UNIVERSITY OF SOUTHAMPTON
Faculty of Engineering and Applied Science
School of Electronics and Computer Science

Low Density Parity Check Coding

by

Feng Guo

A thesis submitted in partial fulfilment of the requirements for the award of Doctor of Philosophy at the University of Southampton

January 2005

SUPERVISOR: Prof. Lajos Hanzo
FREng, FIEEE, DSc, Dipl Ing, MSc, Ph.D

This thesis is dedicated to:

My mum A. S. Xu, my dad S. R. Guo and grandma C. Z. Wu in Shanghai for their love and care since my birth.

UNIVERSITY OF SOUTHAMPTON

ABSTRACT

FACULTY OF ENGINEERING AND APPLIED SCIENCE
SCHOOL OF ELECTRONICS AND COMPUTER SCIENCE

Doctor of Philosophy

Low Density Parity Check Coding

by Feng Guo

This thesis explores the properties of the family of Low Density Parity Check Codes (LDPCC). In addition to Gallager's original binary regular LDPCCs, the class of irregular LDPCCs and non-binary LDPCCs developed from the binary regular LDPCC is also studied. Further, a novel reliability ratio based bit flipping decoding algorithm is proposed for providing a low-complexity decoding solution. A novel three-layer iterative decoding scheme is also designed for the symbol-based joint decoding of non-binary LDPCC aided space-time coding operating at a low complexity. Furthermore, classic binary LDPCCs have been concatenated with both space-time coding and source coding schemes for the sake of enhancing the achievable system performance.

Gallager's binary regular LDPCCs achieve a near-capacity performance, while maintaining a relatively low decoding complexity. Furthermore, Gallager suggested that the family of LDPCCs exhibits good distant properties, provided that certain Parity Check Matrix (PCM) construction constraints are satisfied. A novel LDPC Block Coded Modulation (LDPC-BCM) scheme was proposed, which was shown to outperform the Turbo Trellis Coded Modulation (TTCM) based benchmark scheme by 1.5 dB at a BER of 10^{-5}, when communicating over an uncorrelated Rayleigh fading channel using QPSK modulation, while maintaining an effective throughput of 1 bit per symbol. Rather than assigning each message node or check node in the PCM a constant weight, the class of irregular LDPCCs constructs the PCM using a pre-determined density profile, i.e. provides a non-uniform weight distribution. This approach may introduce weight-two message nodes into the PCM, which can result in less attractive distance properties for the code, potentially resulting in an error floor. Hence, a technique referred to as Yang's method is invoked for reducing this potential error floor, while still benefiting from the irregular construction of the PCM. We found that the employment of Richardson's PCM construction approach is more feasible applications, which are not delay sensitive, while Yang's approach is more attractive in applications having a moderate coded block length, especially at high coding rates.

Davey and MacKay further developed the family of LDPCCs in order to create non-binary LDPC codes. The advantage of non-binary LDPCCs is that they may achieve a reduced probability of incurring short cycles in the PCM in comparison to the LDPCCs having an equivalent binary PCM. However, non-binary LDPCCs do not always perform better than binary LDPCCs. The choice of the decoding field, column weight and also the coding rate will affect the attainable performance of non-binary LDPCCs.

We have applied these non-binary LDPCCs to design a purely symbol-based joint decoding and demodulation aided transmit diversity scheme, which is capable of exploiting the soft information generated by the LDPC decoder by re-evaluating the soft channel output provided by the demodulator. Upon employing non-binary LDPCCs defined over the Galois field $\mathbf{GF}(q)$ chosen according to the specific modulation scheme used, the proposed non-binary-LDPCC aided transmit diversity scheme achieved a coding gain of nearly 2dB at a BER

of 10^{-5} in comparison to the bit-based binary-LDPCC aided benchmark scheme, when communicating over an uncorrelated Rayleigh fading channel. Furthermore, as a benefit of the low-complexity FFT-based decoding approach of the non-binary LDPCC, upon invoking the proposed symbol-based LDPC-aided space-time coding scheme, we benefit from both an improved BER performance and a reduced decoding complexity.

Variable Length Coding (VLC) schemes are widely used for the sake of bit rate reduction during the source coding stage. Two trellis-based VLC MAP decoding schemes, namely the symbol-based and the bit-based trellis decoding algorithms were proposed by Bauer and Hagenauer. We have applied this soft output decoding algorithm to provide a solution for iterative joint source and channel decoding. Various VLC coding schemes are investigated and these different VLC coding schemes are jointly decoded in conjunction with various channel codes, namely by iteratively exchanging information with a recursive systematic convolutional code, a turbo code and an LDPC code. It is demonstrated that a source code having a higher free distance is capable of achieving a higher iteration gain.

Various bit-flipping based decoding algorithms are investigated because they provide low-complexity LDPC decoding, and a novel reliability-ratio based bit-flipping algorithm is proposed. Generally, the bit-flipping algorithms are capable of operating at a significantly lower complexity in comparison to the well-known sum-product algorithm, although at the cost of an inferior error correction capability. The proposed novel reliability-ratio based bit-flipping algorithm was shown to be superior in comparison to the family of previously proposed bit-flipping algorithms at no extra complexity.

Acknowledgements

I would like to thank my supervisor Prof. Lajos Hanzo for his continual help, enthusiasm and encouragement during my work. I am also grateful to all my colleagues in the Communications Group, both past and present, for their friendship and their help.

Finally I must thank my parents for all their support during my studies.

List of Publications

1. **F. Guo, S. X. Ng and L. Hanzo**, "LDPC assisted block coded modulation for transmission over Rayleigh fading channels", Proceedings of IEEE Vehicular Technology Conference Spring, Jeju, Korea, April 22-25, 2003, Volume 3, pp. 1867 - 1871.

2. **F. Guo and L. Hanzo** "Low complexity non-binary LDPC and modulation schemes communicating over MIMO channels", Proceedings of IEEE Vehicular Technology Conference Fall, Los Angeles, USA, Sept. 26-29, 2004

3. **F. Guo and L. Hanzo** "Reliability ratio based weighted bit-flipping decoding for low-density parity-check codes", IEE Electronics Letters, Volume 40, Issue. 21, 14 Oct. 2004, pp. 1356-1357.

4. **F. Guo and L. Hanzo** "Reliability ratio based weighted bit-flipping decoding for LDPC codes", accepted by Vehicular Technology Conference Spring 2005, Stockholm, Sweden.

5. **F. Guo and L. Hanzo** " On the design of three-stage serially concatenated turbo transceivers", submitted to Vehicular Technology Lettes.

6. **O. Alamri, F. Guo, M. Jiang and L. Hanzo** " Turbo detection of symbol-based non-binary LDPC-coded space-time signals using sphere packing modulation", submitted to Vehicular Technology Conference Fall 2005, Dallas, USA.

7. **J. Y. Chung, M. Y. Alias, F. Guo and L. Hanzo**, "LDPC and turbo coding assisted space-time block coded OFDM for H.26L", Proceedings of PIMRC' 2003, Beijing, China, September 2003, pp. 2702-2706.

8. **J. Y. Chung, F. Guo, S. X. Ng and L. Hanzo**, "Multi-mode joint-detection CDMA/H.26L based wireless video telephony", Proceedings of IEEE Vehicular Technology Conference Fall, Orlando, Florida, USA, October 2003, Volume 3 , pp. 6-9.

9. **M. Y. Alias; F. Guo; S. X. Ng; T. H. Liew and L. Hanzo**, "LDPC and turbo coding assisted space-time block coded OFDM", Proceedings of IEEE Vehicular Technology Conference Spring, Jeju, Korea, April 22-25, 2003, Volume 3, pp. 2309-2313

10. **S. X. Ng; F. Guo; J. Wang; L.-L. Yang and L. Hanzo**, " Joint source-coding, channel-coding and modulation schemes for AWGN and Rayleigh fading channels", Electronics Letters ,Volume: 39 , Issue: 17 , 21 Aug. 2003 Pages:1259- 1261

11. **S. X. Ng; F. Guo; J. Wang; L.-L. Yang and L. Hanzo**, " Jointly optimised iterative source-coding, channel-coding and modulation for transmission over wireless channels", Proceedings of IEEE Vehicular Technology Conference Spring, Milan, Italy, 17-19 May 2004, Volumne 1, pp. 313 - 317

12. **H. Wei, B. L. Yeap, F. Guo and L. Hanzo**, " Blind per-survivor processing-based multiuser detection for channel-coded multicarrier DS-CDMA systems", Proceedings of IEEE Vehicular Technology Conference Spring, Milan, Italy, 17-19 May 2004, Volumne 3, pp. 1376 - 1380

13. **S. X. Ng, J. Y. Chung, F. Guo and L. Hanzo**, " Turbo-detection aided serially concatenated MPEG-4/TCM videophone transceiver", Proceedings of IEEE Vehicular Technology Conference Fall, Los Angeles, USA, Sept. 26-29, 2004

14. **S. X. Ng, F. Guo and L. Hanzo**, " Iterative detection of diagonal block space time trellis codes, TCM and reversible variable length codes for transmission over Rayleigh fading channels", Proceedings of IEEE Vehicular Technology Conference Fall, Los Angeles, USA, Sept. 26-29, 2004

Contents

Chapter 1

Introduction

1.1 Historic Background

In 1963, Gallager [1] [2] devised the family of Low Density Parity Check Codes (LDPCC) during his Ph.D study at MIT. At this early phase of the evolution of channel coding, this scheme made little impact on the research of the channel coding community, despite its impressive performance, which was unprecedented prior to the invention of turbo coding [3]. This modest interest in LDPCCs was a consequence of its high storage requirements and complexity in the light of the state-of-the-art in 1963. Following their conception, LDPCCs remained dormant for a decade or so. The complexity of LDPCCs was evaluated by Zyablov and Pinsker in 1975 [4], while in [5] Tanner suggested the employment of a recursive approach for the construction of LDPC codes and presented a graph representation of the LDPCC's parity check matrix. Sipser and Spielman [6] presented the LDPCC's parity check matrix using expander graphs.

However, during the 1990s, the channel coding community's interest in LDPCCs was revived. As summarised in Table 1.1, LDPC codes have become an extremely hot topic in the wireless communication research community. MacKay and Neal [7] [8] experimented with LDPCCs having a high blocklength and illustrated that LDPCCs are capable of outperforming turbo codes, when communicating over AWGN channels. Motivated by the outstanding performance of LDPCCs, they have been studied in many different contexts. The *density evolution* (DE) algorithm was proposed by Richardson *et al.* [9] for calculating the asymptotic performance of LDPCCs transmitted over AWGN channels. Later Chung *et al.* [10] [11] simplified the *density evolution* algorithm. The density evolution algorithm is now widely recognised as an accurate method of predicting the asymptotic performance of the LDPCCs transmitted over AWGN channels, and it has been used by numerous authors, such as Fossorier [12], Chen *et al.* [13], Narayanaswami *et al.* [14], Anastasopoulos [15] and Kumar *et al.* [16]. The performance bounds of LDPCCs rate were studied by Burshtein *et al.* in [17] [18] [19]. The family of LDPC codes has also been utilised in a variety of different systems, such as OFDM [20–25], MIMO and space-time coding schemes [26, 27], in the context of the binary erasure channel [28, 29], the partial response channel [16, 30, 31], as well as a range of other channel models [32–34]. Bandwidth efficient coded modulation schemes using LDPCCs were studied in [35–37]. Furthermore, LDPC codes

have also been widely used in diverse applications such as magnetic recording [38–42] by Song *et al.*, in image transmission [43, 44] by Zhang *et al.* and in optical data storage [45]. Many algorithms have been proposed for specifically designing LDPCCs for hardware implementation, such as the techniques advocated by Zhang *et al.* [46–48], Rupp *et al.* [49], Hocevar [50], Thorpe [51], Lu *et al.* [52] and Shanbhag *et al.* [53].

In addition to the performance evaluation studies and application-oriented aspects of LDPC codes, researchers endeavoured to improve the stand-alone performance of LDPCCs by modifying the decoding algorithms and/or optimising the structure of the parity check matrix. Non-binary LDPCCs were proposed by Davey *et al.* [54–56] that under certain conditions are capable of outperforming their binary counterpart. A low-complexity decoding algorithm was proposed for non-binary LDPCCs by Barnault *et al.* [57]. The family of non-binary LDPCCs was also applied by Song *et al.* [38], Nakamura *et al.* [58] and Li *et al* [59]. Upon imposing an irregular construction on the LDPCC's parity check matrix [60–66] for the sake of improving their performance, they become capable of approaching the Shannon limit [67]. Various ways of constructing the irregular LDPCC's parity check matrix were proposed in [65, 66, 68, 69]. Increasing the length of the shortest cycles within the LDPC code's parity check matrix is another technique of improving their performance, which has the potential of lowering their error floor. Moura *et al.* [70–73], Lin *et al.* [74] and Williamson *et al.* [75] proposed various of ways of constructing the PCM in an attempt to remove the short cycles. Lentmaier proposed the class of *Generalised Low Density Parity Check Codes* in [76], which was facilitated by replacing the rows in the LDPCC's parity check matrix by a Hamming code. This technique also attracted the interest of Zhang *et al.* [77] [78], Hirst *et al.* [79] [80] and Boutros *et al.* [81].

Apart from increasing the error correction capability of LDPCCs, other researchers endeavoured to reduce the encoding and decoding complexity of LDPC codes. Although Mackay and Neal [8] have demonstrated that upon randomly constructing the LDPCC's PCM, a near capacity performance can be achieved, it has also been observed that upon constructing the PCM following different rules, a similar error correction capability can be achieved using a reduced-complexity encoding process that can be implemented using shift registers. For example, the analytical approach of finite geometry was utilised by Kou *et al.* [82–87] for constructing the LDPCC's PCM. This finite geometry based technique was also used by Pados [88] as well as Vasic *et al.* [89, 90]. Honary *et al.* [91, 92] utilised the Balanced Incomplete Block Design (BIBD) technique for constructing the LDPC code's PCMs in a (quasi)-cyclic way. A range of other analytical techniques of constructing the LDPCC's PCM have been contrived, such as for example the schemes proposed by Vontobel [93], Ahn [94] and Okamura [95]. All these schemes have been shown to be capable of attaining a performance which is as good as that of randomly constructed LDPC codes, whilst the encoding complexity may be significantly reduced. Substantial research efforts have also been invested in reducing the LDPC's decoding complexity. The FFT based decoding algorithm proposed by Richardson and Urbanke [9, 96] and the linearly decodable LDPC codes proposed by Spielman [97] constitute a few examples of low-complexity decoders. Furthermore, there are numerous other low-complexity decoding algorithms invented by Pothier [98], Narayanan [99] and Fossorier *et al.* [100].

1948	Shannon limit quantified; Shannon [67]
1962	LDPCCs invented; Gallager [1]
1975	LDPCCs' complexity quantified; Zyablov *et al.* [4]
1983	Tanner Graph introduced for the LDPCC parity check matrix; Tanner [5]
1997	Near-Shannon-Limit performance reported; MacKay *et al.* [7]
1998	Non-binary LDPCCs invented; Davey *et al.* [54,55]
	Irregular LDPCCs proposed; Luby *et al.* [60]
	Reduced complexity decoding algorithm using FFT proposed; Richardson *et al.* [9]
	Density evolution algorithm proposed; Richardson *et al.* [9]
1999	Generalised LDPC proposed; Lentmaier [105]
2001	Finite geometry based LDPC proposed; Kou *et al.* [83,84,106]
	Bit-flipping decoding of LDPCCs proposed; Kou *et al.* [83]
	EXIT-chart invented; ten Brink [107]
	Gaussian approximated density evolution method proposed; Chung *et al.* [11]
2002	BIBD based LDPC proposed; Ammar *et al.* [91,92]
	Bootstrap decoding algorithm proposed; Nouh *et al.* [102]
	Bit-based three-layer LDPC-MIMO proposed; Meshkat *et al.* [108]
2004	Bit-flipping decoding algorithm improved; Zhang *et al.* [101]
	Symbol-based three-layer LDPC-MIMO proposed; Guo *et al.* [27]
	Reliability ratio based bit-flipping algorithm proposed; Guo *et al.* [103]

Table 1.1: Mile-stones in LDPC coding research

In addition to the sub-optimum Sum-Product Algorithm (SPA) devised for decoding LDPC codes, a significantly less complex bit-flipping based algorithm was proposed by Kou *et al.* [82]. This algorithm was further improved by Zhang *et al.* in [101]. Furthermore, a bootstrap bit-flipping algorithm was proposed by Nouh *et al.* in [102]. The reliability-ratio based algorithm was further developed by Guo *et al.* in [103,104]. All these low-complexity bit-flipping algorithms constitute a useful supplement to the popular SPA algorithm in the context of applications, where the system's complexity is limited.

All these historic findings constitute the motivation of the underlying research described in this thesis.

1.2 Outline of the Thesis and its Novel Contributions

The outline of the thesis is as follows. In Chapter 2 we introduce Gallager's original LDPCC construction algorithm with the aid of an example. Additionally, a general description of the LDPC decoding process is provided and the chapter is concluded with a range of performance results.

- A binary LDPC aided joint coding and modulation scheme was designed and benchmarked against a Turbo Trellis Coded Modulation (TTCM) scheme. At a BER of 10^{-5} an E_b/N_0 gain of about 1.5 dB was achieved using 15 iterations in comparison to the TTCM benchmarker using

4 iterations. The effective throughput of the system was 1 bit per symbol [37].

The irregular construction of binary Parity Check Matrices (PCM) will be discussed in Chapter 3. Two different PCM construction methods will be introduced. One of them is capable of providing a high error correction performance at long blocklengths, while the other one will be aiming at lowering the error floor encountered, when moderate blocklengths and a high code rate have to be used.

Chapter 4 will illustrate how LDPCCs may be constructed and decoded over non-binary fields.

- These non-binary codes, invented by Davey and MacKay [54], will then be invoked for constructing a novel purely symbol-based MIMO scheme. This scheme is benchmarked against a bit-based MIMO scheme invoking a binary LDPC. An E_b/N_0 gain of about 2 dB was obtained at a BER of 10^{-5} for a MIMO scheme operating over $\mathbf{GF}(4)$, $\mathbf{GF}(8)$ as well as $\mathbf{GF}(16)$ and an uncorrelated Rayleigh fading channel. Furthermore, the decoding complexity imposed was also significantly reduced by the purely symbol-based scheme [27].

In Chapter 5, the family of Variable Length Codes (VLC) is introduced and it is demonstrated, how they may be utilised for error correction. Different types of VLCs are serially concatenated with various channel codecs and decoded in an iterative fashion. The associated performance trends are highlighted, providing an insight on how to choose the VLCs in the context of serially concatenated joint source and channel coding schemes using Extrinsic Information Transfer Charts (EXIT-Charts).

Chapter 6 offers an alternative technique of decoding LDPCCs using bit-flipping.

- A novel Reliability Ratio based Weighted Bit Flipping (RRWBF) algorithm is proposed. It is demonstrated that an improved BER performance may be attained at no decoding complexity penalty. When communicating over an AWGN channel, a (1000,500,5.0) LDPC code decoded by the RRWBF algorithm achieved an E_b/N_0 gain of about 1.5 dB at a BER of 10^{-5} in comparison to the set of known bit-flipping decoding algorithms [103, 104].

Finally, Chapter 7 summarises the findings of the thesis and offers a range of further research topics.

Chapter 2

Binary LDPC codes

2.1 Introduction

S		U			N
S	O	U	P		
	O		P	E	N

Figure 2.1: Introductory example

Let us commence our discussions on a light-hearted note, considering the construction of Figure 2.1, which is reminiscent of that of a cross-word puzzle. Let us assume that the number of letters in the same column are identical and each row has to be a valid word. Let us now change for example the character 's' in the first column or 'p' in the fourth column of Figure 2.1, which can be viewed as the effect of a transmission error imposed by the channel. An intelligent human or a smart channel decoder may be able to spot this error and might be able to correct it, such that each horizontal line still remains a valid word, although in some cases ambiguous solutions may exist. The reason that we are able to spot and probably even correct the error is not only due to the redundancy inherent in the English language, but also because we may find clues confirming certain letters with a high confidence from the other words seen in the different rows in Figure 2.1. LDPC codes have similar properties to those of our example.

The most important parameter or descriptor of the family of LDPCCs is their parity check matrix. Luby [60] showed that the LDPC code's performance may potentially be increased, when an irregular parity check matrix construction is applied. However, at this stage we only consider LDPCCs having a regular parity check matrix construction, as it was initially proposed by Gallager [1].

In order to specify an LDPCC, its parity check matrix has be to defined first. Then the corresponding generator matrix can be derived for this parity check matrix. Based on these parameters, the encoder will be in the position to be able to generate the encoded bits for transmission.

2.2 Linear block codes

LDPCCs belong to the family of linear block codes [1], hence the code can be defined by a *parity check matrix* $\mathbf{H}$ and a corresponding *generator matrix* $\mathbf{G}$. The parity check matrix $\mathbf{H}$ and the generator matrix $\mathbf{G}$ have a size of $(N-K) \times N$ and $(K \times N)$, respectively. Explicitly, $\mathbf{H}$ can be represented as:

$$\mathbf{H} = (\mathbf{I} \mid \mathbf{A}), \tag{2.1}$$

where $\mathbf{I}$ is an $(N-K) \times (N-K)$-dimensional identity matrix and $\mathbf{A}$ is a non-singular i.e. invertible matrix, while the corresponding generator matrix $\mathbf{G}$ can be represented as:

$$\mathbf{G} = (-\mathbf{A}^T|\mathbf{I'}), \tag{2.2}$$

where $\mathbf{I'}$ is a $(K \times K)$-dimensional identity matrix and $\mathbf{A}^T$ is the transpose of the non-singular matrix $\mathbf{A}$ used for defining $\mathbf{H}$. The product of the matrixes $\mathbf{H}$ and $\mathbf{G}^T$ is by definition an all-zero matrix. More explicitly,

$$\mathbf{H}_{(N-K)\times N} \cdot \mathbf{G}^T_{N\times K} = \mathbf{0}_{(N-K)\times K}. \tag{2.3}$$

For each source information block $\mathbf{S}$ of size $1 \times K$, encoding is carried out by multiplying it with the generator matrix $\mathbf{G}$ and the resultant $1 \times N$-dimensional vector is the encoded codeword $\mathbf{C}$, which is formulated as:

$$\mathbf{C}_{1\times N} = \mathbf{S}_{1\times K} \cdot \mathbf{G}_{K\times N}. \tag{2.4}$$

The validity of the codeword can be verified by calculating the syndrome vector upon multiplying $\mathbf{C}$ with $\mathbf{H}^T$, which becomes an all-zero syndrome vector, if the codeword is legitimate.

$$\mathbf{Syndrome}_{(1\times M)} = \mathbf{C}_{(1\times N)} \cdot {\mathbf{H}^T}_{(N\times M)}. \tag{2.5}$$

The encoding and parity-checking process is shown more explicitly below:

$$[c_1, c_2, \ldots c_N] = [s_1, s_2, \ldots s_K] \begin{bmatrix} g_{1,1} & \cdots & g_{1,N} \\ \vdots & \ldots & \vdots \\ g_{K,1} & \cdots & g_{K,N} \end{bmatrix}.$$

$$[0_1, 0_2, \ldots 0_M] = [c_1, c_2, \ldots c_N] \begin{bmatrix} h_{1,1} & \ldots & h_{1,M} \\ \vdots & \ldots & \vdots \\ h_{N,1} & \ldots & h_{N,M} \end{bmatrix},$$

where the notation c_j, s_j represents the j^{th} element of the codeword and the source information vector, respectively. The notations $g_{i,j}$ and $h_{i,j}$ are for representing the element of the generator matrix $\mathbf{G}$ and the transpose of the parity check matrix $\mathbf{H}$, i.e. $\mathbf{H}^T$ at position (i,j), respectively.

2.3 Parity check matrix

A conventional parity check code may be formed by combining a block of binary digits checking the information part of the codeword. In the example seen in Figure 2.2, each parity check bit is the

modulo 2 sum of a specific set of information bits. We will use the terminology of *information bits* to denote the uncoded source bits. By contrast, *parity bits* are defined as the redundant bits appended to the information bits in the codeword. As seen in Figure 2.2, each row of the parity check set can be written as a *parity check equation* given at the right of Figure 2.2. For any legitimate codeword, all the parity check equations have to be satisfied.

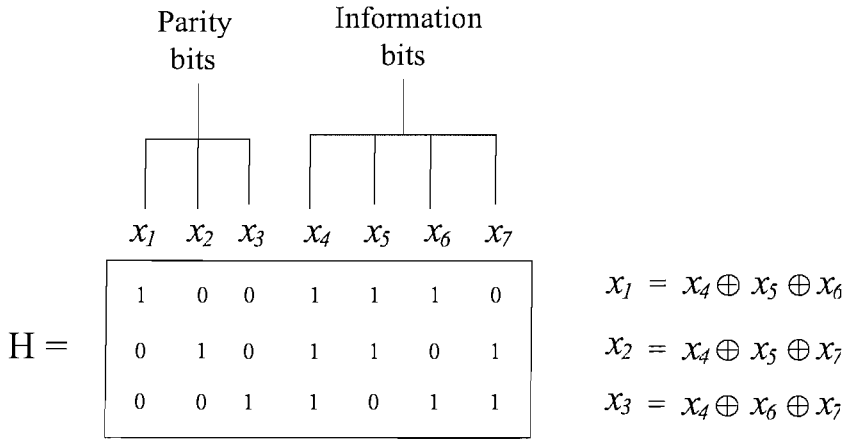


Figure 2.2: Conventional parity check set designed for block codes

LDPCCs, as suggested by the nomenclature, are typically specified by a matrix predominantly containing logical zeros, and only a small number of logical ones, hence the term "low density". Each LDPCC can be uniquely defined by the associated $(N-K) \times N$-dimensional parity check matrix. The notation M is used for representing the number of rows in the parity check matrix $\mathbf{H}$, which equals $(N-K)$. The parity check matrix can be specified by the parameters (N, w_c, w_r), where N is the encoded block-length, while w_c and w_r are the *average* Hamming weight of the columns and rows, respectively. More explicitly, when w_c is an integer and a so-called regular LDPC PCM construction is applied, according to the definition of regular construction all the columns will have the same weight of w_c. However, if w_c is not an integer, say 2.5, then half of the columns will have weight two, and the remaining half will have weight three. These codes will be referred to as near-regular-construction LDPC codes. By contrast, the terminology of irregular LDPC codes will refer to those LDPC codes, whose PCM has significant variation in column weights. This type of irregular LDPC codes will be introduced later in Chapter 3. Provided that the parity check matrix $\mathbf{H}$ has full rank,[1] the LDPCC's code rate can be calculated as $r = K/N$. Since the number of non-zero entries in the parity check matrix is a constant, we have $N \cdot w_c = M \cdot w_r$. Hence the code rate can also be represented by $r = 1 - (w_c/w_r)$. In the scenario, when there are dependent rows in the parity check matrix $\mathbf{H}$, the actual code rate will be higher than the figure calculated above. There is an alternative way of specifying a parity check matrix by the coded blocklength, the information blocklength and the column weight, i.e. as (N, K, w_c).

An example of the LDPC Parity Check Matrix (PCM) is given in Table 2.1. This is the PCM Gallager used in his seminal paper on LDPCCs in 1963.

[1]In this LDPC context having a $\mathbf{H}$ matrix which is of full rank implies that all the rows in the $\mathbf{H}$ matrix are independent.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2	0	0	0	0	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0	1	1	1	1	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	0	0	0	0
5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1
6	1	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0	0	0	0	0
7	0	1	0	0	0	1	0	0	0	1	0	0	0	0	0	0	1	0	0	0
8	0	0	1	0	0	0	1	0	0	0	0	0	0	1	0	0	0	1	0	0
9	0	0	0	1	0	0	0	0	0	0	1	0	0	0	1	0	0	0	1	0
10	0	0	0	0	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0	1
11	1	0	0	0	0	1	0	0	0	0	0	1	0	0	0	0	0	1	0	0
12	0	1	0	0	0	0	1	0	0	0	1	0	0	0	0	1	0	0	0	0
13	0	0	1	0	0	0	0	1	0	0	0	0	1	0	0	0	0	0	1	0
14	0	0	0	1	0	0	0	0	1	0	0	0	0	1	0	0	1	0	0	0
15	0	0	0	0	1	0	0	0	0	1	0	0	0	0	0	0	0	0	1	1

H=

Table 2.1: Example of a low density parity check matrix (PCM) for N=20, w_c=3, w_r=4 [1] with full rank, where N, w_c, w_r are used for representing the total number of columns, ie. the codeword length, the number of non-zero entries per column, and the number of non-zero entries per row, respectively. Finally, the number of rows in the parity check matrix is $M = N \times w_c/w_r = 15$. The coding rate is defined as $r = (N - M)/N$.

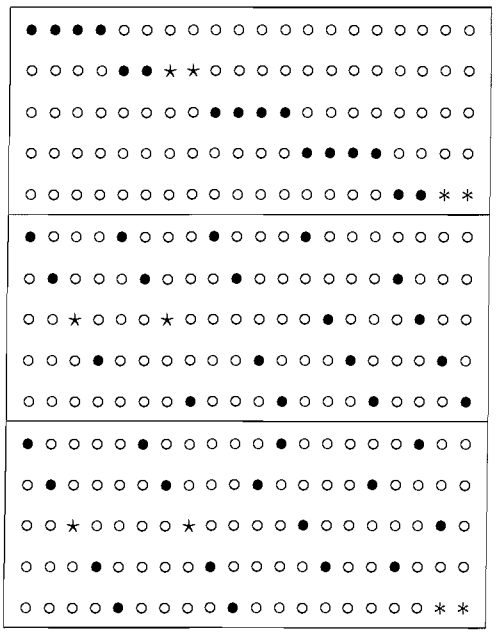


Table 2.2: Reproduction of Table 2.1 highlighting the nodes forming cycles of length 4 ($*$) and 6 ($\star$).

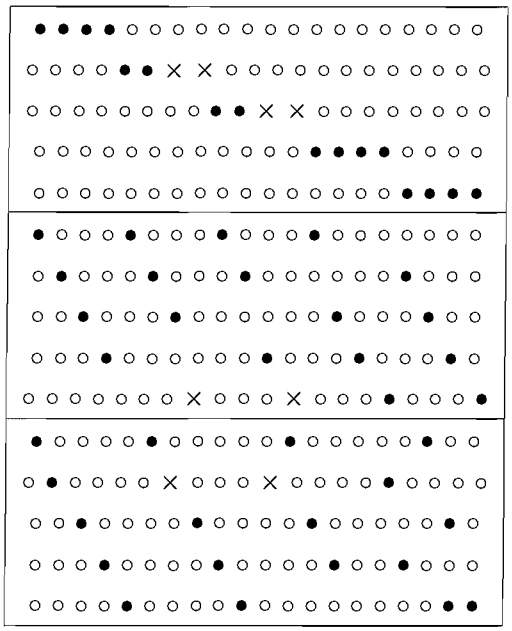


Table 2.3: Reproduction of Table 2.1 highlighting the nodes forming cycles of length 8 (×)

Gallager suggested a regular construction for the parity check matrix which is shown in Table 2.1. This construction is described as follows. As it can be seen in Table 2.1, the rows may be divided into three subsets. In the first subset, the binary 1s in the i^{th} row will occupy the columns spanning from $[(i-1) \times w_r + 1]$ to $[i \times w_r]$, and their position indicates, which information bits participate in the i^{th} parity check equation, i.e. the i^{th} row of the PCM, where each of the 15 rows represents one of the 15 parity check equations. As becomes explicit in Table 2.1, each of the parity check equations is checking the parity of both some information bits and some parity bits. For example, in the (N, w_c, w_r)=(20, 3, 4) code of Table 2.1, the binary 1s in the first row occupy the positions spanning from $[(1-1) \times 4] + 1 = 1$ to $1 \times 4 = 4$. Similarly, the binary 1s in the second row occupy the positions spanning from $[(2-1) \times 4] + 1 = 5$ to $2 \times 4 = 8$, hence the second row of the PCM checks the parity of the bits located at the 5^{th}, the 6^{th}, the 7^{th} and the 8^{th} column of the PCM.

When the first subset has been constructed in this way, a number of further subsets separated by the horizontal lines in Table 2.1 are created by random permutation of the columns obeying an equal probability of permutation. For example, in the second horizontal partition of Table 2.1, the 5^{th} and 2^{nd} columns have been swapped, hence the associated 6^{th} parity check equation involves the 5^{th} column, while the 7^{th} parity check equation involves the 2^{nd} column. Furthermore, the 3^{rd} and the 9^{th} columns were exchanged, etc. According to Gallager's description of the parity check matrix, the whole matrix will be divided into w_c subsets, where w_c is the number of non-zero entries per column, i.e. the column weight. When the block-length N increases, while keeping the parameters w_c and w_r constant, the parity check matrix becomes more and more sparse. Hence the minimum distance of the code, which is defined as the number of bit positions, where the two nearest code words differ, increases as well, provided that the column weight of the PCM equals or greater than three [1]. Gallager showed [1] that for a large N, the Cumulative Density Function (CDF), more precisely the histogram of the minimum distance approaches a unit step at a fixed fraction δ of the total blocklength. The corresponding Probability Density Function (PDF) of the minimum distance

is similar to a Delta function of height $\frac{1}{N}\delta$ concentrated at the weight $n\delta$, indicating that practically all the codewords in the ensemble have a minimum distance that is similar to $N\delta$.

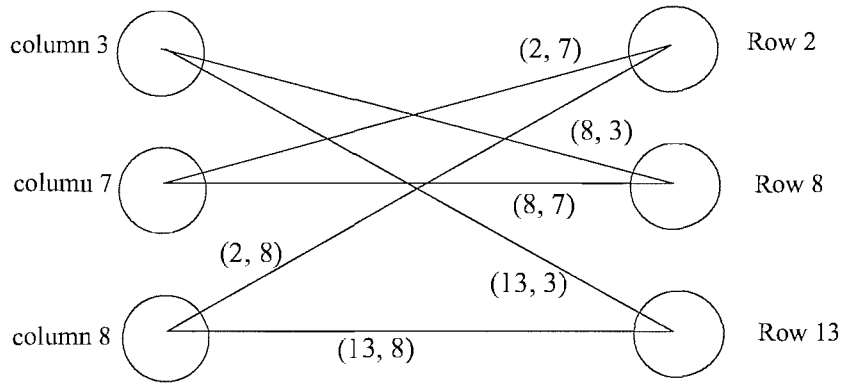


Figure 2.3: Bipartite graph representation of a length-6 cycle, highlighted using the marker $\star$ in Table 2.2.

The PCM may also be represented using a Tanner graph [5], which is constituted by nodes and connections. Figure 2.3 shows a Tanner graph representing a fraction of the PCM seen in Table 2.2. The nodes on the left of the Tanner graph are called the *message nodes*, which represent the specific columns of the PCM indicated at the left of the figure. On the right of the Tanner graph are the *check nodes*, which correspond to the specific rows of the PCM identified at the right of the figure. A connection between a message node and a check node represents the corresponding non-zero entry in the PCM. More explicitly, as shown in Figure 2.3, the connection between the column-3 message node and the row-13 check node represents the non-zero entry at position (13,3) of the PCM in Table 2.2. During the decoding process, the information will be passed from a message node to a check node via the connection between them, and the check node will also feed the updated information back to the message node through the connections.

As a result of further research efforts, the concept of *cycles* was devised [5], where the length of a *cycle* refers to the number of non-zero entries in the parity check matrix, which can be connected to form a cycle, as will be explained below. The concept of cycles can also be represented in the Tanner graph, where the connections between the nodes form a closed loop. An example of a *length-6 cycle* is shown in Figure 2.3. More explicitly, there are six connections in Figure 2.3 representing a closed loop, which is referred to as a *length-6 cycle*. These six lines represent the non-zero entries (2,7), (8,7), (8,3), (13,3), (13,8) and (2,8) of the PCM seen in Table 2.1, and this *length-6 cycle* is highlighted using the marker $\star$ in Table 2.2.

It will be shown in Section 2.5 that this concept is related to the LDPCC's decoding algorithm. More explicitly, during the LDPC decoding process, the decoding information will be exchanged both vertically and horizontally between the non-zero entries of the PCM. As for the Tanner graph representation, the decoding information will be travelling back and forth between the message nodes and the check nodes. The length of the cycle determines the amount of extrinsic information that a particular non-zero entry is supplied with. However, for the sake of providing some intuition into the deeper significance of cycles, suffice to say here that the formation of long cycles indicates that the PCM promotes the efficient exchange of decoding information. By contrast, the presence of short

H=

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	0	1	1	1	0	0	0	1	0	0	1	0	0	0	0
2	0	0	0	1	1	0	1	1	0	1	0	0	0	0	0
3	1	0	0	0	0	0	1	0	0	0	0	1	0	1	1
4	1	1	0	0	0	1	0	0	1	1	0	0	0	0	0
5	1	0	0	0	0	1	0	0	1	0	0	0	1	1	0
6	0	0	0	0	1	0	0	0	1	0	1	0	1	0	0
7	0	0	0	0	0	0	1	0	0	1	0	1	0	0	1
8	0	0	0	0	0	1	0	0	0	0	0	0	1	1	1
9	0	1	1	1	1	0	0	0	0	0	0	0	0	0	0
10	0	0	1	0	0	0	0	1	0	0	1	1	0	0	0

Table 2.4: Example of a parity check matrix **H** having N=15, w_c=3, w_r=4.5, M=$N-K$=$N\times w_c/w_r$ = 10 and r=1/3.

cycles implies a limited exchange of information, leading to a limited decoding performance.

The more entries are needed for forming a cycle, the more decoding information is passed around during the decoding process and hence the better the performance of the parity check matrix concerned. In Section 2.5, where the process of probabilistic decoding is described, it will be shown that the information needed for decoding will be passed between the non-zero entries of the same row or of the same column. Thus if the length of the cycle is high, each non-zero entry in the parity check matrix is capable of benefiting from more parity-related information provided by other entries within the cycle. The shortest possible cycle of the PCM of Table 2.1 is a *length-4 cycle*, marked as $*$ in Table 2.2, since there are two non-zero entries in the fifth row and another two non-zero entries in the 15^{th} row of the parity check matrix, both pairs occupying the last two columns. Hence during the construction of the parity check matrix, the existence of the shortest cycle, namely that of the *length-4 cycle* should be eliminated, if possible. In this thesis a parity check matrix generated by computer search will be introduced, which will be used during the rest of our discussions on the LDPCC encoding and decoding process. The size of this PCM is quite small for the sake of simplicity, which is hence inadequate for practical applications. Owing to the restricted size of the matrix, it has *length-4 cycles*.

The parity check matrix of the LDPCC is invoked at the decoder's output for determining whether the decoded codeword is a valid one. As stated previously, the parity check matrix is used for creating the generator matrix, which is applied at the encoder for generating the parity bits to be appended to the original information bits. Thus the product of the received codeword **C**, which is a $(1\times N)$-dimensional row vector, and the parity check matrix **H** becomes zero if there are no errors after the iterative decoding process. A simple example highlighting this property is provided by assuming that an encoded frame of 15 consecutive zero bits has been transmitted through a channel, and the 1^{st} as well as the 3^{rd} bit were corrupted, which gives a binary 1 both at bit position 1 and 3 in this received codeword. Upon multiplying the received data stream of 101000000000000 with $\mathbf{H}^T$, which is the transpose of the parity check matrix given by Table 2.4, we arrive at the stream of 1011100011. This shows that the 1^{st}, 3^{rd}, 4^{th}, 5^{th}, 9^{th} and the 10^{th} parity check equations are violated. Thus the decoder will continue iterating, as it will be shown in Section 2.5, unless the decoding process has reached the

predetermined maximum number of iterations.

Gallager also showed [1] that when this encoding scheme is applied for transmission over the Binary Symmetric Channel (BSC) [109], the results approach the optimum in terms of the achievable bit error probability. While the BSC constitutes a useful model for initial theoretical investigations, it is by no means a practical channel model. For correcting transmission errors, Gallager proposed the employment of **probabilistic decoding**, sometimes also referred to as the **belief propagation** algorithm [110] or sum-product algorithm , which evaluates the *a posteriori* [2] probabilities of the bits of the various received words, as it will be shown in the next section.

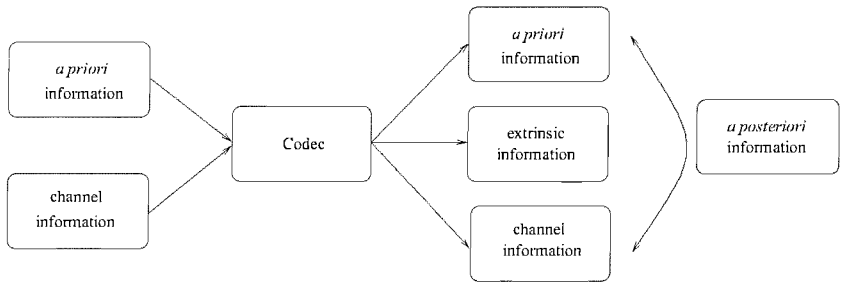


Figure 2.4: The *extrinsic* information is generated by the codec and constitutes the overall *a posteriori* information together with the input *a priori* information and the channel information.

2.4 LDPC encoding

When using a channel code for protecting messages, we have to define a mapping from the original uncoded information bit vector, $\mathbf{S}$ of dimension $1 \times K{=}1 \times (N-M)$, where M is the number of rows of the parity check matrix, which can be expressed as $M = N \times \frac{w_c}{w_r}$, when the parity check matrix $\mathbf{H}$ has full rank. All the notations used bear the same meaning as previously. The symbol $\mathbf{S}$ was introduced here for representing an uncoded information message, which is encoded into a codeword $\mathbf{C}$ of dimension $1 \times N$ carrying the uncoded systematic source information $\mathbf{S}$ at the end of the codeword. Only linear mapping schemes will be considered here, which can be written in a matrix form as $\mathbf{C}{=}\mathbf{S}{\cdot}\mathbf{G}$, where $\mathbf{G}$ is the *generator matrix* derived from the PCM $\mathbf{H}$ exemplified in Table 2.4, as will be shown during our forthcoming discussion. For an LDPC code having a parity check matrix $\mathbf{H}$, any legitimate codeword has to satisfy $\mathbf{C}{\cdot}\mathbf{H}^T{=}\mathbf{0}$. The generator matrix $\mathbf{G}$ can be calculated using the following steps.

Since the parity check matrix $\mathbf{H}$ has an $(N-K) \times N{=}M \times N$-dimensional structure, it can be divided into an $(N-K) \times (N-K){=}M \times M$-dimensional matrix $\mathbf{A}$ and an $M \times (N-M)$-dimensional matrix $\mathbf{B}$. Furthermore, since we have $\mathbf{C}{\cdot}\mathbf{H}^T{=}\mathbf{0}$, this equation can also be written as:

$$\mathbf{C} \cdot \mathbf{H}^T = \mathbf{C} \cdot (\mathbf{A};\mathbf{B})^T = \mathbf{P} \cdot \mathbf{A}^T + \mathbf{S} \cdot \mathbf{B}^T = \mathbf{0}, \tag{2.6}$$

[2] We will use the terminology of *a priori* information, *extrinsic* information and *a posteriori* information to specify the input and output information of a codec. The *a priori* information refers to the additional information, with respect to the channel information, which is provided by an external codec. The terminology of *extrinsic* information corresponds to the extra new information generated internally inside the codec. The *a posteriori* information is a combination of the *a priori* information, *extrinsic* information and the channel information. An illustration is provided in Figure 2.4.

where ";" denotes matrix partition, and $\mathbf{S}$ and $\mathbf{P}$ are row vectors of dimension $1 \times K$ and $1 \times M$=$1 - (N - K)$, used for representing the information bits and the parity bits, respectively. Provided that the submatrix $\mathbf{A}^T$ is non-singular and hence may be inverted, from Equation 2.6 we have

$$\mathbf{P} = \mathbf{S} \cdot (\mathbf{B}^T \cdot (\mathbf{A}^T)^{-1}). \tag{2.7}$$

Thus, using the parity bits $\mathbf{P}$ derived in this way, the LDPC encoded codeword $\mathbf{C}$ can be constructed by appending the information segment $\mathbf{S}$ at the end of the parity bit segment $\mathbf{P}$. However, since the parity check matrix has a pseudo-random construction, it cannot be guaranteed that the submatrix $\mathbf{A}^T$ is invertible. Hence, the columns of the parity check matrix may have to be reordered for the sake of being able to calculate the inverse of the matrix $\mathbf{A}^T$. If the original matrix $\mathbf{A}^T$ is singular, we will randomly select a column from $\mathbf{B}$ and swap it with a randomly chosen column of $\mathbf{A}$. This process continues until the matrix $\mathbf{A}^T$ becomes non-singular. Thus, after reordering the columns of the matrix $\mathbf{H}$, we arrive at the reordered parity check matrix $\mathbf{H}_r$, and based on the new PCM $\mathbf{H}_r$, the generator matrix $\mathbf{G}$ can be constructed by calculating the matrix $(\mathbf{B}^T \cdot (\mathbf{A}^T)^{-1})_{(K \times M)}$ as described based on $\mathbf{H}_r$. Then a $K \times K$-dimensional identity matrix $\mathbf{I}_{(K \times K)}$ is appended to the right of the matrix $(\mathbf{B}^T \cdot (\mathbf{A}^T)^{-1})_{(K \times M)}$. Hence we arrive at the $(K \times N)$-dimensional generator matrix $\mathbf{G}_{(K \times N)}$ given by:

$$\mathbf{G}_{(K \times N)} = [(\mathbf{B}^T \cdot (\mathbf{A}^T)^{-1})_{(K \times M)}; \mathbf{I}_{(K \times K)}]. \tag{2.8}$$

Since the column order of the original PCM $\mathbf{H}$ has been altered, the encoded codewords generated upon multiplying the information sequences by $\mathbf{G}$ created from $\mathbf{H}_r$ will be incompatible with the original PCM $\mathbf{H}$, and hence the equation $\mathbf{C} \cdot \mathbf{H}^T = \mathbf{0}$ will not hold. Therefore, after the generator matrix $\mathbf{G}$ has been created from $\mathbf{H}_r$, we should use $\mathbf{H}_r$ for future manipulations. Let us now consider the following encoding example.

When using the PCM $\mathbf{H}$ of Table 2.4, we first have to obtain $\mathbf{H}_r$ for the sake of constructing the generator matrix $\mathbf{G}$. Since the column order of the original PCM $\mathbf{H}$ has been altered for the sake of finding an invertible submatrix $\mathbf{A}^T$ as described previously, the PCM $\mathbf{H}_r$ is obtained in the form of Table 2.5. The matrixes $\mathbf{A}$ and $\mathbf{B}$ in Equation 2.6 constitute the left and right parts of $\mathbf{H}_r$, separated by the vertical line in Table 2.5.

By calculating $\mathbf{B}^T \cdot (\mathbf{A}^T)^{-1}$ in Equation 2.7, we arrive at the $K \times M$=$(N - M) \times M$-dimensional matrix seen in Table 2.6. Upon multiplying the source bit stream $\mathbf{S}$ with this matrix, the parity bits $\mathbf{P}$ can be obtained as seen in Equation 2.7. Thus the encoded codeword $\mathbf{C}$ is generated by multiplying the information bit stream $\mathbf{S}$ with the generator matrix $\mathbf{G}$, as follows:

$$\mathbf{C}_{(1 \times N)} = \mathbf{S}_{(1 \times K)} \cdot \mathbf{G}_{(K \times N)} \tag{2.9}$$

Assuming that we have a 5-bit information bit stream of $\mathbf{S}$=01001, by multiplying $\mathbf{S}$ with the generator matrix $\mathbf{G}$ of Table 2.7, the LDPC encoded codeword is given by $\mathbf{C}$ = 101110100101001. The last five bits are the original information bits, while the first 10 bits are the parity bits. Upon multiplying this encoded codeword $\mathbf{C}$ with the parity check matrix $\mathbf{H}_r^T$ of Table 2.5, we will arrive at the 10-component all-zero vector, which implies that the encoded codeword was indeed a legitimate one.

$\mathbf{H}_r$=

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	1	1	0	1	0	0	0	0	1	0	1	0	0	0	0
2	0	1	0	0	0	1	1	0	1	1	0	0	0	0	0
3	0	0	1	0	0	0	1	0	0	0	0	1	0	1	1
4	1	0	1	0	1	0	0	1	0	1	0	0	0	0	0
5	0	0	1	0	1	0	0	1	0	0	0	0	1	1	0
6	0	0	0	0	0	1	0	1	0	0	1	0	1	0	0
7	0	0	0	0	0	0	1	0	0	1	0	1	0	0	1
8	0	0	0	0	1	0	0	0	0	0	0	0	1	1	1
9	1	1	0	1	0	1	0	0	0	0	0	0	0	0	0
10	0	0	0	1	0	0	0	0	1	0	1	1	0	0	0

Table 2.5: The PCM $\mathbf{H}_r$ constructed from $\mathbf{H}$ seen in Table 2.4. The PCM $\mathbf{H}_r$ has N=15, M=10, w_c=3 and w_r=4.5. The i^{th} column of the PCM $\mathbf{H}_r$ is constructed by using the o_i^{th} column of the PCM $\mathbf{H}$, where we have $\mathbf{o}_i = \{2, 4, 1, 3, 6, 5, 7, 9, 8, 10, 11, 12, 13, 14, 15\}$. The submatrixes $\mathbf{A}$ and $\mathbf{B}$ are the left and right parts of $\mathbf{H}_r$, respectively.

$\mathbf{B}^T \cdot (\mathbf{A}^T)^{-1}$=

	1	2	3	4	5	6	7	8	9	10
1	1	1	1	0	0	0	1	1	1	1
2	0	1	0	1	0	0	1	0	0	0
3	0	0	1	0	1	0	1	1	0	1
4	0	0	0	0	1	0	1	0	0	1
5	1	1	1	0	1	0	0	0	0	1

Table 2.6: The matrix product of $\mathbf{B}^T \cdot (\mathbf{A}^T)^{-1}$ required for calculating the generator matrix $\mathbf{G}$ of Equation 2.8, which corresponds to the rearranged PCM $\mathbf{H}_r$ in Table 2.5.

2.5 LDPC decoding

In this section, two decoding schemes will be described. The first one is referred to as *exhaustive enumeration* based decoding [111], which represents the optimum Maximum Likelihood decoding of the codeword. The second decoding method is based on probabilistic propagation [1] and this is a sub-optimum decoding technique that is capable of achieving near-optimum performance.

2.5.1 Exhaustive enumeration based decoding

The optimum exhaustive enumeration decoding method [111] evaluates the probability of encountering each legitimate codeword based on the product of the individual bit probabilities quantified as the demodulator's soft-output. This approach compares the received codeword to all legitimate codewords for the sake of finding the most likely original transmitted codeword. However, this decoding philosophy becomes impractical for a high blocklength, since the number of legitimate codewords increases exponentially with respect to the blocklength. Therefore, we introduce the low-complexity but sub-optimal probabilistic decoding algorithm in Section 2.5.2.

$\mathbf{G}=$

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	1	1	1	0	0	0	1	1	1	1	1	0	0	0	0
2	0	1	0	1	0	0	1	0	0	0	0	1	0	0	0
3	0	0	1	0	1	0	1	1	0	1	0	0	1	0	0
4	0	0	0	0	1	0	1	0	0	1	0	0	0	1	0
5	1	1	1	0	1	0	0	0	0	1	0	0	0	0	1

Table 2.7: The generator matrix $\mathbf{G}=[(\mathbf{B}^T \cdot (\mathbf{A}^T)^{-1})_{(K\times M)}; \mathbf{I}_{(K\times K)}]$ corresponding to the parity check matrix $\mathbf{H}_r$ of Table 2.5.

2.5.2 Probabilistic decoding (Gallager's method)

The second decoding method to be introduced provides a better trade-off between the achievable performance and the associated complexity. When decoding a received sequence, two important aspects have to be taken into consideration. Firstly, how likely it is that a bit has been received incorrectly owing to the channel effects. Secondly, whether the redundant bits checking the parity of a particular bit may be considered as error-free. Hence we have to calculate the probability of the received symbol being a binary 1 or 0, conditioned on receiving a specific contaminated demodulator soft-output sample y and also conditioned on the event S that this specific bit satisfies all the w_c parity check equations containing it. This statement can be written in a compact form as:

$$P[x_j = 1|\{y\}, S]. \tag{2.10}$$

Let P_j^1 be the probability that a binary 1 was transmitted at bit position $j = 1...N$, based on the demodulator's soft output. Furthermore, let us use $P_{i,j}^1$ to denote the probability of encountering a binary 1 at the bit position j, $(j = 1...N)$ in the i^{th}, $(i = 1...M)$ parity check of Table 2.5. Gallager provided a formula for evaluating the ratio of the probabilities of a binary 1 and a binary 0 being transmitted at bit position $j = 1 \dots N$ in the following form, as it was shown later in Appendix B [1]:

$$\frac{P[x_j = 0|\{y\}, S]}{P[x_j = 1|\{y\}, S]} = \frac{1 - P_j^1}{P_j^1} \prod_{i\in\{R_j\}} \left[\frac{1 + \prod_{l\in\{C_i\}, l\neq j}(1 - 2P_{i,l}^1)}{1 - \prod_{l\in\{C_i\}, l\neq j}(1 - 2P_{i,l}^1)} \right], \tag{2.11}$$

where $\{C_i\}$ is used for representing the set of column indices of the non-zero entries in the i^{th} row of the PCM of Table 2.5, and $\{R_i\}$ is used for representing the set of row indices of the non-zero entries in the i^{th} column of the PCM of Table 2.5. For example, in Table 2.5 we have $\{C_2\}$ = {2,6,7,9,10} and $\{R_{12}\}$={3,7,10}. Furthermore, $\{R_j\}$ gives the set of row indices of the j^{th} bit in Table 2.5. Equation 2.11 illustrates the method of calculating the *a posteriori* probability ratio of the j^{th} bit conditioned on the knowledge of the channel's soft output y and on satisfying the parity check equations associated with the j^{th} bit. This *a posteriori* probability ratio is calculated from the intrinsic probability of the j^{th} bit provided by the channel's soft output y_j, and from the probabilities of the neighbouring non-zero entries denoted as $P_{i,l}$ in Equation 2.11. This will be exemplified with the aid of a worked example in Section 2.6, while the proof of Equation 2.11 is given in Appendix B.

Let us introduce the Likelihood Ratio (LR) expressed as [1]:

$$LR_{i,j} = \frac{1 + \prod_{l\in\{C_i\}, l\neq j}(1 - 2P_{i,l}^1)}{1 - \prod_{l\in\{C_i\}, l\neq j}(1 - 2P_{i,l}^1)} \quad i = 1 \dots M, j = 1 \dots N, \tag{2.12}$$

and the Probability Ratio (PR) as:

$$PR_{i,j} = \frac{1 - P_j^1}{P_j^1} \prod_{k \in \{R_j\}, k \neq i} LR_{k,j} \tag{2.13}$$

and

$$PR(x_j) = \frac{1 - P_j^1}{P_j^1} \prod_{i \in \{R_j\}} LR_{i,j} \quad j = 1 \ldots N. \tag{2.14}$$

Equation 2.13 describes how the PR information of each non-zero entry of the PCM is updated. More explicitly, $LR_{i,j}$ in Equation 2.12, $i = 1 \ldots M, j = 1 \ldots N$ specifies the ratio of the probabilities that the j^{th} encoded bit is a binary 0 normalised to the probability of it being a binary 1 based on the product of soft information $P_{i,l}$ in Equation 2.12 provided by the *other* non-zero PCM entries in the i^{th} row, but excluding the current one according to Equation 2.12. The proof of Equation 2.12 is provided in Appendix A. Furthermore, the *Probability Ratio* $PR_{i,j}$ in Equation 2.13, represents similar information to $LR_{i,j}$ in Equation 2.12, except that the associated soft information is supplied by all the other non-zero entries in the j^{th} column. Finally, the notation $PR(x_j)$ in Equation 2.14 gives the overall *a posteriori* probability ratio of the j^{th} coded bit. Additionally, we will also introduce the notation $P_r(x_j)$ for representing the *intrinsic* rather than *a posteriori* probability ratio of the j^{th} bit, which will be used in Section 2.6.

The LDPC decoding process involves both a vertical and a horizontal message passing operation within each decoding iteration. These two operations are implemented based on the iterative calculation of $PR_{i,j}$ and $LR_{i,j}$ expressed in Equations 2.13 and 2.12, respectively. Let us consider the non-zero PCM entry of Table 2.5 at position (3,3) for example for the sake of describing the calculation of the message $LR_{i,j}$ in Equation 2.12. There are another four non-zero entries in the same row of Table 2.5 as entry (3,3), which may be found in the 7^{th}, 12^{th}, 14^{th} and 15^{th} column. Thus the non-zero entry at (3,3) will receive *extrinsic* information from all other non-zero PCM entries in the 3^{rd} row and based on Equation 2.12 it will calculate its own $LR_{i,j}$ message. More explicitly, based on Equation 2.12 the *extrinsic* information specifies the probability of the 3^{rd} bit being a binary 0 on the basis of the probabilities of all other bits involved in the 3^{rd} parity check set of Table 2.5. By contrast, $PR_{i,d}$ of Equation 2.13 is calculated by exploiting the soft-values of all other entries in the same column. Referring again to the non-zero PCM entry at position (3,3) of Table 2.5, the updating of this information is based on Equation 2.13, exploiting the soft information provided by positions (4,3) and (5,3) in Table 2.5. During the iterative update of Equations 2.12, 2.13 and 2.14, the employment of PR expressed in Equations 2.13 and 2.14 is two-fold. The PR value of each *bit*, namely $PR(x_j)$ will be calculated as in Equation 2.14 after each iteration. Based on $PR(x_j)$ a hard decision will be made and the resultant bit sequence will be tested with the aid of the PCM. If the corresponding parity check failed and hence another iteration is necessary for finding a legitimate codeword, the PR value of each *non-zero PCM entry*, namely $PR_{i,j}$, $i = 1 \ldots M$, $j = 1 \ldots N$ will be evaluated according to Equation 2.13. The difference between the PR value in Equation 2.14 of the above-mentioned *bit* and that of the *non-zero PCM entry* is that the $PR(x_j)$ value of the *bit* in Equation 2.14 incorporates the information provided by *all* the non-zero PCM entries within the column weighted by the *intrinsic* probability ratio of the j^{th} bit, denoted as $\frac{1-P_j^1}{P_j^1}$ in Equation 2.14. By contrast, the $PR_{i,j}$ value of a *non-zero PCM entry* in Equation 2.13 is calculated based on both the knowledge of all *other* entries in the column combined with the *intrinsic* probability ratio of the j^{th} bit, i.e. by $\frac{1-P_j^1}{P_j^1}$ in Equation 2.13.

This measure is used for the sake of ensuring that the same information will not be used twice during a given iteration.

Algorithm 1 *A step-by-step description of Gallager's probabilistic LDPC decoding algorithm is provided as follows.*

1. Based on the received soft values y_j at the output of the channel, the intrinsic probability of the j^{th} bit being a binary 1 or binary 0 can be calculated as: [112]

$$P_j^1 = P(y_j|x_j = 1) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(\frac{-(y_j+1)^2}{2\sigma^2}\right) \tag{2.15}$$

$$P_j^0 = P(y_j|x_j = 0) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(\frac{-(y_j-1)^2}{2\sigma^2}\right), \tag{2.16}$$

where y_j and σ denotes the j^{th} received soft channel output value and the standard deviation of the channel's soft output, respectively.

2. The $P_{i,j}^1$ values shown in Equation 2.12 are initialised by the P_j^1 values obtained from Equation 2.15.

3. The $LR_{i,j}$ values corresponding to each non-zero entry in a given row of the PCM are updated according to Equation 2.12, which is repeated here for convenience.

$$LR_{i,j} = \frac{1 + \prod_{l\in\{C_i\}, l\neq j}(1-2P_{i,l}^1)}{1 - \prod_{l\in\{C_i\}, l\neq j}(1-2P_{i,l}^1)} \quad i = 1\ldots M, j = 1\ldots N, \tag{2.17}$$

4. The $PR_{i,j}$ values corresponding to each non-zero entry in a given column of the PCM are updated according to Equation 2.13, which is repeated here for convenience.

$$PR_{i,j} = \frac{1-P_j^1}{P_j^1} \prod_{k\in\{R_j\}, k\neq i} LR_{k,j} \tag{2.18}$$

5. For each coded bit, Equation 2.14 is used for updating the $PR(x_j)$. This is provided here again for the ease of the reader.

$$PR(x_j) = \frac{1-P_j^1}{P_j^1} \prod_{i\in\{R_j\}} LR_{i,j} \quad j = 1\ldots N. \tag{2.19}$$

6. The $P_{i,j}^1$ value corresponding to each non-zero entry of the PCM is updated according to $1/(1+PR_{i,j})$, where $PR_{i,j}$ represents the updated values from step 4.

7. Based on the $PR(x_j)$ values updated in step 5, a tentative hard decision is made and this tentatively decoded codeword is multiplied with $\mathbf{H}_r^T$.

8. If the resultant syndrome vector is an all-zero vector, we declare a legitimate codeword has been found and the iterative decoding process is terminated.

9. By contrast, if the syndrome vector is not an all-zero vector and the maximum number of LDPC iterations is reached, we will declare a decoding failure and output the tentatively decoded codeword.

10. If the maximum affordable complexity has not been exhausted, go back to step 3.

We will elaborate on these issues in more detail using a quantitative example in Section 2.6.

2.6 LDPC decoding example

First of all, let us assume that the parity check matrix $\mathbf{H}_r$ of Table 2.5 is used and the relevant generator matrix $\mathbf{G}$ is the one seen in Table 2.7. Let us assume that the information bit stream of $\mathbf{S}$=01101 is transmitted. Thus, upon multiplying the source information sequence $\mathbf{S}$ by the generator matrix $\mathbf{G}$ of Table 2.7 according to Equation 2.9, the encoded codeword $\mathbf{C}$ is found to be 100100010001101, which is listed in Table 2.8.

x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9	x_{10}	x_{11}	x_{12}	x_{13}	x_{14}	x_{15}
1	0	0	1	0	0	0	1	0	0	0	1	1	0	1

Table 2.8: LDPC encoded codeword generated according to Equation 2.9 using the generator matrix of Table 2.7

The encoded sequence of Table 2.8 is transmitted through an AWGN channel having a noise standard deviation of $\sigma = 0.9$ using BPSK modulation. A logical 0 is transmitted as +1 and a logical 1 is represented as −1. Once the encoded bit stream was transmitted through the channel, the noise-contaminated received sequence shown in the 3^{rd} column of Table 2.9 may be received. This corresponds to the demodulator's soft output samples.

	Transmitted Bits	Received Samples	Probability Ratio($P_r(x_j)$)	Probability P_j^1	Decoded Bit	
x_1	1	-0.89	0.11	0.9	1	
x_2	0	+1.19	19	0.05	0	
x_3	0	+1.576	49	0.02	0	
x_4	1	-1.19	0.0526	0.95	1	
x_5	0	+0.25	1.857	0.35	0	
x_6	0	+1.193	19	0.05	0	
x_7	0	+0.081	1.222	0.45	0	
x_8	1	-0.164	0.667	0.6	1	
x_9	0	+1.115	15.67	0.06	0	
x_{10}	0	+0.56	4	0.2	0	
x_{11}	0	+1.865	100	0.01	0	
x_{12}	1	+0.164	1.5	0.4	0	Error
x_{13}	1	-1.19	0.0526	0.95	1	
x_{14}	0	+1.19	19	0.05	0	
x_{15}	1	-0.89	0.11	0.9	1	

Table 2.9: LDPC decoding example, where a logical 0 corresponds to a positive received sample and vice versa. The P_j^1 is calculated from Equation 2.15 during step 1 of Algorithm 1 and the probability ratio $P_r(x_j)$ is calculated by $(1 - P_j^1)/P_j^1$.

The 4^{th} column of Table 2.9, denoted as P_r, specifies the ratio of the probability that the bit was originally transmitted as a binary zero over the probability that the bit was a binary one, i.e. the ratio

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	0.11	19		0.0526					15.67		100				
2		19				19	1.222		15.67	4					
3			49				1.222					1.5		19	0.11
4	0.11		49		1.857			0.667		4					
5			49		1.857			0.667					0.0526	19	
6						19		0.667			100		0.0526		
7							1.222			4		1.5			0.11
8					1.857								0.0526	19	0.11
9	0.11	19		0.0526		19									
10				0.0526					15.67		100	1.5			

Table 2.10: Initial probability ratio value $PR_{i,j}$ of each non-zero PCM entry initialised by using the $P_r(x_j)$ values seen in Table 2.9 according to the PCM of Table 2.5 following step 2 of Algorithm 1.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	0.9	0.05		0.95					0.06		0.01				
2		0.05				0.05	0.45		0.06	0.2					
3			0.02				0.45					0.4		0.05	0.9
4	0.9		0.02		0.35			0.6		0.2					
5			0.02		0.35			0.6					0.95	0.05	
6						0.05		0.6			0.01		0.95		
7							0.45			0.2		0.4			0.9
8					0.35								0.95	0.05	0.9
9	0.9	0.05		0.95		0.05									
10				0.95					0.06		0.01	0.4			

Table 2.11: Initial probability value $P^1_{i,j}$ of entry (i,j) of the PCM indicating the chances that the j^{th} bit is a transmitted binary 1 using the notation P^1_j of Table 2.9 according to the PCM of Table 2.5

$P_r(x_j) = P^0_j/P^1_j$. This represents the *intrinsic* probability ratio of each bit, which is calculated with the aid of the Gaussian PDF function given by $\frac{1}{\sqrt{2\pi}\sigma}\exp\left(\frac{-(y-m)^2}{2\sigma^2}\right)$, where m denotes the mean of the Gaussian distribution. When communicating over an AWGN channel, the only source of modulated signal corruption is the channel-induced AWGN. Upon substituting m= +1 and −1 into the PDF for the specific case of BPSK modulation, we arrive at the probability ratio of the d^{th} bit as:

$$P_r(x_j) = P_r(y_j|x_j) = \frac{P(y_j|x_j=0)}{P(y_j|x_j=1)} = \frac{\frac{1}{\sqrt{2\pi}\sigma}\exp\left(\frac{-(y_j-1)^2}{2\sigma^2}\right)}{\frac{1}{\sqrt{2\pi}\sigma}\exp\left(\frac{-(y_j+1)^2}{2\sigma^2}\right)} = \exp 2y_j/\sigma^2. \quad (2.20)$$

According to Equation 2.20, the probability ratio $P_r(x_j)$ is computed from the demodulator's soft output by using the Gaussian Probability Density Function (PDF), which provides the corresponding *intrinsic information* of the bit concerned. The terminology *intrinsic* implies that this information was acquired from the channel output related to the specific bit concerned, rather than from any surrounding bits of the same codeword, where the surrounding bits provide external or *extrinsic* information as regards to the specific bit considered. According to step 1 of Algorithm 1, the 5^{th} column of Table 2.9, namely P^1_j, gives the probability of the bit being a binary 1 corresponding to the P_r values in the 4^{th} column calculated in Equation 2.20 as:

$$P^1_j = \frac{1}{1+P_r(x_j)}. \quad (2.21)$$

At the beginning of the decoding process the *intrinsic* probability ratio seen in the 4^{th} column of

Table 2.9 will be used for initialising the non-zero entries' PR values in the PCM as in step 2 of Algorithm 1, as listed in Table 2.10. For the sake of convenient demonstration of the decoding process, Table 2.11 gives the initial probability at position (i,j) that the j^{th} bit is a transmitted binary 1 corresponding to the values seen Table 2.10. For example, the values $P^1_{i,12}$ in the 12^{th} column of Table 2.11 are calculated from the values $P_r(x_{12})$ in the 12^{th} column of Table 2.10 with the aid of Equation 2.21 as follows:

$$P^1_{i,12} = \frac{1}{1+PR_{i,12}} = \frac{1}{1+1.5} = 0.4, \quad i \in \{R_{12}\}, \tag{2.22}$$

where R_{12} represents the indices of the non-zero PCM entries in the 12^{th} row. Furthermore, the values seen in the 2^{nd} column of Table 2.11 are calculated by using the values found in the corresponding column of Table 2.10 as:

$$P^1_{i,2} = \frac{1}{1+PR_{i,2}} = \frac{1}{1+19} = 0.05, \quad i \in \{R_2\}. \tag{2.23}$$

The probabilities found in Table 2.11 will be used for calculating $LR_{i,j}$ according to Equation 2.12 following the 3^{rd} step of Algorithm 1. For example, we have:

$$\begin{aligned} LR_{3,12} &= \frac{1+\prod_{l\in\{C_3\},l\neq 12}(1-2P^1_{3,l})}{1-\prod_{l\in\{C_3\},l\neq 12}(1-2P^1_{3,l})} \\ &= \frac{1+(1-2P^1_{3,3})(1-2P^1_{3,7})(1-2P^1_{3,14})(1-2P^1_{3,15})}{1-(1-2P^1_{3,3})(1-2P^1_{3,7})(1-2P^1_{3,14})(1-2P^1_{3,15})} \\ &= \frac{1+(0.96)\times(0.1)\times(0.9)\times(-0.8)}{1-(0.96)\times(0.1)\times(0.9)\times(-0.8)} \\ &= \frac{0.93088}{1.06912} = 0.87, \end{aligned} \tag{2.24}$$

where all the extrinsic probabilities associated with the non-zero PCM entries were taken into account, except for the entry $P^1_{3,12}$. Similarly, we have

$$\begin{aligned} LR_{7,12} &= \frac{1+(1-2P^1_{7,7})(1-2P^1_{7,10})(1-2P^1_{7,15})}{1-(1-2P^1_{7,7})(1-2P^1_{7,10})(1-2P^1_{7,15})} \\ &= \frac{1+(0.1)\times(0.6)\times(-0.8)}{1-(0.1)\times(0.6)\times(-0.8)} \\ &= \frac{0.952}{1.048} = 0.908; \end{aligned} \tag{2.25}$$

$$\begin{aligned} LR_{10,12} &= \frac{1+(1-2P^1_{10,4})(1-2P^1_{10,9})(1-2P^1_{10,11})}{1-(1-2P^1_{10,4})(1-2P^1_{10,9})(1-2P^1_{10,11})} \\ &= \frac{1+(-0.9)\times(0.88)\times(0.98)}{1-(-0.9)\times(0.86)\times(0.98)} \\ &= \frac{0.22384}{1.77616} = 0.126. \end{aligned} \tag{2.26}$$

All these LR values are summarised in Table 2.12. Then the calculated message $LR_{i,j}$ is used for the sake of updating the probability ratio $PR_{i,j}$ with the aid of Equation 2.13 to carry out the 4^{th}

step of Algorithm 1. For example, we have:

$$\begin{aligned} PR_{3,12} &= \frac{1-P_{12}^1}{P_{12}^1} \prod_{k\in\{R_{12}\},k\neq 3} LR_{k,12} \\ &= \frac{1-0.4}{0.4} \times LR_{7,12} \times LR_{10,12} \\ &= 1.5 \times 0.908 \times 0.126 = 0.1716, \end{aligned} \tag{2.27}$$

where all the non-zero entries of the 12^{th} column in Table 2.11 were taken into account, except for $LR_{3,12}$, because the factor $\frac{1-P_{12}^1}{P_{12}^1}$ already takes into account the intrinsic information available for bit 11 of the codeword. Similarly, we have

$$\begin{aligned} PR_{7,12} &= \frac{0.6}{0.4} \times LR_{3,12} \times LR_{10,12} \\ &= 1.5 \times 0.87 \times 0.126 = 0.1644; \end{aligned} \tag{2.28}$$

$$\begin{aligned} PR_{10,12} &= \frac{0.6}{0.4} \times LR_{3,12} \times LR_{7,12} \\ &= 1.5 \times 0.87 \times 0.908 = 1.185. \end{aligned} \tag{2.29}$$

The corresponding PR values are summarised in Table 2.13.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	0.177	4.27		0.234					4.48		3.62				
2		1.1				1.1	2.5		1.1	1.153					
3			0.972				0.757					0.87		0.97	1.035
4	0.933		1.06		1.2			0.757		1.096					
5			1.1		1.368			0.621					0.9	1.11	
6						1.428		0.115			1.387		0.700		
7							0.825			0.968		0.9084			1.025
8					4.673								0.645	1.55	0.609
9	0.157	4.672		0.214		4.673									
10				1.426					0.700		0.727	0.126			

Table 2.12: Likelihood ratio value $LR_{i,j}$ of each non-zero entry calculated using Equation 2.12 after the first iteration according to the PCM seen in Table 2.5 following step 3 of Algorithm 1.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	0.016	100		0.016					12.05		99.78				
2		333.33				125	0.763		50	4.255					
3			58.82				2.512					0.172		32.258	0.07
4	0.0031		52.63		11.905			0.048		4.464					
5			50		10.42			0.172					0.024	28.57	
6						100		0.314			262.52		0.03		
7							2.31			5.05		0.165			0.07
8					3.058								0.033	20.41	0.118
9	0.018	90.91		0.017		30.3									
10				0.0026					76.92		501	1.186			

Table 2.13: Probability ratio value $PR_{i,j}$ of each non-zero entry calculated using Equation 2.13 after the first iteration according to the PCM seen in Table 2.5 following step 4 of Algorithm 1.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	0.984	0.01		0.984					0.076		0.01				
2		0.003				0.008	0.567		0.02	0.191					
3			0.017				0.284					0.854		0.03	0.935
4	0.997		0.019		0.078			0.955		0.183					
5			0.019		0.087			0.945					0.977	0.034	
6						0.01		0.761			0.004		0.97		
7							0.302			0.165		0.0.859			0.935
8					0.246								0.968	0.047	0.895
9	0.982	0.011		0.983		0.032									
10				0.997					0.013		0.002	0.457			

Table 2.14: Probability values of entry (i, j) indicating that the j^{th} bit is a transmitted binary 1 after one iteration according to the PCM seen in Table 2.5, which were calculated from the values tabulated in Table 2.13 following step 6 of Algorithm 1.

Bit	$PR(x_d)$	Bit	$PR(x_d)$	Bit	$PR(x_d)$
x_1	0.00285	x_6	139.466	x_{11}	365
x_2	416.94	x_7	1.9079	x_{12}	0.149
x_3	55.534	x_8	0.036	x_{13}	0.021
x_4	0.00375	x_9	54.055	x_{14}	31.71
x_5	14.245	x_{10}	4.893	x_{15}	0.07

Table 2.15: *A posteriori* probability ratio of each of the 15 encoded bits after the first iteration calculated from Equation 2.14 following step 5 of Algorithm 1.

The above two steps accomplish the information passing from the message nodes to the check nodes and back to message nodes, which is the result of the first LDPC iteration. Following the above worked examples, all the elements of Table 2.12 and 2.13 may be calculated, which constitute the result of updating the LR and PR values of each individual non-zero entry in the PCM for the first iteration by using Equations 2.12 and 2.13, respectively. Furthermore, the probability values of entry (i, j) calculated from the corresponding probability ratio, which indicate the chances that the j^{th} bit is a transmitted binary 1 after the first iteration are summarised in Table 2.14. After each iteration, as described at the 5^{th} step of Algorithm 1, the likelihood ratios $LR_{i,j}$ summarised in Table 2.12 are utilised for generating the *a posteriori* probability ratio according to Equation 2.14. For example, using the LR values seen in column 12 of Table 2.12, we have:

$$\begin{aligned} PR(x_{12}) &= P_r(x_{12}) \times LR_{3,12} \times LR_{7,12} \times LR_{10,12} \\ &= 1.5 \times 0.87 \times 0.908 \times 0.126 = 0.149. \end{aligned} \tag{2.30}$$

These values are listed in Table 2.15. Furthermore, at step 6 of Algorithm 1, the $PR_{i,j}$ values obtained after step 4 of Algorithm 1 are used for updating the value of $P^1_{i,j}$, as seen in Table 2.14. The *a posteriori probability* generated by the decoder at its output is then fed to the decision making decoder stage for hard decision and the corresponding hard decision decoded codeword will be tested with the aid of the parity check matrix $\mathbf{H}_r$. If the result of this verification stage using $\mathbf{C}\cdot\mathbf{H}_r^T$ is not an all zero vector, then the decoded codeword is not a legitimate one and hence the operations described above will be carried out for another iteration. It can be observed that the 12^{th} coded bit, which was

originally in error in Table 2.9, has an *a posteriori* probability ratio of 0.149 in Table 2.15, where the probability ratio is defined as the ratio of the probability of the transmitted bit being a binary zero over the probability that the bit was a binary one, under the assumption that a zero and a one have an equal probability of occurrence. Therefore the threshold for flipping a bit is one. More explicitly, when the probability ratio becomes lower than the threshold, we flip the bit from a binary zero to a binary one and vice versa. Hence this bit is flipped to a binary one and we can compare with Table 2.9 and see that this erroneous bit has been corrected.

As for the 7^{th} step of Algorithm 1, upon carrying out a hard decision based on the values seen in Table 2.15 and calculated during the first iteration, it can be seen that the error at position 12 has been detected and corrected. By multiplying the hard decision based results of the decoded bit sequence with the transpose of the PCM $\mathbf{H}_r$ of Table 2.5, the product becomes an all-zero vector. This indicates that the obtained codeword is legitimate, thus the iterations are concluded. Hereby, step 8 is carried out for outputting a legitimate codeword and the decoding process is accomplished. If, however, the resultant product is not an all-zero vector, this will indicate that the decoded codeword is not a legitimate one, thus a further iteration will be carried out as described at step 9 in Algorithm 1, provided that the maximum affordable number of iterations has not been exhausted. The values listed in Table 2.14 will be used for the remaining operations in the same fashion, as described previously.

2.7 Generalised LDPC decoding procedure

In the previous section, a worked decoding example was provided, using Gallager's notation. However, during the past decade, numerous researchers have been working on LDPC codes, and resulting in further advances, such as non-binary LDPC [55] or reduced complexity decoding [9]. In his research, Gallager used the *Probability Ratio* and *Likelihood Ratio* of bits, assuming a binary coding and modulation scenario. As seen in Appendix A, the updating of the *likelihood ratio* is carried out by calculating the probability of encountering an even number of logical 1s in a parity check equation. In order to avoid these limitations in this section a more general description of LDPC decoding will be given. This generalised description will be cross-referenced with our previous notation according to Gallager's original work.

2.7.1 Generalised notation

The iterative decoding procedure concerned involves passing probabilities between the non-zero entries within the parity check matrix. Since the decoding information is always passed from a column to a row or vice versa, the probabilities are circulating among the message nodes and the check nodes of the bipartite graph defined in Figure 2.3, where we defined the columns as message nodes and the rows as check nodes. Using the example seen in Figure 2.3, we arrive at the more detailed Figure 2.5, indicating the exchange of information among the nodes.

To elaborate further in Figure 2.5, the message denoted as $Q^a_{i,j}$ is passed from the j^{th}, $j = 1 \dots N$, message node on the left to the i^{th}, $i = 1 \dots M$, check node seen at the right. More explicitly, $Q^a_{i,j}$ represents the probability passed from the message node j to the check node i indicating that the message node j is in state $a \in (0, 1)$. The quantity $Q_{i,j}$ is similar to the $PR_{i,j}$ quantity defined in

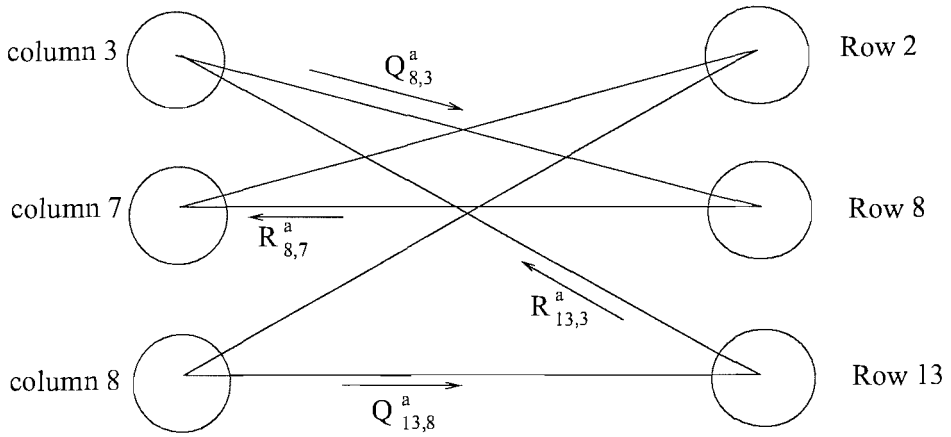


Figure 2.5: Bipartite graph of message passing using the structure of Figure 2.3.

Equation 2.13 and exemplified in Table 2.10 and 2.13, except that $PR_{i,j}$ was expressed in a form of ratio. Hence we have $PR_{i,j} = Q^0_{i,j}/Q^1_{i,j}$. The subscript of $Q^a_{i,j}$ denotes the row and column index of the corresponding non-zero entry in the parity check matrix, which is related to a specific connection between the two nodes in Fig 2.5. By contrast, the message denoted as $R^a_{i,j}$ is passed from the i^{th} check node to the j^{th} message node, quantifying the probability that the i^{th} check is satisfied based on the probability of all the participating nodes in the i^{th} check except node j, which is in state a. Hence $R^a_{i,j}$ corresponds to the LR defined in Equation 2.12, and we have $LR_{i,j} = R^0_{i,j}/R^1_{i,j}$.

The notations $Q^a_{i,j}$ and $R^a_{i,j}$ have a similar meaning to $PR^a_{i,j}$ and $LR^a_{i,j}$ respectively, and these quantities are stored in the non-zero entries of the PCM in a fashion similar to Table 2.13 and Table 2.12, respectively. At the beginning of the iterations, as the $PR_{i,j}$ of each non-zero entry is initialised to the value of the intrinsic probability ratio, i.e. to the values seen in the 4^{th} column of Table 2.9 for example, $Q^a_{i,j}$ is initialised to P^a_j, which is the *intrinsic* probability that the j^{th} symbol assumed a transmitted binary value of a. If no *a priori* or independent statistical knowledge is available concerning this bit, encountering any of the possible states will have the same probability. Since the message $R^a_{i,j}$ quantifies the probability that the i^{th} check is satisfied, when the j^{th} symbol is in state a, it can be represented using the following expression [56]:

$$R^a_{i,j} = \sum_{\mathbf{C}:c_j=a} P(z_i = 0|\mathbf{C}) \prod_{k\in\{C_i\},k\neq j} Q^{c_k}_{i,k}. \quad i = 1\ldots M, j = 1\ldots N. \tag{2.31}$$

The notations c_j and C_j in Equation 2.31 are used to represent the j^{th} bit of the codeword $\mathbf{C}$, and the set of column indices of the j^{th} row of the PCM, respectively. Equation 2.31 provides a recipe as to how the message $R^a_{i,j}$ is updated, which is interpreted in more detail below. From this equation, we can see that we have to carry out the summation for all legitimate codewords, where the j^{th}, $j = 1\ldots N$, symbol is in state a and which satisfies the i^{th}, $i = 1\ldots M$, parity check z_i. More explicitly, the probability $P(z_i = 0|\mathbf{C})$ is 1, if the testing of the codeword configuration $\mathbf{C}$ satisfies the i^{th} check z_i and 0 otherwise. In other words, $P(z_i = 0|\mathbf{C})$ is a binary flag, which returns a value of 1, if $\mathbf{C}$ is a legitimate codeword and zero otherwise. The flag $P(z_i = 0|\mathbf{C})$ also has to be multiplied with the product of all the probabilities $Q^{x_k}_{i,k}$ of the message node k being in state c_k for the i^{th} check. Finally, the set $\{C_i\}$ contains the column indices of all the non-zero entries participating in the i^{th} row of the PCM.

Updating $Q_{i,j}^a$ is quite similar, obeying the following equation [56]:

$$Q_{i,j}^a = \alpha_{i,j} P_j^a \prod_{k \in \{R_j\}, k \neq i} R_{k,j}^a, \tag{2.32}$$

where $\{R_j\}$ denotes the set of row indices that have non-zero entries in column j, $j = 1 \ldots N$, of the PCM. Furthermore, P_j^a is the *intrinsic* probability associated with the channel output assuming that C_j is in state a, and the scaling factor $\alpha_{i,j}$ is used for ensuring that we have $\sum_a Q_{i,j}^a = 1$.

The decoding process is based on simply iteratively updating the values of $R_{i,j}^a$ and $Q_{i,j}^a$ and after each update of $R_{i,j}^a$ and $Q_{i,j}^a$, a tentative hard decision will be made for the sake of determining the polarity of the product of $R_{i,j}^a$ and each symbol's *intrinsic* probability, i.e. that of

$$P(C_j) = P_j^a \prod_{k \in \{R_j\}} R_{k,j}^a. \tag{2.33}$$

The specific binary symbol having the higher probability from the set of two will be chosen as the survivor. If the hard-decision based codeword satisfies the parity check matrix $\mathbf{H}_r$, the decoding operations are terminated and the corresponding codeword is output. Otherwise, the decoder will carry on updating the quantities $R_{i,j}^a$ and $Q_{i,j}^a$ according to Equation 2.31 and Equation 2.32, until a valid codeword is found or a predetermined maximum number of iterations is reached. The flow diagram of the iterative decoding process is portrayed in Figure 2.6.

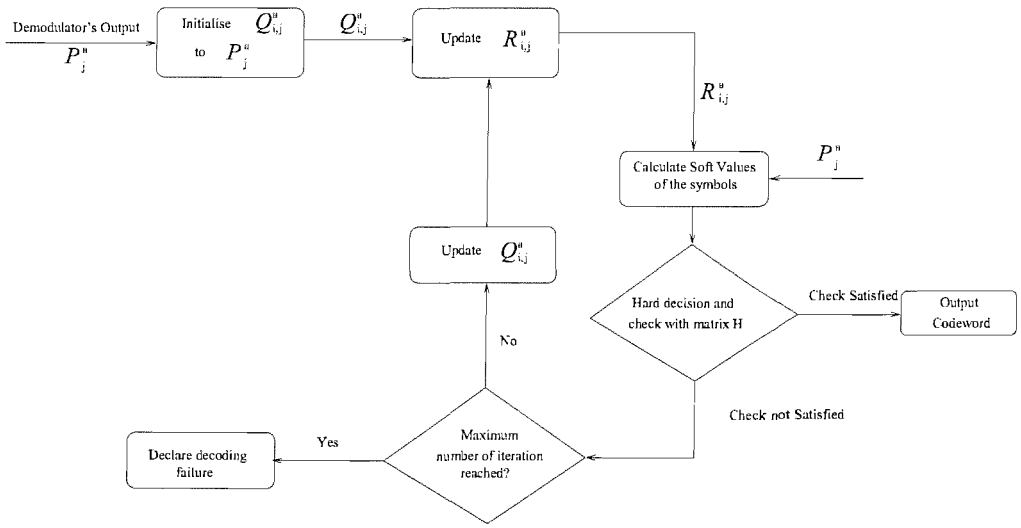


Figure 2.6: LDPCC's probabilistic decoding flow diagram

2.7.2 Reduced complexity calculation of the message $R_{i,j}^a$

As seen in Equation 2.31, the updating of the message $R_{i,j}^a$ involves tentatively evaluating all the legitimate codewords that have their j^{th} bit in state a. This is a rather complex operation and may become prohibitively complex to implement, when the block-length is high. Hence Gallager's [1] suggestion outlined in this thesis' Appendix A was the introduction of a reduced complexity method. With the same objective in mind, Richardson proposed an efficient way of updating $R_{i,j}^a$ using the

Fast Fourier Transform (FFT) [9] , which may be further generalised for employment in non-binary LDPC codes, as shown in Chapter 4.

In Equation 2.31, the quantity we have to find is the probability of the j^{th} coded bit being in state a by exploiting with the knowledge of the other coded bits having a certain probability and upon assuming that the i^{th} parity check equation is satisfied. Let us use the notation $PDF_{i,j}$ for representing the probability density function of the j^{th} bit stored in the non-zero entry of the PCM at position (i,j), where we have $PDF_{i,j} = \{P^0_{i,j}, P^1_{i,j}\}$. Our aim is now to calculate the probability density function $PDF_{i,j}$ of the non-zero PCM entry at position (i,j). All the non-zero entries in the i^{th} row are modulo-2 added after a tentative hard-decision for the sake of testing, whether a legitimate codeword was found. Therefore the probability density function of the j^{th} LDPC coded bit has to match the rest of the bits participating in this check for the sake of ensuring that their modulo-2 sum equals zero. In other words, the state of the j^{th} bit has to be the same as the modulo-2 sum of all other bits involved in this check. Thus the probability density function of the j^{th} LDPC coded bit may be obtained by evaluating the convolution of the probability density function of all other bits involved in the parity check equation, which is expressed as [56]:

$$PDF_{i,j} = \bigotimes_{t\in\{C_i\},t\neq j} PDF_{i,t} \tag{2.34}$$

where $\bigotimes$ denotes convolution. However, the convolution may become a high-complexity operation.

The associated complexity may be reduced, if the operations are first Fourier-transformed, multiplied and then inverse Fourier-transformed. These operations may be carried out more efficiently using the Fast Fourier Transform and Inverse Fast Fourier Transform [113] over the corresponding finite Galois field, namely GF(2). More explicitly, Equation 2.34 may be formulated in the frequency domain as:

$$\mathcal{F}(PDF_{i,j}) = \prod_{t\in\{C_i\},t\neq j} \mathcal{F}(PDF_{i,t}). \tag{2.35}$$

Let us introduce the function f, representing a function having abscissa values defined over GF(2). More explicitly, let $f(0)$ and $f(1)$ represent the probability of a particular bit being a binary zero and a binary one, respectively. Furthermore, let us use $\mathcal{F}(f)$ to represent the Fourier transform of f, which is defined as [114]:

$$\mathcal{F}(f)(k) = \sum_{n=0}^{2-1} f(n)e^{-j2\pi kn/2}, \quad k = 0, 1. \tag{2.36}$$

Therefore, we arrive at

$$\begin{aligned} \mathcal{F}(f)(0) &= f(0) + f(1) = 1, \\ \mathcal{F}(f)(1) &= f(0) - f(1). \end{aligned} \tag{2.37}$$

Let $f_{i,j} = PDF_{i,j}$, $f_{i,t} = PDF_{i,t}$ and substitute these functions into Equation 2.35, yielding

$$\begin{aligned} \mathcal{F}(f_{i,j})(0) &= \prod_{t\in\{C_i\},t\neq j} \mathcal{F}(f_{i,t})(0), \\ \mathcal{F}(f_{i,j})(1) &= \prod_{t\in\{C_i\},t\neq j} \mathcal{F}(f_{i,t})(1). \end{aligned} \tag{2.38}$$

Apparently, the function $\mathcal{F}(f_{i,j})(0)$ in Equation 2.38 equals one, since we have $\mathcal{F}(f)(0) = f(0) + f(1) = 1$ in Equation 2.37. By carrying out the inverse FFT of the results obtained in Equation 2.38, the value of $R^a_{i,j}$ is obtained. The inverse FFT requires exactly the same operation as those seen in Equation 2.37, although an additional normalisation factor has to be incorporated, yielding [56]:

$$\begin{aligned} f(0) &= \frac{\mathcal{F}(f)(0) + \mathcal{F}(f)(1)}{2} = 0.5 + \frac{\mathcal{F}(f)(1)}{2}, \\ f(1) &= \frac{\mathcal{F}(f)(0) - \mathcal{F}(f)(1)}{2} = 0.5 - \frac{\mathcal{F}(f)(1)}{2}. \end{aligned} \tag{2.39}$$

Let us now illustrate that this method results in the same formulation as Gallager's original work. Let us introduce the notation of $m_{i,j} = \log((Q^0_{i,j}/Q^1_{i,j}))$. Then we have

$$Q^0_{i,j} - Q^1_{i,j} = \frac{Q^0_{i,j} - Q^1_{i,j}}{Q^0_{i,j} + Q^1_{i,j}} = \frac{Q^0_{i,j}/Q^1_{i,j} - 1}{Q^0_{i,j}/Q^1_{i,j} + 1} = \frac{e^{m_{i,j}} - 1}{e^{m_{i,j}} + 1} = \tanh(m_{i,j}/2), \tag{2.40}$$

and hence Equation 2.38 may be rewritten as

$$\mathcal{F}(f_{i,j})(0) = 1 \tag{2.41}$$

$$\mathcal{F}(f_{i,j})(1) = \prod_{t \in \{C_i\}, t \neq j} \tanh \frac{m_{i,t}}{2} \tag{2.42}$$

By taking the inverse FFT of Equations 2.41 and 2.42, $R^a_{i,j}$ of Equation 2.31 may be expressed as:

$$\begin{aligned} R^0_{i,j} &= \frac{\mathcal{F}(f_{i,j})(0) + \mathcal{F}(f_{i,j})(1)}{2} = (1 + \prod_{t \in \{C_i\}, t \neq j} \tanh \frac{m_{i,t}}{2}) \cdot 0.5, \\ R^1_{i,j} &= \frac{\mathcal{F}(f_{i,j})(0) - \mathcal{F}(f_{i,j})(1)}{2} = (1 - \prod_{t \in \{C_i\}, t \neq j} \tanh \frac{m_{i,t}}{2}) \cdot 0.5. \end{aligned} \tag{2.43}$$

As in Gallager's LR format, where $LR_{i,j} = R^0_{i,j}/R^1_{i,j}$ holds, we have

$$LR_{i,j} = R^0_{i,j}/R^1_{i,j} = \frac{1 + \prod_{t \in \{C_i\}, t \neq j} \tanh \frac{m_{i,t}}{2}}{1 - \prod_{t \in \{C_i\}, t \neq j} \tanh \frac{m_{i,t}}{2}}. \tag{2.44}$$

Since we have $\tanh(m_{i,j}/2) = Q^0_{i,j} - Q^1_{i,j} = 1 - 2Q^1_{i,j} = 1 - 2P^1_{i,j}$, we arrive at Equation 2.12, resulting in Gallager's original representation of the LDPC decoding formulation.

As introduced in Section 2.7.1, $Q^1_{i,j}$ corresponds to Gallager's original notation $P_{i,j}$, as introduced in Section 2.6, hence the values seen in Table 2.11 can be directly used for the sake of computing the message $R^a_{i,j}$. Observing Equation 2.26, the message $LR_{10,12}$ was calculated as $(1 - 0.77616)/(1 + 0.77616) = 0.126$. Let us now provide a brief example of updating the message $R^a_{10,2}$ with the aid of the FFT and show that it is consistent with the approach demonstrated in Section 2.6.

In order to calculate $R^a_{10,2}$, we have carried out the FFT for all the probabilities of the non-zero

entries participating in the 10^{th} row of Table 2.11 using Equation 2.37 as follows:

$$\begin{aligned}
\mathcal{F}(f_{10,4})(0) &= f_{10,4}(0) + f_{10,4}(1) = 1, \\
\mathcal{F}(f_{10,4})(1) &= f_{10,4}(0) - f_{10,4}(1) = -0.9; \\
\mathcal{F}(f_{10,9})(0) &= f_{10,9}(0) + f_{10,9}(1) = 1, \\
\mathcal{F}(f_{10,9})(1) &= f_{10,9}(0) - f_{10,9}(1) = 0.88; \\
\mathcal{F}(f_{10,11})(0) &= f_{10,11}(0) + f_{10,11}(1) = 1, \\
\mathcal{F}(f_{10,11})(1) &= f_{10,11}(0) - f_{10,11}(1) = 0.98; \\
\mathcal{F}(f_{10,12})(0) &= f_{10,12}(0) + f_{10,12}(1) = 1, \\
\mathcal{F}(f_{10,12})(1) &= f_{10,12}(0) - f_{10,12}(1) = 0.2.
\end{aligned} \tag{2.45}$$

Upon multiplying the transformed values obtained with the aid of Equation 2.45, the Fourier transform of the PDF of entry $(10, 12)$ can be updated as seen in Equation 2.38, yielding:

$$\begin{aligned}
\mathcal{F}(f_{10,12})(0) &= \prod_{t\in\{C_i\},t\neq 12} \mathcal{F}(f_{10,t})(0) = \mathcal{F}(f_{10,4})(0) \times \mathcal{F}(f_{10,9})(0) \times \mathcal{F}(f_{10,11})(0) \\
&= 1 \times 1 \times 1 = 1; \\
\mathcal{F}(f_{10,12})(1) &= \prod_{t\in\{C_i\},t\neq 12} \mathcal{F}(f_{10,t})(1) = \mathcal{F}(f_{10,4})(1) \times \mathcal{F}(f_{10,9})(1) \times \mathcal{F}(f_{10,11})(1) \\
&= (-0.9) \times 0.88 \times 0.98 = -0.77616.
\end{aligned} \tag{2.46}$$

The results obtained according to Equation 2.46 have to be inverse Fourier transformed for the sake of generating the PDF of the non-zero entry $(10, 2)$ of Table 2.11 using Equation 2.39, yielding:

$$\begin{aligned}
f_{10,12}(0) &= \frac{\mathcal{F}(f_{10,12})(0) + \mathcal{F}(f_{10,12})(1)}{2} = \frac{1 - 0.77616}{2} = 0.11192, \\
f_{10,12}(1) &= \frac{\mathcal{F}(f_{10,12})(0) - \mathcal{F}(f_{10,12})(1)}{2} = \frac{1 + 0.77616}{2} = 0.88808.
\end{aligned} \tag{2.47}$$

Thus the results obtained in Equation 2.47 can be assigned to $R^a_{10,2}$ as follows:

$$\begin{aligned}
R^0_{10,12} &= f_{10,12}(0) = 0.11192, \\
R^1_{10,12} &= f_{10,12}(1) = 0.88808,
\end{aligned} \tag{2.48}$$

yielding $LR_{10,12} = R^0_{10,12}/R^1_{10,12} = 0.126$ as according to Equation 2.44, which is consistent with the value previously tabulated in Table 2.12.

2.7.3 Complexity of the LDPC decoder

As described in Section 2.7.2, the complexity of the LDPC decoder can be reduced with the aid of the FFT. In this subsection, the complexity of the LDPC decoder will be calculated in terms of the number of additions and multiplications required.

During the process of updating the message $R^a_{i,j}$, the FFT process of Equation 2.37 requires two additions. When the FFT has been carried out, the multiplications entailed in the evaluation of Equation 2.38 are carried out. Equation 2.38 may be more efficiently evaluated by using *forward and backward* multiplications [56]. A simple example is given as follows.

Operation	Value of *temp*	Value of updated entry
Initialise $temp$=1	1	n/a
$A_{new} = temp$	1	A_{new}=1
$temp = temp \times A_{old}$	A_{old}	
$B_{new} = temp$	A_{old}	B_{new}=A_{old}
$temp$=$temp \times B_{old}$	$A_{old} \times B_{old}$	n/a
$C_{new} = temp$	$A_{old} \times B_{old}$	C_{new}=$A_{old} \times B_{old}$
$temp$=$temp \times C_{old}$	$A_{old} \times B_{old} \times C_{old}$	n/a
$D_{new} = temp$	$A_{old} \times B_{old} \times C_{old}$	$D_{new} = A_{old} \times B_{old} \times C_{old}$
$temp$=$temp \times D_{old}$	$A_{old} \times B_{old} \times C_{old} \times D_{old}$	n/a
Reset $temp$=1	1	n/a
$D_{new} = temp \times temp$	1	D_{new}=$A_{old} \times B_{old} \times C_{old}$
$temp$ =$temp \times D_{old}$	D_{old}	n/a
$C_{new} = temp \times temp$	D_{old}	$C_{new} = A_{old} \times B_{old} \times D_{old}$
$temp\ temp \times = C_{old}$	$C_{old} \times D_{old}$	n/a
B_{new} =$temp \times temp$	$C_{old} \times D_{old}$	$B_{new} = A_{old} \times C_{old} \times D_{old}$
$temp$ =$temp \times B_{old}$	$B_{old} \times C_{old} \times D_{old}$	n/a
$A_{new} = temp \times = temp$	$B_{old} \times C_{old} \times D_{old}$	$A_{new} = B_{old} \times C_{old} \times D_{old}$
$temp\ temp \times = A_{old}$	$A_{old} \times B_{old} \times C_{old} \times D_{old}$	n/a

Table 2.16: Process of forward and backward multiplications for a vector having four entries.

Let us assume that we have a vector containing four values denoted as $\{A_{old}, B_{old}, C_{old}, D_{old}\}$. We have to replace the value of each entry in the vector with a product of other values within the vector. More explicitly, we have

$$\begin{aligned} A_{new} &= B_{old} \cdot C_{old} \cdot D_{old}, \\ B_{new} &= A_{old} \cdot C_{old} \cdot D_{old}, \\ C_{new} &= A_{old} \cdot B_{old} \cdot D_{old}, \\ D_{new} &= A_{old} \cdot B_{old} \cdot C_{old}. \end{aligned} \tag{2.49}$$

The evaluation of Equation 2.49 requires $(w_r - 2) \cdot w_r$ multiplications, where w_r is the number of entries within the vector, i.e. four in this example. We can observe that the complexity increases dramatically, when w_r is high in Equation 2.49. We will now show that the *forward and backward* multiplications require $3w_r$ multiplications, a complexity which increases linearly with respect to the size of the vector. Using *forward and backward* multiplications will reduce the arithmetic complexity for $w_r > 5$, which is applicable for regular LDPC codes having a coding rate in excess of 0.4 and for most of the irregular LDPC codes introduced in Chapter 3. The process of updating values in the vector of Equation 2.49 is tabulated in Table 2.16, where a temporary variable denoted as *temp* is introduced.

By summing the number of multiplication operations in Table 2.16, a total of $3w_r$ multiplications are invoked for calculating state a. For the binary case, where a has two legitimate states, a total of $6w_r$ multiplications are required.

Similarly to the FFT operation, the IFFT operation of Equation 2.39 requires a further two additions. Furthermore, owing to the normalisation process of the IFFT another two multiplications are necessary.

Thus by summing all the operations to be evaluated for a single row of the PCM having w_r non-zero entries, we require $4w_r$ additions and $8w_r$ multiplications for updating message R for each individual row. By summing all the M rows, we have a total of $4Mw_r$ number of additions and $8Mw_r$ multiplications. Since we have $M \times w_r = N \times w_c$, where w_c is the column weight, the number of additions and multiplications can be represented by $4N \times w_c$ and $8N \times w_c$, respectively.

The update of message Q will involve another *forward and backward* multiplication, thus again, each column will require $6w_c$ multiplications. Hence the overall decoding complexity associated with the detection of each coded bit in one iteration will be $4w_c$ additions and $14w_c$ multiplications.

2.8 Theoretical performance bound

Gallager stated [1] that a mathematical analysis of the performance of probabilistic decoding is extremely challenging, but a weak analytical bound on the error probability can be derived for the specific case of having a column weight of three, which is expressed as [1]:

$$p_{i+1} = p_0 - p_0\left[\frac{1+(1-2p_i)^{w_r-1}}{2}\right]^{w_c-1} + (1-p_0)\left[\frac{1-(1-2p_i)^{w_r-1}}{2}\right]^{w_c-1}. \quad (2.50)$$

In the equation, i is the iteration index, while p_0 is the intrinsic probability of the bit at the position concerned being in error. The second term on the right of the Equation 2.50 is the probability that the bit concerned is received in error, while the parity check is satisfied due to errors at other positions. Finally, the third term represents the probability that the bit concerned is actually received correctly, but due to errors in the parity check set, it was modified to be in error.

Gallager derived Equation 2.50 based on the simplifying assumption that the Binary Symmetric Channel (BSC) was used. When the parity check set is not satisfied then the bit concerned was simply toggled. Based on this simplistic decoding method, this technique is limited to provide a weak bound on the achievable performance, since it is based on hard-decision decoding. When soft-decision based simulations are conducted using Equation 2.11, the attainable performance is almost an order of magnitude better than the results obtained from Equation 2.50.

2.9 Simulation results

In this section our simulation results will be discussed for the sake of characterising the achievable performance of the regular construction binary LDPC codes described in the previous sections, when communicating over both AWGN and uncorrelated Rayleigh fading channels. The various LDPC codes' performance will be benchmarked against that of turbo convolutional codes [3] [115] and Turbo Trellis Coded Modulation (TTCM) [115] [116] . All the parity check matrices of the LDPC codes are constructed in a random fashion. More explicitly, according to the column weight, the

LDPC code	Channel	Maximum number of iterations	LDPCC's column weight
(1000, 500)	AWGN	2, 4, 8, 20, 50, 100	3
(500, 250)	AWGN	2, 4, 8, 20, 50, 100	3
(200, 100)	AWGN	2, 4, 8, 20, 50, 100	3
(1000, 500)	uncorrelated Rayleigh	2, 4, 8, 20, 50, 100	3
(500, 250)	uncorrelated Rayleigh	2, 4, 8, 20, 50, 100	3
(200, 100)	uncorrelated Rayleigh	2, 4, 8, 20, 50, 100	3

Table 2.17: Simulation parameters for three half-rate LDPCCs investigated, when communicating over an AWGN channel and an uncorrelated Rayleigh fading channel using different maximum numbers of iterations.

positions of the non-zero entries are assigned to the PCM randomly, while maintaining a constant row weight at the same time. The number of length 4 cycles is reduced to the lowest possible level.

2.9.1 Effect of the number of LDPC iterations

Similar to the celebrated family of turbo convolutional codes, the benefit of iteratively exchanging soft information is that during the iterative decoding process the performance of the channel decoder improves upon increasing the number of iterations, i.e. the decoder's complexity.

In this subsection, we will use three different half-rate LDPC codes having various block-lengths. The code (N, K) listed in Table 2.17 represents a half-rate LDPC code having a coded block-length of N bits and conveying K information bits. The column weight for each of the three codes seen in Table 2.17 is three. In this experiment an AWGN and the uncorrelated Rayleigh fading channel were applied. We will use **G** and **UR** for abbreviating the AWGN channel and the uncorrelated Rayleigh fading channel in the following tables.

The performance of the LDPC codes listed in Table 2.17 is illustrated in Figures 2.7-2.12. The E_b/N_0 required by each of the LDPC codes for achieving a BER of 10^{-4} when communicating over the two different channels is tabulated in Table 2.18. Furthermore, the performance of the uncoded scenario is used as a benchmarker and the coding gains achieved by using various maximum numbers of iterations for the three different LDPC codes are tabulated in Table 2.19 and plotted in Figure 2.13. The corresponding coding gain versus arithmetic complexity curves are also plotted in Figure 2.14. The complexity quoted in Figure 2.14 characterises the complexity associated with the decoding of each **information** bit, calculated from the complexity figures obtained in Section 2.7.3 and divided by the LDPC code's coding rate. Observe in Figure 2.13 that even though the simulations were conducted for a maximum of 100 iterations, the coding gain started to saturate after an iteration index of 20. Hereby, we introduce another quantity termed as the *iteration efficiency*, which is defined as follows. We take the coding gain achieved when using a maximum of 100 iterations as a reference and define the iteration efficiency as the percentage of the maximum achievable coding gain at a given number of iterations. This quantity is plotted in Figure 2.15.

It may be inferred from Figure 2.13 that the soft information based iterative decoder achieves most

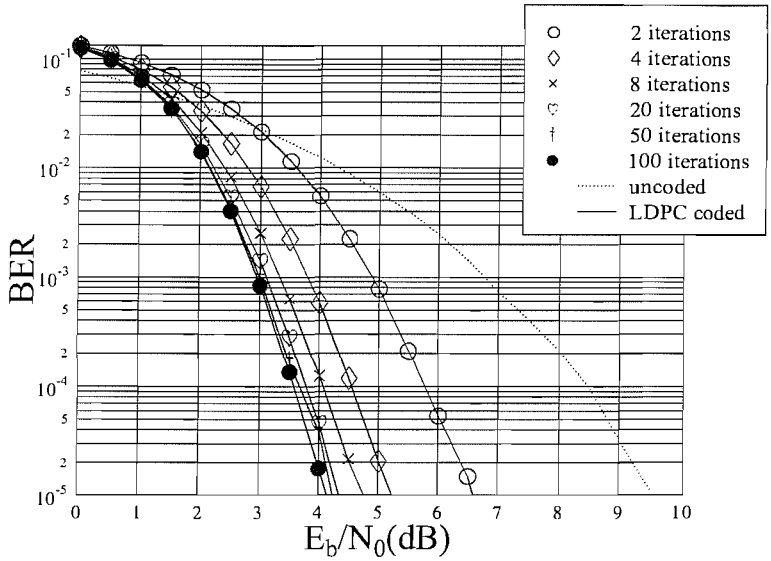


Figure 2.7: BER performance of the half-rate LDPC code (200, 100) parameterised in Table 2.17, using different maximum numbers of iterations, when communicating over an AWGN channel. The achievable coding gain of the various schemes at a BER of 10^{-4} will be summarised in Figure 2.13 and Table 2.27.

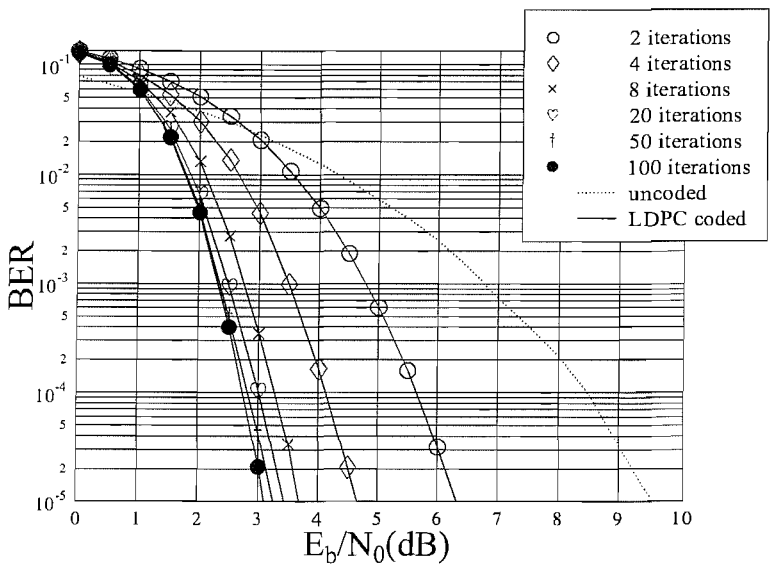


Figure 2.8: BER performance of the half-rate LDPC code (500, 250) parameterised in Table 2.17, using different maximum numbers of iterations, when communicating over an AWGN channel. The achievable coding gain of the various schemes at a BER of 10^{-4} will be summarised in Figure 2.13 and Table 2.27.

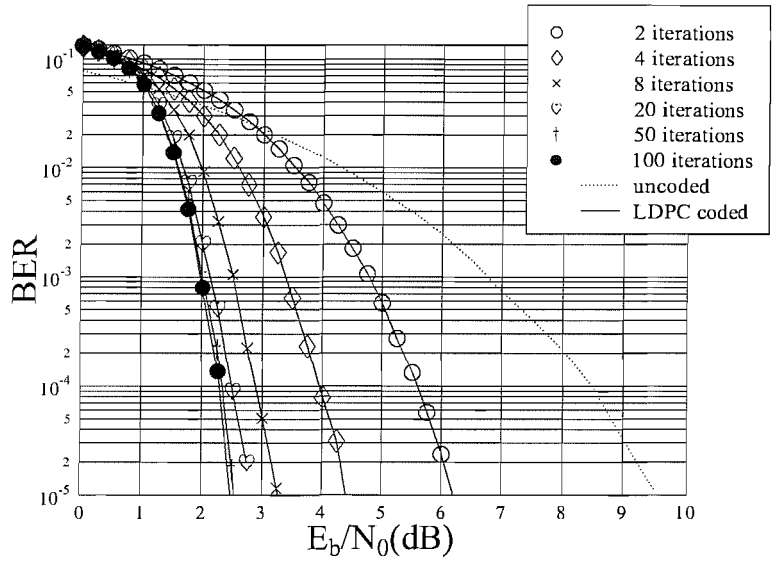


Figure 2.9: BER performance of the half-rate LDPC code (1000, 500) parameterised in Table 2.17, using different maximum numbers of iterations, when communicating over an AWGN channel. The achievable coding gain of the various schemes at a BER of 10^{-4} will be summarised in Figure 2.13 and Table 2.27.

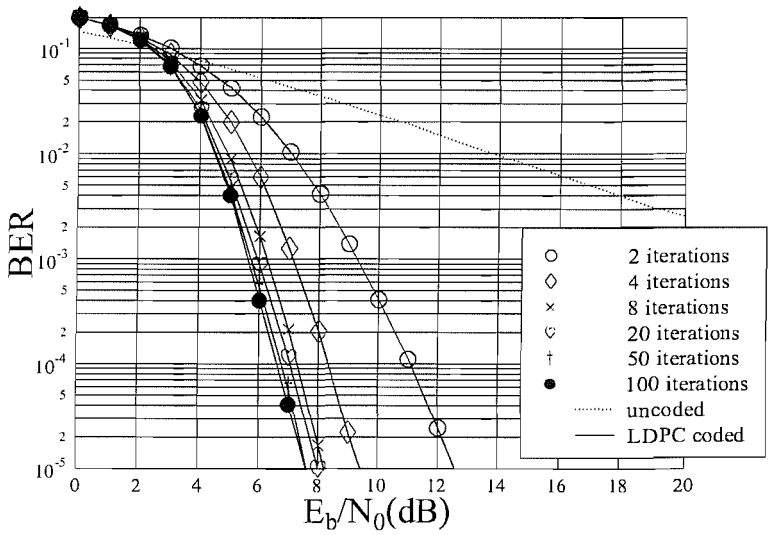


Figure 2.10: BER performance of the half-rate LDPC code (200, 100) parameterised in Table 2.17, using different maximum numbers of iterations, when communicating over an uncorrelated Rayleigh fading channel. The achievable coding gain of the various schemes at a BER of 10^{-4} will be summarised in Figure 2.13 and Table 2.27.

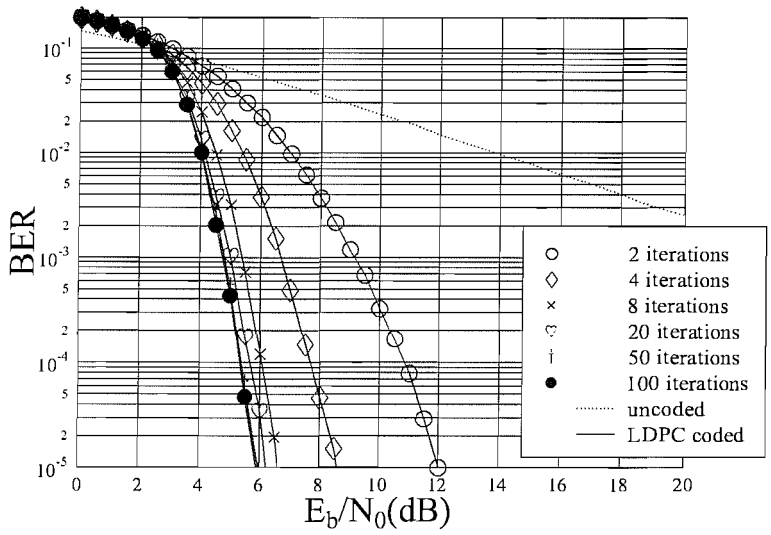


Figure 2.11: BER performance of the half-rate LDPC code (500, 250) parameterised in Table 2.17, using different maximum numbers of iterations, when communicating over an uncorrelated Rayleigh fading channel. The achievable coding gain of the various schemes at a BER of 10^{-4} will be summarised in Figure 2.13 and Table 2.27.

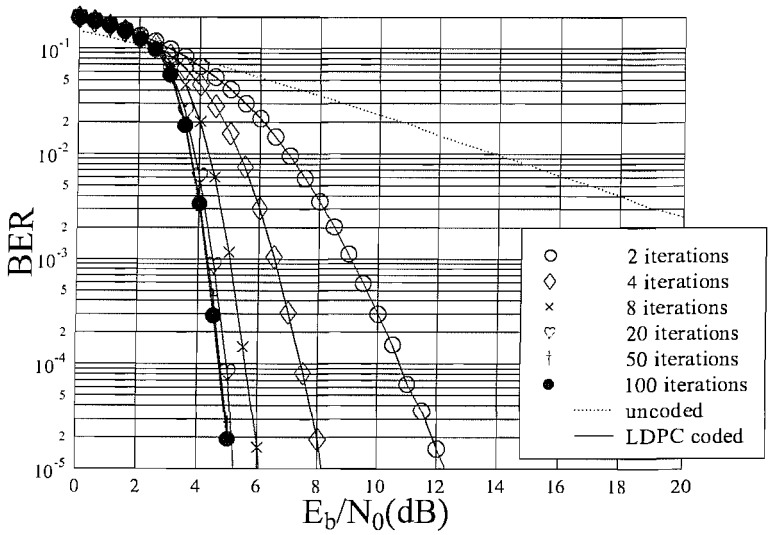


Figure 2.12: BER performance of the half-rate LDPC code (1000, 500) parameterised in Table 2.17, using different maximum numbers of iterations, when communicating over an uncorrelated Rayleigh fading channel. The achievable coding gain of the various schemes at a BER of 10^{-4} will be summarised in Figure 2.13 and Table 2.27.

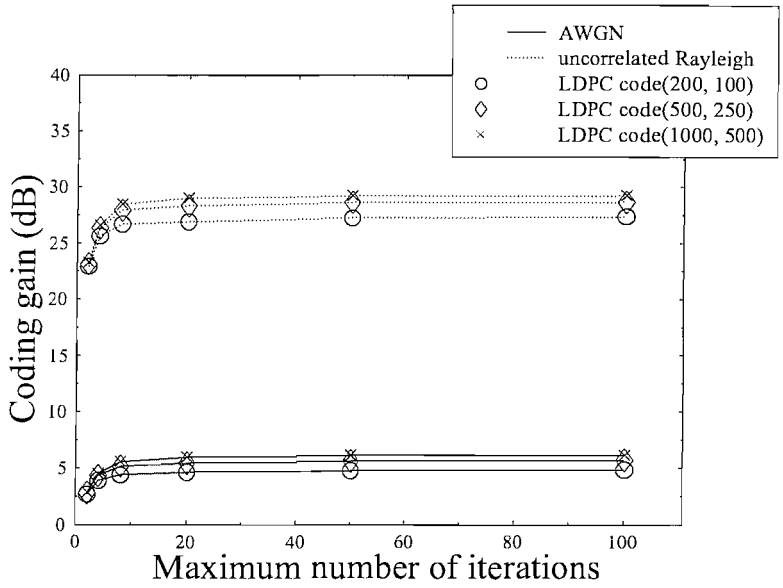


Figure 2.13: Coding gain achieved by the three different half-rate LDPC codes parameterised in Table 2.17, at a BER of 10^{-4}, when communicating over AWGN and uncorrelated Rayleigh fading channels.

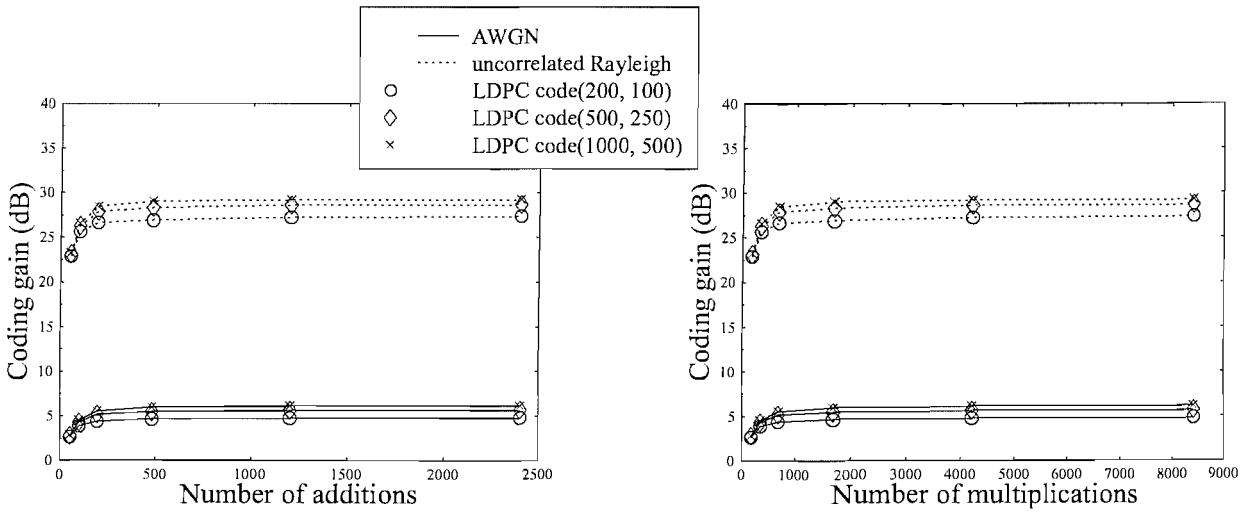


Figure 2.14: Coding gain achieved by the three different half-rate LDPC codes parameterised in Table 2.17, at a BER of 10^{-4}, when communicating over AWGN and uncorrelated Rayleigh fading channels.

LDPC code	Maximum number of iterations	Required E_b/N_0 (**G**) (dB)	Required E_b/N_0 (**UR**) (dB)
(200, 100)	2	5.765	11.059
	4	4.559	8.353
	8	4.059	7.294
	20	3.794	7.059
	50	3.676	6.745
	100	3.588	6.589
(500, 250)	2	5.647	10.824
	4	4.118	7.647
	8	3.265	6.059
	20	3.000	5.647
	50	2.824	5.353
	100	2.735	5.294
(1000, 500)	2	5.588	10.706
	4	3.941	7.412
	8	2.882	5.529
	20	2.470	4.941
	50	2.324	4.764
	100	2.294	4.706
uncoded	N/A	8.47	34

Table 2.18: E_b/N_0 required by the three half-rate LDPC codes parameterised in Table 2.17 for achieving a BER of 10^{-4}, when communicating over both AWGN (**G**) and uncorrelated Rayleigh fading (**UR**) channels.

of its attainable coding gain in the context of both channels after the first few iterations. Further iterations in excess of 20 achieve only a modest further performance improvement at the cost of a high additional decoder complexity. Comparing the curves plotted in Figure 2.15, it can be observed that when communicating over the same type of channel, the iteration efficiency improves at nearly the same rate for the three LDPC codes having different blocklengths. Upon using a maximum of eight iterations, all codes have already achieved over 90% of the maximum attainable coding gain, in the context of the AWGN channel. For the uncorrelated Rayleigh channel the iteration efficiency has already exceeded 95% after eight iterations. By contrast, in the AWGN channel a maximum of 20 iterations is necessitated for achieving over a 95% iteration efficiency, provided that the associated complexity is affordable.

2.9.2 BER as a function of the LDPC bit-index

The family of TTCM schemes [115] [116] employs set partitioning for mapping the input bits to be transmitted to the phaser constellation. Hence each individual bit in a coded symbol has a different BER [115]. Motivated by this observation, in this subsection we briefly investigate whether each coded bit in an LDPC codeword is equally protected. This experiment was carried out by calculating the bit

LDPC code	Maximum no. of iterations	Coding gain (**G**)(dB)	Coding gain (**UR**) (dB)	Iteration efficiency(**G**)	Iteration efficiency(**UR**)
(200, 100)	2	2.705	22.941	55.4%	83.7%
	4	3.911	25.647	80.1%	93.6%
	8	4.411	26.706	90.3%	97.4%
	20	4.676	26.941	95.8%	98.3%
	50	4.794	27.255	98.2%	99.4%
	100	4.882	27.411	100.0%	100.0%
(500, 250)	2	2.823	23.176	49.2%	80.7%
	4	4.352	26.353	75.9%	91.8%
	8	5.205	27.941	90.8%	97.3%
	20	5.470	28.353	95.4%	98.7%
	50	5.646	28.647	98.4%	99.8%
	100	5.735	28.706	100.0%	100.0%
(1000, 500)	2	2.882	23.294	46.7%	79.5%
	4	4.529	26.588	73.3%	90.8%
	8	5.588	28.471	90.4%	97.2%
	20	6.000	29.059	97.1%	99.2%
	50	6.146	29.236	99.5%	99.8%
	100	6.176	29.294	100.0%	100.0%

Table 2.19: Coding gain achieved by the three half-rate LDPC codes parameterised in Table 2.17 at a BER of 10^{-4}, when communicating over both AWGN (**G**) and uncorrelated Rayleigh (**UR**) fading channels.

error ratio at each individual bit position of the entire LDPC codeword evaluated after each LDPC iteration when communicating over an AWGN channel. The simulation parameters used are given in Table 2.20. These experiments are conducted for two specific noise levels, i.e. for $E_b/N_0 = 1.5dB$ and $E_b/N_0 = 2.5dB$.

The simulation results acquired for the LDPC scenarios of Table 2.20 are given in Figure 2.16 and Figure 2.17. The horizontal axis labelled as bit-index gives the bit position concerned in a codeword. As a benefit of iterative decoding, the BER of each constituent bit gradually decreased, as we increased the number of iterations, but no significant BER difference was observed for the various bit positions.

Channel	Maximum number of iterations	LDPCC's column weight	E_b/N_0(dB)	LDPC code
AWGN	5	3	1.5	(1000, 250)
			2.5	

Table 2.20: Simulation parameters for an quarter-rate LDPC code communicating over an AWGN channel at $E_b/N_0 = 1.5$ and $2.5dB$.

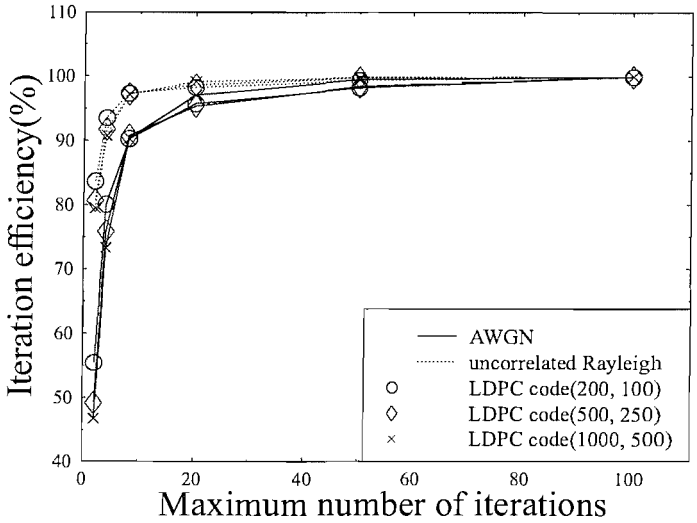


Figure 2.15: Iteration efficiency of the three half-rate LDPC codes of different block-lengths listed in Table 2.19, when communicating over AWGN and uncorrelated Rayleigh fading channels.

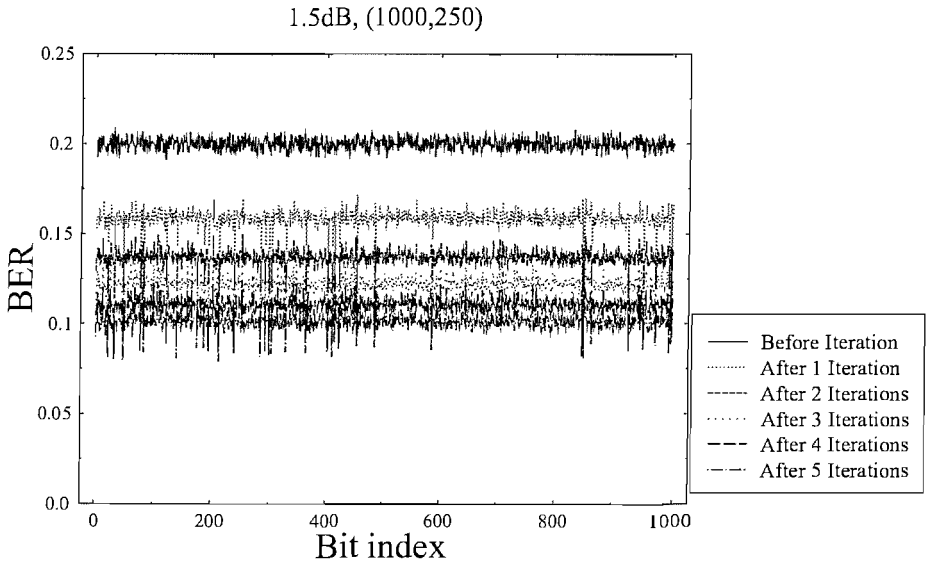


Figure 2.16: BER performance of each individual bit of the quarter-rate LDPC code (1000, 250), when communicating over an AWGN channel at $E_b/N_0 = 1.5dB$.

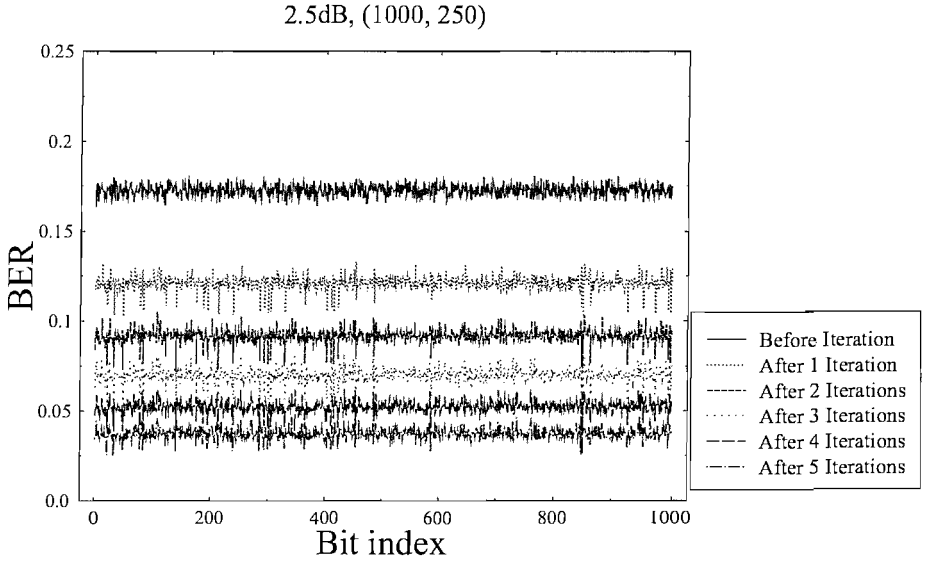


Figure 2.17: BER performance of each individual bit of the quarter-rate LDPC code (1000, 250), when communicating over an AWGN channel at $E_b/N_0 = 2.5dB$.

2.9.3 Probability of undetected errors

Apart from having a good performance, LDPC codes are also capable of error detection with the aid of the PCM. More explicitly, after each iteration and tentative hard-decision decoding, the decoded codeword will be multiplied with the PCM. If the resultant vector is an all-zero vector, the decoder declares the detection of a legitimate codeword and the iterations are terminated. This error detection property of LDPC codes has the potential of reducing the decoding complexity by terminating the decoding process, when the decoder declares successful detection. However, even if the decoded codeword is legitimate, this does not necessarily imply that the decoded codeword is the error-free transmitted codeword, since the original transmitted codeword may have been decoded incorrectly to another legitimate codeword. In this situation the erroneous bits of the codeword cannot be eliminated by further iterations. These errors are referred to as *undetected errors*. By contrast, the bit errors found in the codewords which result in a non-all-zero vector after multiplication by the PCM are termed as *detectable errors*. Undetected errors occur with a higher probability when the minimum Hamming distance of the code is relatively low, since the minimum distance of an LDPC code can increase linearly with the block-length, provided that the minimum column weight is higher than three [2]. We will now demonstrate that for a moderate LDPC coded block-length associated with a uniformly distributed column weight of three, the likelihood of encountering undetected errors is fairly low.

The LDPC codes listed in Table 2.21 were characterised, when communicating over the AWGN channel and the uncorrelated Rayleigh fading channel. We evaluated the achievable Frame Error Ratio (FER) using two different methods. The first evaluation method compared the decoded codeword to the original transmitted codeword on a bit-by-bit basis and hence this method gives the exact

LDPC code	Channel	Maximum number of iterations
(500, 250)	AWGN	25
(200, 100)	AWGN	25
(100, 50)	AWGN	25
(50, 25)	AWGN	25
(500, 250)	uncorrelated Rayleigh	25
(200, 100)	uncorrelated Rayleigh	25
(100, 50)	uncorrelated Rayleigh	25
(50, 25)	uncorrelated Rayleigh	25

Table 2.21: Simulation parameters of four half-rate LDPC codes having different block-lengths, when communicating over AWGN and uncorrelated Rayleigh fading channels.

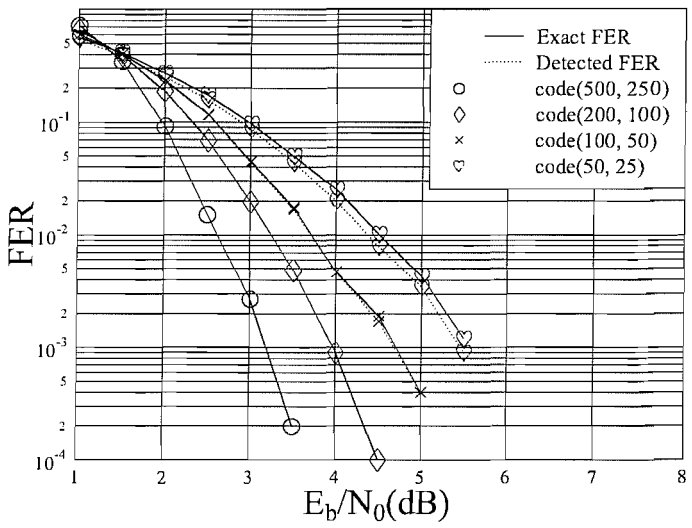


Figure 2.18: FER performance of the four half-rate LDPC codes listed in Table 2.21, when communicating over an AWGN channel.

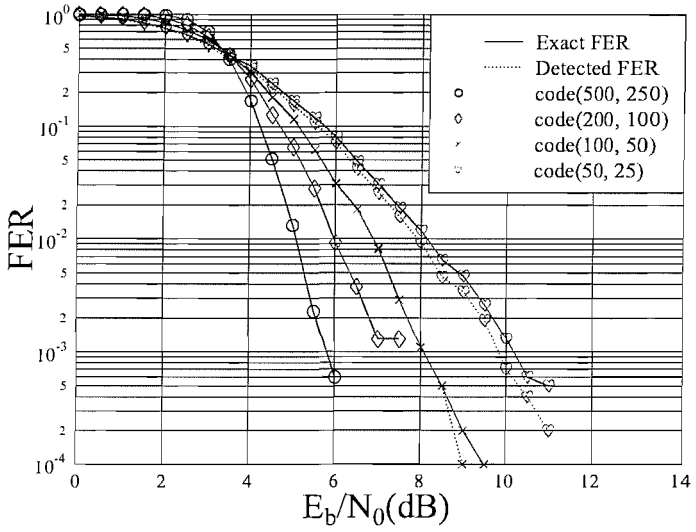


Figure 2.19: FER performance of the four half-rate LDPC codes listed in Table 2.21, when communicating over an uncorrelated Rayleigh fading channel.

FER, as seen in Figure 2.18 and Figure 2.19. The second evaluation method assumes that we have no knowledge of the original transmitted codeword and the only technique available for error detection is that the received codeword is checked by the PCM. If the check result indicates that the decoded codeword is a legitimate one, then this codeword is assumed to be the originally transmitted one. This is denoted as the *Detected FER* in Figure 2.18 and Figure 2.19. It may be observed from Figures 2.18 and 2.19 that the two LDPC codes having a higher block-length, i.e. the codes (500, 250) and (200, 100), have an indistinguishable *Exact* as well as *Detected* FER owing to their comparably longer block-length, i.e. as a result of their higher minimum distance when compared to the remaining two LDPC codes. Again, since no undetected errors are experienced, the two curves indicating the *Exact FER* and the *Detected FER* are merged. By contrast, the LDPC code (100, 50) has a slightly lower minimum distance than the code (200, 100). Hence the FER performances evaluated by the two different methods are different, indicating that some erroneously decoded codewords have been deemed to be error-free by the second evaluation method and thus undetected errors exists for the code (100, 50). However, the two FER curves recorded for the code (100, 50) are nearly identical, which indicates that the likelihood of undetected errors is low. When the block-length is further reduced, the two FER curves of the code (50, 25) generated using the two different evaluation methods differ more significantly. Thus, in this scenario a significant fraction of undetected errors is encountered.

In conclusion, in the context of applications requiring a moderate or long block-length, LDPC codes are unlikely to experience undetected errors owing to the fact that the minimum distance of regular LDPC codes increases linearly with the block-length, when all columns have a weight no less than three. However, in applications requiring a short block-length a higher probability of undetected errors will be experienced.

We note furthermore that, even when long block-length are applied, the minimum distance of the LDPC code will not increase linearly with respect to the block-length, unless the column weight is at least three. Hence undetected errors may occur and the LDPC code may experience a higher error floor. This issue will be further discussed in Chapter 3.

2.9.4 Performance of LDPC codes at various coding rates

In this section, the performance of LDPC codes will be evaluated at different coding rates, benchmarked against the family of turbo convolutional codes having the same code-rate. The simulation parameters of the LDPC codes used are summarised in Table 2.22, while those of the turbo convolutional benchmarker are given in Table 2.23. The coding rate of the LDPC code is varied by changing the size of the parity check matrix. As for the turbo convolutional code, the puncturing pattern listed in Table 2.24 was proposed by Acikel [117], which was applied for arriving at the desired coding rate. The component codes of the turbo code are half-rate Recursive Systematic Convolutional (RSC) codes having a constraint length of four. The forward and feedback generator polynomial is given in octal format as 13 and 15, respectively. In Table 2.24, the upper and lower puncturing pattern is represented by a binary stream of 0s and 1s. The notation 1 and 0 implies that the encoded bit generated by the corresponding component code is retained or punctured, respectively. For example, in Table 2.24 the upper puncturing pattern of the half-rate turbo convolutional code is 10, while the lower puncturing pattern is 01, since at each time instant there will be two parity bits generated by the two component encoders, but they are punctured alternatively. More explicitly, when each of the two convolutional encoder outputs a parity bit, the puncturer will retain the parity from the upper convolutional encoder, while that emerging from the lower encoder is discarded. At the next time instant, the parity accruing from the lower encoder is retained and the upper parity bit is punctured. Thus having two information bits and two parity bits, the encoder constitutes a half-rate encoding scheme.

Channel	AWGN Uncorrelated Rayleigh fading channel
Modulation mode	BPSK
Decoder	Probabilistic decoder
Maximum number of iterations	25
Coded block-length	3000 bits
Coding rate	1/3, 1/2, 2/3, 3/4, 5/6

Table 2.22: Simulation parameters of the LDPC codes having various coding rates.

Figures 2.20 and 2.21 characterise the achievable BER and FER performance of the LDPC and turbo convolutional codes defined in Table 2.22 and Table 2.23, respectively. As we can see, the turbo convolutional code outperforms the identical-rate and identical-length LDPC codes. The attainable performance advantage gain of the turbo code over the LDPC scheme was found to be the highest at a rate of 1/3, which was gradually reduced as the coding rate was increased. The coding gain versus complexity of the LDPC codes specified in Table 2.22 and the turbo convolutional code defined in Table 2.23 is compared in Figure 2.22. The LDPC curves seen in Figure 2.22 represent various

Channel	AWGN Uncorrelated Rayleigh fading channel
Modulation mode	BPSK
Component code	$n = 2$, k=1, K=4, G_0=13, G_1=15
Decoder	LOG-MAP (Log-MAP) decoder
Number of iterations	8
Coded block-length	3000 bits
Coding rate	1/3, 1/2, 2/3, 3/4, 5/6

Table 2.23: Simulation parameters of the turbo convolutional code having various coding rates.

Coding rate	1/3	1/2	2/3	3/4	5/6
Upper puncturing pattern	1	10	1000	100000	1000000000
Lower puncturing pattern	1	01	0001	000001	0000000001

Table 2.24: Puncturing patterns for the turbo convolutional code [117]

coding rates ranging from 0.33 to 0.83 from right to left. By contrast, the turbo convolutional codes' different coding rate are associated with the vertical solid line seen in Figure 2.22 from top to bottom. In Figure 2.22, the complexity is quantified in terms of the number of multiplications required by the decoding of each information bit, while the number of additions are ignored here owing to their relatively lower computational complexity compared to the multiplication process. The complexity of an $(n_c, 1, K_c)$ turbo convolutional code, where n_c determines the coding rate of the constituent encoder and K_c represents the convolutional code's constraint length, can be formulated as [24]:

$$comp\{TC(n_c, 1, K_c)\} = 40(2^{K_c-1}) + 12n_c - 22, \tag{2.51}$$

where it can be observed that the complexity of the turbo convolutional code is a constant for various coding rates, as seen in Figure 2.22. More explicitly, the complexity of the turbo convolutional code increases linearly with n_c, in other words, a lower coding rate and hence more powerful LDPC code results in a higher decoding complexity. The decoding complexity of the turbo convolutional code increases exponentially with the constraint length. By contrast, as seen in Figure 2.22, the LDPC codes achieve a higher coding gain, when the complexity is increased, i.e. when the corresponding coding rate is reduced. As seen in Figure 2.22, when the coding rate is higher than half, LDPC codes exhibit a significantly reduced complexity requirement compared to turbo convolutional codes at the cost of a slightly degraded performance.

2.9.5 Performance of LDPC codes at various coded blocklengths

Since an LDPC code is uniquely defined by its sparse parity check matrix, the performance of LDPC codes is dependent on the size of the PCM. The larger, the better. In this subsection we would like to discuss how the block-length should be chosen for the sake of achieving a satisfactory performance. The LDPC codes characterised in Table 2.25 were simulated and compared to the turbo convolutional code having the same block-length and coding rate. The puncturing pattern seen in Table 2.24 was

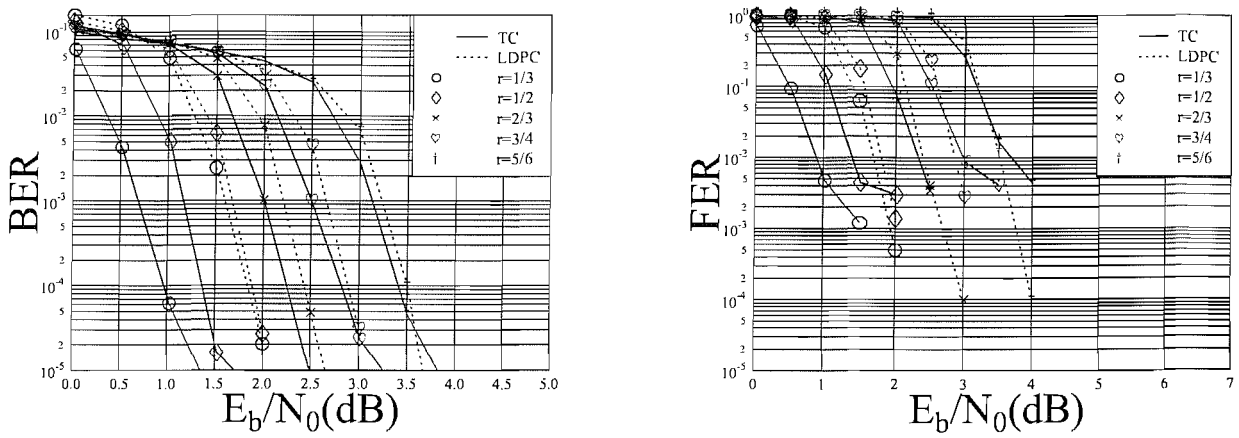


Figure 2.20: BER performance of the LDPC codes specified in Table 2.22 and turbo convolutional codes listed in Table 2.23 using the puncturing pattern seen in Table 2.24, when communicating over an AWGN channel. The associated effective throughputs for all code rates from r=1/3 to r=5/6 are 1/3, 1/2, 2/3, 3/4, 5/6 bps, respectively. The achievable coding gain of the various schemes at a BER of 10^{-4} will be summarised in Figure 2.22 and Table 2.28.

Code rate	1/3, 1/2, 2/3
Coded block-length(bits)	90, 300, 1200, 3000, 6000
Maximum number of iterations	25
Channel	AWGN or Uncorrelated Rayleigh fading
Modulation mode	BPSK

Table 2.25: Simulation parameters for LDPC codes having different blocklengths

applied to the turbo convolutional code for the sake of achieving the corresponding coding rate.

It can be observed in Figures 2.23 to 2.28 that for each individual channel, the turbo convolutional code exhibits a slightly better performance than the LDPC codes. However, the performance difference between the two codes is reduced, when the coding rate is increased, especially, when longer blocklengths are utilised. Furthermore, only a modest additional performance gain may be attained, when the block-length of the LDPC code is extended from 3000 to 6000 bits. Thus we conclude that for both AWGN and uncorrelated Rayleigh fading channels employing a block-length of 3000 coded bits constitutes an appropriate compromise between the achievable performance and the associated coding delay.

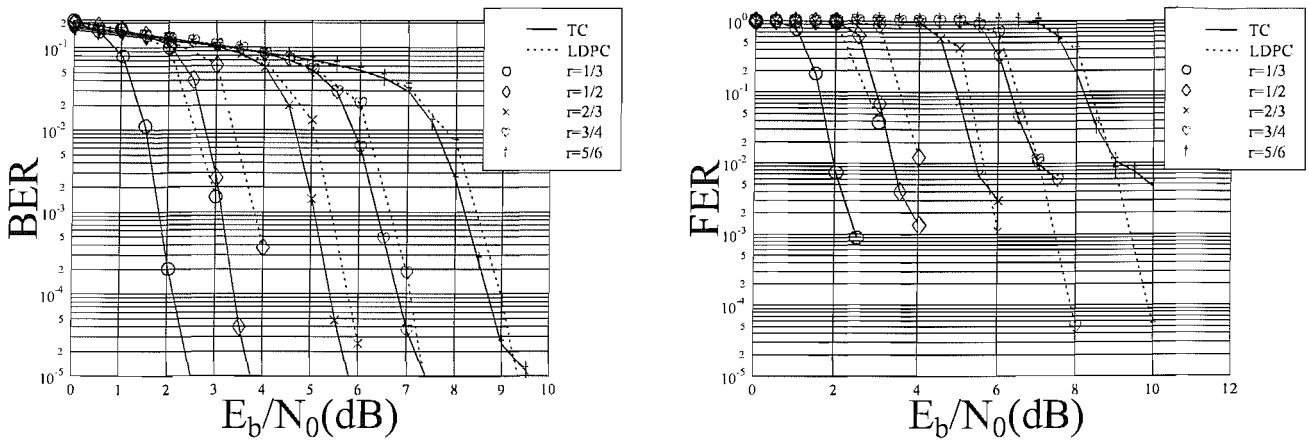


Figure 2.21: BER performance of the LDPC codes specified in Table 2.22 and turbo convolutional codes listed in Table 2.23 using the puncturing pattern seen in Table 2.24, when communicating over an uncorrelated Rayleigh fading channel. The associated effective throughputs for all code rates from r=1/3 to r=5/6 are 1/3, 1/2, 2/3, 3/4, 5/6 bps, respectively. The achievable coding gain of the various schemes at a BER of 10^{-4} will be summarised in Figure 2.22 and Table 2.28.

2.9.6 Performance of LDPC-aided coded modulation over Rayleigh fading channels

When a channel code's coding rate is less than unity, the system's effective throughput is reduced. However, by increasing the number of bits per symbol transmitted, i.e. increasing the size of the modulated signal constellation, the effective throughput of the uncoded system can be maintained. This scheme is referred to as a *coded modulation* arrangement. Historically, the first coded modulation scheme was invented by Ungerböck, which was termed as Trellis Coded Modulation (TCM) [118]. Inspired by the concept of turbo coding, Robertson and Wörtz proposed a more sophisticated scheme referred to as Turbo Trellis Coded Modulation (TTCM) [119].

Since LDPC codes have a less than unity rate, by combining LDPC codes with higher-order modulation schemes, an LDPC Block Coded Modulation (LDPC-BCM) scheme may be created, when absorbing the parity bits in the extended modem constellation without any bandwidth expansion. When using for example a half-rate LDPC codec and employing BPSK modulation, the system's effective throughput has been reduced from 1 bit/symbol of the uncoded system to 0.5 bit/symbol. However, if a QPSK modulation scheme is employed in conjunction with a half-rate LDPC code, the system's effective throughput can be maintained at 1 bit/symbol.

Previously, the performance of LDPC codes using BPSK modulation has been evaluated, when communicating over both AWGN and uncorrelated Rayleigh fading channels. For the sake of increasing the effective throughput of the system, typically a higher number of modulation levels is used. Hereby we would like to investigate the performance of LDPC-BCM, when communicating over both uncorrelated and correlated Rayleigh fading channels. The TTCM scheme will be used as our benchmarker. The simulation parameters are tabulated in Table 2.26. The maximum number

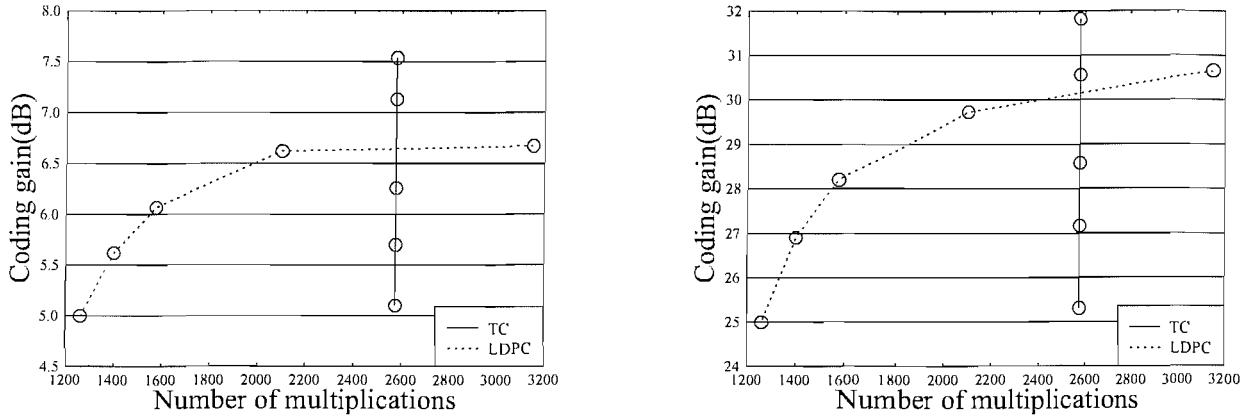


Figure 2.22: Coding gain versus complexity of the LDPC codes specified in Table 2.22, where the decoding complexity was varied by varying the code rate, but using always 25 iterations. The points in the graph correspond to r =1/3, 1/2, 2/3, 3/4 and 5/6. The turbo convolutional codes were listed in Table 2.23 using the puncturing patterns seen in Table 2.24 resulting in exactly the same code rate as in the LDPC codes. Transmissions were carried out over an AWGN(left) and uncorrelated Rayleigh fading channel(right).

of LDPC-BCM iterations was adjusted for the different modulation modes for the sake of ensuring that the maximum decoding complexity of the LDPC-BCM scheme did not exceed that of the TTCM benchmarker. A channel interleaver of 3000 coded symbols was utilised, when communicating over the correlated Rayleigh fading channel having a normalised Doppler frequency of 3.25×10^{-5}, in order to separate the symbols suffering from the bursty error effects of deep fades. The effect of removing the channel interleaver from the system, when communicating over correlated Rayleigh fading channels was also evaluated.

Figure 2.29, Figure 2.30 and Figure 2.31 demonstrate that in the context of uncorrelated Rayleigh channels LDPC-BCM outperforms the TTCM benchmarker scheme by almost 3dB, 1.5dB and 3dB, when using QPSK, 8PSK and 16QAM modulation, respectively at the BER of 10^{-5}. It is shown in Figures 2.29, 2.30 and 2.31 that increasing the number of iterations to 50 results in an approximately 0.5dB further coding gain for the LDPC-BCM scheme.

When communicating over a correlated Rayleigh fading channel, we observe in Figure 2.32, Figure 2.33 and Figure 2.34 that LDPC-BCM and TTCM achieve a similar performance, provided that a channel interleaver is incorporated into the system. However, the LDPC-BCM operates with no performance degradation, when the channel interleaver is removed from the system. By contrast, the TTCM scheme suffers from an approximately 5 to 7 dB E_b/N_0 degradation at the BER of 10^{-5}. This is a benefit of the randomly constructed parity check matrix and high block-length of LDPC-BCM, since each of the parity check equations is checking several random bit positions in a codeword, which has a similar effect to that of the channel interleaver. By contrast, the TTCM scheme is decoded using the trellis-based symbol-by-symbol *Maximum A-Posteriori* (MAP) algorithm [115], where the

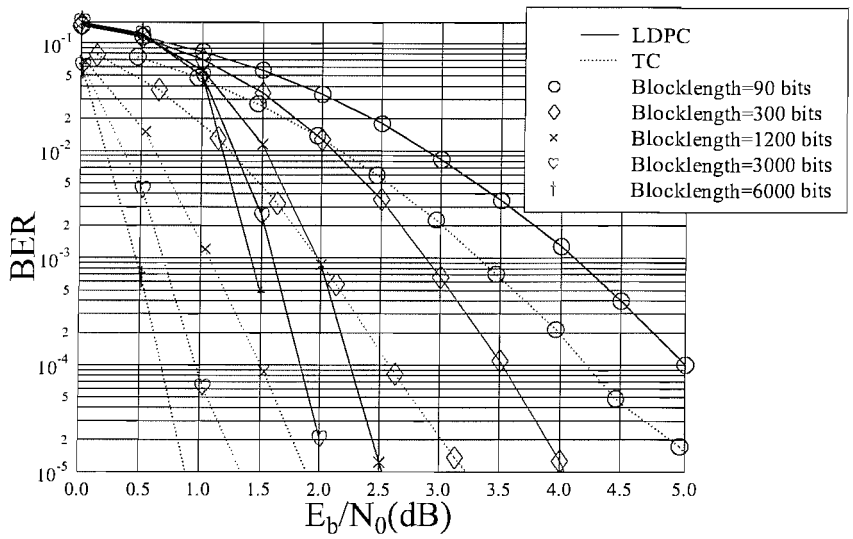


Figure 2.23: BER performance of the third-rate LDPC codes characterised in Table 2.25 and turbo convolutional codes described in Table 2.23 using the puncturing pattern seen in Table 2.24, when communicating over an AWGN channel. The achievable coding gain of the various schemes at a BER of 10^{-4} will be summarised in Table 2.29.

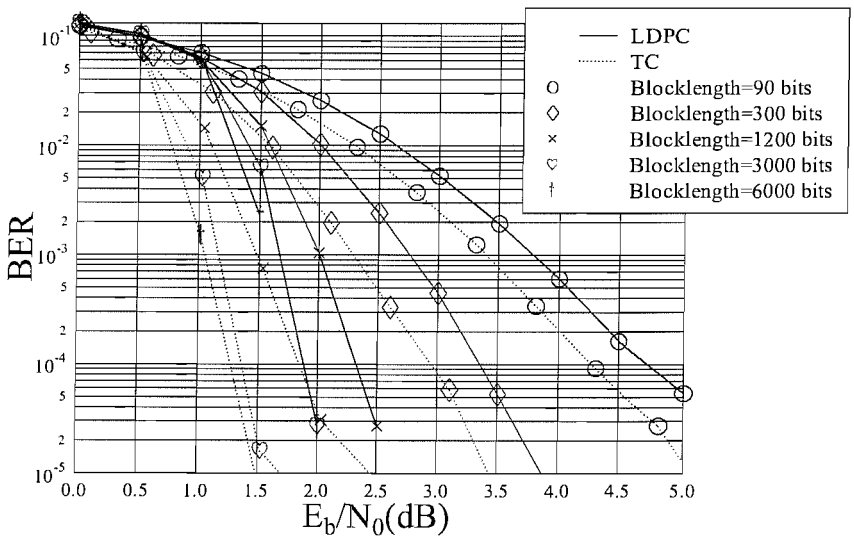


Figure 2.24: BER performance of the half-rate LDPC codes characterised in Table 2.25 and turbo convolutional codes described in Table 2.23 using the puncturing pattern seen in Table 2.24, when communicating over an AWGN channel. The achievable coding gain of the various schemes at a BER of 10^{-4} will be summarised in Table 2.29.

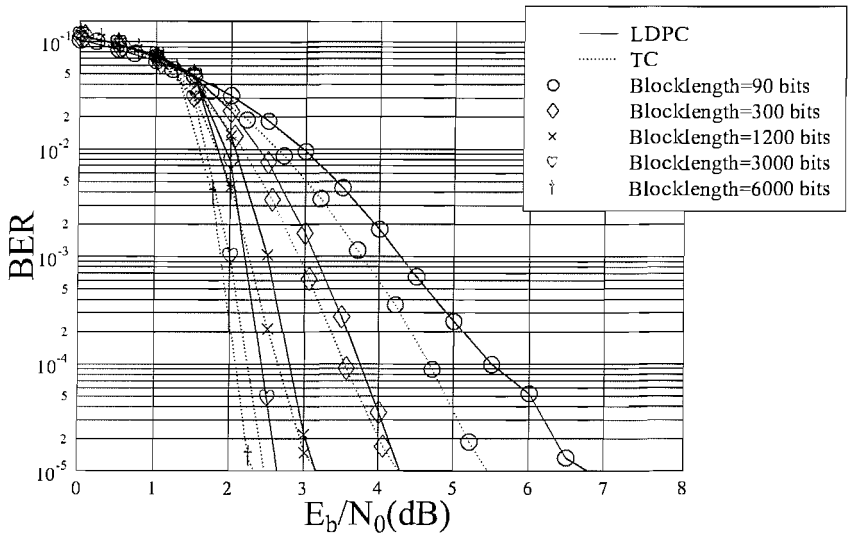


Figure 2.25: BER performance of the two-third-rate LDPC codes characterised in Table 2.25 and turbo convolutional codes described in Table 2.23 using the puncturing pattern seen in Table 2.24, when communicating over an AWGN channel. The achievable coding gain of the various schemes at a BER of 10^{-4} will be summarised in Table 2.29.

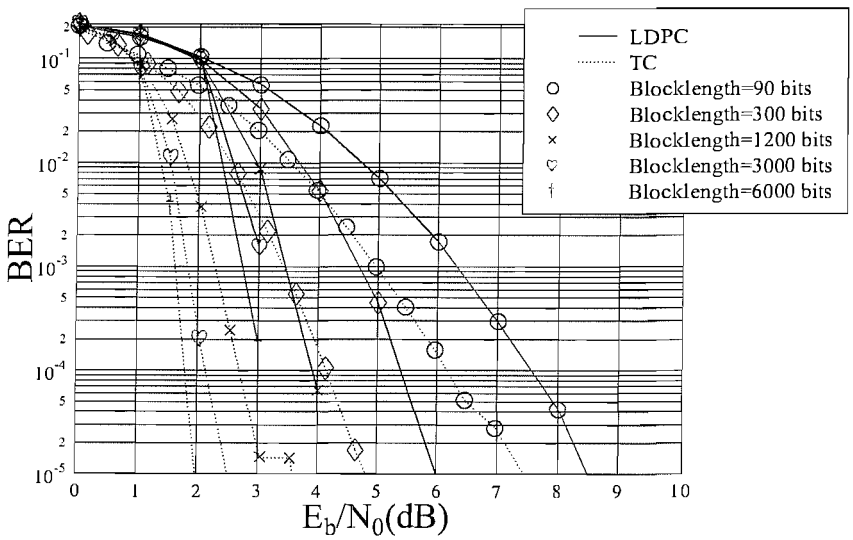


Figure 2.26: BER performance of the third-rate LDPC codes characterised in Table 2.25 and turbo convolutional codes described in Table 2.23 using the puncturing pattern seen in Table 2.24, when communicating over an uncorrelated Rayleigh fading channel. The achievable coding gain of the various schemes at a BER of 10^{-4} will be summarised in Table 2.29.

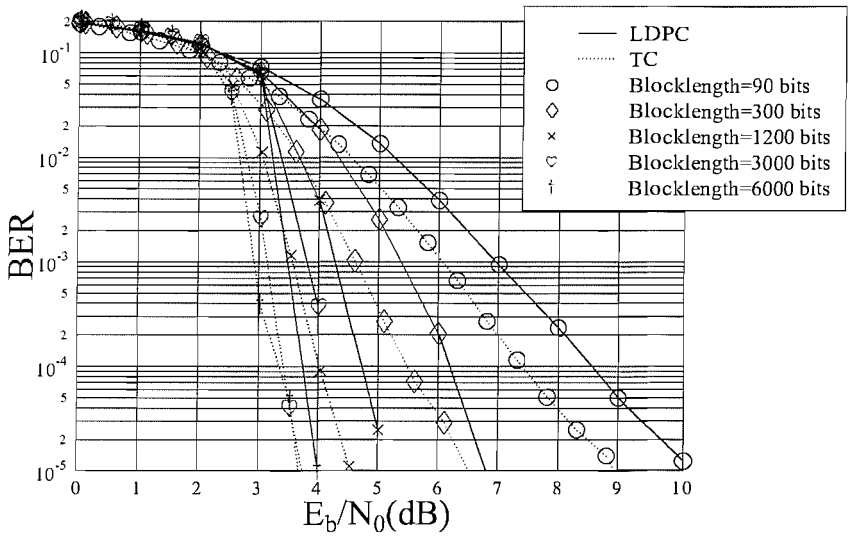


Figure 2.27: BER performance of the half-rate LDPC codes characterised in Table 2.25 and turbo convolutional codes described in Table 2.23 using the puncturing pattern seen in Table 2.24, when communicating over an uncorrelated Rayleigh fading channel. The achievable coding gain of the various schemes at a BER of 10^{-4} will be summarised in Table 2.29.

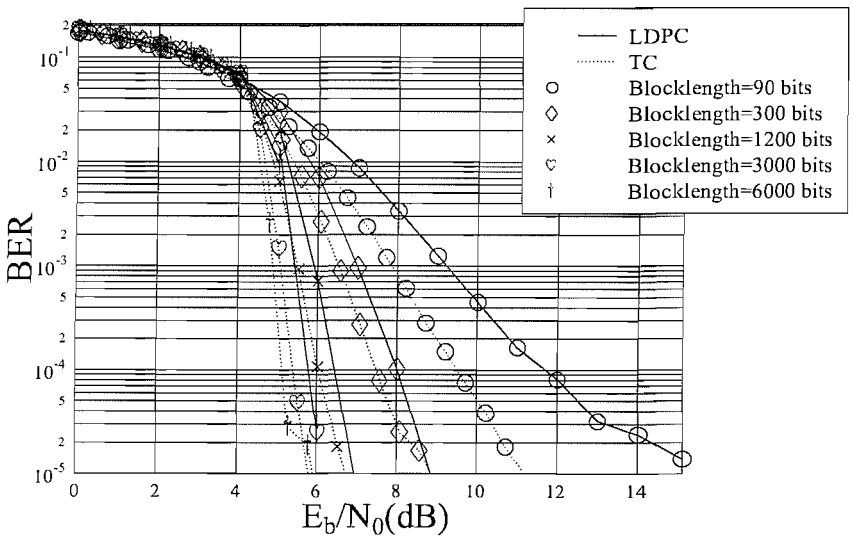


Figure 2.28: BER performance of the two-third-rate LDPC codes characterised in Table 2.25 and turbo convolutional codes described in Table 2.23 using the puncturing pattern seen in Table 2.24, when communicating over an uncorrelated Rayleigh fading channel. The achievable coding gain of the various schemes at a BER of 10^{-4} will be summarised in Table 2.29.

Tx. Burst Length, L	1000 Symbols
Coding Rate, R	1/2, 2/3, 3/4
Modulation Mode	QPSK(QPSK), 8PSK(8PSK), 16QAM(16QAM)
Channel	Uncorrelated Rayleigh Fading Correlated Rayleigh Fading
Normalised Doppler Frequency	3.25×10^{-5}
Channel Interleaver Length	3000 Symbols
LDPC-BCM Column Weight	3
Maximum No. of Iterations for LDPC-BCM	50
TTCM Iteration	4
No. of LDPC-BCM Iterations	QPSK:15, 8PSK:10, 16QAM:10

Table 2.26: LDPC-BCM System Parameters

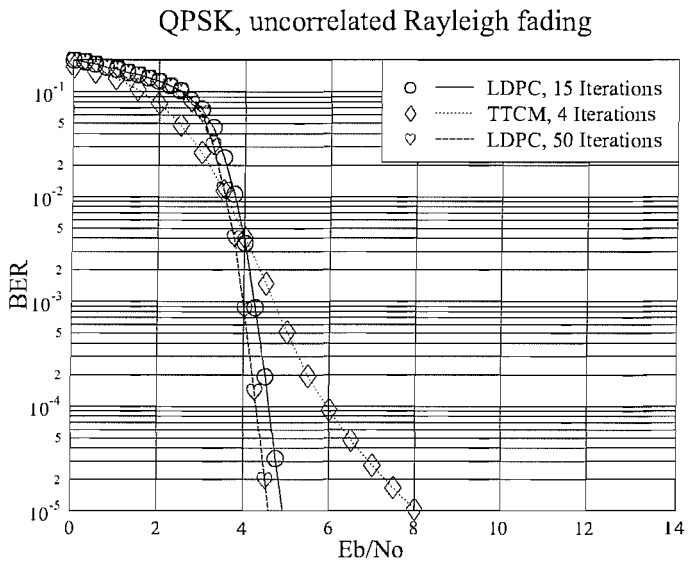


Figure 2.29: BER performance of LDPC and TTCM parameterised in Table 2.26, utilising QPSK when communicating over uncorrelated Rayleigh fading channel.

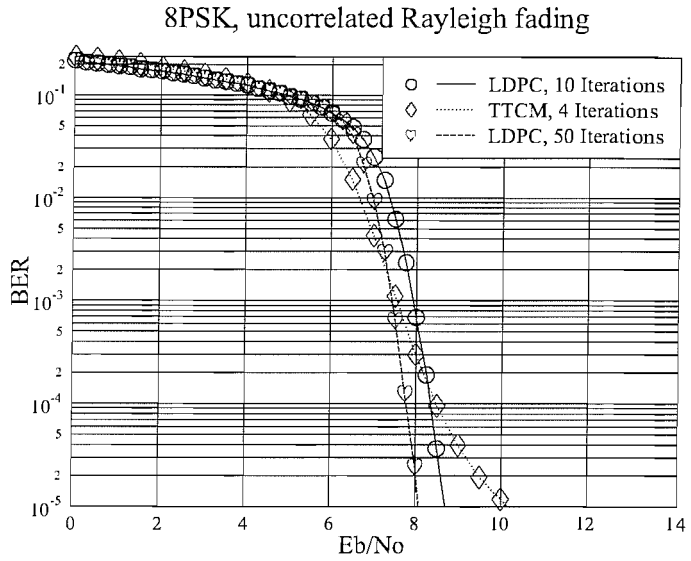


Figure 2.30: BER performance of LDPC and TTCM parameterised in Table 2.26, utilising 8PSK when communicating over uncorrelated Rayleigh fading channel.

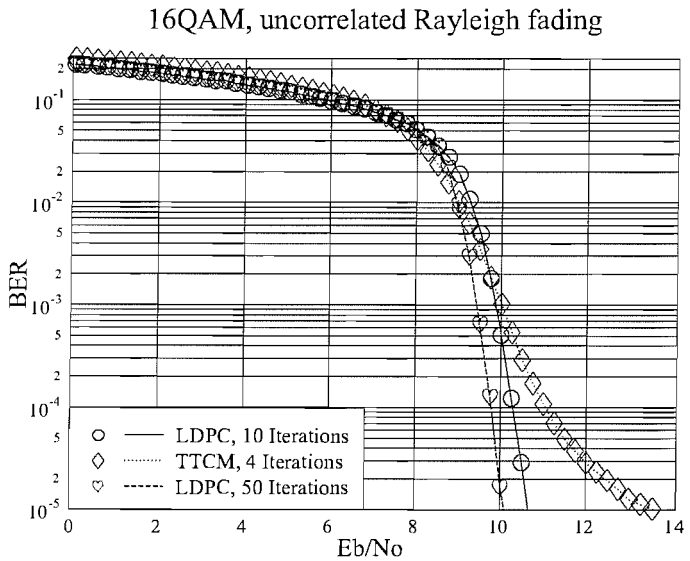


Figure 2.31: BER performance of LDPC and TTCM parameterised in Table 2.26, utilising 16QAM when communicating over uncorrelated Rayleigh fading channel.

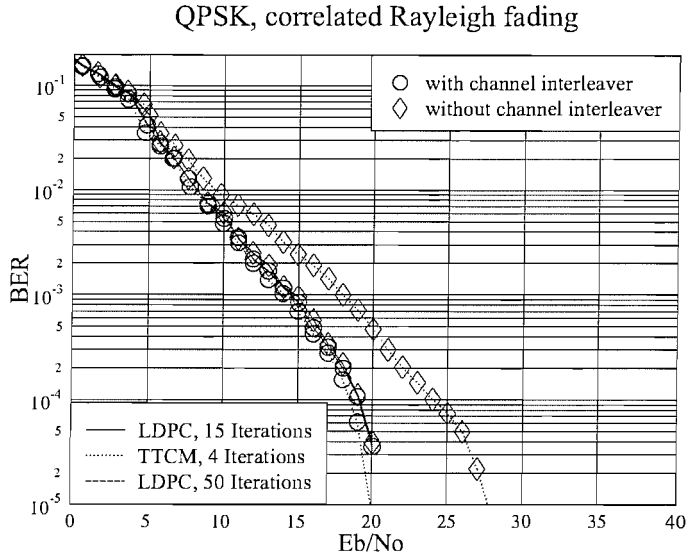


Figure 2.32: BER performance of LDPC and TTCM parameterised in Table 2.26, utilising QPSK when communicating over correlated Rayleigh fading channel.

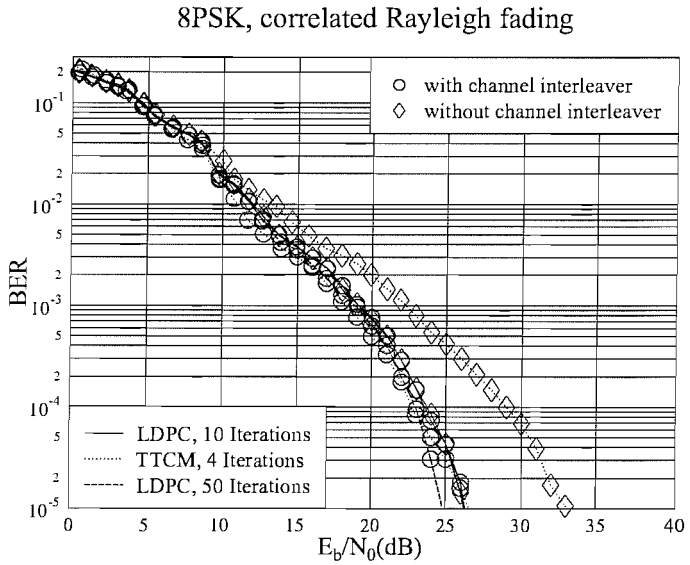


Figure 2.33: BER performance of LDPC and TTCM utilising 8PSK parameterised in Table 2.26, when communicating over correlated Rayleigh fading channel.

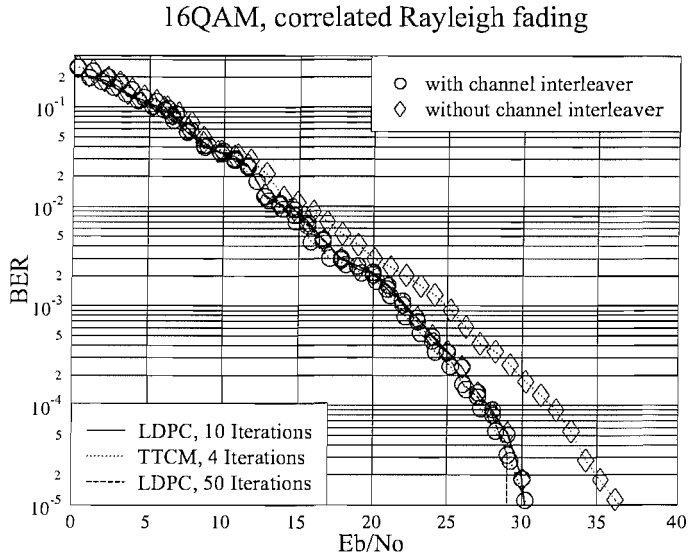


Figure 2.34: BER performance of LDPC and TTCM utilising 16QAM parameterised in Table 2.26, when communicating over correlated Rayleigh fading channel.

neighbouring symbols are exposed to correlated channel fading effects and hence they are less capable of coping with bursty channel errors. Thus in the context of TTCM, the channel interleaver plays a crucial role in dispersing the bursty channel errors.

2.10 Summary and conclusion

In this chapter, the regular construction LDPC codes were introduced. Commencing in Section 2.3 from Gallager's original proposed PCM , the characteristics of LDPCCs such as the column weight, the nature of cycles and the bipartite graph representation of the PCM were described. Section 2.4 provided a detailed description of the LDPC encoding process, i.e. how the generator matrix $\mathbf{G}$ is obtained from $\mathbf{H}_r$, which is a column permutated PCM generated from the original PCM $\mathbf{H}$. It was also outlined how the parity check bits are generated from $\mathbf{G}$ and appended to the original information bits. Section 2.5 gave an introduction to two LDPC decoding algorithms, namely the optimum *exhaustive enumeration based decoding* and the sub-optimum *probabilistic decoding algorithm*. *Probabilistic decoding* was described using Gallager's original approach. Furthermore, a detailed decoding example using *probabilistic decoding* was given in Section 2.6. In order to generalise Gallager's LDPC model, which only considered binary transmission, a more general LDPC decoding procedure was provided and Richardson's [9] reduced complexity calculation of the message R was presented which involved the FFT. Furthermore, the arithmetic complexity of LDPC decoding using Richardson's FFT approach was quantified using the number of additions and multiplications in Section 2.7.3. Even though a mathematical analysis of the performance of probabilistic decoding is challenging, a weak bound was briefly introduced in Section 2.8 for LDPC coded systems communicating over the BSC channel .

In Section 2.9, the performance of LDPCCs was characterised in various scenarios. The effect of increasing the LDPC decoder's complexity, i.e. the number of iterations was studied in Section 2.9.1, and as seen in Figure 2.13, Figure 2.14 and Figure 2.15, using eight to 20 iterations achieves a good compromise between the attainable coding gain and the associated decoding complexity. The BER of each constituent bit of the LDPC codeword was observed and found to be equally protected in Section 2.9.2. In Section 2.9.3, four half-rate LDPC codes having different block-lengths were characterised using two different FER measures. It was observed that LDPC codes have attractive minimum distance properties, provided that the minimum column weight is no less than three. LDPC codes having different coding-rates were compared to the turbo convolutional codes having identical rates in Section 2.9.4. More explicitly, the LDPC codes were found to have a similar BER performance to the turbo convolutional codes, while potentially imposing a lower complexity, when the coding-rate is higher than half. Furthermore, the effects of increasing an LDPCC's coded block-length was demonstrated in Section 2.9.5, and a block-length of 3000 bits was found to strike an appropriate compromise between the achievable performance and the associated coding delay.

In Section 2.9.6 a novel coded modulation scheme, namely the LDPC-BCM arrangement was proposed, invoking high-order modulation schemes. In the classic turbo trellis coded modulation scheme there is an un-protected bit, which may limit the achievable error correction performance in certain propagation scenarios. Hence, as a design alternative, we contrived an LDPC-based BCM scheme based on the design philosophy of protecting all the bits and absorbing the increased number of channel-coded bits by appropriately extending the modulated signal constellation. It was observed that LDPC-BCM constituted a more attractive scheme in comparison to the TTCM benchmarker scheme in terms of the attainable BER performance when communicating over uncorrelated Rayleigh fading channels. It was also noted that no channel interleaver was necessary for the LDPCC, when communicating over correlated Rayleigh fading channels, provided the block-length of LDPCC bridges over several channel fades.

A coding gain table is provided here for the sake of summarising the LDPCC's BER performance, when the maximum number of iterations, the blocklength and the code rate are varied.

LDPC code	Maximum no. of iterations	Coding gain (**G**)(dB)	Coding gain (**UR**)(dB)
(200,100)	2	2.705	22.941
	4	3.911	25.647
	8	4.411	26.706
	20	4.676	26.941
	50	4.794	27.255
	100	4.882	27.411
(500,250)	2	2.823	23.176
	4	4.352	26.353
	8	5.205	27.941
	20	5.470	28.353
	50	5.646	28.647
	100	5.735	28.706
(1000,500)	2	2.882	23.294
	4	4.529	26.588
	8	5.588	28.471
	20	6.000	29.059
	50	6.146	29.236
	100	6.176	29.294

Table 2.27: Coding gain achieved by the three half-rate LDPC codes parameterised in Table 2.17 at a BER of 10^{-4}, when communicating over both AWGN (AWGN) (**G**) and uncorrelated Rayleigh fading (**UR**) channels.

Code rate	Coding gain (**G**)(dB)	Coding gain (**UR**)(dB)
1/3	6.66	30.58
1/2	6.58	29.66
2/3	6.08	28.18
3/4	5.58	26.83
5/6	5.1	25

Table 2.28: Coding gain achieved by the LDPC code parameterised in Table 2.22 at a BER of 10^{-4}, when communicating over both AWGN (**G**) and uncorrelated Rayleigh fading (**UR**) channels.

Code rate	Coded blocklength(bits)	Coding gain (**G**)(dB)	Coding gain (**UR**)(dB)
1/3	90	3.47	26.43
	300	4.97	28.57
	1200	6.22	30.14
	3000	6.613	30.43
	6000	6.76	30.86
1/2	90	3.76	25.43
	300	5.113	27.86
	1200	6.15	29.29
	3000	6.613	29.72
	6000	6.72	30.29
2/3	90	3.03	22.32
	300	4.81	26
	1200	5.7	27.58
	3000	6.03	28.32
	6000	6.14	28.5

Table 2.29: Coding gain achieved by the LDPC code parameterised in Table 2.25 at a BER of 10^{-4}, when communicating over both AWGN (**G**) and uncorrelated Rayleigh fading (**UR**) channels.

Chapter 3

Irregular LDPC codes

3.1 Introduction

In the previous chapter, the family of regular-construction LDPCCs was introduced and their decoding procedure was detailed with the aid of a worked example. The code was referred to as "regular", since the Hamming weights of all the columns and rows were constant. However, Luby *et al.* [60] have shown that by imposing a carefully chosen non-uniform distribution of the column and row weights during the generation of the parity check matrix, the code may become capable of outperforming its regular-construction LDPCCs counterpart, as will be shown in Section 3.8 of this chapter. Richardson proposed the employment of the *Density Evolution* (DE) technique for predicting the performance of LDPCCs having an infinite length at a given column and row density distribution profile [9], and showed how to use DE for determining the optimum density distribution profile in [61]. Since DE is a complex operation, Chung simplified Richardson's high complexity *Density Evolution* algorithm using *Gaussian Approximation* (GA) [11]. The intuition of replacing regular LDPCC by their irregular counterparts may be explained by using the so-called Tanner graph [5]. Figure 3.1 portrays a fraction of the parity check matrix seen in Table 2.4.

As introduced in Chapter 2, an LDPCC's PCM can be represented by a Tanner graph. The Tanner graph is constituted by the so-called message nodes seen at the left of Figure 3.1, which are associated with each individual column of the PCM. The check nodes are at the right of the Tanner graph of Figure 3.1, which represent each individual row of the PCM. By choosing the 12^{th}, 13^{th}, 14^{th} and 15^{th} column randomly in Table 2.4, they are represented by the message nodes seen at the left of Figure 3.1, and the randomly selected 1^{st}, 2^{nd}, 9^{th} and 10^{th} rows of Table 2.4 are shown as the check nodes at the right of Table 2.4. The line connecting the j^{th} left node and the i^{th} right node represents the non-zero entry at the intersection of the i^{th} row and the j^{th} column in Table 2.4. For example, the non-zero entry found in the first row and the second column of the parity check matrix of Table 2.4 is represented by the first horizontal line at the top of Figure 3.1. The dotted lines indicate encountering a short cycle of length 4.

According to the decoding process of the LDPCC described in Section 2.5.2, the message nodes and the check nodes exchange their information iteratively. Recall from Chapter 2 that a regular-construction LDPC parity check matrix has an uniform column weight and a constant or near constant row weight. However, for the message nodes of an LDPC code, it is better to have a high column weight,

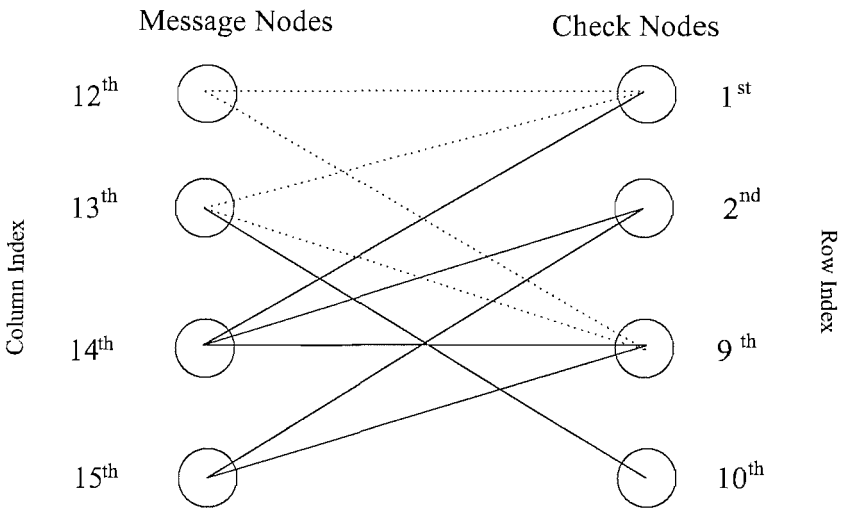


Figure 3.1: Tanner node representation of part of the parity check matrix of Table 2.4. The dotted lines represent a *length-4 cycle*.

since the increased number of check nodes become capable of providing more reliable information for the message nodes. By contrast, it is desirable for the check nodes to have a low number of entries in a row, because then they have to check the parity of a reduced number of columns, which results in less ambiguity concerning the index of the unreliable entries and hence more useful parity information may be exchanged with the connected message nodes. Based on this intuition, in 1998, Luby [60] and his colleagues proposed the introduction of irregular-constructed parity check matrices. More explicitly, instead of maintaining a constant column weight and row weight across the entire parity check matrix, as proposed by Gallager, they varied the column density and row density, i.e. use a pre-determined *density-profile*, which resulted in an improved performance over that of the regular-construction parity check matrix when opting for a long block-length, provided that the density-profile used was carefully chosen. The rationale of this design philosophy was that a message node associated with an increased column weight received more information from the check nodes, and hence it was capable of more reliably converging to its correct value, consequently providing more valuable information for the check nodes, which in turn assisted the message nodes having a lower weight in converging to their correct values more reliably.

3.2 Definition of the row and column density distribution

There have been loose definitions which specify the weight of a specific column and the profiles which specify the density of weight-i nodes across the whole PCM. In some literature, for example in [61] [60], the terminology *degree i* was used to specify a message node having weight i. In [120], the term *density distribution* is used to specify the density distribution of the PCM, while in [121], the term *weight distribution* is used. For the sake of maintaining consistency within this thesis, the term *weight i node* will be used to specify a certain node or nodes having i edges connecting with the neighbouring nodes as in Figure 3.1. For example, the top left message node in Figure 3.1 is a weight two message node because there are two edges connecting this node with the 1^{st} and the 9^{th} check node; and the top

right node is a weight three check node, since the 2^{nd}, 3^{rd} and 4^{th} message nodes are connected to it.

In our forthcoming discourse the term *density* is used, which is associated with the percentage of *weight i nodes* across the PCM. We will use the same definition as used by Urbanke *et al* in [11]. A density profile can be defined by

$$\lambda_e(x) = \sum_{i}^{w_{c(max)}} \lambda_i x^{i-1} \tag{3.1}$$

and

$$\rho_e(x) = \sum_{i}^{w_{r(max)}} \rho_i x^{i-1}, \tag{3.2}$$

which correspond to the column density profile and row density profile, respectively [122]. The notations $w_{c(max)}$ and $w_{r(max)}$ are used for representing the maximum column weight and the maximum row weight of the density profile, while λ_i and ρ_i quantify the fraction of edges belonging to degree-i variable and check node, respectively.

To explain the terminology *density* more explicitly, let's use an example as:

$$\lambda_e(x) = 0.4x^1 + 0.6x^2, \tag{3.3}$$

Equation 3.3 gives an example of a column density profile for the irregular LDPC code represented using Tanner graph in Figure 3.1. This profile shows the information that 40% of the edges are connected to weight-two message nodes, and the remaining 60% connects to the weight-three message nodes. To elaborate a little further, in the literature, two different ways of representing the density profile are used. The first one is referred to as the *edge perspective* representation [9] [11]. This representation specifies the proportion of edges in the Tanner graph which are connected to the corresponding nodes associated with a specific term in Equation 3.3 for example, having a certain polynomial degree. The notations of $\lambda(x)$ and $\rho(x)$ are used for representing the column's and the row's density distribution, respectively.

There are a total of 10 edges in Figure 3.1, and two edges are connected to both the 12^{th} message node as well as to the 15^{th}. Because the 12^{th} and the 15^{th} message nodes are weight-two message nodes, 40% of the 10 edges are connected to weight-two message nodes, and the remaining 60% are connected to the weight-three message nodes, thus yielding $\lambda_e(x) = 0.4x^1 + 0.6x^2$ as in Equation 3.3. For the corresponding expression of $\rho(x)$, the four check nodes seen in Figure 3.1 have a weight and corresponding polynomial degree varying from 1 to 4. Explicitly, 10% of the edges are connected to the weight-1 check node, 20% to the weight-two, and 30% as well as 40% to the weight-three and weight-four check nodes respectively. Hence we arrive at the expression of $\rho_e(x) = 0.1x^0 + 0.2x^1 + 0.3x^2 + 0.4x^3$. The other way of defining the density profile using $\lambda_n(x)$ and $\rho_n(x)$ is termed as *node perspective*, rather than *edge perspective* representation. In this representation, the polynomials $\lambda_n(x)$ and $\rho_n(x)$ specify the proportion of the nodes having a particular weight of i. As seen in Figure 3.1, since there are two message nodes of weight 2 and two of weight 3, i.e. 50% of the message nodes have a weight of two and another 50% have a weight of three, the *node perspective* representation is $\lambda_n(x) = 0.5x^1 + 0.5x^2$. As for the check nodes, the weights of the four check nodes from the top downwards are 3, 2, 4, 1, respectively. Thus $\rho_n(x)$ may be written as $\rho(x) = 0.25x^0 + 0.25x^1 + 0.25x^2 + 0.25x^3$. In order to distinguish these two representations, we will use the notation of $\lambda_n(x)$ and $\rho_n(x)$ for the *node perspective* representation, while $\lambda_e(x)$ and $\rho_e(x)$ will refer to the *edge perspective* representation. The

constraints of $\lambda(1) = 1$ and $\rho(1) = 1$ have to be satisfied, since the probabilities of all of the different weights should sum to unity in both representations.

3.3 Performance of irregular LDPC codes

The beneficial BER effects of using irregular LDPCCs are illustrated in Figures 3.2, 3.3 and 3.4, where an AWGN channel having $E_b/N_0 = 3.2dB$ was used. The associated density profiles of the codes characterised in these three figures are as follows:

$$\begin{aligned} \lambda_n(x) &= 0.37787x^1 + 0.34903x^2 + 0.09643x^5 + 0.06730x^6 + 0.10937x^{19} \\ \rho_n(x) &= x^{27}; \end{aligned} \tag{3.4}$$

$$\begin{aligned} \lambda_n(x) &= 0.41564x^1 + 0.34806x^2 + 0.03181x^3 + 0.2045x^9 \\ \rho_n(x) &= 0.80952x^{15} + 0.19048x^{16}; \end{aligned} \tag{3.5}$$

$$\begin{aligned} \lambda_n(x) &= 0.00024x^0 + 0.17516x^1 + 0.6796x^2 + 0.0905x^6 + 0.0545x^7 \\ \rho_n(x) &= 0.31665x^{18} + 0.68335x^{19}, \end{aligned} \tag{3.6}$$

where we used the notation of $\lambda_n(x) = \sum \lambda_i x^{i-1}$ and $\rho_n(x) = \sum \rho_i x^{i-1}$ for representing the column density distribution and row density distribution, respectively. The exponent i of x in the summation represents the density concerned, while the coefficient λ_i or ρ_i gives the corresponding relative frequency of a term having a given weight. As we can see in Equation 3.6, only a small fraction of the columns has a weight of 1. This is because the column density has to be consistent with the row density profile. More explicitly, the number of non-zero entries counted column-by-column has to be the same as that counted on a row-by-row basis. However, since the weight-1 columns constitute only a small fraction of the PCM, their impact on the decoding performance is minor.

Focusing our attention on Figures 3.2, 3.3 and 3.4, it can be observed that the message nodes associated with a higher exponent, i.e. a higher weight, converge to higher LLRs within a lower number of iterations. However, the LLR curve associated with the weight-20 nodes in Figure 3.2 converges initially fast, especially during the first few iterations, but the achievable convergence rate reduces after a few iterations and becomes more slowly than that of the nodes associated with a weight of six and seven. This is because the message node associated with weight 20 has more column entries, and hence initially it is benefiting from more parity information than the nodes having a weight of six or seven. However, having more non-zero entries in the columns is expected to introduce cycles associated with comparatively low lengths. Owing to the message passing decoding algorithm of the LDPCC, the *extrinsic* information is passed between the non-zero entries in the parity check matrix, both vertically and horizontally. If a message node is associated with two non-zero entries, which belong to a short cycle of length four to consider the worst-case scenario, then these two non-zero entries are only receiving information from two parity checks and another message node, thus the *extrinsic* information being passed around amongst these nodes will reduce at a faster rate, since only a limited amount of independent *extrinsic* information may be provided by the 4 nodes concerned after a few iterations. By contrast, in the extreme case, when there are no cycles within a parity check matrix, all the message nodes are benefiting from the *extrinsic* information provided by all other message nodes and all the check nodes. In this case the magnitude of the *extrinsic* information will

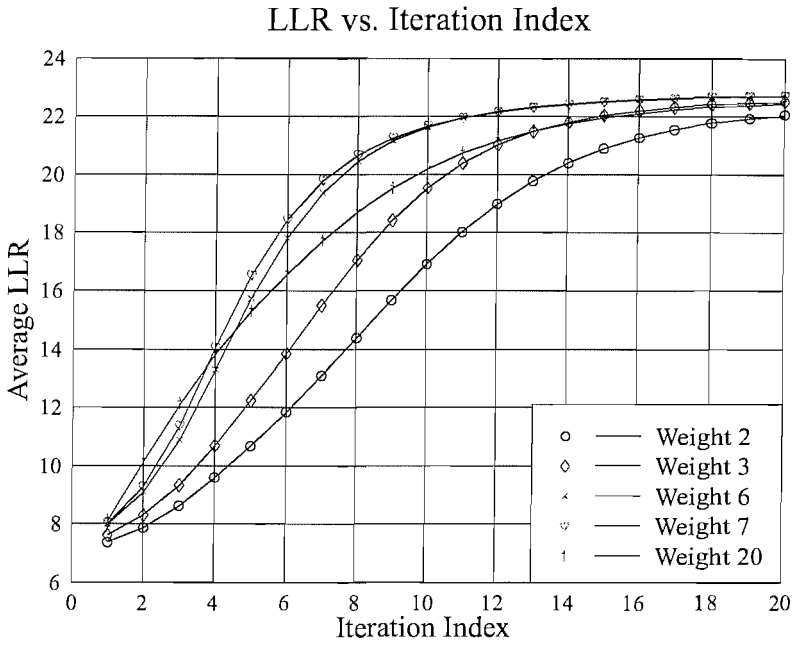


Figure 3.2: Average LLR versus iteration index characteristics of the (4161,3430) irregular LDPC code using the density distribution of Equation 3.4, when communicating over an AWGN channel at $E_b/N_0 = 3.2dB$

decrease at a lower rate. For this reason, the node associated with weight 20 is unable to converge as fast as the nodes associated with weight 6 and 7 after the first few iterations.

3.4 Density evolution

Richardson *et. al* [9] demonstrated that the average asymptotic behaviour of a belief-propagation based LDPC decoder is numerically computable by using the *density evolution* algorithm. The *density evolution*(DE) algorithm allows one to calculate the E_b/N_0 value required for error free communication by an irregular LDPC code having a given density profile, when the block-length of the code tends to infinity. This algorithm determines a quantity termed as the *threshold.* The *threshold* is defined as the maximum value of a specific channel parameter, such as for example the standard deviation of the AWGN of the channel or the crossover probability for the Binary Symmetric Channel (BSC), which allows the decoder to achieve an arbitrary low error probability upon invoking an arbitrarily high number of iterations. The process of density evolution will be described as follows.

As mentioned previously, the LDPC code's decoding algorithm iteratively passes information between the non-zero entries in the parity check matrix, i.e. between the message nodes and check nodes in the Tanner graph used. The philosophy of the density evolution algorithm is based on the *local tree assumption*, which implies that the LDPC code may be represented by a tree structure, as illustrated

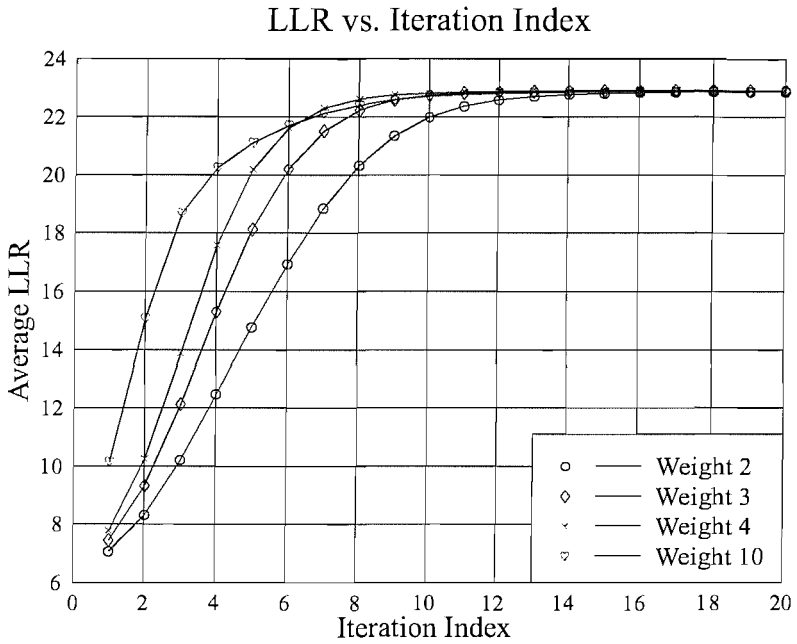


Figure 3.3: Average LLR versus iteration index characteristics of the (4000,3000) irregular LDPC code using density distribution of Equation 3.5, when communicating over an AWGN channel at $E_b/N_0 = 3.2dB$

in Figure 3.5.

To elaborate a little further, in Figure 3.5, the hollow circles and the filled circles represent the check nodes and the message nodes seen in Figure 3.1. The lines horizontally connected to the message nodes represent the *intrinsic* information provided by the channel's output. Let us assume that the filled node at the top of Figure 3.5 represents the k^{th} coded symbol in the LDPC block of N symbols, which is termed as the *root node* [10]. The *root node* receives information from the check nodes it is connected to at the level seen below it in Figure 3.5 and those check nodes also receive information from the message nodes they are connected to at the next level down, etc. The dotted lines in Figure 3.5 indicate that the above process is repeated further by expanding the tree. The number of connections associated with a message node (excluding the line representing the *intrinsic* information) represents the column weight of this particular message node, while the number of connections associated with a check node represents the corresponding row weight. The DE process exploits the fact that information is passed from the message nodes to the check nodes in the upwards direction, and further up again to the message nodes during the completion of one LDPC iteration. This entire DE process mimics the decoding process of LDPC codes communicating over a channel characterised by a channel parameter referred to as the *threshold*. The above-mentioned *local tree assumption* implies that there are no repeated nodes in Figure 3.5 within a certain number of consecutive tree levels in Figure 3.5. More explicitly, since each message node and check node is related to a non-zero entry in the PCM, the *local tree assumption* implies that there will be no more than one message node or check node representing

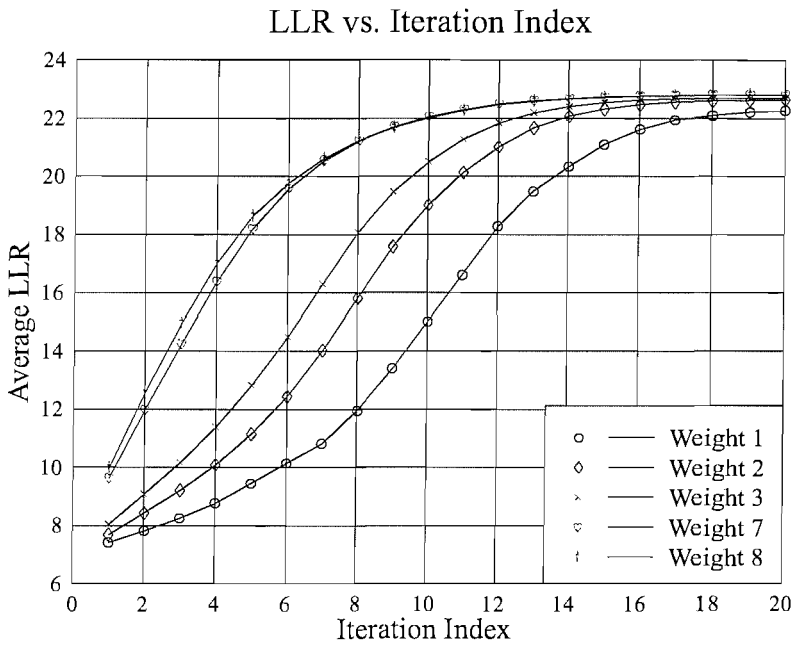


Figure 3.4: Average LLR versus iteration index characteristics of the (4161,3430) irregular LDPC code using density distribution of Equation 3.6, when communicating over an AWGN channel at $E_b/N_0 = 3.2dB$

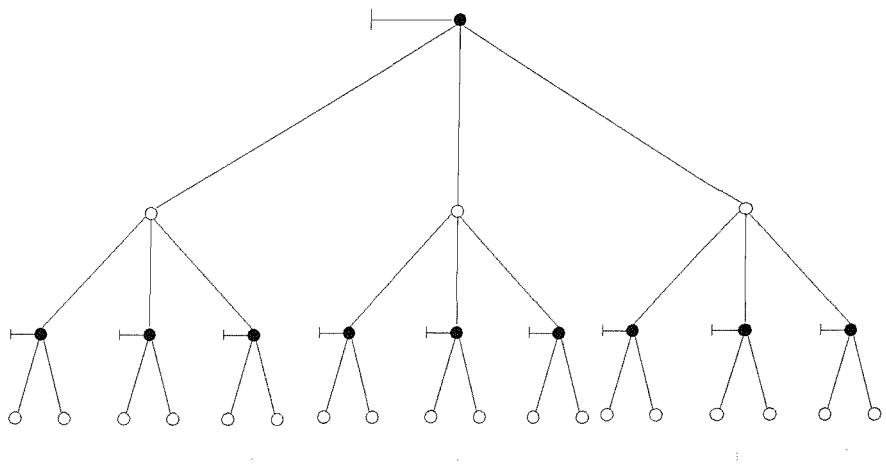


Figure 3.5: A tree representation of LDPC code, the hollow and filled circles represents the check nodes and the message nodes seen in Figure 3.1.

the same non-zero entry in the PCM. In other words, the *local tree assumption* ensures that there is no *cycle* in the tree structure of Figure 3.5. It is possible to create a cycle-free Tanner graph, when the PCM is constructed carefully. However, Vardy *et al.* suggested that a cycle-free Tanner graph does not result in good codes [123]. In practice, as long as the short cycles, such as the *length-4* and *length-6* cycles are eliminated, a good decoding performance can be achieved [7]. As pointed out in [10], for almost all randomly constructed LDPC codes, the decoder's performance will be close to that predicted under the *local tree assumption*, provided that the block length of the LDPC code is sufficiently high. Hence, in order to estimate the performance bound of an irregular or regular LDPC code having a given density profile, a comparably long block length has to be chosen. The decoding process is the same as that described in Section 2.5.2. By recursively tuning the *threshold*, the E_b/N_0 boundary of the error-free region may be found asymptotically upon increasing the block length.

As described in Section 2.7, the values $R^a_{i,j}$ and $Q^a_{i,j}$ depicted in Equation 2.31 and 2.32 are iteratively updated. Let us assume that we are using a regular LDPC code, having a constant column weight w_c and row weight w_r for transmission over an AWGN channel. Initially, $Q^a_{m,n}$ will be set to a value corresponding to the demodulator's soft output. When assuming a binary scenario, it is more convenient to represent $Q_{i,j}$ in the form of the log-likelihood ratio of $\log \frac{p(y|x=1)}{p(y|x=-1)}$.

Having initialised $Q^a_{i,j}$, its value is used for updating $R^a_{i,j}$ according to Equation 2.31, or using the "tanh rule" [124] [125] [9] described in Section 2.7.2, as follows:

$$\tanh\left(\frac{R_{i,j}}{2}\right) = \prod_{j' \in \{C_i\}, j' \neq j} \tanh\left(\frac{Q_{i,j'}}{2}\right), \tag{3.7}$$

where

$$\tanh(x/2) = \frac{e^x - 1}{e^x + 1}. \tag{3.8}$$

Following these operations, the values of $R^a_{i,j}$ are used for updating $Q^a_{i,j}$ employing Equation 2.31, while the *a posteriori* probability density of Q is updated using Equation 2.33. This two-stage operation referred to as density evolution assists us in finding the above-mentioned channel-quality related threshold for the belief propagation decoding algorithm. If the channel's E_b/N_0 parameter is above the threshold, the probability density of Q is shifted towards $+/-\infty$. In other words, the LLR of the bits associated with a logical 0 (mapped to $+1$) should converge to $+\infty$, while those related to a logical 1 should tend towards $-\infty$. When communicating over a Gaussian channel, the LLR obeys the Gaussian distribution. When the mean of the Gaussian distributed variable Q tends to ∞, the probability of error decreases as the number of iterations increases. Thus the decoder is capable of achieving an arbitrarily low bit error rate. However, when E_b/N_0 is below the required threshold, the probability density of Q will result in a finite probability of error, which cannot be reduced with the aid of increasing the number of iterations.

3.5 Density evolution using Gaussian approximation

The density evolution process highlighted in the previous section is capable of predicting the threshold, which determines the E_b/N_0 region of error free operation, provided that the code's density profile is specified. The entire process relies on tentatively setting a certain channel parameter, followed by invoking the decoding process using a high number of iterations and exploiting the assumption that there

are no cycles within the parity check matrix in order to verify whether the bit error rate will decrease to a pre-determined low level. If the BER converges to a finite non-zero value, then this implies that the tentatively channel parameter E_b/N_0 is currently lower than the threshold. Hence a higher tentative E_b/N_0 channel parameter is set and the above process is repeated, until the BER becomes lower than a pre-determined low level. Since this process is computationally demanding, Chung [11] suggested the employment of a significantly less complex method for calculating the threshold concerned by using the so-called *Gaussian Approximation*(DEGA) [11]. Let us represent the quantities $Q^a_{i,j}$ and $R^a_{i,j}$ in their Log Likelihood Ratio (LLR) format as $Q_{i,j} = log(Q^0_{i,j}/Q^1_{i,j})$ and $R_{i,j} = log(R^0_{i,j}/R^1_{i,j})$. Then the quantity $Q_{i,j}$ is initialised according to the channel output as:

$$\begin{aligned} Q_{i,j} = LLR_{x_j} &= \log(\frac{P(y_j|x_j=+1)}{P(y_j|x_j=-1)}) \\ &= \log(\frac{\exp(-\frac{E_b}{2\sigma^2}(y_j-1)^2)}{\exp(-\frac{E_b}{2\sigma^2}(y_j+1)^2)}) \\ &= (-\frac{E_b}{2\sigma^2}(y_j-1)^2) - (-\frac{E_b}{2\sigma^2}(y_j+1)^2) \\ &= \frac{E_b}{2\sigma^2}4 \cdot y_j. \end{aligned} \tag{3.9}$$

In Equation 3.9 E_b represents the bit energy, which is set to unity as usual and y_j represents the channel's soft output for the j^{th} symbol. Since y_j is Gaussian distributed with a unity mean and variance of σ^2, the quantity $Q_{i,j}$ calculated in Equation 3.9 is also Gaussian distributed.

Since the product of a constant a and a Gaussian variable having a mean b and variance c will have a mean of ab and a variance of a^2c, hence $Q_{i,j}$ of Equation 3.9 is the product of the constant $4E_b/2\sigma^2$ in Equation 3.9 and the Gaussian variable y_j associated with a mean of unity and a variance of σ^2. The mean of $Q_{i,j}$ is calculated as $4E_b/2\sigma^2 \times 1 = 2/\sigma^2$, while the variance as $(4E_b/2\sigma^2)^2 \times \sigma^2 = 4/\sigma^2$.

Since all the iterative decoding operations are linear in the log arithmetic domain, the quantities $R_{i,j}$ and $Q_{i,j}$ will remain Gaussian during the message passing process. Thus using the mean and the variance of the associated Gaussian distribution adequately characterises the message's probability density [10]. More explicitly, Chung's method approximates the PDF of the received signal using a Gaussian PDF or Gaussian Mixtures[1] [11] for regular and irregular LDPCCs, respectively.

By using the mean of the Gaussian density, the E_b/N_0 threshold to be satisfied may be readily calculated without any degradation of the achievable accuracy [11] of the threshold estimation process. Without loss of generality, we assume that an all-zero codeword is transmitted, since the all-zero codeword is always a valid codeword. The LLR of the message $Q^{(0)}$ received over the AWGN channel can be represented as $L_c y$, where $L_c = 4a\frac{E_b}{2\sigma^2}$ is defined as the channel reliability [115]. Assuming that the BPSK modulator used maps a logical 0 to +1, and that the bit energy equals unity, we have $L_c = \frac{2}{\sigma^2}$. Since y is Gaussian distributed having a mean of unity in case of BPSK modulation and a variance of σ^2, the LLR of the message $Q^{(0)}$ has a mean of $2/\sigma^2$ and a variance of $4/\sigma^2$. The superscript 0 of Q indicates that $Q^{(0)}$ was received from the channel before the first iteration, while $Q^{(l)}$ denotes the value of Q after l iterations. Furthermore, m_x represents the mean of the variable x.

[1]Since the message nodes of the irregular LDPC codes have various weights, thus a node will get i.i.d. messages from its neighbours, where each of these messages is a random mixture of different Gaussian probability density functions of its neighbours weighted by different weighting factor.

The mean of the quantity $Q^{(l)}$ following the l_{th} iteration can be calculated with the aid of Equation 2.32 but using additions in the logarithmic domain instead of product seen in Equation 2.32 as follows [11]:

$$m_{Q^{(l)}} = m_{Q^{(0)}} + (w_c - 1)m_{R^{(l-1)}}. \tag{3.10}$$

When interpreting Equation 3.10 we notice that the second term on the right simply implies the multiplication of the mean of the quantity $R^{(l)}$ at the $(l-1)^{st}$ iteration by (w_c-1), where w_c represents the average column weight of the LDPCC's parity check matrix. This is, how we take into account the extrinsic information provided by the $(w_c - 1)$ other non-zero entries. Therefore, the quantity $m_{Q^{(l)}}$ still only reflects the mean of the *extrinsic* information, corresponding to Equation 2.32. Initially we have $m_{R^{(0)}} = 0$ since the initial information from any check node before the commencement of iterations is 0. Using this multiplication instead of summation is valid, because even though the $(w_c - 1)$ R quantities found in the different rows of the w_c column considered will have a different polarity value, they are independent and identically distributed (i.i.d) variables, thus their means are identical.

The update of the quantity R was defined in Equation 2.31, and in Equation 3.7 using the *tanh rule* [124] [125] [9]. By taking the expectation of both sides of Equation 3.7, we arrive at:

$$E\left[\tanh\left(\frac{R^{(l)}}{2}\right)\right] = E\left[\tanh\left(\frac{Q^{(l)}}{2}\right)\right]^{k-1}. \tag{3.11}$$

Suppose the variable $R^{(l)}$ is Gaussian distributed having a mean of $m_{R^{(l)}}$ and a variance of $2m_{R^{(l)}}$, where the expectation $E[\tanh\left(\frac{R^{(l)}}{2}\right)]$ depends only on the mean of $R(l)$, we have

$$E\left[\tanh\left(\frac{R^{(l)}}{2}\right)\right] = \frac{1}{\sqrt{4\pi m_{R^{(l)}}}} \int_R \tanh\left(\frac{r}{2}\right) e^{-\frac{(r-m_{R^{(l)}})^2}{4m_{R^{(l)}}}} dr. \tag{3.12}$$

Hereby we will introduce a new function $\phi(x)$ for the sake of later convenience during the iterative calculation of the mean of the variable Q, where $\phi(x)$ is formulated as [11]:

$$\phi(x) = \begin{cases} 1 - \frac{1}{\sqrt{4\pi x}} \int_R \tanh\left(\frac{r}{2}\right) e^{-\frac{(r-x)^2}{4x}} dx, & if \quad x > 0 \\ 1, & if \quad x = 0. \end{cases} \tag{3.13}$$

From Figure 3.6, it can be readily seen that $\phi(x)$ is continuous and monotonically decreasing over the interval $[0, \infty)$.

Given the definition of $\phi(x)$ in Equation 3.13, the formula derived from Equation 3.10 for recursively calculating the mean of the quantity $R^{(l)}$ following the l_{th} iteration is given by [11]:

$$m_{R^{(l)}} = \phi^{-1}\left(1 - \left[1 - \phi\left(m_{Q^{(0)}} + (j-1)m_{R^{(l-1)}}\right)\right]^{(k-1)}\right), \tag{3.14}$$

where $m_{Q^{(0)}} = 0$ is the initial value of $m_{Q^{(0)}}$. Hence, the mean of the quantity $Q^{(l)}$ can be recursively updated by using Equation 3.10. As argued before, the quantity $Q^{(l)}$ and $R^{(l)}$ are Gaussian distributed with a mean of $m_{Q^{(l)}}$ and $m_{R^{(l)}}$, respectively, and their variances are $2m_{Q^{(l)}}$ and $2m_{Q^{(l)}}$, respectively. Thus when the value of $m_{Q^{(l)}}$ or $m_{R^{(l)}}$ tends to infinity, the corresponding Gaussian PDFs move

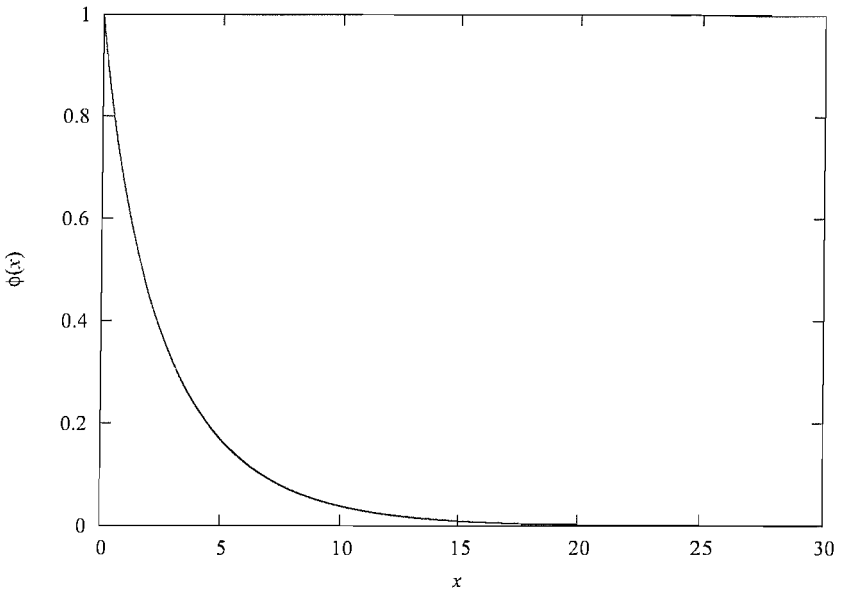


Figure 3.6: The function $\phi(x)$ in Equation 3.13

towards infinity. Hence the error probability, calculated by integrating the Gaussian Q-function over the interval of $(-\infty, 0]$, where erroneous decisions are encountered, tends to zero. Hence with the aid of this method, we can fix the a value of the E_b/N_0 channel parameter and run this calculation for a sufficiently high number of iterations. If the mean m_{Q^l} and m_{R^l} tends to zero, then we know that the E_b/N_0 channel parameter is above the threshold required for maintaining error free communications. However, when the chosen E_b/N_0 channel parameter is below the threshold, the mean m_{Q^l} and m_{R^l} will converge to a fixed non-zero value.

By the same token, the equation required for calculating the mean m_{Q^l} and m_{R^l} of irregular LDPC codes can be derived in the following manner. Owing to the irregularity of the LDPCC's density distribution, the messages $R^{(l)}$ and $Q^{(l)}$ that are exchanged between the nodes are Gaussian mixtures rather than having a Gaussian PDF as in the regular LDPC scenario. Hence the mean of $R^{(l)}$ and $Q^{(l)}$ are determined by taking into account the specific edge perspective densities $\lambda_e(x)$ and $\rho_e(x)$. Corresponding equation of irregular LDPCCs is given by [11] :

$$m_{R^{(l)}} = \sum_{t=2}^{w_{r(max)}} \rho_t \phi^{-1}\left(1 - \left[1 - \sum_{i=2}^{w_{c(max)}} \phi\left(m_{Q^{(0)}} + (i-1)m_{R^{(l-1)}}\right)\right]^{(t-1)}\right) \tag{3.15}$$

Chung provided an example for illustrating the accuracy of this Gaussian approximation algorithm [11]. A half-rate irregular LDPCC's density profile is given by [10]:

$$\begin{aligned} \lambda_e(x) &= 0.23403x + 0.21242x^2 + 0.114690x^5 + 0.10284x^6 + 0.30381x^{19}, \\ \rho_e(x) &= 0.71875x^7 + 0.28125x^8, \end{aligned} \tag{3.16}$$

while the effect of the coded block-length and that of the number of iterations are not addressed here, since the Gaussian approximation is used to predict the asymptotic performance of an infinite long codeword employing a sufficiently high number of iterations. By using the density evolution technique of Section 3.4, the E_b/N_0 threshold required by the LDPCC having the density profile given

in Equation 3.16 when communicating over an AWGN channel was found to be $0.2923dB$. When using the Gaussian approximation of DE instead, the corresponding threshold was $E_b/N_0 = 0.5dB$, ($0.47dB$ in [11]), resulting in a deviation of $0.21dB$ from the threshold found by density evolution. Hence we may conclude that the Gaussian approximation is reasonably accurate. In Figure 3.7 and 3.8, the density of the message $R^{(l)}$ after each iteration is plotted for two different E_b/N_0 values above the threshold of $E_b/N_0 = 0.5dB$. Still assuming that the all-zero codeword was transmitted and that a logical 0 was mapped to +1, the two settings were $E_b/N_0 = 5.288dB$ and $E_b/N_0 = 9.269dB$, which corresponds to a noise standard deviation of $\sigma = 0.544$ and $\sigma = 0.344$, respectively. From Figures 3.7 and 3.8, it can be observed that the density of the message moves to the right towards ∞ after each iteration. We can also observe that the higher E_b/N_0 value leads to a faster convergence, reaching a logarithmic mean value in excess of 40 within a mere 3 iterations, as in Figure 3.8 compared to the results in Figure 3.7 where the logarithmic mean value exceeds a value of 40 after a number of 10 iterations. This is because when the E_b/N_0 is well above the threshold value, there are less bits contaminated by the AWGN, and also the bits staying in their correct states are more confident, i.e. having a higher LLR, comparing to the scenario of encountering a lower E_b/N_0. In Figure 3.9, the evolution process of an LDPC code using the same density profile in Equation 3.16 communicating over the AWGN channel associated with $E_b/N_0 = -1.896dB$ is characterised. This E_b/N_0 value is $2.396dB$ lower than the required threshold. Observe in Figure 3.9 that the associated Gaussian PDF curve stays at about the same average LLR value of 0 while the decoder iterates, since the decoder is unable to enhance the LLRs owing to the excessive noise level. As seen from Figure 3.10, where we have $E_b/N_0 = -1.896dB$, the mean LLR remains near zero, regardless of the number of iterations.

The Gaussian approximation allows us to check whether an LDPC code associated with a given density distribution is capable of achieving error-free transmissions upon increasing the number of iterations. Furthermore, it provides a mean of estimating, how rapidly the error probability may be reduced as a function of the number of iterations, as seen in Figures in 3.7, 3.8 and 3.9. As suggested by Equations 3.14 and 3.15, the mean of the quantity $R^{(l)}$ emerging from the l^{th} iteration is recursively updated with the mean of $R^{(l-1)}$ after the previous iteration. Thus by observing the difference of the mean between two consecutive iterations, we may be able to anticipate slow and fast converging phases, as noted in [61]. We will use the notation $\triangle m_{R^{(l)}}$ for representing the increment of the mean value of $R^{(l)}$ after the l^{th} iteration. Assuming again that the density profile given in Equation 3.16 is used, the mean increment curves of the three previously considered experimental E_b/N_0 values, i.e. $E_b/N_0 = 5.288dB$, $E_b/N_0 = 9.269dB$ and $E_b/N_0 = -1.896dB$ are plotted in Figure 3.11 and Figure 3.12.

As seen in Figure 3.11, both curves are well above zero, which implies benefiting from desirable increase of the quantity $\triangle m_R$. The lower curve corresponds to $E_b/N_0 = 5.288dB$, which is $4.788dB$ above the threshold. After four iterations, the increment of $\triangle m_R$ reached a saturation, and the mean value of R keeps increasing until after ten iterations, moving towards ∞. By comparing to the right hand side illustration of Figure 3.10, the curve corresponding to $E_b/N_0 = 5.288dB$ initially has a slightly lower gradient for the first two-three iterations. This matches the results seen in Figure 3.11, since $\triangle m^R$ is lower for the first two-three iterations than later. The upper curve, which represents $E_b/N_0 = 9.269dB$, has a higher gradient, tending towards ∞ after the 3^{rd} iteration.

By contrast, the curve representing the scenario of communicating over an AWGN channel at

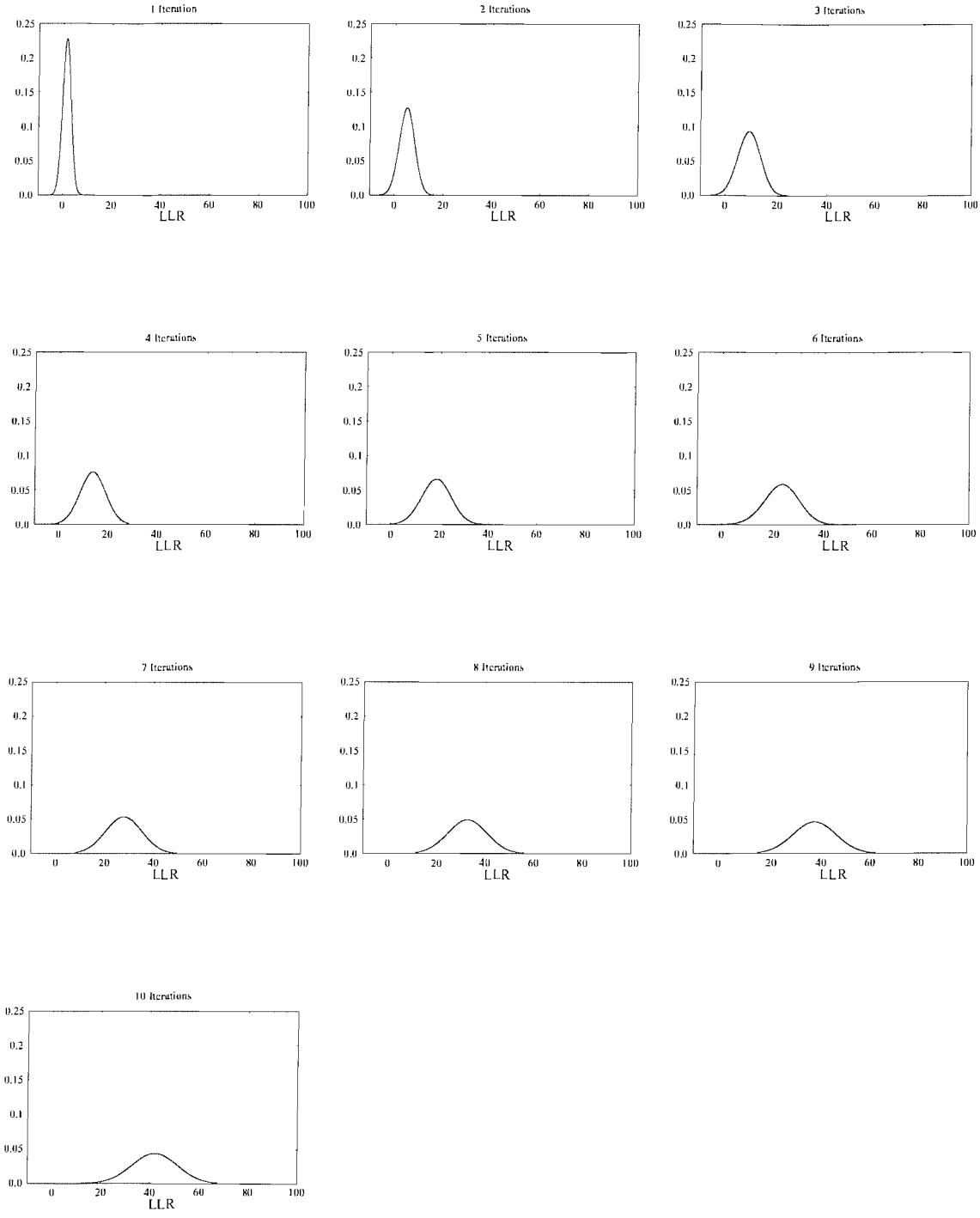


Figure 3.7: The density evolution of the message R using the Gaussian approximation for the half-rate irregular LDPCC associated with the density distribution given in Equation 3.16, when communicating over an AWGN channel at $E_b/N_0 = 5.288dB$.

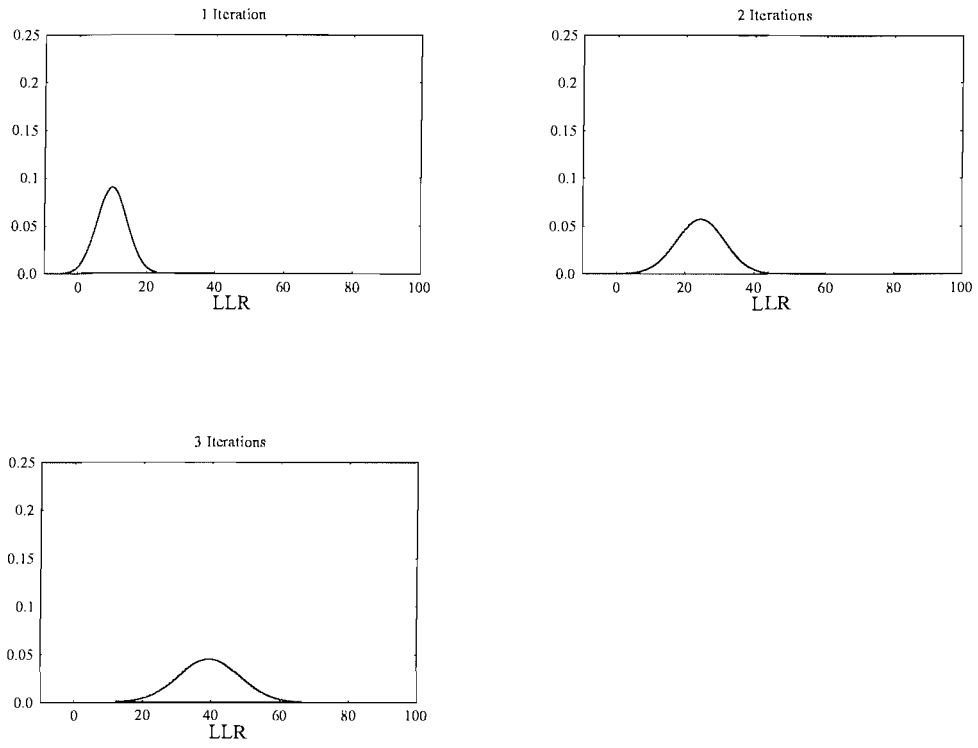


Figure 3.8: The density evolution of the message R using the Gaussian approximation for the half rate irregular LDPCC associated with the density distribution given in Equation 3.16, when communicating over an AWGN channel at $E_b/N_0 = 9.269dB$.

$E_b/N_0 = -1.896dB$, is plotted in Figure 3.12. The y-axis is plotted on a logarithmic scale for clearer illustration. The quantity $\triangle m^R$ decreases as a function of the number of iterations, and eventually tends to zero after 15 iterations. As long as we have $\triangle m^R = 0$, further iterations will not provide any benefits and thus the error-probability cannot be reduced any further. This result is consistent with Figure 3.10 and Figure 3.9.

3.6 LDPC density distribution optimisation

As described in the previous section, the Gaussian approximation algorithm can be used for fairly accurately estimating a certain code's behaviour for a specific density distribution. Thus, this method facilitates the search for an optimum density distribution in conjunction with one or several design constraints, such as a given code rate or average row weight.

The technique of searching for a beneficial density distribution can be described as follows. Commencing from a randomly chosen density profile, such as for example MacKay's regular construction [56], we assign an appropriate E_b/N_0 channel parameter value and run the density evolution in Section 3.4 for a pre-determined number of iterations j. The error probability ϵ arrived at after j

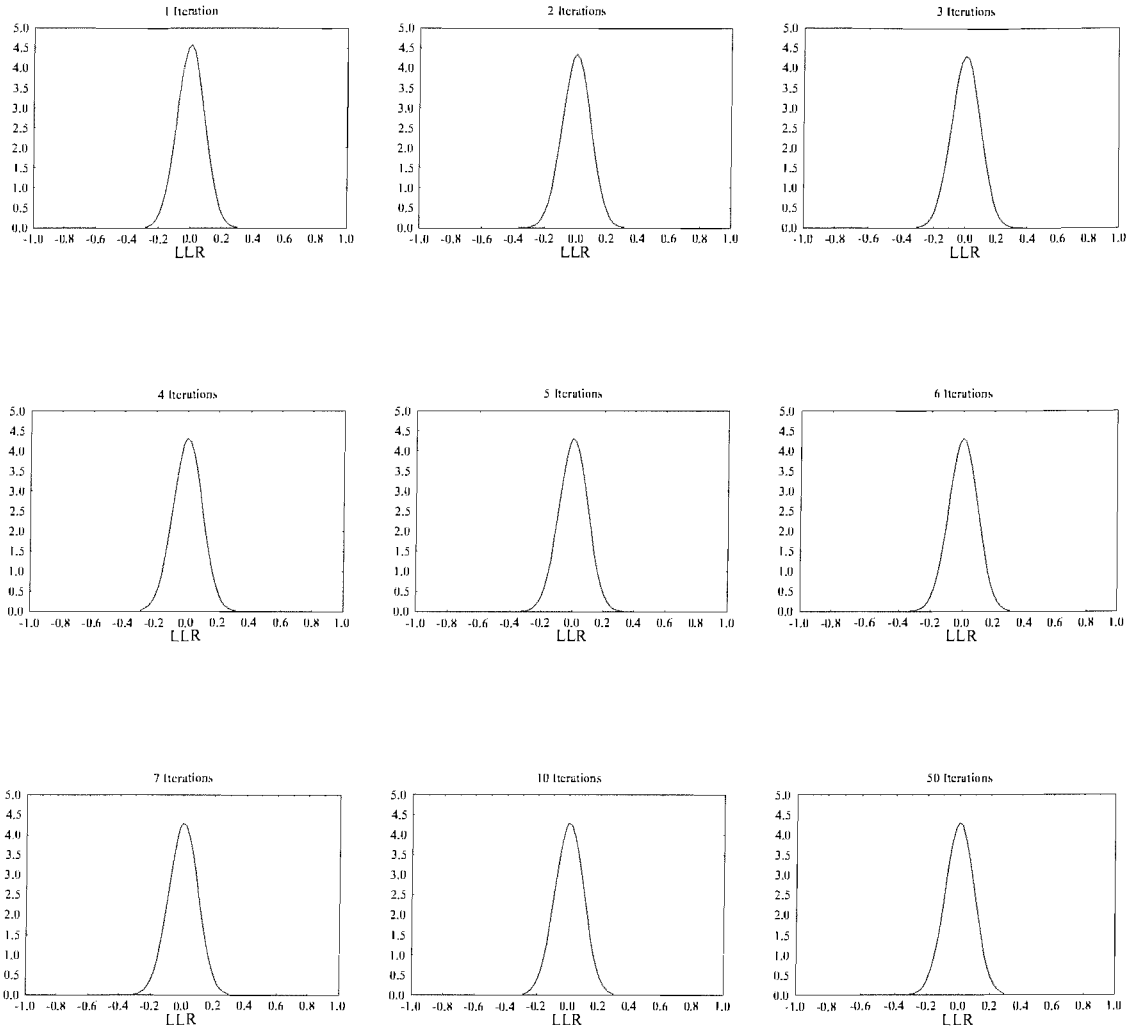
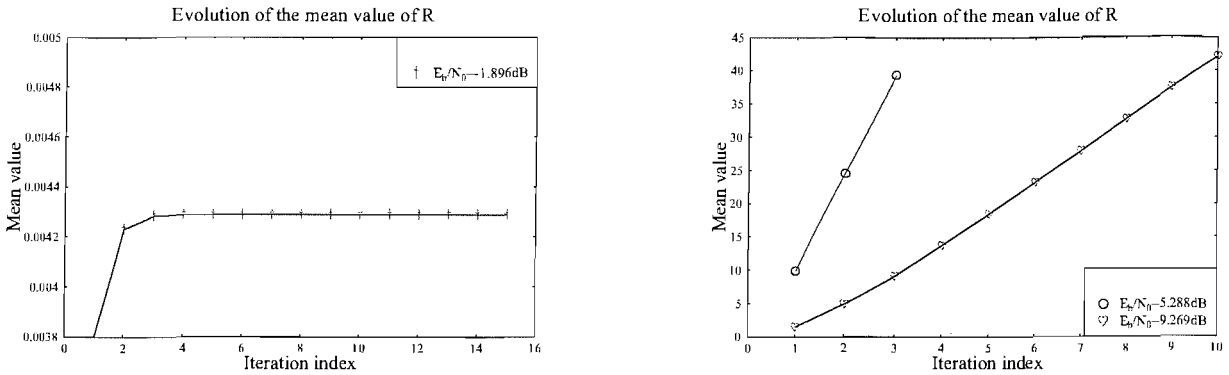


Figure 3.9: The density evolution of the message R using the Gaussian approximation for the half rate irregular LDPCC associated with the density distribution given in Equation 3.16, when communicating over an AWGN channel at $E_b/N_0 = -1.896dB$.



(a) (b)

Figure 3.10: The evolution of the mean of the message R for the half rate irregular LDPCC associated with density distribution given in Equation 3.16, when communicating over an AWGN channel a $E_b/N_0 = 5.288dB$, $E_b/N_0 = 9.269dB$ and $E_b/N_0 = -1.896dB$.

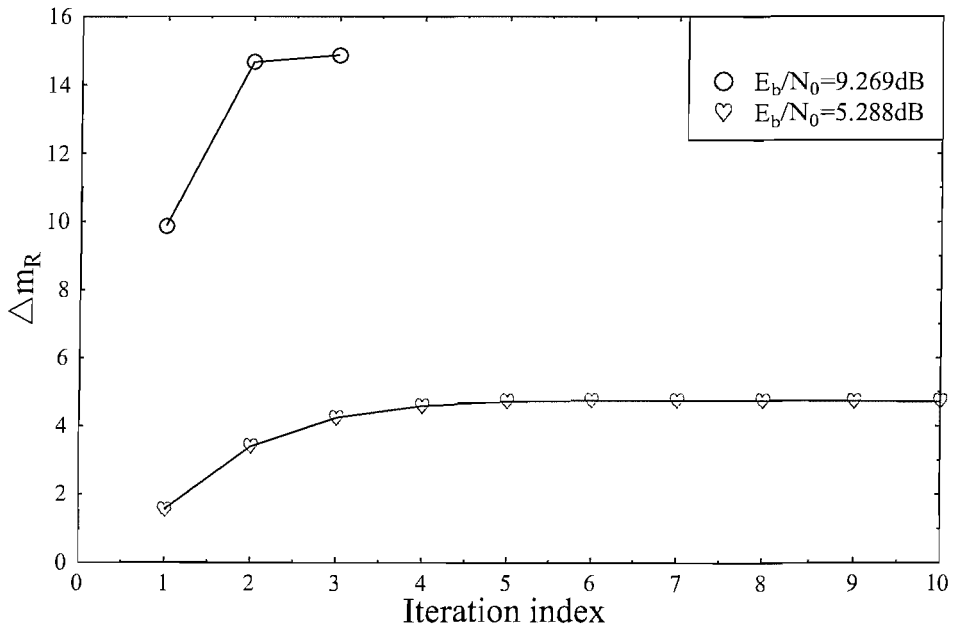


Figure 3.11: The evolution of the quantity $\triangle m_R$ for the half-rate irregular LDPCC associated with the density distribution given in Equation 3.16, when communicating over an AWGN channel at $E_b/N_0 = 5.288dB$ and $E_b/N_0 = 9.269dB$.

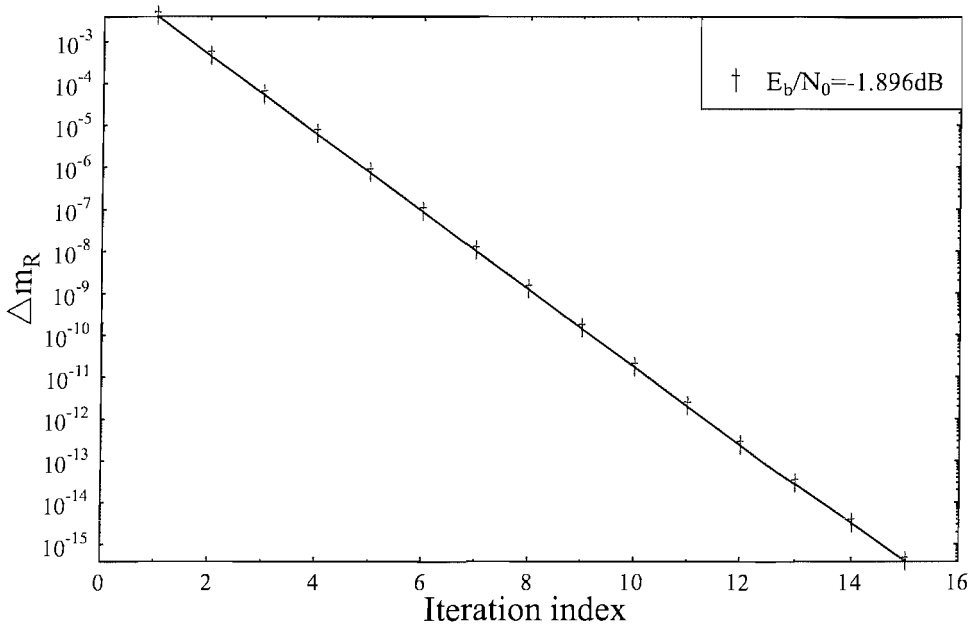


Figure 3.12: The evolution of the quantity Δm_R for the half-rate irregular LDPCC associated with the density distribution given in Equation 3.16, when communicating over an AWGN channel at $E_b/N_0 = -1.896dB$.

iterations will be calculated as a result of the density evolution process. Furthermore, if the Gaussian approximation method introduced in Section 3.5 is applied, we will determine the mean of the PDF after j iterations. The corresponding density profile will be stored as best density distribution so far. Then we slightly change the profile by changing the coefficients of the density distribution exemplified by Equation 3.4, while satisfying the constraint that $\lambda(1) = 1$ and $\rho(1) = 1$. This is followed by repeating the DE process described above and checking whether any BER improvement was achieved. More explicitly, the associated improvement is expected to result in a lower error probability than ϵ after j iterations, or reaching an error probability of ϵ using a lower number of iterations than j. When the Gaussian approximation is applied, the associated improvement manifests itself in terms of achieving a higher mean LLR value at a given number of iterations. If there is any BER improvement in comparison to the previous best density distribution, then the new density profile will be stored as the best density distribution. This process is repeated as many times as it is affordable according to the complexity constraints imposed.

This density distribution optimisation process is computationally complex. However, significant computational efforts may be saved, if we reduce the size of the search space. Richardson *et. al* found [9] that an attractive density distribution is expected to have a low number of non-zero elements. As far as the row weight is concerned, Chung showed in [11] that the rows of the PCM should be designed to have only one or two consecutive non-zero elements. As for the column weights, the

corresponding exponents of the polynomial may be limited to including terms having an exponent of 1, 2 and the maximum exponent of $w_c^{max} - 1$, plus perhaps a few further terms having other exponents in between [61].

3.7 Variability of error protection versus bit-position in irregular LDPC codes

As illustrated in Section 2.9.2 in the context of regular-construction LDPC codes, each individual coded bit may be regarded as equally protected. However, the equal-protection property applies to the family of regular LDPC codes only and for the irregular LDPC codes discussed in this chapter, the bits associated with a high column weight are more strongly protected. In other words, their soft-value tends to converge faster to its correct transmitted value as a function of the number of iterations. This property is consistent with Ruby's original intuition concerning the family of irregular LDPC codes, suggesting that variable nodes associated with a higher weight may be expected to converge faster to their correct value. This is illustrated in Figures 3.13 and 3.14 by using a LDPC code having the same block-length, namely 1000 bits and the same coding rate of $r = 1/4$ as the regular LDPC example provided in Figure 2.16 and Figure 2.17 of Section 2.9.2. In this experiment, rather than evaluating the BER of each individual bit, the average BER of all the nodes having a particular column weight is determined, since they are expected to be similar. Furthermore, in contrast to constructing the parity check matrix in a regular fashion as in Chapter 2, we will use the optimum density distribution calculated by Chung's degree optimisation program found in [126]. Optimality in this context implies finding the distribution profile achieving the best known near-capacity performance at a specific coding rate. The density distribution is defined in terms of the node-oriented description defined in Equation 3.17 as:

$$\begin{aligned} \lambda_n(x) &= 0.6x + 0.21504x^2 + 0.06571x^4 + 0.04541x^5 + 0.07268x^{11} + 0.00116x^{12}, \\ \rho_n(x) &= 0.55556x^3 + 0.44444x^4. \end{aligned} \tag{3.17}$$

Similar to the example provided in Section 2.9.2, our BER evaluations were carried out for a maximum of ten iterations at both $E_b/N_0 = 1.5dB$ and $2.5dB$. The related simulation results are provided in Figures 3.13 and 3.14.

We can see in Figures 3.13 and 3.14that the average BER of all the message nodes having different column weights is similar and is between about 15% to 20% at the output of the demodulator, before the commencement of iterations. However, when the iterative decoding starts, the high-weight message nodes result in a reduced BER compared to the low-weight message nodes. Observe in Figure 3.13 that between the first iteration and the third iteration, nodes having a higher column weight tend to converge to their final values faster owing to having a higher number of non-zero entries in the column, which in turn provide more information for the message node. Consequently, during the subsequent iterations most of the high-weight nodes are associated with correct bit values and hence they are capable of providing correct information for the nodes having a lower column weight. Hence the low-weight nodes tend to steadily converge towards their correct values at a near-linear rate, while the convergence of the higher-weight nodes becomes slower during this phase.

In summary, we may conclude that each individual message nodes of an irregular LDPC code tends

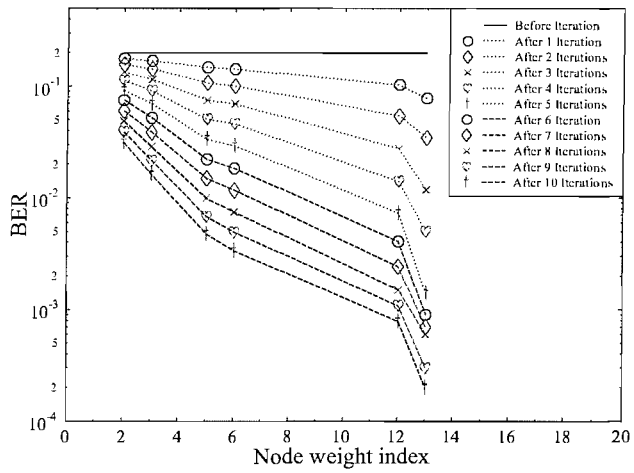


Figure 3.13: Average BER for all the nodes having a particular column weight after each decoding iteration, when communicating over an AWGN channel at $E_b/N_0 = 1.5dB$. The code rate was $r = 0.25$ and the density distribution was given in Equation 3.17.

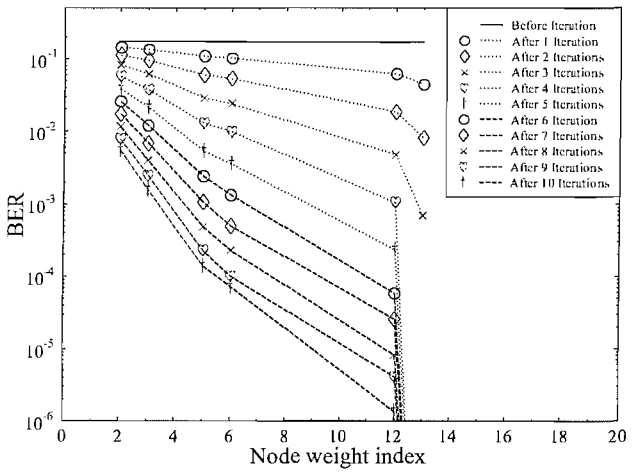


Figure 3.14: Average BER for all the nodes having a particular column weight after each decoding iteration, when communicating over an AWGN channel at $E_b/N_0 = 2.5dB$. The code rate was $r = 0.25$ and the density distribution was given in Equation 3.17.

to be unequally protected, especially if the maximum number of iterations is limited. However, owing to the specific iterative decoding mechanism of irregular LDPC codes, if we allow a higher number of iterations, the higher-weight nodes will assist the low-weight nodes in converging towards their correct value. Thus, provided that the affordable decoding complexity is not unduly limited, the various nodes may be deemed to be near-equally protected.

3.8 Parity check matrix construction for irregular LDPC codes

In this section we continue our discourse by introducing two different types of LDPC matrix construction techniques. More explicitly, two different methods of choosing the density distribution for an irregular LDPC parity check matrix are presented. The first one is based on using Richardson's density evolution concept [61], or Chung's Gaussian approximation method [11] designed for finding the specific density distribution which offers the best asymptotic performance, assuming an infinite block-length and a high number of iterations. This construction is capable of achieving a performance as close to the Shannon limit as possible.

The second method to be portrayed was devised by Yang *et al.* [127]. This method has the potential of providing a lower error floor for high rate LDPC codes in comparison to Richardson's approach, at the cost of a slightly degraded performance at low SNRs.

3.8.1 Richardson's construction method

Again, Richardson's construction [61] concerned here is based on using a specific density distribution, which guarantees a performance as close to the capacity limit as possible. The technique of finding this density distribution has been described in Section 3.6. The corresponding density distribution can be calculated with the aid of the publically available program found at [126]. The performance of the family of LDPC codes using this construction method will be evaluated in this section in comparison to that of turbo convolutional codes having similar parameters, which were summarised in Table 2.23, while the corresponding puncturing patterns are given in Table 2.24. Similarly, the parameters of the LDPC codes studied were listed in Table 2.22. The corresponding density profiles for each coding rate considered are given as follows.

Degree distribution for rate = $r = 1/3$:

$$\begin{aligned} \lambda_n(x) &= 0.54453x + 0.22746x^2 + 0.00341x^4 + 0.15449x^5 + 0.07011x^{19}, \\ \rho_n(x) &= 0.82353x^5 + 0.17647x^6. \end{aligned} \tag{3.18}$$

Degree distribution for rate = $r = 1/2$:

$$\begin{aligned} \lambda_n(x) &= 0.48603x + 0.27696x^2 + 0.06258x^5 + 0.10026x^6 + 0.07417x^{19}, \\ \rho_n(x) &= 0.27273x^7 + 0.72727x^8. \end{aligned} \tag{3.19}$$

Degree distribution for rate = $r = 2/3$:

$$\begin{aligned} \lambda_n(x) &= 0.42725x + 0.27089x^2 + 0.16988x^4 + 0.13198x^{19}, \\ \rho_n(x) &= 0.51613x^14 + 0.48387x^{15}. \end{aligned} \tag{3.20}$$

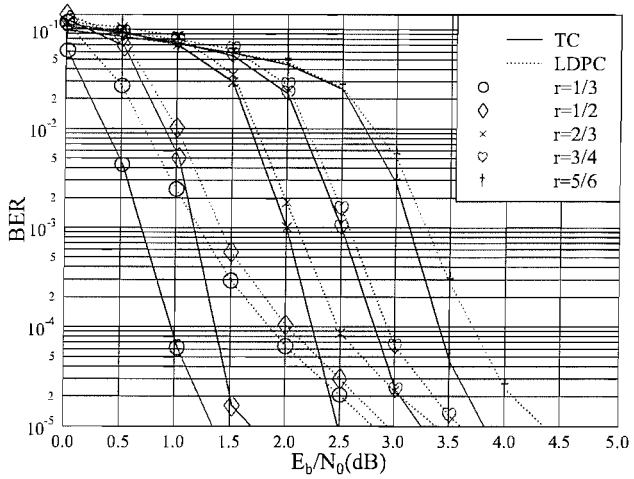


Figure 3.15: BER performance of irregular LDPC codes constructed using the density profiles specified in Equations 3.18 to 3.22 based on Richardson's construction method, benchmarked against turbo convolutional codes using the parameters summarised in Table 2.23 and Table 2.24, when communicating over an AWGN channel. BPSK modulation applied. The effective throughputs of both codes are 1/3, 1/2, 2/3, 3/4, 5/6 bps according to their respective code rates.

Degree distribution for rate $= r = 3/4$:

$$\begin{aligned} \lambda_n(x) &= 0.40056x^1 + 0.29759x^2 + 0.12986x^4 + 0.03807x^5 + 0.13392x^{19}, \\ \rho_n(x) &= x^{20}. \end{aligned} \tag{3.21}$$

Degree distribution for rate $= r = 5/6$:

$$\begin{aligned} \lambda_n(x) &= 0.37106x + 0.3348x^2 + 0.05634x^4 + 0.10363x^5 + 0.13416x^{19}, \\ \rho_n(x) &= x^{31}. \end{aligned} \tag{3.22}$$

Comparing the results illustrated in Figure 3.15 and Figure 3.16 to the results portrayed in Figure 2.20 and Figure 2.21 using regular construction, the irregular construction LDPC codes outperform the regular codes, when the BER is higher than 10^{-3}. However, the irregular construction LDPC codes have a tendency to exhibit an error floor at higher E_b/N_0 values. This disadvantageous property of the irregular codes using Richardson's method is a consequence of employing the density evolution algorithm. This is because in Gallager's seminal paper [1] it has been shown that the code's column weight should be higher than or equal to three for the sake of ensuring that the code's distance increases linearly with the block-length. However, if weight two columns exists in the PCM, the achievable distance increases only logarithmatically, rather than linearly with the block-length [1]. Secondly, as pointed out in [127], when there are cycles which involve only weight-two columns, these cycles constitute the 'weakest link' in the parity check matrix. This is because during the vertical update of message $Q^a_{i,j}$ of Equation 2.32 in the PCM, the message $Q^a_{i,j}$ of the two non-zero entries in the j^{th} column is only affected by the value of the other non-zero entry in the column, rather than

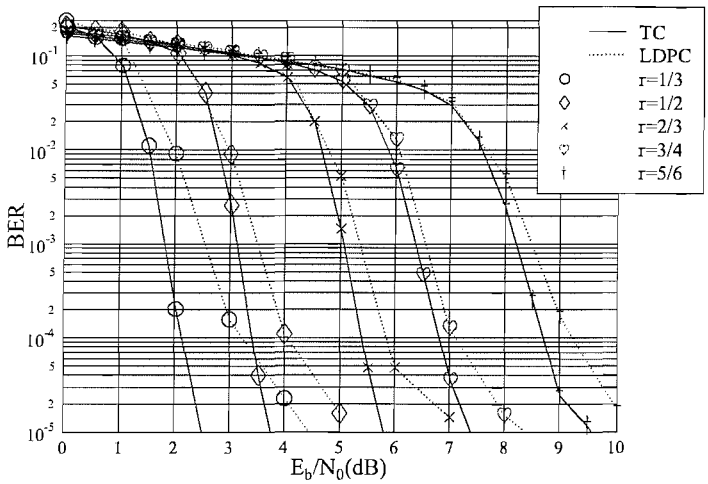


Figure 3.16: BER performance of irregular LDPC codes constructed using the density profiles specified in Equations 3.18 to 3.22 based on Richardson's construction method, benchmarked against turbo convolutional codes using the parameters summarised in Table 2.23 and Table 2.24, when communicating over an uncorrelated Rayleigh fading channel. BPSK modulation applied. The effective throughputs of both codes are 1/3, 1/2, 2/3, 3/4, 5/6 bps according to their respective code rates.

by two or more entries. Hence, during the vertical update process of $Q^a_{i,j}$, the non-zero entries are getting only limited parity information from other check rows. As suggested by Yang in [127], cycles which involve only weight-two columns are preferably avoided during the construction of the PCM. Let us now illustrate the BER and FER performance of both the third-rate and half-rate LDPC codes in Figure 3.17 using the same density profiles, as in Equation 3.18 and Equation 3.19, respectively while employing two different BER and FER evaluation techniques. Firstly, as usual, we compare the decoded bit sequence to the originally transmitted bits for the calculation of the *Exact BER* and *Exact FER*, as seen in Figure 3.17. The second evaluation method is based on exploiting the PCM for checking whether the decoded codeword is a legitimate one. If the codeword is deemed to be a legitimate codeword, despite that it might have been incorrectly decoded, it is considered as error-free. More explicitly, undetected errors resulting in a legitimate codeword are not taken into account, when determining the *Detected BER* and *Detected FER* in Figure 3.17.

As demonstrated in Figure 3.17, both LDPC codes experience undetected errors in the low BER range. As illustrated in Figure 2.18 and Figure 2.19 of Section 2.9.3, LDPC codes employing a regular construction associated with a column weight three have a very low probability of undetected errors even at a block-length of a mere 200 bits. By contrast, in case of irregular LDPC codes the significantly longer block-length of 3000 bits produced incorrectly decoded codewords, as seen in Figure 3.17. This comparison suggests that the distance properties of the irregular LDPC codes having weight-two columns are worse than those of the family of regular LDPC codes having a minimum weight of three. When an undetected decoding error occurred, the decoder was unable to flag this event and hence in Figure 3.17 the *Detected BER* and *Detected FER* curves do not consider the effects of the undetected

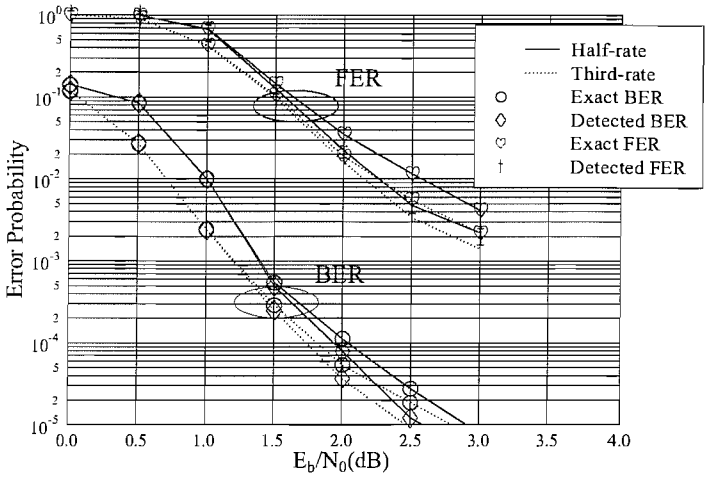


Figure 3.17: BER and FER performance of irregular LDPC codes constructed using the density profiles of Equations 3.18 and 3.19 based on Richardson's construction method, when communicating over an AWGN channel.

errors. This is why these curves appear to be more optimistic than the *Exact* curves.

The evolution of the nodes' average LLR with respect to the LDPCC's iteration index was demonstrated in Figures 3.2, 3.3 and 3.4. Recall from Section 3.1 that the column weight should be high and the row weight should be low. However, as it was shown in [9], the presence of a weight-two column is necessary for ensuring a beneficial irregularity of the code, in order to counter-balance the effects of low-weight rows. In Gallager's seminal paper [1] it has been shown that the column weight should be equal to or higher than three for the sake of ensuring that the code's minimum distance increases linearly with the block-length. However, if the column weight is only two, the achievable distance increases only logarithmically with the block-length [1]. Hence the weight-two columns constitute the 'weakest link' in the entire parity check matrix and the cycles which involve only weight-two columns should be preferably avoided, if possible. The irregular constructions used in [56] were shown to be superior to their regular counterparts, provided that the block-length was sufficiently long. However, in real-time interactive multimedia communications the system's delay constitutes a crucial design constraint and viewed from this perspective the block-length used in [56] was somewhat excessive.

3.8.2 Yang's construction method

For the sake of mitigating the problems arising from the existence of undetected decoding errors in Figure 3.17 when Richardson's PCM construction method is applied, in [127] Yang suggested avoiding having low-weight message nodes, such as weight-three or weight-four message nodes. However, in the context of the irregular PCM construction, we have to apply weight-2 columns in the PCM for the sake of maintaining a sparse PCM in order to counter-balance the employment of some high weight columns. By employing this method, the density profiles of a half-rate irregular LDPC code according

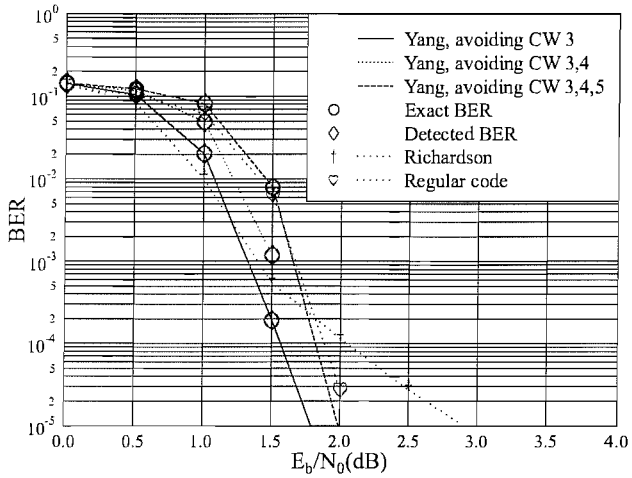


Figure 3.18: BER performance of the (3000,1500) irregular LDPC codes employing the PCM constructed using the density profiles of Equation 3.23 to 3.25, when communicating over an AWGN channel. A (3000, 1500) irregular LDPC code using Richardson's density profile given in Equation 3.19 and a regular LDPC code, where all the message nodes had a weight of three were used as benchmarkers.

to Yang's suggestion may be determined by using the density profile optimisation program found at [126], yielding:

$$\begin{aligned} \lambda_n(x) &= 0.48996x + 0.41282x^3 + 0.07557x^{18} + 0.02165x^{19}, \\ \rho_n(x) &= x^8. \end{aligned} \tag{3.23}$$

$$\begin{aligned} \lambda_n(x) &= 0.49745x + 0.44548x^4 + 0.05707x^{19}, \\ \rho_n(x) &= 0.27273x^7 + 0.72727x^8. \end{aligned} \tag{3.24}$$

$$\begin{aligned} \lambda_n(x) &= 0.49746x + 0.47729x^5 + 0.02525x^{19}, \\ \rho_n(x) &= 0.27273x^7 + 0.72727x^8. \end{aligned} \tag{3.25}$$

More explicitly, Equation 3.23 gives the density profile of the $r = 0.5$ LDPC code according to Yang's suggestion in [127], when the presence of weight-three message nodes is avoided. We can see from Equation 3.23 that the minimum weight of the message node is two, but there is no weight-three term in it. By following a similar approach, the density profile which avoids having both weight-three and weight-four nodes is given in Equation 3.24. Furthermore, in Equation 3.25 all the weight-three, weight-four and weight-five message nodes are avoided. The performance of a $(3000, 1500)$ irregular LDPC code utilising the above three different density profiles outlined in Equations 3.23 to 3.25 and communicating over an AWGN channel is characterised in Figure 3.18. This (3000, 1500) irregular code is benchmarked against the half-rate irregular LDPC code featuring in Figure 3.17 and against a regular LDPC code, where all the message nodes had a weight of three at an identical rate and block-length.

In Figure 3.18 the BER performance of the irregular LDPCCs using Yang's density profiles given by Equations 3.23 to 3.25 were evaluated using both the *Exact BER* and *Detected BER*, as defined previously in the context of Figure 3.17. The half-rate *Exact BER* curve previously shown in Figure 3.17 was reproduced here together with an identical-rate identical-block-length regular LDPC code as the benchmarkers. By comparing Figure 3.18 and Figure 3.17 we can observe that by avoiding some of the low-weight message nodes, for example the weight-three nodes during the construction of the PCM, the LDPC codes constructed according to Yang's suggestion become less prone to experiencing undetected errors, which suggests that the distance properties of these LDPC codes are better than those of the code constructed using Richardson's method. Upon avoiding more and more low-weight message nodes, the BER curves decay faster in the high-SNR region at the cost of a degraded residual BER performance. In conclusion, the LDPC code using the density profile specified in Equation 3.23 constitutes the best solution in Figure 3.18.

Furthermore, in practice often near-unity-rate coding schemes are invoked for the sake of increasing the system's overall effective throughput. When the coding rate increases, the PCM will have an increased number of columns and reduced number of rows. It was shown in [127] that the maximum number of weight-two columns that may be encountered before a cycle involving columns all having a weight of two is created is $(N-K-1)$, corresponding to the number of rows in the parity check matrix minus 1. Thus, limiting the number of weight-two message nodes, especially in a high-rate scenario, assists in reducing the error floor. Hereby, we would like to characterise the performance of a $r = 0.82$ (4161,3430) irregular LDPC code, when the number of weight-two message nodes is limited to 730 according to Yang's suggestion. This code will also be benchmarked against Richardson's proposed density profile. The designs according to Richardson's density profile as well as to Yang's approach are as follows:

Richardson's density profile :

$$\begin{aligned} \lambda_n(x) &= 0.398x^1 + 0.38118x^2 + 0.22082^7, \\ \rho_n(x) &= 0.90411x^{20} + 0.09589x^{21}. \end{aligned} \tag{3.26}$$

Yang's density profile :

$$\begin{aligned} \lambda_n(x) &= 0.0002x^0 + 0.17543x^1 + 0.64797x^2 + 0.1764x^7, \\ \rho_n(x) &= 0.90411x^{20} + 0.09589x^{21}. \end{aligned} \tag{3.27}$$

We infer from Equation 3.26, which gives the Richardson's density profile, that the proportion of weight-two message nodes is 0.398, where the number of weight-two message nodes is calculated as $0.398 \times 4161 = 1656$, which is significantly higher than the corresponding number of 730 weight-two nodes proposed by Yang. More explicitly, for Yang's design outlined in Equation 3.27, the number of weight-two message nodes is $0.17543 \times 4161 = 730$. The performance of the (4161, 3430) irregular LDPC code using the two different density profiles given in Equation 3.26 and Equation 3.27 is shown in Figure 3.19, where we can see that the error floor effects exhibited by Richardson's density profile were mitigated, when Yang's density profile was employed.

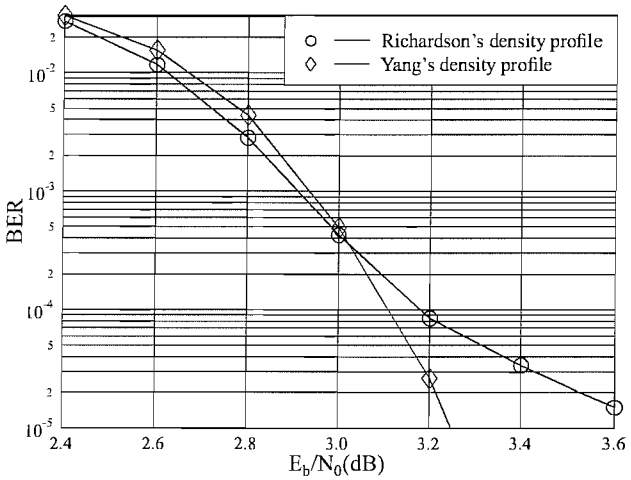


Figure 3.19: BER performance of the irregular (4161,3430) LDPC code in conjunction with two different types of PCM construction, when communicating over an AWGN channel using BPSK modulation. The density profiles for the two types of PCM construction method is specified by Equation 3.26 and Equation 3.27.

3.9 Performance of irregular LDPC codes communicating over AWGN channels

As we have seen in Figure 3.19, Richardson's and Yang's identical-rate identical-block-length irregular LDPC codes transmitted over AWGN channels perform differently. In this section, the performance of irregular LDPC codes will be further studied in various scenarios.

First the effects of various block-lengths are evaluated for transmissions over an AWGN channel. The various irregular LDPC codes were constructed using both Richardson's construction criteria introduced in Section 3.8.1 and Yang's construction method described in Section 3.8.2. The density profiles for the five coding rates used are given in Table 3.1. The density profiles constructed by Yang comply with the constraints given in Section 3.8.2, i.e. the number of weight-two message nodes does not exceed the number of check nodes minus one. Furthermore, by observing Yang's density profiles summarised in Table 3.1, we can see that the existence of weight-three message nodes was avoided during the PCM construction stage for the sake of improving the LDPC codes' distance properties.

The BER performance of these irregular LDPC codes having various block-lengths and code rates may be studied in Figures 3.20 to 3.24. The LDPC codes were constructed using the density profiles summarised in Table 3.1. The attainable coding gain at a BER of 10^{-4} is plotted in Figure 3.25 against the blocklength for the various coding rates considered. As we can see from Figures 3.20 to 3.24 as well as from Figure 3.25, Yang's construction was superior in comparison to Richardson's construction method. Especially at high code rates and higher E_b/N_0 values, a better BER performance was achieved by Yang's construction when compared to the LDPC codes using PCMs constructed by

Coding rate	Density profile
Richardson's density profiles	
1/3	$\lambda_n(x) = 0.54453x + 0.22746x^2 + 0.00341x^4 + 0.15449x^5 + 0.07011x^{19}$
	$\rho_n(x) = 0.82353x^5 + 0.17647x^6$
1/2	$\lambda_n(x) = 0.48603x + 0.27696x^2 + 0.06258x^5 + 0.10026x^6 + 0.07417x^{19}$
	$\rho_n(x) = 0.27273x^7 + 0.72727x^8$
2/3	$\lambda_n(x) = 0.42725x + 0.27089x^2 + 0.16988x^4 + 0.13198x^{19}$
	$\rho_n(x) = 0.51613x^{14} + 0.48387x^{15}$
3/4	$\lambda_n(x) = 0.40056x + 0.29759x^2 + 0.12986x^4 + 0.03807x^5 + 0.13392x^{19}$
	$\rho_n(x) = x^{20}$
4/5	$\lambda_n(x) = 0.38319x + 0.32031x^2 + 0.08342x^4 + 0.07914x^5 + 0.13393x^{19}$
	$\rho_n(x) = 0.50943x^{25} + 0.49058x^{26}$
Yang's density profiles	
1/3	$\lambda_n(x) = 0.54354x + 0.3703x^3 + 0.00148x^{11} + 0.00529x^{12} + 0.07938x^{19}$
	$\rho_n(x) = 0.63636x^5 + 0.36364x^6$
1/2	$\lambda_n(x) = 0.48728x + 0.41431x^3 + 0.00831x^{18} + 0.09011x^{19}$
	$\rho_n(x) = 0.81633x^8 + 0.18367x^9$
2/3	$\lambda_n(x) = 0.33333x + 0.5669x^3 + 0.09977x^{19}$
	$\rho_n(x) = 0.21127x^{13} + 0.78873x^{14}$
3/4	$\lambda_n(x) = 0.25x + 0.65018x^3 + 0.09982x^{19}$
	$\rho_n(x) = 0.61165x^{19} + 0.38835x^{20}$
4/5	$\lambda_n(x) = 0.2x + 0.69877x^3 + 0.10122x^{19}$
	$\rho_n(x) = 0.90335x^{25} + 0.09665x^{26}$

Table 3.1: Density profiles constructed by Richardson and Yang for various LDPC code rates.

Richardson's density profile. However, Yang's method only improves the irregular LDPC code's distance properties when the existence of weight-two message nodes is inevitable, in which case it provides a way of reducing the error floor. However, Yang's method is also incapable of entirely eliminating the error floor, as illustrated in Figures 3.20 to 3.24, when short block-lengths are utilised. In order to mitigate this problem, Yang's design approach may be invoked for avoiding not only the weight-three, but also other relatively low-weight message nodes. By using the following density profiles constructed according to Yang's design criterion, we avoid having both weight-three and weight-four message nodes in Equation 3.28 and in Equation 3.29 we further avoid the weight-five message nodes:

$$\begin{aligned} \lambda_n(x) &= 0.2x + 0.74539x^4 + 0.05462x^{19}, \\ \rho_n(x) &= 0.90335x^{25} + 0.09665x^{26}. \end{aligned} \tag{3.28}$$

$$\begin{aligned} \lambda_n(x) &= 0.19999x + 0.79864x^5 + 0.00137x^{19}, \\ \rho_n(x) &= 0.90335x^{25} + 0.09665x^{26}. \end{aligned} \tag{3.29}$$

By studying the results illustrated in Figure 3.26 and 3.27, it can be seen that when a short blocklength is concerned, the LDPC code's minimum distance is relatively low. Hence, upon avoiding the weight-three and weight-four message nodes, an improved performance is observed in Figure 3.26.

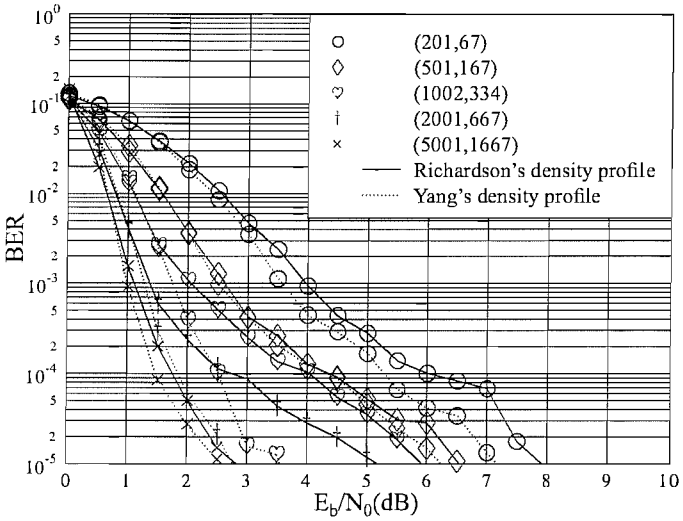


Figure 3.20: BER performance of different-length third-rate irregular LDPC codes constructed using both Richardson's and Yang's density profiles summarised in Table 3.1, when communicating over an AWGN channel using BPSK modulation. The achievable coding gain of the various schemes at a BER of 10^{-4} will be summarised in Figure 3.25 and Table 3.1.

However, when we further avoid the weight-five message nodes, an inferior performance is observed in Figure 3.26. Thus, when the minimum distance of the LDPC code is relatively high, further avoiding the low-weight message nodes results in a less sparse PCM, which consequently results in an inferior performance. This phenomenon may be further clarified as follows. When a relatively high blocklength is concerned, the code's minimum distance becomes higher than that of the identical-rate but lower block-length LDPC codes constructed using the same density profile, when both scenarios only avoids weight-three message nodes, as seen in Table 3.1. Hence, by further avoiding the existence of weight-four message nodes and weight-five message nodes will degrade the achievable BER performance. This is because the density profiles seen in Equations 3.28 and 3.29 are incapable of producing codes approaching the Shannon limit as closely as those summarised in Table 3.1, where less constraints were imposed on the codes during the density profile optimisation.

3.10 Summary and conclusion

Following Luby's approach [60], in this chapter we studied both the design and the achievable performance of irregular LDPC codes in comparison to their regular counterparts. The higher weights associated with the message nodes are capable of providing more useful parity information, which has the potential of assisting the code in converging faster, while the messages are passed from node to node. Richardson's density evolution method [9] outlined in Section 3.4 provides an attractive theoretical approach of predicting the performance of a given density distribution. Chung's Gaus-

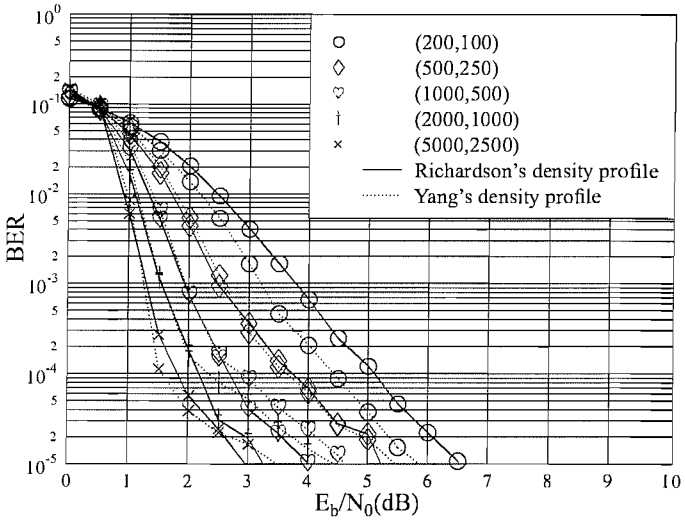


Figure 3.21: BER performance of different-length half-rate irregular LDPC codes constructed using both Richardson's and Yang's density profiles summarised in Table 3.1, when communicating over an AWGN channel using BPSK modulation. The achievable coding gain of the various schemes at a BER of 10^{-4} will be summarised in Figure 3.25 and Table 3.1.

sian approximation approach [11] to density evolution has significantly reduced the complexity of Richardson's original density evolution method at the cost of a slightly reduced accuracy, as it was shown in Section 3.5. However, due to the presence of weight-two columns in the Richardson's density distribution design, the minimum distance of irregular LDPCCs does not increase linearly with the blocklength, which is the case for the family of regular LDPC codes. Thus, when an irregular construction is applied to a short or medium-blocklength LDPC code, in the low SNR region irregular LDPCCs may outperform the family of regular LDPC codes. However, when the SNR increases, the family of irregular LDPC codes will demonstrate a slower convergence and eventually these codes exhibit an error floor.

Yang suggested an approach of reducing the error floor engendered by the existence of weight-two message nodes. This problem may be mitigated by limiting the number of weight-two columns and/or by avoiding some of the lower-weight nodes, such as weight-three or weight-four nodes during the density profile optimisation process. Yang's approach has been shown to be effective for reducing the error floor of short blocklength codes. However, for longer block-lengths, for example for 2000 bits and above, it might not be necessary to avoid the existence of weight-four and weight-five message nodes, provided that a slightly degraded performance may be acceptable. Therefore, we recommend using Richardson's PCM construction approach for those applications which are in-sensitive to delays or which attempt to optimise the performance in the low-SNR region. As for the scenarios when satisfying a certain delay constraint is important or a sharp BER convergence is required, Yang's approach will be the better choice.

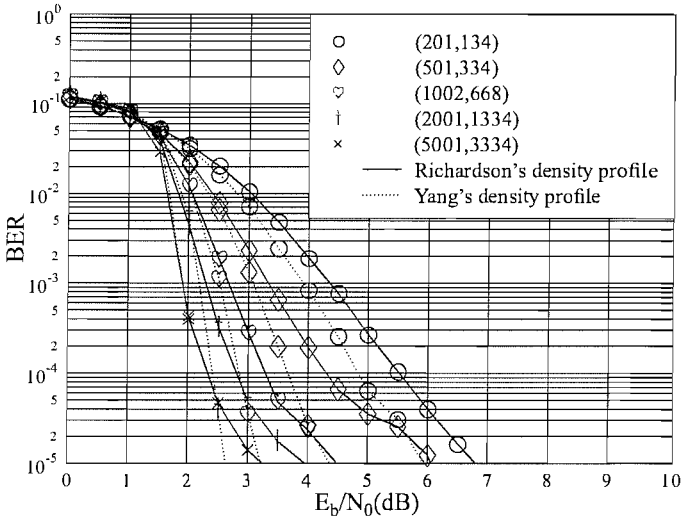


Figure 3.22: BER performance of different-length two-third-rate irregular LDPC codes constructed using both Richardson's and Yang's density profiles summarised in Table 3.1, when communicating over an AWGN channel using BPSK modulation. The achievable coding gain of the various schemes at a BER of 10^{-4} will be summarised in Figure 3.25 and Table 3.1.

A coding gain table is hereby provided for the sake of summarising the achievable BER performance of the irregular LDPC codes discussed in this chapter.

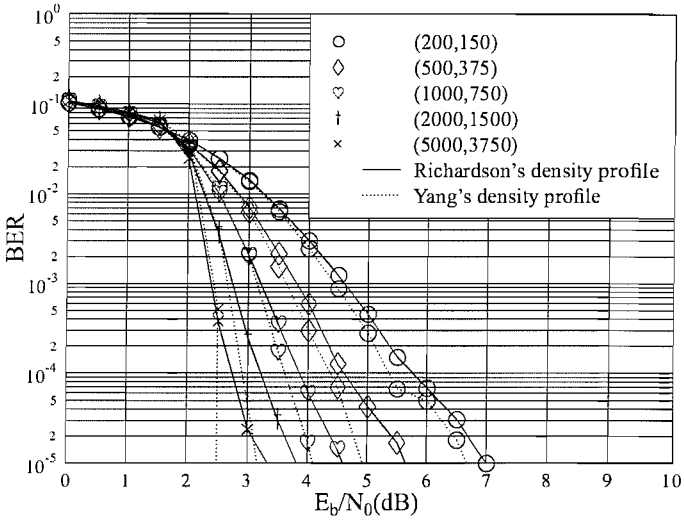


Figure 3.23: BER performance of different-length three-quarter-rate irregular LDPC codes constructed using both Richardson's and Yang's density profiles summarised in Table 3.1, when communicating over an AWGN channel using BPSK modulation. The achievable coding gain of the various schemes at a BER of 10^{-4} will be summarised in Figure 3.25 and Table 3.1.

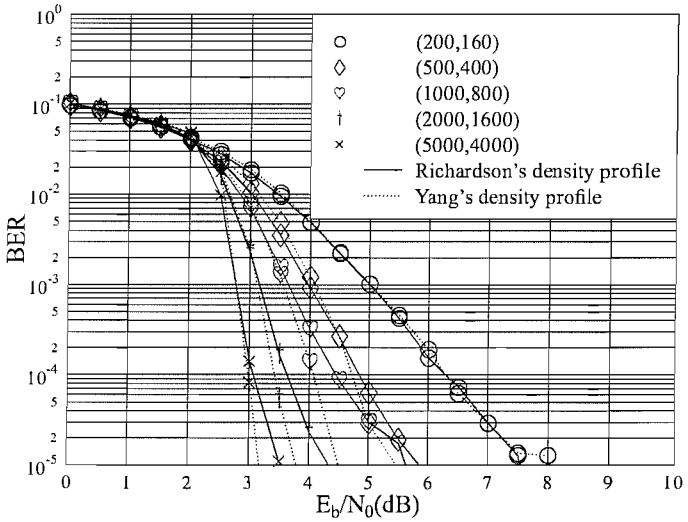


Figure 3.24: BER performance of different-length four-fifth-rate irregular LDPC codes constructed using both Richardson's and Yang's density profiles summarised in Table 3.1, when communicating over an AWGN channel using BPSK modulation. The achievable coding gain of the various schemes at a BER of 10^{-4} will be summarised in Figure 3.25 and Table 3.1.

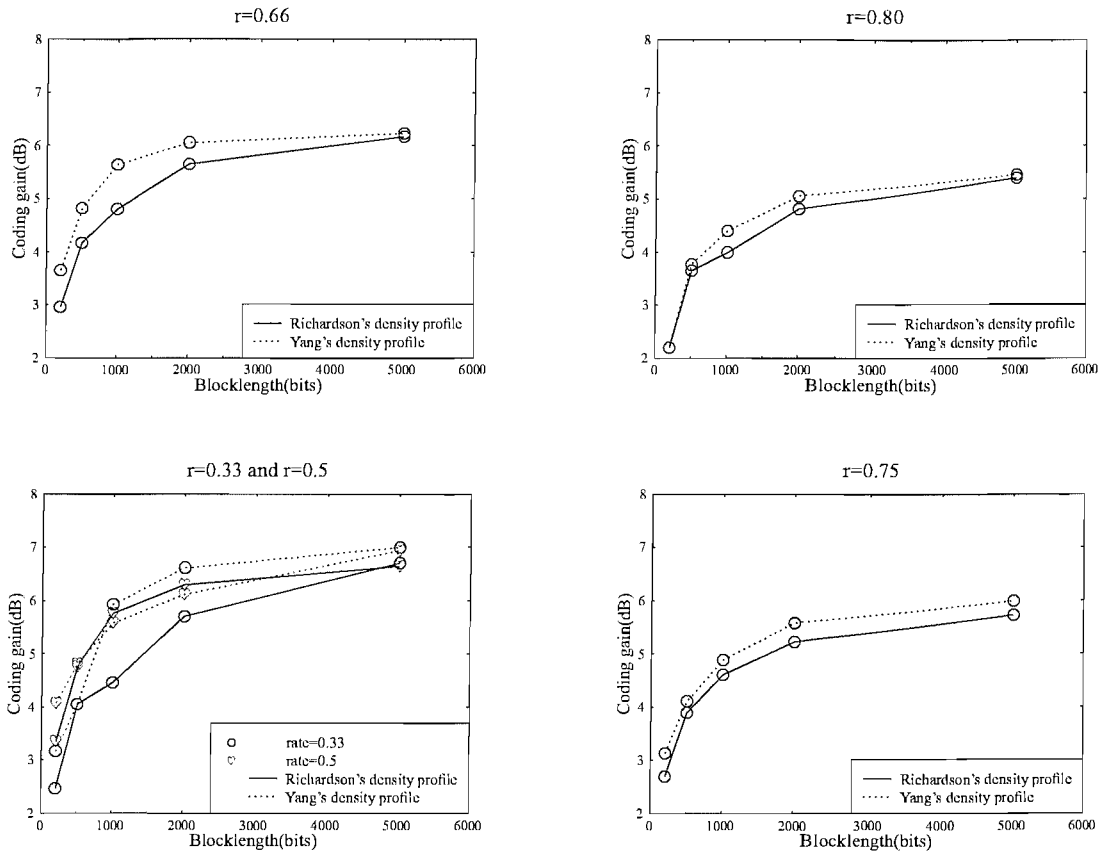


Figure 3.25: Coding gain achieved at a BER of 10^{-4} when irregular LDPC codes were communicating over an AWGN channel using BPSK modulation using different block-lengths at various coding rates. The PCMs are constructed using both Richardson's and Yang's density profile tabulated in Table 3.1.

Code rate	Coding gain (**A**)(dB)	Coding gain (**UR**)(dB)
1/3	6.627	30.75
1/2	6.47	30
2/3	5.97	28.25
3/4	5.53	26.875
5/6	4.78	24.75

Table 3.2: Coding gain achieved by the LDPC code constructed according to Richardson's density profile specified in Equations 3.18 to 3.22 at a BER of 10^{-4}, when communicating over AWGN and uncorrelated Rayleigh fading channels. A coded blocklength of 3000 bits was utilised and the LDPC decoder invokes a maximum of 25 iterations.

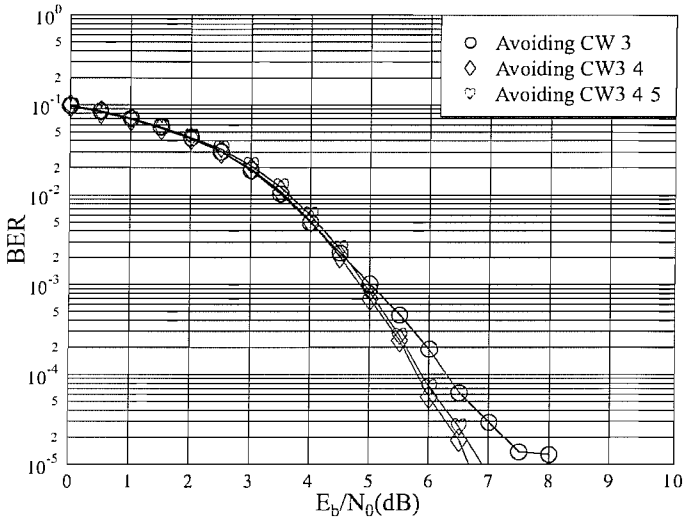


Figure 3.26: BER performance of the four-fifth rate (200,160) LDPC codes constructed by using Yang's density profile in Table 3.1 which avoids the weight-three nodes, and Yang's density profiles in Equation 3.28 and 3.29 which further avoid weight-four and weight-five nodes, respectively.

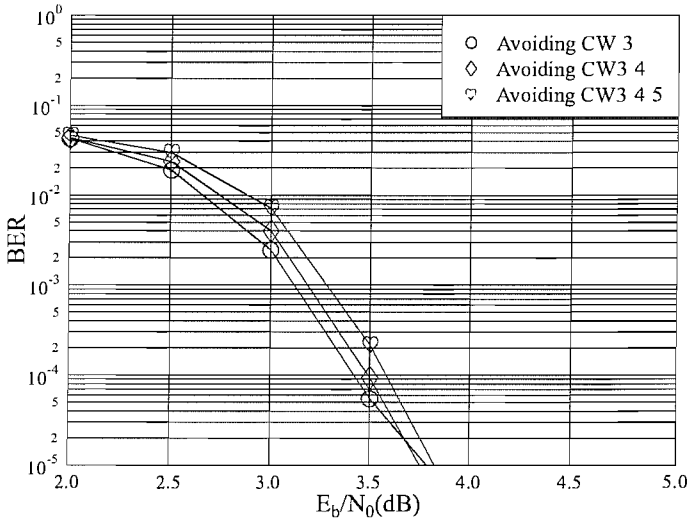


Figure 3.27: BER performance of the four-fifth rate (2000,1600) LDPC codes constructed by using Yang's density profile in Table 3.1 which avoids the weight-three nodes, and Yang's density profiles in Equation 3.28 and 3.29 which further avoid weight-four and weight-five nodes, respectively.

Code rate	Coded blocklength	Coding gain (**A**)(dB) Richardson	Coding gain (**A**)(dB) Yang
1/3	201	2.47	3.176
	501	4.06	4.06
	1002	4.47	5.94
	2001	5.71	6.62
	5001	6.71	7
1/2	200	3.36	4.08
	500	4.77	4.82
	1000	5.77	5.59
	2000	6.3	6.12
	5000	6.65	6.94
2/3	201	2.96	3.65
	501	4.18	4.83
	1002	4.82	5.65
	2001	5.65	6.06
	5001	6.18	6.24
3/4	200	2.70	3.14
	500	3.90	4.12
	1000	4.62	4.89
	2000	5.23	5.59
	5000	5.735	6
4/5	200	2.2	2.2
	500	3.65	3.77
	1000	4	4.41
	2000	4.82	5.06
	5000	5.41	5.47

Table 3.3: Coding gain achieved at a BER of 10^{-4} when irregular LDPC codes were employed for communicating over an AWGN channel using BPSK modulation using different block-lengths at various coding rates. The PCMs are constructed using both Richardson's and Yang's density profile tabulated in Table 3.1.

Chapter 4

Non-Binary LDPC-aided Transmitter and Receiver Diversity Schemes

4.1 State-of-the-art

The *belief propagation* algorithm was devised by Pearl [110], which has been widely used in the area of Artificial Intelligence (AI) [128] [129]. Following the invention of turbo codes by Berrou [130] in 1993, the research community showed sustained interest in contriving iterative decoding algorithms. McEliece [131] introduced Pearl's belief propagation algorithm to the information theory research community and further elucidated why turbo codes are capable of achieving near-capacity performance. McEliece also pointed out that many other error correction coding schemes, such as Gallager's Low Density Parity Check decoding algorithm [1] also constitute instances of Pearl's belief propagation algorithm [131].

In recent years Multiple Input and Multiple Output (MIMO) schemes [115] [132] [133] using multiple antennas have enjoyed a rapid evolution as powerful fading counter-measures in wireless communication, owing to their attractive diversity and/or multiplexing gain. Motivated by these trends, Meshkat and Jafarkhani proposed a bit-by-bit joint detection scheme for an amalgamated LDPC code combined with a Space-Time (ST) code using BPSK modulation [108]. *The novel contribution of this chapter is that the purely binary LDPC-coded space-time scheme of [108] is further developed to a symbol-by-symbol detection scheme, which benefits from a purely symbol-based message exchange with a symbol-based non-binary LDPC code. We will demonstrate that the explicit benefit of this entirely symbol-based scheme is that an improved BER performance is achieved at a reduced detection complexity.* Hence in this chapter, non-binary LDPC codes constructed over a Galois field of size q and proposed by Davey and MacKay [54] are introduced for the sake of developing a purely symbol-based joint LDPC-Space Time (LDPC-ST) detection scheme. The structure of this chapter is as follows.

Section 4.2 gives a brief introduction to both Bayesian networks and to Pearl's belief propagation algorithm. In Section 4.3 Davey's non-binary LDPC code [56] will be introduced. The decoding process for non-binary LDPC codes is outlined with the aid of a worked example and the associated decoding complexity issues are also addressed. Specifically, we will demonstrate that the FFT-based

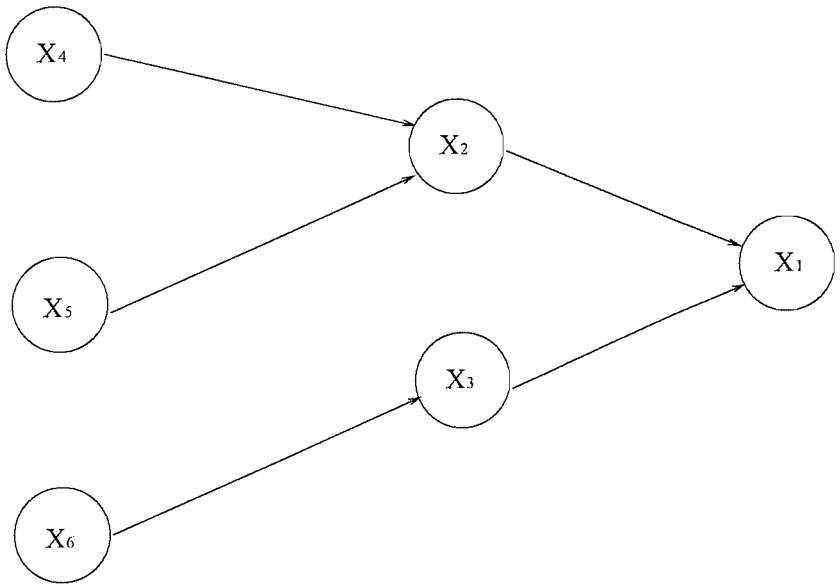


Figure 4.1: Bayesian network having six nodes

LDPC decoding technique of [56] has a low complexity, which is independent of the number of bits per LDPC coded symbol. This advantageous property allows us to decode non-binary LDPC codes operating over a large Galois field at a low complexity. Section 4.4 characterises the performance trends of the family of non-binary LDPC codes employing different configurations. A bit-based joint detection algorithm proposed by Meshkat and Jafarkhani is introduced in Section 4.5 and this algorithm is further developed to a symbol-based scheme in Section 4.6. Section 4.7 and Section 4.8 evaluate the achievable performance gain of the bit-based and symbol-based system, respectively, and in Section 4.3.5 the decoding complexity of the two systems is compared.

4.2 Bayesian networks and Pearl's belief propagation algorithm [110, 131]

Bayesian networks consist of a set of random variables denoted by $X = \{X_1, X_2 ... X_n\}$ and these random variables are represented by the nodes of the network. In Figure 4.1 a Bayesian network having six nodes is portrayed as an example. The nodes are connected by directed links representing the relationship of the so-called *parent nodes* with the so-called *child nodes*. For example, since there is a directed link from X_6 to X_3, this implies that there is message flow from X_6 to X_3. Thus X_6 is termed as the *parent node* of X_3, and hence X_3 is the *child node* of X_6. Similarly, X_4 and X_5 are the *parent nodes* of node X_2. Furthermore, due to the direct link from X_2 to X_1, node X_2 is the *parent node* of X_1. Some of the nodes may correspond to random variables, whose values have been encountered and hence observed. The corresponding nodes are also referred to as *evidence nodes* or *observation nodes*. More explicitly, if nodes X_4, X_5 and X_6 in Figure 4.1 correspond to three channel output samples and their information is processed by a particular algorithm such as for example, an LDPC decoder, to provide parity information for nodes X_2 and X_3 as well indirectly for node X_1, the nodes X_4, X_5 and X_6 are the so-called evidence nodes or observation nodes. When a specific set of

variables corresponding to the evidence nodes is observed, Pearl's belief propagation algorithm may be invoked for inferring the *a posteriori* information corresponding to the rest of the nodes in the network, as was demonstrated in the context of Figure 4.1. For example, the *a posteriori* probability of node X_1 is calculated as follows:

$$P(X_1) = P(X_1|X_2, X_3) \cdot P(X_2|X_4, X_5) \cdot P(X_3|X_6) \cdot P(X_4) \cdot P(X_5) \cdot P(X_6). \quad (4.1)$$

The mechanism of belief propagation which is expressed synonymously as probability propagation becomes explicit in Equation 4.1 and Figure 4.1. If we use the notation $PN(X_i)$ for representing the parent node set of node i, Equation 4.1 can be rewritten as:

$$P(X_1) = \prod_{i=1}^{N} P(x_i|PN(x_i)). \quad (4.2)$$

Recall from Chapter 3 that since an LDPC code's PCM can be represented using a Tanner graph as seen in Figure 3.1, hence Figure 3.1 represents the Baysian network structure of the LDPC code. More explicitly, all the message nodes at the left of Figure 3.1 are the "evidence nodes", since they receive initial intrinsic information from the channel's output. Then the information propagates through the connections from the message nodes seen at the left to the check nodes portrayed on the right in Figure 3.1. During this message passing procedure, the message nodes are the *parent nodes* and the check nodes are the *child nodes*. When a message has to be passed from the right to the left, the check nodes become the *parent nodes* and the variable nodes represent the *child nodes*, because the message nodes receive information from the check nodes. Hence the LDPC decoding procedure also constitutes an instance of Pearl's belief propagation algorithm [131].

4.3 Non-binary LDPC codes

4.3.1 Introduction

In Chapter 2 the family of binary regular-construction LDPCCs has been introduced. The entries in the parity check matrix are either ones or zeroes, hence they are defined over Galois Field (2). Davey and MacKay developed the classic binary LDPCCs further [55], where the parity check matrices are constructed by inserting non-zero entries defined over the Galois Field (q), i.e. over $\mathbf{GF}(q)$ $(q = 2^p)$. More explicitly, the non-zero entries within the parity check matrix of a non-binary LDPCC may assume values from the set spanning from 1 to $(q-1)$. For the sake of distinguishing the PCM $\mathbf{H}$ and the corresponding generator matrix $\mathbf{G}$ defined over the binary field from those defined over the non-binary Galois field $\mathbf{GF}(q)$, we would like to introduce the notation of $\mathbf{H}_q$ and $\mathbf{G}_q$ for representing the non-binary PCM and the non-binary generator matrix, respectively. By contrast, the binary PCM and the binary generator matrix will be denoted as $\mathbf{H}_b$ and $\mathbf{G}_b$, respectively.

Hence the class of non-binary LDPCCs constitutes a generalisation of the binary LDPC codes introduced in Chapter 2 and this generalisation will be detailed here with reference to the work of Davey and MacKay [54, 56]. In the family of non-binary LDPCCs, the non-zero entries of the parity check matrix are inserted in the same way as in the binary case. However, in addition to deciding where to insert the non-zero entries, the value of the non-zero entries has to be determined over $\mathbf{GF}(q)$.

Furthermore, as in the case of binary codes, a legitimate non-binary codeword also has to satisfy the constraint that its product with the parity check matrix results in an all-zero vector. In the context of non-binary LDPCC decoding over $\mathbf{GF}(q)$, the additions and multiplications involved in computing the product of the codeword and the PCM should be carried out over the corresponding finite field, rather than over the binary field as in Chapter 2.

For the non-binary LDPC defined over $\mathbf{GF}(q)$, each non-zero entry of both $\mathbf{H}_q$ and $\mathbf{G}_q$ can be associated with a $(p \times p)$-dimensional submatrix, where we have $p = \log_2(q)$ [56]. More explicitly, since the non-zero entries of the $\mathbf{H}_q$ and $\mathbf{G}_q$ matrix may assume values from 1 to $(q-1)$ for a non-binary LDPCC defined over $\mathbf{GF}(q)$, there should be a total of $(q-1)$ different submatrices associated with each individual legitimate value of the non-zero entry in the matrices $\mathbf{H}_q$ and $\mathbf{G}_q$. Furthermore, each coded non-binary symbol defined over $\mathbf{GF}(q)$ can be represented in a binary format using p binary bits. If we replace all the non-zero entries in $\mathbf{H}_q$ and $\mathbf{G}_q$ with their individually associated binary-valued submatrix, the equivalent binary PCM $\mathbf{H}_2$ and generator matrix $\mathbf{G}_2$ can be derived, which correspond to $\mathbf{H}_q$ and $\mathbf{G}_q$. Assume for example that we have a $q-ary$ source symbol b defined over $\mathbf{GF}(q)$, and a is a non-zero entry in the parity check matrix also defined over $\mathbf{GF}(q)$. Then the multiplication of symbol b by a is equivalent to the matrix multiplication of the binary representation of the source symbol b and the submatrix associated with the non-zero entry a of the parity check matrix. Hence, if each non-zero entry in the $q-ary$ matrix $\mathbf{G}_q$ and $\mathbf{H}_q$ defined for non-binary LDPCCs is replaced with their corresponding binary submatrix, we will have the resultant binary generator matrix and parity check matrix $\mathbf{G}_b$ and $\mathbf{H}_b$ in a binary form, while the sizes of the matrices are increased by a factor of p both vertically and horizontally. Suppose we have a source symbol sequence $\mathbf{S}_q$ over $\mathbf{GF}(q)$ and the corresponding binary representation of $\mathbf{S}_q$ can be denoted as $\mathbf{S}_b$. Then, upon multiplying $\mathbf{S}_q$ with $\mathbf{G}_q$ using finite field arithmetics, we arrive at the encoded symbol sequence $\mathbf{C}_q$. Similarly, the binary representation of a codeword $\mathbf{C}_b$ may be obtained by multiplying $\mathbf{S}_b$ and $\mathbf{G}_b$ over the binary field. As expected, the sequence $\mathbf{C}_b$ is the binary representation of the symbol sequence $\mathbf{C}_q$ [56]. However, the advantages and disadvantages of using binary versus higher order decoding fields will be highlighted at a later stage.

The matrices $\mathbf{H}_b$ and $\mathbf{H}_q$ can be used for producing $\mathbf{G}_b$ and $\mathbf{G}_q$ which in turn may be involved for generating the same codeword in either bit or symbol format. Clearly, the non-binary LDPCC decoder is a symbol based decoder. Each column of $\mathbf{H}_q$ represents a symbol message node constituted by p bits. If the input to the decoder provides only bit probabilities, the decoder has to interpret p bits output by the demodulator as a single $q-ary$ symbol and approximate the symbol probability by using the product of the probability values of each constituent bit, assuming that all the bits are independent. This independence is only an approximately valid assumption, for example in the case of using a gray-coded modem constellation, where the gray-mapping imposes constraints on the transmitted bits. Following this philosophy, the symbol probability is calculated from the demodulator's soft output values as:

$$P^a = \prod_{j=1}^{p} P_{x_j}^{a_j}, \tag{4.3}$$

where $P_{x_j}^{a_j}$ represents the probability that the j^{th} constituent bit x_j is equal to a_j, while $(a_1, ...a_p)$ is the binary representation of symbol a.

4.3.2 Advantages and disadvantages of non-binary LDPC codes

MacKay pointed out in [8] that LDPCCs should have a high column weight for the sake of achieving a high performance. As illustrated in Figures 3.2 to 3.4, the high-weight message nodes of an irregular LDPC code converge to their correct states at a higher rate than the low-weight nodes. This intuition, which led to Luby's [60] irregular construction, also implies that when having more non-zero entries in a column, a message nodes becomes more confident in deciding which state it should be in. However, the resultant higher number of non-zero entries in a row may lead to confusion for the check nodes, because now an increased number of non-zero entries participate in the parity check equation, thus an increased number of configurations are available to satisfy the check equation. Furthermore, when a moderate PCM size is applied, using a higher column weight will render the parity check matrix less sparse and hence cycles having a short length will be introduced. As seen from Figure 4.2, by extending $\mathbf{GF}(2)$ to $\mathbf{GF}(q)$, the binary column weight of the equivalent binary parity check matrix portrayed at the right of the figure is increased. Furthermore, the row weight of the equivalent binary PCM is also increased.

Since we argued previously that $\mathbf{H}_q$ and $\mathbf{G}_q$ have their corresponding binary equivalent matrices $\mathbf{H}_b$ and $\mathbf{G}_b$, hence the family of non-binary LDPCCs is subjected to the same design dilemma as binary irregular LDPCCs. More explicitly, using a large non-binary decoding field will result in a high equivalent binary column weight for the PCM, but the equivalent binary row weight is also increased. However, the merit of non-binary LDPCCs arises from their reduced probability of forming short cycles.

By observing the cycles using the bipartite representation of Figure 4.2, it can be seen that using non-binary LDPCCs results in no cycles, while the equivalent binary PCM has many cycles, among which two *length-4* cycles are highlighted using dotted lines in Figure 4.2. From this example we may infer that non-binary LDPCCs are less likely to incur short cycles. The binary equivalent PCMs are constructed upon replacing the non-zero entries defined over $\mathbf{GF}(8)$ by their corresponding binary companion matrices. Details of finding the binary companion matrix can be found in Appendix C. Since each binary submatrix is of dimension 3-by-3, we have three times more message nodes as well as the check nodes in comparison to the non-binary PCM seen at on the left of Figure 4.2. The Tanner graph of the binary equivalent matrix is constructed in a similar fashion as the non-binary PCM, i.e. by connecting the message nodes and the check nodes according to the position of the non-zero entries in the PCM. Thus, in the message passing based decoding process the non-zero PCM entries are capable of receiving information from more neighbouring nodes, which is expected to improve the achievable decoding performance. However, the disadvantage of non-binary LDPCCs arises from having an increased number of possible values for the non-zero entries in the PCM. Since the number of possible states is increased by a factor of q, the non-zero entries in the row may assume a higher number of possible values rendering them more difficult to classify. The associated decoding complexity is also increased owing to the increased number of possible states. Davey has demonstrated in [56] that non-binary LDPCCs are not always superior to their binary counterparts, as a consequence of the above-mentioned advantages and disadvantages. However, the non-binary LDPCCs lead themselves to purely symbol-based decoding, which will be shown to be advantageous in the context of the non-binary LDPCC-aided space-time code of Section 4.6.

In our further discussion, we will express the coded blocklength of a non-binary LDPCC associated

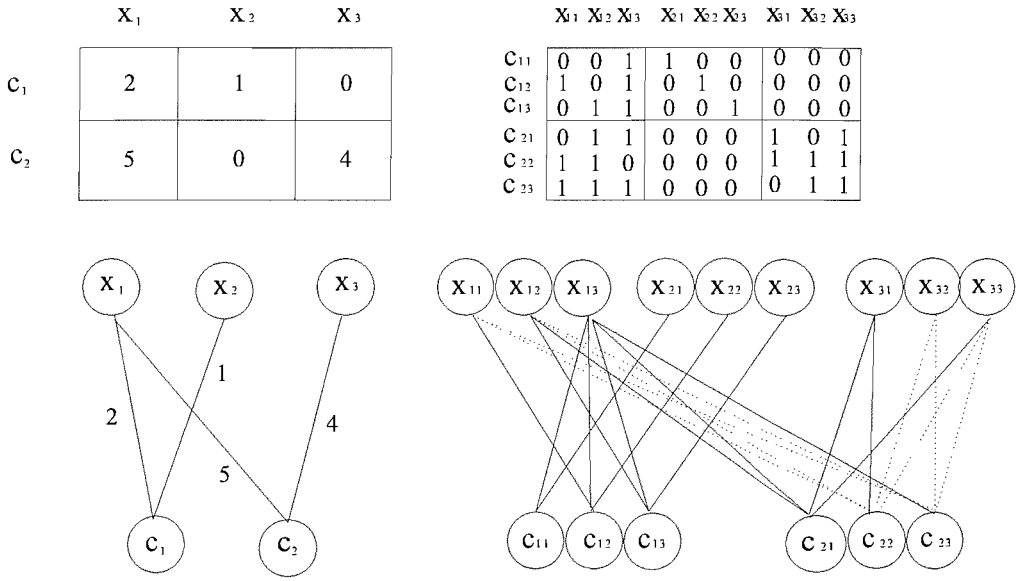


Figure 4.2: Bipartite graph representation of two PCMs constructed over **GF**(8) and **GF**(2)

with $\mathbf{H}_q$ in terms of the number of symbols, while the blocklength of the equivalent binary representation of the codeword will be quoted in bits. For example, the PCM of Figure 4.2 is related to a non-binary LDPC code having a length of three *non-binary symbols*. At the same time, we may refer to this non-binary LDPC code as having a coded blocklength of 9 bits, since it was constructed over **GF**(8). Furthermore, we could also use the number of *non-binary symbols* and *bits* for representing the size of the PCMs $\mathbf{H}_q$ and $\mathbf{H}_b$, respectively. All LDPC codes represented by $(N_q, K_q)_q$ refer to a code having K_q *non-binary information symbols*, while the encoded blocklength is N_q *non-binary symbols*.

4.3.3 Decoding process

The decoding process is quite similar to that outlined in Chapter 2 for the binary LDPCC scenario, apart from the fact that all the calculations have to be carried out in the corresponding non-binary Galois field. However, since the Galois field order has been increased from binary to **GF**(q), the complexity of updating R using Equation 2.31 is increased by a factor of q^2. Fortunately, the reduced complexity decoding method of Richardson and Urbanke outlined in Section 2.7.2 can be further generalised for non-binary LDPCCs.

Algorithm 2 *The procedure of decoding a non-binary LDPCC is outlined here as follows:*

Step 1: Initialisation

During the initialisation step, the quantities $Q_{i,j}^a, i = 1 \ldots M_q, j = 1 \ldots N_q$ of each non-zero entry in the PCM $\mathbf{H}_q$ have to be initialised to the specific value, which is provided by the channel's soft output, i.e. to the intrinsic probability P_j^a in Equation 2.32, where a and j represent the symbol state and the symbol index in the codeword, respectively.

Step 2: Horizontal update

The horizontal update step is dedicated to the updating process of the quantity $R^a_{i,j}$ using the quantity $Q^a_{i,j}$. A legitimate codeword has to satisfy all parity check equations in the PCM. When updating $R^a_{i,j}$ using Equation 2.31, the participating message nodes have to satisfy the i^{th} parity check formulated as:

$$\sum_{j' \in C_i} c_{i,j'} \cdot H_{i,j'} = 0, \tag{4.4}$$

where $c_{i,j}$ represents the j^{th} symbol, which participates in the i^{th} parity check. More explicitly, Equation 4.4 requires that the product of the message nodes participating in the i^{th} check, whose column indices are the elements of the vector C_i and the corresponding non-zero entries of the i^{th} row of the PCM $\mathbf{H}_{i,j'}$ evaluated over $\mathbf{GF}(q)$ becomes zero.

In the non-binary scenario, all message nodes have q possible values. Since we have to evaluate the probabilities for each legitimate symbol state, we can use the probability of each message node in the decoding process which we quantify with the aid of the corresponding discrete PDF. According to the Galois Field addition rule [134], the sum of two identical elements of $\mathbf{GF}(q)$ is zero, thus Equation 4.4 can be rewritten as:

$$\sum_{j' \in C_i, j' \neq k} c_{i,j'} \cdot H_{i,j'} = c_{i,k} \cdot H_{i,k}, k \in C_i. \tag{4.5}$$

It becomes clear from Equation 4.5 that the variable $c_{i,k} \cdot H_{i,k}, k \in C_i$ can be expressed as the sum of other variables $c_{i,j'} \cdot H_{i,j'}, j' \in C_i, j' \neq k$.

As mentioned in Section 2.7.2, the PDF of the quantity $R^a_{i,j}$ of Equation 2.34 can be obtained by evaluating the convolution of the PDFs of the quantities $Q^a_{i,j'}$ of the other message nodes participating in the i^{th} check.

An example has been provided in Section 2.7.2 for the binary case, and a detailed worked example will be supplied later in Section 4.3.4 for the non-binary scenario. Let us now introduce a variable $v_{i,j}$, which represents the product of the symbol value $a \in (0, q-1)$ of the j^{th} message node and the matrix entry $H_{i,j}$ at the i^{th} row and j^{th} column obeying $v_{i,j} = a \cdot H_{i,j}$. When non-binary LDPC codes are concerned, the PDF of the product $\check{R}^v_{i,j}$ can be obtained by calculating the convolution of the PDFs of the quantities $\check{Q}^v_{i,j'}$ of the other message nodes participating in the i^{th} check. The notations $\check{Q}^v_{i,j'}$ and $\check{R}^v_{i,j'}$ are hereby also introduced for the sake of distinguishing them from $Q^a_{i,j'}$ and $R^a_{i,j'}$. The reason that we used $R^a_{i,j}$ and $Q^a_{i,j}$ directly in Section 2.7.2, rather than $R^v_{i,j}$ and $Q^v_{i,j}$ is because we had $H_{i,j} = 1$ for all i and j, when binary LDPCCs are concerned. Therefore we have $v = a \cdot H_{i,j} = a$. Thus we arrive at:

$$PDF\{\check{R}^v_{i,j}\} = \bigotimes_{j' \in \{C_i\}, j' \neq j} PDF\{\check{Q}^v_{i,j'}\}, \tag{4.6}$$

where $\bigotimes$ indicates convolution.

Step 2.1: Permutation of the PDF entries of the quantity $Q_{i,j'}$

As seen from Equation 4.6, the PDF of the variable $\check{Q}^v_{i,j'}$ is used to update $\check{R}^v_{i,j'}$. The PDF of the quantity $\check{Q}^v_{i,j'}$ can be obtained upon permuting the PDF of variable $Q^a_{i,j'}$. Considering $\mathbf{GF}(4)$ as an example and assuming that the PDF of $Q^a_{i,j'}$ for a message node is $Q^a_{i,j'} = (Q^0_{i,j'}, Q^1_{i,j'}, Q^2_{i,j'}, Q^3_{i,j'})$

and assuming furthermore that the corresponding PCM entry is $H_{i,j'} = 2$ over **GF**(4), the product $((0 \cdot H_{i,j'}), (1 \cdot H_{i,j'}), (2 \cdot H_{i,j'}), (3 \cdot H_{i,j'}))$ yields $(0, 2, 3, 1)$, since the sequence $a = (0, 1, 2, 3)$ is multiplied with $H_{i,j'} = 2$ over **GF**(4). An arithmetic operation table over **GF**(4) is provided in Table 4.1 for the reader's convenience. Thus, by reordering the original PDF $Q^a_{i,j} = (Q^0_{i,j}, Q^1_{i,j}, Q^2_{i,j}, Q^3_{i,j})$ of the message nodes $Q^a_{i,j'}$, $i = 1 \ldots M_q$, $j = 1 \ldots N_q$ according to the sequence $(0, 2, 3, 1)$ yields $(Q^0_{i,j'}, Q^3_{i,j'}, Q^1_{i,j'}, Q^2_{i,j'})$. More explicitly, since according to Table 4.1 we have $2 \times 2 = 3$ over **GF**(4), thus $2 \cdot H_{i,j'}$ will make a contribution to the i^{th} parity check of the value of 3 over **GF**(8), with a probability of $Q^2_{i,j'}$. More explicitly, we have $v = 2 \times 2 = 3$ over **GF**(8), hence $\check{Q}^3_{i,j'} = Q^2_{i,j'}$. Thus the probability of $Q^2_{i,j'}$ should be placed at position three in the PDF of the variable $\check{Q}^v_{i,j'}$ and hence the PDF of $\check{Q}^v_{i,j'}$ becomes $(Q^0_{i,j'}, Q^3_{i,j'}, Q^1_{i,j'}, Q^2_{i,j'})$, which is indeed a permuted version of $Q^a_{i,j'} = (Q^0_{i,j'}, Q^1_{i,j'}, Q^2_{i,j'}, Q^3_{i,j'})$.

+	0	1	2	3
0	0	1	2	3
1	1	0	3	2
2	2	3	0	1
3	3	2	1	0

•	0	1	2	3
0	0	0	0	0
1	0	1	2	3
2	0	2	3	1
3	0	3	1	2

Table 4.1: GF(4) addition and multiplication table

As mentioned in Section 2.7.2 and defined by Equation 4.6, upon convolving the PDFs of all the variables $\check{Q}^v_{i,j'}$, except for that of the message node at the j^{th} column, as shown in Equation 4.6, the result specifies the PDF of $\check{R}^v_{i,j}$. The elements of the PDF vector have to be permuted again, in the opposite sense in comparison to the previous permutation, for the sake of generating the PDF of the quantity $\check{R}^v_{i,j}$. This process will be explained in more detail with the aid of a worked example in Section 4.3.4.

Step 2.2: FFT

As previously suggested in the context of Equation 2.35 of Section 2.7.2, the complex convolution can be more efficiently implemented in the frequency domain with the aid of the FFT [9]. Equation 2.37 has provided the corresponding FFT formula for the binary case. In a non-binary LDPCC scenario the FFT has to be implemented over a non-binary finite field, during the update of the quantity $R^a_{i,j}$ of Equation 2.31 [56].

As suggested in Equation 2.37 of Chapter 2, the FFT of the function f defined over **GF**(2) is given by $\mathcal{F}(f)(0) = f(0) + f(1)$, $\mathcal{F}(f)(1) = f`(0) - f(1)$. The notations $\mathcal{F}(f)(a)$ and $f(a)$ have the similar definitions as in Equation 2.37, while here they are generalised for a decoding field of size q, rather than binary.

Since the PDF of each message node is defined over **GF**(q), thus the FFT is not a one-dimensional q-point FFT, but a p-dimensional two-point FFT, where we have $2^p = q$. When the FFT is represented in a matrix format, the matrix is the same as the Walsh-Hadamard (WH) matrix [38]. By representing the FFT to be carried out over **GF**(2) in Equation 2.37 in a matrix format, we arrive at:

$$\begin{pmatrix} \mathcal{F}(f)(0) \\ \mathcal{F}(f)(1) \end{pmatrix} = \begin{pmatrix} f(0) \\ f(1) \end{pmatrix} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}. \tag{4.7}$$

The (2×2) matrix in Equation 4.7 is termed the *FFT matrix* (FFTM) here, and it may be used as a component during the construction of other FFTMs for FFTs to be carried out over larger Galois fields. The creation of a FFTM defined over a Galois field of size 2^p is a matrix manipulation process. When the size of the FFTM is doubled, i.e. when we use the FFTM valid for **GF**(2) for the sake of constructing the FFTM generated for **GF**(4), we can create the FFTM valid for **GF**(4) from four partitions, each having the same size as the FFTM created for **GF**(2). Three replicas of the original FFTM will be placed in the upper left, upper right and lower left partition, respectively, while the original copy will be positioned at the lower right corner of the new FFT matrix, where all elements in this and only this (2×2) component FFTM will have their polarity inverted according to the recursive WH matrix generation rules. A FFTM constructed over **GF**(4) and obtained by using the matrix seen in Equation 4.7 is as follows:

$$\begin{pmatrix} \mathcal{F}(f)(0) \\ \mathcal{F}(f)(1) \\ \mathcal{F}(f)(2) \\ \mathcal{F}(f)(3) \end{pmatrix} = \begin{pmatrix} f(0) \\ f(1) \\ f(2) \\ f(3) \end{pmatrix} \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \end{pmatrix}. \tag{4.8}$$

Furthermore, by using the FFTM obtained in Equation 4.8, the FFTM created over **GF**(8) may be developed, as follows:

$$\begin{pmatrix} \mathcal{F}(f)(0) \\ \mathcal{F}(f)(1) \\ \mathcal{F}(f)(2) \\ \mathcal{F}(f)(3) \\ \mathcal{F}(f)(4) \\ \mathcal{F}(f)(5) \\ \mathcal{F}(f)(6) \\ \mathcal{F}(f)(7) \end{pmatrix} = \begin{pmatrix} f(0) \\ f(1) \\ f(2) \\ f(3) \\ f(4) \\ f(5) \\ f(6) \\ f(7) \end{pmatrix} \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 & 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 & 1 & -1 & -1 & 1 \\ 1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 \\ 1 & -1 & 1 & -1 & -1 & 1 & -1 & 1 \\ 1 & 1 & -1 & -1 & -1 & -1 & 1 & 1 \\ 1 & -1 & -1 & 1 & -1 & 1 & 1 & -1 \end{pmatrix}. \tag{4.9}$$

Let us use $(\hat{Q}^0_{i,j'}, ..., \hat{Q}^{q-1}_{i,j'})$ for representing the FFT of the permuted PDF, which specifies the PDF of $\check{Q}^v_{i,j'}$, namely that of $\left(\check{Q}^0_{i,j'}, ..., \check{Q}^{q-1}_{i,j'}\right)$.

Step 2.3: Updating the PDF of the variable $\check{R}^v_{i,j}$

Since now we carry out the convolution in the frequency domain after the FFT, the convolution process can be implemented with the aid of multiplications, as follows [56]:

$$(\hat{R}^0_{i,j}, ..., \hat{R}^{q-1}_{i,j}) = \Big(\prod_{j' \in C(i) \setminus j} \hat{Q}^0_{i,j'}, \prod_{j' \in C(i) \setminus j} \hat{Q}^1_{i,j'}, ..., \prod_{j' \in C(i) \setminus j} \hat{Q}^{q-1}_{i,j'} \Big), \tag{4.10}$$

where $(\hat{R}^0_{i,j}, ..., \hat{R}^{q-1}_{i,j})$ represents the FFT of the PDF for the variable $R^a_{i,j} \cdot H_{i,j}$, i.e. $(\check{R}^0_{i,j}, ..., \check{R}^{q-1}_{i,j})$. More explicitly, Equation 4.10 represents the FFT of the PDF for all the products $\check{Q}^v_{i,j'}, i = 1 \ldots M_q, j' = 1 \ldots N_q$ participating in the i^{th} row, except for the j^{th} message node.

Step 2.4 IFFT

By carrying out an inverse FFT, this vector is transformed back to the time domain. The IFFT is carried out by following the procedure described in Chapter 2. More explicitly, the IFFT is carried

out over $\mathbf{GF}(q)$ by multiplying the corresponding FFTM, i.e. the same as during the FFT process, except that the resultant probability sequence has to be normalised for the sake of ensuring that all probabilities sum up to unity.

Step 2.5 Backward permutation

After the IFFT has been completed and all the probabilities have been normalised, the resultant PDF defines the probability distribution of the variable $\check{R}^{v}_{i,j}$. For the sake of retrieving the PDF of the variable $R^{a}_{i,j}$, the elements of the resultant PDF vector obtained during Step 2.4 are permuted for the sake of eliminating the effect of the matrix entry $H_{i,j}$. This operation can be carried out in the direction opposite to the way we permuted the PDF of the variable $Q^{a}_{i,j'}$ in Step 2.1.

Step 3: Vertical update

Once Step 2 has been accomplished, the value of $R^{a}_{i,j}$ may now be used for vertically updating the message $Q^{a}_{i,j}$ in Equation 2.32. The *a posteriori probability* of each individual coded symbol can be evaluated by using Equation 2.33.

Step 4: Hard decision and parity check using $\mathbf{H}_q$

Using the *a posteriori probability* obtained from Step 3, the symbol having the highest probability will be chosen as the survivor during the hard decision phase. The non-binary symbol sequence obtained from the hard decision will be multiplied with the PCM $\mathbf{H}_q$ for verification using the non-binary arithmetics operating over $\mathbf{GF}(q)$. If the resultant vector is an all-zero vector, then a legitimate codeword has been found and the result will be output by the decoder. If the resultant vector is not an all-zero vector, provided that the maximum number of iterations has not been reached, the decoding process will be continued by invoking Step 2 for the next iteration.

4.3.4 Non-binary LDPC decoding example

In this section, the iterative decoding of LDPC codes generated over GF(q) will be described in detail using a worked example. The example will illustrate the decoding process of a half-rate LDPC code constructed over $\mathbf{GF}(4)$, when communicating over an AWGN channel. A randomly generated parity check matrix $\mathbf{H}_q$ is listed in Table 4.2, and the corresponding generator matrix $\mathbf{G}_q$ given in Table 4.3 can be obtained in the same way as described in Section 2.4, except that all the arithmetic operations are carried out over $\mathbf{GF}(4)$. The generator matrix $\mathbf{G}_q$ will produce a codeword with the original systematic information symbols concatenated at the end of the codeword. The arithmetic operations carried out over $\mathbf{GF}(4)$ are previously summarised in Table 4.1 for the reader's convenience.

Assuming that a sequence of five source symbols given by {2, 0, 1, 2, 2} has been encoded upon multiplying them with $\mathbf{G}_q$ given in Table 4.3 as follows:

$$\mathbf{C}_{q\ (1\times N_q)} = \mathbf{S}_{q\ (1\times K_q)} \cdot \mathbf{G}_{q\ (K_q\times N_q)}$$

$$[2\ 3\ 1\ 2\ 3\ 2\ 0\ 1\ 2\ 2] = [2\ 0\ 1\ 2\ 2] \cdot \begin{bmatrix} 1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 2 & 1 & 2 & 1 & 3 & 0 & 1 & 0 & 0 & 0 \\ 2 & 2 & 3 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 1 & 1 & 1 & 1 & 2 & 0 & 0 & 0 & 1 & 0 \\ 0 & 3 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

$\mathbf{H}_q =$

	1	2	3	4	5	6	7	8	9	10
1	0	0	3	0	0	0	1	2	3	0
2	1	1	0	0	0	0	3	0	0	3
3	0	0	0	2	3	0	0	0	3	0
4	2	0	1	0	2	2	0	0	0	0
5	0	2	0	2	0	2	0	3	0	1

Table 4.2: Example of a low density parity check matrix (PCM) $\mathbf{H}_q$ for $N_q = 10$, $M_q = N_q - K_q = 5$ and $w_c = 2$ constructed over $\mathbf{GF}(4)$

$\mathbf{G}_q =$

	1	2	3	4	5	6	7	8	9	10
1	1	1	0	0	0	1	0	0	0	0
2	2	1	2	1	3	0	1	0	0	0
3	2	2	3	0	0	0	0	1	0	0
4	1	1	1	1	2	0	0	0	1	0
5	0	3	0	0	0	0	0	0	0	1

Table 4.3: Generator matrix $\mathbf{G}_q$ for the PCM $\mathbf{H}_q$ of Table 4.2 specifying the non-binary half-rate code LDPC $(10, 5)_q$ over $\mathbf{GF}(4)$.

The multiplications are carried out in $\mathbf{GF}(4)$. For example, the second parity symbol, i.e. 3, is calculated by $2 \times 1 + 0 \times 1 + 1 \times 2 + 2 \times 1 + 2 \times 3 = 2 + 0 + 2 + 2 + 1 = 3$.

Upon multiplying this codeword by the transpose of the parity check matrix $\mathbf{H}_q$ given in Table 4.2 over $\mathbf{GF}(4)$ with the aid of Table 4.1, we arrive at:

$$\begin{bmatrix} 2 & 3 & 1 & 2 & 3 & 2 & 0 & 1 & 2 & 2 \end{bmatrix} \cdot \begin{bmatrix} 0 & 1 & 0 & 2 & 0 \\ 0 & 1 & 0 & 0 & 2 \\ 3 & 0 & 0 & 1 & 0 \\ 0 & 0 & 2 & 0 & 2 \\ 0 & 0 & 3 & 2 & 0 \\ 0 & 0 & 0 & 2 & 2 \\ 1 & 3 & 0 & 0 & 0 \\ 2 & 0 & 0 & 0 & 3 \\ 3 & 0 & 3 & 0 & 0 \\ 0 & 3 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \end{bmatrix},$$

the resultant all-zero vector indicates that the codeword is legitimate.

The encoded codeword is mapped onto bits for BPSK transmission over the AWGN channel. The probability of each symbol assuming any of the four possible values is calculated according to Equation 4.3 using the bit-based soft channel outputs, where our inherent assumption is that the bits of a symbol are independent, although in case of a gray-coded modulation constellation this is only approximately true. The resultant symbol probabilities are listed in Table 4.4. The far left column in Table 4.4 is the symbol index, while the top row specifies all the possible symbols. Finally, columns 6 and 7 of Table 4.4 represent the surviving symbol and whether the hard decision is correct. The

symbol value having the highest probability is chosen as the survivor in column 6 of Table 4.4, and the decoded codeword constituted by the surviving symbol is checked with the aid of the PCM to ascertain whether the decoded codeword is a legitimate one.

	0	1	2	3	Survivor	Result
1	0.0333887	0.000155706	0.96197	0.00448607	2	Correct
2	0.00157199	0.0076656	0.168601	0.822161	3	Correct
3	0.968776	0.0210919	0.00991634	0.000215896	2	Error
4	0.0647001	0.000175584	0.932593	0.00253088	2	Correct
5	1.0302×10^{-5}	0.0595526	0.000162658	0.940274	3	Correct
6	0.825772	0.00632498	0.166626	0.00127627	0	Error
7	0.582222	0.0116672	0.398132	0.00797822	0	Correct
8	0.000738692	0.997418	1.36396×10^{-6}	0.00184169	1	Correct
9	0.00926008	5.02575×10^{-7}	0.990686	5.37678×10^{-5}	2	Correct
10	0.00487535	3.97331×10^{-5}	0.987041	0.00804418	2	Correct

Table 4.4: Intrinsic symbol probabilities calculated using the channel's soft output for the LDPCC(10,5) generated over GF(4) using the PCM of Table 4.2.

The surviving or most likely transmitted symbols are {2, 3, 0, 2, 3, 0, 0, 1, 2, 2}, as seen in the 6^{th} column of Table 4.4. The third and sixth symbols are erroneous, thus the product of the corresponding codeword sequence with the transpose of the PCM $\mathbf{H}_q$ seen in Table 4.2 is calculated as:

$$\begin{bmatrix} 2 & 3 & 0 & 2 & 3 & 0 & 0 & 1 & 2 & 2 \end{bmatrix} \cdot \begin{bmatrix} 0 & 1 & 0 & 2 & 0 \\ 0 & 1 & 0 & 0 & 2 \\ 3 & 0 & 0 & 1 & 0 \\ 0 & 0 & 2 & 0 & 2 \\ 0 & 0 & 3 & 2 & 0 \\ 0 & 0 & 0 & 2 & 2 \\ 1 & 3 & 0 & 0 & 0 \\ 2 & 0 & 0 & 0 & 3 \\ 3 & 0 & 3 & 0 & 0 \\ 0 & 3 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 3 & 0 & 0 & 2 & 3 \end{bmatrix},$$

which is a non-zero vector. Thus we have to follow the decoding steps described in Section 4.3.3 for the sake of eliminating the errors.

First all the non-zero entries of $\mathbf{H}_q$ seen in Table 4.2 will have their associated $Q_{i,j}^a$ values initialised to the intrinsic probabilities of Table 4.4, which are denoted as P_j^a in step 1 of the decoding process outlined in Section 4.3.3. Then the resultant $Q_{i,j}^a$ values will be used for updating $R_{i,j}^a$ as seen in the second step, employing the reduced complexity FFT-based method of Equation 4.8. The process of using the FFT for updating $R_{i,j}^a$ will now be described with the aid of a numerical example.

Suppose we have to update $R_{i,j}^a$ for the non-zero entries in the first row of $\mathbf{H}_q$ seen in Table 4.2. The message symbols involved in the 1^{st} row of $\mathbf{H}_q$ in Table 4.2 are {3, 7, 8, 9}. First we have to permute the probabilities of each variable node according to the matrix elements in $\mathbf{H}_q$ for the sake of

obtaining the PDF of the variable $\check{Q}^v_{1,j}$, where j denotes the column indices of each non-zero entry in the first row of $\mathbf{H}_q$ in Table 4.2. Thus the permuted probabilities after considering the matrix elements of the four message nodes in the first parity check are listed in Table 4.5.

	0	1	2	3
3	0.968776	0.00991634	0.000215896	0.0210919
7	0.582222	0.0116672	0.398132	0.00797822
8	0.000738692	0.00184169	0.997418	1.36396×10^{-6}
9	0.00926008	0.990686	5.37678×10^{-5}	5.02575×10^{7}

Table 4.5: $\check{Q}^v_{i,j}$ values of the entries in the first row of $\mathbf{H}_q$ given in Table 4.2 after permuting the PDF of $Q^a_{i,j}$.

More explicitly, in Table 4.5, the top row quantifies the probability of each of the four legitimate symbol states, and the first column on the left indicates the column index of the non-zero PCM entries participating in the first row of $\mathbf{H}_q$ seen in Table 4.2. Following the probability permutation carried out according to the corresponding matrix entry $\mathbf{H}_{1,j'}$ seen in Table 4.2 for the sake of obtaining the PDF of the variable $\check{Q}^v_{1,j'}$, the FFT is carried out for each symbol listed in Table 4.5 over $\mathbf{GF}(4)$ using the matrix defined by Equation 4.8. The result of the FFT is provided in Table 4.6, where we are using the same notation of $\hat{Q}$, as in Equation 4.10 for denoting the result of the FFT. For example, the PDF of the non-zero entry in the 8^{th} column and the 1^{st} row of Table 4.2 is summarised in the 8^{th} row of Table 4.4 as $(Q^0_{1,8}, Q^1_{1,8}, Q^2_{1,8}, Q^3_{1,8}) = (0.000738692, 0.997418, 1.36396 \times 10^{-6}, 0.00184169)$. Upon taking into account that the matrix entry at position $(1,8)$ of Table 4.2 is 2, the PDF of the variable $\check{Q}^v_{1,8}$ is a permuted version of the PDF of the variable $Q^a_{1,8}$. Since according to Table 4.1 the product of the sequence $(0,1,2,3)$ and the element 2 over $\mathbf{GF}(4)$ results in (0, 2, 3, 1), the PDF of the variable $\check{Q}^v_{1,8}$ is $(Q^0_{1,8}, Q^3_{1,8}, Q^1_{1,8}, Q^2_{1,8}) = (0.000738692, 0.00184169, 0.997418, 1.36396 \times 10^{-6})$. The PDF of $\check{Q}^v_{1,8}$ will now be multiplied by the FFTM defined by Equation 4.8 to perform the FFT as follows:

$$\begin{pmatrix} \hat{Q}^0_{1,8} \\ \hat{Q}^1_{1,8} \\ \hat{Q}^2_{1,8} \\ \hat{Q}^3_{1,8} \end{pmatrix} = \begin{pmatrix} 0.000738692 \\ 0.00184169 \\ 0.997418 \\ 1.36396e-06 \end{pmatrix} \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \end{pmatrix} = \begin{pmatrix} 1 \\ 0.996314 \\ -0.994839 \\ -0.99852 \end{pmatrix}, \tag{4.11}$$

which constitutes the row corresponding to the 8^{th} symbol of the codeword seen in the 3^{rd} row of Table 4.6.

	$\hat{Q}^0_{1,j}$	$\hat{Q}^1_{1,j}$	$\hat{Q}^2_{1,j}$	$\hat{Q}^3_{1,j}$
3	1	0.937983	0.957384	0.979736
7	1	0.960709	0.187779	0.180401
8	1	0.996314	-0.994839	-0.99852
9	1	-0.981372	0.999891	-0.981479

Table 4.6: FFT of the variables seen in Table 4.5 computed using Equation 4.8.

Following the FFT, $\hat{R}^q_{1,j}$ will be updated by the product of $Q^{\hat{a}}_{1,j'}$ of the other entries in the 1^{st} row

of $\mathbf{H}_q$ according to Equation 4.10. For example, the $\hat{R}^2_{1,7}$ value associated with the 7^{th} symbol in the codeword is calculated according to Equation 4.10 using the $\hat{Q}^a$ values seen in Table 4.6 as $0.957384 \times -0.994839 \times 0.999891 = -0.95234$. Hence, after the update of $R^a_{i,j}$ according to Equation 2.31, the quantities $R^a_{i,j}$ listed in Table 4.7 are obtained.

	$\hat{R}^0_{1,j}$	$\hat{R}^1_{1,j}$	$\hat{R}^2_{1,j}$	$\hat{R}^3_{1,j}$
3	1	-0.939338	-0.186789	0.176797
7	1	-0.917118	-0.95234	0.960166
8	1	-0.884343	0.179757	-0.173471
9	1	0.897808	-0.178849	-0.176483

Table 4.7: Results after updating the FFT of the variables seen in Table 4.6 using Equation 4.10.

Having completed the updating process of Equation 4.10, the inverse FFT is required for retrieving the PDF of the variable $\hat{R}^a_{0,j'}$, i.e. the PDF of the variable $\check{R}^v_{1,j'}$. The inverse FFT is carried out by multiplying the PDF vector with the FFTM in the same way as during the FFT process, except that all the resultant probabilities have to be normalised to ensure that we have $\sum_i \hat{R}^i_{1,j'} = 1$. Finally, the sequence obtained after the IFFT has to be permuted again for the sake of obtaining the PDF of $R_{i,j}$.

When the process outlined above has been completed for all the rows, the update of the quantity $R^a_{i,j}$ obeying Equation 4.10 of step 2.3 of Algorithm 2 is deemed completed for this iteration. Then the update of $Q^a_{i,j}$ according to Equation 2.32 of step 3 will be carried out using both the value of $R^a_{i',j}, i' \in R(j), i' \neq i$ as well as the intrinsic probability calculated from the channel's soft output, as seen in Equation 2.32. For example, there are two non-zero entries in the sixth column of Table 4.2 in row 4 and row 5. The previously calculated values of $R^a_{i,j}$, which correspond to these two entries in the fifth column of Table 4.2 were given in Table 4.8.

As seen in Table 4.2, the Hamming-weight of the 6^{th} column is only two, thus according to Equation 2.32 the non-zero entry in the 4^{th} row and the 6^{th} column will use the product of the value $R^a_{5,6}$ and the intrinsic symbol probability provided by the channel for updating the value of $Q^a_{4,6}$. Recall from Table 4.4 that the channel output defines the fifth symbol's intrinsic probabilities is now {0.825772, 0.00632498, 0.166626, 0.00127627} for the four possible **GF**(4) symbols of 0 to 3. For example, according to Equation 2.32 the value of $Q^0_{4,6}$ is updated by evaluating the product of the intrinsic probability of the 6^{th} symbol, which is 0.825772 in Table 4.4, and the value of $R^0_{5,6} = 0.0616686$ seen in Table 4.8. The updated value of $Q^0_{4,6}$ obeying Equation 2.32 becomes 0.0509242. Hence, using the same mechanism, $Q^q_{4,6}$, $q = 0 \ldots 3$, will be updated according to Equation 2.32, yielding {0.0509242, 0.000128521, 0.126089, 0.000205854}, respectively. After normalisation by a factor of $\sum_{a=0}^{q-1} Q^a_{4,6} = 0.177347$ according to Equation 2.32 of step 3 in Algorithm 2, the quantity $Q^a_{4,6}$ is obtained as {0.287143, 0.000724683,

	0	1	2	3
4	0.0151071	0.878271	0.0204158	0.0862058
5	0.0616686	0.0203196	0.756719	0.161293

Table 4.8: $R_{i,j}$ values for the two non-zero entries in the sixth column and row 3 and row 4 after the horizontal update process.

	0	1	2	3	Survivor	Result
1	0.00105059	5.48387×10^{-5}	0.866057	0.132838	2	Correct
2	6.61436×10^{-6}	0.00186265	0.00736306	0.990768	3	Correct
3	0.0201037	0.467753	0.512142	1.83549×10^{-6}	2	Error
4	7.38459×10^{-5}	1.40357×10^{-5}	0.999912	1.17234×10^{-8}	2	Correct
5	3.13059×10^{-7}	0.000590241	0.000433858	0.998976	3	Correct
6	0.22144	0.0324903	0.740962	0.00510794	2	Correct
7	0.871147	1.5601×10^{-5}	0.105775	0.0230627	0	Correct
8	0.00287799	0.96791	4.58809×10^{-6}	0.0292073	1	Correct
9	0.00909031	3.7477×10^{-9}	0.990825	8.4621×10^{-5}	2	Correct
10	0.000853625	8.65283×10^{-6}	0.998793	0.000344312	2	Correct

Table 4.9: A posteriori symbol probabilities after one iteration

	0	1	2	3	Survivor	Result
1	0.0132287	4.76994×10^{-6}	0.985821	0.000945989	2	Correct
2	0.00037386	0.0169579	0.0222603	0.960408	3	Correct
3	0.000245489	0.97209	0.0276643	6.13229×10^{-8}	1	Correct
4	0.00184083	7.75363×10^{-5}	0.998081	8.69016×10^{-7}	2	Correct
5	4.64757×10^{-7}	0.00588021	2.05722×10^{-5}	0.994099	3	Correct
6	0.0110328	0.00145767	0.987503	6.23818×10^{-6}	2	Correct
7	0.814087	2.0928×10^{-6}	0.184293	0.00161807	0	Correct
8	0.00401789	0.989768	5.66533×10^{-8}	0.00621357	1	Correct
9	0.00013802	1.49272×10^{-8}	0.999834	2.82329×10^{-5}	2	Correct
10	0.000980569	8.14098×10^{-5}	0.996534	0.00240409	2	Correct

Table 4.10: A posteriori symbol probabilities after two iterations

0.710972, 0.00116073}.

According to Equation 2.33, the *a posteriori* symbol probability will be calculated as the product of all the $R^a_{i,j}$ values in the column and the corresponding intrinsic probability provided by the channel. Thus, after normalisation according to Equation 2.33, the *a posteriori* probability of all the symbols in the codeword is obtained and these are summarised in Table 4.9 after the first iteration.

Comparing the results of Table 4.4 and Table 4.9, we can observe that the erroneous symbol in the sixth position of the codeword has been corrected. Although the third symbol remains incorrect, the probability of its original correct value has been increased from 0.02 to 0.467. During the second iteration, the *a posteriori* symbol probabilities are calculated using the quantity $Q^a_{i,j}$ determined in the previous iteration and the intrinsic symbol probabilities provided by the channel. The *a posteriori* probabilities generated after the second iteration are listed in Table 4.10.

As seen from Table 4.10, the second iteration has succeeded in correcting the remaining erroneous symbol and the codeword constituted by the surviving symbols results in an all-zero vector upon multiplying it by the PCM $\mathbf{H}_q$ of Table 4.2. Hence the decoder declares that a legitimate codeword

has been found and the iterations are curtailed.

4.3.5 Complexity

As described in Section 2.7.2, employing the FFT-based decoding method during the update of the quantity $R_{i,j}^{a}$ significantly reduces the decoding complexity imposed. In this section, the decoding complexity of the non-binary LDPCC will be evaluated. Let us consider the complexity calculation using an example in the context of **GF**(8).

The FFT matrix derived for **GF**(8) in Equation 4.9 is reproduced here for the reader's convenience:

$$\begin{pmatrix} \mathcal{F}(f)(0) \\ \mathcal{F}(f)(1) \\ \mathcal{F}(f)(2) \\ \mathcal{F}(f)(3) \\ \mathcal{F}(f)(4) \\ \mathcal{F}(f)(5) \\ \mathcal{F}(f)(6) \\ \mathcal{F}(f)(7) \end{pmatrix} = \begin{pmatrix} f(0) \\ f(1) \\ f(2) \\ f(3) \\ f(4) \\ f(5) \\ f(6) \\ f(7) \end{pmatrix} \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 & 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 & 1 & -1 & -1 & 1 \\ 1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 \\ 1 & -1 & 1 & -1 & -1 & 1 & -1 & 1 \\ 1 & 1 & -1 & -1 & -1 & -1 & 1 & 1 \\ 1 & -1 & -1 & 1 & -1 & 1 & 1 & -1 \end{pmatrix}. \tag{4.12}$$

Since the matrix given in Equation 4.12 is symmetric according to the matrix construction outlined in Section 4.3.3, the complexity of implementing the FFT is given by $qlog_2(q)$ **GF**(q) additions, where q is the size of the Galois field. Explicitly, the evaluation of Equation 4.12 is detailed as follows. Firstly, we calculate the values hosted by the each pair of two consecutive elements of the matrix product, namely $f(0)$ and $f(1)$, $f(2)$ and $f(3)$, etc. The calculation to be carried out at this step is as follows:

$$\begin{aligned} &(f(0)+f(1)) \\ &(f(2)+f(3)) \\ &(f(4)+f(5)) \\ &(f(6)+f(7)) \\ &(f(0)-f(1)) \\ &(f(2)-f(3)) \\ &(f(4)-f(5)) \\ &(f(6)-f(7)). \end{aligned} \tag{4.13}$$

The operations outlined in Equation 4.13 requires $q = 8$ additions in the first step. Then, the values calculated during the first step using Equation 4.13 will be invoked for obtaining the following values:

$$\begin{aligned} &(f(0)+f(1))+(f(2)+f(3)) \\ &(f(0)+f(1))-(f(2)+f(3)) \\ &(f(4)+f(5))+(f(6)+f(7)) \\ &(f(4)+f(5))-(f(6)+f(7)) \\ &(f(0)-f(1))+(f(2)-f(3)) \\ &(f(0)-f(1))-(f(2)-f(3)) \\ &(f(4)-f(5))+(f(6)-f(7)) \\ &(f(4)-f(5))-(f(6)-f(7)). \end{aligned} \tag{4.14}$$

This second step formulated in Equation 4.14 requires a further $q = 8$ additions. Finally, the values generated during the second step can be used for calculating the final result of the FFT, as follows:

$$\begin{aligned}
&\Big((f(0)+f(1))+(f(2)+f(3))\Big)+\Big((f(4)+f(5))+(f(6)+f(7))\Big)\\
&\Big((f(0)-f(1))+(f(2)-f(3))\Big)+\Big((f(4)-f(5))+(f(6)-f(7))\Big)\\
&\Big((f(0)+f(1))-(f(2)+f(3))\Big)+\Big((f(4)+f(5))-(f(6)+f(7))\Big)\\
&\Big((f(0)-f(1))-(f(2)-f(3))\Big)+\Big((f(4)-f(5))-(f(6)-f(7))\Big)\\
&\Big((f(0)+f(1))+(f(2)+f(3))\Big)-\Big((f(4)+f(5))+(f(6)+f(7))\Big)\\
&\Big((f(0)-f(1))+(f(2)-f(3))\Big)-\Big((f(4)-f(5))+(f(6)-f(7))\Big)\\
&\Big((f(0)+f(1))-(f(2)+f(3))\Big)-\Big((f(4)+f(5))-(f(6)+f(7))\Big)\\
&\Big((f(0)-f(1))-(f(2)-f(3))\Big)-\Big((f(4)-f(5))-(f(6)-f(7))\Big).
\end{aligned} \tag{4.15}$$

Thus again, the final step in Equation 4.15 requires $q = 8$ additions. By summing up the number of additions listed above, a total of $qlog(q)$ additions are required for carrying out the FFT. Since the IFFT will follow the same routine, so the number of additions required for carrying out the IFFT will also be $qlog_2(q)$. However, additionally, the normalisation operation is also required during the IFFT, hence a further q multiplications are necessary. Thus the overall complexity required for each symbol during the FFT of quantity $Q^a_{i,j'}$ and the IFFT of quantity $R^a_{i,j'}$ is $2qlog_2(q)$ additions and q multiplications.

The above complexity calculations only considered the complexity required for the FFT and the IFFT. However, according to Equation 4.10, updating the quantity $R^a_{i,j}$ using Equation 4.6 also involves the multiplications using the $Q^a_{i,j'}$ values from other message nodes participating in the i^{th} check. Thus referring back to Table 2.16 in Section 2.7.3, $3w_rq$ multiplications are needed for the forward-backward calculation of updating the quantity $R^a_{i,j'}$ in Equation 4.6 for all the symbols in the row, where k represents the row weight. Summing up all the complexity contributions, updating the quantity $R^a_{i,j}$ according to Equation 4.6 will incur $2w_rq \cdot log_2(q)$ additions and $4w_rq$ multiplications for each row.

By summing up all the complexity contribution of the rows, the overall complexity of all the non-zero entries in the PCM required for updating $R^a_{i,j}$ according to Equation 4.6 will be $2M_qw_rq \cdot log_2(q)$ additions and $4M_qw_rq$ multiplications, where M_q represents the number of rows in the PCM $\mathbf{H}_q$ of Table 4.2. Since the number of non-zero entries in the parity check matrix is constant, we have $M_qw_r = N_qw_c$, with N_q and w_c representing the number of coded symbols and the mean column weight of the parity check matrix, respectively. By replacing M_qw_r with N_qw_c in the overall complexity calculations, followed by a division of the total number of coded symbols N_q, each coded symbol will impose a computational complexity of $2w_cq \cdot log_2(q)$ additions and $4w_cq$ multiplications per iteration.

Updating the quantity $Q^a_{i,j}$ using Equation 2.32 will incur another $3w_rq$ multiplications using the forward-backward recursion calculation according to Equation 2.32, thus the overall complexity of each coded symbol in one iteration will be $2w_cq \cdot log_2(q)$ additions and $7w_cq$ multiplications. Furthermore, since a symbol decoding step carried out over the Galois field of size q represents $p = log_2(q)$ constituent bits, thus if we evaluate the bit-wise decoding complexity, the above-mentioned decoding complexity per coded symbol has to be divided by p, resulting in a complexity figure for each coded bit corresponding to $2w_cq$ additions and $7w_cq/log_2(q)$ multiplications. Hence, the decoding complexity related to each original information bit becomes $2w_cq/r$ additions and $7w_cq/(log_2(q) \cdot r)$ multiplica-

$\mathbf{H}_q$	$(500, 250)_q$
	$(1000, 500)_q$
	$(2000, 1000)_q$
Average symbol column weight	2.5, 3
Decoding GF	GF(2), GF(4), GF(8), GF(16)
Modulation scheme	BPSK
Channel	AWGN
Number of LDPC iterations	25

Table 4.11: Simulation parameters for three different half-rate non-binary LDPCCs decoded over GF(q) using BPSK modulation, when communicating over an AWGN channel. The values of q used for the various decoding field are 2, 4, 8, 16.

tions, where r represents the code rate. Naturally, the true decoding complexity critically depends on the specific implementation considered and hence the above-mentioned complexity estimations are only indicative.

4.4 Performance of non-binary LDPC codes

Having estimated the decoding complexity, in this section, the achievable performance of the non-binary LDPCCs will be evaluated.

4.4.1 Performance when the size of $\mathbf{H}_q$ is maintained

As mentioned in Section 4.3, upon increasing the decoding field order, while maintaining the size of the PCM, the error correction capability of the LDPCC may be improved owing to its higher equivalent binary column weight. Hence in this section, we will demonstrate how the performance of the non-binary LDPCC may be improved with the aid of increasing the decoding field $\mathbf{GF}(q)$, without altering the size of the PCM $\mathbf{H}_q$. The related simulation parameters can be found in Table 4.11.

As seen in Figures 4.3 to 4.5, the BER performance of the LDPC codes characterised in Table 4.11 increases with respect to the decoding field order. However, when using an average symbol column weight of 2.5, the non-binary LDPC codes suffer from the same error floor problem, as described in Chapter 3. The LDPC codes having a short blocklength in Figure 4.3, and associated with $\mathbf{GF}(2)$ and $\mathbf{GF}(4)$ showed a performance curve cross-over, when different average symbol column weights were applied. By contrast, this cross-over phenomenon does not occur for higher-order Galois fields such as $\mathbf{GF}(8)$ and $\mathbf{GF}(16)$ at this blocklength. When the blocklength is increased in Figure 4.4 and Figure 4.5, the associated error floor phenomenon becomes less significant, although it is still visible.

Generally speaking, when increasing the decoding field order, while maintaining the size of $\mathbf{H}_q$, the BER performance of the non-binary LDPC code can be improved, although the associated improvement is achieved at the cost of a higher decoding complexity. However, as outlined in Section 4.3.5, the decoding complexity can be reduced, when the FFT and IFFT are used. More explicitly, the associated decoding complexity defined in Section 4.3.5 and evaluated for the codes specified in Table 4.11

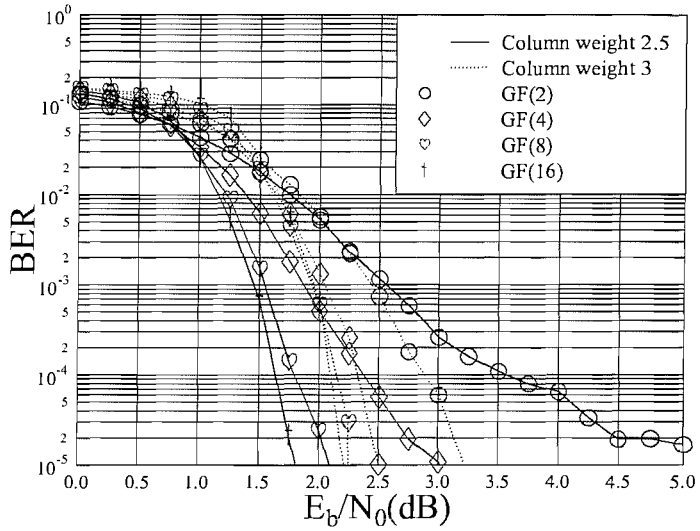


Figure 4.3: BER performance of the non-binary LDPC code $(500, 250)_q$ parameterised in Table 4.11 operating over various Galois fields, when communicating in an AWGN channel. The achievable coding gain of the various schemes at a BER of 10^{-4} will be summarised in Figure 4.6 and Table 4.20.

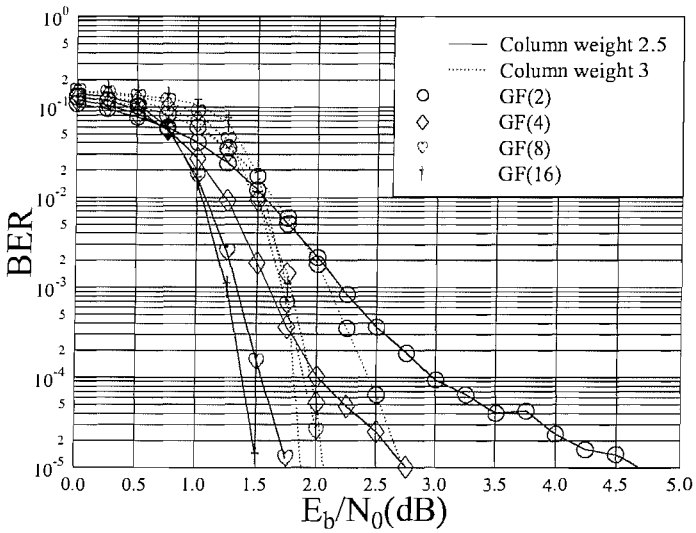


Figure 4.4: BER performance of the non-binary LDPC code $(1000, 500)_q$ parameterised in Table 4.11 operating over various Galois fields, when communicating in an AWGN channel. The achievable coding gain of the various schemes at a BER of 10^{-4} will be summarised in Figure 4.6 and Table 4.20.

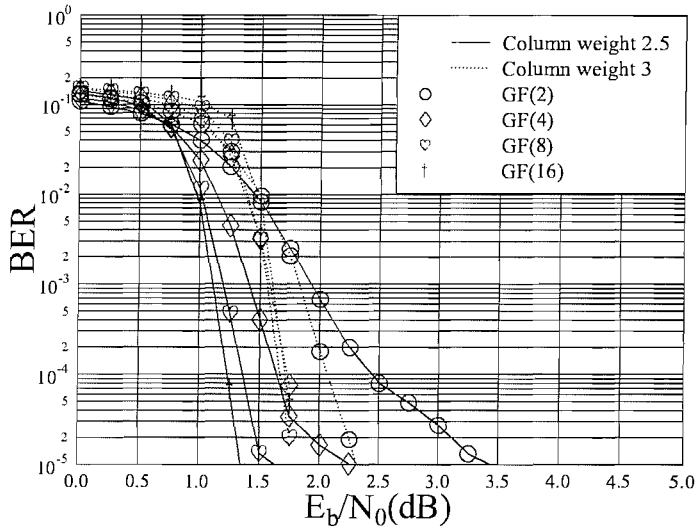


Figure 4.5: BER performance of the non-binary LDPC code $(2000, 1000)_q$ parameterised in Table 4.11 operating over various Galois fields, when communicating in an AWGN channel. The achievable coding gain of the various schemes at a BER of 10^{-4} will be summarised in Figure 4.6 and Table 4.20.

utilising various decoding fields can be found in Figure 4.6.

More explicitly, in Figure 4.6, the coding gain versus the associated decoding complexity per information bit is plotted. It can be observed that upon using higher-order decoding fields, a higher coding gain is attained at the cost of an increased complexity. When the decoding field is extended from binary to $\mathbf{GF}(4)$, most of the attainable coding gain was achieved and hence upon further extending the decoding field to $\mathbf{GF}(8)$ and $\mathbf{GF}(16)$ results in modest further improvements, despite the fact that the complexity burden imposed upon extending the field from binary to $\mathbf{GF}(4)$ is significantly lower than to $\mathbf{GF}(8)$ or the $\mathbf{GF}(16)$ decoding field. In fact, the number of multiplications per information bit while operating over $\mathbf{GF}(4)$ is the same as that for $\mathbf{GF}(2)$. Thus it can be observed that $\mathbf{GF}(4)$ achieves an attractive trade-off between the achievable coding gain and the associated decoding complexity. Furthermore, it becomes clear in Figure 4.3 to 4.5 that an error floor might occur for the LDPC codes having a symbol column weight of 2.5, hence the codes using a symbol column weight of three might constitute a better choice in case of binary decoding. By contrast, as it transpires from Figures 4.3 to 4.5 the codes having symbol column weight 2.5 are superior in comparison to the codes having an symbol column weight of three, when operating in a higher decoding field.

4.4.2 Performance when the size of $\mathbf{H}_b$ is maintained

In Section 4.4.1, the performance of non-binary LDPC codes decoded over various Galois fields was evaluated, while the size of the PCM $\mathbf{H}_q$ was maintained. More explicitly, in Section 4.4.1 the LDPC codes studied had the same number of columns and rows. Since the non-zero entries within the PCM are defined over Galois fields having various sizes, the size of the equivalent binary PCM $\mathbf{H}_b$ is

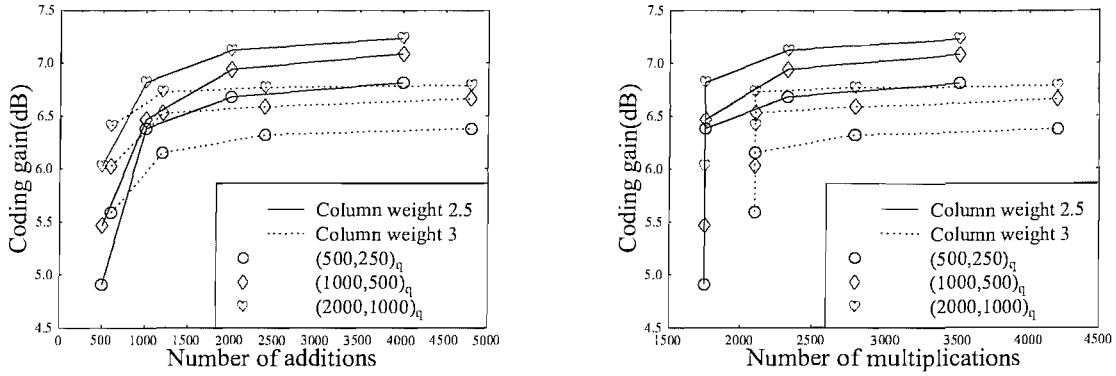


Figure 4.6: Coding gain versus decoding complexity of the non-binary LDPC codes tabulated in Table 4.11, when communicating over an AWGN channel. The complexity quoted quantifies the number of arithmetic operations per information bit. The four q values used are 2, 4, 8, 16 and the complexity value related to each decoding field are plotted in the figure from left to right while the decoding field order is increased.

increased in the context of a larger Galois field. Thus in this section, we will maintain the size of the equivalent binary PCM $\mathbf{H}_b$ and evaluate the achievable performance of the same LDPC codes decoded over $\mathbf{GF}(q)$ at two different code rate, when communicating over an AWGN channel. The associated simulation parameters are summarised in Table 4.12.

Figure 4.7 and 4.8 characterise the attainable performance of the quarter-rate and half-rate LDPC codes studied having an equivalent binary PCM size of 4000 bits. For the quarter-rate LDPC codes characterised in Figure 4.7 decoding over a larger Galois field degraded the achievable performance, when the average column weight is three. As discussed previously in Section 4.3.2, the effect of extending the decoding Galois field is two-fold. Recall that in this experiment the size of $\mathbf{H}_q$ was reduced for larger Galois fields for the sake of maintaining a constant equivalent binary PCM size. Hence the code becomes less sparse and the cycles identified in the PCM will have a shorter length. Hence in this case the disadvantages introduced by using a larger decoding field cannot be compensated by the corresponding advantages. However, as seen in Figure 4.7 if we reduce the average column weight to 2.5, then the overall performance becomes better than that of its column-weight=3 counterpart and ultimately the codes operating over larger Galois fields exhibit a better BER performance.

As seen in Figure 4.8, when the code rate is increased to half, the codes having a column weight of 2.5 exhibit similar performance trends to those observed for the quarter-rate scheme characterised in Figure 4.7. For the scenario, which an average column weight of three, decoding over $\mathbf{GF}(4)$ still achieves a marginally better performance than $\mathbf{GF}(2)$. However, when the decoding field is further increased to $\mathbf{GF}(8)$ and $\mathbf{GF}(16)$, the achievable BER performance becomes worse than the $\mathbf{GF}(2)$ owing to similar reasons to those mentioned in the quarter-rate case discussed above.

$\mathbf{H}_q$ Quarter-rate coding	$(4000, 1000)_q$, GF(2)
	$(2000, 500)_q$, GF(4)
	$(1336, 334)_q$, GF(8)
	$(1000, 250)_q$, GF(16)
$\mathbf{H}_q$ Half-rate coding	$(4000, 2000)_q$, GF(2)
	$(2000, 1000)_q$, GF(4)
	$(1334, 667)_q$, GF(8)
	$(1000, 500)_q$, GF(16)
Average symbol column weight	2.5, 3
Modulation mode	BPSK
Channel	AWGN
Number of LDPC iterations	25

Table 4.12: Simulation parameters for two sets of non-binary LDPCCs having a code-rate of one-fourth and a half and operating over $\mathbf{GF}(q)$ using BPSK modulation, when communicating over an AWGN channel. The decoding Galois field associated with each specific LDPC code is given and the size of $\mathbf{H}_q$ is tabulated, while the size of the equivalent binary PCM $\mathbf{H}_b$ is maintained at 4000 bits. (Slightly higher for $\mathbf{GF}(8)$.)

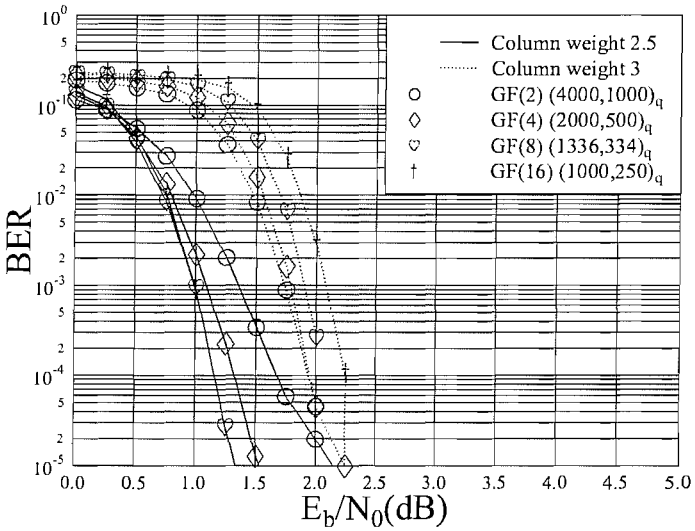


Figure 4.7: BER performance of the various quarter-rate non-binary LDPC codes characterised in Table 4.12 having an average column weight of 2.5 as well as 3 and decoded over their associated Galois fields, when communicating over an AWGN channel. The achievable coding gain of the various schemes at a BER of 10^{-4} will be summarised in Table 4.21.

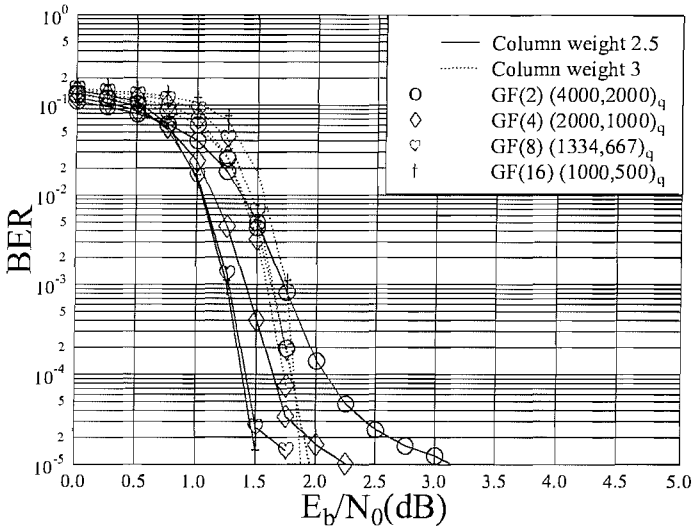


Figure 4.8: BER performance of the various half-rate non-binary LDPC code in Table 4.12 having an average column weight of 2.5 as well as 3 and decoded over their associated Galois fields, when communicating over an AWGN channel. The achievable coding gain of the various schemes at a BER of 10^{-4} will be summarised in Table 4.21.

4.4.3 Performance of non-binary LDPC codes using various code rates

In Section 4.4.1, the performance of a half-rate non-binary LDPC codes has been evaluated at various blocklengths. In this section, we will demonstrate how the LDPC codes behave at various code rates, when the coded blocklength was fixed at 2100 non-binary symbols.

The simulation parameters are given in Table 4.13.

Figures 4.9 to 4.13 have illustrated the achievable BER performance of the non-binary LDPC codes summarised in Table 4.13 at various code rates. The attainable coding gain at a BER of 10^{-4} is plotted in Figure 4.14 for the various code rates considered. For a relatively low code rate ranging from $r = 0.33$ to $r = 0.66$, using a column weight of 2.5 renders the LDPC codes more robust to channel errors for all the Galois fields considered than the column-weight 3 codes. By contrast, for the rates of $r = 0.75$ and $r = 0.8$ we observe in Figure 4.14 that owing to error floor problem discussed in Section 4.4.1, the codes having an average column weight of 2.5 suffer from undetected errors, when size of the Galois field utilised is limited. However, crossover of the BER curves is observed when the decoding field order is increased and thus on balance we may argue that using a column weight of 2.5 still constitutes a better configuration than using an average column weight of three.

Non-binary LDPC code	$(2100, 700)_q$, r=0.33
	$(2100, 1050)_q$, r=0.50
	$(2100, 1400)_q$, r=0.66
	$(2100, 1575)_q$, r=0.75
	$(2100, 1680)_q$, r=0.80
Channel	AWGN
Modulation Mode	BPSK
Decoding Galois fields	**GF**(2), **GF**(4), **GF**(8), **GF**(16)
LDPC average column weight	2.5, 3

Table 4.13: Simulation parameters for non-binary LDPCCs having a blocklength of 2100 non-binary symbols operating at various coding rates, when communicating over an AWGN channel using BPSK modulation.

4.5 Bit-based joint detection scheme

Having introduced and characterised the family of non-binary LDPCCs, let us now focus our attention on a novel application in the context of space-time codes. The roots of this non-binary scheme germinated in the binary system proposed by Meshkat and Jafarkhani [108], which is shown in Figure 4.15. The basic philosophy of this space-time coding scheme is reminiscent of Alamouti's simple repetition-based two-antenna coding scheme [135], although a half-rate LDPC code was used by the authors. Hence the number of bits was doubled by the LDPC encoder and two separate antennas were used for transmitting the bits. Provided that the two antennas are sufficiently far apart, their fading envelope may be expected to be independent. Since the scheme proposed by Meshkat and Jafarkhani employs a powerful half-rate LDPCC instead of Alamouti's less potent half-rate repetition-code, the achievable performance is substantially improved. For further background on space-time coding, please refer to [115].

Let us now outline the approach of this binary LDPCC-based space-time code in more detail. Suppose the source bit stream $\mathbf{U}$ is encoded into a binary codeword $\mathbf{B}$ by a binary LDPC encoder constituted by the bits $(b_0, \ldots, b_3)$ in Figure 4.15. According to the modulation scheme used, the coded bits $(b_0, \ldots, b_3)$ are mapped onto their corresponding non-binary signal constellations and a symbol sequence $\mathbf{S}$=(S_0, S_1) is obtained. The symbol sequence $\mathbf{S}$ will be transmitted using n_t transmitters, and at the receiver side, there are n_r receivers. The n_r received samples are correlated with each other since they originate from the same set of n_t transmitted samples generated from $\mathbf{S}$. This inter-sample correlation may be beneficially exploited upon exchanging it also with the LDPC decoder, and hence extra iterative coding gain may be achieved. A Tanner graph is used in Figure 4.16 for demonstrating the process of information passing during the joint decoding process of the space-time and LDPC decoder.

The nodes marked $\mathbf{r}$, $\mathbf{v}$ and $\mathbf{c}$ represent the received signal samples, the LDPC code's message nodes and the check nodes, respectively. The message exchanged between the vectors $\mathbf{r}$ and $\mathbf{v}$ represents the information exchange between the demodulator and the LDPC decoder. By contrast, the message flow between the vectors $\mathbf{v}$ and $\mathbf{c}$ is part of the internal decoding process of the LDPC decoder.

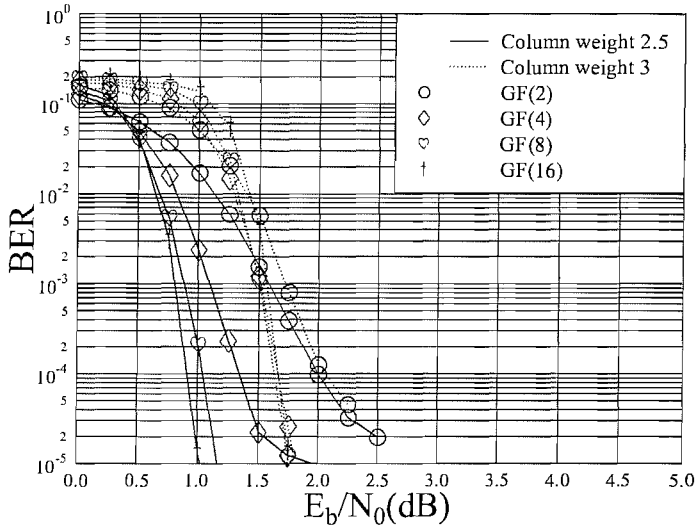


Figure 4.9: BER performance of the $(2100, 700)_q$ non-binary LDPC code characterised in Table 4.13 and decoded over four different Galois fields, when communicating over an AWGN channel. We note that the different codes have an identical code rates but the number of encoded bits per LDPC codeword is proportionally increased with respect to the associated Galois field. Hence the number of LDPC coded bits per codeword becomes 2100, 4200, 6300 and 8400, respectively for **GF**(2), **GF**(4), **GF**(8) and **GF**(16). The achievable coding gain of the various schemes at a BER of 10^{-4} will be summarised in Figure 4.14 and Table 4.22.

In comparison to the conventional scheme, where the demodulator only evaluates the channel's soft output once and leaves all the remaining operations for the channel decoder to carry out, this scheme allows the demodulator to accept extra information from the channel decoder for the sake of exploiting the channel's soft output as best as possible. The demodulator receives the channel output samples, and calculates the soft channel-output metric, which can be expressed for a Gaussian channel as follows [115]:

$$M_{r->v}(b_k) = \sum_{all\ \mathbf{B}_i} \frac{1}{(\sqrt{2\pi}\sigma_n)^{n_r}} \exp\left(-\frac{|\mathbf{r} - \mathbf{HS}(\mathbf{B}_i, b_k)|^2}{2\sigma_n^2}\right) \prod_{b_j \in \mathbf{B}_i} M_{v->r}(b_j), \tag{4.16}$$

where $M_{v->r}(b_j)$ represents the *a priori* information corresponding to the original information bits' soft estimate at the output of the LDPC decoder.

In Equation 4.16, σ_n represents the standard deviation of the Gaussian noise, while $\mathbf{H}$ is an $(n_r \times n_t)$-dimensional matrix containing the complex valued fading coefficients of each transmission path, where n_r and n_t are the numbers of receiver and transmitter antennas, respectively. Using *bps* for representing the number of bits per symbol for the corresponding modulation scheme, $\mathbf{B}_i$ represents the i^{th} set of $(n_t \times bps) - 1$ transmitted bits, but excludes the k^{th} bit b_k, which directly contributes to the value of the received vector $\mathbf{r}$ at the n_r number of receivers, while $\mathbf{S}(\mathbf{B}_i, b_k)$ is a vector of n_t components containing the modulated symbols corresponding to the bit set of $\mathbf{B}_i$ and b_k.

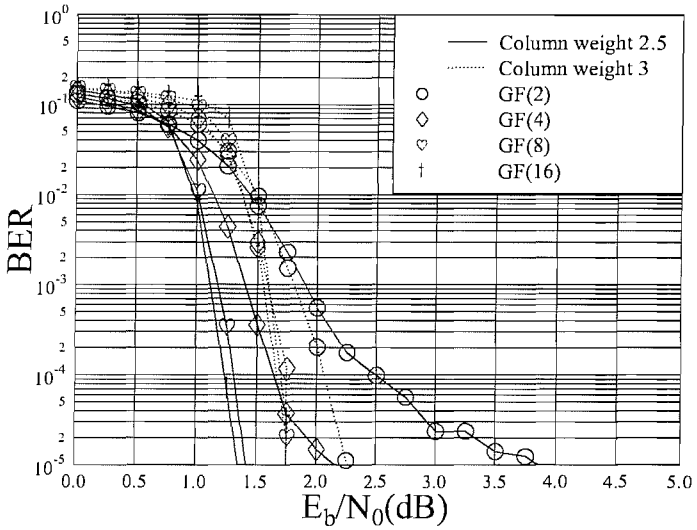


Figure 4.10: BER performance of the $(2100, 1050)_q$ non-binary LDPC code characterised in Table 4.13 and decoded over four different Galois fields, when communicating over an AWGN channel. We note that the different codes have an identical code rates but the number of encoded bits per LDPC codeword is proportionally increased with respect to the associated Galois field. Hence the number of LDPC coded bits per codeword becomes 2100, 4200, 6300 and 8400, respectively for **GF**(2), **GF**(4), **GF**(8) and **GF**(16). The achievable coding gain of the various schemes at a BER of 10^{-4} will be summarised in Figure 4.14 and Table 4.22.

More explicitly, let us consider Figure 4.15, for example. When using QPSK modulation, four bits are mapped into two QPSK symbols and transmitted by the two transmitters to the three receivers. If $M_{r->v}(b_2)$ is under consideration, then we have $\mathbf{B}_i = \{b_0, b_1, b_3\}$ and $(\mathbf{B}_i, b_2) = \{b_0, b_1, b_2, b_3\}$. The vector $\mathbf{r}$ will contain elements of $\{r_1, r_2, r_3\}$. For a particular set of $\mathbf{B}_i = \{b_0 = 1, b_1 = 1, b_3 = 0\}$, since QPSK modulation is employed in Figure 4.15, thus the notation $\mathbf{S}(\mathbf{B}_i, b_2)$ represents $\{11, 00\}$ or $\{11, 10\}$, depending on the specific value of b_2 concerned. For the sake of generating the soft channel-output metric $M_{r->v}(b_i)$ of Equation 4.16, we have to sum the terms associated with all possible bit-combinations incurred by the n_t transmitters using a particular multi-level modulation scheme. Using Figure 4.15 for example, for the sake of calculating the probability $M_{r->v}(b_2 = 1)$, we have to consider all possible values of $\mathbf{B}_i = \{b_0, b_1, b_3\}$ which includes three bits. Thus, there is a total of $2^3 = 8$ bit-combinations for $\mathbf{B}_i$ in conjunction with $b_2 = 1$, as seen in Table 4.14.

For each individual configuration of $(\mathbf{B}_i, b_2 = 1)$ shown in Table 4.14, the product seen at the right of Equation 4.16 quantifies the total *a priori* information available for b_2. For example, if the configuration listed in the final row of Table 4.14, which is given by $(\mathbf{B}_i, b_2 = 1) = (1, 1, 1, 1)$, is concerned, the product seen in Equation 4.16 will quantify the *a priori* probability available from the LDPC decoder, corresponding to the joint probability of $(b_0 = 1, b_1 = 1, b_3 = 1)$.

Again, the quantity $M_{v->r}$ in Equation 4.16 is the *a priori* information corresponding to the

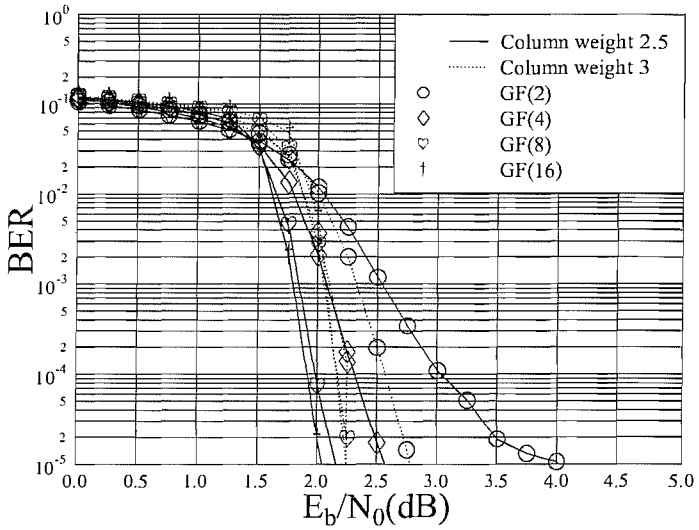


Figure 4.11: BER performance of the $(2100, 1400)_q$ non-binary LDPC code characterised in Table 4.13 and decoded over four different Galois fields, when communicating over an AWGN channel. We note that the different codes have an identical code rates but the number of encoded bits per LDPC codeword is proportionally increased with respect to the associated Galois field. Hence the number of LDPC coded bits per codeword becomes 2100, 4200, 6300 and 8400, respectively for **GF**(2), **GF**(4), **GF**(8) and **GF**(16). The achievable coding gain of the various schemes at a BER of 10^{-4} will be summarised in Figure 4.14 and Table 4.22.

$(\mathbf{B}_i, b_2 = 1)$				$\mathbf{S}(\mathbf{B}_i, b_2 = 1)$	
Tx 1		Tx 2		Tx 1	Tx 2
b_0	b_1	b_2	b_3	S_0	S_1
0	0	1	0	0	2
0	1	1	0	1	2
1	0	1	0	2	2
1	1	1	0	3	2
0	0	1	1	0	3
0	1	1	1	1	3
1	0	1	1	2	3
1	1	1	1	3	3

Table 4.14: All possible bit-combinations of a 4QAM, two antenna MIMO system for $\mathbf{B}_2$ in conjunction with $b_2 = 1$.

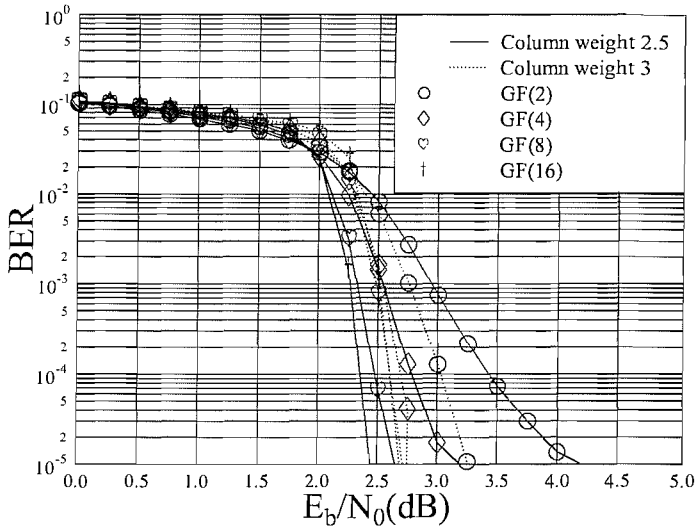


Figure 4.12: BER performance of the $(2100, 1575)_q$ non-binary LDPC code characterised in Table 4.13 and decoded over four different Galois fields, when communicating over an AWGN channel. We note that the different codes have an identical code rates but the number of encoded bits per LDPC codeword is proportionally increased with respect to the associated Galois field. Hence the number of LDPC coded bits per codeword becomes 2100, 4200, 6300 and 8400, respectively for **GF**(2), **GF**(4), **GF**(8) and **GF**(16). The achievable coding gain of the various schemes at a BER of 10^{-4} will be summarised in Figure 4.14 and Table 4.22.

original information bits' soft estimate at the output of the LDPC decoder, and it is provided by the LDPCC's message nodes **v**, rather than the check nodes. When evaluating Equation 4.16 for the first time, the product at the right-hand side of Equation 4.16 is not calculated, since the quantity $M_{v->r}$ is not available as yet. Equation 4.16 provides the soft information $M_{r->v}$, which is used for calculating the metric $M_{v->c_j}$ according to [108]:

$$M_{v->c_j}(b_i) = \alpha M_{r->v} \prod_{c_k \in C(b_i), k \neq j} M_{c_k->v}(b_i). \tag{4.17}$$

This process is actually the same as the updating the Q message of Equation 2.32 during the description of the LDPC code's iterative decoding process outlined in Chapter 2, where α is the normalisation factor, which ensures that the probabilities of b_i summed over all possible states add up to unity. Furthermore, $C(b_i)$ represents all the checks of the LDPCC's parity check matrix involving bit b_i. The multiplicative term $M_{r->v}$ seen in Equation 4.17 is quantified in Equation 4.16, which acts as the intrinsic probability f_j^a provided by the demodulator as in Equation 2.32. The soft information $M_{c_k->v}(b_i)$ is provided by the check nodes for the message nodes, which corresponds to the quantity $R_{k,j}^a$ in Equation 2.32. When Equation 4.17 is invoked for the first time, since the soft information $M_{c_k->v}(b_i)$ is not available as yet, hence the metric $M_{v->c_j}(b_i)$ in Equation 4.17 will be initialised to the value provided by $M_{r->v}$ of Equation 4.16.

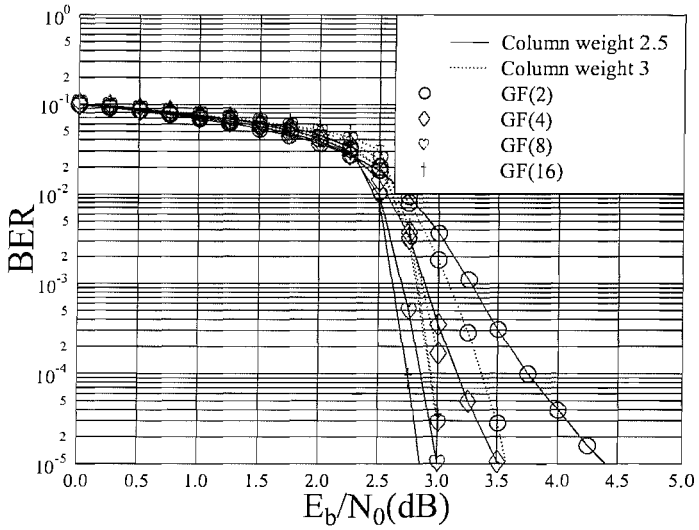


Figure 4.13: BER performance of the $(2100, 1680)_q$ non-binary LDPC code characterised in Table 4.13 and decoded over four different Galois fields, when communicating over an AWGN channel. We note that the different codes have an identical code rates but the number of encoded bits per LDPC codeword is proportionally increased with respect to the associated Galois field. Hence the number of LDPC coded bits per codeword becomes 2100, 4200, 6300 and 8400, respectively for **GF**(2), **GF**(4), **GF**(8) and **GF**(16). The achievable coding gain of the various schemes at a BER of 10^{-4} will be summarised in Figure 4.14 and Table 4.22.

Since the message nodes **v** are the nodes representing all the original coded bits we intended to decode, the "belief", or *a posteriori probability* has to be calculated at this stage, which is expressed as:

$$M_{apo}(b_i) = \alpha M_{r->v}(b_i) \prod_{c_k \in C(b_i)} M_{c_k->v}(b_i). \tag{4.18}$$

The resultant bit sequence generated by subjecting the *a posteriori probability* derived from Equation 4.18 to a hard decision will be checked against the LDPC's PCM. If there are unsatisfied checks, further iterations will be invoked until a pre-defined maximum number of iterations is reached. The metric $M_{r->v}(b_i)$ corresponds to the P_j^a term in Equation 2.33, while $M_{c_k->v}(b_i)$ represents $R_{k,j}^a$ accordingly.

Upon determining $M_{v->c_j}$ from Equation 4.17, the quantity $M_{c_j->v}$, which is the soft information provided by the check node **c** is calculated exactly the same way, as the updating of the message $R_{i,j}^a$ in Equation 2.31 of Chapter 2. The soft-metrics $M_{c_j->v}$ are then combined as follows:

$$M_{v->r}(b_i) = \alpha \prod_{c_k \in C(b_i)} M_{c_k->v}(b_i), \tag{4.19}$$

before being passed onto the **r** node shown in Equation 4.16. This equation is similar to the *a posteriori probability* calculation seen in Equation 2.33 of Chapter 2, although the *intrisic* information term P_j^a

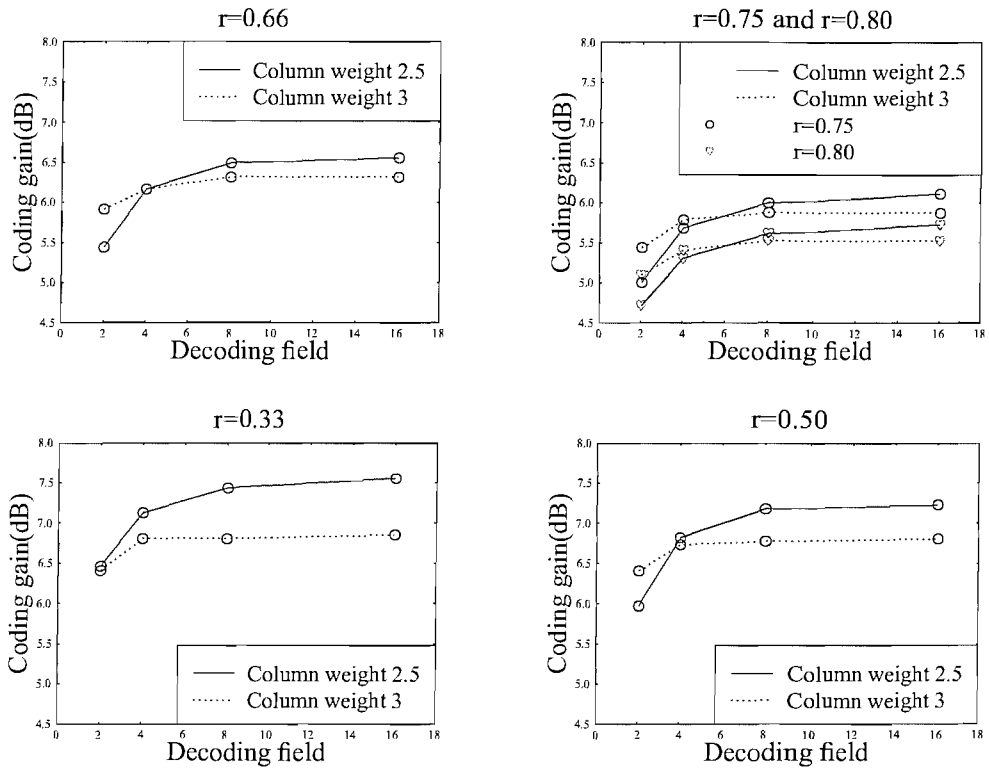


Figure 4.14: Coding gain versus decoding field size for the various non-binary LDPC codes characterised in Table 4.13 at a BER of 10^{-4}, when communicating over an AWGN channel.

of Equation 2.33 was omitted for the sake of providing only the *extrinsic information* for feeding it back to Equation 4.16, since the intrinsic probability $M_{r->v}$ was originally provided by the $\mathbf{r}$ node seen in Equation 4.16.

4.6 Symbol-based joint detection scheme

Having outlined the philosophy of the binary LDPC-based space-time codec scheme, let us now improve its performance using non-binary LDPCCs and a purely symbol-based space-time codec. Noting that in Equation 4.16 of the previous section it has been assumed that during the calculation of the product of the *a priori information* of the bits, the bits participating in the n_r number of received channel outputs were treated as independent variables. However, their independence is only approximately valid in Gray-coded non-binary modulation schemes. Furthermore, as it will be discussed in Section 4.9, the complexity of the bit-based scheme of Section 4.5 increases dramatically, when the number of bits per symbol is increased.

Hence, in this section we will further develop the bit-based LDPC-aided space-time codec of Section 4.5 into a symbol-based algorithm, which facilitates purely symbol-based message passing, involving a non-binary LDPC code.

Still using Figure 4.15 as an example, instead of using the probabilities of the bits in Figure 4.15, say

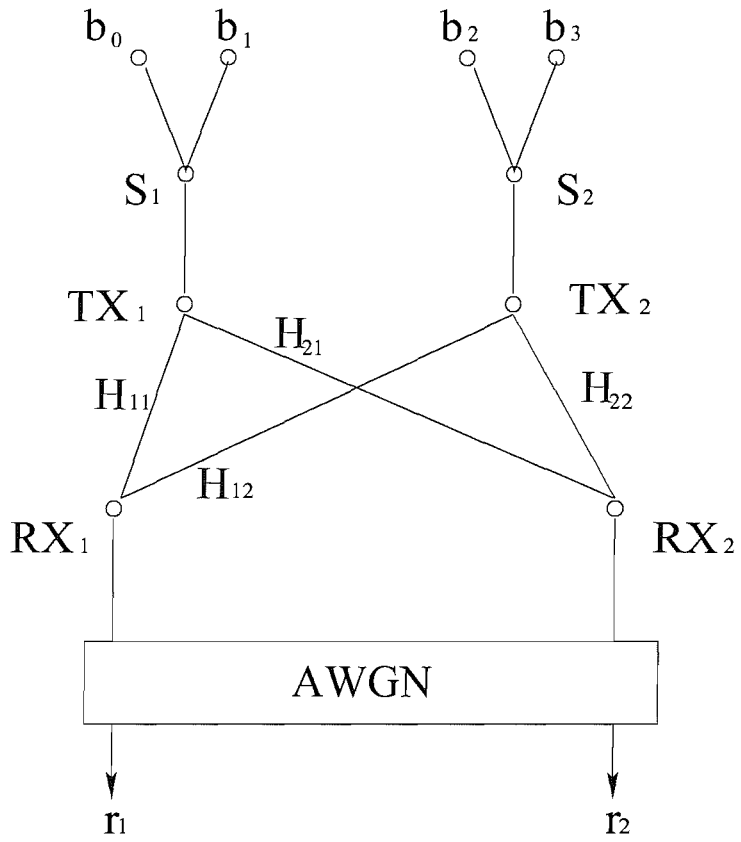


Figure 4.15: A two-transmitter, two-receiver system using binary LDPCC-based QPSK-modulated space-time coding.

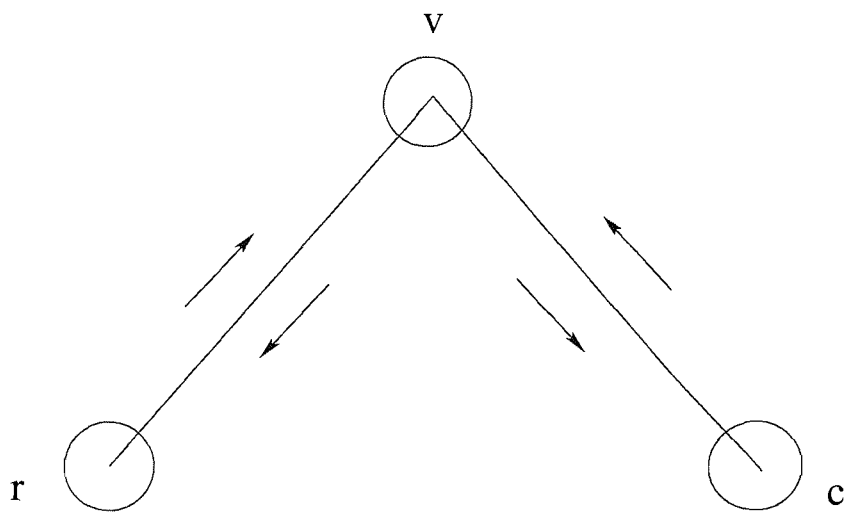


Figure 4.16: Tanner graph of the LDPC-aided space-time coding system, where **r**, **v** and **c** represent the received samples, as well as message and check nodes, respectively.

b_0, b_1, b_2, b_3, the probabilities of the symbols s_0, s_1 will be used. Thus Equation 4.16 can be rewritten as:

$$M_{r->v}(s_k) = \sum_{all\ S_i} \frac{1}{(\sqrt{2\pi}\sigma_n)^{n_r}} e^{-\frac{|\mathbf{r}-\mathbf{HS}(S_i,s_k)|^2}{2\sigma_n^2}} \cdot \prod_{s_j \in S_i} M_{v->r}(s_j). \quad (4.20)$$

In Equation 4.20, S_i now represents a set of $(n_t - 1)$ symbols rather than $(n_t \times bps - 1)$ bits, including all symbols, except for s_k, which directly contributes to the value of the received vector $\mathbf{r}$ at the output of the n_r receivers, and $\mathbf{S}(S_i, s_k)$ is a vector of size n_t containing the symbols including s_0 and s_1, as in Figure 4.15. More explicitly, rather than using the bits representing the symbols in Figure 4.15 for the calculation of the *a priori information*, the symbols are employed directly in this scheme. Hence, if symbol s_1 of Figure 4.15 is concerned, the vector S_i will have only one element, namely $\{s_0\}$. The *a priori information* $M_{v->r}(s_j)$ represents the probability of the j^{th} symbol, rather than that of the bits in Equation 4.19. Correspondingly, Equation 4.17, 4.18 and 4.19 can be modified to their corresponding symbol-based format, yielding:

$$M_{v->c_j}(s_i) = \alpha M_{r->v} \prod_{c_k \in C(b_i), k \neq j} M_{c_k->v}(s_i), \quad (4.21)$$

$$M_{apo}(s_i) = \alpha M_{r->v}(s_i) \prod_{c_k \in C(s_i)} M_{c_k->v}(s_i), \quad (4.22)$$

$$M_{v->r}(s_i) = \alpha \prod_{c_k \in C(b_i)} M_{c_k->v}(s_i). \quad (4.23)$$

Since the non-binary LDPC decoder introduced in Section 4.3 operates using symbol probabilities, thus by specifically choosing a decoding field for the non-binary LDPC code, which matches the modulation scheme used, facilitates purely symbol-based message passing. More explicitly, we can use for example 4QAM together with non-binary LDPC decoding over GF(4), or employing 16QAM modulation scheme, while choosing GF(16) for the non-binary LDPC.

4.7 Performance of the binary LDPC-aided space-time codec

In this section, the achievable performance of the jointly decoded LDPC-aided space-time codec will be evaluated. Alamouti's popular $\mathbf{G}_2$ space-time code will be used as a benchmarker both with and without being concatenated to other channel codecs. The $\mathbf{G}_2$ space-time code is a two-transmitter two-receiver scheme, using two time slots for transmitting two replicas of the original two bits. More explicitly, as seen at the left of Table 4.15, during the first time slot the two transmitter antennas emit two independent information symbols x_1 and x_2, while the modified replicas $-x_2^*$ and x_1^* of the two independent information symbols transmitted during the first time slot are sent during the second time slot. Hence during the period of two time slots, the space-time code $\mathbf{G}_2$ transmitted the two information symbols x_1 and x_2, yielding an effective throughput of 1 symbol/time slot. Similarly, as seen at the right of Table 4.15 for the case where the half-rate binary LDPC-aided space-time codec proposed by Meshkat and Jafarkhani [108] is applied, during the first time slot, the original information symbol x_1 is mapped onto the first antenna, while the corresponding parity symbol x_{p1} is mapped onto the second antenna and these two symbols are transmitted simultaneously during the first time slot. In the second time slot, another information symbol x_2 and another parity symbol x_{p2}

Time Slot	Antenna 1	Antenna 2
1	x_1	x_2
2	$-x_2^*$	x_1^*

Time Slot	Antenna 1	Antenna 2
1	x_1	x_2
2	x_3	x_4

Time Slot	Antenna 1	Antenna 2
1	x_1	x_{p1}
2	x_2	x_{p2}

Table 4.15: Comparison of the effective throughput of the $\mathbf{G}_2$ code (left) and the LDPC-aided space-time codec (right), both of which correspond to 1 symbol/time slot. By contrast, the scheme characterised in the middle achieves an effective throughput of 2 symbols/time slot at the cost of attaining no diversity gain.

are mapped onto the first and the second antenna, respectively. Thus, the half-rate binary LDPC-aided space-time codec achieves the same effective throughput as Alamouti's space-time $\mathbf{G}_2$ code, although owing to the employment of powerful LDPC coding, rather than simple repetition coding, a better BER performance is expected. Hence for the half-rate binary LDPC-aided space-time codec, the effective throughput can be calculated as:

$$\begin{aligned} throughput\ (bps) &= channel\_code\_rate \\ &\times number\_of\_transmitter\_antennas \\ &\times modulator's\_bit\_per\_symbol. \end{aligned} \tag{4.24}$$

More explicitly, the throughput of the two schemes may be compared in Table 4.15 as follows. The $r = 1/2$-rate, repetition-coding, based $\mathbf{G}_2$ scheme characterised at the left transmits a total of two independent symbols, namely x_1 and x_2 in two time-slots using two antennas. Similarly, when considering two consecutive time-slots rather than a single one, the $r = 1/2$-rate LDPC-coded space-time codec featured at the right of Table 4.15 also transmits two independent symbols, namely x_1 and x_2 in addition to the LDPC parity bits/symbols denoted by x_{p1} and x_{p2} at the right of Table 4.15.

By contrast, the scheme characterised in the middle of Table 4.15 represents a simple system, where no channel codec is used. In this case, the transmitter may transmit four independent source symbols during the two consecutive time slots with the aid of two transmitter antennas, which yields a doubled throughput in comparison to the other two schemes characterised at the left and right of Table 4.15. However, this scheme increases the effective throughput at the cost of surrendering diversity gain. Since the four symbols transmitted during the two time slots are independent, thus the performance of this scheme will be similar to the uncoded single transmitter scenario.

4.7.1 Effects of increasing the number of joint detection iterations

As in all other iterative detection schemes, extra performance gains may be achieved with the aid of carrying out an increased number of iterations, i.e. at the cost of an increased complexity. Therefore we will gradually increase the number of joint detection iterations in the context of a two-transmitter two-receiver system utilising a (1500, 750) regular-construction binary LDPC code having an average column weight of 2.5.

As seen from Figure 4.17, the achievable performance of the system significantly improves when the number of iterations is increased from one to two, although the further incremental performance gains

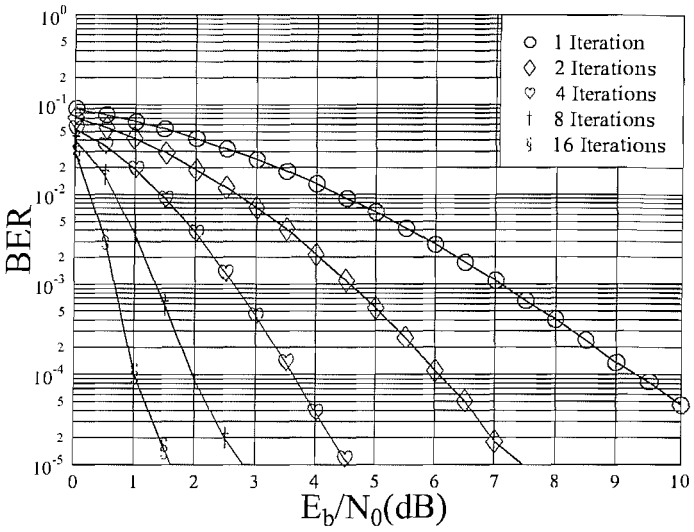


Figure 4.17: BER performance of a two-transmitter two-receiver LDPC-aided space-time codec using (1500, 750) binary LDPC codes having an average column weight of 2.5, when communicating over an uncorrelated Rayleigh fading channel. The number of joint detection iterations used was one, two, four and eight.

gradually erode upon further increasing the number of iterations. Thus, in our later experiments four iterations will be used in order to achieve a relatively good performance without excessively increasing the complexity of the decoder.

4.7.2 Effects of increasing the number of transmission antennas

In space-time coding, typically several antennas are used both at the transmitter as well as at the receiver. In this section, we will evaluate the achievable performance, when the number of transmitters is increased from one to four, while the number of antennas at the receiver was fixed to two. Following these investigations, the number of transmission antennas will be fixed to two, while the number of receiver antenna will be varied from one to four. The component LDPC code used was the same as the one employed in Section 4.7.1. A binary (1500, 750) LDPC code having an average column weight of 2.5 is used. The corresponding simulation parameters are summarised in Table 4.16.

In Figure 4.18 we can see that when a higher number of transmitters is used, the attainable performance slightly degrades. This is because when a higher number of transmitters is used in Figure 4.18, more transmitters antennas' signal will interfere upon arriving at each receiver antenna, which may result in decision conflicts at the output of the LDPC decoder. Fortunately, owing to the benefits of the iterative detection process, the BER degradation imposed is not significant, despite having an increased effective throughput of 0.5, 1, 1.5 and 2 bps in case of $n = 1, 2, 3, 4$, respectively. Another factor resulting in a degraded performance for $n > 2$ is that for the sake of fair comparison, the total transmit power has to be constant, resulting in a reduced transmit power for each antenna.

Performance using various number of transmitter antennas				
Modulation Scheme	Channel Codec	Tx. No.	Rx. No.	Throughput (symbol per time slot)
BPSK	LDPC(1500, 750)	1	2	0.5
		2	2	1
		3	2	1.5
		4	2	2
Performance using various number of receiver antennas				
Modulation Scheme	Channel Codec	Tx. No.	Rx. No.	Throughput (symbol per time slot)
BPSK	LDPC(1500, 750)	2	1	1
		2	2	1
		2	3	1
		2	4	1

Table 4.16: Simulation parameters of the LDPC-aided space-time codec using different configurations, where the number of transmitter antennas and receiver antennas is varied.

Hence the BER curves recorded on an SNR scale have to be shifted to the right in Figure 4.18 on an E_b/N_0 scale.

Figure 4.19 illustrates the associated performance trends for the proposed system, when the number of receivers, rather than transmitters, is increased. In this scenario, since the number of transmitters was fixed to two, the system's effective throughput is maintained at 1bps. Since an increased number of receivers was employed, more copies of the original transmitted signal were captured by the multiple receivers. These replicas of the original signal propagated through different transmission paths, thus even if some replicas were severely corrupted owing to encountering channel fades, the less corrupted signals were still able to provide reliable information for the LDPC decoder. Therefore, the achievable performance was significantly improved upon using a higher number of receivers in the LDPC-aided space-time codec.

4.7.3 Performance of the binary LDPC-aided space-time codec

In this section, the performance of the proposed binary LDPC-aided space-time codec will be studied in conjunction with multilevel modulation for the sake of achieving an increased effective throughput. The space-time code $\mathbf{G}_2$ will be used as a benchmarker. The $\mathbf{G}_2$ code will also be concatenated with an LDPC code having various code rates for the sake of achieving the same effective throughput as the binary LDPC-aided space-time codec studied. For the sake of comparing the performance of the LDPC-ST scheme and the LDPC-encoded $\mathbf{G}_2$ scheme, both arrangements employed four iterations. Additionally, since a turbo decoding iteration has a similar complexity to eight LDPC iterations when using a constraint length four component RSC code [24], we also plotted the corresponding performance curves, when the TC concatenated with the $\mathbf{G}_2$ scheme invoked eight iterations. The LDPC-encoded $\mathbf{G}_2$ scheme is also characterised, when using 64 LDPC iterations. The latter two benchmarkers have a significantly higher complexity, than the LDPC-ST scheme and the LDPC-encoded $\mathbf{G}_2$ scheme using

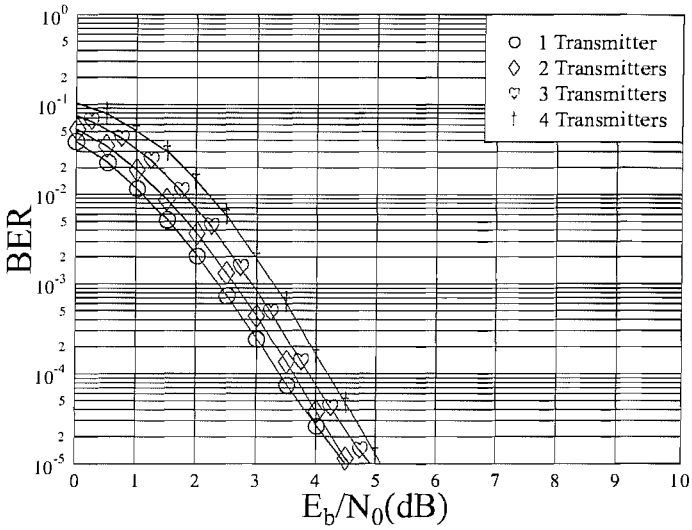


Figure 4.18: BER performance of an n-transmitter two-receiver LDPC-aided space-time codec using (1500, 750) binary LDPC codes having an average column weight of 2.5, when communicating over an uncorrelated Rayleigh fading channel employing BPSK modulation. Four joint detection iterations were used and we had $n = 1, 2, 3, 4$ transmitters. The effective throughput is 1, 2, 3, 4 bps, respectively, this is why the required E_b/N_0 was increased.

4 iterations. However, these two high-complexity benchmarkers also provide interesting insights. The stand-alone effective throughput of these two schemes is different, even though they were both configured as two-transmitter two-receiver schemes. More explicitly, the $r = 1/2$-rate repetition-code-like $\mathbf{G}_2$ code emits for example two QPSK-modulated data symbols from the two transmitter antennas within one of the two time slots, while in the second time slot essentially the inverted and conjugated replicas of the original two data symbols are transmitted. Since essentially the same information has been transmitted twice using a 'code' reminiscent of repetition coding upon utilising the $\mathbf{G}_2$ code, the effective throughput of the $\mathbf{G}_2$ space-time code is reduced by a factor of two, resulting an effective throughput of one QPSK symbol per $\mathbf{G}_2$-coded time-slot. Similarly, the $r = 1/2$-rate binary LDPC-aided space-time codec transmits two original uncoded QPSK-modulated data symbols within one time slot as seen in Table 4.15. Thus the uncoded effective throughput for the space-time codec proposed by Meshkat without being concatenated with any channel codec will be doubled in comparison to that of Alamouti's space-time $\mathbf{G}_2$ code. The associated simulation parameters are summarised in Table 4.17. The turbo convolutional code concatenated with the space-time $\mathbf{G}_2$ code has a constraint length of four, and the turbo decoder employed eight iterations. The puncturing patterns listed in Table 2.24 were used for achieving various code rates for the sake of appropriately adjusting the rate of the turbo convolutional code. All LDPC codes characterised in Table 4.17 used an average column weight of 2.5 and employed decoding over the binary Galois field. The number of joint detection iterations used by the binary LDPC-aided space-time codec was four.

We can observe in Figure 4.20 that when a low throughput is desired, the LDPC-coded $\mathbf{G}_2$ scheme

	Scheme	Tx. No.	Rx. No.	Channel Codec	Modem	Figure Number
1 bps	LDPC-ST	2	2	LDPC(1500, 750), $r = 1/2$	BPSK	4.20
	LDPC-ST	4	2	LDPC(1500, 375), $r = 1/4$	BPSK	
	$\mathbf{G}_2$	2	2	LDPC(1500,750)	QPSK	
	$\mathbf{G}_2$	2	2	TC, $r = 1/2$	QPSK	
	$\mathbf{G}_2$	2	2	N/A	BPSK	
2 bps	LDPC-ST	2	2	LDPC(1500, 750), $r = 1/2$	QPSK	4.21
	LDPC-ST	4	2	LDPC(1504, 376), $r = 1/4$	QPSK	
	LDPC-ST	4	2	LDPC(1500, 750), $r = 1/2$	BPSK	
	LDPC-ST	3	2	LDPC(1500, 1000), $r = 2/3$	BPSK	
	LDPC-ST	3	2	LDPC(1500, 500), $r = 1/3$	QPSK	
	LDPC-ST	2	2	LDPC(1504, 376), $r = 1/4$	16QAM	
	$\mathbf{G}_2$	2	2	LDPC(1504,752)	16QAM	
	$\mathbf{G}_2$	2	2	LDPC(1500,1000)	8PSK	
	$\mathbf{G}_2$	2	2	TC, $r = 1/2$	16QAM	
	$\mathbf{G}_2$	2	2	TC, $r = 2/3$	8PSK	
	$\mathbf{G}_2$	2	2	N/A	QPSK	
3 bps	LDPC-ST	3	2	LDPC(1500, 750), $r = 1/2$	QPSK	4.22
	LDPC-ST	2	2	LDPC(1500, 750), $r = 1/2$	8PSK	
	$\mathbf{G}_2$	2	2	N/A	8PSK	
	$\mathbf{G}_2$	2	2	LDPC(1500,1128)	16QAM	
	$\mathbf{G}_2$	2	2	TC, $r = 3/4$	16QAM	
4 bps	LDPC-ST	2	2	LDPC(1504, 752), $r = 1/2$	16QAM	4.23
	LDPC-ST	4	2	LDPC(1504, 752), $r = 1/2$	QPSK	
	$\mathbf{G}_2$	2	2	N/A	16QAM	

Table 4.17: Simulation parameters for the various space-time codecs employing multilevel modulator for achieving an effective throughput of 1, 2, 3 and 4 bits per symbol. The $\mathbf{G}_2$ space-time codec was concatenated with turbo convolutional codes having various code rates and using different modulation schemes for attaining the same effective throughput as the binary LDPC-aided space-time codec. It is worth noting that Alamouti's $\mathbf{G}_2$ space-time block code essentially uses a 'repetition-code-like' transmit diversity regime, which requires the employment of additional turbo convolutional code for achieving a low BER. By contrast, the benefit of the propose LDPC-ST scheme is that it inherently incorporates half-rate LDPC coding and hence no additional channel coding is required. This allows us to maintain a factor two higher effective throughput than that of the half-rate turbo-coded $\mathbf{G}_2$ space-time block code.

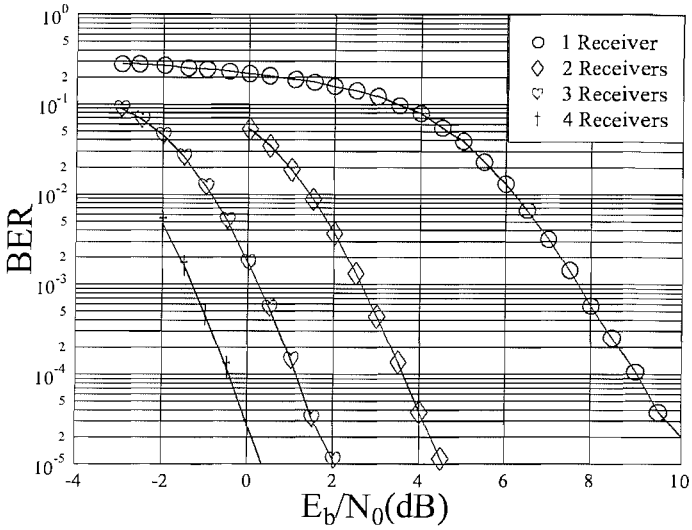


Figure 4.19: BER performance of a two-transmitter n-receiver LDPC-aided space-time codec using (1500, 750) binary LDPC codes having an average column weight of 2.5, when communicating over an uncorrelated Rayleigh fading channel employing BPSK modulation. Four joint detection iterations was used and we had $n = 1,\ 2,\ 3,\ 4$ receivers. The effective throughput is maintained at 1 bps.

invoking four iterations achieved the best performance, excluding the two high-complexity benchmarkers. When the throughput is increased to 2 bps as in Figure 4.21, the performance degradation of the low-complexity four-iteration MIMO schemes compared to the high-complexity benchmarkers was reduced in comparison to that of the 1 bps scenario characterised in Figure 4.20. Furthermore, we observed that the LDPC-ST scheme performed approximately 1 dB better than the LDPC-coded $\mathbf{G}_2$ scheme at a BER of 10^{-4}. When the throughput is further increased to 3 bps, as in Figure 4.22, the high-complexity scheme hardly achieved any benefit compared to the low-complexity scheme and the LDPC-ST scheme using a half-rate LDPC code transmitting over three antennas using QPSK modulation achieved the best performance in the low-complexity category. Finally, in the 4 bps throughput scenario, there is no attractive schemes for concatenation with the $\mathbf{G}_2$ scheme. Although it is possible to extend the constellation to 64QAM while using a two-third rate channel code, it has been observed in Figures 4.20 to 4.23 that a high number of modulation levels will typically lead to a poor performance. Therefore, the four-transmitter LDPC-ST which refrains from using separate channel coding is the optimal solution for achieving a throughput of 4 bps.

4.8 Performance of the non-binary LDPC-aided space-time codec

In this section, the performance of the symbol-based non-binary LDPC-aided space-time codec described in Section 4.6 will be compared to that of the bit-based system proposed by Meshkat and Jafarkhani [108].

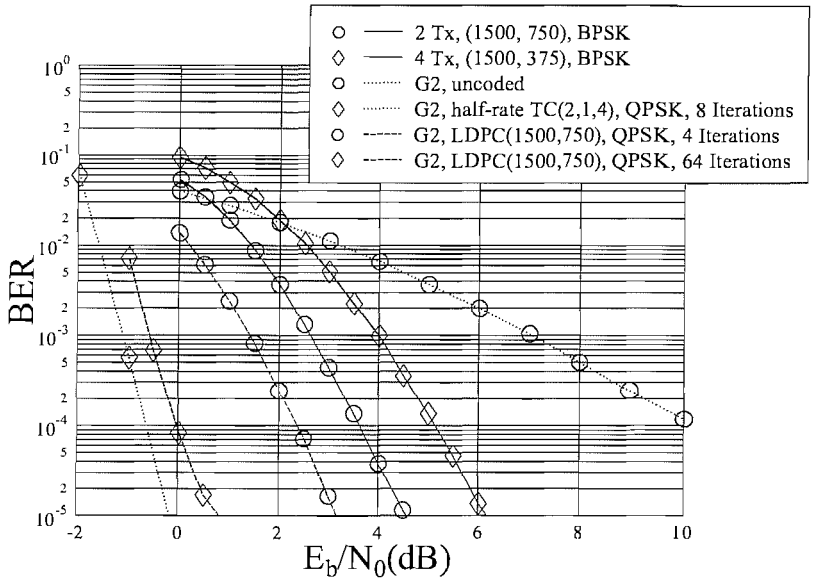


Figure 4.20: BER performance of the various space-time coding configurations summarised in Table 4.17 designed for achieving an effective throughput of 1 bps, when communicating over an uncorrelated Rayleigh fading channel. The achievable coding gain of the various schemes at a BER of 10^{-4} will be summarised in Table 4.23.

In order to achieve an identical effective throughput of two, three and four bits per symbol, the parameters of the symbol-based non-binary LDPC-aided space-time codec were selected as summarised in Table 4.18, benchmarked against the bit-based system having an identical modulation mode. The non-binary LDPC codes described in Section 4.3 will be used in the symbol-based MIMO system for the sake of providing a purely symbol-based message passing mechanism.

The performance of the system is characterised in Figure 4.24. Observe that the performance of the symbol-based system incorporating the non-binary LDPC codes is superior in comparison to the bit-based MIMO system using a binary LDPC decoder. As seen in Figure 4.8 of Section 4.4.2, if the size of the decoding Galois field is increased, while the size of the equivalent binary PCM $\mathbf{H}_b$ is maintained and when an average column weight of 2.5 is used, a larger Galois field will render the LDPC codes less prone to transmission errors. Thus the superior performance of the symbol-based system is essentially the consequence of employing of a stronger channel codec. Before concluding this chapter, we will now show that when using the symbol-based system, the associated decoding complexity can be significantly reduced in conjunction with a higher number of modulation levels or a higher number of transmitters.

4.9 Implementational complexity

In this section we will demonstrate that the symbol-based non-binary LDPC-aided space-time codec constitutes a solution imposing a significantly lower decoding complexity than its bit-based LDPC-

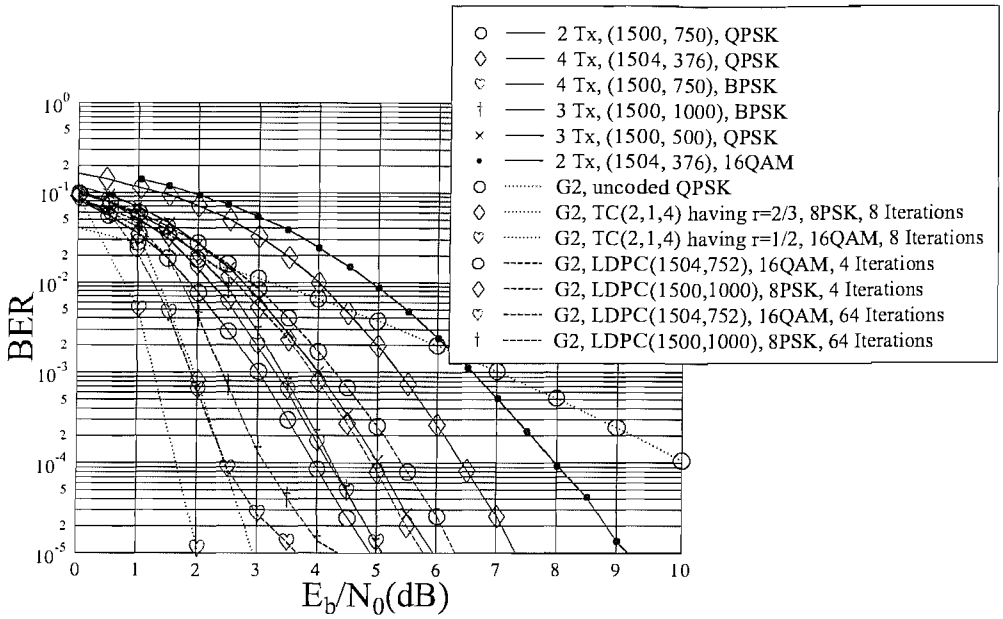


Figure 4.21: BER performance of the various space-time coding configurations summarised in Table 4.17 designed for achieving an effective throughput of 2 bps, when communicating over an uncorrelated Rayleigh fading channel. The achievable coding gain of the various schemes at a BER of 10^{-4} will be summarised in Table 4.23.

aided space-time codec counterpart.

Recall from Section 4.5 related to the bit-based system and from Section 4.6 on the symbol-based system, the Equation 4.16 and Equation 4.20 defined the bit-based and symbol-based joint detection approach, respectively. For the bit-based joint detection procedure using Equation 4.16, the decoding operations require the evaluation of all possible input symbol configurations containing the k^{th} bit of the original space-time coded codeword, as well as the calculation of the *a priori probability* provided by the neighbouring bits. Hence, for a system having n_t transmitters and bps number of bits per symbol, the total number of metric evaluations using Equation 4.16 will be $2^{bps \cdot n_t}$. Each metric evaluation requires $n_t \times n_r$ multiplications for determining $\mathbf{HS}(B_i, b_k)$ in Equation 4.16, one multiplication for evaluating the square, and one for carrying out the required division. One subtraction is needed for finding the Euclidean distance between the received sample and each of the constellation points. Furthermore, for each metric evaluation, $(bps \times n_t - 1)$ multiplications are needed for calculating the *a priori probability*. Thus, for each decoded bit, the required number of multiplications becomes $((bps \times n_t - 1) + (n_t \times n_r + 2)) \times 2^{bps \times n_t} = (n_t \times (bps + n_r) + 1) \times 2^{bps \times n_t}$. The required number of additions is $2^{bps \times n_t}$.

By contrast, for the proposed non-binary system using Equation 4.20, the number of multiplications needed for the *a priori probability* calculation is reduced to $(n_t - 1)$, since we are directly determining the symbol probability. Thus the total number of multiplications per bit for the symbol based system is $(n_t \times (n_r + 1) + 1) \times 2^{bps \times n_t}/bps$ and the number of additions becomes $2^{bps \times n_t}/bps$ per bit.

On the other hand, upon employing non-binary LDPC codes in the symbol-based approach, the

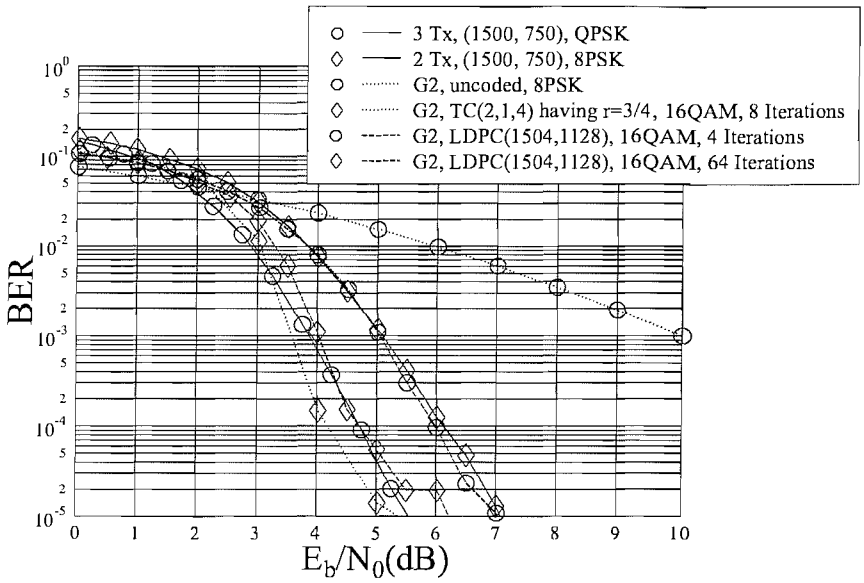


Figure 4.22: BER performance of the various space-time coding configurations summarised in Table 4.17 designed for achieving an effective throughput of 3 bps, when communicating over an uncorrelated Rayleigh fading channel. The achievable coding gain of the various schemes at a BER of 10^{-4} will be summarised in Table 4.23.

message passing between the system components becomes implementationally more complex owing to the increased GF size. The number of multiplications and additions required for each coded bit can be represented as $2w_c q$ additions and $7w_c q/log_2(q)$, as detailed in Section 4.3.5, where w_c and q are the LDPC code's column weight and the LDPC decoding field size, respectively.

Hence the overall complexity imposed by the two systems in each of the modulation schemes is listed in Table 4.19, and plotted in Figure 4.25.

4.10 Summary and conclusion

In this chapter, the family of non-binary LDPCCs proposed by Davey and MacKay [54] [56] was introduced in Section 4.3 and the achievable performance of various non-binary LDPCCs has been studied as a function of the code rate, using two different average column weights, namely 2.5 and three. When the size of the non-binary PCM $\mathbf{H}_q$ is fixed, the simulation results provided in Section 4.4.1 showed that the performance of the non-binary LDPCCs improves upon increasing the decoding field size. By contrast, when the size of the equivalent binary PCM $\mathbf{H}_b$ was fixed, as in Section 4.4.2, and when an average column weight of 2.5 was applied, the attainable performance of the non-binary LDPCCs improved upon increasing the size of the decoding field. However, when an average column weight of three was used, the advantages and disadvantages of using a larger decoding field were less pronounced, and using non-binary LDPCCs operating in a large **GF** were not always beneficial. During the complexity discussion of Section 4.3.5 related to non-binary LDPCCs we found that using **GF**(4)

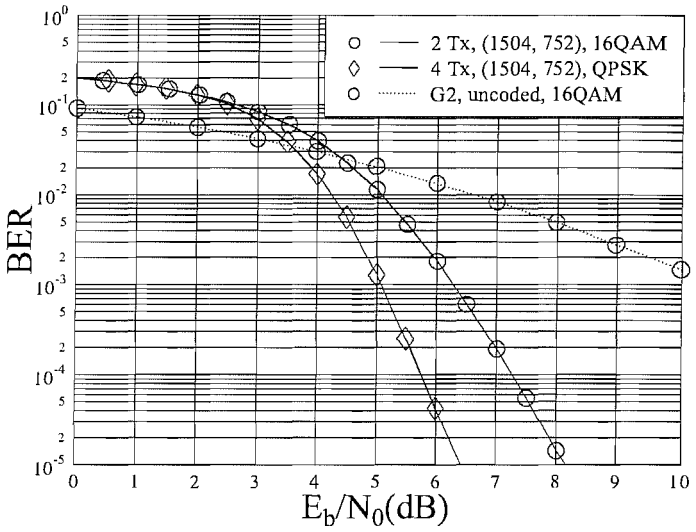


Figure 4.23: BER performance of the various space-time coding configurations summarised in Table 4.17 designed for achieving an effective throughput of 4 bps, when communicating over an uncorrelated Rayleigh fading channel. The achievable coding gain of the various schemes at a BER of 10^{-4} will be summarised in Table 4.23.

strikes a good compromise between the attainable performance gain and the associated decoding complexity.

The binary LDPC-aided joint space-time detection scheme proposed by Meshkat and Jafarkhani [108] was described in Section 4.5. This scheme has the drawback that an assumption was made that the adjacent bits in a constellation point are independent, which is not exactly true in the context of Gray mapping. Furthermore, the detection complexity increases exponentially with the number of antennas and with the number of modulation levels. Therefore, this scheme was further developed to create a novel purely symbol-based detection scheme in Section 4.6, where the non-binary LDPC code was embedded into the system for the sake of providing a purely symbol-based message exchanging mechanism. The attainable performance was evaluated for the bit-based system in Section 4.7 using various number of iterations, transmitters and receivers. Similarly, the performance of the symbol-based system was studied in Section 4.8 and appeared to be superior in comparison to the bit-based system owing to the employment of higher-performance non-binary LDPCCs. The major advantage of applying the symbol-based system over the bit-based scheme was highlighted in Section 4.9, where the complexity of the two schemes was compared. More explicitly, it appears that by using the symbol-based scheme proposed, the number of arithmetic operations may be significantly reduced, especially for the more complex scenarios using high-order phaser constellations.

A coding gain summary is provided in Tables 4.20 to 4.23 for the sake of characterising the attainable BER performance of the non-binary LDPC code considered and that of the LDPC-ST scheme. As seen in Table 4.20, upon increasing the size of the decoding Galois field while maintaining

Throughput	Bit-based	Symbol-based
2 bps	(1500, 750)	$(750, 375)_q$
QPSK	GF(2)	GF(4)
3 bps	(1500, 750)	$(500, 250)_q$
8PSK	GF(2)	GF(8)
4 bps	(1496, 748)	$(376, 188)_q$
16QAM	GF(2)	GF(16)
Average LDPC Column Weight		2.5
Number of Iterations		5
Number of Transmitters		2
Number of Receivers		2
Channel		Uncorrelated Rayleigh fading
LDPC Coded Blocklength		1500 bits(approximately)

Table 4.18: Simulation parameters for the bit-based and symbol-based LDPC-aided space-time codec utilising both binary and non-binary LDPC codes, when communicating over an uncorrelated Rayleigh fading channel.

Multiplications	Bit-based	Symbol-based
QPSK	179	91
8PSK	739	198
16QAM	3338	518

Additions	bit based	symbol based
QPSK	27	28
8PSK	75	61
16QAM	267	144

Table 4.19: Complexity comparison between the bit-based LDPC-aided space-time codec of [108] and the symbol-based algorithm characterised in Figure 4.24 using the simulation parameters listed in Table 4.18.

the same non-binary PCM size quantified in terms of the number of non-binary symbols, an improved BER performance may be achieved using a higher-order decoding field. However when the size of the binary equivalent PCM is fixed, decoding over higher order Galois field is not always beneficial. While using various column-weights for the PCM, a different performance trend may be observed, as seen in Table 4.21. Table 4.23 summarises the decoding performance of the novel LDPC-ST scheme and that of the benchmarking LDPC-coded $\mathbf{G}_2$ scheme. It has been observed that the LDPC-coded $\mathbf{G}_2$ scheme performs better than the LDPC-ST scheme, when the throughput is as low as 1 bps. However, when a higher throughput is desired, the LDPC-ST constitutes a superior solution compared to the LDPC-coded $\mathbf{G}_2$ scheme. Furthermore, the higher-complexity TC-coded $\mathbf{G}_2$ scheme and an LDPC-coded $\mathbf{G}_2$ arrangement are used as additional benchmarkers, while invoking a significantly higher number of iterations. It has been observed that the extra performance gain of these high-complexity schemes was reduced, when the throughput was increased.

Column weight	LDPC code	Coding gain(dB)			
		GF(2)	GF(4)	GF(8)	GF(16)
w_c=2.5	(500,250)	4.91	6.38	6.68	**6.82**
	$(1000, 500)_q$	5.47	6.47	6.94	**7.09**
	$(2000, 1000)_q$	6.03	6.82	7.12	**7.24**
w_c=3.0	$(500, 250)_q$	5.59	6.15	6.32	**6.38**
	$(1000, 500)_q$	6.03	6.53	6.59	**6.67**
	$(2000, 1000)_q$	6.41	6.73	6.77	**6.794**

Table 4.20: Coding gain of the non-binary LDPC codes specified in Table 4.11 at a BER of 10^{-4}, when communicating over an AWGN channel. The best scheme is highlighted using bold fonts.

Code rate	Column weight	Coding gain(dB)			
		GF(2)	GF(4)	GF(8)	GF(16)
r=1/4	w_c=2.5	6.79	7.13	**7.28**	**7.28**
	w_c=3.0	6.53	6.53	**6.41**	**6.41**
r=1/2	w_c=2.5	6.41	6.845	7.03	**7.09**
	w_c=3.0	6.66	**6.72**	6.66	6.66

Table 4.21: Coding gain of the non-binary LDPC codes specified in Table 4.12 at a BER of 10^{-4}, when communicating over an AWGN channel. The best scheme is highlighted using bold fonts.

Column weight	Code rate	Coding gain(dB)			
		GF(2)	GF(4)	GF(8)	GF(16)
w_c=2.5	$r = 1/3$	6.47	7.132	7.441	7.56
	$r = 1/2$	5.97	6.823	7.18	7.235
	$r = 2/3$	5.441	6.17	6.5	6.56
	$r = 3/4$	5.004	5.69	6	6.12
	$r = 4/5$	4.72	5.309	5.62	5.735
w_c=3.0	$r = 1/3$	6.412	6.81	6.81	6.86
	$r = 1/2$	6.412	6.735	6.78	6.81
	$r = 2/3$	5.912	6.17	6.323	6.323
	$r = 3/4$	5.44	5.794	5.882	5.882
	$r = 4/5$	5.103	5.412	5.53	5.53

Table 4.22: Coding gain of the non-binary LDPC codes specified in Table 4.13 at a BER of 10^{-4}, when communicating over an AWGN channel.

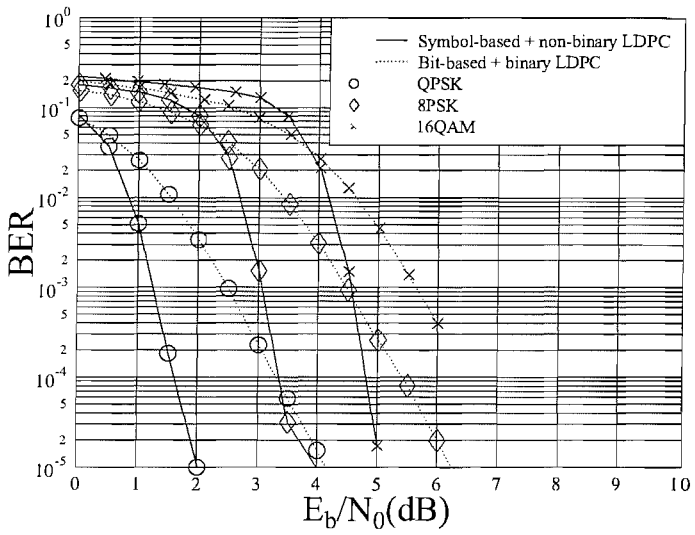


Figure 4.24: BER performance of the bit-based and symbol-based MIMO systems summarised in Table 4.18 utilising binary and non-binary LDPC codes, when communicating over an uncorrelated Rayleigh fading channel.

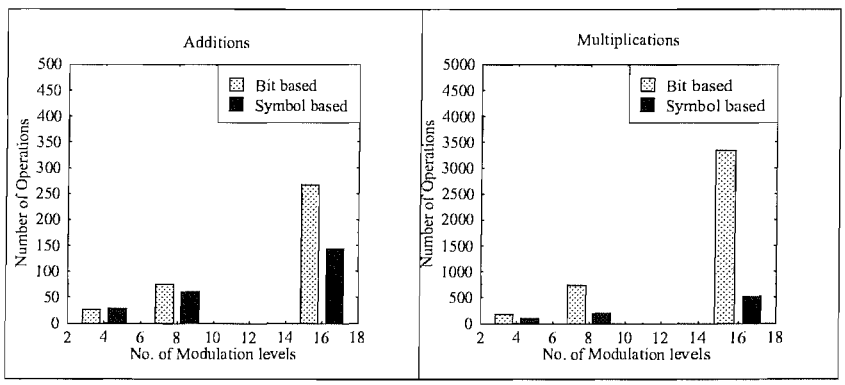


Figure 4.25: Complexity comparison between the bit-based LDPC-aided space-time codec of [108] and the symbol-based algorithm characterised in Figure 4.24 using the simulation parameters listed in Table 4.18.

Throughput(bps)	Scheme				E_b/N_0(dB) required at BER=10^{-4}
	No. of transmitters	LDPC code	Modulation mode		
1 bps	2	(1500,750)	BPSK	LDPC-ST	3.63
	4	(1500,375)	BPSK	LDPC-ST	5.185
	2	(1500,750)	QPSK	$\mathbf{G}_2$	**2.357**
2 bps	2	(1500,750)	QPSK	LDPC-ST	**3.94**
	4	(1504,376)	QPSK	LDPC-ST	6.437
	4	(1500,750)	BPSK	LDPC-ST	4.19
	3	(1500,1000)	BPSK	LDPC-ST	4.25
	3	(1500,500)	QPSK	LDPC-ST	5
	2	(1504,376)	16QAM	LDPC-ST	8
	2	(1504,752)	16QAM	$\mathbf{G}_2$	5.41
	2	(1500,1000)	8PSK	$\mathbf{G}_2$	4.88
3 bps	3	(1500,750)	QPSK	LDPC-ST	**4.75**
	2	(1500,750)	8PSK	LDPC-ST	6.12
	2	(1504,1128)	16QAM	$\mathbf{G}_2$	6
4 bps	2	(1504,752)	16QAM	LDPC-ST	7.25
	4	(1504,752)	QPSK	LDPC-ST	**5.75**

Table 4.23: E_b/N_0 values for the various LDPC-ST scheme parameterised in Table 4.17 required for the sake of achieving an effective system throughput of 1, 2, 3 and 4 bps at BER= 10^{-4}, respectively, when communicating over an uncorrelated Rayleigh fading channel. The best scheme is highlighted using bold fonts.

Chapter 5

Joint Source and Channel Coding Using Variable Length Codes

5.1 Historical perspective

Variable Length Codes (VLC) are widely used in both audio [136] and video compression schemes [109]. The conceptually simplest VLC philosophy is based on entropy-coding [67] [109], where a frequently encountered source symbol is assigned a short VLC symbol, while less frequent source symbols are encoded using longer VLC symbols. When using this procedure, the achievable average symbol length is typically reduced, provided that the original source symbols exhibit unequal probabilities on some degree. The well-known Huffman coding scheme [137] belongs to the family of VLC codes. Huffman codes are capable of achieving an average symbol length close to the entropy of the source symbol sequence. A lot of research attention has been devoted to efficiently decoding Huffman codes, as outlined for example in [138–143]. However, in the absence of correlation between the consecutive source symbols, Huffman codes may in fact result in an increased bit rate in some practical scenarios, where the symbols are nearly equi-probable. Furthermore, a disadvantage of Huffman codes is that they are vulnerable to transmission errors, especially when communicating over high-BER mobile radio channels. Note that in the presence of transmission errors a Huffman-coded symbol may be corrupted to another legitimate symbol having a different number of bits, which leads to synchronisation errors as well as to the loss of some bits and symbols. This problem is aggravated by the fact that this error event may remain undetected. This is because in Huffman codes no short symbol is allowed to constitute a prefix of any of the longer symbols for reasons of unique detection capability. More explicitly, the Huffman decoder immediately outputs a decoded symbol upon identifying its corresponding received bits and in the presence of transmission errors this could be incorrect. With the aim of mitigating the above-mentioned deficiency of Huffman codes, Reversible Variable Length Codes (RVLC) were proposed by Takishima *et al* [144], which are less efficient source coding schemes compared to Huffman codes. However, RVLCs have the capability of decoding the bit stream from either the beginning or the end, When the decoder has read a high number of bits without finding a valid codeword. The family of RVLCs can be categorised into two classes, symmetric and asymmetric RVLCs. An RVLC

is defined as a symmetric code, if all the VLC symbols [1] have a symmetric bit pattern, such as the VLC symbols of 010 or 1001. By contrast, the VLC symbols 01, 10 belong to an asymmetric RVLC. Different methods have been applied for constructing the code table of an RVLC [145–148] designed for different applications. RVLCs typically have a slightly higher average symbol length compared to Huffman codes, but in exchange for this they offer a better error detection and correction capability. Hence if an error is detected, the decoder can start decoding the codewords from the other end of the bit stream for the sake of minimising the chance of encountering a decoding failure. As further design alternatives, Variable Length Error Correcting (VLEC) codes were proposed by Buttigieg and Farrell [149], which were also studied by Lamy [150,151]. VLECs have an even higher average symbol length than RVLCs, with the added benefit that they exhibit an error correction capability similar to classic block and convolutional codes, except that the length of the VLEC codes is not constant. Generally the VLC symbols are prone to channel impairment and hence powerful error correction methods have to be applied for the protection of the vulnerable VLC symbols. By invoking the classic MAP algorithm [152], originally devised by Bahl *et. al*, Bauer and Hagenauer [153,154] proposed two powerful VLC detection schemes by exploiting the residual redundancy found in the VLC. There are other soft information based decoding techniques designed for VLCs such as those in [155] and [156]. The soft-in soft-out (SISO) decoder principle of [152] can be used for constructing an iterative joint decoding system, such as those proposed in [157–160], exploiting the residual redundancy left in the source-encoded sequence, owing to a sub-optimum source encoder failing to reach the lowest possible source coded rate(bounded by the entropy). Furthermore, this source-related extrinsic information may also be exchanged with the deliberately introduced extrinsic information of the channel codec.

5.2 Decoding of variable length codes

As stated previously, VLCs are vulnerable to transmission errors encountered in wireless channels. In order to enhance the robustness of the VLC decoder, the idea of representing a variable length code using a tree structure was proposed by Buttigieg and Farrell in [161]. In [162,163], the performance of the stand-alone VLEC code was demonstrated to be better than that of a cascaded Huffman/Hamming code. Based on the trellis representation of the VLC codes as proposed by Buttigieg and Farrell, Bauer and Hagenauer [153] [154] further developed the Maximum A Posteriori (MAP) algorithm [152] for decoding VLCs transmitted over AWGN channels. *This was based on recognising the plausible fact that any encoder, which has some grade of redundancy, predictability or unequal probability of occurrence for the bits at its output may be viewed as an error correction encoder.*

5.2.1 Symbol based decoding of VLCs

This section will describe the symbol based trellis decoding of VLCs, following the approach of [154]. Assuming we have a variable length code C and a discrete $T-ary$ random variable $U = \{0, ..T-1\}$, each symbol u of the set U is mapped to a binary symbol of the VLC C, where the mapping is denoted as $c(u)$. The maximum and minimum symbol length is represented as l_{max} and l_{min}, respectively.

[1] In this chapter, for a non-binary VLC source symbol we will use the terminology of *source symbol*, while the binary representation of a VLC source symbol after encoding by a VLC encoder is referred to as *VLC symbol*.

There are O source symbols in a source-symbol packet processed by the decoder, i.e. we have $\mathbf{u} = (u_1, u_2, ...u_O)$. After mapping the $T-ary$ source symbol to the VLC, a VLC-encoded bit stream of length W is obtained, which is denoted as $\mathbf{b} = (b_1, b_2, ...b_W)$. The j^{th} bit of the o^{th} VLC-encoded source symbol is denoted as $C_{o,j}$. At the output of the channel the noisy sequence $y = (y_1, y_2...y_W)$ is received by the VLC decoder from the demodulator.

When we have a ternary source symbol associated with three legitimate values, the VLC encoder may map these three symbols to $c(0) = 1$, $c(1) = 01$, $c(2) = 00$. The source-symbol packet size is chosen to be $O = 4$, thus for example a source symbol sequence of $u = (0, 2, 0, 1)$ may be transmitted.

The four source symbols are then mapped by the VLC encoder to the binary representation of the VLC symbols given by (1,00,1,01). Hence we have a bit sequence of length 6, where the knowledge of the length of the symbol sequence and that of the length of the bit sequence can be used as side-information for constructing the trellis describing the operation of the VLC encoder analogously to that of a classic convolutional encoder, as seen in Figure 5.1. Naturally, the same trellis is used for the MAP decoding of the VLC symbols.

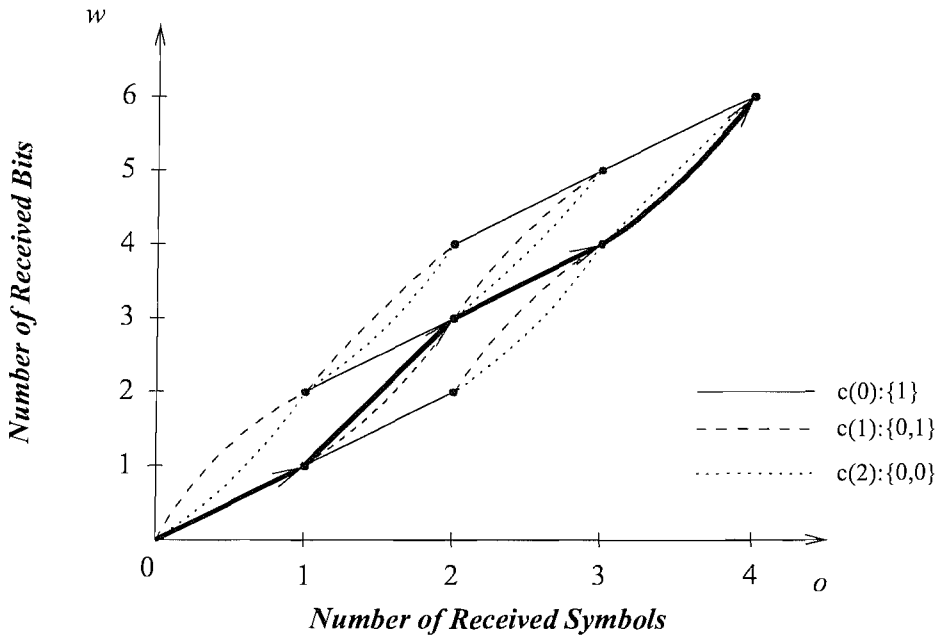


Figure 5.1: Trellis structure based on the received number of VLC-encoded symbols and bits

In Figure 5.1, the horizontal axis represents the number of VLC symbols that have been received, while the vertical axis represents the number of received bits. The bold line corresponds to the trellis path associated with the given input symbol sequence, and the remaining trellis transitions of Figure 5.1 represent all the other possible transitions incurred by all the legitimate input VLC symbols. In Figure 5.1, the transition engendered by the VLC symbols $c(0)$, $c(1)$ and $c(2)$ are represented by the bold lines, dashed lines and dotted lines, respectively. Commencing from the origin and by interpreting the first VLC symbol of $c(0)$, a binary 1 is deemed to have been received. Thus the bold transition emerges from the origin to the trellis state (1,1). The next step along the bold path of transitions is engendered by the VLC symbol $c(2)$. Since the VLC symbol $c(2)$ comprises 2 bits, thus a total of 3 binary bits are now received, hence the associated trellis transition evolves to state (2,3) in Figure 5.1.

However, all the legitimate trellis paths have to merge in state (4,6) eventually, i.e. in the state (O, W).

Observe that at trellis states (2,4) and (3,5), there is only a single transition emerging from these two states. This is because these two states are constrained by the total number of VLC-encoded symbols O and the total number of bits W representing the O symbols. More explicitly, in state (2,4) there are two remaining VLC-encoded symbols to be detected and there are also only two remaining received bits to be interpreted. Thus the only possible VLC symbol at this state will be symbol $c(0)$, which has the minimum symbol length of 1. The detection of all other VLC symbols arising from this state will be forbidden. Upon using the relation of

$$v = w - (o \cdot l_{min}), \tag{5.1}$$

the trellis seen in Figure 5.1 may be transformed to the equivalent form seen in Figure 5.2, where the variables v, n and k in Equation 5.1 represent the trellis state index, the number of received bits and the number of received symbols, respectively. The maximum number of trellis states seen in Figure 5.2 may be calculated by

$$v_{max} = W - (O \cdot l_{min}) + 1, \tag{5.2}$$

given the *a priori* knowledge that O VLC-encoded symbols and W bits are transmitted.

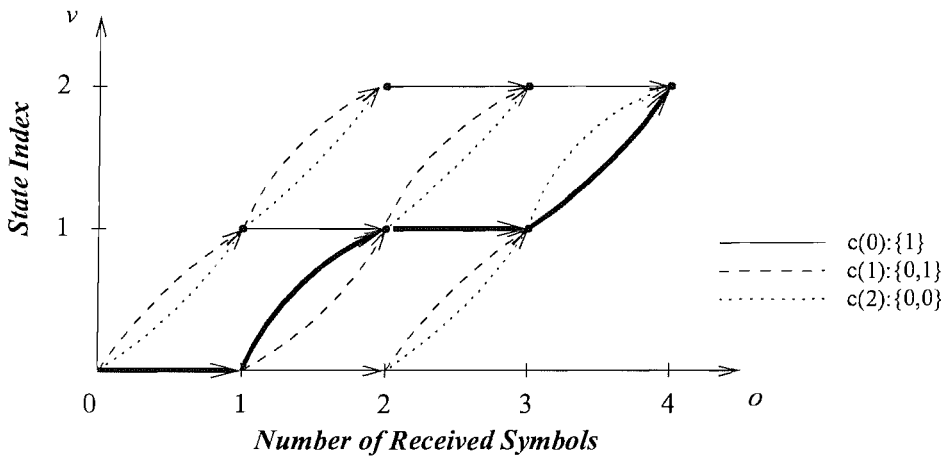


Figure 5.2: Modified trellis structure based on the received number of VLC-encoded symbols and bits

Hence Equations 5.1 and 5.2 may be used jointly for detecting the occurrence of illegitimate input symbols emerging from a given trellis state. For example, state (2,4) in Figure 5.1 corresponds to the trellis state (2,2) in Figure 5.2. Given the coordinates of the trellis state, with the aid of Equation 5.1 and Figure 5.1, it is possible to infer that a total of 4 bits have been received. Therefore, if the VLC-encoded symbol $c(1)$ is deemed to have been received based on arriving in this state, the next trellis state to be encountered may be calculated as from Equation 5.1 as $(4+2) - 3 \cdot 1 = 3$, which is a higher number than the maximum state index of 2 obtained by applying Equation 5.2 and therefore it is illegitimate. Thus there is only one legitimate transition from state (2,2), as seen in Figure 5.2.

To elaborate a little further by focusing our attention on the trellis section bounded by the states (2,3), (2,4), (3,4) and (3,5) in Figure 5.1, we have a more explicit view of the trellis transitions, as seen in Figure 5.3.

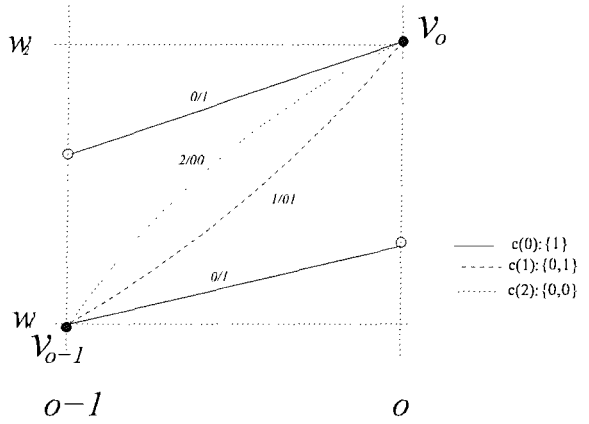


Figure 5.3: Trellis transitions determined by the states (2,3), (2,4), (3,4) and (3,5) in Figure 5.1

Explicitly, the numbers printed next to the transitions indicate the VLC-encoded symbols $c(0)$, $c(1)$ and $c(2)$ followed by the corresponding binary representation of the VLC symbol. Note that parallel transitions exist in the trellises of VLCs, as long as there are several VLC-encoded symbols having the same number of bits. This trellis transition diagram will be used for highlighting how to invoke the MAP algorithm later in this section.

Based on the trellises seen in Figures 5.1 and 5.2, various VLC decoding strategies may be implemented. In this thesis, the implementation of the popular Maximum-A-Posteriori (MAP) algorithm is described. The MAP algorithm was proposed by Bahl, Cocke, Jelinek and Raviv in 1974 [152] for estimating the *a posteriori* probabilities (APP) of the various states and the state transitions of a Markov source, provided that the channel encountered is memoryless. This algorithm is also often referred to as the BCJR algorithm after its inventors. This algorithm provides the estimated symbol sequence, based on maximising the APP of each symbol, which is essential for employment in any iterative scheme.

In the symbol-based MAP algorithm described here, the objective of the VLC decoder is that of determining the APP of the transmitted VLC symbol $u_o, (1 < o < O)$ from the soft output y_o of the channel, which is formulated as

$$P(\hat{u_o}|\mathbf{y}) = max\Big(P(u_o|\mathbf{y})\Big), \quad o = 1 \ldots O. \tag{5.3}$$

A tutorial on the MAP algorithm may be found for example in [115]. Suffice to say here that the MAP algorithm includes a forward recursion, a backward recursion and a symbol probability evaluation for the computation of the probabilities of the various VLC symbols associated with the trellis transitions. By extracting the trellis section of Figure 5.1 between $o = 1$ and $o = 3$, the corresponding application of the MAP algorithm can be illustrated with the aid of Figure 5.4.

The symbol probability

$$\gamma_i(y_{w'}^{w}, w', w), \tag{5.4}$$

quantifies the probability of a transition from state $v_i = w'$ to $v_{i+1} = w$, engendered by a received VLC-encoded symbol associated with a given soft channel output y_w. By applying the basic rules of

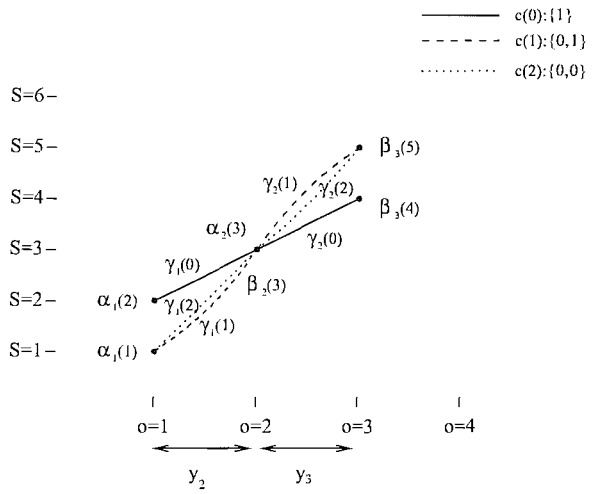


Figure 5.4: Recursive calculation of the α and β values for the trellis structure of Figure 5.1 between time instant $o = 1$ and $o = 3$.

conditional probabilities, γ_i can be split into three multiplicative factors as follows:

$$\gamma_i(y_{w'}^w, w', w) = q_o(u_o = i|w', w) \cdot p(y_{w'}^w|u_o = i) \cdot t_o(w|w'), \tag{5.5}$$

where $t_o(w|w')$ represents the state transition probability that given the current state w', the next state will be w. In Figure 5.3, the value of $t_{o-1}(w_1|w_2)$ is given by the sum of the probabilities of encountering symbol $c(2)$ and symbol $c(1)$, since both of these two symbols will engender the transition from w_1 to w_2. By contrast, if for example, w_2 is calculated to be an invalid state according to Equation 5.2, the probability value of t will be 0.

Upon returning to Equation 5.5, $q_o(u_o = i|w', w)$ is the probability that given the previous state w' and the current state w, how likely it is that this transition was engendered by the VLC-encoded symbol i. As seen in Figure 5.3, only symbol $c(1)$ and symbol $c(2)$ may engender the trellis transition from w_1 to w_2. Hence the probability $q_{o-1}(u_{o-1} = 1|w_1, w_2)$ is given by $\frac{P(u_o=1)}{P(u_o=1)+P(u_o=2)}$, which corresponds to the probability of occurrence of symbol $c(1)$ over the sum of the probability of symbol $c(1)$ and symbol $c(2)$.

Finally, $p(y_{w'}^w|u_o = i)$ in Equation 5.5 represents the probability of encountering the soft channel output associated with the transition $w' \rightarrow w$, given that the transmitted symbol was $u_o = i$. In the memoryless AWGN channel, this probability may be expressed as the product of the bitwise transition probabilities as follows:

$$p(y_{w'}^w|u_o = i) = \prod_{j=0}^{l(c_i)-1} p(y_{w'+j}|C_{o,j}), \tag{5.6}$$

where $C_{o,j}$ represents the j^{th} bit in the o^{th} VLC symbol's binary representation. After the calculation of γ according to Equation 5.5, we carry out the forward recursion for the sake of obtaining the values of α as follows:

$$\alpha_o(w) = P(v_o = w, \mathbf{y}_1^o), \tag{5.7}$$

where the variable α_o quantifies the joint probability of receiving the VLC symbol o, $o = 1 \cdots O$, at state w yielding $v_o = w$ and that the first o soft channel outputs were $\mathbf{y}_1^o$, where the notation

$\mathbf{y}_1^o$ represents the vector of soft channel outputs $\mathbf{y}_1^o = [y_1 \dots y_o]$. Equation 5.7 indicates that the α value at instant o is related to all the channel soft inputs spanning from the beginning of decoding at $o = 0$ to the instant $o, o > 0$. However, as seen in Figure 5.4, the quantity α_o can be recursively calculated with the aid of the α_{o-1} values generated during the previous time instant. Hereby we will relate the trellis structure seen in Figure 5.1 to the MAP decoding structure outlined in Figure 5.4 for the sake of demonstrating the recursive calculation of the quantity α. The vertical axis on the left of Figure 5.4 corresponds to the number of received bits in Figure 5.1, while the horizontal axis in Figure 5.4 corresponds to the number of received symbols in Figure 5.1. The notation $\alpha_o(w)$ in Figure 5.4 is used for representing the quantity α at time instant o, while at state w. The notations $\beta_o(w)$ and $\gamma_o(w)$ follow the same rationale. On receiving the soft channel output y_2 in Figure 5.4, the trellis transition emerging from state $v = 2$ to state $v = 3$ will be encountered, when the received source symbol is $C(0)$. By contrast, two parallel transitions are encountered from state $v = 1$ to state $v = 3$, if either a source symbol of $C(1)$ or $C(2)$ is received. Thus, as seen in Figure 5.4, the quantity $\alpha_2(3)$ is calculated as:

$$\alpha_2(3) = \alpha_1(2)\gamma_1(0) + \alpha_1(1)\gamma_1(2) + \alpha_1(1)\gamma_1(2). \tag{5.8}$$

Commencing the corresponding recursion exemplified in Equation 5.8 from the first symbol by assuming $\alpha_0(0) = 1$, i.e. that the trellis starts from the state $v = 0$ in Figure 5.1 or Figure 5.2 and following the rationale of Section 5.3.3 in [115], we have:

$$\alpha_o(w) = \frac{\sum_{w' \in W_{o-1}} \sum_{i=0}^{T-1} \gamma_i(y_{w'}^w, w', w) \cdot \alpha_{o-1}(w')}{\sum_{w \in W_o} \sum_{w' \in W_{o-1}} \sum_{i=0}^{T-1} \gamma_i(y_{w'}^w, w', w) \cdot \alpha_{o-1}(w')}. \tag{5.9}$$

All the α values corresponding to each time instant at different trellis states of Figure 5.2 can be similarly calculated. More explicitly, in the numerator of Equation 5.9 the inner summation $\sum_{i=0}^{T-1} \gamma_i(y_{w'}^w, w', w) \cdot \alpha_{o-1}(w')$ quantifies the probability of encountering transitions from a *particular state* $v_{o-1} = w'$ to state $v_o = w$ engendered by all possible input VLC symbols. By contrast, the outer summation calculates the overall probabilities of the transitions from *all possible states* $v_{o-1} = w', w' \in \{W_{o-1}\}$ in the previous instant $(o-1)$ to the current state $v_o = w$ engendered by their corresponding VLC symbol. The denominator has a third summation summing over all possible states w, $w \in \{W_o\}$, at the current instant o, which acts as a normalisation factor for the sake of ensuring that we have $\sum_{w \in W_o} \alpha_o(w) = 1$, since this condition has to be satisfied by the probability of the associated events.

Similarly, the associated backwards recursion of Section 5.3.3 in [115] calculates the value of

$$\beta_k(w) = P(\mathbf{y}_{w+1}^W | v_o = w), \tag{5.10}$$

which gives the probability that given the trellis state $v_o = w$ was encountered at symbol instant o, the future received soft channel output sequence will be $\mathbf{y}_{w+1}^W$. Similar to the recursive computation of α_o, the quantity β_o can be backwards recursively calculated with the aid of the β_{o+1} value at the next time instant. More explicitly, while still considering Figure 5.4 and following the same philosophy as for the recursive calculation of $\alpha_o(w)$ illustrated in Equation 5.8, the quantity $\beta_2(3)$ can be calculated as follows:

$$\beta_2(3) = \beta_3(5)\gamma_2(1) + \beta_3(5)\gamma_2(2) + \beta_3(4)\gamma_2(0). \tag{5.11}$$

The backwards recursion outlined in Equation 5.11 may be implemented in a similar way to Equation 5.9, but commencing the recursion from the end of the VLC symbol sequence and assuming that the trellis terminates at state $v = W$, i.e. that we have $\beta_O(W) = 1$, yielding [115]:

$$\beta_o(w) = \frac{\sum_{w' \in W_{o+1}} \sum_{i=0}^{T-1} \gamma_i(y_{w'}^w, w', w) \cdot \beta_{o+1}(w')}{\sum_{w \in W_o} \sum_{w' \in W_{o+1}} \sum_{i=0}^{T-1} \gamma_i(y_{w'}^w, w', w) \cdot \beta_{o+1}(w')} i. \tag{5.12}$$

Explicitly, the numerator sums up all transition probabilities associated with arriving at all possible states $v_{o+1} = w', w' \in \{1 \cdots W\}$ at the next instant $w + 1$, from the current state $v_o = w$ engendered by their corresponding input VLC symbols. The denominator normalises the probabilities $\beta_o(w)$, $w \in \{1 \cdots W_o\}$ ensuring that $\sum_{w \in W_o} \beta_o(w) = 1$ is satisfied, since all probabilities of the corresponding events have to sum to unity.

After all the values of Equation 5.7, 5.10 and 5.5 have been obtained, the *a posteriori* probability may be expressed as:

$$P(u_o = t|\mathbf{y}) = \frac{\sum_{w \in W_o} \sum_{w' \in W_{o-1}} \gamma_t(y_{w'}^w, w', w) \cdot \alpha_{o-1}(w') \cdot \beta_o(w)}{\sum_{w \in W_o} \sum_{w' \in W_{o-1}} \sum_{i=0}^{T-1} \gamma_i(y_{w'}^w, w', w) \cdot \alpha_{o-1}(w') \cdot \beta_o(w)}. \tag{5.13}$$

Interpreted in physical tangible terms, the *a posteriori* probability calculated using Equation 5.13 quantifies the probability of encountering a specific VLC symbol t at the o^{th} time instant. This probability is calculated by summing up all the probabilities for the transitions between the time instants o and $o + 1$, which are engendered by the VLC symbol i. The probability $P(u_o = i|\mathbf{y})$ is normalised in Equation 5.13 so that we have $\sum_{i \in T} P(u_o = i|\mathbf{y}) = 1$.

Finally, Equation 5.3 is applied to identify the specific VLC symbol associated with the highest probability as the survivor symbol.

In this section, the process of the symbol based trellis decoding of VLCs has been discussed. The construction of the trellis-based VLC decoder requires the *a priori* knowledge of both the number of VLC symbols and the number of bits per transmission burst to be transmitted to the decoder as side-information. Since the length of the VLC decoder's trellis is determined by the total number of VLC symbols within the transmitted frame, the size of the VLC trellis may become excessive. At the output of the trellis based VLC decoder, the associated symbol probabilities will be provided. However, when the VLC scheme is combined with other bit-based channel codecs, such as a turbo convolutional code or an LDPC code, symbol to bit conversion will be necessitated, which may result in information loss. In order to solve this problem, Bauer and Hagenauer proposed another trellis representation for VLCs [153], which enables the bit-based VLC decoding.

5.2.2 Bit-based decoding of VLCs

The bit based decoding of VLCs [153] requires no additional side information, other than the number of bits transmitted. The size of the trellis is only related to that of the VLC codeword table, which is relatively small compared to that of the VLC symbol-based trellis, even if the transmission packet size is large. The output of the bit-based VLC decoder is the probability of the bits, thus the associated soft outputs may be input directly to the channel decoder for the sake of forming a serially concatenated joint source/channel coding system, without requiring symbol-to-bit probability conversion.

The bit-based trellis of the VLC is constructed as follows [153]. We have to define three different types of nodes in the trellis. The first one is referred as a root node (R), from which a new VLC symbol emerges. The second one is the so-called terminal node (T), where a VLC symbol ends. The third one is termed the intermediate node (I), which represents an intermediate trellis state associated with the bit-based representation of a VLC symbol that has not been completely received. Let us now consider the associated bit-based trellis construction using an example.

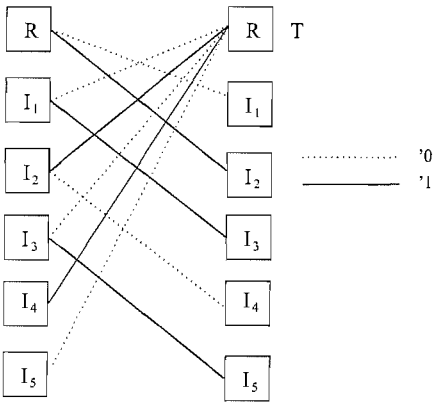


Figure 5.5: Bit-based trellis structure of a VLC, where the dotted lines indicate a transition engendered by a '0' and the solid lines by a '1'

Let us assume that we have a RVLC code table of $c = \{c_1, \cdots c_5\} = \{00, 11, 101, 010, 0110\}$. In Figure 5.5, the dotted lines represent the transitions engendered by a bit 0 and the solid lines represent the transitions engendered by a bit 1. The VLC decoder starts from the root node, (R) of Figure 5.5. Upon receiving the first bit 0, since the decoder has not received a complete VLC symbol, the corresponding transition leads to node I_1. From I_1, if a further 0 is received, then the VLC symbol of 00 has been received, hence the symbol has been completed and the trellis is terminated at the terminal node (T), which constitutes the root node for the next VLC symbol. Otherwise, if a 1 is received upon emerging from I_1, since 01 has not formed a valid VLC symbol as yet, the trellis branch traverses to another intermediate state I_3 in Figure 5.5. Eventually, the trellis transitions should converge at the terminal node.

The MAP algorithm may also be applied to this bit-based trellis, although there is some difference between the symbol-based trellis and the bit-based trellis. In the bit-based trellis, the transitions converge only at the root node and the number of transitions emerging from each node is less than or equal to 2. Both the bit-based and the symbol-based MAP algorithms have been detailed in [115].

5.3 Levenshtein distance

There are various types of variable length codes. Huffman coding has been considered as the most efficient code in terms of providing the shortest average symbol length. However, there are other VLCs, which are capable of outperforming Huffman codes in terms of other performance metrics, for example owing to their stronger error correction capability [164]. Reversible VLCs (RVLS) have the advantage of facilitating decoding from both ends of the received bit sequence, which is beneficial in

the event of losing symbol synchronisation owing to channel errors [144].

In the context of VLCs, the capability of correcting erroneous symbols is more important than correcting bits. In addition to using the most straightforward performance metric, namely the symbol error statistics based on symbol by symbol comparisons, in this section we introduce the Levenshtein distance [165].

Since the variable length codes are capable of self-synchronisation, a single corrupted bit may not inflict severe error propagation. However, when insertion and/or deletion of VLC symbols occurs, a simple symbol by symbol based comparison invoked for generating the associated symbol error statistics may not be a fair procedure. Hence in numerous research papers the Levenshtein distance was used for evaluating the associated symbol error statistics [153] [166] [167].

The Levenshtein distance was named after the Russian scientist Vladimir Levenshtein, who devised the concept. Since its introduction it has been used for quantifying the similarity of two strings. More specifically, the distance is defined as the number of insertions, deletions or substitutions required for transforming one string into another. For example, let us assume that string one is the word *specification* and the second string is also the word *specification*. The Levenshtein distance between these two strings is 0, since they are identical. If, however, the second string is *specfication*, then the Levenshtein distance is 1, since there is a deletion (a missing i), in the second string. As English words often contain some predictability or redundancy, despite the deletion of a character in the second string, we are able to recognise that the second string was meant to be *specification*. The associated symbol error rate in this case is 1/13, while the symbol error rate would be 9/13, if we used character by character based symbol comparisons.

5.4 Performance of VLCs as error correction codes

As a benefit of the trellis-based decoding of the VLC introduced in Section 5.2, both the symbol-based and the bit-based approach is capable of facilitating their employment as a weak error correction codec. Hence here we would like to characterise the achievable BER performance of the VLC using both the symbol-based and bit-based decoding methods, when communicating over an AWGN channel. The parameters of the source codec used are listed in Table 5.1.

Probability	Huffman Code	RVLC Code 1	RVLC Code 2
0.33	00	00	00
0.30	01	11	01
0.18	11	010	10
0.10	100	101	111
0.09	101	0110	11011
Average VLC symbol length	2.19 bits	2.46 bits	2.37 bits
R_s	0.97	0.87	0.9

Table 5.1: Source code parameters [153]

Three different VLCs were designed and their code table is given in Table 5.1. In addition to

Channel	AWGN
Modulation scheme	BPSK
Block length	100 VLC symbols
Source codec	Huffman, RVLC Code 1, RVLC Code 2 as in Table 5.1

Table 5.2: Simulation parameters for the symbol-based VLC trellis decoding using BPSK modulation, when communicating over an AWGN channel.

the classic Huffman codec, two different types of RVLC are utilised. The first RVLC denoted as the *RVLC Code 1* in Table 5.1 is symmetric and hence the associated binary encoded bits are identical with respect to the beginning and end of each VLC symbol, while the second RVLC in Table 5.1 is asymmetric. As we can see in Table 5.1, the same original source symbol set results in a different average VLC symbol length after encoding by the above-mentioned three different encoding methods. In other words, the VLC has a code rate R_s, which can be defined as:

$$R_s = H/\bar{N}, \tag{5.14}$$

where H is the binary entropy of the original source symbol set and $\bar{N}$ is the average symbol length as listed in Table 5.1. The binary entropy H can be calculated as [137]:

$$\sum_i \frac{1}{P_i} log_2(P_i), \tag{5.15}$$

where P_i represents the i^{th} symbol's probability of occurrence. By using Equation 5.15, the entropy of the source symbol set seen in Table 5.1 was found to be 2.139 bits. Thus the corresponding code rate of each individual VLC code was listed in Table 5.1. Given the above parameters, the corresponding E_b/N_0 ratio of the system is calculated as:

$$E_b/N_0 = \frac{E_c}{N_0 \cdot B \cdot R_s \cdot R_c}, \tag{5.16}$$

where E_c is the transmitted energy per modulated symbol, R_s and R_c represent the code rate of the source codec and the channel codec, respectively, while B specifies the number of *bits per symbol* for the corresponding modulation mode utilised.

5.4.1 Symbol-based VLC decoding performance

The symbol-based VLC decoding procedure assumes that the receiver has perfect knowledge of the number of symbols as well as the number of bits within a transmission block. The simulation parameters for the symbol-based VLC decoding scheme are investigated as follows.

As demonstrated in Figure 5.6, the RVLC 1 scheme achieves the best performance while the Huffman code exhibited the worst performance. Comparing these three source codecs in Table 5.1, we can observe that the Huffman code achieves the shortest average symbol length, while the RVLC 1 arrangement has the longest average symbol length. In comparison to the entropy computed for the given set of symbol probabilities, the Huffman code's average symbol length is the closest, which configures that the Huffman coding is the most efficient way of encoding VLC symbols. By contrast,

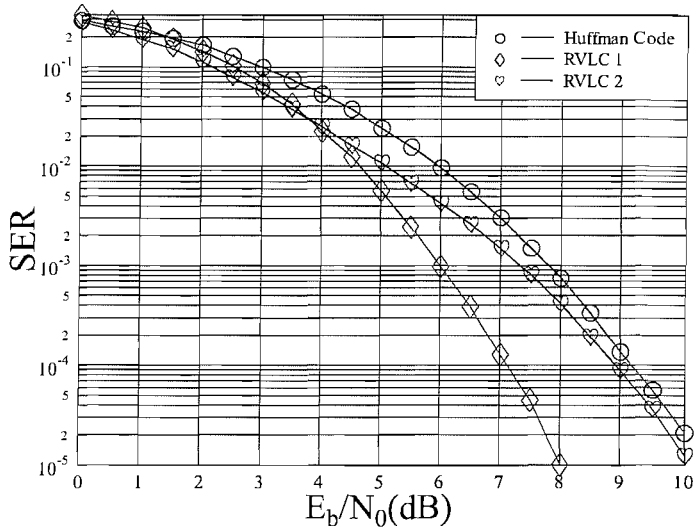


Figure 5.6: Performance of symbol-based VLC decoding using the parameter summarised in Table 5.2 when communicating over an AWGN channel.

the RVLC 1 scheme has the longest average symbol length. By comparing the RVLC 1 and RVLC 2 codewords in Table 5.1, it can be observed that the RVLC 1 scheme is a symmetric RVLC, while the RVLC 2 arrangement is an asymmetric one. Hence the RVLC 1 scheme has more residual redundancy in the code, and this redundancy is now exploited by the trellis decoding algorithm, resulting in the best error correction performance among the three VLC codec studied.

5.4.2 Bit-based VLC decoding performance

The symbol-based VLC decoding proposed by Hagenauer [154] assumes that the receiver has explicit knowledge of the number of symbols and the number of bits within a transmission block. Furthermore, the resultant symbol sequence is not designed for conveniently delivering soft information of each individual bit for the sake of exchanging information with other Soft In Soft Out (SISO) decoders. By contrast, the bit-based VLC trellis decoding algorithm of [153] invokes trellis decoding in a bit-by-bit manner, and the resultant bit-based soft information provides a convenient approach for constructing an iterative decoding scheme, when the VLC decoder is concatenated to other SISO decoders. A further advantage of the bit-based VLC decoding algorithm that it requires only the knowledge of the number of bits within the transmission block for constructing the decoding trellis, while the symbol-based VLC decoder required both the number of bits as well as the number of symbols. Our experiments using bit-based trellis decoding of VLCs will assume that the length of each transmission block expressed in terms of the number of bits transmitted is fixed. We will inform the receiver of the exact number of VLC encoding bits transmitted in the block with the aid of a side-information sequence, as seen in Figure 5.7.

As seen in Figure 5.7, the overall length of the transmission block is fixed. At the beginning of

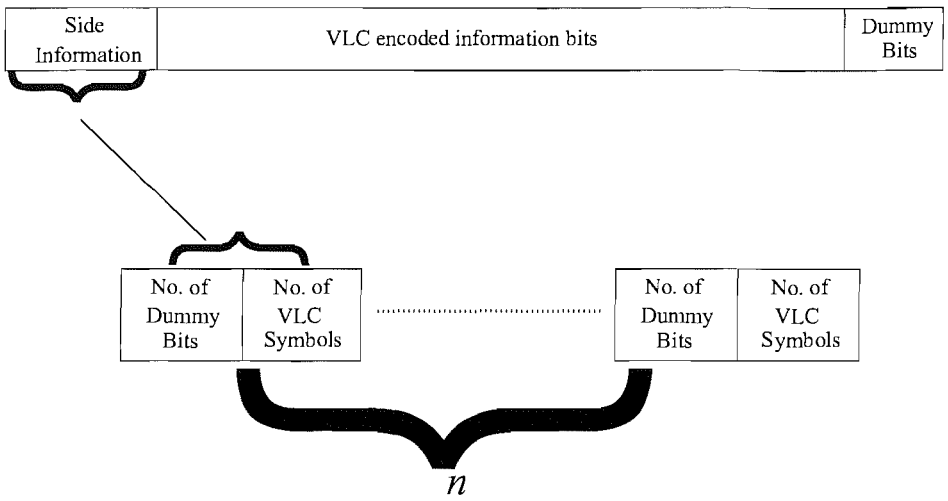


Figure 5.7: Transmission frame structure of bit-based VLC decoding, where the side information (the number of dummy bits and the number of VLC symbols) is repeated n times for the sake of invoking majority logic decision at the decoder.

the transmission frame we include some side information for the sake of informing the receiver as to exactly how many VLC-encoded information bits are transmitted within the block. Following the side information the VLC-encoded information bits representing the original VLC-encoded source symbols are included. As seen in Figure 5.7, the side information is constituted by the information indicating how many dummy bits have to be included at the fixed-length transmission frame. The number of original VLC-encoded source symbols is also supplied to the decoder as side information, each repeated n times for the sake of facilitating majority logic decisions, although this is not explicitly required by the bit-based VLC-decoder. Despite this extra protection, the side information may be corrupted and hence the synchronous detection of the transmitted frame cannot be guaranteed. The side information conveying the number of dummy bits and the number of VLC-encoded symbols is represented by using a binary vector. Since the size of the side information vector and the overall block length is fixed, the remaining capacity of the transmission frame dedicated to the VLC encoded information bits is also fixed. The encoder will map each source symbol to the binary representation of its VLC symbol and amp this binary information sequence to the transmission frame. It might occur that the incoming source symbols result in a transmission frame length which is longer than the remaining 'capacity' of the transmission block. In this case, the corresponding source symbol has to be mapped to the next transmission frame, and dummy bits will be inserted into the frame for the sake of filling up the last a few positions. Thus the maximum number of dummy bits that may have to be inserted into the block will be the maximum VLC-encoded symbol length minus one. Hence, the size of the binary vector required for representing the dummy bits is determined by the binary length of the longest VLC symbol. The maximum number of VLC-encoded symbols that can be mapped to a single transmission frame is approximately given by the transmission frame size divided by the minimum VLC symbol length. Thus the required size of the binary vector representing the number of symbols is $log_2(block\ size/minimum\ symbol\ length)$. The number of VLC symbols is also transmitted as side information, although this is not necessary for the bit-based VLC decoding algorithm, as stated

Channel	AWGN
Modulation scheme	BPSK
Block length	256 bits, including side information and dummy bits
Source codec	Huffman, RVLC Code 1, RVLC Code 2 of Table 5.1
No. of side information repetitions	3

Table 5.3: Simulation parameters for the bit-based VLC trellis decoder using BPSK modulation, when communicating over an AWGN channel.

above. This information is nonetheless required for evaluating the symbol error statistics during the simulations. Transmitting this extra information will slightly reduce the achievable throughput, thus the simulation results to be provided in Section 5.5 and Figures 5.8 and 5.9 are slightly worse than in the scenario where the number of VLC symbols is not transmitted as side information. The simulation parameters used for investigating the bit-based algorithm are given in Table 5.3.

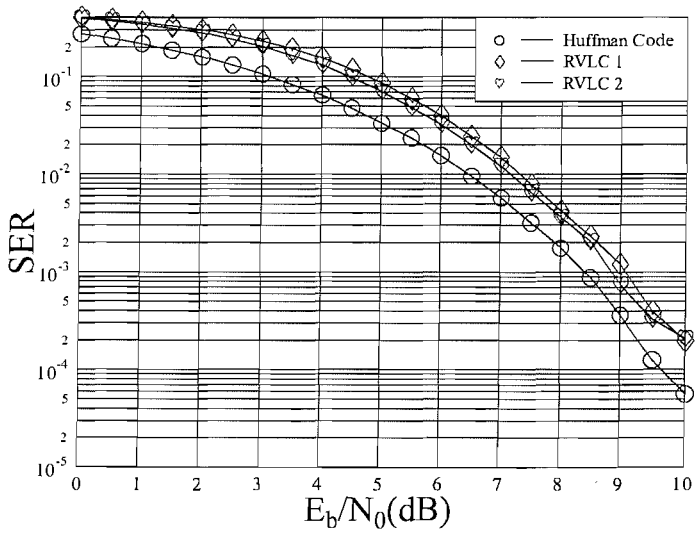


Figure 5.8: Performance of bit-based VLC decoding using the parameters summarised in Table 5.3 when communicating over an AWGN channel.

In comparison to the results shown in Figure 5.6 for the symbol-based VLC decoding techniques, the three VLC source decoders using the bit-based algorithm exhibited a similar performance trend to each other. More explicitly, the three SER curves decay at a similar rate, which is different from the results recorded, when using the symbol-based algorithm, as shown in Figure 5.8, where the Huffman code achieves the best performance, while the RVLC 1 arrangement the worst. More explicitly, the three VLC codecs have a similar BER performance when the bit-based VLC decoding algorithm is applied, although we observe three different BER performance curves owing to the E_b/N_0 shift according to Equation 5.16 imposed by taking into account the three different average VLC symbol

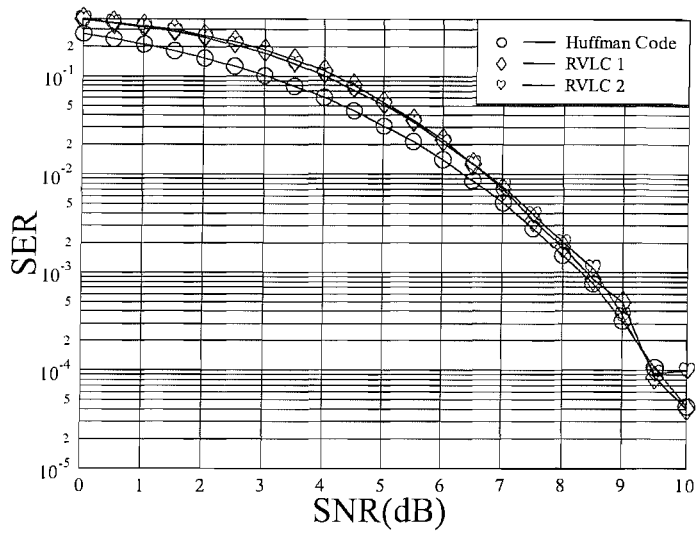


Figure 5.9: Performance of bit-based VLC decoding using the parameters summarised in Table 5.3 when communicating over an AWGN channel, when the entropy-based shifting of the BER curves was not applied.

lengths, when using the different source encoders. The fact that the three VLC codecs have a similar BER versus E_b/N_0 performance is demonstrated in Figure 5.9, where in contrast to the E_b/N_0 curves of Figure 5.8, the BER curves were not shifted according to the different average symbol lengths of the three codes. Even though the source symbol sequence has a non-uniform probability of occurrence distribution, the VLC symbols will be mapped to bits, which will results in a uniformly distributed probability of encountering both a binary one and a binary zero. Since the bit-based VLC trellis decoding algorithm by definition operates on a bit-by-bit basis, thus all the three bit-based VLC source decoder exhibits a similar performance. However, when the bit-based VLC decoding scheme is concatenated with a range of other SISO decoders in Section 5.5, a different performance trend will be observed.

5.5 Joint source and channel decoding using VLCs

As discussed in Section 5.4.2, the trellis decoding of variable length codes is beneficial in the context of joint source and channel decoding, since the bit-based decoding of VLCs directly provides the bit probability, which can be exchanged iteratively with the channel decoder. In order to capitalise on these potential benefits, in this section, we invoke bit-based trellis decoding of VLCs in the context of iterative source and channel decoding.

Figure 5.10 provides the detailed schematic of the joint source/channel decoding scheme considered. As seen in the figure, the channel decoder accepts both the soft output of the demodulator and the *a priori* source information produced by the source decoder as its input. Since in the first iteration

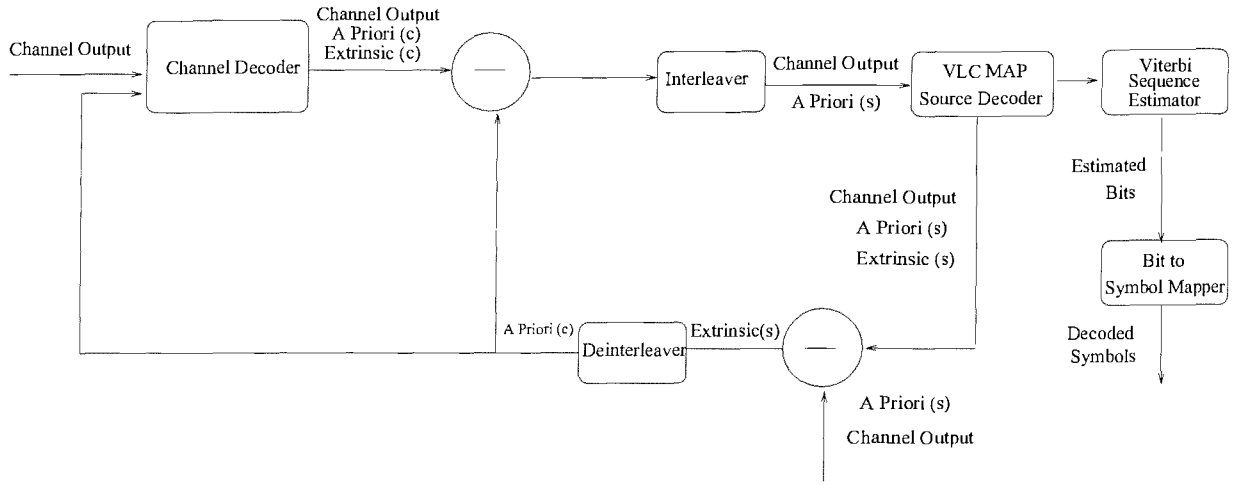


Figure 5.10: Iterative joint source and channel decoding schematic

there is no *a priori* information, the source decoder typically delivers an *a priori* information of 0 in the first iteration, which corresponds to having a probability of 0.5 for both a binary one and a zero. After channel decoding, the extrinsic information *Extrinsic* (*c*) of Figure 5.10 is produced and, together with the demodulator's soft output, this information is forwarded to the source after being deinterleaved for the sake of presenting the information bits in their original order of appearance. The *Extrinsic* information supplied by the channel decoder will act as the *a priori* information for the source decoder. Again, the source decoder will generate the extrinsic information *Extrinsic* (*s*), which will constitute the *a priori* information for the channel decoder. After the affordable number of turbo iterations has been reached, the output of the source decoder will be passed to a sequence estimator. The sequence estimator in Figure 5.10 is necessary, because the received codewords are decoded in the source decoder block using the MAP algorithm operating on the bit-based VLC trellis, and the VLC MAP decoder attempts to maximise the *a posteriori* probability of each individual bit. Therefore, it is not guaranteed that the bit-stream output by the VLC MAP source decoder may be directly mapped to the VLC symbols. Hence the Viterbi algorithm is invoked in the sequence estimator block to provide the best possible VLC-string estimate of the entire bit stream, before outputting a legitimate sequence that is decodable by the bit to symbol mapper.

In the rest of this section we will characterise the achievable performance of a joint source and channel codec. We will use three different SISO channel decoders, namely a convolutional code, an LDPC code and a turbo convolutional code for constructing a serially concatenated jointly decoded system, as shown in Figure 5.10. The associated simulation parameters are summarised in Table 5.4. The number of LDPC iterations was set to five for the sake of maintaining a similar decoding complexity as that of the RSC(2,1,5) code for the sake of a fair comparison. The turbo channel decoder in Figure 5.10 used a single iteration, however the complexity of the turbo decoder was still twice that of the other two channel decoders, since it has two constituent RSC decoders.

It can be observed in Figures 5.11 to 5.13 and 5.15 to 5.17 that when the VLC source codec is concatenated with a channel codec, the three different source encoding methods exhibit different performance trends. Huffman coding has the minimum residual redundancy residing in the source code,

Channel		AWGN,Uncorrelated Rayleigh
Modulation		BPSK
VLC block length		1024 bits
Side information repetition time		3
Number of joint iteration		0, 1, 2, 4
Source codec		Huffman, RVLC 1, RVLC 2
Channel codec		
RSC(2, 1, 5)	Generator Polynomial	23, 35
	Code termination	true
Turbo Convolutional Code	Component RSC code	RSC(2,1,5)
	Puncturing pattern	10, 01
	Number of TC iteration	1
LDPC	Column weight	3
	PCM construction	regular
	Number of LDPC iterations	5

Table 5.4: Simulation parameters for joint source and channel decoding, using the three VLC source codecs of Table 5.1.

hence regardless of which of the three channel codecs is applied, the concatenated system achieves only a modest joint iteration gain. By contrast, the RVLC 1 source codec has the longest average symbol length of 2.46bits/symbol, implying the presence of a higher amount of residual source redundancy which results in a more significant joint iteration gain being achieved. Finally, the RVLC 2 source codec has an average symbol length of 2.37 bits/symbol, and hence the attainable performance of the RVLC 2 source codec is in between that recorded for the Huffman codec and the RVLC 1 source codec.

Furthermore, it can be observed in the Figures that the curves corresponding to RVLC 1 are steeper than those related to Huffman code and to the RVLC 2 scheme. This trend is particularly dominant when the schemes are concatenated with the turbo convolutional code and communicating over an AWGN channel, as shown in Figure 5.12. By observing the source code assignment in Table 5.1 it can be observed that the distance between the VLC codewords is two for the RVLC 1 scheme, while it is only one for the Huffman code and the RVLC 2 code. As seen from Figures 5.14 and 5.18, for either an AWGN or an uncorrelated Rayleigh fading channel, the RVLC 1 code having a free distance of two had a superior performance in comparison to that of the other VLC codes having a free distance of one. Therefore, the free distance of the VLC code should also be taken into consideration, when designing a good VLC source codec.

5.6 Complexity

Since the decoding of VLC codes can be characterised by a trellis structure, thus the complexity of the VLC decoding can be quantified by the number of trellis transitions. The complexity of the symbol-based MAP VLC decoding algorithm will be calculated on a *per symbol* basis, while for bit-based VLC decoding, the complexity per decoded bit will be quantified.

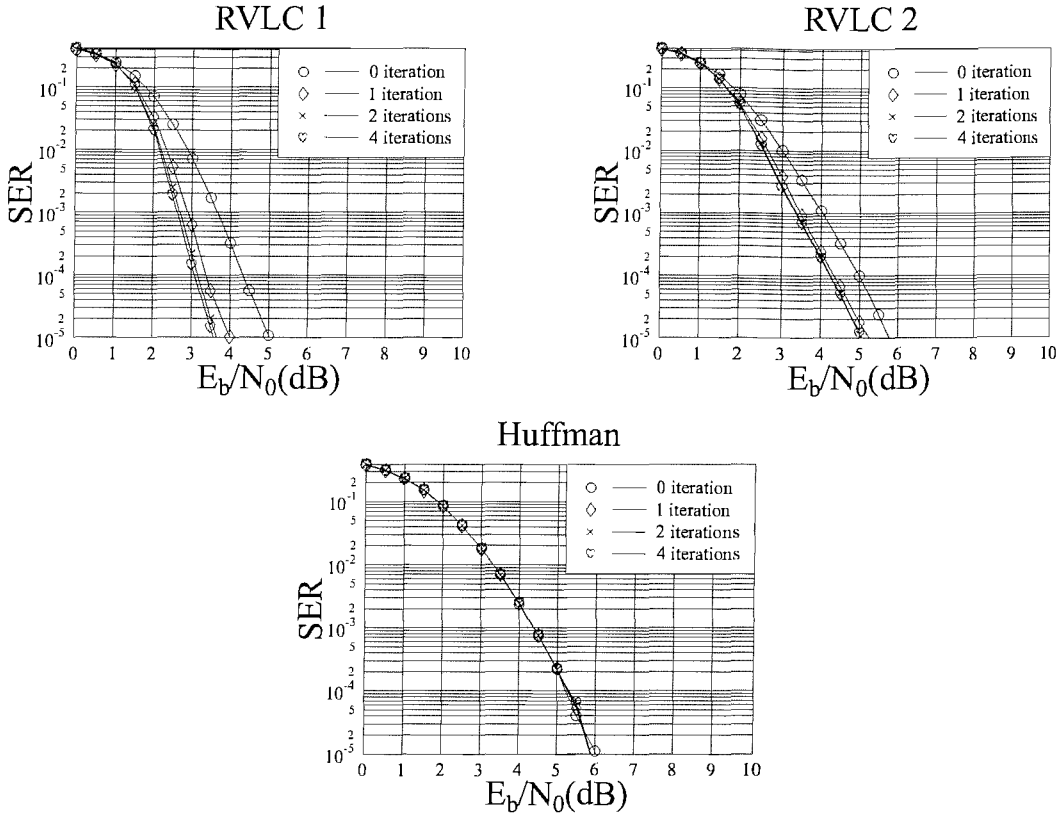


Figure 5.11: VLC SER versus E_b/N_0 performance of the three VLC source codecs of Table 5.1 concatenated with the RSC(2,1,5) convolutional code of Table 5.4, when communicating over an AWGN channel. The VLC frame length was 1020 bits, while the overall length of the transmission burst after channel encoding was 2048 bits. The number of joint iterations invoked was increased from 1 to 4. The VLC frame length was 1020 bits rather than 1024 bits for the LDPC case was because there are four termination bits required for the RSC(2,1,5) code during encoding. The required E_b/N_0 of the various schemes at a BER of 10^{-4} will be summarised in Figure 5.14 and Table 5.5.

For the case of the symbol-based trellis, the size of the trellis may become excessive, when the size of the transmission burst is increased. As seen in Figure 5.2, most of the time intervals will have a fixed number of trellis transitions, except at the beginning where the trellis start from zero state and at the final stage, where all transitions have to converge to a single final stage. Hence for a comparatively long block length we may deem the average number of trellis transitions to be constant, since the effect of the reduced number of transitions at the beginning and the end of the trellis may be neglected. Explicitly, the number of trellis transitions per time interval can be quantified as $v_{max} \cdot T$, where v_{max} is the maximum number of trellis states as defined by Equation 5.2 and T is the size of the source symbol alphabet, i.e. the number of possible source symbols. In Equation 5.2 the knowledge of the number of transmitted VLC symbols and the corresponding number of bits per transmission block is assumed to be known, but the quantity v_{max} will vary from block to block. However, statistically speaking, the average number of bits W per transmission block should be the average number of VLC symbols O times the average VLC symbol length. Thus, the longer the block, the higher v_{max} becomes. Hence the complexity of the symbol-based VLC decoding algorithm increases, when a long

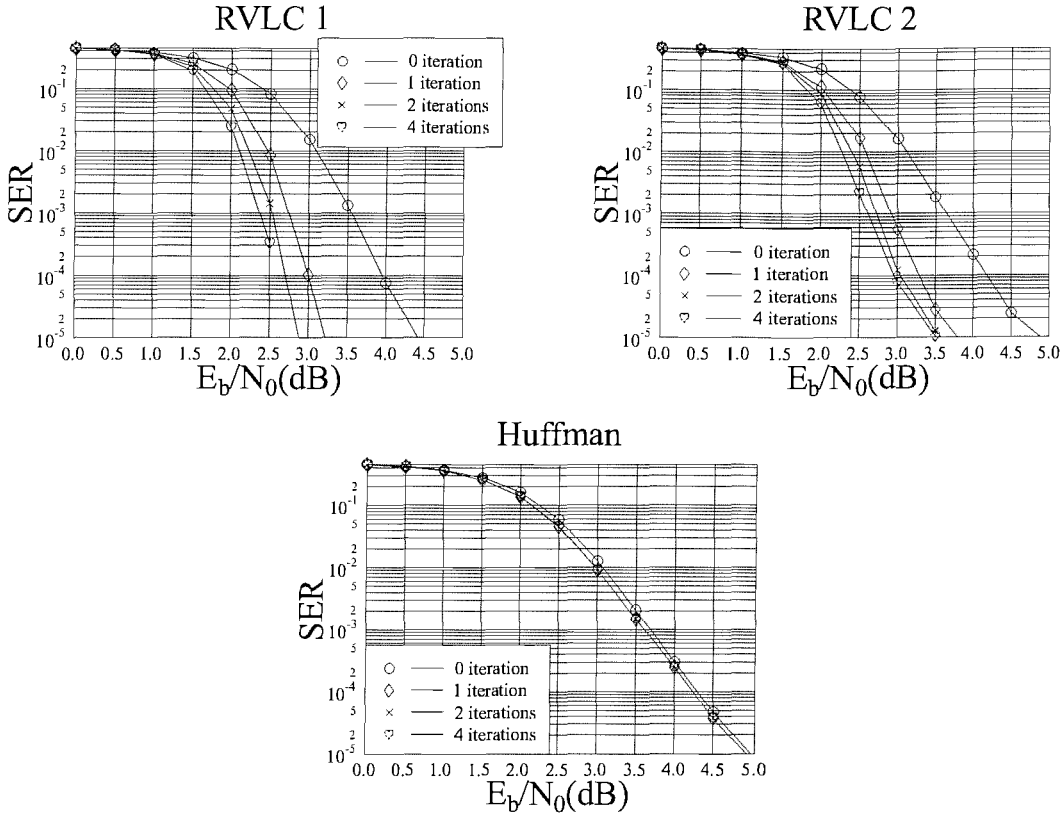


Figure 5.12: VLC SER versus E_b/N_0 performance of the three VLC source codecs of Table 5.1 concatenated with the turbo convolutional code of Table 5.4, when communicating over an AWGN channel. The VLC frame length was 1020 bits, while the overall length of the transmission burst after channel encoding was 2048 bits. The number of joint iterations invoked was increased from 1 to 4. The VLC frame length was 1020 bits rather than 1024 bits for the LDPC case was because there are four termination bits required for the turbo convolutional code during encoding. The required E_b/N_0 of the various schemes at a BER of 10^{-4} will be summarised in Figure 5.14 and Table 5.5.

transmission block length is encountered.

When using a bit-based trellis, since the trellis structure only depends on the size of the VLC symbol alphabet, the associated decoding complexity is a constant and will be quantified below. The bit-based trellis structure of the three VLC codecs characterised in Table 5.1 can be found in Figure 5.19.

As seen from Figure 5.19, the Huffman code has 8 legitimate transitions. By contrast, the two RVLC codes both have 10 transitions. Thus the Huffman code is inherently less complex. Since the Log-MAP algorithm is used, the VLC decoding complexity may be approximated similarly to that of a turbo convolutional code, which was estimated in [115] as $3 \times no\ of\ transitions$. The multiplier factor of 3 is present, because the MAP algorithm requires the calculation of the α, β and γ terms, which implies that the decoder has to traverse through the trellis three times. Therefore, the complexity of the bit-based VLC trellis decoding scheme can be quantified by the number of trellis transitions. The RVLC 1 and RVLC 2 schemes have a decoding complexity, which is about 25% higher than that of the Huffman code.

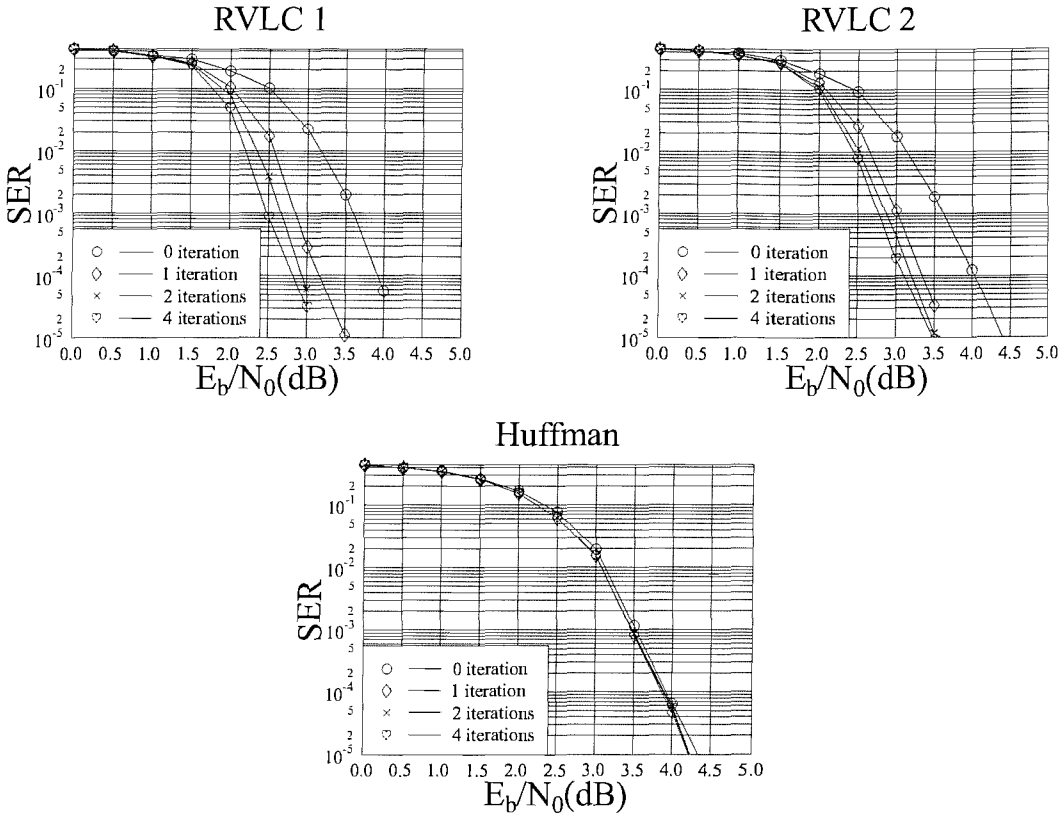


Figure 5.13: VLC SER versus E_b/N_0 performance of the three VLC source codecs of Table 5.1 concatenated with the LDPC code of Table 5.4, when communicating over an AWGN channel. The VLC frame length was 1024 bits, while the overall length of the transmission burst after channel encoding was 2048 bits. The number of joint iterations invoked was increased from 1 to 4. The required E_b/N_0 of the various schemes at a BER of 10^{-4} will be summarised in Figure 5.14 and Table 5.5.

5.7 Summary and conclusion

Trellis based joint source/channel decoding scheme constitutes an efficient way of exploiting the residual redundancy inherent in the source-coded stream as well as imposed by the channel code. The symbol-based trellis decoding [154, 161] of VLCs was introduced in Section 5.2.1. Buttigieg and Farrell [162, 163], as well as Bauer and Hagenauer [154] used the MAP algorithm for decoding VLC assumed perfect knowledge of the number of VLC symbols and the number of bits in the received block, which is unrealistic in practice. Furthermore, the size of the trellis constructed for the symbol-based VLC decoding algorithm is related to both the number of symbols and bits, thus for high-blocklength applications the trellis construction process might be memory-intensive. In order to improve the VLC symbol-based decoder's attributes, Hagenauer [153] proposed the employment of bit-based trellis decoding of the VLCs, which requires the knowledge of the number of bits in the transmission block, but not of the VLC symbols. Furthermore, the bit-based process is capable of providing bit-based soft information, which renders the message passing in iterative joint source and channel decoding more straightforward. The VLC trellis based decoding uses a slightly modified MAP algorithm, which was outlined with the aid of Figure 5.4 in Section 5.2.

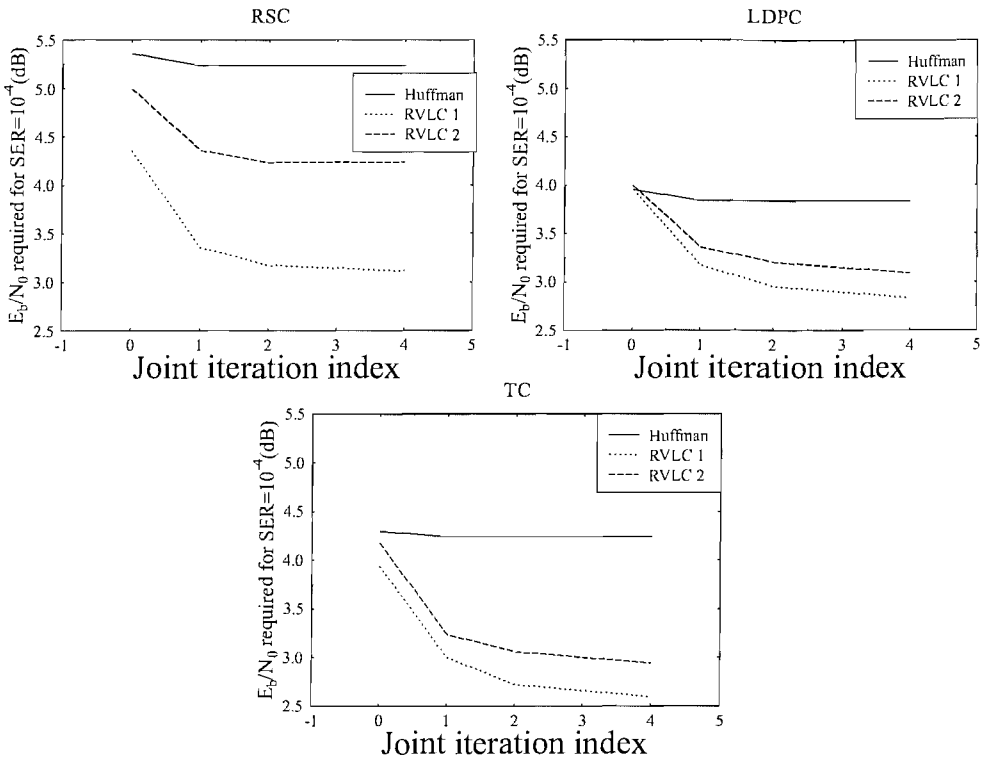


Figure 5.14: E_b/N_0 required for achieving a SER=10^{-4} for the three VLC source codecs of Table 5.1 concatenated with the three channel codecs of Table 5.4, when communicating over an AWGN channel. The VLC frame length was 1024 bits, while the overall length of the transmission burst after channel encoding was 2048 bits. The number of joint iterations invoked was increased from 1 to 4.

When using the trellis-based VLC decoding algorithm, a VLC codec was capable of acting as a weak error correction codec. The performance of the stand-alone VLC codec using symbol-based trellis decoding was evaluated in Section 5.4.1. Since the symbol-based trellis is capable of exploiting both the symbol and bit-related received information, the symmetric-construction RVLC 1 scheme achieves the best performance amongst the three candidate schemes owing to the residual redundancy inherent in the code. When the bit-based trellis decoder is used for the three VLC codecs of Section 5.4.2, the three VLC codecs behave similarly to each other, since all the bit-based trellis decoding algorithms rely on exploiting bit-oriented information. Thus although the VLC source symbols have different probability of occurrence, when they are encoded to bits, the occurrence probability of a binary zero and binary one becomes similar.

In Section 5.5, we proposed a jointly optimised source-channel decoding scheme and three different channel codecs were concatenated with the three VLC source codecs using bit-based trellis decoding. The performance of the joint source-channel coding scheme characterised in Figures 5.11 to 5.18 was evaluated in various concatenated scenarios and both the AWGN channel and the uncorrelated Rayleigh fading channel have been used for investigating the different behaviours of the various schemes in Figures 5.11 to 5.18. It was concluded that although using a code having a longer average symbol length will reduce the coding efficiency of the source codec, when employed in an iterative scheme, the associated higher redundancy may be expected to benefit the scheme by exchanging extrinsic

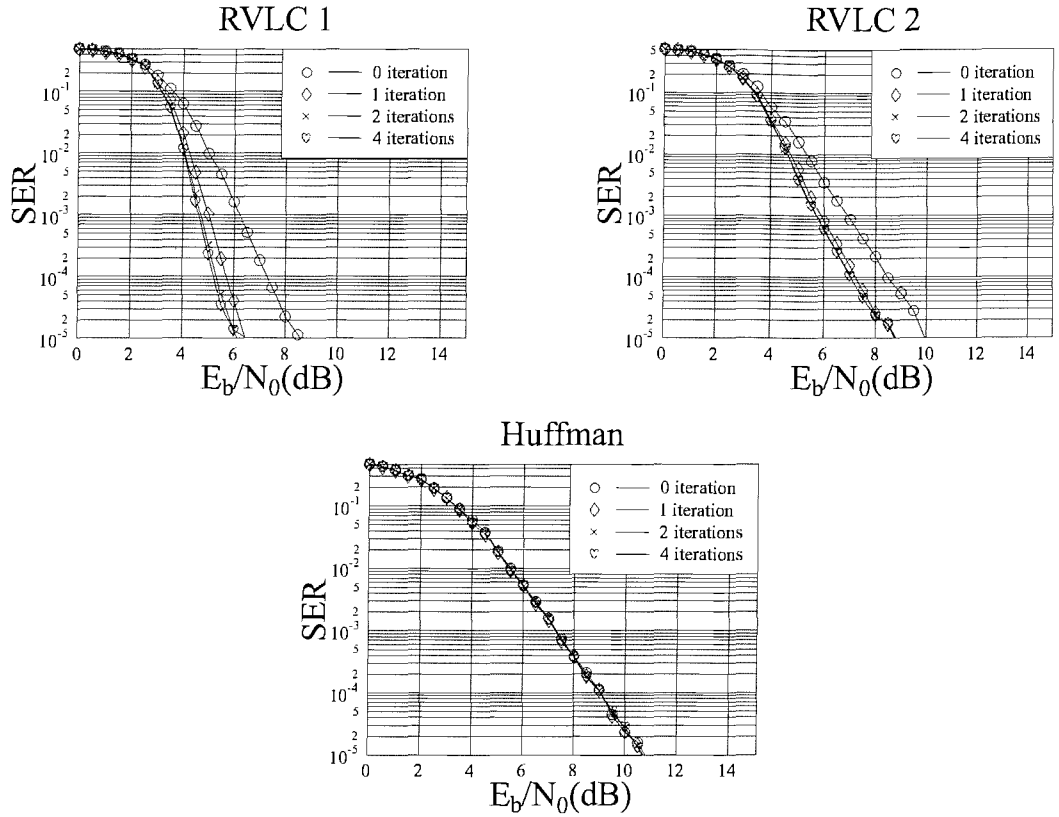


Figure 5.15: VLC SER versus E_b/N_0 performance of the three VLC source codecs of Table 5.1 concatenated with the RSC(2,1,5) convolutional code of Table 5.4, when communicating over an uncorrelated Rayleigh fading channel. The VLC frame length was 1020 bits, while the overall length of the transmission burst after channel encoding was 2048 bits. The number of joint iterations invoked was increased from 1 to 4. The VLC frame length was 1020 bits rather than 1024 bits for the LDPC case was because there are four termination bits required for the RSC(2,1,5) code during encoding. The required E_b/N_0 of the various schemes at a BER of 10^{-4} will be summarised in Figure 5.18 and Table 5.5.

information between the source and channel decoder, thus the overall system performance may be improved. Furthermore, it has been observed in Figure 5.12 that the RVLC 1 code has a faster convergence rate compared to the other two VLC coding schemes. This is a consequence of its higher free distance of two between the RVLC 1 scheme's codewords, while the other two VLC codecs have a free distance of one. Therefore, the free distance of the VLC codewords also beneficially contributes to the overall performance gain of the joint source and channel coding scheme considered.

The choice of the VLC can be made with the aid of EXtrinsic Information Transfer chart (EXIT-chart) analysis. The EXIT-chart was devised by ten Brink [107], where the iterative exchange of the extrinsic information can be graphically visualised. Therefore, the EXIT-Chart is extremely useful, when designing concatenated systems. The EXIT-chart analysis stipulates the assumption that both the input *a priori* and the output *a posteriori* information of the soft-in soft-out decoder are Gaussian distributed, provided the decoding blocklength is sufficiently long. The EXIT-chart of the three VLC codes shown in Table 5.1 are given in Figure 5.20. It can be observed from Figure 5.20 that the RVLC 1 code having a free distance of two is capable of providing the best extrinsic information. By contrast,

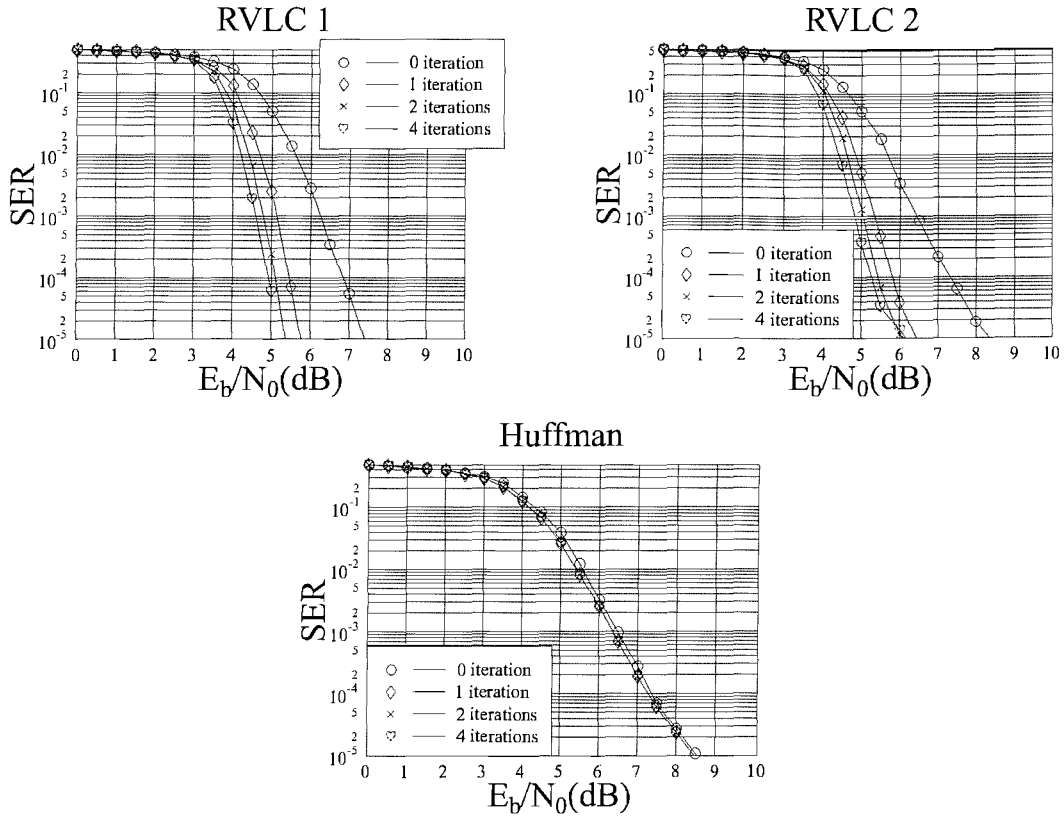


Figure 5.16: VLC SER versus E_b/N_0 performance of the three VLC source codecs of Table 5.1 concatenated with the turbo convolutional code of Table 5.4, when communicating over an uncorrelated Rayleigh fading channel. The VLC frame length was 1020 bits, while the overall length of the transmission burst after channel encoding was 2048 bits. The number of joint iterations invoked was increased from 1 to 4. The VLC frame length was 1020 bits rather than 1024 bits for the LDPC case was because there are four termination bits required for the turbo convolutional code during encoding. The required E_b/N_0 of the various schemes at a BER of 10^{-4} will be summarised in Figure 5.18 and Table 5.5.

the Huffman code is hardly contributing any extrinsic output. Therefore, when the Huffman code is used in the joint source-channel decoding scheme, little iteration gain can be achieved.

The E_b/N_0 values required by the joint source and channel coding scheme parameterised in Table 5.4 for achieving SER= 10^{-4} are summarised in Table 5.5, when communicating over an AWGN and an uncorrelated Rayleigh fading channel. As seen in Table 5.5, even though in some cases the TC coded scheme performs better than the LDPC assisted scheme, the decoding complexity of the LDPCC is only about 50% of that of the turbo coded scheme. Therefore, the LDPC-assisted joint source-channel decoding scheme is our best design option.

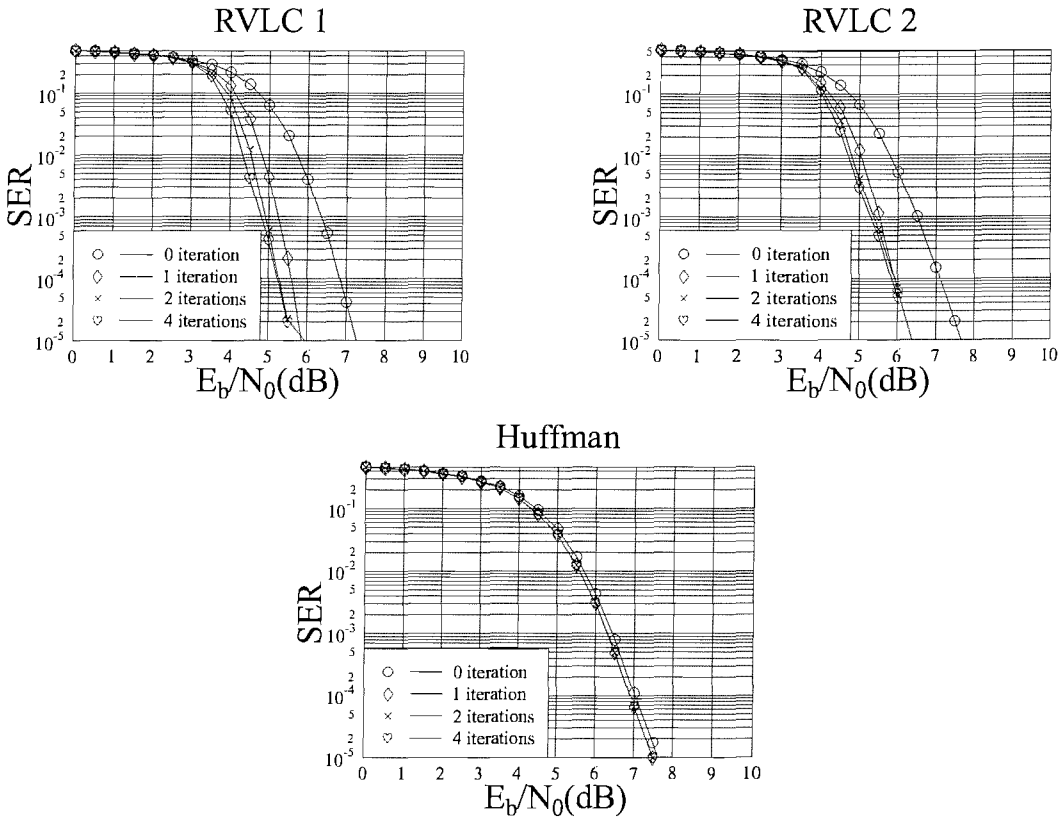


Figure 5.17: VLC SER versus E_b/N_0 performance of the three VLC source codecs of Table 5.1 concatenated with the LDPC code of Table 5.4, when communicating over an uncorrelated Rayleigh fading channel. The VLC frame length was 1024 bits, while the overall length of the transmission burst after channel encoding was 2048 bits. The number of joint iterations invoked was increased from 1 to 4. The required E_b/N_0 of the various schemes at a BER of 10^{-4} will be summarised in Figure 5.18 and Table 5.5.

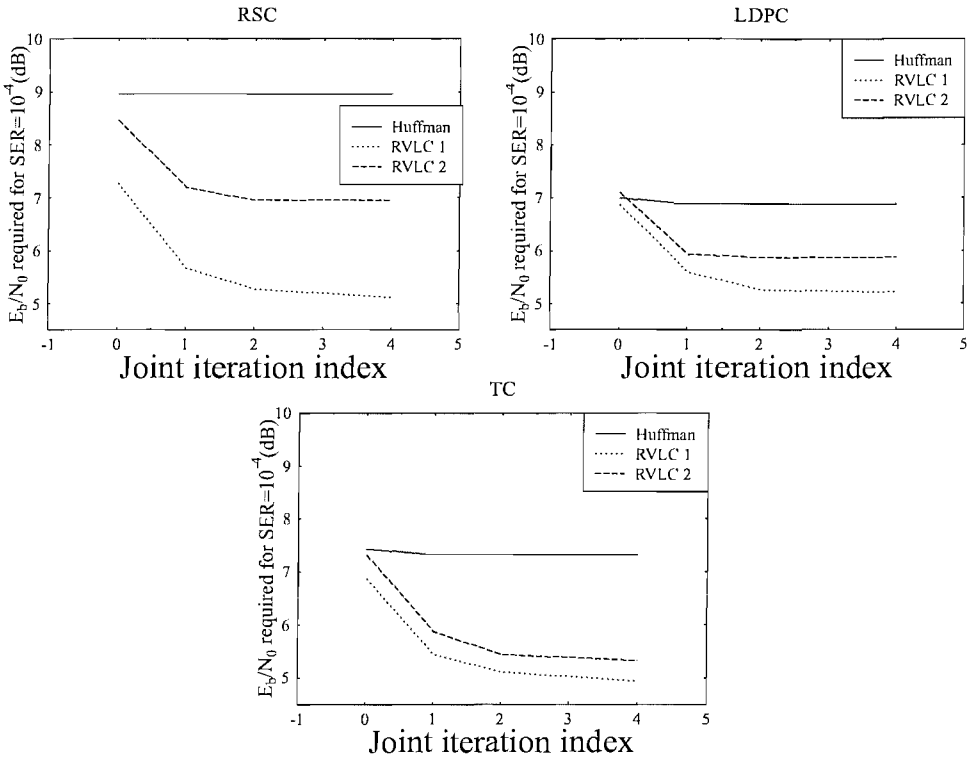


Figure 5.18: E_b/N_0 required for achieving a SER=10^{-4} for the three VLC source codecs of Table 5.1 concatenated with the three channel codecs of Table 5.4, when communicating over an uncorrelated Rayleigh fading channel. A VLC frame length was 1024 bits, while the overall length of the transmission burst after channel encoding was 2048 bits. The number of joint iterations invoked was increased from 1 to 4.

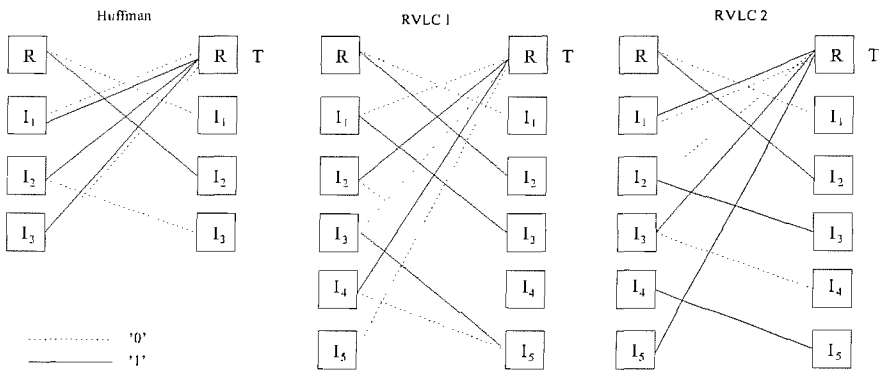


Figure 5.19: Bit-based trellis structure for the three different VLC codes in Table 5.1. The dotted line represents the transition incurred by an input of binary 0, while the solid lines represents the transition triggered by an input of binary 1.

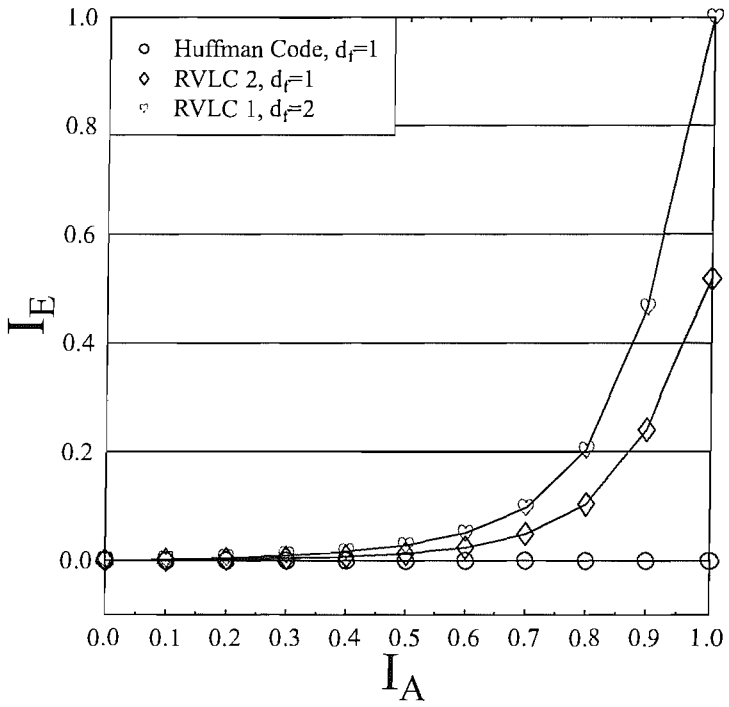


Figure 5.20: EXIT-chart for the five-symbol VLC codes specified in Table 5.1.

Channel	Source codec	Channel codec	E_b/N_0(dB) required for achieving SER=10^{-4}			
			0 Iteration	1 Iteration	2 Iterations	4 Iterations
AWGN	Huffman	RSC	5.36	5.24	5.24	5.24
		TC	4.3	4.24	4.24	4.24
		LDPC	**3.96**	**3.84**	**3.84**	**3.84**
	RVLC 1	RSC	4.36	3.36	3.18	3.12
		TC	3.94	3	2.72	2.6
		LDPC	**3.96**	**3.18**	**2.96**	**2.84**
	RVLC 2	RSC	5	4.36	4.24	4.24
		TC	4.18	3.24	3.06	2.94
		LDPC	**4**	**3.36**	**3.21**	**3.1**
Uncorrelated Rayleigh fading	Huffman	RSC	8.96	8.96	8.96	8.96
		TC	7.44	7.33	7.33	7.33
		LDPC	**7.00**	**6.88**	**6.88**	**6.88**
	RVLC 1	RSC	7.28	5.68	5.28	5.12
		TC	6.88	5.44	5.11	4.94
		LDPC	**6.88**	**5.61**	**5.27**	**5.22**
	RVLC 2	RSC	8.48	7.2	6.96	6.96
		TC	7.33	5.88	5.44	5.33
		LDPC	**7.11**	**5.94**	**5.88**	**5.88**

Table 5.5: E_b/N_0 required for the joint source and channel coding schemes parameterised in Table 5.4 to achieve SER=10^{-4}, when communicating over an AWGN channel and an uncorrelated Rayleigh fading channel.

Chapter 6

Weighted Bit Flipping Decoding of LDPCC

It has been demonstrated in Chapters 2, 3 and 4 that the achievable performance of LDPC codes may approach the Shannon limit, when using probabilistic decoding, also often referred to as the Sum-Product Algorithm (SPA) [15, 99]. A specific drawback of this soft decoder is however that the associated decoding complexity quantified in Section 4.3.5 is high. Furthermore, for the sake of achieving a good performance, the blocklength has to be comparatively long, hence the achievable performance improvement of the soft-decoded LDPC code is attained at the cost of a relatively high delay.

The hard decision based Weighted Bit-Flipping (WBF) algorithm proposed by Kou *et al.* [83,168] strikes a good trade-off between the achievable performance and the associated decoding complexity. We will show in Section 6.1 that the WBF algorithm evaluates a specific error term E_n for each individual message node at position n based on all the check-nodes' information and during each iteration the binary value of the least reliable message node having the highest error term E_n will be inverted. Kou's idea [83, 168] was further improved by Zhang *et al.* [101] by jointly considering both the check-node-based and the message-node-based information upon introducing a weighting factor α_I when evaluating the error term E_n. This improved WBF will be referred to as the IWBF algorithm in our forthcoming discussion. Furthermore, Nouh *et al.* proposed the Bootstrap Weighted Bit-Flipping (BWBF) algorithm [102]. The BWBF algorithm determines, which particular received bit is unreliable by comparing it to an off-line-calculated threshold denoted as α_B. The soft value of the unreliable bit will be erased and replaced by the best possible estimate of this particular bit generated by exploiting the knowledge of the reliable neighbouring message nodes. However, the WBF, IWBF and BWBF algorithms attribute the violation of a particular check exclusively to the least reliable message node participating in the check, while all the information that could be gleaned from the other participating check nodes is discarded.

In this chapter, we commence by describing the WBF, the IWBF and the BWBF decoding algorithms in Sections 6.1, 6.2 and 6.3, respectively. In Section 6.4, we propose a novel Reliability-Ratio based Weighted Bit-Flipping (RRWBF) technique, which takes the soft information of all message nodes into account, which contribute to a specific parity check. Two worked decoding examples are provided in Section 6.5 for the sake of exemplifying the decoding procedures of the BWBF and the

RRWBF algorithms. The corresponding simulation results are summarised in Section 6.6 for the above mentioned four bit-flipping algorithms and finally a summary of the achievable performance gains will be provided in Section 6.8.

6.1 Weighted bit-flipping algorithm

Let us assume that an LDPC encoded codeword $\mathbf{c}$ is transmitted over an AWGN channel using BPSK modulation and a sequence of noise-contaminated soft-values $\mathbf{y}$ is received. Assume furthermore that a hard decision can be made concerning each individual bit based on $\mathbf{y}$ in order to generate the tentative codeword $\hat{\mathbf{C}}$, which is an estimate of $\mathbf{C}$. When the received soft value y_i associated with the i^{th} message node has a low magnitude, then this bit will have a relatively low reliability. It is readily seen that for a binary LDPC code each violated check can be corrected by flipping the state of any of the message nodes participating in this check. Based on this observation, the WBF algorithm attributes the violation of a parity check to the least reliable message node. Below we will detail the decoding steps of the WBF algorithm.

Algorithm 3 *The Weighted Bit-Flipping (WBF) algorithm is described by the following steps, which are also shown in Figure 6.1.*

1. Calculate the syndrome vector s by multiplying $\mathbf{C}$ with the transpose $\mathbf{H}^T$ of the PCM using modulo 2 operations, as in Equation 2.5. If the resultant syndrome vector s is an all-zero vector, then we claim that a legitimate codeword has been found and the iterations are terminated.

2. For each check node i, $i = 1 \cdots M$, the WBF algorithm finds the magnitude of the least reliable received message bit, which obeys:

$$y_i^{min} = \min_{t \in C_i} |y_t|, \tag{6.1}$$

   where the notation $|y_t|$ represents the absolute value of the t^{th} received message, i.e. the magnitude of the t^{th} channel soft output, while C_i represents the set of column indices of the message nodes participating in the i^{th} row of the PCM.

3. For each message node j, $j = 1 \cdots N$, the error term E_j is calculated as:

$$E_j = \sum_{t \in R_j} (2s_t - 1) y_t^{min}, \tag{6.2}$$

   where R_j denotes the row indices of the check nodes participating in the j^{th} column of the PCM. The term $(2s_t - 1)$ in Equation 6.2 indicates that when the t^{th} parity check is violated, i.e. when we have $s_t = 1$ and thus $(2s_t - 1) = 1$, then the magnitude of the least reliable received message bit y_t^{min} is added to the error term E_j as in Equation 6.2. By contrast, when a specific parity check is satisfied, then we have $(2s_t - 1) = -1$ and hence y_t^{min} is subtracted from the error term E_j of Equation 6.2.

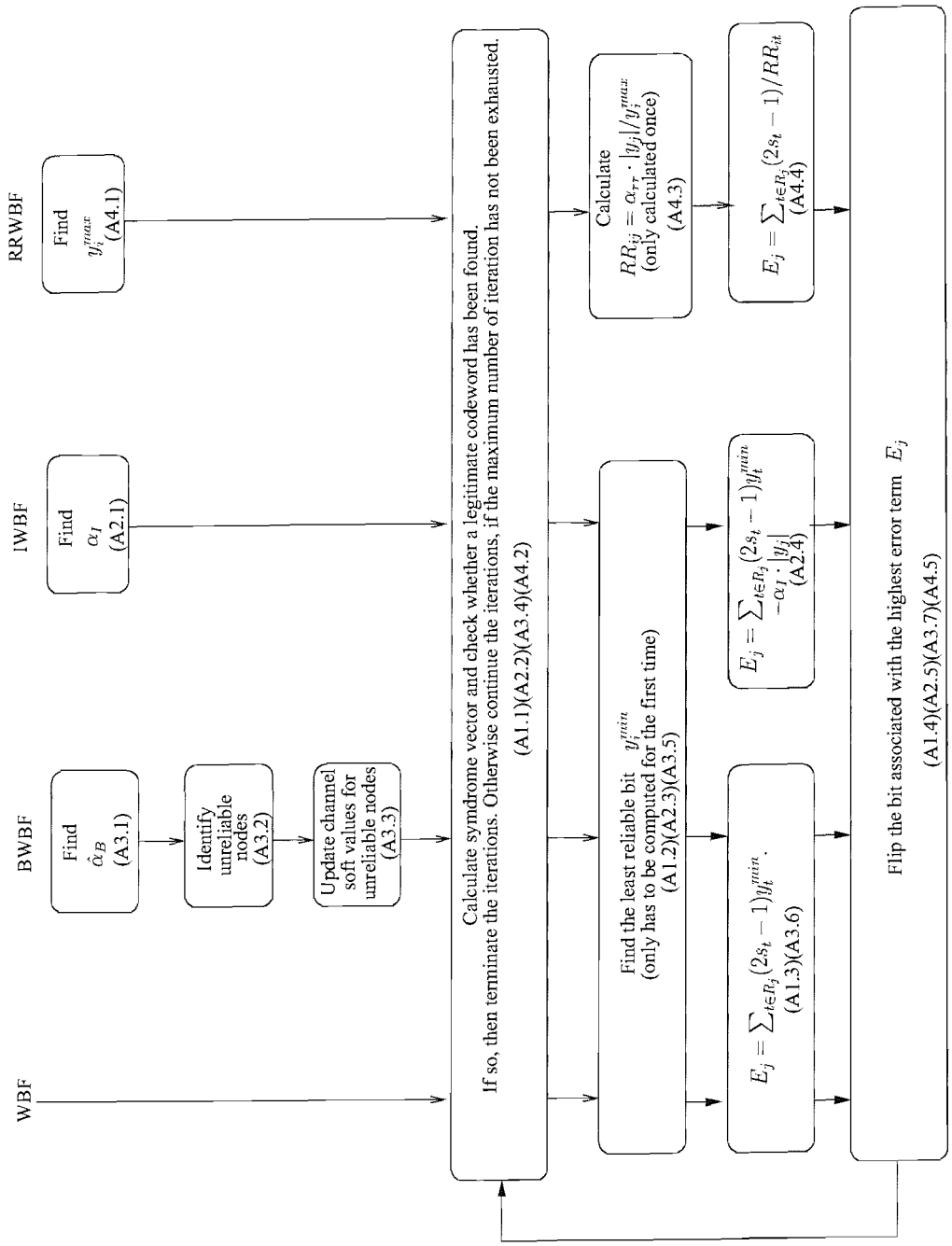


Figure 6.1: Flow chart summarising the decoding procedures of the four bit-flipping LDPC decoders considered. The notation **Ax.y** denotes the block belongs to the y^{th} step of the x^{th} algorithm introduced in Sections 6.1 to 6.4.

4. Flip the specific bit of the tentative codeword $\hat{\mathbf{C}}$ having the highest error term E_j, and repeat steps 1, 3 and 4 until the syndrome vector s becomes an all-zero vector or the maximum affordable complexity has been exhausted.

It can be observed that the real value y_j^{min} used during each iteration is calculated by the second step, which is excluded from the iteration loop of Figure 6.1. Thus the complexity of the WBF algorithm is significantly lower than that of the probabilistic decoding algorithm outlined in Section 2.5.2.

6.2 Improved weighted bit-flipping algorithm

As seen in Equation 6.2, the WBF algorithm proposed by Kou *et al.* [83] only considers the check-node based information during the evaluation of the error-term E_j. By contrast, the Improved WBF (IWBF) algorithm proposed by Zhang and Fossorier [101] enhanced the performance of the WBF algorithm, since it considered both the available check-node based and the message-node based information during the evaluation of E_j. As seen from Equation 6.2, when the error-term E_j is high, the corresponding bit is likely to be an erroneous bit and hence ought to be flipped. However, when the soft-value $|y_j|$ of a certain bit is high, the message node itself is demonstrating some confidence that the corresponding bit should not be flipped. Hence Equation 6.2 was modified in [101] as follows:

$$E_j = \sum_{t \in R_j} (2s_t - 1) y_t^{min} - \alpha_I \cdot |y_j|. \tag{6.3}$$

In Algorithm 4 , the WBF procedure algorithm will be used as a reference algorithm, and the difference of the other three algorithms in comparison to the baseline WBF algorithm will be highlighted using italics font.

Algorithm 4 *The Improved Weighted Bit-Flipping (IWBF) algorithm is described by the following steps, which are also shown in Figure 6.1.*

1. *Pre-determine the optimal threshold α_I, which will be used in the evaluation of the error term E_j of Equation 6.3.*

2. Calculate the syndrome vector s by multiplying $\hat{\mathbf{C}}$ with the transpose $\mathbf{H}^T$ of the PCM using modulo 2 operations as in Equation 2.5. If s is an all-zero vector, then a legitimate codeword has been found and the iterations are terminated.

3. As the WBF algorithm, for each check node i, $i = 1 \cdots M$, the IWBF algorithm finds the magnitude of the least reliable original coded bit, which obeys:

$$y_i^{min} = \min_{t \in C_i} |y_t|, \tag{6.4}$$

where the notation $|y_t|$ represents the absolute value, i.e. the magnitude of the t^{th} soft output of the channel.

4. *For each message node j, $j = 1 \cdots N$, the error term E_j is calculated as:*

$$E_j = \sum_{t \in R_i)} (2s_t - 1) y_t^{min} - \alpha_I \cdot |y_j|. \tag{6.5}$$

The term $(2s_t - 1)$ *in Equation 6.3 indicates that when the* t^{th} *parity check is violated, i.e. when we have* $s_t = 1$ *and thus* $(2s_t - 1) = 1$*, then the magnitude of the least reliable original coded bit* y_t^{min} *is added to the error term* E_j*, as in Equation 6.3. By contrast, when a specific parity check is satisfied, then we have* $(2s_t - 1) = -1$ *and hence* y_t^{min} *is subtracted from the error term* E_j *of Equation 6.3.*

5. Flip the specific bit of the tentative codeword $\hat{\mathbf{C}}$ having the highest error term E_j, and repeat steps 2, 4 and 5 until the syndrome vector $\mathbf{s}$ becomes an all-zero vector or the maximum affordable complexity has been exhausted.

Equation 6.3 considers the extra information provided by the message node itself, thus a message node having a higher soft-value magnitude has a lower chance of being flipped, despite having a high error term E_j owing to encountering unreliable parity checks. We note however that for LDPC codes having different column weights, or operating at different SNRs, we should weight the effect of the soft-value $|y_j|$ differently [101]. Thus, when Equation 6.3 is used for decoding a particular LDPC code, the optimum threshold value α_I should be found experimentally.

6.3 Bootstrap weighted bit-flipping algorithm

The Bootstrap WBF (BWBF) algorithm was proposed by Nouh *et al.* in [102]. Unlike the IWBF algorithm of Section 6.2, which modifies the evaluation of the error term E_i during each iteration, the BWBF algorithm pre-processes the received soft value $\mathbf{y}$, before the soft channel values are passed on to the WBF decoder. The BWBF initially compares the absolute magnitude of each message node's channel soft output to a channel-SNR dependent threshold α_B, which is determined using off-line investigations. If a certain message node has a magnitude less than the threshold value, this message node is considered to be an unreliable node and thus erased. Otherwise, the message node is considered to be a reliable node and the original channel soft value will be retained. After identifying all the unreliable message nodes, the reliability of the check nodes will be determined. A check node is deemed unreliable when there is more than one unreliable message node participating in this check, but as reliable otherwise. The soft value of the unreliable bits will be calculated as follows:

$$y'_j = y_j + \sum_{t \in R_j^r} \begin{cases} +1 \cdot min_{k \in C_t, k \neq j} |y_k| \\ -1 \cdot min_{k \in C_t, k \neq j} |y_k|. \end{cases} \tag{6.6}$$

The variable y_j in Equation 6.6 denotes the channel's soft output for the j^{th} bit, which has been classified as unreliable, while y'_j represents the updated channel soft output for the j^{th} bit. The notation R_j^r has a similar meaning to that of R_j introduced in Section 2.5.2, while the extra superscript r in Equation 6.6 indicates that only the reliable checks will be considered during the BWBF algorithm. As seen in Equation 6.6 for each check that the j^{th} message node is participating in, we have to find the magnitude of the least reliable message node, except for the j^{th} message node. Furthermore, based on the channel's soft output value corresponding to all reliable message nodes participating in the t^{th} check, a hard decision can be made for the sake of producing a sequence of binary 1s and 0s. Observe in Equation 6.6 that if we have an even number of binary 1s in the sequence, the previously found magnitude of the least reliable message node will be multiplied by a plus one. By contrast, if there is an odd number of binary 1s in the sequence obtained by hard decision, a multiplication of minus one is applied. When the soft value of the erased unreliable message nodes is replaced by the values

calculated in Equation 6.6, the new soft sequence is passed on to the WBF decoder for the sake of finding the most likely LDPC codeword. With the advent of these measures, the BWBF algorithm enhances the attainable bit error ratio performance by preventing unreliable information from being propagated through the nodes. It has been observed in [102] that at different SNRs the threshold α_B should be different. Hence we introduce the term *normalised threshold* denoted as $\hat{\alpha}_B = \alpha_B/\sigma$, where σ represents the standard deviation of the AWGN. As noted before the algorithmic steps printed in italics are those that are different from Algorithm 3.

Algorithm 5 *The Bootstrap Weighted Bit-Flipping (BWBF) algorithm is described by the following steps, which are also shown in Figure 6.1.*

1. *Determine the optimal normalised threshold $\hat{\alpha}_B$.*
2. *Determine the unreliable message nodes and check nodes.*
3. *Use Equation 6.6 to update the channel's soft output values for the unreliable message nodes.*
4. Calculate the syndrome vector s by multiplying $\hat{\mathbf{C}}$ with the transpose $\mathbf{H}^T$ of the PCM using modulo 2 operations, as in Equation 2.5. If s is an all-zero vector, then a legitimate codeword has been found and the iterations are terminated.
5. For each check node i, $i = 1 \cdots M$, the WBF algorithm finds the magnitude of the least reliable original coded bit, which obeys:

$$y_i^{min} = \min_{t \in C_i} |y_t|, \tag{6.7}$$

where the notation $|y_t|$ represents the absolute value of the t^{th} channel soft output.

6. For each message node j, $j = 1 \cdots N$, the error term E_j is calculated as:

$$E_j = \sum_{t \in R_j} (2s_t - 1) y_t^{min}. \tag{6.8}$$

The term $(2s_t - 1)$ in Equation 6.2 indicates that when the t^{th} parity check is violated, i.e. when we have $s_t = 1$ and thus $(2s_t - 1) = 1$, then the magnitude of the least reliable original coded bit y_t^{min} is added to the error term E_j, as in Equation 6.2. By contrast, when a specific parity check is satisfied, then we have $(2s_t - 1) = -1$ and hence y_t^{min} is subtracted from the error term E_j of Equation 6.2.

7. Flip the specific bit of the tentative codeword $\hat{\mathbf{C}}$ having the highest error term E_j, and repeat steps 4, 6 and 7 until the syndrome vector s becomes an all-zero vector or the maximum affordable complexity has been exhausted.

6.4 Reliability-ratio based weighted bit-flipping algorithm

It can be observed in Equation 6.2 that the WBF algorithm only considers the check-node-based information. This impediment has been improved by the IWBF algorithm in Section 6.2, which

additionally took into account the information allowing from the message-nodes by introducing the extra term $\alpha \cdot |y_j|$ in Equation 6.3. Furthermore, by pre-processing and hence enhancing the soft values of the unreliable bits, the BWBF algorithm of Section 6.3 becomes capable of improving the achievable bit error ratio performance. However, all the algorithms of Sections 6.1, 6.2 and 6.3 attribute the violation of a particular parity check to the most unreliable participating message node, while all the soft information of the other message nodes involved in this check is discarded. However, every message node involved in this specific check is contributing to its final check state. Thus, even though some of the message nodes have a higher confidence owing to their higher channel soft output magnitude, it is more accurate to state that they are less likely to violate the check. Hence, here we introduce a quantity termed as the *Reliability Ratio* (RR) , which will be used for improving the algorithm's achievable performance. As noted before, the algorithmic steps printed in italics are those that are different from Algorithm 3.

Algorithm 6 *The Reliability Ratio based Weighted Bit-Flipping (RRWBF) algorithm is implemented using the following steps, which also become explicit in Figure 6.1.*

1. *Find the magnitude of the most confident message node in a particular parity check i, which is denoted by y_i^{max}.*

2. *Calculate the Reliability Ratio $RR_{i,j}$ of the j^{th} message node involved in the i^{th} check as:*

$$RR_{i,j} = \alpha_{rr} \cdot |y_j| / y_i^{max}, \tag{6.9}$$

where α_{rr} represents a normalisation factor introduced to ensure that we have $\sum_{j \in C_i} RR_{i,j} = 1$.

3. Calculate the syndrome vector $\mathbf{s}$ by multiplying the tentatively decoded codeword $\hat{\mathbf{C}}$ by the transpose $\mathbf{H}^T$ of the PCM using modulo 2 operations, as in Equation 2.5. If $\mathbf{s}$ is an all-zero vector, then we assume that a legitimate codeword has been found and the iterations are terminated.

4. *Evaluate the error term E_i using the reliability ratio calculated in Equation 6.9 as:*

$$E_j = \sum_{t \in R_j} (2s_t - 1) / RR_{i,t}. \tag{6.10}$$

5. Flip the bit in $\mathbf{C}$ having the highest error term E_j, and repeat steps 3, 4 and 5, until the syndrome vector $\mathbf{s}$ becomes an all-zero vector or the maximum affordable complexity has been exhausted.

The RRWBF algorithm has the advantage that the real-valued variable $RR_{i,j}$ does not have to be calculated during the later iterations. Furthermore, it will be demonstrated in Section 6.6 that under certain channel conditions, the attainable performance of the RRWBF algorithm becomes superior in comparison to that of the WBF, the IWBF and the BWBF algorithms in Sections 6.1, 6.2 and 6.3.

6.5 Decoding examples

Hereby, two numerical examples will be given for illustrating the operation of the BWBF and RRWBF decoding algorithms. No worked examples are provided for the WBF algorithm, because it is embedded

in the BWBF algorithm. Furthermore, the IWBF algorithm of Section 6.2 includes a single additional term in the calculation of the error term E_i and thus it is fairly similar to the WBF algorithm, as it is seen by comparing the flow charts of these two algorithms in Figure 6.1. Hence the numerical illustration of the IWBF decoding process will be omitted.

6.5.1 Bootstrap weighted bit-flipping decoding example

The PCM illustrated in Table 2.5 will be used in this example. For the sake of simple illustration, we assume that an all-zero codeword is BPSK modulated and transmitted through an AWGN channel. The channel's soft output values can be found in Table 6.1.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
C	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
y	+1.2	+0.9	+0.6	+ 0.2	+0.7	-0.2	+1.6	+1.0	+0.7	+3.0	+2.0	+0.85	0.7	+0.6	+0.3
$\hat{\mathbf{C}}$	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0

Table 6.1: The received channel soft output values for the all-zero transmitted codeword. The first line indicates the position of each individual bit. The vector **C** indicates the original LDPC coded bits at each particular position. The vector **y** shows the channel's soft output, while the vector **z** represents the tentative decoded bits based on $\hat{\mathbf{C}}$ using hard decisions.

It may be observed from the values summarised in Table 6.1 that the sixth bit is erroneous. We will show how the BWBF algorithm corrects this error.

Upon invoking the first step of the BWBF decoding algorithm, a threshold value of $\alpha_B = 0.25$ is chosen. Hence any bit that has a magnitude lower than 0.25 will be deemed unreliable and hence recalculated, continuing the second and the third step for BWBF decoder described in Figure 6.1. Observe in Table 6.1 that the fourth and the sixth bits fall into this category. Thus we will calculate the updated soft values of these two bits using the flow chart of Figure 6.1. Since the PCM given in Table 2.5 is used for this worked example, it is reproduced here for the convenience of the reader.

		Message Nodes														
		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
	1	1	1	0	1	0	0	0	0	1	0	1	0	0	0	0
	2	0	1	0	0	0	1	1	0	1	1	0	0	0	0	0
	3	0	0	1	0	0	0	1	0	0	0	0	1	0	1	1
	4	1	0	1	0	1	0	0	1	0	1	0	0	0	0	0
Check	5	0	0	1	0	1	0	0	1	0	0	0	0	1	1	0
Nodes	6	0	0	0	0	0	1	0	1	0	0	1	0	1	0	0
	7	0	0	0	0	0	0	1	0	0	1	0	1	0	0	1
	8	0	0	0	0	1	0	0	0	0	0	0	0	1	1	1
	9	1	1	0	1	0	1	0	0	0	0	0	0	0	0	0
	10	0	0	0	1	0	0	0	0	1	0	1	1	0	0	0

Table 6.2: The PCM **H** of Table 2.5

By observing the structure of the PCM reproduced in Table 6.2, we can see by noting the positions of the 1s in the fourth column that the fourth message node is participating in the first, the ninth and the tenth parity check, while the sixth one is involved in the second, the sixth and the ninth parity check. According to the definition of reliable check nodes, we can identify that the first and the tenths parity check are reliable checks, because the fourth message node is the only unreliable message node these two check nodes are connected to, as seen by jointly considering Table 6.1 and Table 6.2. For the same reason, the second and the sixth check nodes are also reliable nodes, they are only connected to a single message node which is unreliable, namely the sixth. By contrast, the ninth check node is unreliable, since it is connected to both the unreliable fourth and the sixth message node. Hence the ninth check will not be considered during the calculation of the updated channel soft values invoking Equation 6.6 for the fourth and the sixth message node. The soft values of the above two unreliable message nodes will be calculated as follows:

$$y_4' = y_4 + 0.7 + 0.7 = 0.2 + 0.7 + 0.7 = 1.6, \tag{6.11}$$

where the first 0.7 term copied from the ninth column of Table 6.1 indicates that among all the reliable message nodes participating in the first parity check, the ninth message node has the lowest magnitude of 0.7, because all the tentatively decoded bits of the first, the second, the ninth and the 11^{th} message nodes found in Table 6.1 are zeros. Thus according to Equation 6.6, 0.7 is multiplied by +1, yielding the first 0.7 term in Equation 6.11. Following a similar philosophy, it may be readily shown that the tenth check will also provide a contribution of +0.7 to Equation 6.11. Similarly, the updated value y_6' determined for the sixth message node is calculated as follows:

$$y_6' = y_6 + 0.7 + 0.7 = -0.2 + 0.7 + 0.7 = 1.2. \tag{6.12}$$

From the results obtained in Equations 6.11 and 6.12, it can be observed that by invoking the BWBF algorithm, the soft values of the unreliable bits are enhanced. The fourth message node originally had a relatively low confidence of its own state while the sixth message node was erroneously hard decoded. After the BWBF algorithm's pre-processing, the magnitude of the fourth message node was increased, and the sixth message node has had its binary state corrected. By imposing a hard decision on the updated soft values of the two originally unreliable message nodes, a hard decision was made and hence we arrived at an all-zero codeword, which is multiplied by the transpose of the PCM in Table 6.2. An all-zero check vector is resulted, indicating that the decoded codeword is legitimate, as in the fourth step of Figure 6.1. Hence in this example, the BWBF was shown to be capable of correcting the erroneous bits found during the pre-processing stage. If the resultant vector is not an all-zero vector, further steps will be required according to Figure 6.1.

6.5.2 Reliability ratio based weighted bit-flipping Decoding Example

Our RRWBF decoding algorithm example outlined in this section will be using the same experimental data of Table 6.1 as in Section 6.5.1. Similarly, the PCM of Table 6.2 will be used in this example. According to the first step in Figure 6.1, upon multiplying the tentative decoded codeword $\mathbf{z}$ seen in Table 6.1 by the transpose $\mathbf{H}^T$ of the PCM given in Table 6.2, we arrive at the syndrome vector of {0100010010}, which is not the all-zero vector, indicating that the codeword was corrupted. Therefore

the RRWBF algorithm of Section 6.4 is used for correcting it. More explicitly, we invoke Equation 6.9 for each message node at their corresponding parity check, according to the third step of Figure 6.1. Note that the third step will only be calculated once, and in later iterations the reliability ratio will not be re-calculated. For example, the highest magnitude for all message nodes participating in the first check will be 2, which belongs to the 11^{th} message node, as seen in Table 6.1. Thus, the reliability ratio of the other message nodes used in the first parity check is found in Table 6.3.

Message Node Index j	1	2	4	9	11
Reliability Ratio	1.2/2	0.9/2	0.2/2	0.7/2	2/2
Normalised Reliability Ratio $RR_{1,j}$	0.3	0.225	0.05	0.175	0.25

Table 6.3: Normalised reliability ratios of the message nodes participating in the first parity check of Table 6.2.

By carrying out the same calculations as in Table 6.3, the corresponding reliability ratio values can be obtained at all parity checks, which are listed in Table 6.4.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	0.3	0.225		0.05					0.175		0.25				
2		0.14				0.031	0.25		0.1	0.469					
3			0.152				0.405					0.215		0.152	0.076
4	0.185		0.09		0.1			0.15		0.462					
5			0.167		0.194			0.278					0.194	0.167	
6						0.051		0.256			0.513		0.18		
7							0.278			0.52		0.148			0.052
8					0.305								0.305	0.26	0.13
9	0.48	0.36		0.08		0.08									
10				0.053					0.186		0.533	0.228			

Table 6.4: Normalised reliability ratio for each message node participating in each individual parity check using the soft channel output values given in Table 6.1.

Based on the normalised reliability ratio values calculated from Equation 6.10 and summarised in Table 6.4, the third step of the RRWBF outlined in Figure 6.1 is implemented and the error term E_1 is calculated according to Equation 6.10 as follows:

$$\begin{aligned} E_1 &= (2s_1 - 1)/RR_{1,1} + (2s_4 - 1)/RR_{4,1} + (2s_9 - 1)/RR_{9,1} \\ &= [(2 \cdot 0 - 1)/0.3] + [(2 \cdot 0 - 1)/0.185] + [(2 \cdot 1 - 1)/0.48] \\ &= -3.33 + (-5.405) + 2.083 \\ &= -6.652, \end{aligned} \tag{6.13}$$

where the variable s_i denotes the syndrome of the i^{th} parity check previously calculated upon multiplying the tentative decoded codeword $\mathbf{z}$ of Table 6.1 by the transpose $\mathbf{H}^T$ of the PCM seen in Table 6.2. Since the ninth parity check is violated owing to the erroneous bit at position six, the syndrome bit s_9 becomes +1, while the first and the fourth parity check is satisfied, hence the syndrome bits s_1 and s_4 are zeros. By carrying out the calculation of the error term E_i for the remaining message nodes of Table 6.4, the error terms E_i associated with all the message nodes are summarised in Table 6.5.

Upon identifying the specific bit having the highest error term of $E_6 = 50.29$, the sixth bit in the codeword was flipped. As seen in the final step of Figure 6.1, since the only erroneous bit is flipped to

Message Node Index j	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
E_j	-6.652	2.33	-18.74	-26.37	-13.7	50.29	-2.07	-3.39	-5.96	-1.96	-3.93	-12.5	-2.87	-16.41	-30.65

Table 6.5: The error term E_i of each individual coded bit evaluated from Equation 6.10 based on the received soft channel output value seen in Table 6.1 and on the normalised reliability ratio values summarised in Table 6.4.

the correct state, a legitimate codeword is detected owing to a resultant all-zero vector generated upon multiplying the new tentative decoded codeword by $\mathbf{H}^T$. Hence the decoding process is terminated.

6.6 Simulation results

6.6.1 Effects of the number of iterations

In the context of the probabilistic decoding algorithm described in Chapter 2, whether a bit should be flipped or not is decided in the probabilistic decoding algorithm on the basis of the updated *a posteriori* probability. Hence during each iteration the decoder is capable of flipping a certain number of bits. However, since the four hard decision based bit-flipping algorithms of Sections 6.1 to 6.4 only flip the specific bit having the highest error term E_i, the number of correctable bit errors for each iteration is only one. Thus the number of decoding iterations directly affects the attainable BER performance. Therefore, we would like to demonstrate to what extent we can improve the achievable performance upon using an increased number of decoding iterations. We will use three different-length half-rate regular LDPC codes for this experiment and the above-mentioned decoders of Sections 6.1 to 6.4 will be used in order to demonstrate their different error correction capabilities. The detailed simulation parameters are listed in Table 6.6.

It may be observed in Figure 6.2 that for the (200,100,3) LDPC code the maximum number of iterations required is less than 20. Similarly, for the (500,250,3) LDPC code characterised in Figure 6.3 and for the (1000,500,3) LDPC code evaluated in Figure 6.4, no further BER performance improvements may be attained, when the number of iterations becomes more than 40 and 80, respectively. Hence we can see in Figures 6.2 to 6.4 that when the code's blocklength is increased, more iterations are necessary to eliminate the erroneous bits in the codeword. We can see from Figures 6.2 - 6.4 that setting the maximum number of iterations to 10% of the coded blocklength may be deemed sufficiently high for fully exploiting the decoding power of the decoder. Thus in our future investigations we opted for setting the number of iterations to 10% of the LDPC code's blocklength. Furthermore, we can observe from Figures 6.2 - 6.4 that when only a low number of iterations is allowed, the BWBF algorithm achieves the best performance. In other words, the BER curves of the BWBF algorithm do not exhibit as high iterative gains as the other three counterparts. This is because during the initialisation process of the BWBF algorithm, a number of errors are already corrected, which is in contrast to the other three bit-flipping algorithms, which are only capable of correcting errors during the iterative decoding process.

Modem	BPSK
Channel	AWGN
LDPC Code	(200,100,3)
Maximum Number of Iterations	5, 10, 20, 40, 60, 80, 100
LDPC Code	(500,250,3)
Maximum Number of Iterations	10, 20, 40, 60, 80, 100
LDPC Code	(1000,500,3)
Maximum Number of Iterations	10, 20, 40, 60, 80, 100
IWBF optimum α_I	0.4
BWBF threshold $\hat{\alpha}_B$	0.5

Table 6.6: Simulation parameters for three different-length half-rate regular LDPC codes. The LDPC coded bitstreams are BPSK modulated and are transmitted over an AWGN channel while invoking various maximum number of iterations. The IWBF algorithm used $\alpha_I = 0.4$ [101] and the normalised BWBF algorithm's threshold of 0.5 was taken from [102].

6.6.2 Reliability of the bit flipping algorithms

Since all the four above mentioned bit-flipping algorithms of Sections 6.1 - 6.4 are based on flipping the bits having the highest error term E_i, we will now investigate how accurate or reliable these algorithms are in terms of locating the erroneous bits by resorting to the calculation of E_i. When a particular bit is flipped during the simulation, the updated state of this bit will be compared to the original coded bit. If these two states are identical, then this flip will be classified as a correct one. Otherwise the flip is classified as an incorrect one. We will quantify the percentage of both the correct bit flipping operations and the fraction of incorrect bit flipping actions upon normalising them to the total number of bits that have been flipped. The simulations were carried out using BPSK modulation when communicating over an AWGN channel. A half-rate regular (500,250,3) LDPC code was used and the results are shown for the WBF, the IWBF, the BWBF and the RRWBF algorithms. The maximum number of iterations was set to 60.

It can be observed from Figure 6.5 that in the high-SNR region the bit-flipping algorithm is quite accurate in terms of locating the position of the erroneous bit. However, when the SNR is low, the IWBF algorithm and the RRWBF algorithm are superior in comparison to the other two owing to additionally considering the message-node-based soft information. The WBF and the BWBF algorithms do not perform particularly well in the low-SNR region, because when the SNR is low, a check is often violated by more than one erroneous bits. Hence, the calculation of the error term E_i expressed in Equation 6.2 is not sufficiently reliable, when only considering the check-node based information.

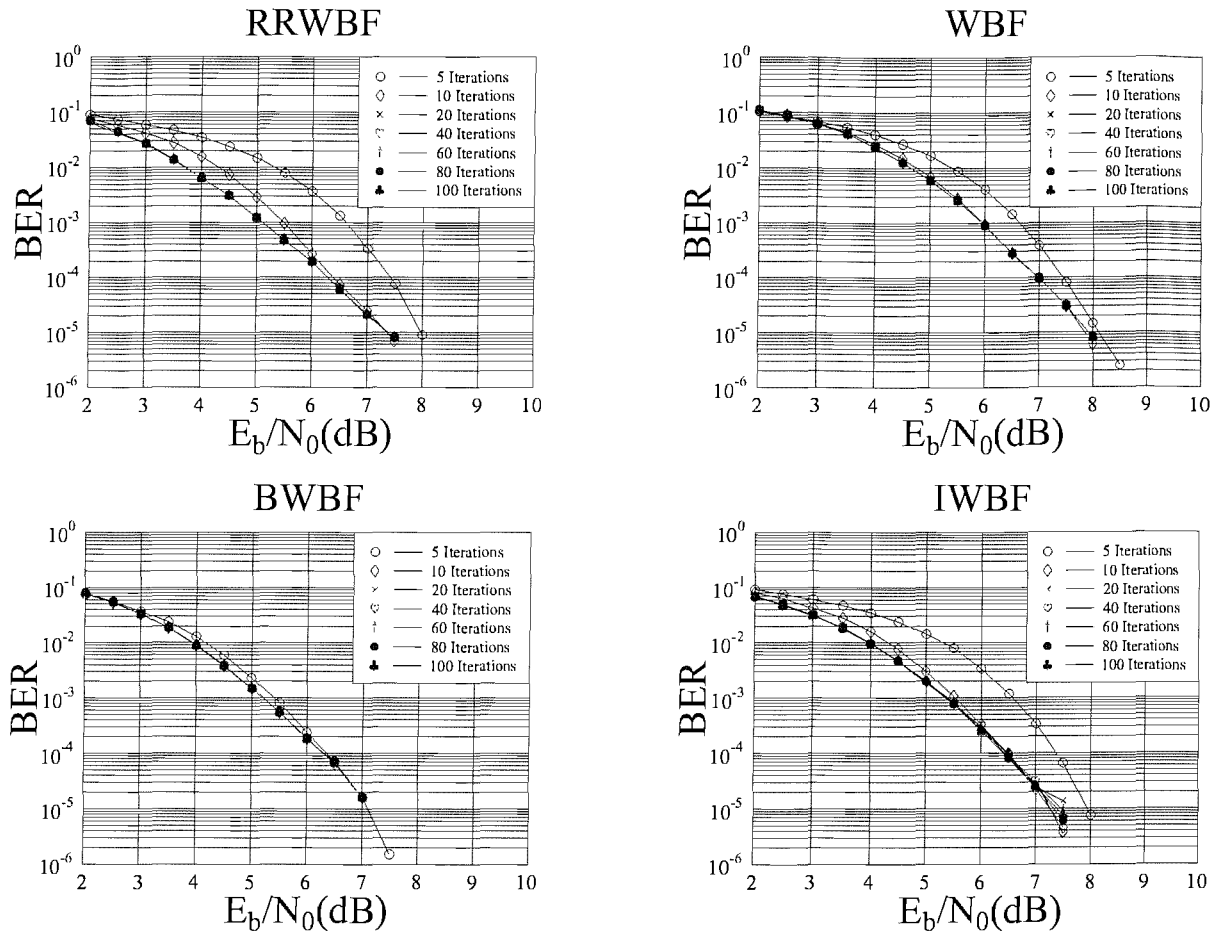


Figure 6.2: BER performance of the (200,100,3) regular LDPC code decoded by the WBF decoder, the IWBF decoder, the BWBF decoder and the RRWBF decoder, when communicating over an AWGN channel using BPSK modulation. The achievable coding gain of the various schemes at a BER of 10^{-4} will be summarised in Table 6.11.

6.6.3 Effects of the various blocklengths

As seen in Chapter 2, the blocklength of the LDPC code is important, when using the probabilistic decoding algorithm of Section 2.5.2 owing to the associated increased minimum distance. In this subsection we will investigate the associated performance trends, when the blocklength of a half-rate LDPC code is increased. The corresponding simulation parameters are listed in Table 6.7.

Observe in Figures 6.6, 6.7 and 6.8 that no significant coding gain can be achieved upon increasing the blocklength when communicating over AWGN or uncorrelated Rayleigh-fading channel. By contrast, it is expected that over correlated fading channels longer codes will tend to perform better owing to their ability to cope with bursty channel errors. A BER performance comparison of the four bit-flipping algorithms is offered in Figure 6.9 for transmission over an AWGN channel. The RRWBF algorithm achieved the best performance, with BWBF algorithm obtaining a marginally inferior performance.

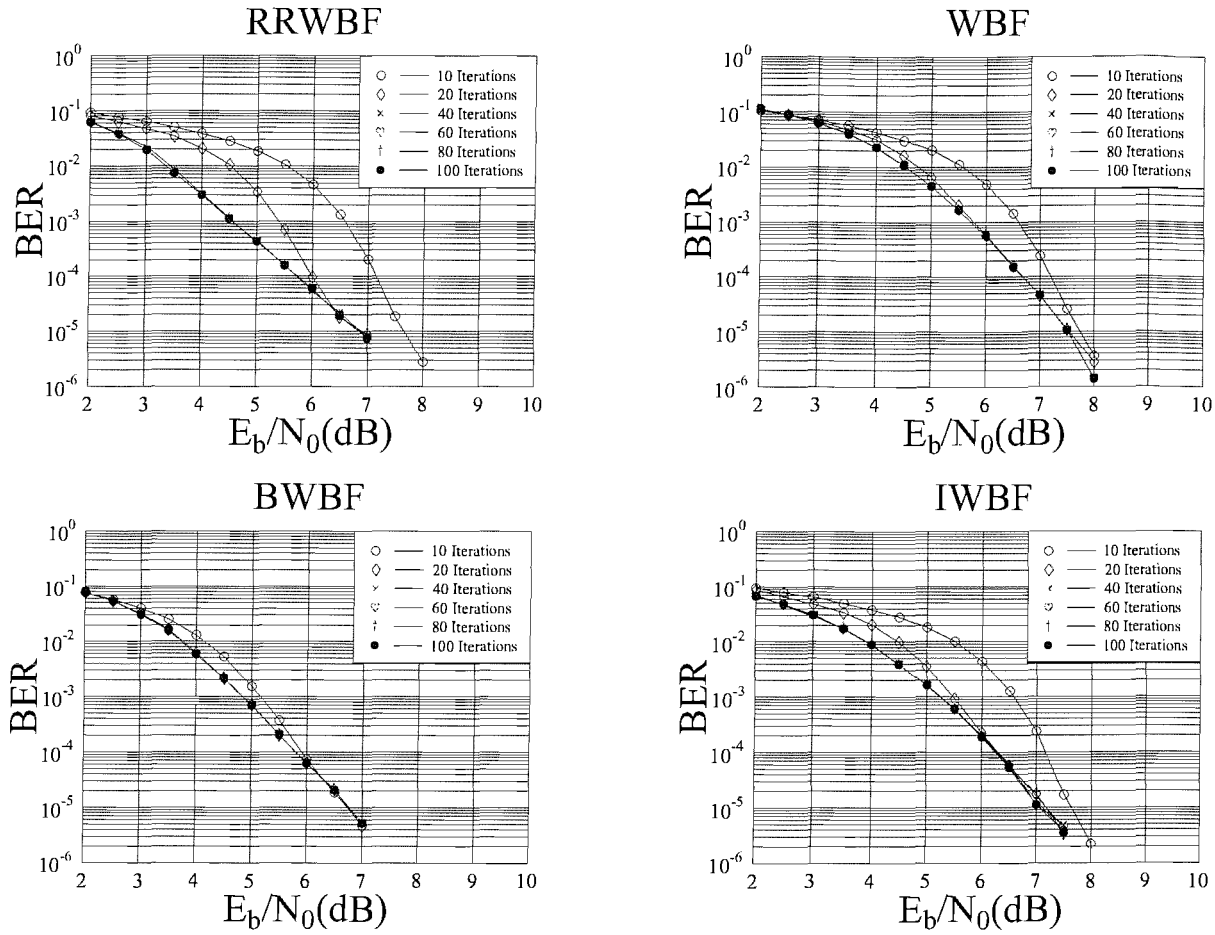


Figure 6.3: BER performance of the (500,250,3) regular LDPC code decoded by the WBF decoder, the IWBF decoder, the BWBF decoder and the RRWBF decoder, when communicating over an AWGN channel using BPSK modulation. The achievable coding gain of the various schemes at a BER of 10^{-4} will be summarised in Table 6.11.

6.6.4 Effects of using various code rates

In this subsection, the BER performance of five regular LDPC codes having the same coded blocklength of 900 bits, but having a different code rate will be investigated, when communicating over AWGN channels. The simulation parameters are listed in Table 6.8.

When the code rate is varied in the AWGN channel scenario considered, we observe from the results shown in Figure 6.10 that the higher the code rate, the higher the coding gain. This is because the slight SNR loss of higher coding rates is outweighed by their advantage of having significantly less parity bits. When we consider the uncorrelated fading channel, the performance trends associated with varying the code rate have changed, as evidenced by Figure 6.12. For the WBF, IWBF and BWBF decoder, the achievable BER performance becomes poorer, as the code rate is increased. By contrast, for the RRWBF decoder, the coding gain versus coding rate curve peaks around $r = 0.75$ and then it decays around $r = 0.8$. We can also see in Figure 6.12 that the performance of the RRWBF is in fact the worst in the uncorrelated Rayleigh fading channel at low code rates. This is because in the above experiments we were using a column-weight of three for all the LDPC decoders. However,

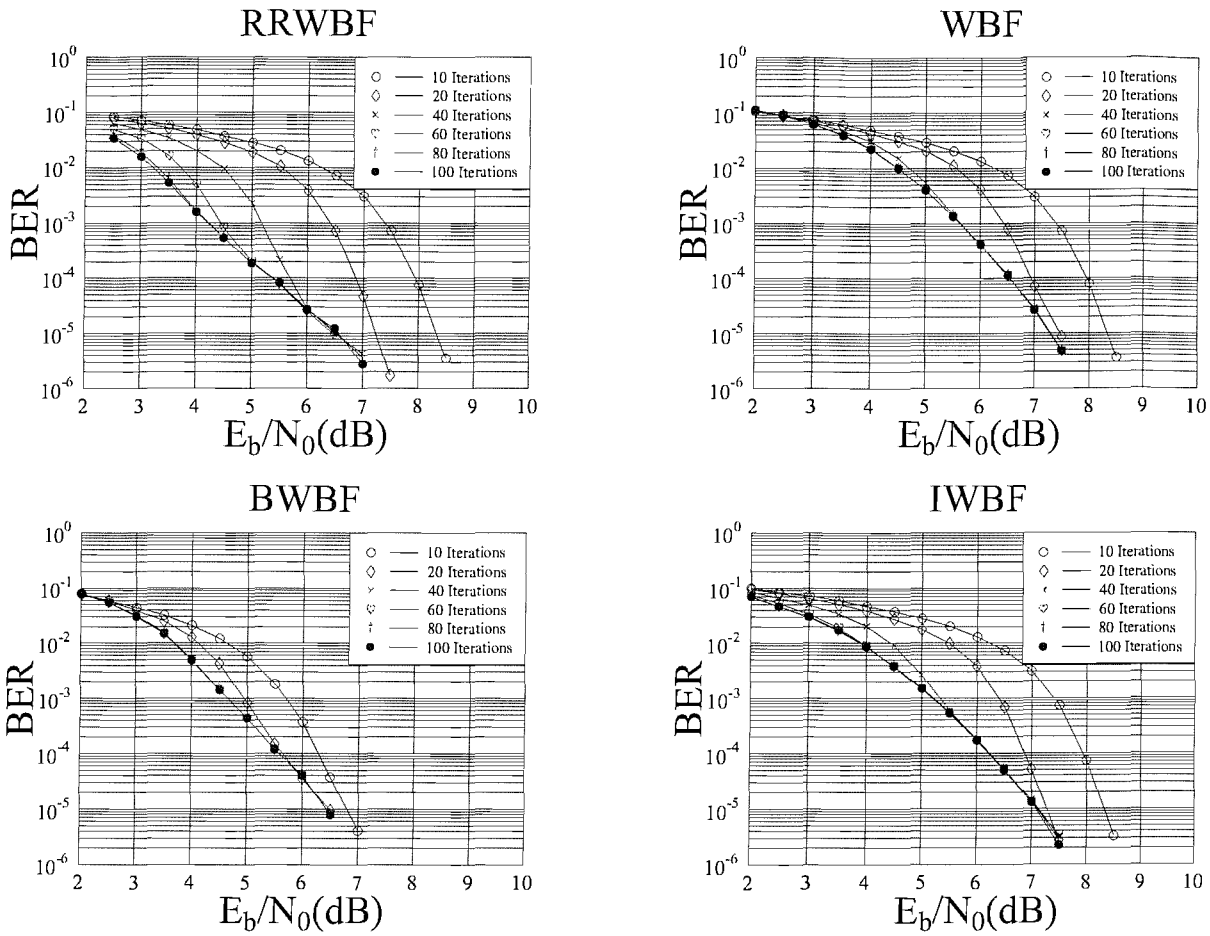


Figure 6.4: BER performance of the (1000,500,3) regular LDPC code decoded by the WBF decoder, the IWBF decoder, the BWBF decoder and the RRWBF decoder, when communicating over an AWGN channel using BPSK modulation. The achievable coding gain of the various schemes at a BER of 10^{-4} will be summarised in Table 6.11.

since the RRWBF decoder calculates the reliability ratio based on all the participating message nodes for a given check node, the decoder would benefit from having more samples for the sake of having a more accurate reliability ratio calculation. We can see that since in an AWGN channel the channel conditions are benign, the RRWBF decoder was capable of confidently calculating the reliability ratio. However, when the channel becomes more hostile in a fading channel scenario and when a low code rate is desired, the row weight of the LDPC code's PCM becomes relatively low in comparison to that of a higher rate LDPC code. Hence, upon increasing the code rate the RRWBF becomes capable of calculating the reliability ratio values more accurately. Observe however that in Figure 6.12 a lower coding gain was achieved at $r = 0.8$ in comparison to $r = 0.75$, indicating that there is a trade-off between providing more accurate information for the RRWBF to calculate the reliability ratio and the associated error correction capability of the corresponding code rate.

There is another way of improving the RRWBF decoder's confidence in calculating the reliability ratio. Specifically, we can increase the column weight proportionately to the increase of the row weight. Furthermore, upon increasing the column weight the minimum distance of the LDPC code

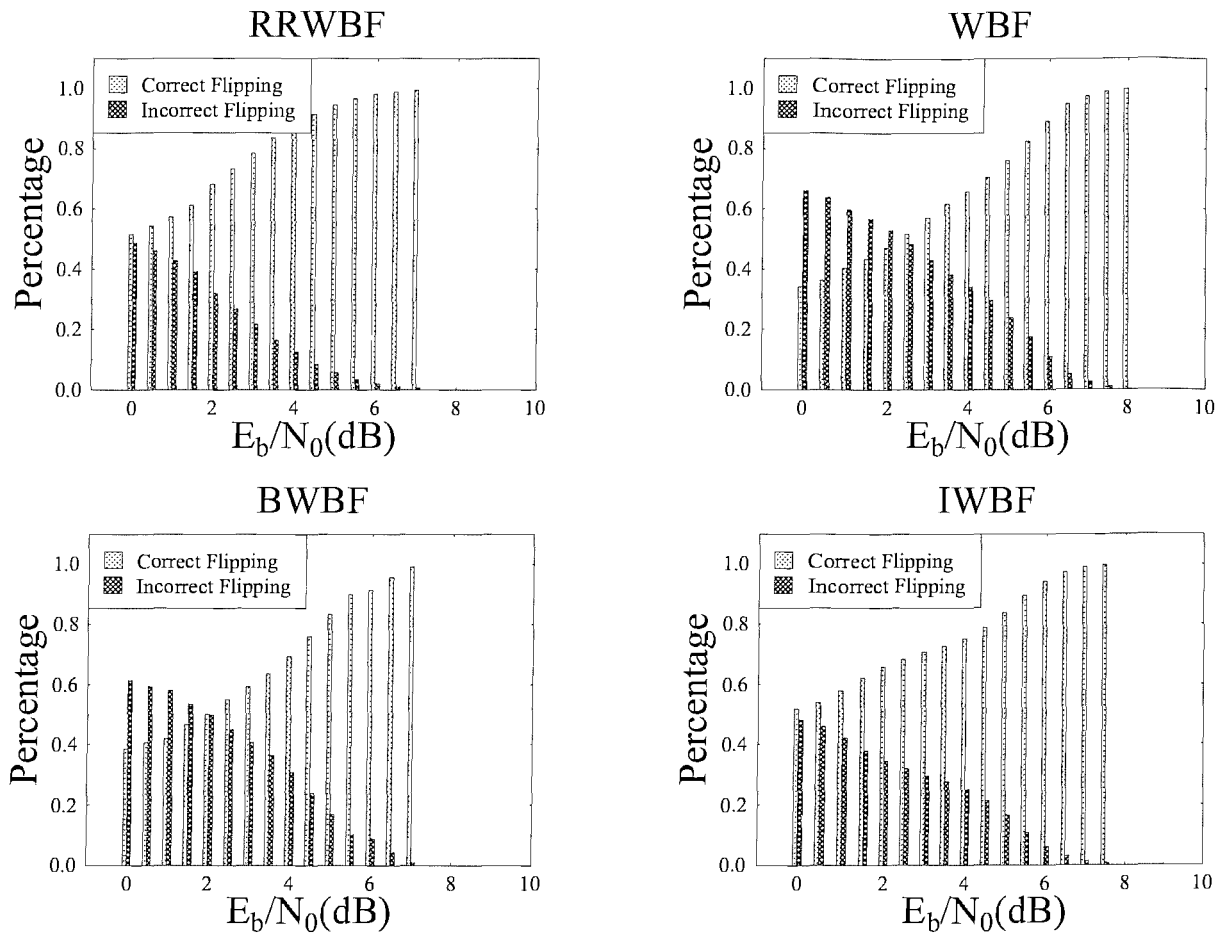


Figure 6.5: Percentage of correct bit-flipping and incorrect bit-flipping for a half-rate regular (500,250,3) LDPC code, when communicating over an AWGN channel using BPSK modulation. The maximum number of iterations used for each decoder is 60.

is also increased. In Figure 6.13, we portray the BER versus E_b/N_0 results for a (1000,500,5) LDPC code when communicating over uncorrelated Rayleigh fading channel, while having an average LDPC PCM column weight of five.

To elaborate a little further, it can be seen from Figure 6.13 for a half-rate LDPC code having a codeword length of 1000 bits transmitted over an uncorrelated Rayleigh fading channel, that increasing the average column weight of the LDPC is beneficial for the RRWBF algorithm in terms of being capable of computing the reliability ratio more accurately. In Figure 6.14 the experiments conducted were similar to those in Figure 6.13 for the four bit-flipping decoders in Sections 6.1 to 6.4 having various coding rates. The coding gains achieved at a target BER of 10^{-4} are plotted in Figure 6.15. It becomes clear from Figure 6.15 that when the LDPC code's column weight is increased in proportion to the row weight, while aiming for a low coding rate, the RRWBF algorithm constitutes the best design option. However, when the code rate is increased, the error correction capability of the RRWBF decoder in Section 6.4 decays faster than that of the other bit-flipping decoders introduced in Sections 6.1 to 6.4. For $r > 0.66$ and an average column weight of five, the BWBF decoder and the IWBF decoder of Sections 6.3 and 6.2 offered the highest coding gain.

Modem	BPSK
Channel	AWGN Uncorrelated Rayleigh Fading
LDPC Code	(200,100,3) (400,200,3) (600,300,3) (800,400,3) (1000,500,3)
Maximum Number of Iterations	10% of the coded blocklength

Table 6.7: Simulation parameters for five half-rate regular LDPC codes having various coded blocklengths.

Modem	BPSK
Channel	AWGN Uncorrelated Rayleigh Fading
Code	(900,300,3) (900,450,3) (900,600,3) (900,675,3) (900,720,3)
Maximum Number of Iterations	90

Table 6.8: Simulation parameters for five regular LDPC codes having various code rates at a coded blocklength of 900 bits.

Interestingly, if we compare the results of Figure 6.12 and Figure 6.15, we find that at a low code rate the performance of all the bit-flipping decoders improved when a higher column weight was invoked. However, the achievable error correction capability was impaired by increasing the column weight, when the code rate was high, and simultaneously also the row weight was high. More explicitly, in the context of bit-flipping decoders a higher row weight was capable of providing more confident information for the RRWBF decoder during the calculation of the reliability ratio. However, when the row weight is set to a high value, it may result in confusing the decoder, because the violation of a check might be inflicted by an increased number of unreliable message nodes. The decoder has to choose a specific message node for bit flipping based on the calculated error term E_i from a large number of participating message nodes. However, its decision might not be sufficiently accurate owing to the low channel quality experienced. Hence, the employment of a lower column weight is suggested for high code rate applications, whilst opting for a slightly higher column weight may be suitable when the code rate employed is low.

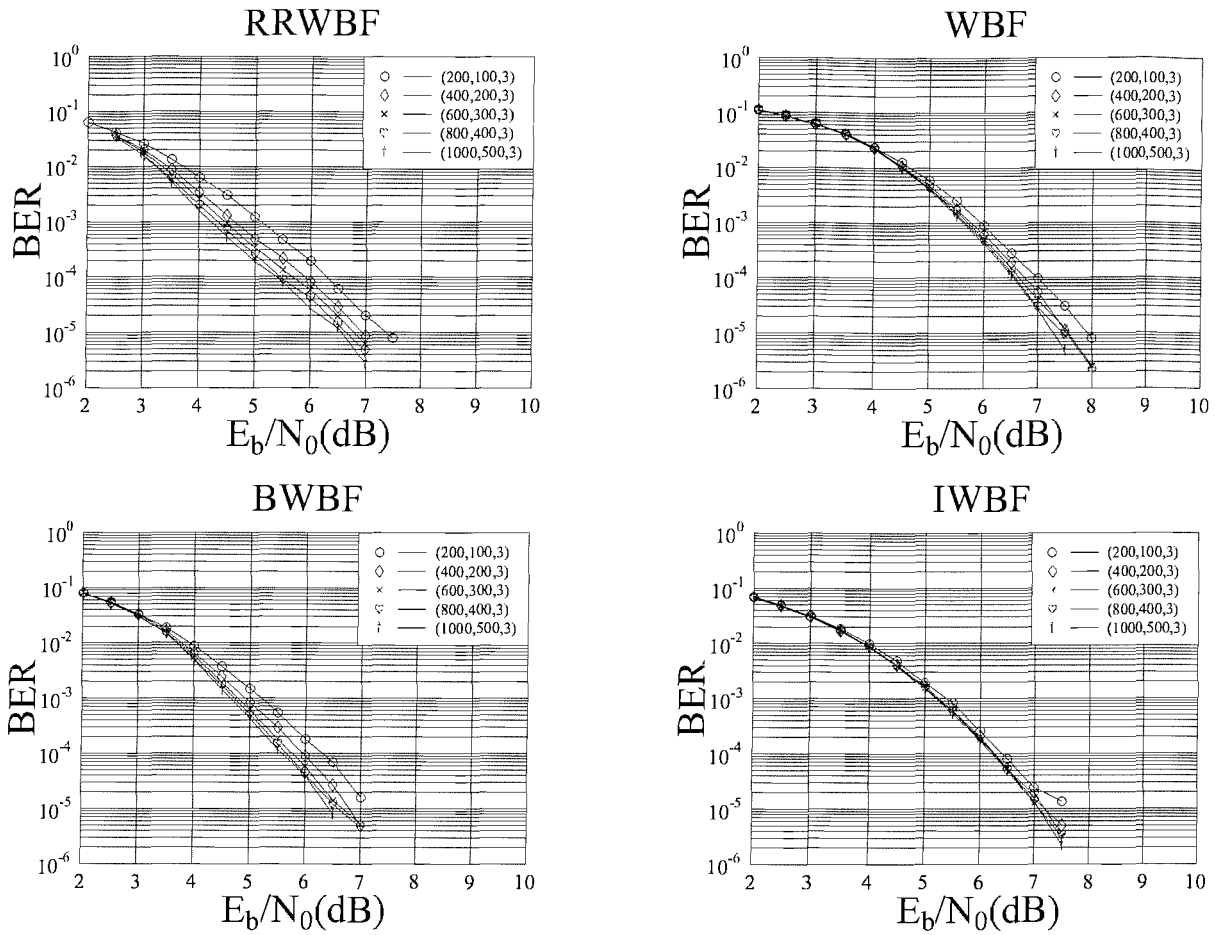


Figure 6.6: BER performance of the five different-length half-rate regular LDPC codes listed in Table 6.7, which are decoded by the WBF decoder, the IWBF decoder, the BWBF decoder and the RRWBF decoder, when communicating over an AWGN channel using BPSK modulation. The achievable coding gain of the various schemes at a BER of 10^{-4} will be summarised in Figure 6.8 and Table 6.11.

6.6.5 Performance comparisons against MAP decoding

It becomes clear from the results of Section 6.6.4 that using a code rate around $r = 0.66$ has the highest coding gain when communicating over an AWGN channel. By contrast, for a soft decoder of length 3000 bits, the best coding gain was achieved at $r = 0.33$, as seen in Figure 2.20. Hence we now continue our discussions by comparing the performance of the family of bit-flipping decoders to that of the soft decision aided probabilistic MAP decoder of Section 2.5.2 at different coding rates. The associated simulation parameters are listed in Table 6.9.

It can be observed in Figure 6.16 that as the code rate increases, the BER performance of the LDPC codes studied does not suffer from dramatic degradation when decoded by the hard-decision-based bit-flipping decoders of Sections 6.1 - 6.4. By contrast, the BER performance of the probabilistic decoder of Section 2.5.2 is degraded, when the code rate is increased. For example, at a code rate of $r = 0.8$, the performance of the probabilistic decoder is only about 1.5dB better than that of the RRWBF decoder. In the next section, we will compare the decoding complexity of different decoders.

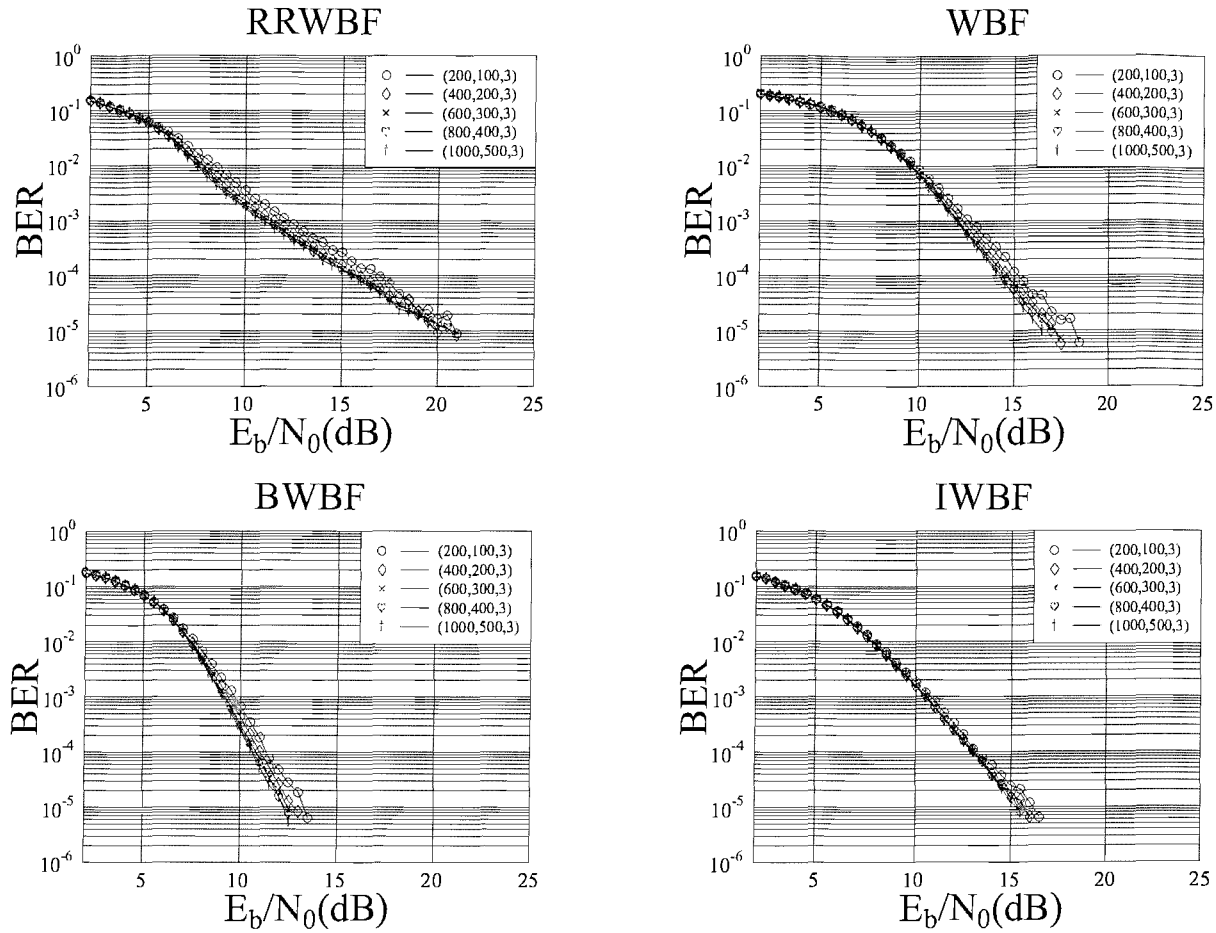


Figure 6.7: BER performance of the five different-length half-rate regular LDPC codes listed in Table 6.7, which are decoded by the WBF decoder, the IWBF decoder, the BWBF decoder and the RRWBF decoder, when communicating over an uncorrelated Rayleigh fading channel using BPSK modulation. The achievable coding gain of the various schemes at a BER of 10^{-4} will be summarised in Figure 6.8 and Table 6.11.

6.7 Decoding complexity

In this section, the complexity of the above mentioned four bit-flipping based decoders of Sections 6.1 - 6.4 will be quantified. In each WBF algorithm iteration the specific bit having the highest error term E_j will be flipped. The flipping of this particular bit will consequently toggle the state of w_c parity checks. Furthermore, each parity check's state change will affect the value of the error term E_j participating in this check, hence the w_r error terms E_j associated with the message nodes and participating in this parity check have to be recalculated. Thus a total of $w_c \cdot w_r$ error terms have to be recalculated. The notation w_c and w_r has the same meaning as in Chapter 2, which represent the average column weight and average row weight of the PCM. When the WBF algorithm is invoked, the number of additions required is w_c, as seen by considering Equation 6.2. Hence $w_c^2 \cdot w_r$ additions are required by the WBF algorithm during each iteration.

The IWBF algorithm is similar to the WBF algorithm as shown in Figure 6.1, but according to

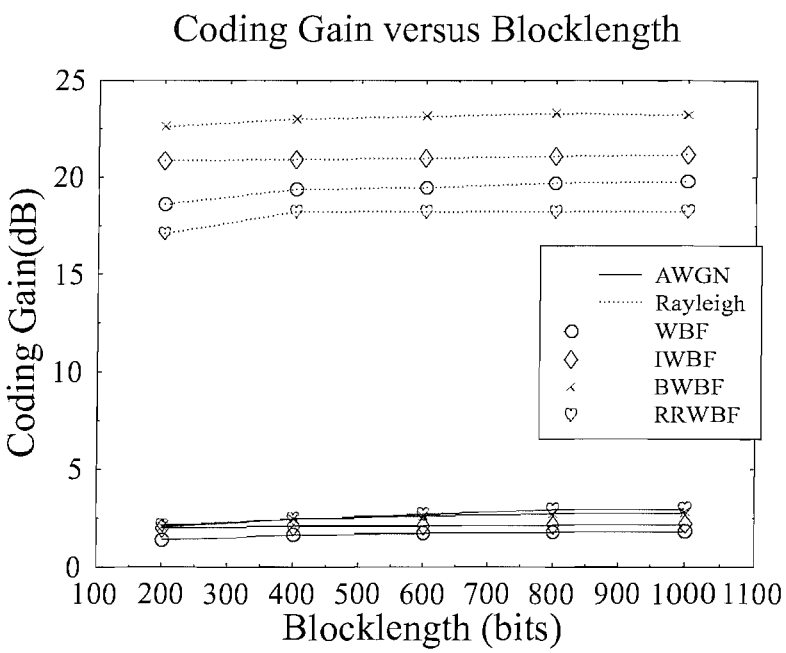


Figure 6.8: Coding gain versus blocklength performance at a target BER at 10^{-4}, extracted from the results shown in Figures 6.6 and 6.7, when communicating over both AWGN and uncorrelated Rayleigh fading channel.

Equation 6.3 the weighting factor α_I has to be additionally multiplied by the magnitude of each bit and it has to be subtracted from the previously calculated E_i term. Hence the IWBF will require $w_c \cdot w_r$ extra additions and $w_c \cdot w_r$ extra multiplications during each iteration.

By the same token, the BWBF algorithm operates exactly the same way as the WBF algorithm, as in Figure 6.1. Hence the complexity associated with each iteration is the same as that of the WBF algorithm.

The RRWBF algorithm only slightly modifies the WBF algorithm, since the reciprocal of the reliability ratio is used instead of using the minimum soft value within a specific row of the PCM, as seen in Figure 6.1 and Algorithm 6. Referring back to Section 4.3.5, the complexity of the probabilistic decoder was found to be $2w_cq$ additions and $7w_cq$ multiplications per coded bit per iteration. For the (900,720,3) code characterised in Figure 6.16, the complexity of all the decoders involved in decoding a codeword is summarised in Table 6.10. The (900,720,3) code associated with $w_c = 3$, $w_r = 15$ and $N = 900$ was investigated and the bit-flipping decoder invoked 90 iterations, while the probabilistic decoder employed 15 iterations, where the specific number of iterations for the different decoders was chosen for the sake of fully exploiting the decoding power of the different decoders.

It becomes explicit from Table 6.10 that the complexity of the bit-flipping algorithms is less than 50% compared to that of the probabilistic decoder in terms of the number of additions required. Furthermore, in terms of multiplications, the probabilistic decoder exhibits a significantly higher decoding complexity than the rest of the decoders. Since a multiplication is deemed to impose a higher complexity than an addition, it can be concluded that the probabilistic decoder is significantly

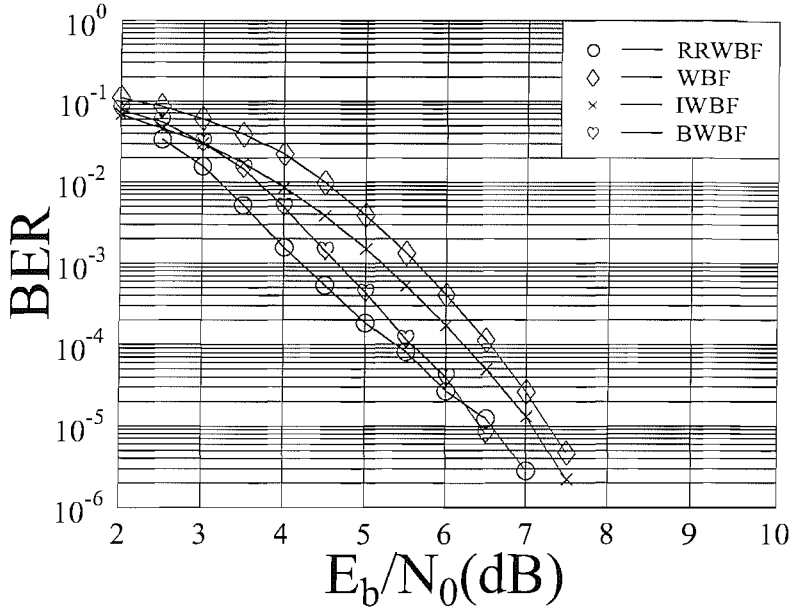


Figure 6.9: BER performance of a (1000,500,3) LDPC code decoded by the four bit-flipping algorithms, when communicating over an AWGN channel. The simulation parameters are listed in Table 6.7.

more complex than the bit-flipping decoders considered in Sections 6.1 - 6.4.

6.8 Summary and conclusion

In this chapter, we introduced a hard-decision based approach for decoding LDPC codes. The WBF, the IWBF and the BWBF algorithms were introduced and their advantages as well as disadvantages were identified. A novel reliability ratio based WBF algorithm was proposed and it was compared to the other three algorithms. Hereby, we would like to conclude by providing a table summarising the achievable coding gain versus various parameters extracted from the Figures provided in Section 6.6.

As seen in Table 6.11, the major results of this chapter are summarised in terms of the achievable coding gain at a target BER of 10^{-4}. It can be observed in Table 6.11 that as expected, in an AWGN channel the required number of iterations is proportional to the codeword blocklength, which is plausible on the basis of the fact that each iteration allows the correction of a single bit. The BWBF algorithm achieves the highest coding gain during the first few iterations owing to the pre-processing of the unreliable message nodes. However, provided that a sufficiently high number of iterations has been carried out, the RRWBF achieved the highest coding gain compared to the other three algorithms. We can observe in Table 6.11 that by using a number of iterations which is about 10% of the codeword length will be sufficient for fully exploiting the error correction power of the decoder. Increasing the codeword length did not achieve a significant BER improvement in either AWGN or uncorrelated Rayleigh fading channels. Observe in Table 6.11 that when the rate of the LDPC code is varied in the AWGN scenario and an average column weight of three is chosen, the highest coding gain

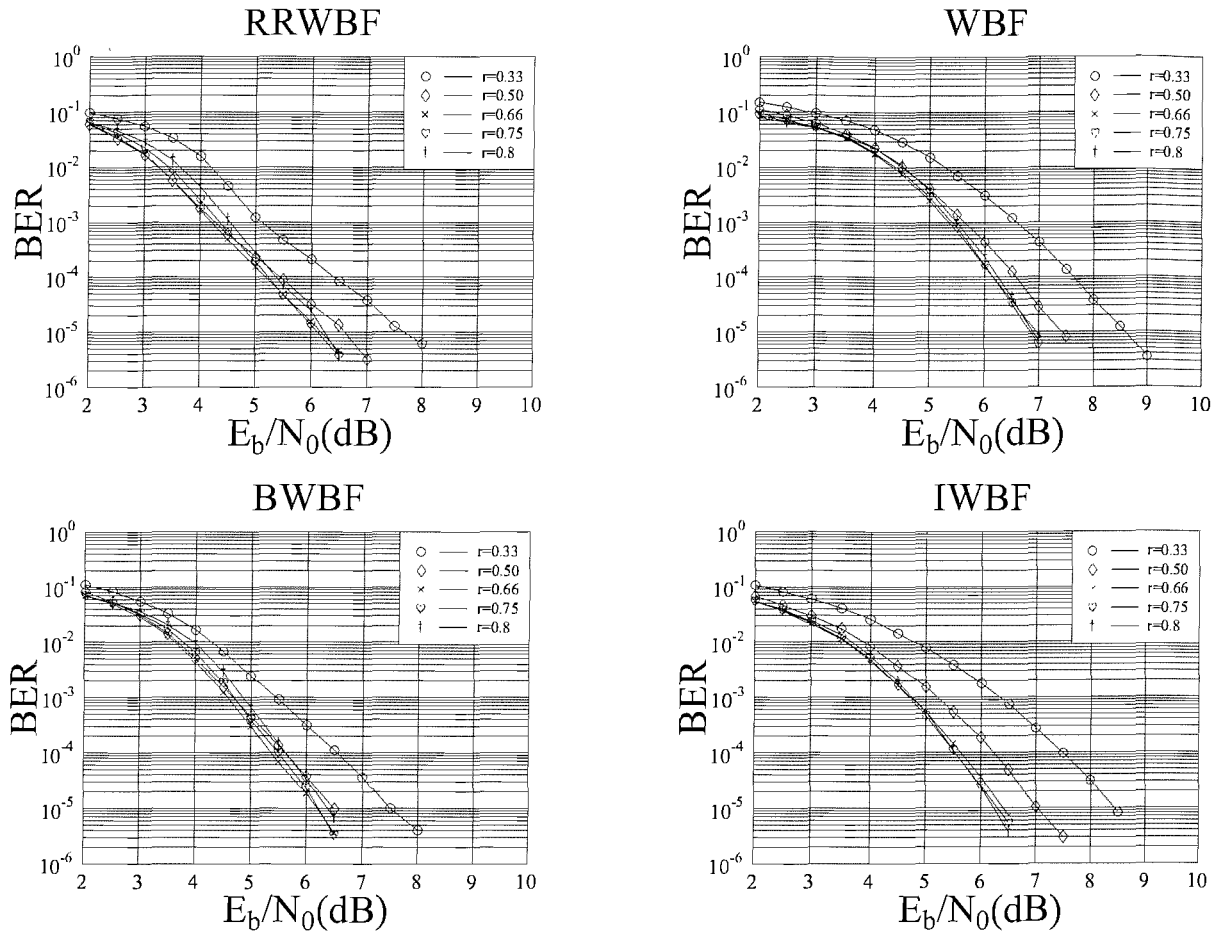


Figure 6.10: BER performance of five regular LDPC codes having various code rates as listed in Table 6.8, decoded by the WBF decoder, the IWBF decoder, the BWBF decoder and the RRWBF decoder, when communicating over an AWGN channel using BPSK modulation. The achievable coding gain of the various schemes at a BER of 10^{-4} will be summarised in Figure 6.12 and Table 6.11.

is achieved, when the code rate is high. However, as portrayed in Table 6.11, when communicating over uncorrelated Rayleigh fading channels, this performance trend was reversed in comparison to the AWGN scenario. As evidenced in Figure 6.11, the best BER versus E_b/N_0 performance is achieved at a low code rate. Observe in Figure 6.8 that when a column weight of three is used, the RRWBF algorithm achieved the best coding gain performance, when communicating over an AWGN channel. By contrast, it constitutes the worst performance in terms of coding gain, when the uncorrelated Rayleigh fading channel is considered. As detailed in Section 6.6.4, the RRWBF's performance relies on the accuracy of the reliability ratio calculation. Therefore, when communicating over Rayleigh fading channels, a higher row weight is desired for a better estimation of the reliability ratio of the message nodes for this particular check. However, there is a trade-off between calculating the reliability ratio more accurately and having more message nodes participating in a single parity check, owing to the increased number of message nodes participating in a violated parity check. As observed in Figure 6.14 that when the average column weight of the LDPC code was increased from three to five, the RRWBF algorithm achieved the best performance at a coding rate of $r = 0.5$. However, owing to having a high number of message nodes participating in a parity check when the column weight is

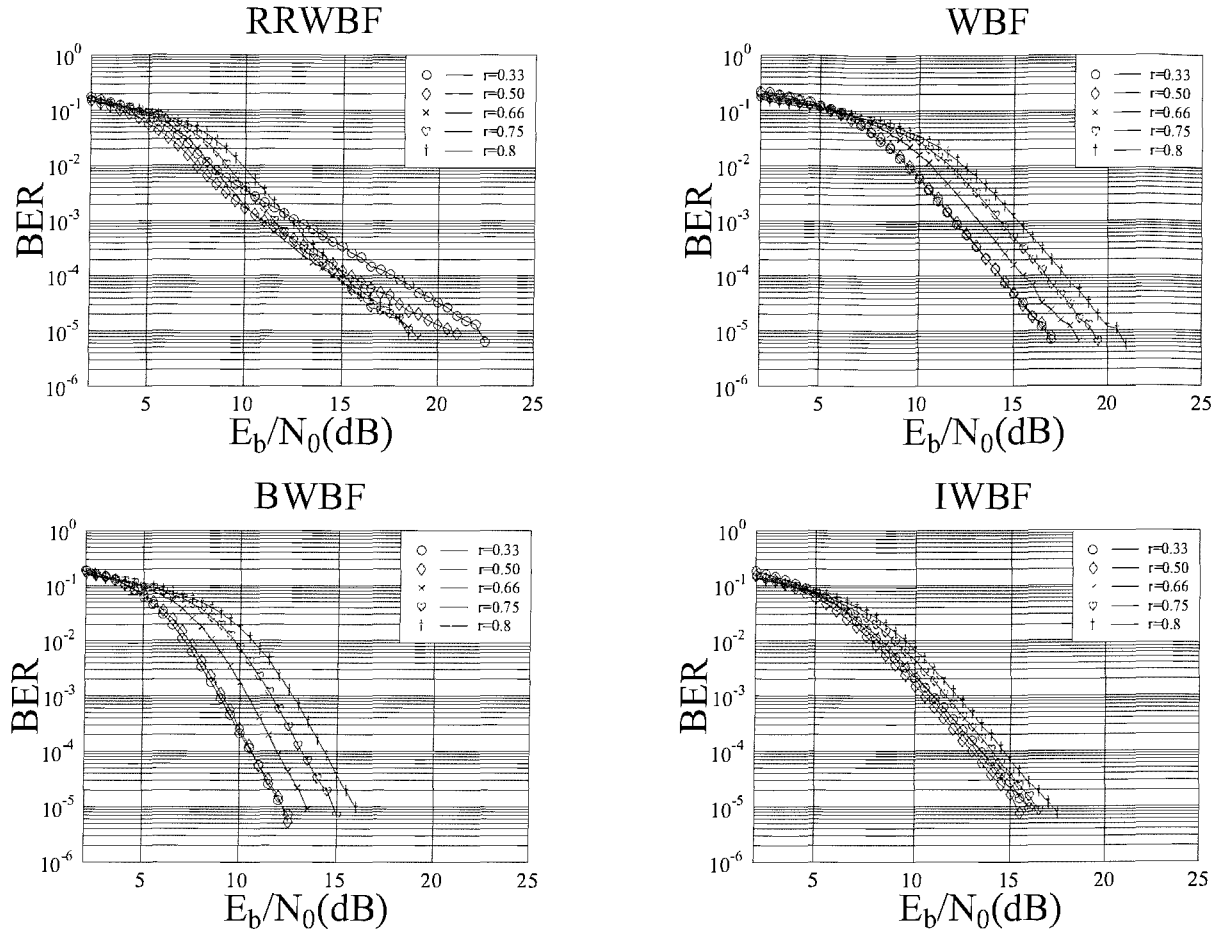


Figure 6.11: BER performance of five regular LDPC codes having various code rates as listed in Table 6.8, decoded by the WBF decoder, the IWBF decoder, the BWBF decoder and the RRWBF decoder, when communicating over an uncorrelated Rayleigh fading channel using BPSK modulation. The achievable coding gain of the various schemes at a BER of 10^{-4} will be summarised in Figure 6.12 and Table 6.11.

five and the coding rate is high, the coding gain performance of the RRWBF algorithm degrades the most dramatically, when the code rate is increased to $r = 0.8$. Hence, it is beneficial for the RRWBF algorithm operating under hostile channel conditions to have a row weight of approximately eight to ten, which will strike a good compromise between achieving an accurate reliability ratio calculation and having an excessive number of message nodes participating in a parity check.

In conclusion, the bit-flipping algorithms were shown to be capable of maintaining a significantly lower decoding complexity compared to the probabilistic decoder of Section 2.5.2, at a E_b/N_0 degradation of about 1.5dB, as demonstrated in Figure 6.16 for a coding rate of $r = 0.8$ while invoking the RRWBF decoding algorithm. The proposed RRWBF algorithm of Section 6.4 constitutes an attractive design alternative to the existing bit-flipping algorithms. At a column weight of three and when communicating over an AWGN channel, the RRWBF decoder exhibited a better BER performance than the other BF decoders in many scenarios, although while sometimes it was marginally outperformed by the BWBF algorithm. When operating in fading channels, the row weight of the LDPC code's

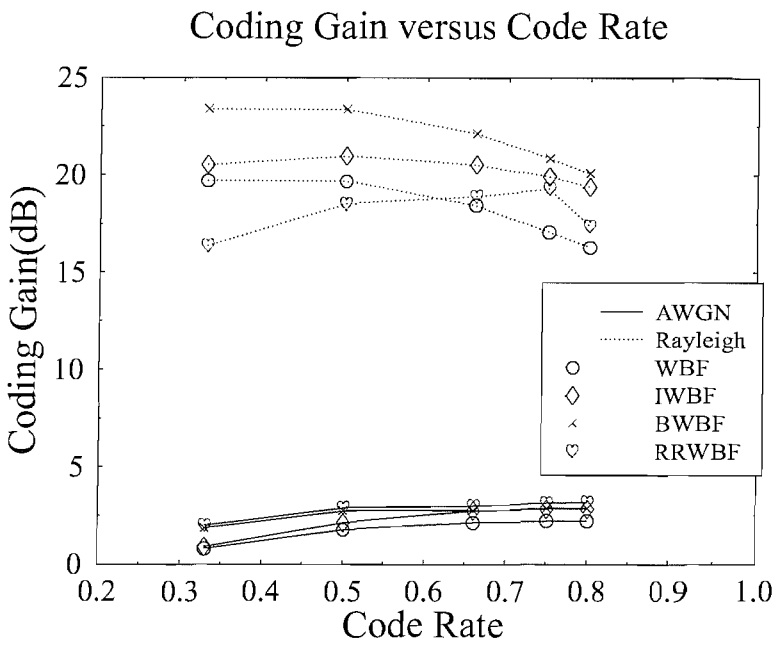


Figure 6.12: Coding gain versus code rate performance at a target BER at 10^{-4}, extracted from the results shown in Figures 6.10 and 6.11, when communicating over both AWGN and uncorrelated Rayleigh fading channel.

PCM has to be selected appropriately for the sake of computing the reliability ratio more accurately. When the code rate is varied as shown in Figures 6.12 and 6.15, two different performance trends are observed for the RRWBF algorithm. When a column weight of three is used, the BER performance becomes better upon increasing the code rate to $r = 0.75$ and then decades. By contrast, when a column weight of five is used, the BER performance becomes better at a low code rate, while degrading, when increasing the code-rate up to $r = 0.8$. This suggests that the row weight of the LDPCC should be carefully chosen, when communicating over fading channels. This is because a row weight between $7 - 10$ will provide a good estimate of the reliability ratio of each message node. During the decoding process, the RRWBF algorithm requires no pre-processing and maintains a similar decoding complexity to the other bit-flipping algorithms.

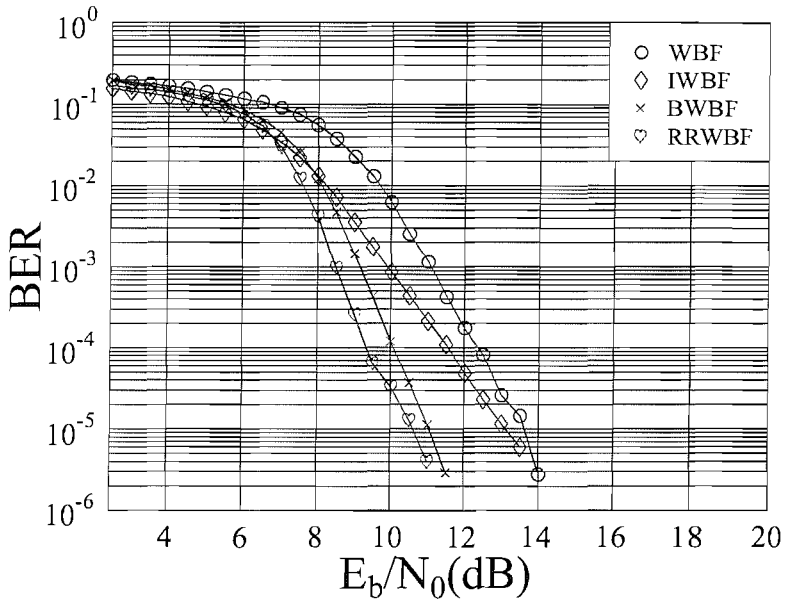


Figure 6.13: BER performance of a half-rate regular LDPC codes having a blocklength of 1000 bits and an average column weight of five, which are decoded by the WBF decoder, the IWBF decoder, the BWBF decoder and the RRWBF decoder, when communicating over uncorrelated Rayleigh fading channel using BPSK modulation. The achievable coding gain of the various schemes at a BER of 10^{-4} will be summarised in Figure 6.15 and Table 6.11.

Modem	BPSK
Channel	AWGN
LDPC code	(900,600,3)
	(900,675,3)
	(900,720,3)
Soft Decoder	Probabilistic Decoder using 15 Iterations
Hard Decoder	WBF, IWBF, BWBF, RRWBF using 90 Iterations

Table 6.9: Simulation parameters for LDPC codes using probabilistic decoder and bit-flipping decoder, when communicating over an AWGN channel using BPSK modulation.

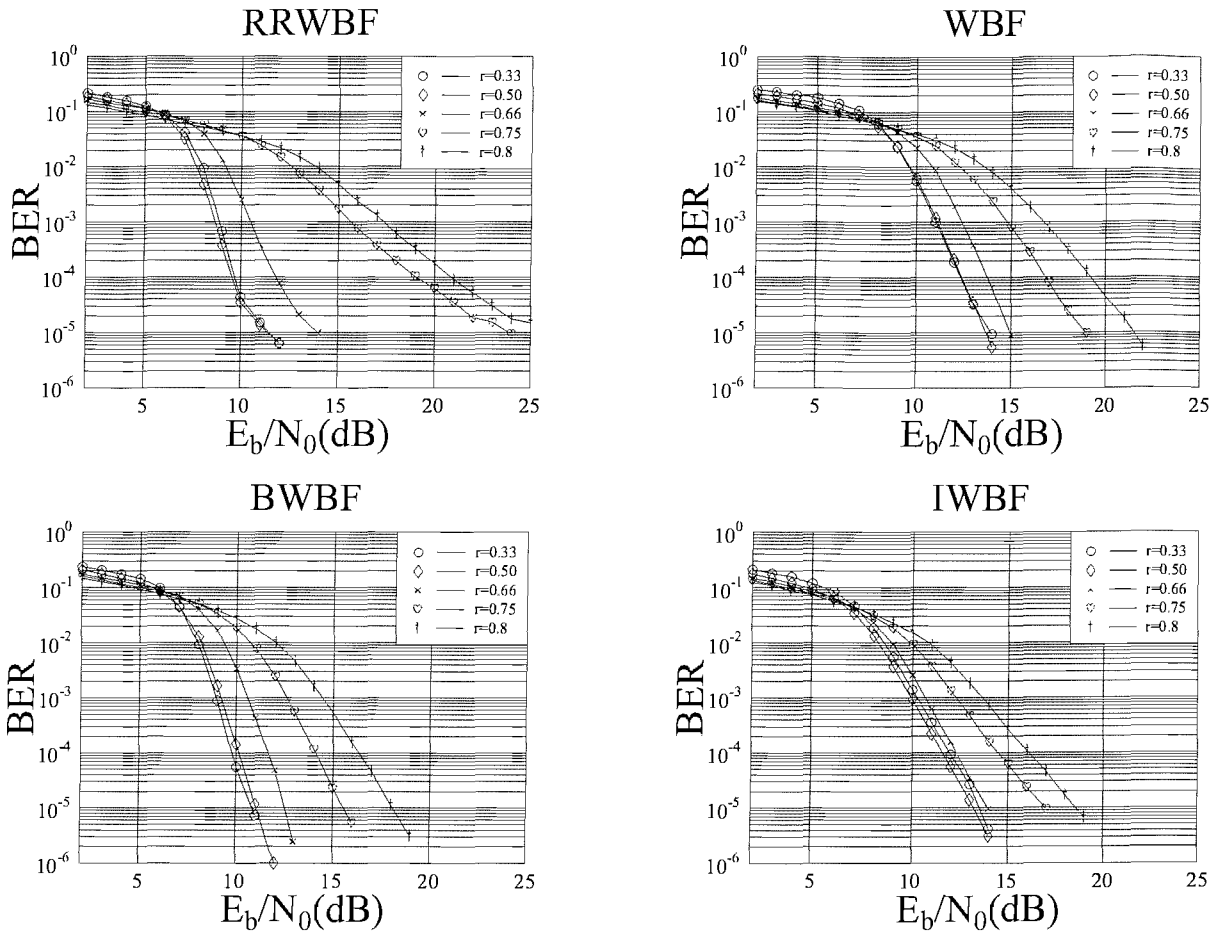


Figure 6.14: BER performance of five regular LDPC codes having various code rates as listed in Table 6.8, decoded by the WBF decoder, the IWBF decoder, the BWBF decoder and the RRWBF decoder, when communicating over an uncorrelated Rayleigh fading channel using BPSK modulation. An average column weight of five is used. The achievable coding gain of the various schemes at a BER of 10^{-4} will be summarised in Figure 6.15 and Table 6.11.

	WBF	IWBF	BWBF	RRWBF	SPA Decoder
+	12150	16200	12150	12150	27000
*	0	4050	0	0	94500

Table 6.10: The number of arithmetic operations required by the various decoders, when decoding the (900,720,3) LDPC code, excluding the pre-processing operations such as finding the optimal weighting factor α_I and α_B for the IWBF and BWBF algorithms, respectively. The arithmetic operations required for the initialisation of the BWBF and the RRWBF during the update of unreliable message bits and the calculation of the reliability ratios are ignored, since these operations are only need to be carried out once.

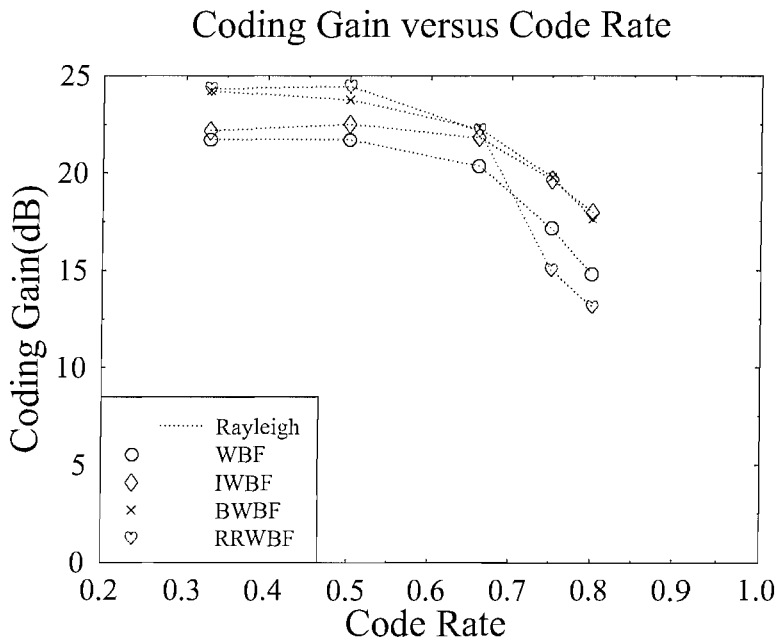


Figure 6.15: Coding gain versus code rate extract from the results shown in Figure 6.14 for an uncorrelated Rayleigh fading channel using the LDPC codes of Table 6.8, with an average column weight of five.

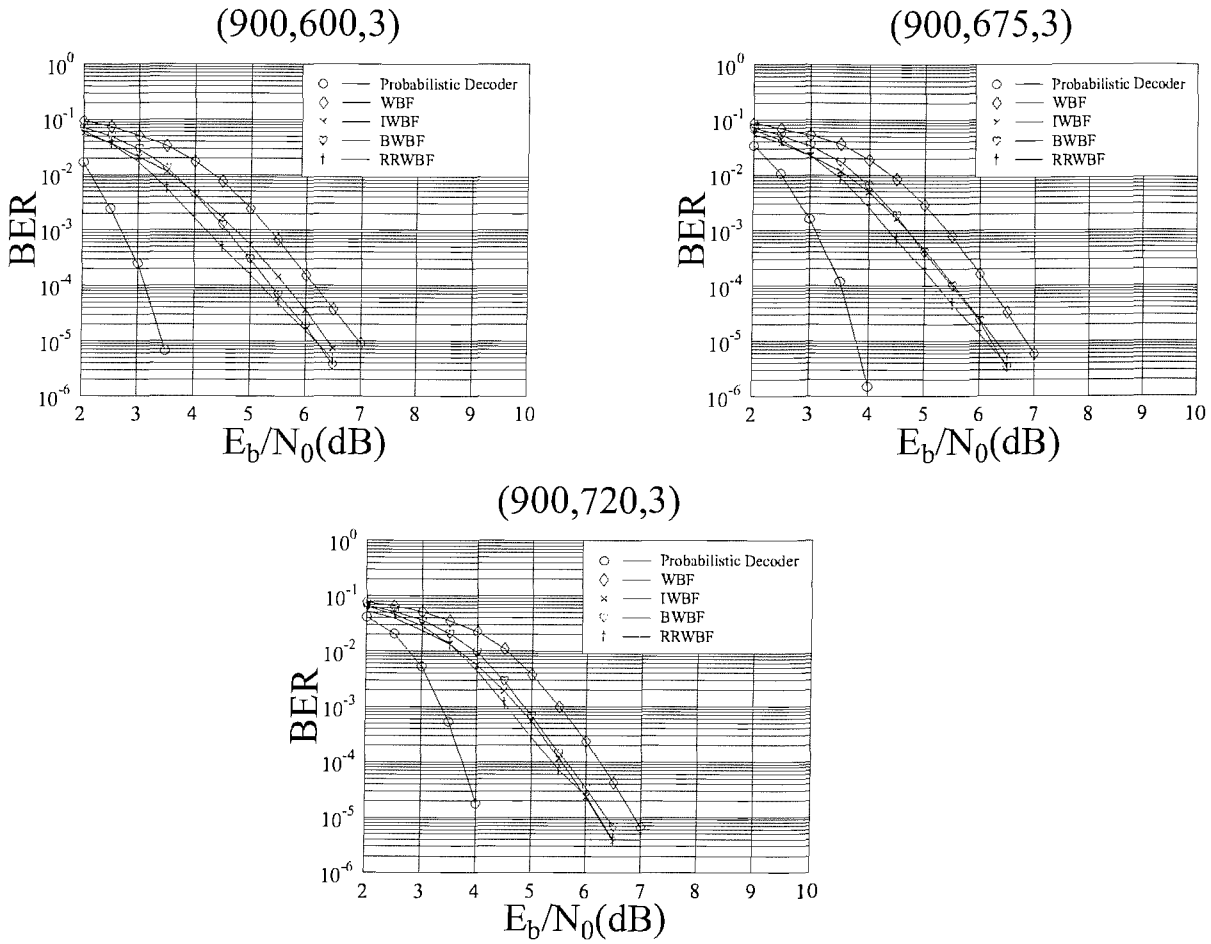


Figure 6.16: BER performance of the three different-rate LDPC codes summarised in Table 6.9 invoking five different decoders, when communicating over an AWGN channel using BPSK modulation.

Parameters			Coding Gain achieved at BER=10^{-4}(dB) AWGN/Uncorrelated Rayleigh Fading			
			WBF	IWBF	BWBF	RRWBF
LDPC column weight equals **three**	Various blocklength at half-rate	LDPC(200,100,3)	1.4/18.6	2/20.85	2.08/22.62	2.13/17.08
		LDPC(400,200,3)	1.65/19.38	2.1/20.92	2.46/23	2.48/18.23
		LDPC(600,300,3)	1.75/19.46	2.3/21	2.62/23.15	2.73/18.23
		LDPC(800,400,3)	1.81/19.7	2.6/21.1	2.75/23.31	2.97/18.23
		LDPC(1000,500,3)	1.83/19.77	2.9/21.15	2.81/23.19	3/18.23
	Various code rates at a codeword length of 1000 bits	$r = 0.33$	0.81/19.7	0.91/20.54	1.86/23.38	2/16.38
		$r = 0.5$	1.81/19.7	2.16/21	2.78/23.38	2.94/18.54
		$r = 0.66$	2.16/18.46	2.78/20.54	2.78/22.15	3.02/18.92
		$r = 0.75$	2.24/17.08	2.86/19.93	2.89/20.85	3.16/19.3
		$r = 0.8$	2.24/16.3	2.86/19.39	2.89/20.08	3.21/17.39
Parameters			Coding Gain achieved at BER=10^{-4}(dB) Uncorrelated Rayleigh Fading			
			WBF	IWBF	BWBF	RRWBF
LDPC column weight equals **five**	Various code rates at a codeword length of 1000 bits	$r = 0.33$	21.73	22.18	24.23	24.34
		$r = 0.5$	21.73	22.53	23.77	24.46
		$r = 0.66$	20.36	21.84	22.3	22.2
		$r = 0.75$	17.18	19.68	19.8	15.02
		$r = 0.8$	14.8	17.98	17.64	13.1
Parameters			Coding Gain achieved at BER=10^{-4}(dB) AWGN			
			WBF	IWBF	BWBF	RRWBF
LDPC column weight equals **three**, Various number of iterations	LDPC(200,100,3)	Iteration = 5	0.98	1.07	1.98	0.98
		Iteration = 10	1.4	1.9	2.07	2.025
		Iteration = 20	1.4	1.98	2.07	2.15
		Iteration = 40	1.4	1.98	2.07	2.15
		Iteration = 60	1.4	1.98	2.07	2.15
		Iteration = 80	1.4	1.98	2.07	2.15
		Iteration = 100	1.4	1.98	2.07	2.15
	LDPC(500,250)	Iteration = 10	1.23	1.23	2.48	1.24
		Iteration = 20	1.73	2.15	2.57	2.4
		Iteration = 40	1.73	2.15	2.57	2.65
		Iteration = 60	1.73	2.15	2.57	2.65
		Iteration = 80	1.73	2.15	2.57	2.65
		Iteration = 100	1.73	2.15	2.57	2.65
	LDPC(1000,500)	Iteration = 10	0.48	0.48	2.15	0.48
		Iteration = 20	1.48	1.57	2.73	1.48
		Iteration = 40	1.82	2.19	2.82	2.7
		Iteration = 60	1.82	2.19	2.82	3.025
		Iteration = 80	1.82	2.19	2.82	3.025
		Iteration = 100	1.82	2.19	2.82	3.025

Table 6.11: Achievable coding gain for the four bit-flipping decoders of Sections 6.1 - 6.4 when the number of LDPC iterations, the codeword blocklength, the code rate and the LDPC code's average column weight is varied. The results were extracted from the BER performance results provided in Section 6.6.

Chapter 7

Summary, Conclusions and Future Research

In this conclusion chapter, a summary of the thesis will be provided and the novelty of our investigations will be highlighted. Additionally, some ideas will be provided for future research.

7.1 Summary

Gallager's original binary regular LDPC codes were introduced and investigated in Chapter 2. The concepts of row and column weight, cycles and the bipartite graph representation of the PCM were highlighted. LDPC codes can be defined by a sparse parity check matrix and they may be decoded by the sub-optimal sum-product algorithm for the sake of achieving near-capacity performance. The encoding and decoding processes of LDPCCs were introduced in Sections 2.4 and 2.5, with the aid of worked examples. Gallager's original work was based on binary LDPCCs. The basic concepts of these codes and a generalized LDPC decoding procedure were introduced in Section 2.7 together with our generalized notation used for describing the LDPC decoding algorithm. These notations were used later in Chapter 3 and Chapter 4. The FFT-based low-complexity decoding approach suggested by Richardson *et al.* [9] was described in Section 2.7.

The performance of various LDPC codes was investigated for transmission over different channels. The family of LDPC codes has attractive distance properties, provided that the all the columns have a weight no less than three. At a coded blocklength of 200 bits, the LDPC code exhibited no undetected errors, when communicating over an AWGN channel. It has been observed that the LDPC code achieves most of the attainable iteration gain within ten to twenty iterations, while communicating over both AWGN and uncorrelated Rayleigh fading channels. The performance of LDPC codes was benchmarked against that of turbo convolutional codes, demonstrating that LDPC codes are capable of achieving a similar BER performance to that of turbo convolutional codes. Furthermore, at a high code rate LDPC codes may outperform turbo codes in the low-BER region.

Coded modulation schemes are commonly used for jointly optimizing the coding and modulation stage of a system. A novel LDPC-aided block coded modulation scheme was proposed in Chapter 2. The extra parity bits introduced by the LDPC code were absorbed by expanding the modulation

constellation without extending the bandwidth. It was observed in Section 2.9.6 that the LDPC-BCM arrangement constituted a more attractive scheme in comparison to the TTCM benchmarker scheme in terms of the attainable BER performance, when communicating over uncorrelated Rayleigh fading channels. As shown in Figure 7.1, a binary LDPC aided joint coding and modulation scheme was found

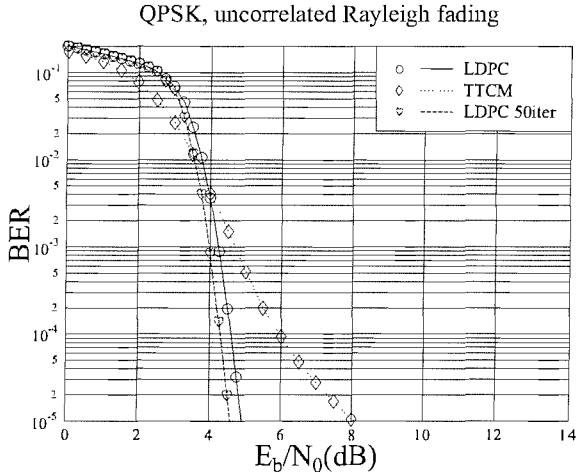


Figure 7.1: BER performance of LDPC and TTCM parameterised in Table 2.26, utilising QPSK when communicating over uncorrelated Rayleigh fading channels.

to achieve an E_b/N_0 gain of about 1.5 dB at a BER of 10^{-5}, when using 15 iterations in comparison to the TTCM benchmarking scheme using 4 iterations. The modulation scheme applied was QPSK and the effective system throughput was 1 BPS.

Luby *et al.* [60] proposed the idea of constructing the LDPC PCM using a non-uniformly distributed density profile, rather than a uniformly distributed as initially suggested by Gallager [1]. The density evolution (DE) algorithm was proposed by Richardson *et al.* [9] for calculating the asymptotic performance of the LDPC code, given a specific density profile. This algorithm was simplified by Chung *et al.* [11] using the Gaussian approximation (GA) of the decoding information and the resultant low-complexity DEGA algorithm was capable of providing an accurate performance prediction as the original high-complexity DE algorithm in [9]. Either the DE or the DEGA algorithm may be used to find the optimal density profile for a long LDPC code. Chung *et al.* demonstrated in [169] that LDPCC achieved a performance within 0.0045dB of the Shannon limit. It was shown in Section 3.8 that the irregular LDPC PCM construction will often introduce weight-two columns, the minimum distance of irregular-construction LDPC codes does not increase linearly with the blocklength. However, when a long block length is considered, Richardson's PCM construction approach outlined in Section 3.8.1 will achieve a good BER performance. By contrast, it was shown in Section 3.8.2 when a moderate block length and high coding rates are considered, Richardson's approach will lead to an error floor. Therefore, Yang [127] suggested to limit the number of weight-two columns in the PCM and we may also aim for avoiding the weight-three columns and weight-four columns. This approach mitigates the error floor incurred by having a high number of weight-two columns in the PCM, which is achieved at the cost of an inferior BER performance in the low SNR region. Based on the comparative

study of the two different PCM construction approaches devised by Richardson [9] and Yang [127], we found that it was more beneficial to use Richardson's approach, provided that the system's delay is not an important design constraint. By contrast, Yang's approach [127] may be implemented, when a moderate block length and/or high coding rate are desired.

The idea of decoding LDPC codes over a non-binary field was devised by Davey and MacKay [54, 55]. The family of non-binary LDPC codes has the advantage of forming cycles with a reduced probability, when compared to their binary counterparts. However, their drawback is that a non-binary symbol has a higher number of legitimate values. Furthermore, the associated decoding complexity is also increased. Davey [54, 56] further developed Richardson's FFT-based decoding algorithm [9] for the non-binary scenario, while ensuring that the decoding complexity does not increase exponentially with respect to the Galois field size. The family of non-binary LDPC codes was characterized in various scenarios and their performance was found to be sometimes better, sometimes worse than that of their binary counterparts. The symbol-based column weight of the non-binary LDPC code has to be carefully chosen for the sake of achieving a better performance. A bit-based LDPC space-time diversity scheme was introduced by Meshkat and Jafarkhani [108], which was characterized in Section 4.5. This scheme iteratively improves the soft channel output arriving at each receiver antenna with the aid of the extra *a priori* information provided by the LDPC code. The drawback of this scheme is that it assumes each individual bit in a phasor constellation is independent, which is not true for a Gray-mapped constellation. Furthermore, the complexity of the iterative evaluation of the soft channel output increases exponentially with respect to the number of transmitter antennas and the number of bits per modulation symbol.

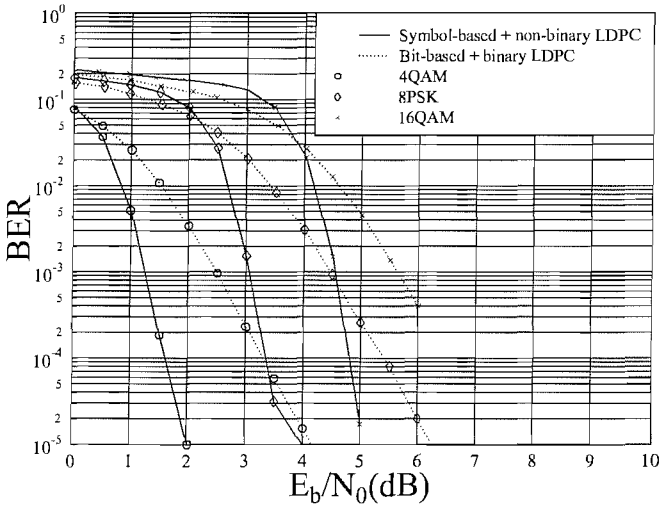


Figure 7.2: BER performance of the bit-based and symbol-based MIMO systems summarized in Table 4.18 utilizing both binary and non-binary LDPC codes, when communicating over an uncorrelated Rayleigh fading channel.

Therefore, a novel purely symbol-based LDPC-ST scheme was developed by invoking Davey's

non-binary LDPC code as a powerful channel code. By using this symbol-based LDPC-ST scheme, the decoding complexity was significantly reduced in comparison to the bit-based scheme proposed by Meshkat and Jafarkhani [108]. Furthermore, an improved BER performance was also observed in Figure 4.24, when communicating over an uncorrelated Rayleigh fading channel using two transmitters and two receivers. As seen in Figure 7.2, the symbol-based MIMO schemes achieved an E_b/N_0 gain of approximately 2 dB at a BER of 10^{-5} in comparison to their bit-based counterparts having the same throughput, when operating over $\mathbf{GF}(4)$, $\mathbf{GF}(8)$, $\mathbf{GF}(16)$ and an uncorrelated Rayleigh fading channel. The LDPC-ST scheme was also characterized in Section 4.7.3 at various throughputs, when benchmarked against a channel-coded $\mathbf{G}_2$ scheme. It has been found that when the required effective throughput is low, it is feasible to concatenate Alamouti's $\mathbf{G}_2$ code with a powerful channel code without having to expand the modulation constellation. However, when a higher effective throughput is desired, the LDPC-ST is superior in comparison to the $\mathbf{G}_2$ coded scheme, since the simple 'repetition-like-code' of $\mathbf{G}_2$ is outperformed by the powerful non-binary LDPC code of the in LDPC-ST scheme.

The idea of constructing a trellis structure for variable length codes was proposed by Buttigieg and Farrell [161, 163]. Based on the trellis structure of Buttigieg and Farrell, Bauer and Hagenauer [154] used the MAP algorithm for decoding both symbol-based and bit-based VLC codes. Based on the work by Bauer and Hagenauer [153, 154], we serially concatenated a variable length source code with various channel codes and exploited the extrinsic information provided by the VLC decoder, which was iteratively exchanged between the source and channel decoding stages. Several different VLCs were introduced in Table 5.1, namely the classic Huffman code and two different types of RVLCs. Using these different encoders led to a different code table and consequently different average VLC symbol lengths were obtained. Having different average VLC symbol lengths implies that the VLC symbols coded by various encoding schemes have different amount of residual redundancy in the symbols and this residual redundancy can be exploited for enhancing the achievable error correction capability. It was observed in Figures 5.11 to 5.13 that the VLC having the highest amount of residual redundancy in comparison to the entropy and the highest free distance, namely the RVLC 2 scheme of Table 5.1, was capable of achieving the highest iteration gain, when concatenated with a channel codec.

Chapter 6 introduced the bit-flipping based decoding of LDPCCs. It was shown that the bit-flipping algorithm is capable of operating at a significantly lower decoding complexity in comparison to the commonly used sum-product algorithm. However, the achievable performance of the bit-flipping algorithm was found to be inferior to that attained by the sum-product algorithm. Hence, an improved weighted bit-flipping algorithm as well as a bootstrap decoding scheme were proposed in Sections 6.2 and 6.3 for improving the attainable BER performance. These two improved schemes were based on the philosophy of applying an optimized weighting factor during decoding, which was obtained by off-line pre-processing. A further developed scheme based on the so-called reliability-ratio was proposed in Section 6.4, which required no *a priori* knowledge or weighting factor. This reliability-ratio based bit-flipping algorithm has been shown to be capable of outperforming the other bit-flipping algorithms considered, as evidenced in Section 6.6.

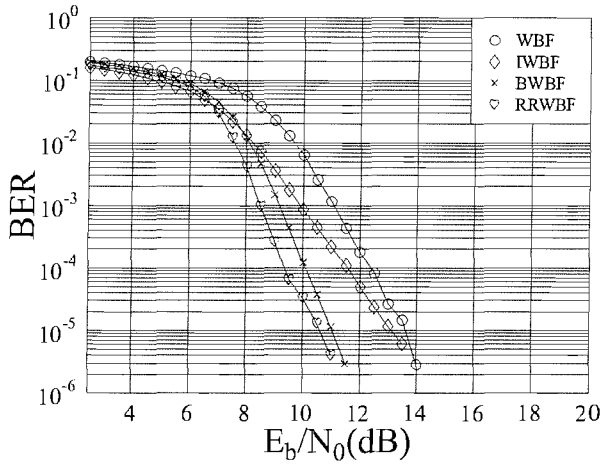


Figure 7.3: BER performance of a half-rate regular LDPC codes having a blocklength of 1000 bits and an average column weight of five, which is decoded by the WBF decoder, the IWBF decoder, the BWBF decoder and the RRWBF decoder of Sections 6.1 to 6.4, respectively, when communicating over an uncorrelated Rayleigh fading channel using BPSK modulation.

7.2 Conclusion

As evidenced by the results of Chapter 2, LDPC codes are attractive channel codes, exhibiting a performance close to that of turbo convolutional codes. Additionally, the family of LDPC codes has numerous advantageous properties over turbo convolutional codes, such as for example exhibiting a reliable error detection capability; straightforward PCM construction; flexible adjustment of the code rate; a lower error floor than that of turbo codes when having PCM columns with a weight no less than three, etc. The novel LDPC-BCM scheme of Chapter 2 constitutes an application example of LDPC codes, where a powerful channel code was integrated with a modulation scheme and exhibited a better performance than the TTCM benchmarker scheme. As outlined in Chapter 3, upon using a non-uniform weight distribution of the PCM, the performance of regular LDPC codes can be further improved. Richardson's [9] and Yang's [127] irregular PCM constructions become beneficial in different applications. Richardson's approach can be used to achieve a good performance, provided the blocklength is sufficiently high. Therefore, Richardson's approach can be used for magnetic recording or broadcast type applications. Yang's approach is more beneficial for employment in delay-sensitive applications, such as interactive video or audio communications. These applications are capable of tolerating a higher error rate, but require a short delay. Both the DE as well as the DEGA algorithms of Sections 3.4 and 3.5 have been frequently invoked for accurately predicting the convergence of LDPC codes. EXIT-charts have also been used for characterizing the convergence of LDPC codes [170]. Sayir *et al.* [170] invoked EXIT-charts for evaluating the decoding performance of the so-called sum-min algorithm. The decoding trajectory of LDPC codes can be conveniently visualized for the sake of characterizing and improving the convergence of the algorithm. By contrast, density evolution simply determines whether the code is powerful or not, without providing any indications, as to how to improve it.

As evidenced in Chapter 4, non-binary LDPC codes [54, 56] devised by Davey and MacKay were demonstrated to have a better BER performance than their binary counterparts, when the column weight is appropriately chosen, but this may not always be achieved. As outlined in Section 4.6, the bit-based LDPC-ST scheme of [108] was further developed to create a novel purely symbol-based scheme by invoking a non-binary LDPC code. This scheme performed better than the bit-based scheme of Meshkat and Jafarkhani [108] in terms of the achievable BER performance, while imposing a reduced complexity. Furthermore, the LDPC-ST scheme of [27] is attractive for employment in high throughput scenarios, i.e. when the required number of bits per symbol was relatively high. Since a high throughput may be achieved in a variety of ways, for example by increasing the number of transmit antennas, the number of modulation levels and the channel coding rate, therefore we aimed to find the optimum configurations at each possible effective throughput. However, increasing the value of either of the above mentioned parameters may result in some disadvantages. When the number of transmitters is increased, the receiver will receive the superposition of the faded symbols sent from each individual transmitter antenna. Hence it is difficult for the receiver to decide the values of the originally transmitted symbols. Furthermore, when the number of modulation levels is increased, the minimum Euclidean distances of the modulated symbols are gradually decreased. Consider BPSK, QPSK and 8PSK for example. The minimum Euclidean distance of these three modulation schemes are reduced from 2 to 1.414 and further down to 1. A reduced minimum Euclidean distance will result in an inferior detection performance. Additionally, a higher channel coding rate will lead to a less powerful channel decoder. As shown in Figure 4.21, when the effective throughput was 2 bps, the LDPC-ST scheme using 2 transmitters and a half-rate LDPC code as well as QPSK modulation achieved the best performance. Comparing the 2-transmitter and 3-transmitter LDPC-ST configurations in Figure 4.21, it was observed that when the number of transmit antennas was fixed, invoking a high rate code in conjunction with a low number of modulation levels will be better than using a low rate code in conjunction with a high number of modulation levels. In the scenario when the channel code rate was fixed, the performance trends concerning the number of transmitters and modulation levels were unclear. For example, as shown in Figure 4.21, when the LDPC(1500,750) code was used, the 2-transmitter scheme performed better than the 4-transmitter scheme. This was because the Euclidean distances for BPSK and QPSK are 2 and 1.414, respectively, while invoking the 4-transmitter scheme will result in each receiver receiving the superposition of four samples, which are received from each of the four individual transmitters. By contrast, as shown in Figure 4.22, the 3 bits per symbol effective throughput 3-transmitter QPSK scheme performed better than the 2-transmitter 8PSK scheme. In this case, the Euclidean distance of 8PSK is 1, which is smaller than the distance of QPSK and the increased inter-antenna interference of the 3-transmitter scheme does not significantly degrade the achievable performance. Therefore, the choice of modulation scheme and the number of transmitters has to be carefully considered, since the modification of each individual simulation parameter in a configuration as listed in Table 4.17 will improve or degrade the error correction capability of the overall system.

As discussed in Chapter 5, the trellis structure devised for VLCs by Buttigieg and Farrell [162,163] was also used by Bauer and Hagenauer [154]. The authors invoked both symbol-based [154] and bit-based VLC [153] decoding using the the MAP decoding algorithm [152]. A VLC can be used to act as a weak error correction code and it has been beneficially concatenated with other channel codes to construct a joint source and channel decoding scheme. It has been observed that maintaining a high

free distance for the VLC is important for the sake of providing reliable extrinsic information after VLC decoding, which will be fed to the channel decoder for the sake of achieving a high iteration gain. For the three VLC codes investigated in Section 5.4, the RVLC 1 arrangement portrayed in Table 5.1 and having a free distance of two performed better than the other two VLCs having a free distance of one. Even though the average symbol length of the RVLC 1 arrangement of Table 5.1 was slightly higher than that of its other two counterparts, the associated extra redundancy provided more extrinsic information during the VLC decoding process. EXIT-charts were invoked for providing a graphically visualized insight in to the convergence of these soft-in soft-out decoders. Hence the EXIT-chart enabled us to understand the behavior of each individual soft-in soft-out constituent decoder and facilitated the construction of attractive concatenated coding schemes.

The family of bit-flipping algorithms was studied in Chapter 6 and a novel RRWBF algorithm was proposed in Section 6.4. The RRWBF algorithm was contributed and was found to have a good performance in comparison to the other existing bit-flipping algorithms, while avoiding the requirement of using any off-line pre-processing. The bit-flipping algorithms are low-complexity decoders, which require no multiplication operations during the iterative decoding process. This implies that the bit-flipping algorithms of Chapter 6 are attractive in terms of having a low complexity, when compared to the sum-product algorithm of Chapter 2. A drawback of the bit-flipping algorithms is that they achieve an inferior performance in comparison to the SPA based decoder. Hence, the bit-flipping algorithms are suitable for low-complexity applications, where achieving the highest possible error correction capability is not at premium. Additionally, there are other LDPC decoders, such as the sum-min decoder proposed by Sayir *et al.* [170]. This approach simplifies the complex SPA algorithm with the aid of carrying out low-complexity decoder post-processing, rather than choosing the tanh operations; as in the SPA, when decoding binary LDPCCs. This approach achieves a performance within 0.1 dB of the SPA algorithm and significantly reduces the complexity imposed, when the algorithm is implemented in hardware.

7.3 Future Research Topics

As demonstrated in Section 6.7, the bit-flipping algorithm has a low complexity in comparison to the sum-product algorithm, when decoding a binary LDPC code. Since a higher decoding complexity will be encountered when decoding non-binary LDPC codes using the sum-product algorithm, a slightly more sophisticated symbol-flipping, rather than bit-flipping algorithm combined with the classic Chase algorithm [171] would be worth more research attention in the context of the non-binary LDPCC of Chapter 4.

The EXIT-chart has been demonstrated to provide a straightforward convergence-test approach, while constructing a two-stage concatenated scheme. However, when designing a more sophisticated scheme, such as a three-stage serial concatenated system combining a VLC decoder, an outer and an inner channel codec, constructing a three-dimensional EXIT-chart would be beneficial. Furthermore, the symbol-based EXIT-chart recently contrived in the group would constitute an interesting research area and provide accurate prediction of the purely symbol-based system's performance, when using the non-binary LDPC-ST scheme of Chapter 4.

The non-binary LDPC code of Chapter 4 has by now been combined with other modulation

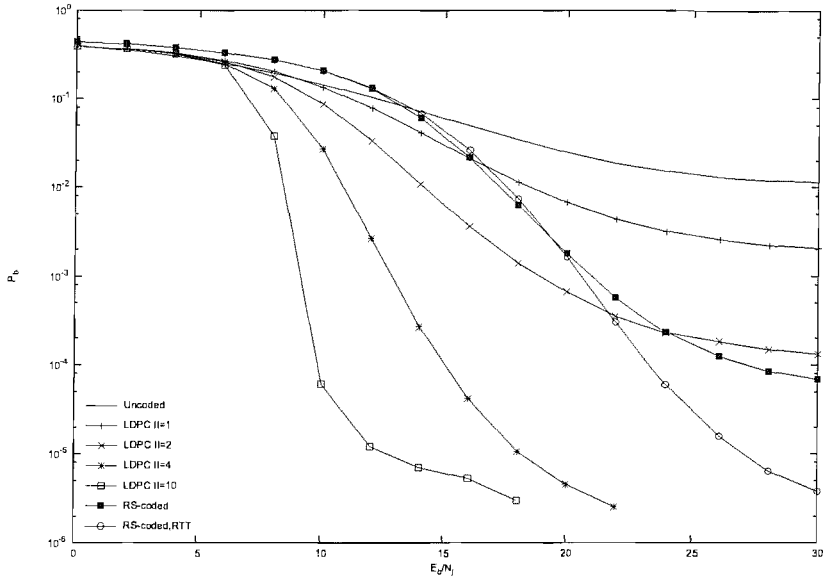


Figure 7.4: BER performance of a non-binary LDPCC assisted slow frequency hopping 16FSK scheme communicating over a Rayleigh fading channel. The E_b/N_0 used was 16 dB and a half-rate LDPC code having a coded block length of 2400 bits was invoked.

schemes such as sphere-packing [172] and MFSK modulation schemes in the group. A coding gain of 2 dB has been achieved at a BER of 10^{-5} by the symbol-based STBC sphere-packing scheme using a non-binary LDPC code against the benchmarking bit-based STBC sphere-packing benchmark scheme combined with a convolutional codec, when communicating over a correlated Rayleigh fading channel having a normalized Doppler frequency of 0.1. Furthermore, as shown in Figure 7.4, the non-binary LDPC code aided Slow Frequency Hopping (SFH) MFSK scheme achieved an E_b/N_0 gain of about 13 dB compared to the classic RS code assisted SFH-MFSK scheme at a BER of 10^{-5}.

It appears promising to jointly optimize these two concatenated components and iterative message passing between the blocks might bring further extra benefits.

As shown in Figure 3.14, message nodes having a higher column weight tend to their correct values more rapidly than those, which have a lower column weight. Therefore in video and audio applications, where unequal protection is desired, irregular LDPC codes may be employed to map the more important data to the high column weight nodes for the sake of ensuring the integrity of the video and audio stream.

Appendix A

Proof of Theorem 2.12

Preparation

In order to prove Theorem 2.11, the following lemma has to be proven first

Lemma 1: Consider a sequence of w_r independent binary digits, where the l^{th} bit is 1 with probability P_l^1. Then the probability that an even number of digits is 1 is given by

$$\frac{1+\prod_{l=1}^{w_r}(1-2P_l^1)}{2} \tag{A.1}$$

Proof : First consider the function $\prod_{l=1}^{w_r}(1-P_l^1+P_l^1 t)$, t is just an arbitrary item in the polynomial which will be set to 1 later for the provement of the theorem. When this expression is expanded into a polynomial format in terms of t, the multiplicative coefficient of $t^i (i = 1...w_r)$ is the probability of i binary 1s within these w_r binary digits. Let us also consider the function $\prod_{l=1}^{w_r}(1-P_l^1-P_l^1 t)$, which is identical except that the coefficients of all the odd powers of t are negative. Adding these two functions, all the coefficients of the even powers of t are doubled, ie, the probability that there is even number of digits will be doubled, and the odd powered terms disappear which means we throw away all the probability that there are odd number of 1s in the sequence of w_r binary digits. So the addition of the two equation has to be halved to obtain the probability of an even number of 1s.

Finally, upon setting $t = 1$ and dividing by 2, the result is the probability of an even number of ones. Also since

$$\frac{\prod_{l=1}^{w_r}(1-P_l^1+P_l^1)+\prod_{l=1}^{w_r}(1-P_l^1-P_l^1)}{2} = \frac{1+\prod_{l=1}^{w_r}(1-2P_l^1)}{2}$$

Thus the Lemma is proved.

Appendix B

Proof of Equation 2.11

According to the Bayes' Rule

$$P(A|B) = P(AB)/P(B) \tag{B.1}$$

So the following equations can be thus derived:

$$\begin{aligned} P[(x_j = 0)|S] &= P[(x_j = 0) \wedge S]/P[S] \\ P[(x_j = 1)|S] &= P[(x_j = 1) \wedge S]/P[S] \\ P[S|(x_j = 0)] &= P[S \wedge (x_j = 0)]/P[x_j = 0] \\ P[S|(x_j = 1)] &= P[S \wedge (x_j = 1)]/P[x_j = 1] \end{aligned}$$

Thus

$$\frac{P[(x_j = 1)|S]}{P[(x_j = 0)|S]} = \left(\frac{P[x_j = 1]}{P[x_j = 0]}\right) \times \left(\frac{P[S|(x_j = 1)]}{P[S|(x_j = 0)]}\right) \tag{B.2}$$

Since $P_j^1 = P[x_j = 1]$ and $(1 - P_j^1) = P[x_j = 0]$, so

$$\frac{P[x_j = 1|\{y\}, S]}{P[x_j = 0|\{y\}, S]} = \frac{P_j^1}{1 - P_j^1}\left(\frac{P(S|x_j = 1, \{y\})}{P(S|x_j = 0, \{y\})}\right) \tag{B.3}$$

Given that $x_j = 0$, a parity check on the bit position j is satisfied, if the other$(w_r - 1)$ positions in the parity check set contain an even number of logical 1s. Since all digits in the ensemble are statistically independent, the probability that all the w_c parity checks are satisfied is the product of the probabilities of the individual checks being satisfied. Using Lemma 1 this is identical to the probability of having an even number of logical 1s, which is given by:

$$P(S|x_j = 1, \{y\}) = \prod_{i=1}^{w_c}\left[\frac{1 - \prod_{l=1}^{w_r - 1}(1 - 2P_{il}^1)}{2}\right] \tag{B.4}$$

$$P(S|x_j = 0, \{y\}) = \prod_{i=1}^{w_c}\left[\frac{1 + \prod_{l=1}^{w_r - 1}(1 - 2P_{il}^1)}{2}\right] \tag{B.5}$$

Upon substituting B.4 and B.5 into B.3, the theorem is thus proven.

Appendix C

Generation of the Companion Matrix

Over the $\mathbf{GF}(2^3)$ decoding field, there are 8 possible non-binary symbols, namely 0...7. We may represent these eight symbols using the notation α^i in Table C.1, where α is a zero of the primitive polynomial $p(x) = x^3 + x + 1$.

Since the decoding field size is $\mathbf{GF}(8)$, each non-binary symbol α^i can be represented by a three-bit binary stream. More explicitly, because α is a zero of the polynomial $p(x)$, thus we have $\alpha^3+\alpha+1 = 0$, which can also be represented as $\alpha^3 = \alpha + 1$. Since the decoding field is $\mathbf{GF}(2^3)$, hence we will use α^2, α^1 and α^0 to represent all the non-binary symbols over $\mathbf{GF}(8)$

$$\begin{aligned}
\alpha^0 &= \alpha^0 = 1 \\
\alpha^1 &= \alpha^1 \\
\alpha^2 &= \alpha^2 \\
\alpha^3 &= \alpha + 1 \\
\alpha^4 &= \alpha \cdot \alpha^3 = \alpha^2 + \alpha \\
\alpha^5 &= \alpha^2 \cdot \alpha^3 = \alpha^2 + \alpha + 1 \\
\alpha^6 &= \alpha^2 \cdot \alpha^4 = \alpha^2 + 1.
\end{aligned} \tag{C.1}$$

Upon mapping the results in the set of Equations C.1, we arrive at Table C.2:

The binary companion matrix T_i for the non-binary symbol α^i is defined as [173]:

$$T_i = \begin{bmatrix} | & | & | \\ \alpha^i & \alpha^{i+1} & \alpha^{i+2} \\ | & | & | \end{bmatrix},$$

where each column is represented in the binary form of α^i as in Table C.2.

Symbol index	0	1	2	3	4	5	6	7
α representation	0	1	α	α^2	α^3	α^4	α^5	α^6

Table C.1: Representation for the eight non-binary symbols over $\mathbf{GF}(8)$.

Symbol index	α representation	Results from Equ. C.1	α^0	α^1	α^2
0	0	0	0	0	0
1	1	1	1	0	0
2	α	α	0	1	0
3	α^2	α^2	0	0	1
4	α^3	$\alpha+1$	1	1	0
5	α^4	$\alpha^2+\alpha$	0	1	1
6	α^5	$\alpha^2+\alpha+1$	1	1	1
7	α^6	α^2+1	1	0	1

Table C.2: Mapping table from non-binary symbols defined over **GF**(8) into the three-digit binary form.

Therefore, we arrive at:

$$1 : T_0 = \begin{bmatrix} | & | & | \\ 1 & \alpha & \alpha^2 \\ | & | & | \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix};$$

$$\alpha : T_1 = \begin{bmatrix} | & | & | \\ \alpha & \alpha^2 & \alpha^3 \\ | & | & | \end{bmatrix} = \begin{bmatrix} 0 & 0 & 1 \\ 1 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix};$$

$$\alpha^2 : T_2 = \begin{bmatrix} | & | & | \\ \alpha^2 & \alpha^3 & \alpha^4 \\ | & | & | \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 1 & 1 \\ 1 & 0 & 1 \end{bmatrix};$$

$$\alpha^3 : T_3 = \begin{bmatrix} | & | & | \\ \alpha^3 & \alpha^4 & \alpha^5 \\ | & | & | \end{bmatrix} = \begin{bmatrix} 1 & 0 & 1 \\ 1 & 1 & 1 \\ 0 & 1 & 1 \end{bmatrix};$$

$$\alpha^4 : T_4 = \begin{bmatrix} | & | & | \\ \alpha^4 & \alpha^5 & \alpha^6 \\ | & | & | \end{bmatrix} = \begin{bmatrix} 0 & 1 & 1 \\ 1 & 1 & 0 \\ 1 & 1 & 1 \end{bmatrix};$$

$$\alpha^5 : T_5 = \begin{bmatrix} | & | & | \\ \alpha^5 & \alpha^6 & \alpha^7 \\ | & | & | \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 0 & 0 \\ 1 & 1 & 0 \end{bmatrix};$$

$$\alpha^6 : T_6 = \begin{bmatrix} | & | & | \\ \alpha^6 & \alpha^7 & \alpha^8 \\ | & | & | \end{bmatrix} = \begin{bmatrix} 1 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{bmatrix}.$$

Thus the binary companion matrices for the eight non-binary symbols are obtained. Upon replacing the non-binary symbols defined over **GF**(8) by their corresponding binary companion matrices obtained here, we arrive at the binary equivalent PCM of the LDPC PCM constructed over **GF**(8), as shown in Figure 4.2.

List of Symbols

General notation

- The superscript * is used to indicate complex conjugation. Therefore, a^* represents the complex conjugate of the variable a.
- The superscript ^{T} is used to indicate matrix transpose operation. Therefore, $\mathbf{a}^T$ represents the transpose of the matrix $\mathbf{a}$.
- The superscript $^{-1}$ is used to indicate inverse matrix operation. Therefore, $\mathbf{a}^{-1}$ represents the inverse matrix of the matrix $\mathbf{a}$.
- The notation $\bigotimes$ denotes the convolution operation. Therefore, $a \bigotimes b$ represents the convolution of the variables a and b.
- The notation $\hat{x}$ represents the estimate of x.
- The notation $\mathcal{F}(f)$ is the Fourier Transform of $f(t)$.

Special symbols

$\mathbf{GF}(q)$: Galois Field of size q.

$\mathbf{H}$: Binary parity check matrix of an LDPC code.

$\mathbf{H}_b$: Same as $\mathbf{H}$.

$\mathbf{H}_q$: Non-binary parity check matrix of an LDPC code defined over $\mathbf{GF}(q)$.

$\mathbf{G}$: Binary generator matrix of the LDPC code.

$\mathbf{G}_b$: Same as $\mathbf{G}$.

$\mathbf{G}_q$: Non-binary generator matrix of an LDPC code defined over $\mathbf{GF}(\mathrm{q})$.

$\mathbf{I}_{K\times K}$: A $(K \times K)$-dimensional identity matrix.

N: Number of LDPC coded bits, i.e. the number of columns in an LDPC code's PCM.

M: The number of rows in an LDPC code's PCM.

K: Number of LDPC information bits, i.e. $N - M$.

N_q: Number of non-binary LDPC coded symbols, i.e. the number of columns in a non-binary LDPC code's PCM.

M_q: The number of rows in a non-binary LDPC code's PCM.

K_q: Number of non-binary LDPC coded information symbols in a codeword, where we have $N_q - M_q$.

$\mathbf{C}$: Codeword vector.

$\mathbf{S}$: Source information vector.

$\mathbf{P}$: Parity-bit vector.

(N, K): A binary LDPC code having K number of input information bits and N coded bits.

(N, K, w_c): A binary LDPC code having K number of input information bits and N coded bits, where the average column weight is w_c.

$(N_q, K_q)_q$: A non-binary LDPC code defined over $\mathbf{GF}(q)$ having K number of input information symbols and N_q coded non-binary symbols.

w_c: Average column weight of the LDPC's PCM, i.e. the average number of non-zero entries in a column (binary or non-binary).

w_r: Average row weight of the LDPC's PCM, i.e. the average number of non-zero entries in a row (binary or non-binary).

P_j^a: The probability that the j^{th} LDPC coded symbol is in state a.

$P_r(x_j)$: The *intrinsic* probability ratio of the j^{th} coded bit.

$PR(x_j)$: The *a posteriori* probability ratio of the j^{th} LDPC coded bit.

$R^a_{i,j}$: The probability that the j^{th} LDPC coded symbol is in state a, based on the probability distributions of *other* coded symbols participating in the i^{th} parity check.

$Q^a_{i,j}$: The probability that the j^{th} LDPC coded symbol is in state a, based on the probability distributions of the coded symbols provided by all *other* parity checks, excluding the i^{th} parity check.

$P^a_{i,j}$: Same as $Q^a_{i,j}$.

$LR_{i,j}$: Same as $R^0_{i,j}/R^1_{i,j}$.

$PR_{i,j}$: Same as $Q^0_{i,j}/Q^1_{i,j}$.

$Q_{i,j}$: $log(Q^0_{i,j}/Q^1_{i,j})$.

$R_{i,j}$: $log(R^0_{i,j}/R^1_{i,j})$.

R_i: Row indices of all the non-zero entries participating in the i^{th} column of the PCM.

C_i: Column indices of all the non-zero entries paritipating in the i^{th} row of the PCM.

n_c: Number of coded bits of a convolutional code.

k_c: Number of input bits of a convolutional code.

K_c: Constraint length of a convolutional code.

λ_e: The column density distribution of an irregular LDPC code's PCM. (Edge perspective)

ρ_e: The row density distribution of an irregular LDPC code's PCM. (Edge perspective)

λ_n: The column density distribution of an irregular LDPC code's PCM. (Node perspective)

ρ_n: The row density distribution of an irregular LDPC code's PCM. (Node perspective)

$Q^{(l)}$: A set of $Q_{i,j}, i = 1 \ldots M, j = 1 \ldots N$ values after the l^{th} iteration.

$R^{(l)}$: A set of $R_{i,j}, i = 1 \ldots M, j = 1 \ldots N$ values after the l^{th} iteration.

$\bar{N}$: Average VLC symbol length.

$\mathbf{u}$: Source symbol stream.

$\mathbf{b}$: VLC encoded bit stream.

$\mathsf{H(U)}$: The binary entropy of the set of VLC codes (U).

l_{max}: Maximum VLC symbol length.

l_{min}: Minimum VLC symbol length.

O: Number of source symbols in a block.

W: Number of VLC encoded bits in a block.

$c(i)$: Encoded VLC symbol for the i^{th} source symbol.

v^k: Trellis state of the VLC at the k^{th} symbol instant.

T: Number of possible source symbols.

R^s: Source code rate.

W_k: The number of legitimate VLC trellis states at the k^{th} symbol instant.

E_i: Error term used in the WBF algorithm for the i^{th} coded bit.

s_i: Syndrome of the i^{th} parity check.

α_I: Optimum weighting factor used in the IWBF algorithm.

$\hat{\alpha}_B$: Normalised BWBF threshold weighting factor.

$RR_{i,j}$: Reliability ratio value of the j^{th} coded bit participating in the i^{th} parity check.

D: *A posteriori* LLR value used in the EXIT-chart investigations.

A: *A priori* LLR value used in the EXIT-chart investigations.

Z: Channel LLR value.

E: Extrinsic LLR value.

d_f: Free distance.

Bibliography

[1] R. Gallager, "Low density parity check codes," *IEEE Transaction on Information Theory*, vol. 8, pp. 21–28, Jan. 1962.

[2] R. Gallager, "Low density parity check codes," *Ph.D thesis,M.I.T,USA*, 1963.

[3] C. Berrou, A. Glavieux and P. Thitimajshima, "Near shannon limit error-correcting coding and decoding : turbo codes," in *Proceedings of the IEEE International Confrence on Communications*, pp. 1064–1070, 1993.

[4] V. V. Zyablov and M. S. Pinsker, "Estimation of the error correction complexity for gallager's low-density codes.," *Problemy Peredachi Informatsii*, vol. 11, no. 1, pp. 23–36, 1975.

[5] M. R. Tanner, "A recursive approach to low complexity codes," *IEEE Transactions on Information Theory*, vol. 27, September 1981.

[6] M. Sipser and D. A. Spielman, "Expander codes," *IEEE Transactions on Information Theory*, vol. 42, pp. 1710–1722, November 1996.

[7] D. J. C MacKay, and R. M. Neal, "Near shannon limit performance of low density parity check codes," *Electronics Letters*, vol. 33, pp. 457–458, 13 March 1997.

[8] D. J. C. MacKay and R. M. Neal, "Good error-correction codes based on very sparse matrices," *IEEE Transactions on Information Theory*, vol. 45, pp. 399–431, March 1999.

[9] T. ichardson, R. Urbanke, "The capacity of low density parity check codes under message-passing decoding," *IEEE Transaction on Information Theory*, vol. 47, pp. 599–618, Feb. 2001.

[10] S. Y. Chung, "On the construction of some capacity-approaching coding schemes," *Ph.D thesis, MIT,USA*, 2000.

[11] S. Y. Chung, T. J. Richardson, R. L. Urbanke, "Analysis of sum-product decoding of low density parity check codes using a gaussian approximation," *IEEE Transactions on Information Theory*, vol. 47, Feb. 2001.

[12] J. Chen and M. Fossorier, "Density evolution for two improved BP-based decoding algorithms of LDPC codes," *IEEE Communication Letters*, vol. 6, pp. 208–210, May 2002.

[13] W. Lin, X. Juan and G Chen, "Density evolution method and threshold decision for irregular LDPC codes," in *IEEE International Conference on Communications, Circuits and Systems*, vol. 1, pp. 25 – 28, 27-29 June 2004.

[14] K. R. Narayanan, I. Altunbas and R. S. Narayanaswami, "Design of serial concatenated MSK schemes based on density evolution," *IEEE Transactions on Communications*, vol. 51, pp. 1283 – 1295, August 2003.

[15] A. Anastasopoulos, "A comparison between the sum-product and the min-sum iterative detection algorithms based on density evolution," in *IEEE Global Telecommunications Conference*, vol. 2, pp. 1021 – 1025, 25 - 29 Nov. 2001.

[16] H. Song, J. Liu and B. V. Kumar, "Convergence analysis of iterative soft decoding in partial response channels," *IEEE Transactions on Magnetics*, vol. 39, pp. 2552–2554, Sept. 2003.

[17] D. Burshtein, M. Krivelevich, M. Litsyn and G. Miller, "Upper bounds on the rate of LDPC codes," *IEEE Transactions on Information Theory*, vol. 48, pp. 2437 – 2449, September 2002.

[18] D. Burshtein, "Bounds on the performance of belief propagation decoding," *IEEE Transactions on Information Theory*, vol. 48, pp. 112 – 122, January 2002.

[19] G. Miller and D. Burshtein, "Bounds on the maximum-likelihood decoding error probability of low-density parity-check codes," *IEEE Transactions on Information Theory*, vol. 47, pp. 2696 – 2710, November 2001.

[20] H. Futaki and T. Ohtsuki, "Low-density parity-check (LDPC) coded OFDM systems with M-PSK," in *IEEE 55th Vehicular Technology Conference*, vol. 2, (Birmingham, AL, USA), pp. 1035 – 1039, 6-9 May. 2002.

[21] H. Futaki and T. Ohtsuki, "Performance of low-density parity-check (LDPC) coded OFDM systems," in *IEEE International Conference on Communications*, vol. 3, pp. 1696 – 1700, 28 April - 2 May. 2002.

[22] H. Futaki and T. Ohtsuki, "Low-density parity-check (LDPC) coded OFDM systems," in *IEEE 54th Vehicular Technology Conference*, vol. 1, (Atlanta City, New Jersey, USA), pp. 82 – 86, 7 - 11 October. 2001.

[23] B. Lu, G. Yue and X. Wang, "Performance analysis and design optimization of LDPC-coded MIMO OFDM systems," *IEEE Transactions on Signal Processing*, vol. 52, pp. 348 – 361, Feb. 2004.

[24] M. Y. Alias, F. Guo, S. X. Ng, T. H. Liew and L. Hanzo, "LDPC and turbo coding assisted space-time block coded OFDM," in *IEEE Vehicular Technology Conference*, vol. 4, (Jeju, Korea), pp. 2309–2313, 22-25 April 2003.

[25] J. Y. Chung, M. Y. Alias, F. Guo and L. Hanzo, "LDPC and turbo coding assisted space-time block coded OFDM for H.26L," in *Proceedings of PIMRC' 2003*, (Beijing, China), pp. 2702–2706, September 2003.

[26] H. Futaki and T. Ohtsuki, "LDPC-based space-time transmit diversity schemes with multiple transmit antennas," in *IEEE 57th Vehicular Technology Conference*, vol. 4, pp. 2589 – 2593, 22 - 25 April 2003.

[27] F. Guo, L. Hanzo, "Low complexity non-binary LDPC and modulation schemes communicating over MIMO channels," in *Accepted by VTC Fall, 2004*, (Los Angeles, CA, USA), 2004.

[28] H. Pishro-Nik and F. Fekri, "On decoding of low-density parity-check codes over the binary erasure channel," *IEEE Transactions on Information Theory*, vol. 50, pp. 439 – 454, March 2004.

[29] D. Burshtein and G. Miller, "An efficient maximum-likelihood decoding of LDPC codes over the binary erasure channel," *IEEE Transactions on Information Theory*, vol. 50, pp. 2837–2844, Nov. 2004.

[30] H. Song, J. Liu and B. V. Kumar, "Low complexity LDPC codes for partial response channel," in *IEEE Global Telecommunications Conference*, vol. 2, pp. 1294 – 1299, 17-21 Nov 2002.

[31] T. Mittelholzer, A. Dholakia and E. Eleftheriou, "Reduced-complexity decoding of low density parity check codes for generalized partial response channels," *IEEE Transactions on Magnetics*, vol. 37, pp. 721–728, March 2001.

[32] M. Yang and W. E. Ryan, "Performance of (quasi-)cyclic LDPC codes in noise bursts on the EPR4 channel," in *IEEE Global Telecommunication Conference*, vol. 5, pp. 2961 – 2965, 25-29 Nov. 2001.

[33] M. Yang and W. E. Ryan, "Performance of efficiently encodable low-density parity-check codes in noise bursts on the EPR4 channel," *IEEE Transactions on Magnetics*, vol. 40, pp. 507–512, March 2004.

[34] D. Yang, R. Molstad and Y. Yip, "Performance evaluation of LDPC code on VR2 channel," in *IEEE International Magnetics Conference*, p. AP2, 28 April - 2 May 2002.

[35] E. Eleftheriou, S. Olcer, "Further results on the performance of LDPC coded modulation for AWGN channels," *ITU-Telecommunication Standardization Sector*, May 2001.

[36] J. Hou, P. H. Siegel, L. B. Milstein and H. D. Pfister, "Capacity-approaching bandwidth-efficient coded modulation schemes based on low-density parity-check codes," *IEEE Transactions on Information Theory*, vol. 49, pp. 2141–2155, September 2003.

[37] F. Guo, L. Hanzo, "LDPC assisted block coded modulation for transmission over Rayleigh fading channels," in *Vehicular Technology Conference, Spring*, vol. 3, (Jeju, Korea), pp. 1867 – 1871, 22-25 April 2003.

[38] H. Song and J. R. Cruz, "Reduced-Complexity Decoding of Q-ary LDPC Codes for Magnetic Recording," *IEEE Transactions on Magnetics*, vol. 39, pp. 1081–1087, March 2003.

[39] H. Song, J. Liu and B. V. Kumar, "Concatenated low density parity check (LDPC) codes for magnetic recording channels," in *IEEE International Magnetics Conference*, pp. DT–12, 28 March - 3 April 2003.

[40] H. Song, R. M. Todd and J. R. Cruz, "Applications of low-density parity-check codes to magnetic recording channels," *IEEE Journal on Selected Areas in Communications*, vol. 19, pp. 918–923, May 2001.

[41] H. Song, R. M. Todd and J. R. Cruz, "Low density parity check codes for magnetic recording channels," *IEEE Transactions on Magnetics*, vol. 36, pp. 2183–2186, Sept. 2000.

[42] S. Sahu, H. Song, B. V. Kumar, "Performance of low density parity check (LDPC) codes on high-density magnetic tape recording signals," in *IEEE International Magnetics Conference*, pp. DT–10, 28 March - 3 April 2003.

[43] Y. Zhan, D. Yuan, H. Zhang, "The application of LDPC codes in image transmission," in *International Conference on Communications Circuits and Systems*, vol. 1, pp. 29 – 33, 27-29 June 2004.

[44] X. Yang, D. Yuan, P. Ma and H. Zhang, "Performance of LDPC codes in image transmission over Rayleigh fading channel," in *International Conference on Communication Technology*, vol. 2, pp. 1444–1446, 9-11 April 2003.

[45] S. Chae and Y. Park, "Low complexity encoding of regular low density parity check codes," in *IEEE Vehicular Technology Conference*, vol. 3, pp. 1822 – 1826, 6-9 Oct. 2003.

[46] H. Zhong and T. Zhang, "Design of VLSI implementation-oriented LDPC codes," in *IEEE Vehicular Technology Conference*, vol. 1, pp. 670–673, 6-9 Oct. 2003.

[47] H. Zhong and T. Zhang, "Joint code-encoder-decoder design for LDPC coding system VLSI implementation," in *International Symposium on Circuits and Systems*, vol. 2, pp. 389–392, 23-26 May 2004.

[48] T. Zhang and K. K. Parhi, "VLSI implementation-oriented (3,k)-regular low-density parity-check codes," in *IEEE Workshop on Signal Processing Systems*, pp. 25–36, 26-28 Sept. 2001.

[49] G. Lechner, J. Sayir and M. Rupp, "Efficient DSP implementation of an LDPC decoder," in *International Conference on Acoustics, Speech and Signal Processing*, vol. 4, pp. 665–668, 17-21 May 2004.

[50] D. E. Hocevar, "LDPC code construction with flexible hardware implementation," in *IEEE International Conference on Communications*, vol. 4, pp. 2708–2712, 11-15 May 2003.

[51] J. Thorpe, "Design of LDPC graphs for hardware implementation," in *IEEE International Symposium on Information Theory*, p. 483, 2002.

[52] Y. Pei, L. Yin, J. Lu, "Design of irregular LDPC codec on a single chip FPGA," in *IEEE 6th Circuits and Systems Symposium on Emerging Technologies: Frontiers of Mobile and Wireless Communications*, vol. 1, pp. 221 – 224, 31 May - 2 June 2004.

[53] M. M. Mansour, M. Shanbhag, "Architecture-aware low-density parity-check codes," in *International Symposium on Circuits and Systems*, vol. 2, pp. 57–60, 25-28 May 2003.

[54] M. C. Davey and D. J. C MacKay, "Low density parity check codes over GF(q)," *IEEE Communications Letters*, vol. 2, pp. 165–167, June 1998.

[55] M. C. Davey and D. J. C. MacKay, "Low density parity check codes over GF(q)," in *Proceedings of the 1998 IEEE Information Theory Workshop*, pp. 70–71, June 1998.

[56] M.C. Davey, "Error-correction using low density parity check codes," *Ph.D thesis, University of Cambridge, UK*, 1999.

[57] L. Barnault and D. Declercq, "Fast decoding algorithm for LDPC over GF(2^q)," in *IEEE Information Theory Workshop*, pp. 70–73, 31 March - 4 April 2003.

[58] K. Nakamura, Y. Kabashima and D. Saad, "Statistical mechanics of low-density parity check error-correcting codes over Galois fields," *Europhysics Letters*, vol. 56, pp. 610–616, November 2001.

[59] X. Li, M. R. Soleymani, J. Lodge and P. S. Guinand, "Good LDPC codes over GF(q) for bandwidth efficient transmission," in *IEEE workshop on Signal Processing Advances in Wireless Communications*, pp. 95–99, 15-18 June 2003.

[60] M. G. Luby, M. Mitzenmacher, M. A. Shokrollahi, and D. A. Spielman, "Improved low density parity check codes using irregular graphs and belief propagations," in *Proceedings of the IEEE International Symposium on Information Theory*, p. 117, 1998.

[61] T. Richardson, M. A Shokrollahi, R. Urbanke, "Design of Capacity Approaching Irregular Low Density Parity Check Codes," *IEEE Transaction on Information Theory*, vol. 47, Febuary 2001.

[62] D. J. C MacKay, S. T Wilson, and M. C Davey, "Comparison of constructions of irregular gallager codes," *IEEE Transactions on Communications*, vol. 47, pp. 1449–1454, October 1999.

[63] D. J. C MacKay, "Punctured and irregular high-rate gallager codes," *Unpublished.*

[64] X. Yang, D. Yuan, P. Ma and M. Jiang, "New research on unequal error protection (UEP) property of irregular LDPC codes," in *IEEE Consumer Communications and Networking Conference*, pp. 361 – 363, 5-8 Jan. 2004.

[65] S. J. Johnson and S. R. Weller, "A family of irregular LDPC codes with low encoding complexity," *IEEE Communications Letters*, vol. 7, pp. 79–81, Feb. 2003.

[66] T. Tao, C. Jones, J. D. Villasenor and R. D. Wesel, "Construction of irregular LDPC codes with low error floors," in *IEEE International Conference on Communications*, vol. 5, pp. 3125–3129, 11-15 May 2003.

[67] C. E. Shannon, "A mathematical theory of communication," *The Bell system Technical Journal*, vol. 27, pp. 379–656, July 1948.

[68] S. Sankaranarayanan, B. Vasic and E. M. Kurtas, "Irregular low-density parity-check codes: construction and performance on perpendicular magnetic recording channels," *IEEE Transactions on Magnetics*, vol. 39, pp. 2567–2569, Sept. 2003.

[69] M. Ardakani, F. R. Kschischang, "Designing irregular LDPC codes using EXIT charts based on message error rate," in *IEEE International Symposium on Information Theory*, p. 454, 2002.

[70] J. Lu, J. M. F. Moura, "Turbo design for LDPC with large girth," in *IEEE workshop on Signal Processing Advanced in Wireless Communications*, pp. 90–94, 15-18 June 2003.

[71] H. Zhang and J. M. F. Moura, "The design of structured regular LDPC codes with large girth," in *IEEE Global Telecommunications Conference*, vol. 7, pp. 4022–4027, 1-5 December 2003.

[72] H. Zhang and J. M. F. Moura, "Large-girth LDPC codes based on graphical models," in *IEEE workshop on Signal Processing Advances in Wireless Communications*, pp. 100–104, 15-18 June 2003.

[73] J. M. F. Moura, J. Lu and H. Zhang, "Structured low-density parity-check codes," *IEEE Signal Processing Magazine*, vol. 21, pp. 42–55, Jan. 2004.

[74] L. Lan, Y. Y. Tai, L. Chen, S. Lin and K. Abdel-Ghaffar, "A trellis-based method for removing cycles from bipartite graphs and construction of low-density parity-check codes," *IEEE Communications Letters*, vol. 8, pp. 443–445, July 2004.

[75] J. A. McGowan, R. C. Williamson, "Loop removal from LDPC codes," in *IEEE Information Theory Workshop*, pp. 230–233, 31 March - 4 April 2003.

[76] M. Lentmaier, "Soft iterative decoding of generalized low-density parity-check codes based on MAP decoding of component hamming codes," *Diploma Thesis, University of Ulm, Germany*, 1997.

[77] T. Zhang and K. K. Parhi, "A class of efficient-encoding generalized low-density parity-check codes," in *IEEE Internationa Conference on Acoustics, Speech, and Signal Processing*, vol. 4, pp. 2477 – 2480, 7-11 May. 2001.

[78] T. Zhang and K. K. Parhi, "High-performance, low-complexity decoding of generalized low-density parity-check codes," in *IEEE Global Telecommunications Conference*, vol. 1, pp. 181 – 185, 25-29 November. 2001.

[79] S. Hirst and B. Honary, "Decoding of generalized low-density parity-check codes using weighted bit-flip voting," in *IEE Proceedings on Communications*, vol. 149, pp. 1 – 5, 2002.

[80] S. Hirst and B. Honary, "Application of efficient Chase algorithm in decoding of generalized low-density parity-check codes," in *IEEE Communications Letters*, vol. 6, pp. 385 – 387, 2002.

[81] J. Boutros, O. Pothier and G. Zemor, "Generalized low density (Tanner) codes," in *IEEE International Conference on Communications*, vol. 1, pp. 441 – 445, 6-10 June, 1999.

[82] Y. Kou, S. Lin and M. Fossorier, "Low density parity check codes: construction based on finite geometries," in *IEEE Global Telecommunications Conference*, vol. 2, pp. 825–829, 27 Nov. - 1 Dec. 2000.

[83] Y. Kou, S. Lin and M. Fossorier, "Low-density parity-check codes based on finite geometries: a rediscovery and new results," *IEEE Transactions on Information Theory*, vol. 47, pp. 2711–2736, November 2001.

[84] J. Xu, H. Tang, Y. Kou, S. Lin and K. Abdel-Ghaffar, "A general class of LDPC finite geometry codes and their performance," in *IEEE International Symposium on Information Theory*, p. 309, 2002.

[85] H. Tang, J. Xu, Y. Kou, S. Lin and K. Abdel-Ghaffar, "On algebraic construction of Gallager and circulant low-density parity-check codes," *IEEE Transactions on Information Theory*, vol. 50, pp. 1269–1279, June 2004.

[86] Y. Kou, J. Xu, H. Tang, S. Lin and K. Abdel-Ghaffar, "On circulant low densty parity check codes," in *IEEE International Symposium on Information Theory*, p. 200, 2002.

[87] M. P. C. Fossorier, "Quasicyclic low-density parity-check codes from circulant permutation matrices," *IEEE Transactions on Information Theory*, vol. 50, pp. 1788–1793, Aug. 2004.

[88] Z. Liu and D. A. Pados, "Low complexity decoding of finite geometry LDPC codes," in *IEEE International Conference on Communications*, vol. 4, pp. 2713–2717, 11-15 May 2003.

[89] I. B. Djordjevic and B. Vasic, "Projective geometry LDPC codes for ultralong-haul WDM high-speed transmission," *IEEE Photonics Technology Letters*, vol. 15, pp. 784–786, May 2003.

[90] O. Milenkovic, I. B. Djordjevic, B. Vasic, "Block-circulant low-density parity-check codes for optical communication systems," *IEEE Journal of Selected Topics in Quantum Electronics*, vol. 10, pp. 294 – 299, March-April 2004.

[91] B. Ammar, B. Honary, Y. Kou and S. Lin, "Construction of low density parity check codes: a combinatoric design approach," in *IEEE International Symposium on Information Theory*, p. 311, 2002.

[92] 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 Transactions on Information Theory*, vol. 50, pp. 1257–1269, June 2004.

[93] J. Rosenthal and P. O. Vontobel, "Constructions of regular and irregular LDPC codes using Ramanujan graphs and ideas from Margulis," in *International Symposium on Information Theory*, p. 4, 24-29 June 2001.

[94] K. S. Kim, S. H. Lee, Y. H. Kim and J. Y. Ahn, "Design of binary LDPC code using cyclic shift matrices," *Electronics Letters*, vol. 40, pp. 325–326, March 2004.

[95] T. Okamura, "Designing LDPC codes using cyclic shifts," in *IEEE International Symposium on Information Theory*, p. 151, 29 June - 4 July 2003.

[96] T. J. Richardson and R. L. Urbanke, "Efficient Encoding of Low-Density Parity-Check Codes," *IEEE Transactions on Information Theory*, vol. 47, pp. 638 – 656, Febuary 2001.

[97] D. A. Spielman, "Linear-time encodable and decodable error-correcting codes," *IEEE Transactions on Information Theory*, vol. 42, pp. 1723–1731, November 1996.

[98] O. Pothier, L. Brunel and J. Boutros, "A low complexity FEC scheme based on the intersection of interleaved block codes," in *IEEE 49th Vehicular Technology Conference*, vol. 1, (Houston, TX, USA), pp. 274 – 278, 16-20 May 1999.

[99] H. Sankar and K. R. Narayanan, "Memory-efficient sum-product decoding of LDPC codes," *IEEE Transactions on Communications*, vol. 52, pp. 1225–1230, Aug. 2004.

[100] J. Chen, A. Deholakia, E. Eleftheriou, M. Fossorier, X. Y. Hu, "Near optimal reduced-complexity decoding algorithms for LDPC codes," in *IEEE International Symposium on Information Theory*, p. 455, 2002.

[101] J. Zhang, M. P. C. Fossorier, "A Modified Weighted Bit-Flipping Decoding of Low-Density Parity-Check Codes," *IEEE Communications Letters*, vol. 8, pp. 165–167, March 2004.

[102] A. Nouh, A. H. Banihashemi, "Bootstrap decoding of low-density parity-check codes," *IEEE Communications Letters*, vol. 6, pp. 391–393, September 2002.

[103] F. Guo, L. Hanzo, "Reliability ratio based weighted bit-flipping decoding for low-density parity-check codes.," *IEE Electronics Letters*, vol. 40, pp. 1356 – 1357, 14 Oct. 2004.

[104] F. Guo, L. Hanzo, "Reliability ratio based weighted bit-flipping decoding for LDPC codes," in *Accepted by VTC 2005 spring*, (Stockholm, Sweden), 2005.

[105] M. Lentmaier and K. Sh. Zigangirov, "On generalized low-density parity-check codes based on Hamming component codes," *IEEE Communication Letters*, vol. 3, pp. 248 – 250, August 1999.

[106] S. Lin, H. Tang and Y. Kou, "On a class of finite geometry low density parity check codes," in *IEEE International Symposium on Information Theory*, p. 2, 24-29 June 2001.

[107] S. ten Brink, "Convergence behavior of iteratively decoded parallel concatenated codes," *IEEE Transactions on Communications*, vol. 49, pp. 1727–1737, October 2001.

[108] P. Meshkat and H. Jafarkhani, "Space-time low-density parity-check codes," vol. 2, (Pacific Grove, Monterey, CA, USA), pp. 1117 –1121, 3-6, Nov 2002.

[109] L. Hanzo, P. Cherriman, J,Streit, *Wireless Video Communications: Second to Third Generation Systems and Beyond.* Wiley & IEEE, 2001.

[110] J. Pearl, "Probabilistic Reasoning in Intelligent Systems," *San Mateo, CA:Morgan Kaufmann*, 1988.

[111] R. M. Neal, "http://www.cs.toronto.edu/ radford/homepage.html,"

[112] L. Hanzo, T. H. Liew, B. L. Yeap, S. X. Ng, *Turbo Coding, Turbo Equalisation and Space-Time Coding for transmission over fading channels*, pp. 317–390. Wiley & IEEE, 2002.

[113] A. V. Oppenheim and A. S. Willsky, *Signal and Systems*, ch. 7, pp. 177–284. Prentice Hall, 1999.

[114] J. G. Proakis and D. G. Manolakis, *Digital signal processing, principles, algorithms and applications, 3rd Edition*, ch. 5, p. 403. Prentice Hall, 1996.

[115] L. Hanzo, T. H. Liew, B. L. Yeap, S. X. Ng, *Turbo Coding, Turbo Equalisation and Space-Time Coding for transmission over fading channels*, ch. 9, pp. 317–390. Wiley & IEEE, 2002.

[116] P. Robertson and T. Worz, "Bandwidth-efficient turbo trellis-coded modulation using punctured component codes," *IEEE Journal on Selected Areas in Communications*, vol. 16, pp. 206–218, February 1998.

[117] O. Acikel and W. Ryan, "Punctured turbo-codes for BPSK/QPSK channels," *IEEE Transactions on Communications*, vol. 47, pp. 1315–1323, September 1999.

[118] G. Ungerboeck, "Channel coding with multilevel/phase signal," *IEEE Transactions on Information Theory*, vol. 28, pp. 55–66, January 1982.

[119] P. Robertson, T. Worz, "Bandwidth-efficient turbo trellis-coded modulation using punctured component codes," *IEEE Journal on Selected Areas in Communications*, vol. 16, pp. 206–218, February 1998.

[120] J. Hou, P. H. Siegel and L. B. Milstein, "Performance analysis and code optimization of low density parity-check codes on Rayleigh fading channels," *IEEE Journal on Selected Areas in Communications*, vol. 19, pp. 924–934, May 2001.

[121] K. Kasai, T. Shibuya and K. Sakaniwa, "Detailedly represented irregular low-density parity-check codes," *IEICE Transactions on Fundamentals*, no. 10, pp. 2435–2444, 2003.

[122] M. G. Luby, M. Mitzenmacher, M. A. Shokrollahi, D. A. Spielman and V. Stemann, "Practical loss-resilient codes," in *Proceedings of 29th Annual ACM Symp. Theory of Computing*, pp. 150–159, 1997.

[123] T. Etzion, A. Trachtenberg and A. Vardy, "Which codes have cycle-free tanner graphs?," *IEEE Transactions on Information Theory*, vol. 45, pp. 2173 – 2181, Sept. 1999.

[124] Joachim Hagenauer, Elke Offer and Lutz Papke, "Iterative decoding of binary block and convolutional codes," *IEEE Transactions on Information Theory*, vol. 42, pp. 429–445, March 1996.

[125] C. R. P. Hartmann and L. D. Rudolph, "An optimum symbol-by-symbol decoding rule for linear codes," *IEEE Transactions on Information Theory*, vol. 22, pp. 514–517, September 1976.

[126] S. Y. Chung, "http://lids.mit.edu/ sychung/gaopt.html,"

[127] M. Yang, W. E. Ryan and Y. Li, "Design of efficiently encodable moderate-length high-rate irregular LDPC codes," *IEEE Transaction on Communication*, vol. 52, pp. 564–571, 2004.

[128] S. Prakash, S. Glenn, and M. Khaled, "Propagation of belief functions: A distributed approach," in *Proceedings of the 2nd Annual Conference on Uncertainty in Artificial Intelligence (UAI-86)*, (New York, NY), pp. 0–0, Elsevier Science Publishing Comapny, Inc., 1986.

[129] H. Y. Hau and R. L. Kashyap, "Belief combination and propagation in a lattice-structured interference network," *IEEE Transaction on Systems, Man and Cybernetics*, vol. 20, pp. 45–57, Jan.-Feb. 1990.

[130] C. Berrou and A. Glavieux, "Near optimum error correcting coding and decoding: turbo-codes," *IEEE Transactions on Communications*, vol. 44, pp. 1261–1271, October 1996.

[131] R. J. McEliece, D. J. C. MacKay, J. F. Cheng, "Turbo decoding as an instance of Pearl's belief propagation algorithm," *IEEE Journal on Selected Areas in Communications*, vol. 16, pp. 140–152, Feb. 1998.

[132] L. Hanzo, M. Muenster, B.J. Choi and T. Keller, *OFDM and MC-CDMA for Broadcasting Multi-User Communications, WLANs and Broadcasting.* Wiley & IEEE, 2003.

[133] L. Hanzo, L-L. Yang, E-L. Kuan and K. Yen , *Single- and Multi-Carrier DS-CDMA: Multi-User Detection, Space-Time Spreading, Synchronisation, Networking and Standards.* Wiley & IEEE, 2003.

[134] K.H.H. Wong and L. Hanzo, *Mobile Radio Communications*, ch. 4.4.4, pp. 425–428. IEEE Press, 445 Hoes Lane, Piscataway, NJ, 08855, USA: IEEE Press and Pentech Press, 1992.

[135] S. M. Alamouti, "A simple transmit diversity technique for wireless communications," *IEEE Journal on Selected Areas in Communications*, vol. 16, pp. 1451–1458, October 1998.

[136] L. Hanzo, F.C.A. Somerville, J.P. Woodard, *Voice Compression and Communications - Principles and Applications for Fixed and Wireless Channels.* Wiley & IEEE, 2001.

[137] S. G. Wilson, *Digital modulation and coding.* Prentice Hall, 1999.

[138] T. B. M. Kawahara, Y. Chiu, "High-speed software implementation of Huffman coding," in *Proceedings of Data Compression Conference, 1998*, p. 553, 30 March - 1 April 1998.

[139] T. M. B. Wei, "A programmable parallel Huffman decoder," in *Proceedings of IEEE International Conference Image Processing*, vol. 3, pp. 668–671, 13-16 Nov 1994.

[140] A. N. M. Aggarwal, "Efficient huffman decoding," in *Proceedings of IEEE International Conference Image Processing, 2000*, vol. 3, pp. 936–939, 10-13 Sept. 2000.

[141] R. Hashemian, "Direct huffman coding and decoding using the table of code-lengths," in *Proceedings of International Conference on Information Technology: Coding and Computing [Computers and Communications]*, pp. 237–241, 28-30 April 2003.

[142] B. W. Y. Wei, T. H. Meng, "A parallel decoder of programmable Huffman codes," *IEEE transactions on Circuits and Systems for Video Technology*, vol. 5, pp. 175–178, April 1995.

[143] R. Hashemian, "Condensed huffman coding, a new efficient decoding technique," in *Proceedings of The 2002 45th Midwest Symposium on Circuits and Systems*, vol. 1, pp. 228–231, 4-7 Aug. 2002.

[144] Y. Takishima, M. Wada, H. Murakami, "Reversible variable length codes," *IEEE Transaction on Communications*, vol. 43, February/March/April 1995.

[145] C. W. Lin, Y. J. Chuang and J. L. Wu, "Generic construction algorithms for symmetric and asymmetric RVLCs," in *Proceedings of The 8th International Conference on Communication Systems*, vol. 2, pp. 968–972, 25-28 Nov. 2002.

[146] K. Lakovic and J. Villasenor, "Design considerations for efficient reversible variable length codes," in *Proceedings of Thirty-Sixth Asilomar Conference on Signals, Systems and Computers*, vol. 1, pp. 262–266, 3-6 Nov. 2002.

[147] K. Lakovic and J. Villasenor, "On design of error-correcting reversible variable length codes," *IEEE Communication Letters*, vol. 6, pp. 337–339, Aug 2002.

[148] C. W. Tsai and J. L. Wu, "On constructing the Huffman-code-based reversible variable-length codes," *IEEE Transactions on Communications*, vol. 49, pp. 1506–1509, Sept 2001.

[149] V. Buttigieg and P. G. Farrell, "On variable-length error-correcting codes," in *IEEE International Symposium on Information Theory*, p. 507, 27 June-1 July 1994.

[150] C. Lamy and J. Paccaut, "Optimised constructions for variable-length error correcting codes," in *Proceedings. 2003 IEEE Information Theory Workshop, 2003*, (Paris, France), pp. 183–186, 31 March - 4 April 2003.

[151] C. Lamy and F. X. Bergot, "Lower bounds on the existence of binary error-correcting variable-length codes," in *Proceedings of IEEE Information Theory Workshop, 2003*, pp. 300–303, 31 March - 4 April 2003.

[152] L. R. Bahl, J. Cocke, F. Jelinek and J. Raviv, "Optimal Decoding of Linear Codes for Minimizing Symbol Error Rate," *IEEE Transactions on Information Theory*, vol. 20, pp. 284–287, March 1974.

[153] R. Bauer, J. Hagenauer, "On variable length codes for iterative source/channel decoding," *Data Compression Conference*, pp. 273–282, 27-29, March 2001.

[154] R. Bauer and J. Hagenauer, "Symbol by symbol MAP decoding of variable length codes," in *3rd ITG Conference Source and Channel Coding*, vol. 1, (Munich, Germany), pp. 111 – 116, 17-19 Jan. 2000.

[155] M. Park and D. J. Miller, "Decoding entropy-coded symbols over noisy channels using discrete HMMs," in *Proceedings of IEEE International Conference on Information Sciences and Systems*, March 1998.

[156] J. T. Wen and J. Villasenor, "Soft-input soft-output decoding of variable length codes," *IEEE Transactions on Communications*, vol. 50, pp. 689–692, May 2002.

[157] E. Fabre, A. Guyader and C. Guillemot, "Joint source-channel turbo decoding of VLC-coded Markov sources," in *Proceedings of IEEE International Conference on Acoustics, Speech, and Signal Processing*, vol. 4, pp. 2657–2660, 7-11 May 2001.

[158] G. Melnikov, G. M. Schuster and A. K. Katsaggelos, "Shape coding using temporal correlation and joint VLC optimization," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 10, pp. 744–754, Aug 2000.

[159] K. Lakovic and J. Villasenor, "On reversible variable length codes with turbo codes, and iterative source-channel decoding," in *IEEE International Symposium on Information Theory*, p. 170, 2002.

[160] S. X. Ng, F. Guo, L. Hanzo, "Joint source-coding, channel-coding and modulation schemes for AWGN and Rayleigh fading channels," *IEE Electronic Letters*, vol. 39, pp. 1259–1261, 21 Aug 2003.

[161] V. Buttigieg and P. G. Farrell, "A maximum likelihood decoding algorithm for variable-length error-correcting codes.," in *Proc. 5th Bangor Symposium on Communications*, (Bangor, Wales), pp. 56–59, 2-3 June 1993.

[162] V. Buttigieg and P. G. Farrell, "A maximum a-posteriori (MAP) decoding algorithm for variable-length error-correcting codes," in *Codes and cyphers: Cryptography and coding IV, The Institute of Mathematics and its Applications.*, (Essex, England), pp. 103–119, 1995.

[163] V. Buttigieg and P. G. Farrell, "Variable-length error-correcting codes," *IEE Proceedings in Communications*, vol. 147, pp. 211–215, Aug. 2000.

[164] V. Buttigieg, "Variable-length error-correction codes," *Ph.D thesis, Department of Electrical Engineering, University of Manchester, UK*, 1995.

[165] V. I. Levenshtein, "Binary codes capable of correcting deletions, insertions and reversals.," *Sov. Phys. Dokl.*, vol. 6, pp. 707–710, 1966.

[166] K. Lakovic, J. Villasenor, "Combining variable length codes and turbo codes," *Vehicular Technology Conference Spring*, vol. 4, pp. 1719–1723, 6-9 May 2002.

[167] J. Kliewer, R. Thobaben, "Combining FEC and optimal soft-input source decoding for the reliable transmission of correlated variable-length encoded signals," *Data Compression Conference*, pp. 83–91, 2-4 April 2002.

[168] Y. Kou, S. Lin and M. Fossorier, "Low density parity check codes based on finite geometries: A rediscovery," in *IEEE International Symposium on Information Theory*, (Sorrento, Italy), 25-30 June 2000.

[169] S. Y. Chung, G. D. Jr. 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 Communications Letters*, vol. 5, pp. 58–60, Feb. 2001.

[170] G. Lechner and J. Sayir, "Improved sum-min decoding of LDPC codes.," 10-13 Oct. 2004.

[171] David Chase, "A Class of Algorithms for Decoding Block Codes With Channel Measurement Information," *IEEE Transactions of Information Theory*, vol. 18, pp. 170–182, January 1972.

[172] O. Alamri, F. Guo, M. Jiang and L. Hanzo, "Turbo detection of symbol-based non-binary LDPC-coded space-time signals using sphere packing modulation.," in *Submitted to Vehicular Technology Conference, 2005 Fall*, (Dallas, USA), 2005.

[173] T. R. N. Rao and E. Fujiwara, *Error-control coding for computer systems*, ch. 3, p. 110. Prentice Hallf, 1989.

Index

W

Author Index