, in the available literature, the relevant studies are exclusively focused on rates $\leqslant 1$ bits/symbol [108, 109]. The first reason for such a situation is that researchers typically hesitate to use degree-1 variable nodes in an LDPC code, as this leads to a zero coding gain. However, for the noiseless BAC, it is in fact theoretically undesirable to use a code which delivers a non-zero coding gain. Since the summation check EXIT curve always ends at $(1, 1)$ given a noiseless channel, a decoder EXIT curve ending at $(\mathrm{I_E} = 1, \mathrm{I_A} < 1)$ is strictly non-optimal. The second reason is that the harmfulness of short cycles formed between parity checks and summation checks was unrecognized. Whenever an error floor is observed, researchers merely spend effort in improving the PC-side interleaver but not the overall interleaving scheme which includes summation checks. For the code design in (7.64), it is not feasible to attain a desirable performance without eliminating short cycles containing hybrid checks.



(a) Based on degree-4 parity checks.



(b) Based on degree-6 parity checks.

Figure 7.42: LDHC-EPA, $R_r = 1$, $R_p = 0.5075$, $N = 4$, noiseless channel.

7.5.2 The Case of $N = 4$

According to Tab. 3.3, the capacity of a 4-user noiseless BAC is about 2.0306 bits/symbol. Let us first check the theoretical supportable rate of LDPC codes for this channel. Since a 3-P EXIT chart provides a better visual aid for code design and it is simulation-free for tuning the degree distributions, from now on we exclude the use of conventional 2-P EXIT charts. We first consider the following degree distribution set:

$$\begin{aligned}\lambda_s(D) &= 1.0D^1 \\ \lambda_p(D) &= 0.626D^1 + 0.275D^2 + 0.068D^6 + 0.021D^8 \\ &\quad +0.003D^{16} + 0.002D^{18} + 0.003D^{26} + 0.002D^{28} \\ \eta_p(D) &= 1.0D^4\ ,\end{aligned} \tag{7.65}$$

which leads to $R_r = 1$ and $R_p = 0.5075$. Given this code design, the achieved bandwidth efficiency is 2.03 bits/symbol, which is in a negligible distance to the capacity. However, it can be seen from Fig. 7.42(a) that the decoding trajectory is extremely dense. Given such a code design, it is not feasible to achieve a decoding convergence with a finite-size graph full of cycles. For any rate below the capacity, the valid code designs are not unique. Based on degree-6 parity checks, the following code design

$$\begin{aligned}\lambda_s(D) &= 1.0D^1 \\ \lambda_p(D) &= 0.57D^1 + 0.28D^2 + 0.03D^3 + 0.045D^5 \\ &\quad +0.02D^7 + 0.04D^{10} + 0.01D^{22} + 0.005D^{150} \\ \eta_p(D) &= 1.0D^6\ ,\end{aligned} \tag{7.66}$$

with $R_r = 1$ and $R_p = 0.5075$, opens the convergence tunnel as well, cf. Fig. 7.42(b). Moreover, one observes that the decoding trajectory density is much lower than the former case. Hence, for a 4-user BAC, LDPC coding based on degree-6 parity checks has more potential than that based on degree-4 parity checks. However, it is still challenging to achieve a decoding convergence. The problem is in the left section of the convergence tunnel. Given an irregular VN degree distribution, the iterative decoding process always starts with high-degree variable nodes. Consequently, high-degree variable nodes have relatively small run-time local girths, which means that the run-time structural quality of a graph is far from that of a cycle-free graph during the first few iterations. This necessitates a large width for the leftmost section of the convergence tunnel. On the other hand, the run-time structural quality of a graph improves as the iterative decoding proceeds, since more and more variable nodes get correctly estimated and virtually eliminated from the graph. For this reason, it is usually not a problem to have a convergence tunnel with a small width in the rightmost region. Hence, given a practical block length, a preferable code design should provide a convergence tunnel with its width continuously decreasing from left to right. The code designs in Fig. 7.42(a) and Fig. 7.42(b) both do not satisfy this criterion. To achieve a decoding convergence for a practical block length, we have to reduce the data rate, so that there is a room for the imperfectness of the graph structure.

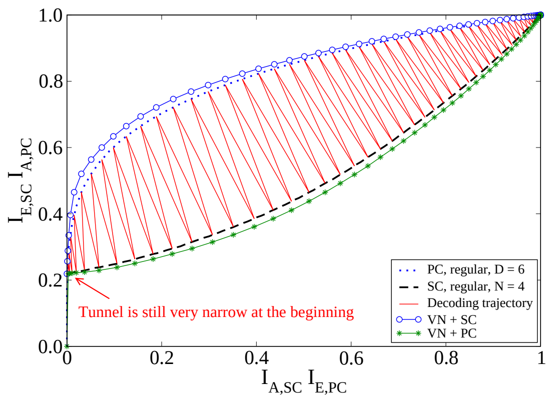
Let us target at a bandwidth efficiency of 1.9 bits/symbol, which requires $R = 0.475$. Since the leftmost section of the convergence tunnel is of the highest priority to be widened, we increase the highest variable node degree from 150 to 181. This leads to a code design as

$$\begin{aligned}
\lambda_s(D) &= 1.0D^1 \\
\lambda_p(D) &= 0.57D^1 + 0.28D^2 + 0.03D^3 + 0.045D^5 \\
&\quad +0.02D^7 + 0.04D^{11} + 0.01D^{22} + 0.005D^{181} \\
\eta_p(D) &= 1.0D^6 \,,
\end{aligned} \tag{7.67}$$

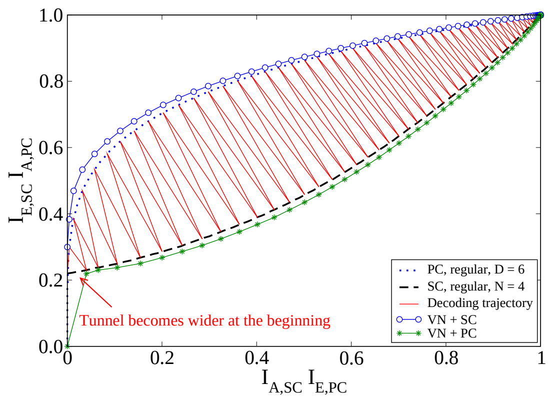
with $R_r = 1$ and $R_p = 0.475$. The resulting EXIT chart is provided in Fig. 7.43(a). One observes that the convergence tunnel does not become wider in the leftmost region. As a result, this new code design does not make the first few iterations easier. The BER results in Fig. 7.43(d) clearly support this statement. Either with a sorted VN alignment or an unsorted VN alignment, the decoder cannot converge at all. This observation tells that adding PC-side repetitions is not effective for the leftmost region, given a regular parity check degree distribution. Now we apply an alternative degree distribution set:

$$\begin{aligned}
\lambda_s(D) &= 0.932D^1 + 0.068D^2 \\
\lambda_p(D) &= 0.57D^1 + 0.28D^2 + 0.03D^3 + 0.045D^5 \\
&\quad +0.02D^7 + 0.04D^{10} + 0.01D^{22} + 0.005D^{150} \\
\eta_p(D) &= 1.0D^6 \,,
\end{aligned} \tag{7.68}$$

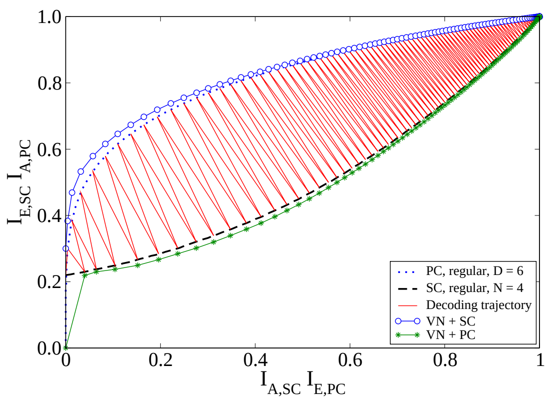
with $R_r = 0.9363$, $R_p = 0.5075$, and $R = R_rR_p = 0.475$. In contrast to the code design in (7.67), all the increased redundancy is now devoted to the SC-side repetitions. As



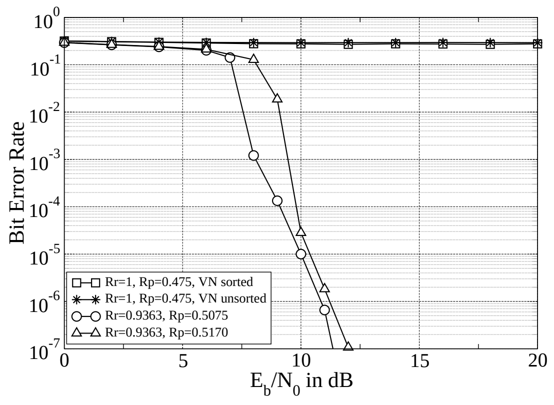
(a) $R_r = 1$, $R_p = 0.475$, $R = 0.475$.



(b) $R_r = 0.9363$, $R_p = 0.5075$, $R = 0.475$.



(c) $R_r = 0.9363$, $R_p = 0.517$, $R = 0.484$.



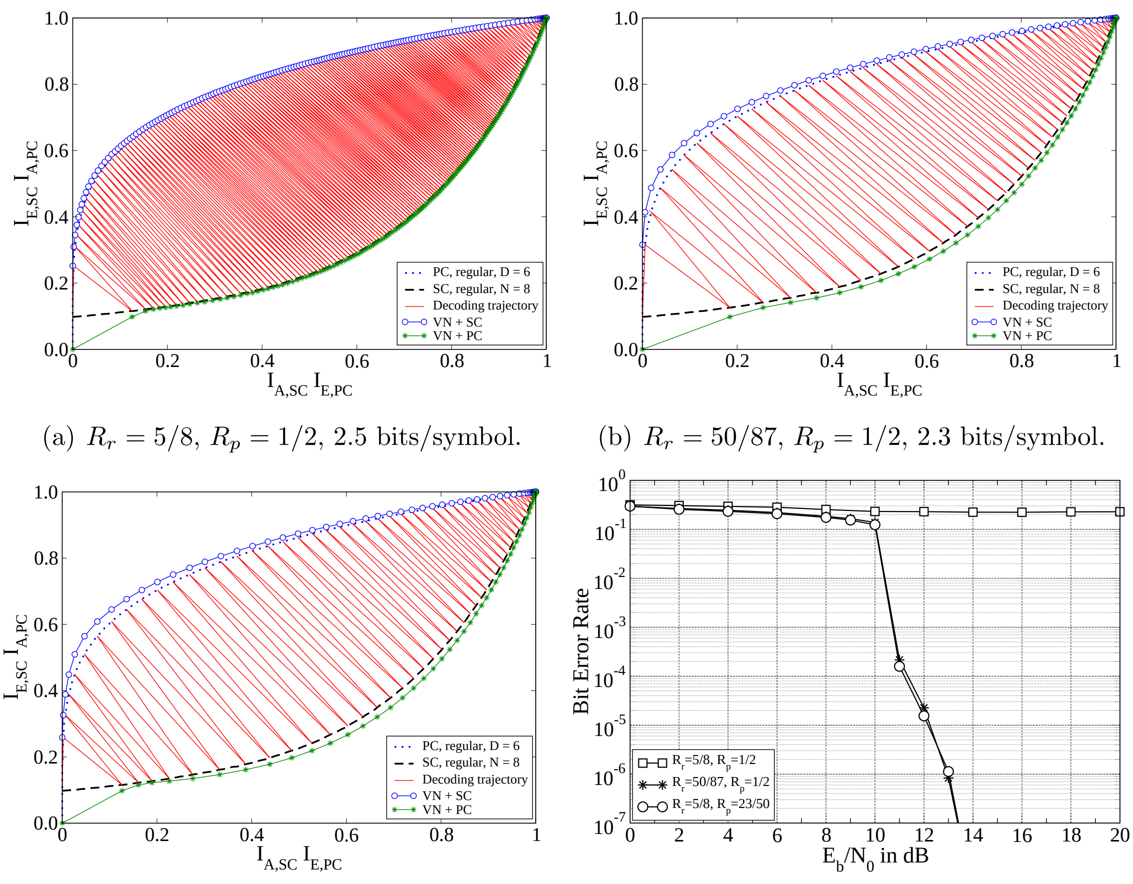
(d) BER vs. E_b/N_0.

Figure 7.43: LDHC-EPA, $N = 4$, $K \approx 80000$, PEG + RGB, 200 iterations.

shown in Fig. 7.43(b), this effectively widens the tunnel in the leftmost region. Applying a block-wise LDHC matrix construction, the decoder converges very well, cf. Fig. 7.43(d). Nevertheless, as the convergence tunnel is now wide for the whole region, there is still space to increase the coding rate. In most cases, the convergence tunnel does not need to be very wide in the right region. Under this motivation, we apply

$$\begin{aligned}\lambda_s(D) &= 0.932D^1 + 0.068D^2 \\ \lambda_p(D) &= 0.597D^1 + 0.253D^2 + 0.035D^3 + 0.04D^5 \\ &\quad +0.02D^7 + 0.04D^{10} + 0.01D^{21} + 0.005D^{148} \qquad (7.69) \\ \eta_p(D) &= 1.0D^6 ,\end{aligned}$$

with $R_r = 0.9363$ and $R_p = 0.517$. This leads to $R = 0.4869$ and $R{\cdot}N = 1.94$ bits/symbol. The resulting EXIT chart is provided in Fig. 7.43(c). One observes that the decoding trajectory becomes very dense in the right region. Nevertheless, this is not a big problem for the decoder, as long as the number of iterations is sufficiently large. The corresponding BER curve in Fig. 7.43(d) verifies the feasibility of such a code design.



(a) $R_r = 5/8$, $R_p = 1/2$, 2.5 bits/symbol.

(b) $R_r = 50/87$, $R_p = 1/2$, 2.3 bits/symbol.

(c) $R_r = 5/8$, $R_p = 23/50$, 2.3 bits/symbol.

(d) BER vs. E_b/N_0.

Figure 7.44: LDHC-EPA, $N = 8$, $K \approx 160000$, 200 iterations.

7.5.3 The Case of $N = 8$

According to Tab. 3.3, the capacity of a noiseless 8-user BAC is about 2.5442 bits/symbol. Let us first try to search a valid code design that achieves a data rate of 2.5 bits/symbol, which stipulates the coding rate to be $R = 5/16$. To ease the task, we assume an infinite block length, such that an EXIT chart analysis gives an accurate performance prediction. Since repetition coding tends to be near-optimal for SM-EPA with a large N, we consider a code design that devotes a considerable amount of redundancy to the SC-side repetitions:

$$\begin{aligned}
\lambda_s(D) &= 0.728D^1 + 0.076D^2 + 0.070D^3 + 0.120D^4 + 0.006D^5 \\
\lambda_p(D) &= 0.480D^1 + 0.359D^2 + 0.012D^3 + 0.050D^4 + 0.046D^7 \\
&\quad +0.022D^8 + 0.015D^{16} + 0.010D^{30} + 0.006D^{88} \qquad (7.70) \\
\eta_p(D) &= 1.0D^6 \,,
\end{aligned}$$

with $R_r = 5/8$ and $R_p = 1/2$. The corresponding EXIT chart in Fig. 7.44(a) reveals that this code design is in fact near-optimum. Clearly, a decoding convergence will only be achievable given an infinite block length. Given a practical block length, we have to relax the convergence tunnel a little bit, particularly in the left region. For the current system setup, there are two opportunities to widen the convergence tunnel. One may increase the degree of SC-side repetitions or the degree of PC-side repetitions. Let us first consider increasing the degree of SC-side repetitions. To leave some space for code adjusting, we reduce the targeted data rate to 2.3 bits/symbol. Correspondingly, the overall coding rate is to be $R = 23/80$. Keeping the parity-check coding rate as $R_p = 1/2$, we reduce the repetition coding rate to $R_r = 50/87$. This leads to a degree distribution set as

$$\begin{aligned}
\lambda_s(D) &= 0.724D^1 + 0.086D^2 + 0.083D^3 + 0.087D^5 + 0.020D^8 \\
\lambda_p(D) &= 0.480D^1 + 0.359D^2 + 0.012D^3 + 0.050D^4 + 0.046D^7 \\
&\quad +0.022D^8 + 0.015D^{16} + 0.010D^{30} + 0.006D^{88} \qquad (7.71) \\
\eta_p(D) &= 1.0D^6 \ .
\end{aligned}$$

Fig. 7.44(b) provides the corresponding EXIT chart. Compared to Fig. 7.44(a), the density of the decoding trajectory is now much lower, i.e., the convergence tunnel is much wider. Fig. 7.44(d) shows that the decoder converges very well, given this code design. On the other hand, given the code design in (7.70), the decoder cannot converge at all, cf. Fig. 7.44(d). Hence, it is indeed necessary to leave some space in the convergence tunnel, given a practical block length. For a systematic study, let us also check the situation when the surplus redundancy is allocated to the PC-side repetitions. After some fine tuning, we obtain the following degree distribution set:

$$\begin{aligned}
\lambda_s(D) &= 0.728D^1 + 0.076D^2 + 0.070D^3 + 0.120D^4 + 0.006D^5 \\
\lambda_p(D) &= 0.480D^1 + 0.359D^2 + 0.016D^3 + 0.046D^4 + 0.046D^7 \\
&\quad +0.022D^8 + 0.015D^{16} + 0.008D^{34} + 0.008D^{100} \qquad (7.72) \\
\eta_p(D) &= 1.0D^6 \ ,
\end{aligned}$$

with $R_r = 5/8$ and $R_p = 23/50$. This code design achieves the same bandwidth efficiency as that by (7.71). The resulting EXIT chart is given in Fig. 7.44(c). It can be seen that the density of the decoding trajectory is also noticeably reduced. Consequently, a decoding convergence is achievable as well for a finite block length, cf. Fig. 7.44(d). Besides, the performance is more or less identical to that of the code design given in (7.71).

7.5.4 The Case of $N = 16$

In Section 7.5.2, we have assigned a small portion of the redundancy to the SC-side repetitions for achieving a good decoding convergence given a finite block length. In

Section 7.5.3, we have assigned a considerable portion of the redundancy to the SC-side repetitions, but the majority of the redundancy is still assigned to the PC-side repetitions. Now, for a 16-user noiseless BAC, we can assign the majority of the redundancy to the SC-side repetitions, without any noticeable loss in the code optimality. The capacity of a 16-user noiseless BAC is about 3.0465 bits/symbol, cf. Tab. 3.3. The following degree distribution set

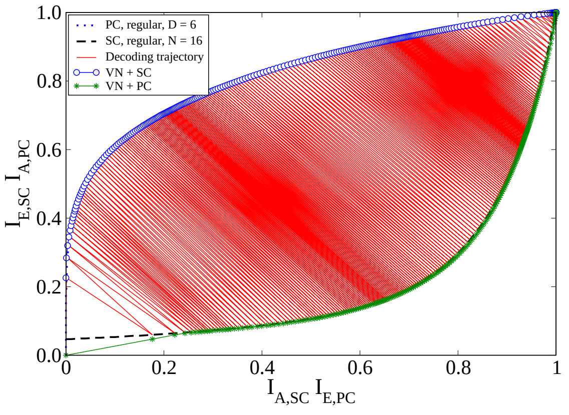
$$\begin{aligned}\lambda_s(D) &= 0.410D^1 + 0.312D^2 + 0.003D^3 + 0.006D^4 + 0.012D^6 + 0.195D^7 + 0.062D^8\\ \lambda_p(D) &= 0.638D^1 + 0.256D^2 + 0.006D^5 + 0.012D^6 + 0.019D^7 + 0.040D^{10}\\ &\quad +0.016D^{12} + 0.004D^{30} + 0.003D^{44} + 0.003D^{54} + 0.003D^{78} \qquad (7.73)\\ \eta_p(D) &= 1.0D^6\ ,\end{aligned}$$

with $R_r = 1/3$ and $R_p = 9/16$, effectively opens the convergence tunnel, as demonstrated in Fig. 7.45(a). This achieves a rate of 3 bits/symbol, assuming an infinite block length. Note that the repetition coding rate is now much lower than the parity-check coding rate. Certainly, to enable a robust decoding performance with a finite block length, we need to reduce the rate. Hence we consider the following degree distribution set

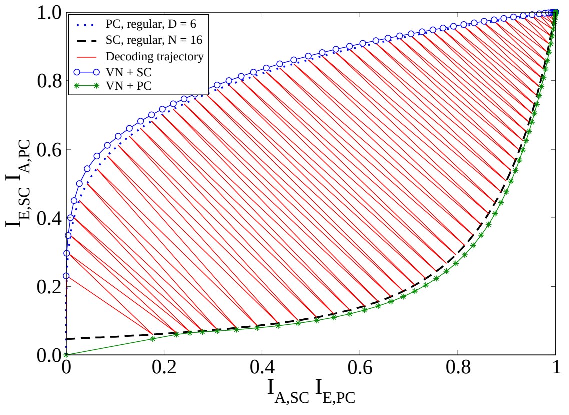
$$\begin{aligned}\lambda_s(D) &= 0.410D^1 + 0.312D^2 + 0.003D^3 + 0.006D^4 + 0.012D^6 + 0.195D^7 + 0.062D^8\\ \lambda_p(D) &= 0.617D^1 + 0.277D^2 + 0.006D^5 + 0.012D^6 + 0.019D^7 + 0.040D^{10}\\ &\quad +0.016D^{12} + 0.004D^{30} + 0.003D^{44} + 0.006D^{100} \qquad (7.74)\\ \eta_p(D) &= 1.0D^6\ ,\end{aligned}$$

which leads to $R_r = 1/3$ and $R_p = 21/40$. From the corresponding EXIT chart in Fig. 7.45(b), one observes that the resulting EXIT curves fit well with each other and there is appropriate surplus space existing in the convergence tunnel. Consequently, a good BER performance is observed in Fig. 7.45(c). Given this code design, the achieved rate is 2.8 bits/symbol, which is higher than that we have achieved via LDSC coding in Section 6.4.5. This shows the advantage of LDHC coding over LDSC coding.

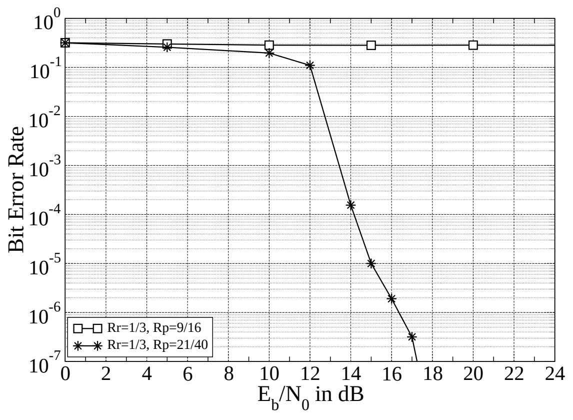
Due to the firm concept on the non-optimality of repetition coding for the binary-input AWGN channel, researchers are normally discouraged to use repetition codes for the binary adder channel in the concern of losing coding optimality. The provided code design examples so far clearly negates this way of thinking. By overlooking the importance of repetition coding, successful applications of sparse-graph codes for BAC with a large N have been rarely reported. This in turn explains the continuous research interest on uniquely decodable codes, despite the fact that this type of codes are proved to be non-capacity-achieving [107]. Easy to imagine, for N approaching the infinity, a pure repetition code will be able to offer an optimal performance for the noiseless BAC.



(a) $R_r = 1/3$, $R_p = 9/16$.



(b) $R_r = 1/3$, $R_p = 21/40$.



(c) BER vs. E_b/N_0.

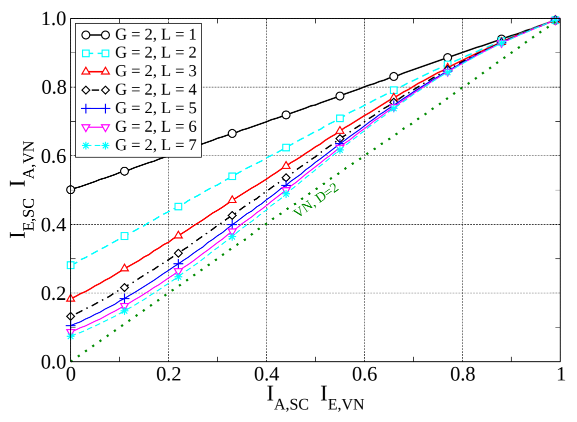
Figure 7.45: LDHC-EPA, $N = 16$, $K = 240000$, RGB + RGB, 200 iterations.

7.6 Code Design Examples for the AWGN Channel

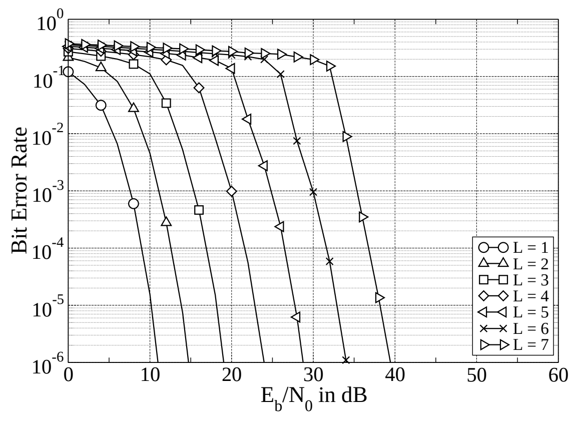
In the previous section, we have provided a selected collection of code designs for achieving the capacity of the noiseless BAC, i.e., for SM-EPA transmission over a noiseless channel. It is shown therein that the framework of LDHC coding is flexible and useful for various system setups. In this section, we will provide code design examples for SM transmission over the AWGN channel. Certainly, the focus will be on the achievable power efficiency instead of the supportable bandwidth efficiency.

7.6.1 Preliminary Remarks

For data transmission over the AWGN channel, SM-GPA offers many practical benefits w.r.t. SM-EPA. As introduced in Section 7.2.1, SM-GPA with a small group size has the best information-to-complexity ratio among the others. Besides, SM-GPA shows less non-bijectivity compared to SM-EPA, and meanwhile does not lose the optimality for transmission over the AWGN channel. The code design examples in Section 7.5 actually reveals one non-trivial drawback from SM-EPA. Given a large N, the convergence tunnel needs to be wide in the left region, in order to achieve a good decoding performance with a finite block length. Since any area between an EXIT curve pair leads to a rate loss relative to the capacity, the above property from SM-EPA with a large N is in fact undesirable for transmission over the AWGN channel. For combatting the non-bijectivity given a limited block length, a considerable code space needs to be consumed, which can otherwise be used to combat the additive noise. Therefore, given a short block length, the practically achievable power efficiency from SM-EPA with a large N is usually much lower than that promised by an optimal code design assuming an infinite block length. In comparison, SM-GPA attains a good trade-off between bijective uniform signal mapping and non-bijective nonuniform signal mapping. For data transmission over Gaussian channels, SM-GPA is the preferable choice. Furthermore, SM-GPA with $G = 2$ gives the most practical solution for high-rate data transmission. Revisiting Fig. 3.11 and Fig. 3.12, one finds that the theoretical performance improvement by using a group size larger than 2 is in fact marginal. Moreover, having a triangular symbol distribution instead of a Gaussian symbol distribution is in fact an advantage from SM-GPA with $G = 2$, since this largely reduces the receiver load in enabling a perfect data separation. Given the above statements, the discussion within this section will mainly focus on SM-GPA with $G = 2$. The property of SM-GPA with a large group size is very similar to that of SM-EPA with a large bit load, and so is the corresponding code design approach. Naturally, for data transmission over the AWGN channel, SM-GPA with a large group size is often inferior to SM-GPA with $G = 2$, assuming a high data rate and an identical block length.



(a) EXIT chart, noiseless channel.



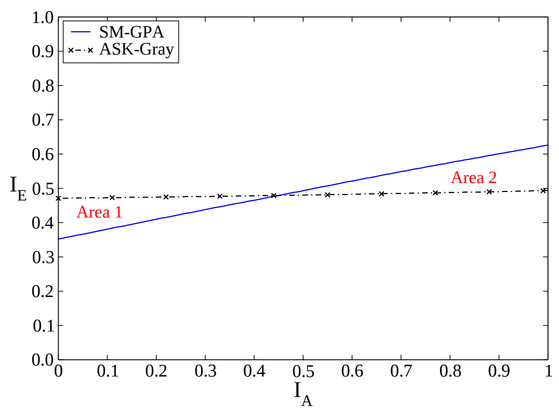
(b) BER vs. E_b/N_0.

Figure 7.46: LDSC-GPA, $SF = 2$, $G = 2$, $K = 20000$, PEG, 100 iterations.

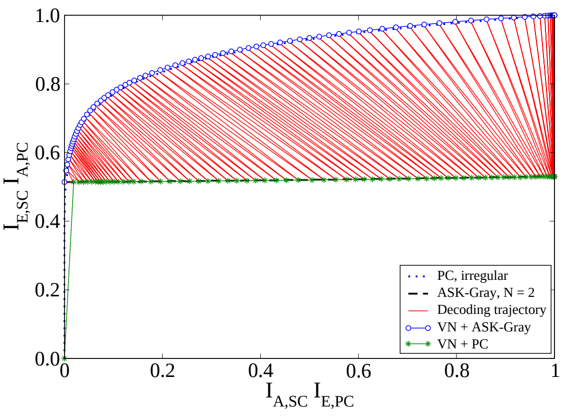
According to the discussion in Section 7.1.2, choosing $R = 1/G$ should principally enable a decoding convergence for SM-GPA with an arbitrary number of power levels. However, the simulation results in Fig. 5.10 clearly do not follow this conjecture. The reason for this is simple, which is the sub-optimality of random interleaving given a limited block length. As shown by Fig. 7.46(a), a rate 1/2 regular repetition code should already be able to support SM-GPA with $G = 2$ and $L = 7$, given an optimized interleaver. This is in fact the true situation when one adopts the framework of LDSC coding and applies the PEG algorithm for the interleaver design, cf. Fig. 7.46(b). The BER results in Fig. 7.46(b) vividly demonstrates the big advantage of SM-GPA w.r.t. SM-EPA for supporting high-rate data transmission.

7.6.2 The Case of 1 bit/symbol

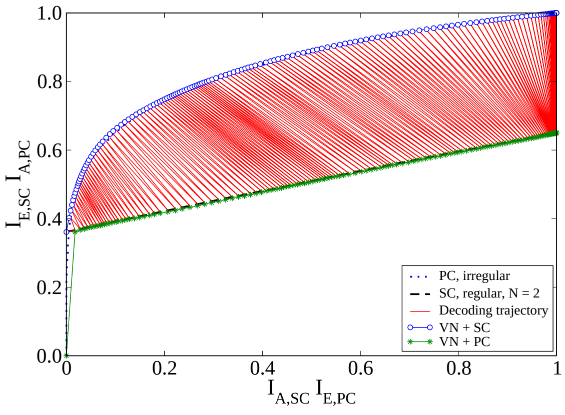
Since there are already plenty of successful code designs available in the literature for a rate of 0.5 bits/symbol, we will directly start with a rate of 1 bits/symbol over the AWGN channel. According to Fig. 3.7(b), SM-GPA with $G = 2$ and $L = 1$, i.e., SM-EPA with $N = 2$, is almost capacity-achieving for a rate of 1 bit/symbol, if not exactly. To achieve a rate of 1 bit/symbol, the coding rate should be $R = 1/2$. In order to get a rough impression for the code design task, we first check the EXIT function of SM-GPA demapping at an SNR given by the Shannon limit, which is about 1.8 dB for 1 bit/symbol. Fig. 7.47(a) compares the EXIT curve of SM-GPA with $G = 2$ and $L = 1$ to that of 4-ASK with Gray labeling. In the left region ASK outperforms SM-GPA, while in the right region SM-GPA outperforms ASK. It is easy to find that area 2 is slightly larger than area 1. The difference between area 2 and area 1 in fact gives the potential power gain that SM-GPA can ultimately achieve w.r.t. ASK. For a fair comparison, we first try to optimize



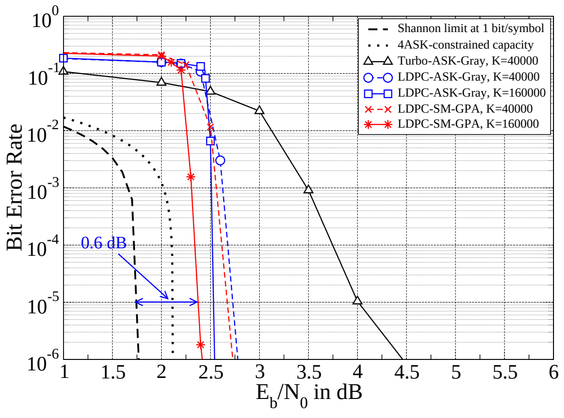
(a) SISO demapping, $E_b/N_0 = 1.8$ dB.



(b) LDPC-coded ASK-Gray, $E_b/N_0 = 2.5$ dB.



(c) LDPC-coded SM-GPA, $E_b/N_0 = 2.1$ dB.



(d) BER performance.

Figure 7.47: LDPC-coded SM-GPA, $R = 1/2$, $G = 2$, $L = 1$, RGB, 100 iterations.

the code for ASK, by using the EXIT emulation technique proposed in this thesis. We identify the following degree distribution pair:

$$\begin{aligned} \lambda_p(D) &= 0.350D^2 + 0.380D^3 + 0.070D^4 + 0.030D^5 + 0.040D^7 + 0.040D^9 \\ &\quad +0.040D^{14} + 0.020D^{17} + 0.010D^{24} + 0.010D^{60} + 0.010D^{85} \\ \eta_p(D) &= 0.100D^8 + 0.800D^{11} + 0.100D^{14} \end{aligned} \tag{7.75}$$

with $R_p = 1/2$. Fig. 7.47(b) gives the corresponding EXIT chart. Assuming an infinite block length, this code design should achieve a decoding threshold of 2.5 dB for ASK with Gray labeling. As a matter of fact, given a block length of 160000, this decoding threshold is almost achieved, cf. Fig. 7.47(d). Hence, the EXIT emulation technique originally derived for non-bijective superposition mapping turns out to be useful for bijective mapping as well. To have a more direct impression on the performance improvement achieved for ASK, we also provide the BER curve of rate 1/2 Turbo-coded 4-ASK as a reference. Though by no means we are referring to this curve as the state-of-the-art results, this curve serves as a good example for the performance of currently popular systems. From Fig. 7.47(d) one observes that the distance between the Shannon limit and the 4-ASK constrained capacity is about 0.37 dB. This value upper bounds the power gain of SM-GPA over ASK. We apply the following degree distribution pair:

$$\begin{aligned}\lambda_p(D) &= 0.620D^2 + 0.330D^3 + 0.020D^{12} + 0.010D^{20} + 0.010D^{33} + 0.010D^{50} \\ \eta_p(D) &= 0.120D^5 + 0.040D^6 + 0.560D^7 + 0.280D^8 \ ,\end{aligned} \tag{7.76}$$

with $R_p = 1/2$, for SM-GPA. The theoretically achievable decoding threshold is at about 2.1 dB, as indicated by Fig. 7.47(c). Given an RGB-designed interleaver, this decoding threshold is also almost achieved for a block length of 160000. Carefully checking Fig. 7.47(d), one finds that the attained power gain of SM-GPA w.r.t. ASK is about 0.2 dB at a BER of 10^{-5}.

7.6.3 The Case of 2 bits/symbol

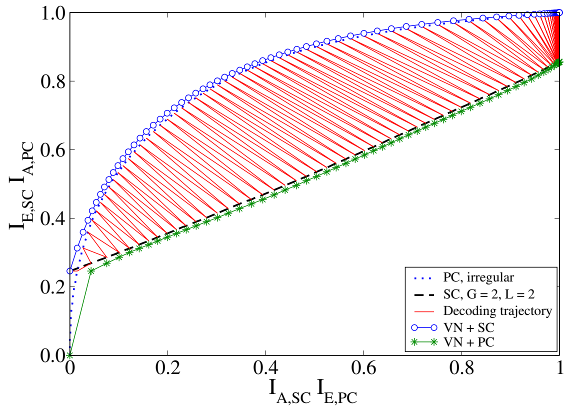
The Shannon limit for 2 bits/symbol is at about $E_b/N_0 = 5.8$ dB. We consider rate $1/G$ LDPC-coded SM-GPA with $G = 2$ and $G = 3$. For $G = 2$, we apply the following degree distribution pair with $R_p = 1/2$:

$$\begin{aligned}\lambda_p(D) &= 0.780D^2 + 0.080D^3 + 0.040D^4 + 0.020D^5 + 0.020D^7 \\ &\quad +0.020D^{10} + 0.020D^{14} + 0.020D^{16} \\ \eta_p(D) &= 0.240D^3 + 0.480D^4 + 0.080D^5 + 0.050D^8 + 0.020D^{10} \\ &\quad +0.020D^{12} + 0.030D^{14} + 0.020D^{17} + 0.040D^{20} + 0.020D^{28} \ .\end{aligned} \tag{7.77}$$

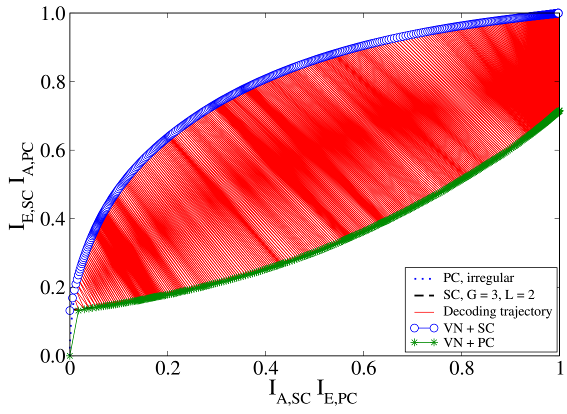
The corresponding EXIT chart is provided in Fig. 7.48(a), which predicts a decoding threshold around 7 dB. For $G = 3$, we apply the following degree distribution pair:

$$\begin{aligned}\lambda_p(D) &= 0.830D^2 + 0.010D^3 + 0.010D^5 + 0.030D^6 + 0.020D^7 \\ &\quad +0.061D^8 + 0.019D^{11} + 0.010D^{25} + 0.010D^{34} \\ \eta_p(D) &= 0.350D^3 + 0.350D^4 + 0.160D^5 + 0.040D^7 + 0.010D^{12} \\ &\quad +0.070D^{15} + 0.020D^{16} \ ,\end{aligned} \tag{7.78}$$

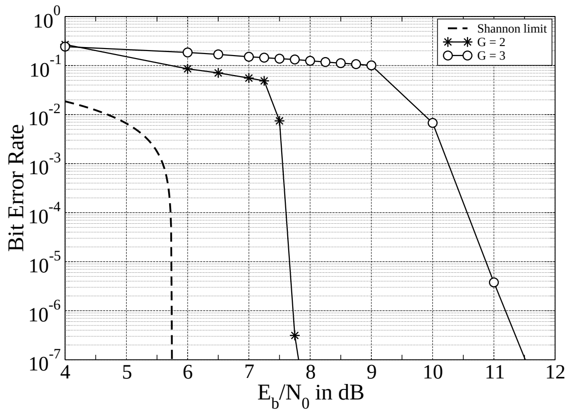
with $R_p = 1/2$. As illustrated in Fig. 7.48(b), this code design promises a near-capacity decoding threshold. Nevertheless, in reality the situation is just reversed, cf. Fig. 7.48(c). The sub-optimal 7 dB code design significantly outperforms the 6 dB near-optimum code design. One finds a similar situation in Fig. 7.38(a). Given $G = 3$, the EXIT curve of SM-GPA demapping is similar to that of SM-EPA demapping, i.e., being convex in the whole region. In this case, a good decoding convergence is only achievable if the convergence tunnel has sufficient surplus area in the left region, which explains the difference between the practically achieved performance and the theoretically predicted performance.



(a) EXIT chart, $G = 2$, $E_b/N_0 = 7$ dB.



(b) EXIT chart, $G = 3$, $E_b/N_0 = 6$ dB.



(c) BER vs. E_b/N_0.

Figure 7.48: LDPC-coded SM-GPA, $R = 1/G$, $L = 2$, $K = 40000$, RGB, 200 iterations.

7.6.4 The Case of 4 bits/symbol

As the last code design example, we target at a data rate of 4 bits/symbol over the AWGN channel. The corresponding Shannon limit is at about 15.1 dB. As well-known, approaching the channel capacity becomes very challenging for such a high data rate. For example, a rate 1/2 Turbo-coded 256-ASK transmission system is about 5 dB away from the Shannon limit, at a BER of 10^{-5}, cf. Fig. 7.49(c). Now, let us check if the situation can be improved by using SM-GPA. We first consider a free-style[4] code design, given by

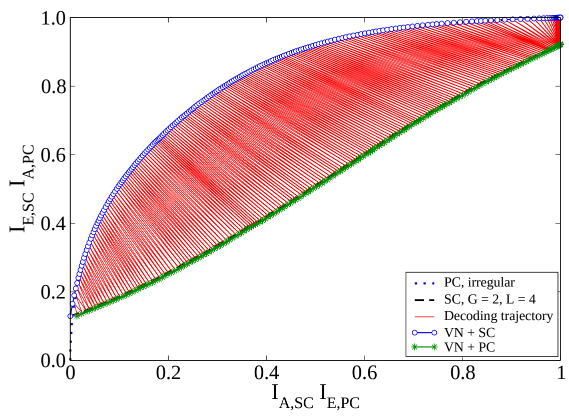
$$\begin{aligned}\lambda_p(D) &= 0.700D^2 + 0.260D^3 + 0.010D^6 + 0.010D^{12} + 0.010D^{30} + 0.010D^{34} \\ \eta_p(D) &= 0.359D^3 + 0.521D^4 + 0.020D^5 + 0.010D^8 + 0.010D^{14} + 0.020D^{16} + 0.020D^{24} \\ &\quad +0.010D^{25} + 0.020D^{26} + 0.005D^{53} + 0.004D^{120} + 0.001D^{204} \end{aligned} \tag{7.79}$$

which leads to $R_p = 1/2$. The resulting EXIT chart is as in Fig. 7.49(a), which predicts a theoretically achievable decoding threshold as 16 dB. Note that parity checks of a degree as high as 204 exists, and the parity check degree distribution is severely dispersed. Hence, such a code design is not suitable for applications with a reasonable block length. The BER result in Fig. 7.49(c) verifies this conjecture. The obtained performance happens to be even worse than that of Turbo-coded ASK. To improve the performance, we make a more realistic code design. Increasing the targeted decoding threshold to 17 dB, we obtain the following degree distribution pair:

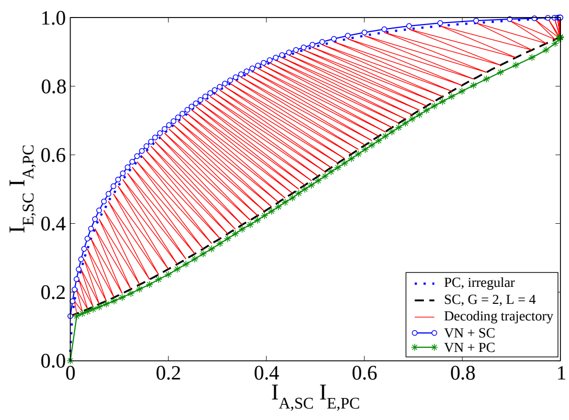
$$\begin{aligned}\lambda_p(D) &= 0.800D^2 + 0.160D^3 + 0.010D^6 + 0.010D^{12} + 0.010D^{30} + 0.010D^{44} \\ \eta_p(D) &= 0.360D^3 + 0.380D^4 + 0.160D^5 + 0.010D^7 + 0.010D^{14} \\ &\quad +0.010D^{23} + 0.040D^{24} + 0.030D^{40}\ , \end{aligned} \tag{7.80}$$

which leads to a coding rate of $R_p = 1/2$ as well. Compared to the previous code design, those very-high-degree parity checks are removed, which considerably relaxes the task of interleaver design. Meanwhile, the highest degree of variable nodes is increased, so as to achieve more surplus in the left section of the convergence tunnel. As demonstrated in Fig. 7.49(c), this new code design achieves a significant performance improvement. The resulting BER curve is about 2.8 dB away from the Shannon limit. Nevertheless, given a data rate of 4 bits per symbol per signal dimension, this performance is already satisfying. Up to this point, we may conclude that SM-GPA, as a non-bijective nonuniform mapping scheme, is not only attractive from a theoretical point of view but also competitive for practical applications.

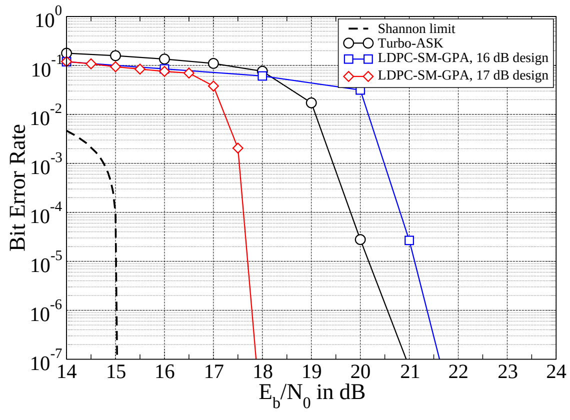
[4]By "free-style" we mean the corresponding code design does not take into account the imperfectness of practical systems.



(a) $R_p = 1/2$, $E_b/N_0 = 16$ dB.



(b) $R_p = 1/2$, $E_b/N_0 = 17$ dB.



(c) BER vs. E_b/N_0.

Figure 7.49: LDHC-GPA, $R = 1/2$, $G = 2$, $L = 4$, $K = 40000$, RGB, 200 iterations.

Chapter 8

Summary and Outlook

Mainly driven by Shannon's suggestion [3] on transmitting Gaussian signals over linear Gaussian channels, non-bijective nonuniform mapping is recently attaining more and more attention in the research community [26–33, 41]. Among various possibilities, superposition mapping (SM) deserves to be an elegant solution. By linearly superimposing binary antipodal component symbols, SM is able to deliver a symbol distribution as Gaussian as desired, without the need of active signal shaping. Superposition mapping and superposition demapping both operate in a symbol-wise manner. Hence, applying modern iteratively decodable channel codes for SM is more or less straightforward. Additionally, also due to linear superposition, low-complexity optimal SISO demapping can be implemented via the tree-based BCJR algorithm. Given these adavantages, superposition mapping has a good chance to be widely applied in future communication systems which demand a high power efficiency and a high bandwidth efficiency.

This thesis studies several open issues of SM and tries to find practical solutions for them. The first contribution is in clarifying the effects of power allocation for SM, both from the aspect of achievable power efficiency and supportable bandwidth efficiency. It is found that equal power allocation (EPA) provides an optimal power efficiency but is inefficient in supporting a high bandwidth effciency. Unequal power allocation (UPA) is efficient in supporting a high bandwidth efficiency but leads to a sub-optimal power efficiency. For this reason, a grouped power allocation (GPA) strategy is proposed, which is a hybrid of EPA and UPA. It maintains the advantages from EPA and UPA but considerably mitigates the disadvantages from both. Compared to conventional mapping schemes, e.g., ASK with Gray labelling, SM-GPA offers not only a higher potential for the achievable power efficiency but also a lower computational complexity for the demapping operation. The second contribution is in breaking the limit of 2 bits per symbol per dimension for the practically achievable bandwidth efficiency of coded SM-EPA. It is shown that an irregular repetition code is able to support a near-capacity rate for SM-EPA, when the bit load is large. To facilitate the relevant system design, a novel concept called low-density summation-check (LDSC) coding is proposed. Since SM-EPA is natively equivalent to the binary adder channel (BAC), all coding techniques proposed in this thesis are inherently applicable for the BAC. The third contribution is in designing optimal/near-optimal codes for superposition mapping. Due to the fact that a superposition demapper is in general strongly interactive with the channel decoder, a code design framework called low-density hybrid-check (LDHC) coding is proposed, which treats superposition mapping operations as a certain type of code constraints. Theoretically optimal/near-optimal channel codes have been identified for SM with various configurations, and superior performances have been achieved in practice. As an interesting result, it is found that LDHC coding is useful for conventional mapping schemes as well. The fourth contribution is in inventing various practical techniques for the code design. A randomized graph construction algorithm is proposed for generating high-performance finite-length interleavers. An EXIT emulation algorithm is proposed for an efficient optimization of degree distributions. These methods pave the road for the practical applications of superposition mapping.

A mapping from binary digits to finite-alphabet symbols takes place in almost all modern communication systems. Hence, the possible applications of SM are uncountable. Currently, orthogonal frequency-division multiplexing (OFDM) and multi-input multi-output (MIMO) transmission are two hot topics. State-of-the-art mapping formats currently used are PSK and QAM, which are uniform signaling methods. Checking the benefits of using SM with these transmission techniques deserves to be an interesting research topic both w.r.t. theory and practice. Naturally, the adoption of SM will create new aspects for the code design, which is also necessary to be studied.

By its nature, coded superposition mapping can be interpreted as a generalization of interleave-division multiplexing (IDM) and interleave-divison multiple access (IDMA) [28, 37, 39, 40, 54, 110–118]. Hence, coding techniques proposed in this thesis can be applied in IDM and IDMA systems in a straightforward way. One basically only needs to adopt multiple parallel channel encoders, which can be optimized via the code design approaches proposed in this thesis, for multiple superimposed data streams. Using mutliple parallel channel encoders offer some advantages as well as some disadvantages over using a single channel encoder for superposition type of systems. The main advantage is in the fact that cycles of length $2(2n+1)$, $n \in \mathbb{Z}^+$, are implicitly avoided among the code checks from multiple channel encoders. The main disadvantages are that some freedom in degree distribution optimization is lost and one also loses some code word length due to sequence splitting.

Bibliography

[1] J. G. Proakis, *Digital Communications*, 4th ed. McGraw-Hill, 2001.

[2] T. M. Cover and J. A. Thomas, *Elements of Information Theory*, 2nd ed. John Wiley & Sons, Inc., 2006.

[3] C. E. Shannon, “A mathematical theory of communication,” *Bell System Technical Journal*, vol. 27, pp. 379–423, 623–656, Jul., Oct., 1948.

[4] C. Berrou, A. Glavieux, and P. Thitimajshima, “Near Shannon limit error-correcting coding and decoding: Turbo-codes,” in *Proc. IEEE Int. Conf. Commun. (ICC ’93)*, vol. 2, Geneva, Switzerland, May 23–26, 1993, pp. 1064–1070.

[5] R. G. Gallager, “Low-density parity-check codes,” *IEEE Trans. Inf. Theory*, vol. 8, no. 1, pp. 21–28, Jan. 1962.

[6] D. J. C. MacKay and R. M. Neal, “Near Shannon limit performance of low-density parity-check codes,” *Electron. Lett.*, vol. 32, no. 18, pp. 1645–1646, Aug. 1996, reprinted in vol. 33, no. 6, pp. 457–458, Mar. 1997.

[7] S.-Y. Chung, J. G. David Forney, T. J. Richardson, and R. Urbanke, “On the design of low-density parity-check codes within 0.0045 db of the Shannon limit,” *IEEE Commun. Lett.*, vol. 5, no. 2, pp. 58–60, Feb. 2001.

[8] G. Ungerboeck, “Channel coding with multilevel/phase signals,” *IEEE Trans. Inf. Theory*, vol. IT-28, pp. 56–57, Jan. 1982.

[9] J. Leech and N. J. A. Sloane, “Sphere packing and error-correcting codes,” *Canad. J. Math.*, vol. 23, pp. 718–745, 1971.

[10] H. Imai and S. Hirakawa, “A new multilevel coding method using error correcting codes,” *IEEE Trans. Inf. Theory*, vol. 23, pp. 371–377, May 1977.

[11] G. Caire, G. Taricco, and E. Biglieri, “Bit-interleaved coded modulation,” *IEEE Trans. Inf. Theory*, vol. 44, no. 3, pp. 927–946, May 1998.

[12] A. G. i Fabregas, A. Martinez, and G. Caire, *Bit-Interleaved Coded Modulation.* Now Publishers Inc., 2008.

[13] G. D. Forney, Jr. and L.-F. Wei, "Multidimensional constellations–Part I: Introduction, figures of merit, and generalized cross constellations," *IEEE J. Sel. Areas Commun.*, vol. 7, pp. 877–892, Aug. 1989.

[14] A. R. Calderbank and L. H. Ozarow, "Nonequiprobable signaling on the Gaussian channel," *IEEE Trans. Inf. Theory*, vol. 36, pp. 726–740, Jul. 1990.

[15] G. D. Forney, Jr., "Trellis shaping," *IEEE Trans. Inf. Theory*, vol. 38, pp. 281–300, Mar. 1992.

[16] G. R. Lang and F. M. Longstaff, "A Leech lattice modem," *IEEE J. Sel. Areas Commun.*, vol. 7, pp. 968–973, Aug. 1989.

[17] P. Fortier, A. Ruiz, and J. M. Cioffi, "Multidimensional signal sets through the shell construction for parallel channels," *IEEE Trans. Commun.*, vol. 40, pp. 500–512, Mar. 1992.

[18] A. K. Khandani and P. Kabal, "Shaping multidimensional signal spaces–Part I: Optimum shaping, shell mapping," *IEEE Trans. Inf. Theory*, vol. 39, pp. 1799–1808, Nov. 1993.

[19] R. Laroia, N. Farvardin, and S. Tretter, "On optimal shaping of multidimensional constellations," *IEEE Trans. Inf. Theory*, vol. 40, pp. 1044–1056, Jul. 1994.

[20] G. D. Forney, Jr., L. Brown, M. V. Eyuboglu, and J. L. Moran, III, "The V.34 high-speed modem standard," *IEEE Commun. Mag.*, vol. 34, pp. 28–33, Dec. 1996.

[21] G. D. Forney, Jr. and G. Ungerboeck, "Modulation and coding for linear Gaussian channels," *IEEE Trans. Inf. Theory*, vol. 44, no. 6, pp. 2384–2415, Oct. 1998.

[22] R. F. H. Fischer, *Precoding and Signal Shaping for Digital Transmission.* New York: John Wiley & Sons, Inc., 2002.

[23] F. R. Kschischang and S. Pasupathy, "Optimal nonuniform signaling for Gaussian channels," *IEEE Trans. Inf. Theory*, vol. 39, no. 3, pp. 913–929, May 1993.

[24] E. Schrödinger, *Statistical Thermodynamics.* Cambridge: Cambridge University Press, 1962.

[25] R. K. Pathria, *Statistical Mechanics.* Elmsford, NY: Pergamon, 1972.

[26] L. Duan, B. Rimoldi, and R. Urbanke, “Approaching the AWGN channel capacity without active shaping,” in *Proc. IEEE Int. Symp. Inform. Theory (ISIT’97)*, Ulm, Germany, Jun./Jul. 1997, p. 374.

[27] X. Ma and Li Ping, “Coded modulation using superimposed binary codes,” *IEEE Trans. Inf. Theory*, vol. 50, no. 12, pp. 3331–3343, Dec. 2004.

[28] H. Schoeneich and P. A. Hoeher, “Adaptive interleave-division multiple access - A potential air interface for 4G bearer services and wireless LANs,” in *Proc. International Conference on Wireless and Optical Communications and Networks (WOCN’04)*, Muscat, Oman, Jun. 2004.

[29] T. Wo and P. A. Hoeher, “Superposition mapping with application in bit-interleaved coded modulation,” in *Proc. 8th International ITG Conference on Source and Channel Coding (SCC’10)*, Siegen, Germany, Jan. 18–21, 2010.

[30] M. Noemm, T. Wo, and P. A. Hoeher, “Multilayer APP detection for IDM,” *Electron. Lett.*, vol. 46, no. 1, pp. 96–97, Jan. 2010.

[31] F. Schreckenbach, “Iterative decoding of bit-interleaved coded modulation,” Ph.D. dissertation, Technical University of Munich, Germany, 2007.

[32] C. Schlegel and D. Truhachev, “Generalized modulation and iterative demodulation,” in *Proc. IEEE International Zurich Seminar on Communications (IZS)*, Zurich, Switzerland, Mar. 12–14, 2008.

[33] D. Zhao, A. Dauch, and T. Matsumoto, “Modulation doping for repetition coded BICM-ID with irregular degree allocation,” in *Proc. International ITG Workshop on Smart Antennas (WSA)*, Berlin, Germany, Feb. 16–18, 2009.

[34] S.-C. Chang, “Coding for a T-user multiple-access channel,” Ph.D. dissertation, University of Hawaii, USA, 1977.

[35] S.-C. Chang and E. J. Weldon, “Coding for T-user multiple-access channels,” *IEEE Trans. Inf. Theory*, vol. 25, no. 6, pp. 684–691, Nov. 1979.

[36] B. L. Hughes and A. B. Cooper III, “Nearly optimal multiuser codes for the binary adder channel,” *IEEE Trans. Inf. Theory*, vol. 42, no. 2, pp. 387–398, Mar. 1996.

[37] P. A. Hoeher and H. Schoeneich, “Interleave-division multiple access from a multiuser point of view,” in *Proc. Int. Symp. on Turbo Codes & Related Topics in conjunction with Int. ITG Conf. on Source and Channel Coding*, Munich, Germany, Apr. 2006.

[38] H. Schoeneich, “Adaptiver Interleave-Division Mehrfachzugriff (IDMA) mit Anwendung in der Mobilfunkkommunikation,” Ph.D. dissertation, University of Kiel, Germany, 2007.

[39] P. A. Hoeher and W. Xu, “Multi-layer interleave-division multiple access for 3GPP long term evolution,” in *Proc. IEEE Int. Conf. Commun. (ICC’07)*, Glasgow, Scotland, Jun. 2007.

[40] P. A. Hoeher, H. Schoeneich, and J. C. Fricke, “Multi-layer interleave-division multiple access: Theory and practice,” *Europ. Trans. Telecomms.*, vol. 19, no. 5, Aug. 2008.

[41] J. Tong, Li Ping, and X. Ma, “Superposition coded modulation with peak-power limitation,” *IEEE Trans. Inf. Theory*, vol. 55, no. 6, pp. 2562–2576, Jun. 2009.

[42] Li Ping, J. Tong, X. Yuan, and Q. Guo, “Superposition coded modulation and iterative linear MMSE detection,” *IEEE J. Sel. Areas Commun.*, vol. 27, no. 6, pp. 995–1004, Aug. 2009.

[43] T. Wo, M. Noemm, D. Hao, and P. A. Hoeher, “Iterative processing for superposition mapping,” *Hindawi Journal of Electrical and Computer Engineering – Special Issue on Iterative Signal Processing in Communications*, vol. 2010, 2010.

[44] G. E. P. Box, J. S. Hunter, and W. G. Hunter, *Statistics for Experimenters: Design, Innovation, and Discovery*, 2nd ed. John Wiley & Sons, Inc., 2005.

[45] J. L. Devore, *Probability and Statistics for Engineering and the Sciences*, 5th ed. Duxbury, 1999.

[46] A. Papoulis and S. U. Pillai, *Probability, Random Variables, and Stochastic Processes*, 4th ed. McGraw-Hill, Inc., 2006.

[47] D. Stirzaker, *Probability and Random Variables: A Beginner’s Guide.* Cambridge University Press, 1999.

[48] P. Jacquet and W. Szpankowski, “Entropy computations via analytic depoissonization,” *IEEE Trans. Inf. Theory*, vol. 45, no. 4, pp. 1072–1081, May 1999.

[49] L. R. Bahl, J. Cocke, F. Jelinek, and J. Raviv, “Optimal decoding of linear codes for minimizing symbol error rate,” *IEEE Trans. Inf. Theory*, vol. 20, no. 2, pp. 284–287, Mar. 1974.

[50] J. Stewart, *Calculus*, 6th ed. Brooks Cole, 2007.

[51] P. Robertson, P. Hoeher, and E. Villebrun, "Optimal and sub-optimal maximum a posteriori algorithms suitable for turbo-decoding," *Europ. Trans. Telecomms.*, vol. 8, no. 2, pp. 119–125, Mar./Apr., 1997.

[52] W. J. Gross and P. G. Gulak, "Simplified MAP algorithm suitable for implementation of turbo decoders," *Electron. Lett.*, vol. 34, no. 16, pp. 1577–1578, Aug. 1998.

[53] H. Schoeneich and P. A. Hoeher, "A hybrid multiple access scheme delivering reliability information," in *Proc.* 5*th International ITG Conference on Source and Channel Coding (SCC'04)*, Erlangen-Nürnberg, Germany, Jan. 2004.

[54] Li Ping, L. Liu, and K. Y. Leung, "A unified approach to multiuser detection and space-time coding with low complexity and nearly optimal performance," in *Proc.* 40*th Allerton Conference on Communication, Control, and Computing*, Monticelli, Illinois, Oct. 2002.

[55] D. Divsalar, H. Jin, and R. McEliece, "Coding theorems for Turbo-like codes," in *Proc.* 36*th Allerton Conference on Communication, Control, and Computing*, Monticelli, Illinois, USA, Sep. 1998, pp. 201–210.

[56] T. J. Richardson and R. L. Urbanke, "The capacity of LDPC codes under message-passing decoding," *IEEE Trans. Inf. Theory*, vol. 47, no. 2, pp. 599–618, Feb. 2001.

[57] T. J. Richardson, M. A. Shokrollahi, and R. Urbanke, "Design of capacity-approaching irregular LDPC codes," *IEEE Trans. Inf. Theory*, vol. 47, no. 2, pp. 619–637, Feb. 2001.

[58] W. E. Ryan and S. Lin, *Channel Codes: Classical and Modern.* Cambridge University Press, 2009.

[59] T. Tian, C. R. Jones, J. D. Villasenor, and R. D. Wesel, "Selective avoidance of cycles in irregular LDPC code construction," *IEEE Trans. Commun.*, vol. 52, no. 8, pp. 1242–1247, Aug. 2004.

[60] J. Pearl, *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference.* Morgan Kaufmann, 1988.

[61] R. D. Shachter, "Probabilistic inference and influence diagrams," *Operations Research*, vol. 36, no. 4, pp. 589–604, Aug. 1988.

[62] G. Shafer and P. Shenoy, "Probability propagation," *Ann. Mat. Art. Intell.*, vol. 2, pp. 327–352, 1990.

[63] J. Yedidia, W. Freeman, and Y. Weiss, “Generalized belief propagation,” *Advances Neural Information Processing Systems (NIPS)*, vol. 13, pp. 689–695, Dec. 2000.

[64] D. J. C. MacKay, *Information Theory, Inference, and Learning Algorithms.* Cambridge University Press, 2003.

[65] N. Wiberg, “Codes and decoding on general graphs,” Ph.D. dissertation, Linköping University, Sweden, 1996.

[66] S. M. Aji and R. J. McEliece, “The generalized distributive law,” *IEEE Trans. Inf. Theory*, vol. 46, no. 2, pp. 325–343, Mar. 2000.

[67] G. D. Forney, “Codes on graphs: Normal realizations,” *IEEE Trans. Inf. Theory*, vol. 47, pp. 520–548, Feb. 2001.

[68] F. R. Kschischang, B. J. Frey, and H.-A. Loeliger, “Factor graphs and the sum-product algorithm,” *IEEE Trans. Inf. Theory*, vol. 47, no. 2, pp. 498–519, Feb. 2001.

[69] H.-A. Loeliger, “An introduction to factor graphs,” *IEEE Signal Process. Mag.*, pp. 28–41, Jan. 2004.

[70] D. J. C. MacKay, “Good error correcting codes based on very sparse matrices,” *IEEE Trans. Inf. Theory*, vol. 45, no. 3, pp. 399–431, Mar. 1999.

[71] X.-Y. Hu, E. Eleftheriou, and D. M. Arnold, “Progressive edge-growth Tanner graphs,” in *Proc. IEEE Global Telecommunications Conf. (GLOBECOM’01)*, San Antonio, Texas, USA, Nov. 2001, pp. 995–1001.

[72] T. Tian, C. R. Jones, J. D. Villasenor, and R. D. Wesel, “Construction of irregular LDPC codes with low error floors,” in *Proc. IEEE Int. Conf. Commun. (ICC’03)*, Anchorage, Alaska, USA, May 11–15, 2003.

[73] H. Xiao and A. H. Banihashemi, “Improved progressive-edge-growth (PEG) construction of irregular LDPC codes,” *IEEE Commun. Lett.*, vol. 8, no. 12, pp. 715–717, Dec. 2004.

[74] J. Thorpe, “Low density parity check (LDPC) codes constructed from protographs,” Jet Propulsion Laboratory (JPL), IPN Progess Report 42-154, Aug. 15, 2003.

[75] Y. Kou, S. Lin, and M. Fossorier, “Low-density parity-check codes based on finite geometries: a rediscovery and new results,” *IEEE Trans. Inf. Theory*, vol. 47, no. 7, pp. 2711–2736, Nov. 2001.

[76] H. Tang, J. Xu, S. Lin, and K. A. S. Abdel-Ghaffar, "Codes on finite geometries," *IEEE Trans. Inf. Theory*, vol. 51, no. 2, pp. 572–596, Feb. 2005.

[77] L. Chen, J. Xu, I. Djurdjevic, and S. Lin, "Near-Shannon-limit quasi-cyclic low-density parity-check codes," *IEEE Trans. Commun.*, vol. 52, no. 7, pp. 1038–1042, Jul. 2004.

[78] L. Lan, L. Zeng, Y. Y. Tai, L. Chen, S. Lin, and K. Abdel-Ghaffar, "Construction of quasi-cyclic LDPC codes for AWGN and binary erasure channels: A finite field approach," *IEEE Trans. Inf. Theory*, vol. 53, no. 7, pp. 2429–2458, Jul. 2007.

[79] B. Vasic and O. Milenkovic, "Combinatorial constructions of low-density parity-check codes for iterative decoding," *IEEE Trans. Inf. Theory*, vol. 50, no. 6, pp. 1156–1176, Jun. 2004.

[80] B. Ammar, B. Honary, Y. Kou, J. Xu, and S. Lin, "Construction of low-density parity-check codes based on balanced incomplete block designs," *IEEE Trans. Inf. Theory*, vol. 50, no. 6, pp. 1257–1268, Jun. 2004.

[81] X.-Y. Hu, E. Eleftheriou, and D. M. Arnold, "Regular and irregular progressive edge-growth Tanner graphs," *IEEE Trans. Inf. Theory*, vol. 51, no. 1, pp. 386–398, Jan. 2005.

[82] C. Di, D. Proietti, I. E. Telatar, T. J. Richardson, and R. L. Urbanke, "Finite-length analysis of low-density parity-check codes on the binary erasure channel," *IEEE Trans. Inf. Theory*, vol. 48, no. 6, pp. 1570–1579, Jun. 2002.

[83] T. J. Richardson, "Error floors of LDPC codes," in *Proc. 41st Annual Allerton Conference on Communication, Control, and Computing*, Monticello, IL, Oct. 2003, pp. 1426–1435.

[84] S. ten Brink, "Convergence of iterative decoding," *Electron. Lett.*, vol. 35, no. 10, pp. 806–808, May 1999, reprinted in vol. 35, no. 13, pp. 1117–1118, Jun. 1999.

[85] ——, "Convergence behavior of iteratively decoded parallel concatenated codes," *IEEE Trans. Commun.*, vol. 49, pp. 1727–1737, Oct. 2001.

[86] S. ten Brink, G. Kramer, and A. Ashikhmin, "Design of low-density parity-check codes for modulation and detection," *IEEE Trans. Commun.*, vol. 52, no. 4, pp. 670–678, Apr. 2004.

[87] I. Land, P. A. Hoeher, and S. Gligorević, "Computation of symbol-wise mutual information in transmission systems with LogAPP decoders and application to EXIT

charts," in *Proc. 5th International ITG Conference on Source and Channel Coding (SCC'04)*, Erlangen-Nürnberg, Germany, Jan. 14–16, 2004.

[88] I. Land, P. A. Hoeher, and J. Sayir, "Bounds on information combining for the accumulator of repeat-accumulate codes without Gaussian assumption," in *Proc. IEEE Int. Symp. on Inform. Theory (ISIT'04)*, Chicago, USA, Jun. 27 – Jul. 2, 2004.

[89] I. Land, S. Huettinger, P. A. Hoeher, and J. B. Huber, "Bounds on information combining," *IEEE Trans. Inf. Theory*, vol. 51, no. 2, pp. 612–619, Feb. 2005.

[90] A. Ashikhmin, G. Kramer, and S. ten Brink, "Extrinsic information transfer functions: Model and erasure channel properties," *IEEE Trans. Inf. Theory*, vol. 50, no. 11, pp. 2657–2673, Nov. 2004.

[91] C. Measson, A. Montanari, and R. Urbanke, "Why we cannot surpass capacity: The matching condition," in *Proc. 43rd Allerton Conference on Communication, Control, and Computing*, Monticello, Illinois, USA, Sep. 28–30, 2005.

[92] K. Bhattad and K. Narayanan, "An MSE based transfer chart to analyze iterative decoding schemes," *IEEE Trans. Inf. Theory*, vol. 53, no. 1, pp. 22–38, Jan. 2007.

[93] N. T. Gaarder and J. K. Wolf, "The capacity region of a multiple-access discrete memoryless channel can increase with feedback," *IEEE Trans. Inf. Theory*, vol. 21, no. 1, pp. 100–102, 1975.

[94] T. Kasami and S. Lin, "Coding for a multiple-access channel," *IEEE Trans. Inf. Theory*, vol. 22, no. 2, pp. 129–137, Mar. 1976.

[95] ——, "Bounds on the achievable rates of block coding for a memoryless multiple-access channel," *IEEE Trans. Inf. Theory*, vol. 24, no. 2, pp. 187–197, Mar. 1978.

[96] H. C. A. V. Tilborg, "An upper bound for codes in a two-access binary erasure channel," *IEEE Trans. Inf. Theory*, vol. 24, no. 1, pp. 112–116, Jan. 1978.

[97] A. J. Viterbi, *CDMA: Principles of Spread Spectrum Communication.* Addison-Wesley, 1995.

[98] D. Tse and P. Viswanath, *Fundamentals of Wireless Communication.* Cambridge University Press, 2005.

[99] P. Wang, J. Xiao, and Li Ping, "Comparison of orthogonal and non-orthogonal approaches to future wireless cellular systems," *IEEE Veh. Technol. Mag.*, vol. 1, no. 3, pp. 4–11, Sep. 2006, correction in vol. 1, no. 4, pp. 42–42, Dec. 2006.

[100] A. Sanderovich, M. Peleg, and S. Shamai, "LDPC coded MIMO multiple access with iterative joint decoding," *IEEE Trans. Inf. Theory*, vol. 51, no. 4, pp. 1437–1450, Apr. 2005.

[101] X. Wang, G. Yue, and K. R. Narayanan, "Optimization of LDPC-coded Turbo CDMA systems," *IEEE Trans. Signal Process.*, vol. 53, no. 4, pp. 1500–1510, Apr. 2005.

[102] A. Chakrabarti, A. de Baynast, A. Sabharwal, and B. Aazhang, "Low density parity check codes for the relay channel," *IEEE J. Sel. Areas Commun.*, vol. 25, no. 2, pp. 280–291, Feb. 2007.

[103] M. Mattas and P. R. J. Östergård, "A new bound for the zero-error capacity region of the two-user binary adder channel," *IEEE Trans. Inf. Theory*, vol. 51, no. 9, pp. 3289–3291, Sep. 2005.

[104] M. Bossert, *Channel Coding for Telecommunications.* John Wiley & Sons, Ltd., 1999.

[105] E. Sharon, A. Ashikhmin, and S. Litsyn, "EXIT functions for the Gaussian channel," in *Proc. 40th Annu. Allerton Conf. on Communication, Control, Computers*, Allerton, IL, Oct. 2003, pp. 972–981.

[106] ——, "EXIT functions for binary input memoryless symmetric channels," *IEEE Trans. Commun.*, vol. 54, no. 7, pp. 1207–1214, Jul. 2006.

[107] R. L. Urbanke and Q. Li, "The zero-error capacity region of the 2-user synchronous BAC is strictly smaller than its Shannon capacity region," in *Proc. IEEE Information Theory Workshop*, Killarney, Ireland, Jun. 22–26, 1998.

[108] A. Amraoui, S. Dusad, and R. L. Urbanke, "Achieving general points in the 2-user Gaussian MAC without time-sharing or rate-splitting by means of iterative coding," in *Proc. IEEE International Symposium on Information Theory*, Lausanne, Switzerland, Jun. 30 – Jul. 5, 2002.

[109] A. Roumy and D. Declercq, "Characterization and optimization of LDPC codes for the 2-user Gaussian multiple access channel," *EURASIP J. Wirel. Commun. Netw.*, vol. 2007, 2007.

[110] H. A. Cirpan and M. K. Tsatsanis, "Chip interleaving in direct sequence CDMA systems," in *Proc. IEEE International Conference on Acoustics, Speech, and Signal Processing*, Munich, Germany, Apr. 21–24, 1997, pp. 3877–3880.

[111] P. Frenger, P. Orten, and T. Ottosson, "Code-spread CDMA using maximum free distance low-rate convolutional codes," *IEEE Trans. Commun.*, vol. 48, no. 1, pp. 135–144, Jan. 2000.

[112] S. Brück, U. Sorger, S. Gligorevic, and N. Stolte, "Interleaving for outer convolutional codes in DS-CDMA systems," *IEEE Trans. Commun.*, vol. 48, no. 7, pp. 1100–1107, Jul. 2000.

[113] F. Brännström, T. M. Aulin, and L. K. Rasmussen, "Iterative detectors for trellis-code multiple-access," *IEEE Trans. Commun.*, vol. 50, no. 9, pp. 1478–1485, Sep. 2002.

[114] R. H. Mahadevappa and J. G. Proakis, "Mitigating multiple access interference and intersymbol interference in uncoded CDMA systems with chip-level interleaving," *IEEE Trans. Wireless Commun.*, vol. 1, no. 4, pp. 781–792, Oct. 2002.

[115] H. Schoeneich and P. A. Hoeher, "A hybrid multiple access scheme approaching single user performance," in *Proc. Sixth Baiona Workshop on Signal Processing in Communications*, Baiona, Spain, Sep. 2003, pp. 163–168.

[116] Li Ping, "Interleave-division multiple access and chip-by-chip iterative multi-user detection," *IEEE Commun. Mag.*, vol. 43, no. 6, pp. S19–S23, Jun. 2005.

[117] Li Ping, L. Liu, K. Wu, and W. K. Leung, "Interleave-division multiple-access," *IEEE Trans. Wireless Commun.*, vol. 5, no. 4, pp. 938–947, Apr. 2006.

[118] H. Schoeneich and P. A. Hoeher, "Iterative pilot-layer aided channel estimation with emphasis on interleave-division multiple access systems," *EURASIP J. Applied Signal Process.*, vol. 2006, pp. 1–15, 2006.

[119] T. M. Apostol, *Mathematical Analysis*, 2nd ed. Addison Wesley, 1974.

[120] G. E. Shilov, B. L. Gurevich, and R. A. Silverman, *Integral, Measure, and Derivative: A Unified Approach.* New York: Dover Publications, 1977.

Appendix A

Acronyms and Abbreviations

A.1 Acronyms

AEP	Asymptotic equipartition property
APP	A posteriori probability
ASK	Amplitude-shift keying
AWGN	Additive white Gaussian noise
BAC	Binary adder channel
BCJR	Bahl-Cocke-Jelinek-Raviv algorithm
BER	Bit error rate
BICM	Bit-interleaved coded modulation
BICM-SM	Bit-interleaved coded modulation with superposition mapping
BPSK	Binary phase-shift keying
CDM	Code-division multiplexing
CDMA	Code-division multiple access
DEM	Demapping/Demodulation
EBL	Effective bit load
EMD	Extrinsic message degree
EPA	Equal power allocation
EXIT	Extrinsic information transfer
FPM	Fraction of positive messages

GA	Gaussian approximation
GF	Galois field
GPA	Grouped power allocation
ICR	Information-to-complexity ratio
ID	Iterative decoding
IDM	Interleave-division multiplexing
IDMA	Interleave-division multiple access
ISI	Inter-symbol interference
ITU	International Telecommunication Union
LDHC	Low-density hybrid-check
LLR	Log-likelihood ratio
LDPC	Low-density parity-check
LDSC	Low-density summation-check
MAP	Maximum-a-posteriori
MI	Mutual information
MIMO	Multi-input multi-output
MLD	Maximum-likelihood decoding
ML-IDMA	Multi-layer interleave-division multiple access
MLSE	Maximum-likelihood sequence estimator
OFDM	Orthogonal frequency-division multiplexing
PC	Parity check
PDF	Probability density function
PEG	Progressive edge growth
PMF	Probability mass function
PSK	Phase-shift keying
PSM	Phase-shifted superposition mapping
QAM	Quadrature amplitude modulation
REP	Repetition
SC	Summation check

SCE	Summation check extrinsic message degree
SCR	Scrambling
SD	Superposition demapping
SIC	Successive interference cancellation
SISO	Soft-input soft-output
SM	Superposition mapping/modulation
SM-EPA	Superposition mapping with equal power allocation
SM-GPA	Superposition mapping with grouped power allocation
SM-UPA	Superposition mapping with unequal power allocation
SNR	Signal-to-noise ratio
UPA	Unequal power allocation
VA	Viterbi algorithm
VN	Variable node

A.2 Abbreviations

et al.	et alii (= and others)
etc.	et cetra (= and so on)
i.i.d.	independent and identically distributed
i.i.f.	if and only if
i.u.d.	independent and uniformly distributed
vs.	versus
w.r.t.	with respect to

Appendix B

Mathematical Notations

b_n	nth bit
c_n	nth chip
C	Channel capacity
E_b	Energy per info bit
E_s	Energy per symbol
P_b	Bit error probability
$H(\cdot)$	Entropy
$h(\cdot)$	Differential entropy
$I(\cdot,\cdot)$	Mutual information
N	Bit load
$\mathcal{X}$	Symbol alphabet
$\Pr\{\cdot\}$	Probability of an event
$p(\cdot)$	Probability density function
$P(\cdot)$	Probability mass function
$\mathbb{F}_2$	Galois field GF(2)
$\mathbb{R}$	Field of real numbers
ϕ	Mapping rule
$\mathcal{N}(\mu,\sigma^2)$	Normal distribution with mean μ and variance σ^2
$\mathcal{B}(n,p)$	Binomial distribution of n independent Bernoulli(p) experiments
$d_{\mathrm{H}}(\mathbf{x},\mathbf{y})$	Hamming distance between $\mathbf{x}$ and $\mathbf{y}$
$d_{\mathrm{E}}(\mathbf{x},\mathbf{y})$	Euclidean distance between $\mathbf{x}$ and $\mathbf{y}$
$\lambda_s(D)$	Variable node degree distribution of an LDSC code
$\eta_s(D)$	Check node degree distribution of an LDSC code
$\lambda_p(D)$	Variable node degree distribution of an LDPC code
$\eta_p(D)$	Check node degree distribution of an LDPC code

Appendix C

Mathematical Definitions & Derivations

C.1 Definition of LLRs

Define the a posteriori LLR of the nth code bit as

$$LLR_a(b_n) \doteq \ln \frac{P(b_n = 0|y)}{P(b_n = 1|y)} \ , \tag{C.1}$$

and the extrinsic LLR of the nth code bit as

$$LLR_e(b_n) \doteq \ln \frac{p(y|b_n = 0)}{p(y|b_n = 1)} \ , \tag{C.2}$$

which is extrinsic w.r.t. the a priori information from the decoder. Correspondingly, define the intrinsic LLR of the nth code bit as

$$LLR_i(b_n) \doteq \ln \frac{P(b_n = 0)}{P(b_n = 1)} \ . \tag{C.3}$$

Using Bayes' rule, one attains the following relationship:

$$LLR_a(b_n) = LLR_e(b_n) + LLR_i(b_n) \ . \tag{C.4}$$

According to the principle of Bayesian inference, the messages being passed over a factor graph should always be extrinsic. In this thesis, by the notation LLR without any subscript, we always mean the extrinsic log-likelihood ratio.

C.2 Entropy of Gaussian Variable

The entropy formula for Gaussian distribution is one of the well-known results from Shannon's original work [3]. The corresponding mathematical derivation is given below.

Let u be a Gaussian variable with probability density function:

$$p(u) = \frac{1}{\sqrt{2\pi\sigma_u^2}} \ \exp\left(-\frac{u^2}{2\sigma_u^2}\right) \ .$$

To derive the entropy formula, it is easier to start by using natural logarithm:

$$\begin{aligned} h(u) &= -\int p(u)\ln p(u)\mathrm{d}u \\ &= -\int p(u)(-\frac{1}{2}\ln 2\pi\sigma_u^2 - \frac{u^2}{2\sigma_u^2})\mathrm{d}u \\ &= \frac{1}{2}\ln 2\pi\sigma_u^2 \int p(u)\mathrm{d}u + \frac{1}{2\sigma_u^2}\int u^2 p(u)\mathrm{d}u \\ &= \frac{1}{2}\ln 2\pi\sigma_u^2 + \frac{1}{2\sigma_u^2}\cdot\sigma_u^2 \\ &= \frac{1}{2}\ln 2\pi\sigma_u^2 + \frac{1}{2} \\ &= \frac{1}{2}\ln 2\pi\sigma_u^2 + \frac{1}{2}\ln e \\ &= \frac{1}{2}\ln 2\pi e\sigma_u^2 \qquad \text{nats.} \end{aligned}$$

Changing the base of the logarithm, we obtain

$$h(u) = \frac{1}{2}\log 2\pi e\sigma_u^2 \qquad \text{bits.} \tag{C.5}$$

This formula is a stepping-stone for the derivation of Gaussian channel capacity.

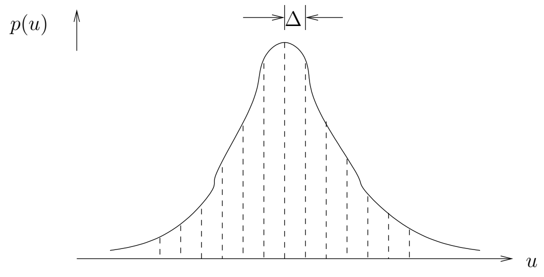


Figure C.1: Quantization of a Gaussian variable.

C.3 Quantization of Gaussian Variable

In classical information theory, the quantization of continuous variable is often utilized to build up a link between differential entropy and discrete entropy [2]. For superposition mapping, the quantization of Gaussian variable is of special interest, as it helps to obtain a nice approximation for the calculation of SM symbol entropy. In the following, we briefly elaborate the relevant mathematical derivation.

Consider a Gaussian variable u with zero mean and variance σ_u^2. We divide the range of u into bins of size Δ, as illustrated in Fig. C.1. Collecting the center of each bin into a discrete alphabet $\mathcal{X}$, one may define a discrete variable x with distribution

$$P(x) = \int_{x-\Delta/2}^{x+\Delta/2} p(u)\ \mathrm{d}u = \int_{x-\Delta/2}^{x+\Delta/2} \frac{1}{\sqrt{2\pi\sigma_u^2}}\ e^{-u^2/(2\sigma_u^2)}\ \mathrm{d}u\ , \quad x \in \mathcal{X}\ ,$$

which gives a linear quantization of Gaussian variable u. Clearly, the following equality

$$\sum_{x\in\mathcal{X}} P(x) = \sum_{x\in\mathcal{X}} \int_{x-\Delta/2}^{x+\Delta/2} p(u)\ \mathrm{d}u = \int_{-\infty}^{+\infty} p(u)\ \mathrm{d}u = 1$$

holds always. Hence, $P(x)$ is a valid probability mass function (PMF). According to the definition of discrete entropy, we have

$$\begin{aligned} H(x) &= -\sum_{x\in\mathcal{X}} P(x) \log P(x) \\ &= -\sum_{x\in\mathcal{X}} \left(\int_{x-\Delta/2}^{x+\Delta/2} p(u)\ \mathrm{d}u \right) \log \left(\int_{x-\Delta/2}^{x+\Delta/2} p(u)\ \mathrm{d}u \right)\ , \end{aligned} \tag{C.6}$$

which is difficult to be directly evaluated. Note that the Gaussian function $p(u)$ is continuous and differentiable over the whole support. Following the mean value theorem, for each quantization bin there exists a value u' such that

$$p(u')\Delta = \int_{x-\Delta/2}^{x+\Delta/2} p(u)\ \mathrm{d}u\ , \quad x \in \mathcal{X}\ . \tag{C.7}$$

Substituting (C.7) into (C.6), we obtain

$$\begin{aligned} H(x) &= -\sum_{u'} p(u')\Delta \log(p(u')\Delta) \\ &= -\sum_{u'} p(u')\Delta \log p(u') - \left(\sum_{u'} p(u')\Delta\right) \cdot \log \Delta \\ &= -\sum_{u'} p(u') \log p(u')\Delta - \log \Delta \; . \end{aligned} \tag{C.8}$$

where the last equality comes from the fact that $\sum_{u'} p(u')\Delta = 1$. It is easy to prove that $p(u) \log p(u)$ is Riemann integrable [119, 120], i.e.,

$$-\sum_{u'} p(u') \log p(u')\Delta = -\int p(u) \log p(u) \mathrm{d}u \qquad \text{for } \Delta \to 0 \; . \tag{C.9}$$

Sequentially, for $\Delta \to 0$ we have

$$\begin{aligned} H(x) &= -\int p(u) \log p(u) \mathrm{d}u - \log \Delta \\ &= \frac{1}{2} \log(2\pi e \sigma_u^2) - \log \Delta \\ &= \frac{1}{2} \log(2\pi e \sigma_u^2/\Delta^2) \end{aligned} \tag{C.10}$$

where the second equality follows from the property of Gaussian distribution, cf. App. C.2. As long as the quatization step Δ is not so large, we may safely use the approximation

$$H(x) \approx \frac{1}{2} \log(2\pi e \sigma_u^2/\Delta^2) \tag{C.11}$$

for some perceptual analysis, e.g., the entropy calcuation of SM symbols. Worthwhile to be noted, the above approximation might also be written as

$$H(x) \approx h(\mathcal{N}(0, \sigma_u^2/\Delta^2) \; , \tag{C.12}$$

which tells that the entropy of the quantization of a Gaussian distribution is approximately the entropy of another Gaussian distribution with the variance being $1/\Delta^2$ times that of the original one. Naturally, as one reduces the quantization step Δ, the discrete entropy $H(x)$ increases. This property is used in Section. 3.5 to improve the supportable bandwidth efficiency of SM while maintaining an optimal achievable power efficiency.

Appendix D

Own Publications Related to the Thesis

[1] P. A. Hoeher and T. Wo, “Superposition modulation: Myths and facts”, *IEEE Commun. Mag.*, vol. 49, no. 12, pp. 110–116, Dec. 2011.

[2] T. Wo, M. Noemm, D. Hao, and P. A. Hoeher, “Iterative processing for superposition mapping”, *Hindawi Journal of Electrical and Computer Engineering – Special Issue on Iterative Signal Processing in Communications*, vol. 2010, 2010.

[3] T. Wo and P. A. Hoeher, “Low-complexity Gaussian detection for MIMO systems”, *Hindawi Journal of Electrical and Computer Engineering – Special Issue on Iterative Signal Processing in Communications*, vol. 2010, 2010.

[4] M. Noemm, T. Wo, and P. A. Hoeher, “Multilayer APP detection for IDM”, *Electron. Lett.*, vol. 46, no. 1, pp. 96–97, Jan. 2010.

[5] Z. Shi, T. Wo, P. A. Hoeher, and G. Auer, “Graph-based soft iterative receiver for higher-order modulation”, in *Proc. IEEE 12th International Conference on Communication Technology (ICCT)*, Nanjing, China, Nov. 2010.

[6] T. Wo and P. A. Hoeher, “A universal coding approach for superposition mapping”, in *Proc. IEEE 6th International Symposium on Turbo Codes & Iterative Information Processing (ISTC)*, Brest, France, Sep. 2010.

[7] Z. Shi, T. Wo, and P. A. Hoeher, “Superposition mapping with adaptive bit loading for BICM-OFDM systems”, in *Proc. IEEE 6th International Symposium on Turbo Codes & Iterative Information Processing (ISTC)*, Brest, France, Sep. 2010.

[8] T. Wo and P. A. Hoeher, “Superposition mapping with application in bit-interleaved coded modulation”, in *Proc. IEEE 8th International ITG Conference on Source and Channel Coding (SCC)*, Siegen, Germany, Jan. 2010.

[9] T. Wo, C. Liu, and P. A. Hoeher, “Graph-based soft channel and data estimation for MIMO systems with asymmetric LDPC codes”, in *Proc. IEEE International Conference on Communications (ICC)*, Beijing, China, May 2008.

[10] T. Wo, C. Liu, and P. A. Hoeher, “Graph-based iterative Gaussian detection with soft channel estimation for MIMO systems”, in *Proc. 7th International ITG Conference on Source and Channel Coding (SCC)*, Ulm, Germany, Jan. 2008.

[11] T. Wo and P. A. Hoeher, “A simple iterative Gaussian detector for severely delay-spread MIMO channels”, in *Proc. IEEE International Conference on Communications (ICC)*, Glasgow, Scotland, Jun. 2007.

[12] A. Scherb, K.-D. Kammeyer, T. Wo, and P. A. Hoeher, “Blind equalization of frequency selective MIMO systems via statistical and trellis-based methods”, in *Proc. 40th Asilomar Conference on Signals, Systems, and Computers (ACSSC)*, Asilomar, USA, Oct. 2006.

[13] T. Wo, J. Ch. Fricke, and P. A. Hoeher, “A graph-based iterative Gaussian detector for frequency-selective MIMO channels”, in *Proc. IEEE Information Theory Workshop (ITW)*, Chengdu, China, Oct. 2006.

[14] T. Wo, P. A. Hoeher, A. Scherb, and K.-D. Kammeyer, “Performance analysis of maximum-likelihood semiblind estimation of MIMO channels”, in *Proc. IEEE 63rd Vehicular Technology Conference (VTC)*, Melbourne, Australia, May 2006.

[15] T. Wo, A. Scherb, P. A. Hoeher, and K.-D. Kammeyer, “Analysis of semiblind channel estimation for FIR-MIMO systems”, in *Proc. 4th International Symposium on Turbo Codes & Related Topics (ISTC) in conjunction with 6th International ITG Conference on Source and Channel Coding (SCC)*, Munich, Germany, Apr. 2006.

[16] T. Wo and P. A. Hoeher, “Semi-blind channel estimation for frequency-selective MIMO systems”, in *Proc. 14th IST Mobile & Wireless Communications Summit*, Dresden, Germany, Jun. 2005.