

# **NON-BINARY COMPOUND CODES BASED ON SINGLE PARITY-CHECK CODES**

**Farzad Ghayoor**

In fulfillment of the Doctor of Philosophy in the College of Agriculture, Engineering  
and Science, University of KwaZulu-Natal

March 2013

As the candidate's Supervisor I agree/~~do not agree~~ to the submission of this thesis.

Signed: \_\_\_\_\_  \_\_\_\_\_

Name: \_\_\_\_\_ F Takawira \_\_\_\_\_


Date: \_\_\_\_\_ 28 March 2013 \_\_\_\_\_

**COLLEGE OF AGRICULTURE, ENGINEERING AND SCIENCE****DECLARATION 1 - PLAGIARISM**

I, Farzad Ghayoor, declare that

1. The research reported in this thesis, except where otherwise indicated, is my original research.
2. This thesis has not been submitted for any degree or examination at any other university.
3. This thesis does not contain other persons' data, pictures, graphs or other information, unless specifically acknowledged as being sourced from other persons.
4. This thesis does not contain other persons' writing, unless specifically acknowledged as being sourced from other researchers. Where other written sources have been quoted, then:
  - a. Their words have been re-written but the general information attributed to them has been referenced
  - b. Where their exact words have been used, then their writing has been placed in italics and inside quotation marks, and referenced.
5. This thesis does not contain text, graphics or tables copied and pasted from the Internet, unless specifically acknowledged, and the source being detailed in the thesis and in the References sections.

Signed




**COLLEGE OF AGRICULTURE, ENGINEERING AND SCIENCE****DECLARATION 2 - PUBLICATIONS**

DETAILS OF CONTRIBUTION TO PUBLICATIONS that form part and/or include research presented in this thesis (include publications in preparation, submitted, *in press* and published and give details of the contributions of each author to the experimental work and writing of each publication)

1. Ghayour F., Takawira F., “High-Rate Non-Binary Generalized Low-Density Parity-Check Codes”, in *Proc. Southern African Telecommunications Networks and Applications Conference (SATNAC)*, Wild Coast Sun, South Africa, Sep. 2008.
2. Ghayour F., Takawira F., “Structured Low-Density Parity-Check Codes for Next Generation Wireless Communications”, in *Proceeding of the 1<sup>st</sup> IEEE African Winter School on Information Theory and Communications*. Kruger National Park, South Africa, Jun 2010.
3. Ghayour F. Takawira F. and Xu H. “High-Rate Non-Binary Product Codes”, in *Proc. Southern African Telecommunications Networks and Applications Conference (SATNAC)*, East London, South Africa, Sep. 2011.
4. Ghayour F. Takawira F. and Xu H. “A Memory Efficient MAP Algorithm for Linear Block Codes”, in *Proc. IEEE AFRICON*, Livingstone, Zambia, Sep. 2011.
5. Ghayour F. Takawira F. and Xu H. “On MAP Decoding of High-Rate Non-Binary Linear Block Codes”, in *Proc. Southern African Telecommunications Networks and Applications Conference (SATNAC)*, George, South Africa, Sep. 2012.

6. Ghayoor F. Takawira F. and Xu H. “On Decoding of Non-Binary Single Parity-check Codes”, submitted to *IEEE Commun. Lett.*

Signed: 

## **ACKNOWLEDGMENT**

I wish to sincerely thank my supervisor, Prof. Fambirai Takawira, for his invaluable help, guidance and support. His willingness to always set aside his time to assist me in any way has been deeply appreciated. I would also like to express my full appreciation to my co-supervisor, Prof. Hongjun Xu, for his valuable input into my research.

I would like to thank my family for their continued support and encouragement. They have always encouraged me in all challenges that I have faced in my life and my PhD was no exception.

Finally, I would like to thank the center of excellence (COE) at the University of KwaZulu-Natal for their much-appreciated financial support and for providing the equipment necessary for the completion of my PhD.

## ABSTRACT

Shannon showed that the codes with random-like codeword weight distribution are capable of approaching the channel capacity. However, the random-like property can be achieved only in codes with long-length codewords. On the other hand, the decoding complexity for a random-like codeword increases exponentially with its length. Therefore, code designers are combining shorter and simpler codes in a pseudorandom manner to form longer and more powerful codewords. In this research, a method for designing non-binary compound codes with moderate to high coding rate is proposed. Based on this method, non-binary single parity-check (SPC) codes are considered as component codes and different iterative decoding algorithms for decoding the constructed compound codes are proposed. The soft-input soft-output component decoders, which are employed for the iterative decoding algorithms, are constructed from optimal and sub-optimal *a posteriori* probability (APP) decoders. However, for non-binary codes, implementing an optimal APP decoder requires a large amount of memory. In order to reduce the memory requirement of the APP decoding algorithm, in the first part of this research, a modified form of the APP decoding algorithm is presented. The amount of memory requirement of this proposed algorithm is significantly less than that of the standard APP decoder. Therefore, the proposed algorithm becomes more practical for decoding non-binary block codes.

The compound codes that are proposed in this research are constructed from combination of non-binary SPC codes. Therefore, as part of this research, the construction and decoding of the non-binary SPC codes, when SPC codes are defined over a finite ring of order  $q$ , are presented. The concept of finite rings is more general and it thus includes non-binary SPC codes defined over finite fields. Thereafter, based on production of non-binary SPC codes, a class of non-binary compound codes is proposed that is efficient for controlling both random-error and burst-error patterns and can be used for applications where high coding rate schemes are required. Simulation results show that the performance of the proposed codes is good. Furthermore, the performance of the compound code improves over larger rings. The analytical performance bounds and the minimum distance properties of these product codes are studied.

## TABLE OF CONTENTS

DECLARATION .....	iii
ACKNOWLEDGMENT .....	vi
ABSTRACT .....	vii
TABLE OF CONTENTS.....	viii
LIST OF FIGURES .....	xi
LIST OF TABLES.....	xiii
LIST OF ACRONYMS .....	xiv
1. INTRODUCTION .....	1
1.1. Introduction.....	1
1.2. Turbo codes.....	2
1.3. Turbo-like codes .....	4
1.3.1. Low-density parity-check codes.....	5
1.3.2. Turbo product codes.....	8
1.4. Motivation for research.....	9
1.5. Thesis overview .....	10
1.6. Original contributions.....	11
1.7. Publications.....	12
2. A POSTERIORI PROBABILITY ALGORITHMS .....	14
2.1. Introduction.....	14
2.2. Background and notation .....	16
2.2.1. General notation .....	16
2.2.2. Trellis for linear block codes.....	17
2.3. Computation of the MAP decoding algorithm.....	19
2.4. Modified APP decoding algorithms.....	21



2.4.1. APP decoding algorithm based on straightforward implementation..	22
2.4.2. APP decoding algorithm based on the code's dual space .....	26
2.4.3. APP decoding algorithm based on Fourier transform .....	30
2.4.4. Complexity comparison .....	32
2.4.5. Simulation results .....	34
2.5. Conclusion .....	37
3. NON-BINARY SINGLE PARITY-CHECK CODES.....	39
3.1. Introduction.....	39
3.2. Encoding of non-binary SPC codes .....	41
3.3. Decoding of non-binary SPC codes.....	42
3.3.1. APP decoding algorithm based on straightforward implementation..	43
3.3.2. APP decoding algorithm based on Fourier transform .....	44
3.3.3. Complexity comparison .....	48
3.4. Simulation results .....	49
3.5. Conclusion .....	57
4. SINGLE PARITY-CHECK TURBO PRODUCT CODES.....	58
4.1. Introduction.....	58
4.2. The structure of product codes.....	61
4.3. Turbo decoding of non-binary SPC code.....	65
4.3.1. Algorithm I.....	68
4.3.2. Modified-Algorithm I.....	69
4.3.3. Algorithm II.....	71
4.4. Simulation results .....	73
4.5. Conclusion .....	86
5. PERFORMANCE ANALYSIS .....	88
5.1. Introduction.....	88

5.2. Performance bound .....	89
5.3. Minimum distance property .....	93
5.4. Conclusion .....	97
6. CONCLUSION .....	98
REFERENCES .....	101

## LIST OF FIGURES

Figure 2.1 Trellis diagram of the (7,4) Hamming code given by the parity-check matrix in (2.1).....	18
Figure 2.2 Simulated BER performance of the (7,4) Hamming code over AWGN channel using BPSK .....	36
Figure 3.1 Addition and multiplication of $\mathbb{F}_2$ versus $\mathbb{Z}_2$ and $\mathbb{F}_4$ versus $\mathbb{Z}_4$ .....	42
Figure 3.2 Performance of the (5,4) SPC code defined over $\mathbb{F}_q$ .....	52
Figure 3.3 Performance of the (5,4) SPC code defined over $\mathbb{Z}_q$ .....	53
Figure 3.4 Performance of the (5,4) SPC code defined over $\mathbb{F}_8$ and $\mathbb{Z}_8$ .....	54
Figure 3.5 Performance of the (5,4) SPC code defined over $\mathbb{Z}_8$ that is decoded with different decoding algorithms.....	55
Figure 3.6 Performance of the (5,4) SPC code defined over $\mathbb{F}_8$ and $\mathbb{Z}_8$ that are decoded with the APP decoding algorithm based on Fourier transform.....	56
Figure 4.1 A general structure for a two-dimensional product code.....	63
Figure 4.2 Two-dimensional SPC product code.....	64
Figure 4.3 Turbo decoder for the 2D-SPC-TP code.....	67
Figure 4.4 Effect of iteration on the performance of the $(8,4,3)_8$ 2D-SPC-TP code that is defined over $\mathbb{Z}_8$ and is decoded by Algorithm I.....	76
Figure 4.5 Effect of iteration on the performance of the $(8,4,3)_8$ 2D-SPC-TP code that is defined over $\mathbb{Z}_8$ and is decoded by algorithm MA-I.....	77
Figure 4.6 Effect of iteration on the performance of the $(8,4,3)_8$ 2D-SPC-TP code that is defined over $\mathbb{Z}_8$ and is decoded by Algorithm II .....	78

Figure 4.7 The performance of the $(63,49,3)_4$ 2D-SPC-TP code that is defined over $\mathbb{F}_4$ and is decoded by different iterative decoding algorithms after two iterations.....	79
Figure 4.8 The performance of the $(63,49,3)_4$ 2D-SPC-TP code that is defined over $\mathbb{Z}_4$ and is decoded by different iterative decoding algorithms after two iterations.....	80
Figure 4.9 The performance of the $(63,49,3)_4$ 2D-SPC-TP code that is defined over $\mathbb{Z}_4$ and is decoded by algorithm MA-I with different number of iterations .....	81
Figure 4.10 The performance of the $(63,49,3)_q$ 2D-SPC-TP code that is defined over $\mathbb{F}_q$ and is decoded by algorithm MA-I after two iterations .....	82
Figure 4.11 The performance of the $(63,49,3)_q$ 2D-SPC-TP code that is defined over $\mathbb{Z}_q$ and is decoded by algorithm MA-I after two iterations.....	83
Figure 4.12 Comparison between the performance of the $(63,49,3)_8$ 2D-SPC-TP code defined over $\mathbb{Z}_8$ and $\mathbb{F}_8$ . The codes are decoded by algorithm MA-I after two iterations.....	84
Figure 4.13 Comparison between the performance of different rate 2D-SPC-TP codes defined over $\mathbb{Z}_4$ and decoded by algorithm MA-I after two iterations.....	85
Figure 5.1 Comparison between the bit weight spectral for a minimum-weight codeword of an SPC code defined over $\mathbb{F}_q$ and $\mathbb{Z}_q$ . $q = (4,8,16,32,64,128)$ .....	95
Figure 5.2 Comparison between the bit weight spectral for a minimum-weight codeword of a 2D-SPC code defined over $\mathbb{F}_q$ and $\mathbb{Z}_q$ . $q = (4,8,16,32,64,128)$ .....	96

## LIST OF TABLES

Table 2.1 Complexity comparison between minimal-storage APP decoding algorithm and the BCJR algorithm for a code defined over $\mathbb{F}_q$ .....	33
Table 2.2 Complexity comparison between the minimal-storage APP decoding algorithms for codes defined over $\mathbb{Z}_q$ with different code rates .....	33
Table 2.3 Complexity of the APP decoding algorithm implemented over the code's dual space by using pre-calculated DFT vectors .....	34
Table 2.4 Complexity comparison between minimal-storage APP decoding algorithm and the BCJR algorithm for decoding the (7,4) Hamming code.....	37
Table 3.1 Complexity comparison between the two proposed algorithms.....	49
Table 3.2 Complexity comparison between the two proposed algorithms for decoding the (5,4) SPC code defined over $\mathbb{Z}_8$ .....	57
Table 4.1 Code parameters for different rate 2D-SPC-TP codes.....	85

## LIST OF ACRONYMS

2D-SPC	two-dimensional single parity-check
2D-SPC-TP	two-dimensional single parity-check turbo product
3GPP LTE	3 <sup>rd</sup> generation partnership project long term evolution
APP	<i>a posteriori</i> probability
AWGN	additive white Gaussian noise
BCH	Bose–Chaudhuri–Hochquenghem
BCJR	Bahl-Cocke-Jelinek-Raviv
BER	bit error rate
BI-AWGN	binary input additive white Gaussian noise
BP	belief propagation
BPSK	binary phase-shift keying
CRC	cyclic redundancy check
DFT	discrete Fourier transform
DVB-S2	digital video broadcasting second generation
EMS	extended min-sum
EXIT	extrinsic information transfer
FEC	forward error correction
FFT	fast Fourier transform
FFT-BP	fast Fourier transform belief propagation
GILD	generalized irregular low-density

GLD	generalized low-density
GSM	global system for mobile communication
HSPA	high speed packet access
LDPC	low-density parity-check
LLR	log-likelihood ratio
Log-FFT-BP	logarithmic fast Fourier transform belief propagation
MA- I	modified-algorithm I
MAP	maximum <i>a posteriori</i>
RS	Reed-Solomon
SISO	soft-input soft-output
SNR	signal-to-noise-ratio
SOVA	soft-output Viterbi algorithm
SPA	sum-product algorithm
SPC	single parity-check
TPC	turbo product code
WiMAX	worldwide interoperability for microwave access

# **CHAPTER 1**

## **INTRODUCTION**

### **1.1. Introduction**

Forward error correction (FEC) codes are used in digital communication systems to detect and correct the errors incurred during data transmission over noisy mediums. For this purpose, some extra information is added to the original message thus at the receiver side, they can be used in crosschecking the original message. It was shown by Shannon [1] in 1948 that by using FEC codes in communication systems, the error rate can be reduced to any negligible level. However, he did not specify any method for constructing good practical codes. Since then, the challenge for coding theorist has been to achieve the Shannon channel capacity limit by using the FEC codes, which have sufficient structure to be encoded and decoded practically.

The design of FEC codes began with the work of Hamming in 1950 [2] and was succeeded with the research on heavily structured codes for almost four decades. The structured codes are either algebraic as in most of the block codes or topological as in most of the convolutional codes. The decoding of a structured code is relatively easy and practical. Therefore, these codes are being used extensively in wireless communication. In deep space communication and satellite systems, where power-limited and low spectral efficiency codes are required, mainly serially concatenated structures are being used. The most commonly



used structured code in satellite communications is constructed from concatenation of an outer Reed-Solomon code over an inner convolutional code. In mobile communication systems where bandwidth is scarce and expensive, bandwidth-efficient coding schemes are required and for example, in global system for mobile communication (GSM) standard, a variety of channel coding schemes including Bose–Chaudhuri–Hochquenghem (BCH) codes, fire codes and cyclic redundancy check (CRC) codes have been used [3]. However, the performance of these structured codes is still far away from the channel capacity.

## 1.2. Turbo codes

In the mid-90s, a major discovery was made in the field of coding theory. This was motivated by the work of Shannon in [1], which showed that the randomly generated long-length codewords are capable of approaching the channel capacity. Traditionally, increasing the code's minimum distance was considered as the only solution for improving the performance of a code. However, in 1993, Berrou, Glavieux, and Thitimajshima [4] considered a different approach to reduce the bit-error rate (BER) of a code. In their approach, instead of increasing the minimum distance of a code, the multiplicity of the minimum weight codeword is reduced. This has resulted in a capacity approaching class of codes named as turbo codes.

Turbo codes are the first class of code that for almost any code rate can get as close as 1.0 dB to the additive white Gaussian noise (AWGN) channel capacity at the moderate BER of  $10^{-5}$ . Therefore, with the exception of very delay-sensitive applications, turbo codes have been employed in most industrial standards. In satellite communication systems, they are an alternative to the Reed-Solomon-Viterbi codes and in mobile communication, they are extensively being used in the third and fourth generations of mobile telephony standards such as high speed packet access (HSPA) and 3<sup>rd</sup> generation partnership project long term evolution (3GPP LTE) [5]. Moreover, turbo codes are the coding scheme for the worldwide interoperability for microwave access (WiMAX), which is a wireless metropolitan network standard [5].

The excellent performance of turbo codes has made them an exciting topic in channel coding theory. The main reason for this excellent performance is the random-like weight spectrum of a turbo code. This means that the distance distribution between any typical codeword of a turbo code and all other codewords of that code resembles the distance distribution for a randomly generated code. The random-like weight spectrum is caused by interleaving or reordering the information symbols (bits) during the encoding process. However, this random-like characteristic can be achieved only for long codes (e.g. with more than  $10^4$  symbols (bits) per codeword).

The decoding complexity for a code with random-like structure increases exponentially with its length. This means that decoding a turbo code is expected to be very complicated. However, turbo codes are cleverly constructed from combination of simple component codes so that their decoding complexity can be avoided and a long-length codeword can be decoded based on the decoding of its shorter and simpler component codes. Furthermore, by using soft-input soft-output (SISO) component decoders in an iterative decoding algorithm, the chance of losing information becomes less. This is because in conjunction with the channel information, the soft output values of the other component decoders, which are known as extrinsic information, can also be used in the decoding process.

The process of a SISO iterative decoding algorithm is initialized by decoding each constituent code individually, where each constituent code is decoded based on the channel information and the *a priori* information of its symbols (bits). Subsequently, the soft output information generated by each component decoder is shared with the other component decoders. In the next iteration, each component decoder uses the shared information of the other component decoders as extrinsic information to improve its data estimations. This process is iterated multiple times and each time the data estimation of every component decoder is improved by the help of extrinsic information received from the other component decoders. Eventually, after certain number of iterations, a hard decision for each symbol (bit) based on combination of the channel information and the extrinsic information for that symbol (bit) can be made.

### 1.3. Turbo-like codes

The concept of concatenated codes was introduced by Forney [6] and in fact, the serial concatenated codes were the best-known codes prior to the invention of turbo codes. Turbo codes were originally constructed as parallel concatenation of two interleaved convolutional codes. Further research showed that other capacity-approaching codes can be constructed from parallel concatenation of interleaved block codes [7] or from serial [8] or even hybrid serial-parallel [9] concatenation of different interleaved component codes. All these capacity-approaching codes can be categorized as turbo-like codes and although there is no generic definition for turbo-like codes, all these codes have the following characteristics:

- Compound structure: Turbo-like codes are constructed from multiple low-complexity component or constituent codes.
- Interleaving: The information symbols (bits) are reordered before being used by each constituent code.
- SISO iterative decoding: Turbo-like codes are decoded by an iterative decoding algorithm where the soft reliability information is repeatedly exchanged between the constituent codes.

Therefore, different turbo-like codes based on different type of constituent codes and concatenation structures can be designed. In the rest of this section, an overview on the literature for the two important categories of turbo-like codes is given. The first category is the turbo-like codes constructed from parallel concatenation of single parity-check (SPC) codes and the second category is the turbo-like codes with the product structure.

### 1.3.1. Low-density parity-check codes

Low-density parity-check (LDPC) codes were first discovered by Gallager in 1962 [10] and had the computing power been available at that time, these codes would have outperformed the best-known codes prior to the invention of turbo codes. However, besides few exceptions such as [11], they were largely neglected until the mid-90s when they were rediscovered by Mackey [12, 13] and shown to form a class of capacity approaching codes. An LDPC code is defined as the null space of an  $M \times N$  sparse parity-check matrix, where  $N$  is the length of the code and  $M$  is the number of parity-check equations. Therefore, an LDPC code can simply be considered as a turbo-like code constructed from  $M$  parallel SPC codes, where the information symbols (bits) are interleaved by the position of nonzero elements of the parity-check matrix.

LDPC codes were originally constructed from parity-check matrices with uniform column and row weights that in the literatures are known as regular LDPC codes. Later, Luby *et al.* [14] introduced the irregular LDPC codes, where their parity-check matrices have non-uniform row and column weights and they can outperform the regular LDPC codes. Furthermore, Davey and MacKay [15,16] studied the construction of non-binary LDPC codes and they showed that the performance of an LDPC code improves over higher order fields. Generally, LDPC codes are constructed either as pseudorandom (or random-like) codes [13-20] or as algebraic and combinatorial codes [21-39]. In most cases, the performance of a pseudorandom LDPC code is better than that of a structured LDPC code. However, a pseudorandom LDPC code does not have sufficient structure and its encoding is complicated. On the other hand, structured LDPC codes have encoding advantages over pseudorandom codes. For example, in structured quasi-cyclic LDPC codes [29-39], the encoders can be implemented from simple shift registers with linear complexity.

An LDPC code is often presented by a bipartite graph known as Tanner graph. A Tanner graph consists of two sets of nodes: variable nodes that correspond to the data and check nodes that correspond to the parity-check equations. A variable node is connected to a parity-check node if the corresponding variable is involved in the corresponding parity-check equation. The performance of an iteratively decoded LDPC code is very much depended on the length of its Tanner graph's shortest cycle [11]. The length of the shortest cycle in a

Tanner graph is known as the girth of that graph. In the design of an LDPC code, small girths, especially a girth of four, should be avoided. This is because the iterative decoding process in a code that has a girth of four becomes correlated after two iterations. It is shown that avoiding short cycles in non-binary LDPC codes is more feasible compared with their binary counterparts [40]. A well-designed LDPC code has a reasonably large-girth Tanner graph and code optimization, based on analyzing the code's behavior under the iterative decoding algorithm, is a crucial process in designing a good-performance LDPC code.

The density evolution method proposed by Gallager [10] is an optimization method and can be used as a practical technique for designing powerful LDPC codes. Richardson *et al.* [17] used the density evolution technique and determined a threshold on signal-to-noise-ratio (SNR) value for different LDPC codes, from which it was claimed that most LDPC codes under SISO iterative decoding algorithm are asymptotically good-performance codes above these SNR thresholds. The simulation results for large block length LDPC codes confirm these thresholds [18]. Moreover, by using density evolution technique, several well-performed  $\frac{1}{2}$  rate LDPC codes were designed [19], including one which its theoretical thresholds approaches within 0.0045 dB of the AWGN channel capacity, and another with the length of  $10^7$  bits that approaches within 0.04 dB of the AWGN channel capacity at BER of  $10^{-6}$ . Density evolution is one of the best-known methods for optimizing an LDPC code, however, it is a computationally intensive process which becomes intractable for non-binary LDPC codes that have alphabet size larger than three [17]. Therefore, other approximated optimization techniques such as Gaussian distribution algorithm [41,42] and extrinsic information transfer (EXIT) chart [43-45] are proposed. One of the fundamental assumptions in density evolution and other approximated techniques is that the channel should be symmetric. However, not all communication channels can be considered as symmetric. In [46], the non-symmetric channels are considered and a systematic approach for designing the non-binary LDPC codes for the memoryless channels is presented.

LDPC codes under SISO iterative decoding algorithm exhibit performance comparable and sometimes even better than turbo codes. The first SISO iterative decoding algorithm for binary LDPC codes was proposed in [10], which is based on probability distribution of the codeword variables over a graph-based model. Mackay *et al.* [12] also used a similar algorithm for decoding LDPC codes. Moreover, it is shown by Mackay that the simplification of the Pearl belief propagation (BP) [47] decoding algorithm over the

Tanner graph representation of a code, provides a powerful tool for decoding an LDPC code. It is shown in [48] that Gallager and Mackay decoding algorithms are specific instances of sum-product algorithm (SPA). Depending on the context, the SISO iterative decoding algorithm for an LDPC code can be called as SPA, BP or iterative message passing decoding algorithm.

A good-performance LDPC code has very long length codewords and implementing SPA algorithm for such a code is computationally complicated. Therefore, other SISO iterative decoding algorithms based on approximation of SPA have been proposed. Fossorier *et al.* in [49] proposed an approximation for SPA algorithm, which is known as min-sum or BP-based algorithm. The min-sum decoding algorithm significantly reduces the computational complexity of SPA and also degrades the performance of a code. In [50], several other reduced-complexity decoding algorithms for binary LDPC codes are proposed. In general, there is a trade-off between the performance of an optimized LDPC code and the complexity of its decoder. However, for an LDPC code whose Tanner graph contains many short cycles, the reduced-complexity decoding schemes may outperform the SPA decoding algorithm [50].

SPA decoding algorithm is also generalized for decoding non-binary LDPC codes [15,16]. However, the computational complexity for SPA decoding of an LDPC code defined over a finite field of order  $q$ ,  $\mathbb{F}_q$ , is dominated by  $\mathcal{O}(q^2)$  operations for each check sum calculation [16]. By employing a fast Fourier transform belief propagation (FFT-BP) decoding algorithm, the decoding complexity can be reduced [51]. Moreover, it is shown [52] that by describing FFT-BP algorithm in the logarithmic domain, a more simplified decoding algorithm, known as Log-FFT-BP, can be achieved. The computational complexity for decoding an LDPC code defined over  $\mathbb{F}_q$  (with  $q = 2^p$  for any positive integer  $p$ ) and decoded by Log-FFT-BP decoding algorithm, is reduced to  $\mathcal{O}(pq)$  operations for each check sum calculation. In [53] a more convenient description for FFT-BP and Log-FFT-BP based on the tensoral representation is given. Moreover, an approximation for FFT-BP algorithm, known as extended min-sum (EMS) decoding algorithm, is proposed in [54]. To reduce the amount of memory requirement for EMS decoding algorithm, a new implementation for this algorithm is presented in [55], however, similar to binary LDPC codes, there is a performance-complexity trade-off for decoding an optimized non-binary LDPC code.

LDPC codes are used in many industrial standards such as digital video broadcasting second generation (DVB-S2) [56], WiMAX (IEEE 802.16e) [57], G.hn standard [58] and ethernet cable transmission (10GBASE-T) [59]. A well-designed, long-length LDPC code performs only a few tenths of dB from the channel capacity; however, for the codes with less than  $10^4$  bits per codewords, turbo codes generally perform better than LDPC codes. Therefore, designing small and medium block length turbo-like codes constructed from SPC constituent code, which can perform close to channel capacity, is still an open research field. The generalized LDPC codes such as generalized low-density (GLD) [60,61] and generalized irregular low-density (GILD) [62] codes can be considered as turbo-like codes constructed from SPC constituent code. The parity-check matrices for both GLD and GILD are denser than that of the LDPC codes. The studies in [62] showed that GILD codes could perform as good as some LDPC codes and it is suggested that variations of GILD codes might be able to match or beat LDPC codes with small or medium block lengths.

### 1.3.2. Turbo product codes

The first idea for constructing long and powerful codes from a combination of short and simple codes dates back to the invention of product codes by Elias in 1954 [63]. Product codes can be constructed from any binary or non-binary, block or convolutional constituent codes in multiple dimensions, however, common choices are the two or three-dimensional product codes that are constructed from BCH codes, binary SPC codes or Hamming codes. A  $d$ -dimensional product code can be considered as a structured interleaved code where each of its information symbols (bits) is employed in  $d$  constituent codes.

The first soft decoding algorithm for a product code was given by Battail [64] and later Hagenauer *et al.* [65] showed that the turbo decoding principles can be used for decoding a product code. A turbo product code (TPC) is a product code that is decoded by a SISO iterative decoding algorithm and it is shown [66-70] that they can achieve good performance. Product codes are efficient for controlling both random-error and burst-error patterns [69] and are suggested for applications with high coding rate requirements such as submarine cables, optical transport networks and networks at 100Gbit/sec [70].

#### 1.4. Motivation for research

In designing a suitable coding scheme for mobile communication systems, in addition to the code performance, other factors such as bandwidth efficiency and quality of service must also be considered. In mobile systems, due to the increase in demand and growth in number of subscribers, bandwidth is scarce and expensive, and thus, codes with high coding rates are required. Also faster encoding and decoding techniques, which result in reducing time delays of the systems, and thus improving the quality of service, are required. However, existing good-performance LDPC and turbo codes are mostly low-rate codes and are computationally complicated [5,19]. This complexity mainly arises from the large block size of these codes. Therefore, there is a great necessity for designing good-performance high-rate codes with small or medium block size.

It is shown that the performance of a moderate-length LDPC code improves over higher order fields [15,16] and since LDPC codes are constructed as parallel concatenation of SPC codes, it can be imagined that codes which are designed based on concatenation of non-binary SPC codes may perform better over short-length or medium-length block size. Moreover, non-binary codes are attractive for high data-rate digital communication where high-order modulations are widely being used and it is more convenient to use non-binary codes with appropriate alphabet size to match the constellation. Furthermore, the research on GLD and GILD codes [60-62] shows that the parallel-concatenated SPC codes with denser parity-check matrices are better choices for designing small or medium length good-performance codes. Therefore, in this research we investigate compound codes that are constructed from non-binary SPC codes with average-density parity-check matrices.

The non-binary SPC component codes can be defined either over a finite field of order  $q$  or over a finite ring of order  $q$ . Furthermore, for designing a compound code, two issues have to be taken into consideration. One is the codeword-weight-distribution of the code and the other is the structure of the decoder. The multiplicity of the minimum weight codewords in a well-designed compound code needs to be reduced and the SISO iterative decoding algorithm needs to be simple so that the decoding latency can be avoided in the system. In this research, we show that these two conditions can be more satisfied when the compound codes are constructed from non-binary SPC codes defined over a finite ring of order  $q$ .



compared with those that are constructed from concatenation of non-binary SPC codes defined over a finite field of order  $q$ .

In this research, the design of non-binary compound codes with moderate to high coding rates is studied. The proposed compound codes are constructed from combination of non-binary SPC component codes. The SISO iterative decoding algorithms for decoding these compound codes are proposed. A SISO component decoder can be constructed from an optimal or sub-optimal *a posteriori* probability (APP) decoder. For non-binary codes, implementing an optimal APP decoder requires a large amount of memory.

In the first part of this research, we present a modification on the APP decoding algorithm to reduce its amount of memory requirements and computational complexity. The amount of memory requirement of the proposed algorithm is significantly less than memory requirement of the original APP decoder, thus the proposed modified APP decoding algorithm is more practical for decoding the non-binary linear block codes. The proposed algorithm is an optimum algorithm and it is not depended on the trellis structure of the code, thus it can be used for any linear block code.

The proposed class of compound codes that is discussed in this research is based on production of two non-binary SPC codes. Turbo product codes are efficient for applications where high coding rate schemes are required. We study the construction and decoding of the non-binary SPC turbo product codes when SPC codes are defined over a finite ring of order  $q$ . This includes non-binary SPC turbo product codes that are defined over finite fields, but this may be more general. The performance bounds for these codes are presented.

## 1.5. Thesis overview

This thesis is divided into six chapters. In Chapter 1, the major forward error correcting codes used in the wireless communication systems were presented. Turbo codes and turbo-like codes as the two important classes of capacity-approaching codes were briefly introduced. Furthermore, we explained the reasons for their extraordinary performance and discussed the issues that should be considered in their design process. Finally, the motivation for the work done in this thesis is discussed.

The optimal APP decoding algorithm for non-binary block codes is discussed in Chapter 2. An algorithm for calculating the symbols' APP values of a non-binary block code, which is based on the Fourier transform, is proposed. By employing this modified APP decoding algorithm, the amount of memory requirement and the computational complexity of the standard APP decoding algorithm are reduced and therefore, optimal APP decoding of a non-binary block codes becomes more feasible.

The non-binary SPC codes are studied in Chapter 3. The majority of the non-binary block codes are defined over the finite fields of order  $q$ . In this chapter, the construction of non-binary SPC codes when they are defined over the finite rings of order  $q$  is studied. Different decoding methods for these codes are proposed.

In Chapter 4, the construction of non-binary SPC turbo product codes is investigated. Different SISO iterative decoding methods for these codes are proposed. The simulation results show that SPC turbo product codes defined over finite rings perform better than SPC turbo product codes defined over finite fields. Moreover, by increasing the order of the ring the performance of non-binary SPC turbo product codes is improved.

The performance analysis of the non-binary SPC product codes are discussed in Chapter 5. The minimum Hamming distance of these codes and the multiplicity of the minimum Hamming weight codewords are presented. Based on the minimum distance of these codes, the upper bound for the bit-error rate is calculated and is compared with the simulation results.

Chapter Six presents the discussion on the conclusions drawn in this thesis.

## 1.6. Original contributions

The original contributions of this thesis include:

1. Derivation of optimum APP decoding algorithm for non-binary block codes to reduce the amount of memory storage and the computational complexity that is required for calculation of APP values of codewords' symbols. [Chapter 2]

2. Study of non-binary SPC codes over finite rings of order  $q$  and derivation of different decoding algorithm for these codes. [Chapter 3]
3. Design of non-binary turbo product codes which are constructed from non-binary SPC component codes defined over finite rings of order  $q$  and a study of the effect of ring order on the performance of an SPC turbo product code. [Chapter 4]
4. Derivation of minimum distance property and analytical bounds of non-binary SPC product codes. [Chapter 5]

### 1.7. Publications

The following publications have resulted from the work done in this thesis:

1. Ghayour F., Takawira F., “High-Rate Non-Binary Generalized Low-Density Parity-Check Codes”, in *Proc. Southern African Telecommunications Networks and Applications Conference (SATNAC)*, Wild Coast Sun, South Africa, Sep. 2008.
2. Ghayour F., Takawira F., “Structured Low-Density Parity-Check Codes for Next Generation Wireless Communications”, in *Proceeding of the 1<sup>st</sup> IEEE African Winter School on Information Theory and Communications*. Kruger National Park, South Africa, Jun 2010.
3. Ghayour F. Takawira F. and Xu H. “High-Rate Non-Binary Product Codes”, in *Proc. Southern African Telecommunications Networks and Applications Conference (SATNAC)*, East London, South Africa, Sep. 2011.
4. Ghayour F. Takawira F. and Xu H. “A Memory Efficient MAP Algorithm for Linear Block Codes”, in *Proc. IEEE AFRICON*, Livingstone, Zambia, Sep. 2011.
5. Ghayour F. Takawira F. and Xu H. “On MAP Decoding of High-Rate Non-Binary Linear Block Codes”, in *Proc. Southern African Telecommunications Networks and Applications Conference (SATNAC)*, George, South Africa, Sep. 2012.

6. Ghayour F. Takawira F. and Xu H. “On Decoding of Non-Binary Single Parity-check Codes”, Submitted to *IEEE Commun. Lett.*

## CHAPTER 2

### A POSTERIORI PROBABILITY ALGORITHMS

#### 2.1. Introduction

After the invention of turbo codes, researchers became interested in the design of iteratively decodable compound codes using soft-input soft-output (SISO) component decoders. The soft input of an optimal component decoder consists of symbol's *a priori* information, channel information and extrinsic information, while the soft output of the decoder consists of the *a posteriori* probability (APP) values for the symbols of that particular component code. The soft output information of each SISO component decoder is shared with the other component decoders, therefore, it can be used as extrinsic information in the next iteration. Subsequently, after certain number of iterations, the maximum *a posteriori* (MAP) decoder chooses the maximum APP values as the decoded symbols.

The standard way for calculating the symbol's APP values was proposed by Bahl-Cocke-Jelinek-Raviv (BCJR) [71], which requires storing of all state metric values. However, the computational complexity and the amount of memory requirement for the BCJR algorithm, which arises from the necessity of computing and storing of all state metric values, are often prohibitive in many practical applications.

For reducing the computational complexity of the MAP algorithm, both optimal algorithms, such as log-MAP [65,73,82], and sub-optimal algorithms, such as soft-output Viterbi algorithm (SOVA) [83] and max-log-MAP [65,72], have been proposed. However, employing sub-optimal decoding algorithms degrades the bit (or symbol) error rate performance of a code. In all these algorithms, whether optimal or suboptimal, although the complexity is reduced, the amount of memory requirement is the same as in the BCJR. This means that all the state metrics are required and have to be stored for the forward and backward recursions.

In [65,72,73], the BCJR algorithm was used for decoding linear block codes. However, implementing the BCJR algorithm for a linear block code usually requires large amount of memory storage. It is shown in [73] that for a linear block code with length  $N$  and dimension  $K$  defined over a finite field of order  $q$ , the number of states may reach up to  $q^{\min\{K, N-K\}}$  states. This means that implementing BCJR algorithm for such a code requires storing  $q^{\min\{K, N-K\}}$  numbers. This large amount of memory requirement makes BCJR an impractical decoding algorithm for many non-binary block codes and even for the codes with relatively small  $N$ . For example, to implement a BCJR decoder for a (31,21) Reed-Solomon code, at least  $32^{10}$  numbers require to be stored. Assuming that each number uses 32 bits of memory, more than  $4 \times 10^6$  gigabytes of memory storage for decoding this code is required. Storing this amount of data imposes great cost, energy consumption and time delay to the system.

Some attempts have been made in [74-77] to reduce the amount of memory requirement in a MAP decoder. It is shown in [74] that using all state metrics for computation of APP values is not required and instead, saving some parts of all state metric values is sufficient. This method is highly depended on the trellis graph of the code. This means that the efficiency of this method is completely related to the code's structure and it is effective only for codes with symmetrical trellis graphs like Reed-Muller codes. Another possible approach for decreasing memory requirement of the BCJR algorithm is presented in [75-77]. In this method, for computing the APP values of a symbol at a particular time, only the metric values for that time and some other adjacent states need to be stored. As it was shown in [73], the number of state metric values for some times may reach up to  $q^{\min\{K, N-K\}}$  states. This is still a large amount of memory requirement for most of non-binary block codes.

In this chapter, we present some modifications on the APP decoding algorithm in order to reduce its amount of memory requirement compared with the BCJR algorithm and therefore, to make it a feasible algorithm to be used for decoding non-binary codes. The reduction in the memory requirement is at the cost of increasing the computational complexity. However, this can be justified from the power consumption point of view, which as it was analyzed in [78,79], data storage and transfer operation consume much more power than data computation operation. Furthermore, the computational complexity of our proposed algorithm is less than that of the minimal-storage APP decoding algorithm and by use of our proposed algorithm, optimal APP decoding of any linear block code becomes feasible although it might force some latency to the system. Our proposed algorithm is not dependent on the trellis structure of a code, and therefore it can be implemented for any linear block code.

This chapter is organized as follows. In Section 2.2 the problem formulation is given. We then, in Section 2.3, provide a description of the MAP algorithm for block codes. In Section 2.4, the minimal-storage APP decoding algorithms and our proposed APP decoding algorithm are presented and the simulation results are given. Moreover, the computational complexity and the memory requirement for each algorithm are discussed. The final section concludes this chapter.

## 2.2. Background and notation

### 2.2.1. General notation

In this chapter, vectors and matrices are denoted in boldface letters and their elements in lower case; e.g.  $v_i$  is the  $i^{\text{th}}$  element of the vector  $\mathbf{v} = (v_0, v_1, \dots, v_{N-1})$ . An  $(N, K)$  code is referred to a non-binary linear block code with length  $N$  and dimension  $K$ . The non-binary codes are mostly defined over a finite field of order  $q$ ,  $\mathbb{F}_q$ , however, they can also be defined over finite rings or finite groups [80,81]. In this chapter, we consider non-binary codes defined over finite ring of integer modulo- $q$ ,  $\mathbb{Z}_q$ , as well as the non-binary codes that are defined over  $\mathbb{F}_q$ . Without loss of generality, all codewords are assumed to be in systematic

format so that the first  $K$  symbols of each codeword are the information symbols and the remaining  $N-K$  symbols are the parity-check symbols. The generator matrix of a systematic  $(N, K)$  code is represented by  $\mathbf{G} = [\mathbf{I}_K | \mathbf{P}]$ , where  $\mathbf{I}_K$  is a  $K \times K$  identity matrix and  $\mathbf{P}$  is a  $K \times (N-K)$  matrix whose elements belong to set  $\{0, 1, \dots, q-1\}$ . The parity-check matrix of a systematic  $(N, K)$  code is represented by  $\mathbf{H} = [-\mathbf{P}^T | \mathbf{I}_{N-K}] = [\mathbf{h}_0, \mathbf{h}_1, \dots, \mathbf{h}_{N-1}]$  where  $\mathbf{h}_i$  denotes the  $i^{\text{th}}$  column of  $\mathbf{H}$ . Furthermore, a sub matrix of  $\mathbf{H}$  that has the first  $i$  columns of  $\mathbf{H}$  is denoted by  $\mathbf{H}_i$  (i.e.  $\mathbf{H}_i = [\mathbf{h}_0, \mathbf{h}_1, \dots, \mathbf{h}_{i-1}]$ ). We assume that the codewords are transmitted over a discrete time memoryless, noisy channel and the soft decision vector  $\mathbf{r} = (r_0, r_1, \dots, r_{N-1})$  that corresponds to a transmitted codeword is available at the decoder.

### 2.2.2. Trellis for linear block codes

Let  $C$  be an  $(N, K)$  non-binary linear block code. A trellis diagram  $T$  for code  $C$  is a directed graph with  $N+1$  levels of vertices and  $N$  levels of edges. For any level  $\{i : 0 \leq i \leq N\}$ ,  $\mathcal{V}^i(C)$  denotes the set of all encoder states at the time  $i$  or in other words,  $\mathcal{V}^i(C)$  consist of all vertices at the  $i^{\text{th}}$  level of trellis  $T$ . Therefore, at the time 0 and at the time  $N$  only one node exists, which are respectively represented by  $S_0$  and  $S_f$ . (i.e.  $\mathcal{V}^0(C) = \{S_0\}$  and  $\mathcal{V}^N(C) = \{S_f\}$ ). Moreover, a branch or edge is a section of trellis between the  $i^{\text{th}}$  level and the  $(i+1)^{\text{th}}$  level that connects the  $S_i \in \mathcal{V}^i(C)$  node to the  $S_{i+1} \in \mathcal{V}^{i+1}(C)$  node and is labeled with a code symbol  $v_i$ , which represents the  $i^{\text{th}}$  element of the codeword vector. Each state in  $T$  can be labeled based on the code's parity-check matrix.

For code  $C$  with parity-check matrix  $\mathbf{H}$ , the label of the state  $S_i \in \mathcal{V}^i(C)$  is represented by  $l(S_i)$  and is defined as  $l(S_i) = \mathbf{a} \odot \mathbf{H}_i^T$ , where  $\mathbf{a}$  is the path in trellis  $T$  that starts from initial state,  $S_0$ , and terminates at the state  $S_i$ ; and  $\odot$  denotes the inner product between the vectors. As an example: consider the (7,4) binary Hamming code with the parity-check matrix given by (2.1). The trellis graph for this code is shown in Figure 2.1, where the vertical axis shows the label of each state and the horizontal axis shows the level of the trellis.



$$\mathbf{H} = \begin{bmatrix} 1101100 \\ 1011010 \\ 1110001 \end{bmatrix} \quad (2.1)$$

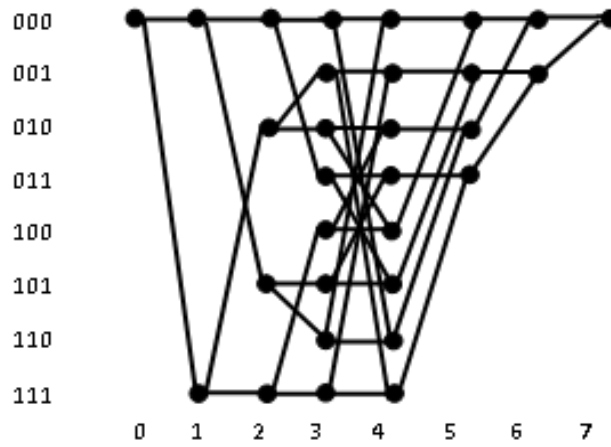


Figure 2.1: Trellis diagram of the (7,4) Hamming code given by the parity-check matrix in (2.1)

Two states that are connected to each other via one branch are called adjacent states. Therefore, a directed path from the initial node  $S_0$  to the final node  $S_f$  with a label sequence  $(v_0, v_1, \dots, v_{N-1})$  exists if and only if  $(v_0, v_1, \dots, v_{N-1})$  is a codeword belonging to  $\mathcal{C}$ . The set of all edges between the state space  $\mathcal{V}^i(\mathcal{C})$  and the state space  $\mathcal{V}^{i+1}(\mathcal{C})$  is denoted by  $\mathcal{E}^i(\mathcal{C})$ . Moreover,  $\mathcal{E}^i(\mathcal{C})$  can be spanned to subsets  $\{\mathcal{E}_j^i(\mathcal{C}) : 0 \leq j \leq q-1\}$  where  $\mathcal{E}_j^i(\mathcal{C})$  corresponds to the symbol  $v_i = j$ . Clearly based on the structure of a non-binary linear block code,  $\{\mathcal{E}^i(\mathcal{C}) = \bigcup_{j=0}^{q-1} \mathcal{E}_j^i(\mathcal{C}) : 0 \leq i \leq N-1\}$ . Let  $|\mathcal{E}_j^i(\mathcal{C})|$  denote the cardinality of the subset  $\mathcal{E}_j^i(\mathcal{C})$ , therefore, we can write

$$|\mathcal{E}_0^i(\mathcal{C})| = |\mathcal{E}_1^i(\mathcal{C})| = \dots = |\mathcal{E}_{q-1}^i(\mathcal{C})| \quad (2.2)$$

### 2.3. Computation of the MAP decoding algorithm

The MAP value for the symbol  $v_i$  of any codeword  $\mathbf{v} = (v_0, v_1, \dots, v_{N-1})$  that belongs to code  $C$  can be calculated as

$$\hat{v}_i = \arg \max_{\mu \in \{0,1,\dots,q-1\}} \left\{ \sum_{\substack{\mathbf{v} \in C \\ v_i = \mu}} Pr(\mathbf{v}|\mathbf{r}) \right\} \quad (2.3)$$

where  $\mathbf{r} = (r_0, r_1, \dots, r_{N-1})$  is the received sequence and  $Pr(\mathbf{v}|\mathbf{r})$  represents the *a posteriori* probability for the codeword  $\mathbf{v}$ . Moreover,  $\sum_{\substack{\mathbf{v} \in C \\ v_i = \mu}} (\cdot)$  denotes the summation for all codewords such as  $\mathbf{v} = (v_0, v_1, \dots, v_{N-1})$  that belongs to code  $C$  and their  $i^{\text{th}}$  symbol is equal to  $\mu$ . Based on BCJR [71], (2.3) can be calculated as

$$\hat{v}_i = \arg \max_{\mu \in \{0,1,\dots,q-1\}} \left\{ \sum_{(S',S) \in \mathcal{E}_\mu^i} \alpha_i(S') \gamma_i(S', S) \beta_{i+1}(S) \right\} \quad (2.4)$$

and by defining  $\{\mathbf{r}_{i,j} = (r_i, r_{i+1}, \dots, r_{j-1}) \quad : \quad 0 \leq i \leq j \leq N\}$  we have

$$\alpha_i(S') = Pr(S_i = S', \mathbf{r}_{0,i}) \quad (2.5)$$

$$\beta_i(S) = Pr(\mathbf{r}_{i,N} | S_i = S) \quad (2.6)$$

$$\gamma_i(S', S) = Pr(S_{i+1} = S, r_i | S_i = S') \quad (2.7)$$

Let  $\Omega_{i-1}(S)$  represent a set of all states at the  $(i-1)^{\text{th}}$  level of trellis  $T$ , which these states are adjacent to state  $S$ . Here  $S$  is a state at the  $i^{\text{th}}$  level of trellis  $T$ . According to the BCJR algorithm for state  $S \in \mathcal{V}^i(C)$  at  $i^{\text{th}}$  level of trellis  $T$ ,  $\alpha_i$  can be calculated recursively as

$$\alpha_i(S) = \sum_{S' \in \Omega_{i-1}(S)} \alpha_{i-1}(S') \gamma_i(S', S) \quad 0 \leq i \leq N \quad (2.8)$$

with the initial value of  $\alpha_0(S_0) = 1$ . Similarly, let  $\Omega_{i+1}(S)$  denote a set of all states at  $(i+1)^{\text{th}}$  level of trellis  $T$ , which these states are adjacent to state  $S$ . Here  $S$  is a state at the  $i^{\text{th}}$  level of trellis  $T$ . Therefore, for state  $S \in \mathcal{V}^i(C)$  at  $i^{\text{th}}$  level of trellis  $T$ ,  $\beta_i$  can be calculated recursively as

$$\beta_i(S) = \sum_{S' \in \Omega_{i+1}(S)} \gamma_i(S, S') \beta_{i+1}(S') \quad 0 \leq i \leq N \quad (2.9)$$

with  $\beta_N(S_f) = 1$ . Calculation of  $\{\alpha_i : 0 \leq i \leq N\}$  is known as the forward recursion and calculation of  $\{\beta_i : 0 \leq i \leq N\}$  is known as the backward recursion. Finally, for a block code with statistically independent information symbols, the branch transition probability used in (2.4) can be calculated as

$$\begin{aligned} \gamma_i(S', S) &= Pr(S_{i+1} = S, r_i | S_i = S') \\ &= Pr(S_{i+1} = S | S_i = S') Pr(r_i | S_{i+1} = S, S_i = S') \\ &= Pr(v_i = \mu) Pr(r_i | \mu) \end{aligned} \quad (2.10)$$

Although different methods for implementing the MAP decoding algorithm have been suggested, the following three major steps are common among all of them:

1. Performing the forward recursion process and storing all the values calculated for  $\{\alpha_i : 0 \leq i \leq N\}$ .
2. Performing the backward recursion process and storing all the values calculated for  $\{\beta_i : 0 \leq i \leq N\}$ .
3. For each received symbol,  $r_i$ , calculating the maximum APP values from (2.4), using the transition probabilities.

Due to the independence of forward and backward recursions from each other, step 1 and step 2 can be done simultaneously.

## 2.4. Modified APP decoding algorithms

As it was shown in the previous section, the main drawback with the original MAP algorithm is its large amount of memory requirement, especially for non-binary block codes. This memory requirement is often costly and prohibitive in many practical applications. In this section, first, we discuss the minimal-storage APP decoding algorithms and later we present our proposed decoding algorithm. The minimum amount of memory requirement for calculating the symbol's APP values of any linear block code can be achieved by straightforward implementation of the trellis [65,82], while for a high-rate code defined over  $\mathbb{Z}_q$ , this can be achieved by the APP decoding algorithm implemented based on the code's dual space [65,84]. This means that for any linear block code defined over  $\mathbb{F}_q$ , regardless of its rate, the minimum amount of memory requirement for calculating the symbol's APP values can be achieved by straightforward implementation of the trellis. However, for linear block codes defined over  $\mathbb{Z}_q$ , the minimal-storage APP decoding algorithm can be achieved, depending on the code rate, either by straightforward implementation of the trellis for low-rate codes ( $K < N-K$ ) or by the APP decoding algorithm implemented based on the code's dual space for high-rate codes ( $K > N-K$ ).

Furthermore, we modify the minimal-storage APP decoding algorithm for high-rate codes defined over  $\mathbb{Z}_q$  to reduce the computational complexity of the algorithm. Based on this modification, the discrete Fourier transform (DFT) vectors are employed to limit the repetitive calculations. Therefore, prior to calculation of symbol's APP values, a DFT vector corresponding to each received symbol is calculated and stored in memory, thus they can be retrieved and used during the calculation. Compared with the minimal-storage APP decoding algorithm for high-rate codes defined over  $\mathbb{Z}_q$ , the memory requirement of this modified algorithm is slightly increased, but its computational complexity is reduced by the factor of  $q$ .

#### 2.4.1. APP decoding algorithm based on straightforward implementation

This algorithm can be implemented for any non-binary linear block code. The APP value for the symbol  $v_i$  of any codeword  $\mathbf{v} = (v_0, v_1, \dots, v_{N-1})$  that belongs to code  $\mathcal{C}$  can be calculated as

$$\Lambda_{i,\mu} = \sum_{\substack{\mathbf{v} \in \mathcal{C} \\ v_i = \mu}} Pr(\mathbf{v}|\mathbf{r}) \quad (2.11)$$

where  $\mathbf{r} = (r_0, r_1, \dots, r_{N-1})$  is the received sequence and subsequently after calculation of  $\Lambda_{i,\mu}$ , the hard decision is made by  $\hat{v}_i = \arg \max_{\mu \in \{0,1,\dots,q-1\}} \{\Lambda_{i,\mu}\}$ . Based on Bayes' rule we have

$$\Lambda_{i,\mu} = \sum_{\substack{\mathbf{v} \in \mathcal{C} \\ v_i = \mu}} \left[ \frac{Pr(\mathbf{r}|\mathbf{v})Pr(\mathbf{v})}{Pr(\mathbf{r})} \right] \quad (2.12)$$

Symbols are transmitted over a memoryless channel. Thus,

$$\Lambda_{i,\mu} = \frac{1}{Pr(\mathbf{r})} \sum_{\substack{\mathbf{v} \in \mathcal{C} \\ v_i = \mu}} \left[ \prod_{j=0}^{N-1} [Pr(r_j|v_j)Pr(v_j)] \right] \quad (2.13)$$

which can be simplified to

$$\Lambda_{i,\mu} = \frac{Pr(\mu)Pr(r_i|\mu)}{Pr(\mathbf{r})} \sum_{\substack{\mathbf{v} \in \mathcal{C} \\ j \neq i}} \left[ \left( \prod_{j=0}^{N-1} [Pr(r_j|v_j)Pr(v_j)] \right) \delta_{0,(\mathbf{e}_i \odot \mathbf{v}) \oplus \mu} \right] \quad (2.14)$$

where depending on the definition of the code,  $\oplus$  denotes addition over  $\mathbb{F}_q$  or  $\mathbb{Z}_q$  and  $\delta_{i,j}$  denotes the Kronecker delta, which is equal to one if  $i = j$  and zero otherwise. Moreover,  $\mathbf{e}_i = (\delta_{0,i}, \delta_{1,i}, \delta_{2,i}, \dots, \delta_{N-1,i})$  is a vector with one at the  $i^{\text{th}}$  position and zero elsewhere. Based on trellis property of linear block codes which is given in (2.2), for all codewords of code  $\mathcal{C}$ , the probability that each of  $\{0, 1, \dots, q-1\}$  symbols occurs as a parity-check symbol is the same, and since all codewords are in systematic format,  $\left\{ Pr(v_j) = \frac{1}{q} \quad : \quad K \leq j \leq N-1 \right\}$ . Thus,

$$\Lambda_{i,\mu} = \frac{Pr(\mu)Pr(r_i|\mu)}{Pr(\mathbf{r}) q^{N-K}} \sum_{\substack{\mathbf{v} \in \mathcal{C} \\ j \neq i}} \left( \left[ \prod_{j=0}^{N-1} Pr(r_j|v_j) \prod_{\substack{l=0 \\ l \neq i}}^{K-1} Pr(v_l) \right] \delta_{0,(\mathbf{e}_i \odot \mathbf{v}) \oplus -\mu} \right) \quad (2.15)$$

where  $-\mu$  is the additive inverse of  $\mu$  such that  $\mu \oplus -\mu = 0$ .

Equation (2.15) is a general formula to find the APP value for the symbol  $v_i$  when the codewords are transmitted over a memoryless channel. Compared with the BCJR algorithm, which is the standard way of calculating the symbol's APP values and is realized by (2.4), the amount of memory that is required by (2.15) is significantly reduced. Based on (2.4), for calculating the symbols' APP values, all  $\{\alpha_i : 0 \leq i \leq N\}$  and  $\{\beta_i : 0 \leq i \leq N\}$  need to be calculated and stored in memory prior to being used in (2.4), which as it was mentioned before, requires a large amount of memory and it is often prohibitive in many practical applications. However, for calculating the symbol's APP values based on (2.15), no prior calculations are required and only  $K$  memory cells for storing the information symbols to generate the codewords of the code space are sufficient.

Furthermore, for reducing the computational complexity of (2.15), we can use the log-likelihood ratio (LLR) value for each symbol. LLR in [82] is defined as

$$L_{i,\mu} \stackrel{\text{def}}{=} \ln \left( \frac{\Lambda_{i,\mu}}{\Lambda_{i,0}} \right) \quad (2.16)$$

and subsequently the hard decision is made by  $\hat{v}_i = \arg \max_{\mu \in \{0,1,\dots,q-1\}} \{L_{i,\mu}\}$ . In addition, LLR can be defined for a pair of joint random variables; for example  $L_\mu(v_i; \mathbf{r})$ , where  $v_i$  is a random variable and  $\mathbf{r}$  is a vector of random variables is given by

$$L_\mu(v_i; \mathbf{r}) \stackrel{\text{def}}{=} \ln \left( \frac{Pr(v_i = \mu; \mathbf{r})}{Pr(v_i = 0; \mathbf{r})} \right) \quad (2.17)$$

By using the concept of LLR we have

$$L_{i,\mu} = L_\mu(v_i; r_i) + \ln \left( \frac{\sum_{v \in C} \left[ \left( \prod_{\substack{j=0 \\ j \neq i}}^{N-1} e^{L_{v_j}(v_j; r_j)(1-\delta_{0,v_j})} \right) \delta_{0, (e_i \odot v) \oplus -\mu} \right]}{\sum_{v \in C} \left[ \left( \prod_{\substack{j=0 \\ j \neq i}}^{N-1} e^{L_{v_j}(v_j; r_j)(1-\delta_{0,v_j})} \right) \delta_{0, e_i \odot v} \right]} \right) \quad (2.18)$$

For the special case of binary codes where the codewords are modulated by binary phase-shift keying (BPSK) modulation scheme (such that,  $\{0,1\}$  bits are mapped to  $\{-1,1\}$  respectively) and are transmitted over an additive white Gaussian noise (AWGN) channel with double-sided noise power spectral density of  $\sigma^2$ ; (2.18) can be simplified to (2.19)

$$L_{i,1} = L_1(v_i) + \frac{2r_i}{\sigma^2} + \ln \left( \frac{\sum_{v \in C} \left[ \left( \prod_{\substack{j=0 \\ j \neq i}}^{N-1} e^{L_1(v_j; r_j)(1-\delta_{0,v_j})} \right) v_i \right]}{\sum_{v \in C} \left[ \left( \prod_{\substack{j=0 \\ j \neq i}}^{N-1} e^{L_1(v_j; r_j)(1-\delta_{0,v_j})} \right) (1 - v_i) \right]} \right) \quad (2.19)$$

where

$$\begin{aligned} L_1(v_i; r_i) &= L_1(v_i) + L_1(v_i | r_i) \\ &= \ln \left( \frac{Pr(v_i = 1)}{Pr(v_i = 0)} \right) + \ln \left( \frac{Pr(v_i = 1 | r_i)}{Pr(v_i = 0 | r_i)} \right) \\ &= \ln \left( \frac{Pr(v_i = 1)}{Pr(v_i = 0)} \right) + \ln \left( \frac{\frac{1}{\sqrt{2\pi\sigma^2}} \exp \left( \frac{-(r_i - 1)^2}{2\sigma^2} \right)}{\frac{1}{\sqrt{2\pi\sigma^2}} \exp \left( \frac{-(r_i + 1)^2}{2\sigma^2} \right)} \right) \\ &= \ln \left( \frac{Pr(v_i = 1)}{Pr(v_i = 0)} \right) + \frac{2r_i}{\sigma^2} \end{aligned} \quad (2.20)$$



### 2.4.2. APP decoding algorithm based on the code's dual space

As it was mentioned in the previous section, for a high-rate linear block code defined over  $\mathbb{Z}_q$ , the minimal-storage APP decoding algorithm can be achieved by the APP decoding algorithm implemented based on the code's dual space. The original idea for calculating the symbol's APP values over the code's dual space can be found in [84]. Further, this algorithm was extended so that it can be employed as SISO decoder for binary [65] and non-binary codes [82]. The proof for this algorithm is presented in [84] for a code  $C$  defined over  $\mathbb{F}_q$  with modulo- $q$  addition. This condition can be satisfied only if  $q$  is prime and  $\mathbb{F}_q = \mathbb{Z}_q$ . However, in [82] this algorithm is extended to  $\mathbb{F}_{2^p}$  ( $p$  is any positive integer), which mathematically cannot be correct and the algorithm is valid only for the codes defined over  $\mathbb{Z}_{2^p}$ .

In this section, a different approach from [82] for calculating the symbol's APP values of a non-binary code using the code's dual space is presented. Let  $C'$  be the dual code for the code  $C$  and  $\mathbf{v}' = (v'_0, v'_1, \dots, v'_{N-1})$  denotes a codeword of  $C'$ . Based on orthogonality, the inner product between any codeword in  $C$  and any codeword in  $C'$  is zero. Therefore, for any  $N$ -tuple  $\mathbf{u} = (u_0, u_1, \dots, u_{N-1})$  defined over  $\mathbb{Z}_q$  we have

$$\sum_{\mathbf{v}' \in C'} e^{j\frac{2\pi}{q}\mathbf{u} \odot \mathbf{v}'} = \begin{cases} q^{N-K}; & \mathbf{u} \in C \\ 0 & ; \text{ otherwise} \end{cases} \quad (2.21)$$

and

$$\sum_{\substack{\mathbf{v} \in C \\ v_i = \mu}} Pr(\mathbf{v}|\mathbf{r}) = \frac{1}{q^{N-K}} \sum_{\substack{\mathbf{u} \in V^N \\ u_i = \mu}} \left[ Pr(\mathbf{u}|\mathbf{r}) \sum_{\mathbf{v}' \in C'} e^{j\frac{2\pi}{q}\mathbf{u} \odot \mathbf{v}'} \right] \quad (2.22)$$

where  $\mathbf{V}^N$  is the vector space of all  $N$ -tuples defined over  $\mathbb{Z}_q$ . Therefore,  $\Lambda_{i,\mu}$  can be calculated as

$$\Lambda_{i,\mu} = \frac{1}{q^{N-K}} \sum_{\substack{\mathbf{u} \in \mathbf{V}^N \\ u_i = \mu}} \left( Pr(\mathbf{u}|\mathbf{r}) \sum_{v' \in C'} e^{j\frac{2\pi}{q}\mathbf{u} \odot v'} \right) \quad (2.23)$$

Based on Bayes' rule and by using the Kronecker delta representation we have

$$\Lambda_{i,\mu} = \frac{1}{q^{N-K} Pr(\mathbf{r})} \sum_{\mathbf{u} \in \mathbf{V}^N} \left[ Pr(\mathbf{u}; \mathbf{r}) (\delta_{0,(\mathbf{e}_i \odot \mathbf{u}) \oplus -\mu}) \sum_{v' \in C'} e^{j\frac{2\pi}{q}\mathbf{u} \odot v'} \right] \quad (2.24)$$

By the orthogonality properties of  $\left\{ e^{j\frac{2\pi}{q}li} : i = 0, 1, \dots, q-1 \right\}$  we have

$$\Lambda_{i,\mu} = \frac{1}{q^{N-K} Pr(\mathbf{r})} \sum_{\mathbf{u} \in \mathbf{V}^N} \left[ Pr(\mathbf{u}; \mathbf{r}) \left( \frac{1}{q} \sum_{l=0}^{q-1} e^{j\frac{2\pi}{q}l((\mathbf{e}_i \odot \mathbf{u}) \oplus -\mu)} \right) \sum_{v' \in C'} e^{j\frac{2\pi}{q}\mathbf{u} \odot v'} \right] \quad (2.25)$$

The inner product between the vectors can be written as

$$e^{j\frac{2\pi}{q}\mathbf{u} \odot v'} = e^{j\frac{2\pi}{q}((u_0 \otimes v'_0) \oplus (u_1 \otimes v'_1) \oplus \dots \oplus (u_{N-1} \otimes v'_{N-1}))} \quad (2.26)$$

and only if  $\oplus$  is defined as modulo- $q$  addition then we have

$$e^{j\frac{2\pi}{q}\mathbf{u}\odot\mathbf{v}'} = \prod_{m=0}^{N-1} e^{j\frac{2\pi}{q}(u_m\otimes v'_m)} \quad (2.27)$$

The modulo- $q$  addition and multiplication over  $\mathbb{Z}_q$  are denoted by  $\oplus$  and  $\otimes$  respectively. Since the codewords are transmitted over a memoryless channel and the message symbols are statistically independent,

$$\Lambda_{i,\mu} = \frac{1}{q^{N-K} Pr(\mathbf{r})} \sum_{\mathbf{v}' \in \mathcal{C}'} \left[ \prod_{m=0}^{N-1} \sum_{n=0}^{q-1} \left[ Pr(n; r_m) e^{j\frac{2\pi}{q}n\otimes v'_m} \left( \frac{1}{q} \sum_{l=0}^{q-1} e^{j\frac{2\pi}{q}l\delta_{im}(n\oplus-\mu)} \right) \right] \right] \quad (2.28)$$

For  $m \neq i$ ,  $\frac{1}{q} \sum_{l=0}^{q-1} e^{j\frac{2\pi}{q}l\delta_{im}(n\oplus-\mu)} = 1$  and for  $m = i$ ,  $\frac{1}{q} \sum_{l=0}^{q-1} e^{j\frac{2\pi}{q}l\delta_{im}(n\oplus-\mu)} = 0$  unless  $(n\oplus-\mu) = 0$  or equivalently  $n = \mu$ . Therefore,

$$\Lambda_{i,\mu} = \frac{Pr(\mu)Pr(r_i|\mu)}{q^{N-K} Pr(\mathbf{r})} \sum_{\mathbf{v}' \in \mathcal{C}'} \left[ \left( e^{j\frac{2\pi}{q}\mu\otimes v'_i} \right) \left( \prod_{\substack{m=0 \\ m \neq i}}^{N-1} \sum_{n=0}^{q-1} \left[ Pr(n; r_m) e^{j\frac{2\pi}{q}n\otimes v'_m} \right] \right) \right] \quad (2.29)$$

Equation (2.29) is the general formula to find the APP value for the symbol  $v_i$ . Compared with the BCJR algorithm, which is the standard way of calculating the symbol's APP values and is realized by (2.4), the amount of memory requirement that is required by (2.29) is significantly reduced. For calculating the symbol's APP values using (2.4), all  $\{\alpha_i : 0 \leq i \leq N\}$  and  $\{\beta_i : 0 \leq i \leq N\}$  need to be calculated and stored in memory prior to be used in (2.4), which as it was mentioned before, is a large amount of memory requirement and is often prohibitive in many practical applications. However, by using (2.29), no prior calculations are required and the symbol's APP values can directly be

calculated from the transition probabilities. The total number of memory cells required by (2.29) is  $N - K$  memory cells so that all the codewords of the dual code space can be generated. Although  $\Lambda_{i,\mu}$  can directly be calculated from (2.15) and (2.29), there are two issues that need to be considered. The first issue is that (2.29) is valid only for linear block codes defined over  $\mathbb{Z}_q$ , while (2.15) can be implemented for any linear block code. The second issue is that the summation in (2.15) is over the code's space, while the summation in (2.29) is over the code's dual space. Therefore, for decoding the linear block codes defined over  $\mathbb{Z}_q$ , the minimal-storage APP decoding algorithm can be achieved depending on the code rate, either by (2.15) for low-rate codes ( $K < N-K$ ) or by (2.29) for high-rate codes ( $K > N-K$ ).

Furthermore, for reducing the computational complexity of (2.29), the concept of LLR can be used. Therefore,

$$\begin{aligned}
 L_{i,\mu} &= L_\mu(v_i; r_i) + \ln \left( \frac{\sum_{v' \in C'} \left[ \left( e^{j\frac{2\pi}{q}\mu \otimes v'_i} \right) \left( \prod_{\substack{m=0 \\ m \neq i}}^{N-1} \sum_{n=0}^{q-1} \left[ Pr(n; r_m) e^{j\frac{2\pi}{q}n \otimes v'_m} \right] \right) \right]}{\sum_{v' \in C'} \left[ \prod_{\substack{m=0 \\ m \neq i}}^{N-1} \sum_{n=0}^{q-1} \left[ Pr(n; r_m) e^{j\frac{2\pi}{q}n \otimes v'_m} \right] \right]} \right) \\
 &= L_\mu(v_i; r_i) + \ln \left( \frac{\sum_{v' \in C'} \left[ \left( e^{j\frac{2\pi}{q}\mu \otimes v'_i} \right) \left( \prod_{\substack{m=0 \\ m \neq i}}^{N-1} \sum_{n=0}^{q-1} \left[ e^{L_n(v_m; r_m)} e^{j\frac{2\pi}{q}n \otimes v'_m} \right] \right) \right]}{\sum_{v' \in C'} \left[ \prod_{\substack{m=0 \\ m \neq i}}^{N-1} \sum_{n=0}^{q-1} \left[ e^{L_n(v_m; r_m)} e^{j\frac{2\pi}{q}n \otimes v'_m} \right] \right]} \right) \quad (2.30)
 \end{aligned}$$

For the special case of binary codes whose codewords are modulated by BPSK scheme (such that  $\{0,1\}$  bits are mapped to  $\{-1,1\}$  respectively) and are transmitted over an AWGN channel with double-sided noise power spectral density of  $\sigma^2$ , (2.30) can be simplified to (2.31).

$$\begin{aligned}
L_{i,1} &= L_1(v_i) + \frac{2r_i}{\sigma^2} + \ln \left( \frac{\sum_{v' \in C'} \left[ (1 - 2v'_i) \left( \prod_{\substack{m=0 \\ m \neq i}}^{N-1} \left[ 1 + (1 - 2v'_m) e^{\left(\frac{2r_m}{\sigma^2}\right)} \right] \right) \right]}{\sum_{v' \in C'} \left[ \prod_{\substack{m=0 \\ m \neq i}}^{N-1} \left[ 1 + (1 - 2v'_m) e^{\left(\frac{2r_m}{\sigma^2}\right)} \right] \right]} \right) \\
&= L_1(v_i) + \frac{2r_i}{\sigma^2} + \ln \left( \frac{\sum_{v' \in C'} \left[ (1 - 2v'_i) \left( \prod_{\substack{m=0 \\ m \neq i}}^{N-1} \left[ \left( \frac{1 - e^{\left(\frac{2r_m}{\sigma^2}\right)}}{1 + e^{\left(\frac{2r_m}{\sigma^2}\right)}} \right)^{v'_m} \right] \right) \right]}{\sum_{v' \in C'} \left[ \prod_{\substack{m=0 \\ m \neq i}}^{N-1} \left[ \left( \frac{1 - e^{\left(\frac{2r_m}{\sigma^2}\right)}}{1 + e^{\left(\frac{2r_m}{\sigma^2}\right)}} \right)^{v'_m} \right] \right]} \right) \quad (2.31)
\end{aligned}$$

#### 2.4.3. APP decoding algorithm based on Fourier transform

The minimal-storage APP decoding algorithm for high-rate codes defined over  $\mathbb{Z}_q$  was presented in the previous section. In this section, we use the concept of Fourier transform to reduce the computational complexity of that algorithm. The idea of using Fourier transform for decoding codes can also be found in decoding of the low-density parity-check (LDPC) codes [51-54]. However, in LDPC codes, the Fourier transform is used to convert the convolution operation into multiplication operation and thus, simplify the decoding algorithm, while in this section we use the concept of Fourier transform to avoid the repetitive calculation and reduce the computational complexity of the decoding algorithm. This proposed algorithm can be used for any linear block code defined over  $\mathbb{Z}_q$ , but it is particularly attractive for high-rate codes. The symbol's APP values for any codeword of code  $C$  can directly be calculated from (2.29). However, by using the concept of DFT, the repetitive calculations can be avoided. Let us consider a  $q$ -dimensional vector  $\{e^{L_n(v_m; r_m)} : 0 \leq n \leq q-1\}$  and define the DFT for  $\{e^{L_n(v_m; r_m)}\}$  as

$$\mathcal{F}_m(s) = \mathcal{F}\{e^{L_n(v_m; r_m)}\} \stackrel{\text{def}}{=} \sum_{n=0}^{q-1} e^{L_n(v_m; r_m)} e^{j \frac{2\pi}{q} n \otimes s} \quad (2.32)$$

Therefore, the vector of log-likelihood ratios  $L_i = \{L_{i,\mu}\}$  for the information symbol,  $v_i$ , can be calculated as

$$L_{i,\mu} = L_\mu(v_i; r_i) + \ln \left( \frac{\sum_{v' \in C'} \left[ \left( e^{j \frac{2\pi}{q} \mu \otimes v'_i} \right) \prod_{\substack{m=0 \\ m \neq i}}^{N-1} \mathcal{F}_m(v'_m) \right] \right)}{\sum_{v' \in C'} \left[ \prod_{\substack{m=0 \\ m \neq i}}^{N-1} \mathcal{F}_m(v'_m) \right]} \right) \quad (2.33)$$

Based on our proposed algorithm, prior to the computation of  $L_{i,\mu}$ , the DFT vectors for all received symbols need to be calculated and stored in memory, thus for computation of  $L_{i,\mu}$ , these DFT vectors can be retrieved and used in (2.33). Even though, compared with the minimal-storage APP decoding algorithm given by (2.30), an additional amount of memory for storing the DFT vectors is required, as we show in the next section, the computational complexity of the proposed algorithm is less than that of the minimal-storage APP decoding algorithm. Furthermore, the amount of memory requirement that is required by (2.33) for calculating the symbol's APP values is significantly less than that of the standard way, which is the BCJR algorithm and is realized by (2.4). For calculating the symbol's APP values based on (2.33),  $N-1$  DFT vectors need to be calculated and stored prior to be used in (2.33). Each DFT vector has  $q$  arrays and since the arrays are complex, two memory cells for storing each array are required. Therefore, the total memory requirement of (2.33) is  $(N - K) + 2(N - 1)q$  memory cells.

Our proposed algorithm is valid only for the finite fields with modulo- $q$  addition and therefore, it cannot be used for the codes that are defined over finite field of order  $2^p$  ( $p$  is any positive integer). However, it can be employed for the codes that are defined over  $\mathbb{Z}_{2^p}$ . Moreover, using fast Fourier transform (FFT) algorithms is more efficient for calculating the DFT vectors when  $q = 2^p$ . The computational complexity for calculating  $\mathcal{F}_m(s) = \mathcal{F}\{e^{L_n(v_m; r_m)}\}$  is  $\mathcal{O}(q^2)$ , while by using an FFT algorithm, the same results are obtained by only  $\mathcal{O}(q \log_2(q))$  operations and therefore, the overall computational complexity is reduced.

#### 2.4.4. Complexity comparison

In this section, we compare the complexity between the BCJR algorithm, the minimal-storage APP decoding algorithms and the proposed APP decoding algorithm. The complexity of an algorithm can be measured by the amount of memory requirement and the number of real operations that is required for computation of  $L_{i,\mu}$ .

According to the original MAP algorithm, the results for the forward or backward recursion must be stored in memory. Therefore, for storing  $\{\alpha_i : 0 \leq i \leq N\}$  or  $\{\beta_i : 0 \leq i \leq N\}$ ,  $\sum_{i=0}^{N-1} |\mathcal{V}^i(C)|$  memory cells are required where  $|\mathcal{V}^i(C)|$  represents the cardinality of  $\mathcal{V}^i(C)$ . Moreover, the transition probability for each received symbol should be saved, which requires  $Nq$  memory cells. Also  $2 \sum_{i=0}^{N-1} |\mathcal{E}^i(C)|$  multiplications and  $2 \sum_{i=0}^{N-1} (|\mathcal{E}^i(C)| - |\mathcal{V}^{i+1}(C)|)$  additions for calculation of  $\{\alpha_i : 0 \leq i \leq N\}$  and  $\{\beta_i : 0 \leq i \leq N\}$  are required. Finally, calculation of (2.4) requires  $2 \sum_{i=0}^{N-1} |\mathcal{E}^i(C)|$  multiplications and  $\sum_{i=0}^{N-1} (|\mathcal{E}^i(C)| - |\mathcal{V}^{i+1}(C)|)$  additions.

The minimal-storage APP decoding algorithm for a non-binary code defined over  $\mathbb{F}_q$  is presented in (2.15) and the  $L_{i,\mu}$  based on this algorithm is calculated by (2.18). As it was mentioned, for computing the APP value of each symbol using (2.15), only  $K$  memory cells are required. The division operation is considered as a multiplication operation and the logarithmic values are read from a lookup table. Therefore,  $2(q^{K-1} - 1)$  additions and  $2(N - 2)q^{K-1} + 1$  multiplications for computing

$$\sum_{v \in C} \left[ \left( \prod_{j \neq i}^{N-1} e^{L_{v_j}(v_j; r_j)(1 - \delta_{0,v_j})} \right) \delta_{0, (e_i \odot v) \oplus -\mu} \right] / \sum_{v \in C} \left[ \left( \prod_{j \neq i}^{N-1} e^{L_{v_j}(v_j; r_j)(1 - \delta_{0,v_j})} \right) \delta_{0, e_i \odot v} \right]$$

is required. A comparison between the complexity of the minimal-storage APP decoding algorithm and the BCJR algorithm for a code defined over  $\mathbb{F}_q$  is given in Table 2.1, where  $\sum_{i=0}^N |\mathcal{V}^i(C)| = V$  and  $\sum_{i=0}^N |\mathcal{E}^i(C)| = \mathcal{E}$ .

Although  $V$  and  $\mathcal{E}$  cannot be represented as a closed formula, they are however dominated by  $\mathcal{O}(q^K)$ . As it is depicted in Table 2.1, the minimal-APP decoding algorithm requires significantly less memory compared with the BCJR algorithm but the reduction in the memory requirement is at the cost of increasing the computational complexity.

Table 2.1  
Complexity comparison between minimal-storage APP decoding algorithm and the  
BCJR algorithm for a code defined over  $\mathbb{F}_q$

	BCJR Algorithm	Minimal-APP Algorithm
Memory Cells	$2V + qN - 1$	$K$
Additions	$6(\mathcal{E} - V + 1)$	$2q^{K-1} - 1$
Multiplication	$8\mathcal{E} + 1$	$2(N - 2)q^{K-1} + 1$

The minimal-storage APP decoding algorithms for low-rate and high-rate codes defined over  $\mathbb{Z}_q$  are presented by (2.18) and (2.30) respectively. It can be seen that  $(q - 1)(N - 1)q^{N-K}$  complex additions and  $[q(N - 1) + (N - 2) + 1]q^{N-K}$  complex multiplications for computing  $\sum_{v' \in C'} \left[ \left( e^{j\frac{2\pi}{q}\mu \otimes v'_i} \right) \left( \prod_{m=0}^{N-1} \sum_{n=0}^{q-1} \left[ e^{L_n(v_m; r_m)} e^{j\frac{2\pi}{q}n \otimes v'_m} \right] \right) \right]$  and  $(q - 1)(N - 1)q^{N-K}$  complex additions and  $[q(N - 1) + (N - 2)]q^{N-K}$  complex multiplications for computing  $\sum_{v' \in C'} \left[ \prod_{m=0}^{N-1} \sum_{n=0}^{q-1} \left[ e^{L_n(v_m; r_m)} e^{j\frac{2\pi}{q}n \otimes v'_m} \right] \right]$  is required. The number of real additions and multiplications required for computation of  $L_{i,\mu}$  in the minimal-storage APP decoding algorithms for codes defined over  $\mathbb{Z}_q$  is given in Table 2.2. These numbers are calculated by considering that each complex addition is equivalent to two real additions and each complex multiplication is equivalent to four real multiplications and two real additions. The operations over  $\mathbb{Z}_q$  are neglected and a lookup table is used for reading the logarithmic values. As it is observed, the minimum amount of memory requirement for calculating the symbol's APP values for a high-rate code is  $N-K$  memory cells.

Table 2.2  
Complexity comparison between the minimal-storage APP decoding algorithms for codes  
defined over  $\mathbb{Z}_q$  with different code rates

	Low-Rate Codes	High-rate Codes
Memory Cell	$K$	$N-K$
Addition	$2q^{K-1} - 1$	$8(N - 1)q^{N-K+1} - 2q^{N-K} + 1$
Multiplication	$2(N - 2)q^{K-1} + 1$	$8(N - 1)q^{N-K+1} + 4(2N - 3)q^{N-K} + 1$



For computation of  $L_{i,\mu}$  based on pre-calculated DFT vectors,  $N-1$  DFT vectors need to be calculated and stored prior to being used in (2.33). Each DFT vector has  $q$  arrays and since the arrays are complex, two memory cells for storing each array are required. Therefore, compared with the minimal-storage algorithm for high-rate codes,  $2(N-1)q$  more memory cells for storing the DFT vectors are required. This extra amount of memory is endurable, considering that by storing the DFT vectors, the computational complexity for calculating the APP values of a high-rate code becomes less. The complexity of the proposed algorithm is given in Table 2.3. For calculating each DFT vector,  $q(q-1)$  complex addition and  $q^2$  complex multiplication is required.

Table 2.3  
Complexity of the APP decoding algorithm implemented over the code's dual space by using pre-calculated DFT vectors

Memory Cell	$2(N-1)q + (N-K)$
Addition	$2(N+2)q^{N-K} + 4(N-1)q^2 - 2(N-1)q - 3$
Multiplication	$4Nq^{N-K} + 4(N-1)q^2 + 1$

As it is observed, the computational complexity of the minimal-storage APP decoding algorithm for high-rate codes is dominated by  $\mathcal{O}(q^{N-K+1})$  operations, while by using pre-calculated DFT vectors this complexity is reduced to the order of  $\mathcal{O}(q^{N-K})$ . Therefore, by using pre-calculated DFT vectors with an endurable increase in the memory requirement compared with the minimal-storage APP decoding algorithm for high-rate codes, the order of the computational complexity is reduced by factor  $q$ .

#### 2.4.5. Simulation results

In this section, the simulation results for the performance of a code under different APP decoding algorithms are presented. The performance of different decoders can be evaluated by comparing the performance of the same code that is separately decoded by each algorithm. However, the APP decoding algorithm based on the code's dual space can be

implemented only for the codes that are defined over  $\mathbb{Z}_q$ . Therefore, we present the simulation results for  $q = 2$  where  $\mathbb{F}_2 = \mathbb{Z}_2$ . In the following chapters, the simulation results for non-binary codes are also presented.

The MAP decoding algorithm for the (7,4) Hamming code that has the parity-check matrix given by (2.1) is simulated. The codewords are modulated by BPSK modulation scheme and are transmitted over an AWGN channel with double-sided noise power spectral density of  $\frac{N_0}{2}$ . The APP decoding algorithm based on direct implementation is realized by (2.19) and the APP decoding algorithm based on the dual code is realized by (2.31).

The code performance is also compared with the soft decoding theoretical upper bound. In [85], the theoretical upper bound for the bit-error probability of an  $(N, K)$  binary block code with the minimum distance of  $d_{min}$  is given as (2.34). It is assumed that the received sequences are coherently detected and decoded by a maximum-likelihood decoding algorithm.

$$p_e \leq \sum_{h=d_{min}}^N \sum_{\omega=1}^K \frac{\omega}{K} A_{\omega,h} Q \left( \sqrt{2R_c h E_b / N_0} \right) \quad (2.34)$$

where  $R_c$  is the code rate and  $A_{\omega,h}$  is the number of codewords that have output weight  $h$  associated with an input sequence of weight  $\omega$ . Furthermore, the  $Q$ -function is defined as

$Q(x) = \frac{1}{\sqrt{2\pi}} \int_x^\infty e^{-\frac{y^2}{2}} dy$ . Based on (2.34) the theoretical upper bound for the (7,4) Hamming code, under the given conditions is equal to

$$p_e \leq \sum_{h=3}^7 \sum_{\omega=1}^4 \frac{\omega}{4} A_{\omega,h} Q \left( \sqrt{1.14 h E_b / N_0} \right) \quad (2.35)$$

The performance of the (7,4) Hamming code decoded by different decoding algorithms is depicted in Figure 2.2. It is observed that the performance of the APP decoder

based on direct implementation matches the performance of the APP decoder based on the dual code.

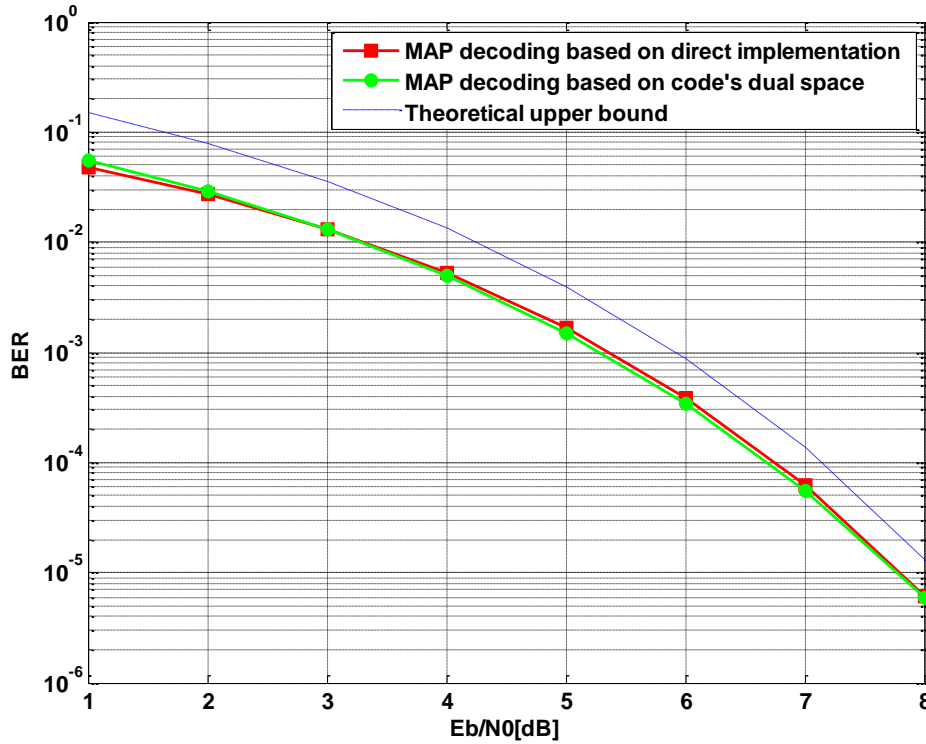


Figure 2.2: Simulated BER performance of the (7,4) Hamming code over AWGN channel using BPSK

In Table 2.4, the computational complexity and the memory requirement for decoding the (7,4) Hamming code using the minimal-storage APP decoding algorithm is compared with that of the BCJR algorithm. Due to the rate of the code, the minimal-storage APP decoding algorithm is implemented based on the code's dual space. It can be observed from Figure 2.1 that for the (7,4) Hamming code,  $V = \sum_{i=0}^7 |\mathcal{V}^i(C)| = 30$  and  $\sum_{i=0}^6 |\mathcal{E}^i(C)| = 44$ .

Table 2.4  
Complexity comparison between minimal-storage APP decoding algorithm and the BCJR algorithm for decoding the (7,4) Hamming code

	BCJR Algorithm	Minimal-APP Algorithm
Memory Cells	73	3
Additions	90	399
Multiplication	353	387

## 2.5. Conclusion

In this chapter, the optimal APP decoding algorithm for the non-binary linear block codes was discussed. It was shown that the computational complexity and the amount of memory requirement for the BCJR algorithm, which is the standard way for calculating the symbol's APP values, are often prohibitive in many practical applications. Therefore, we proposed the minimal-storage APP decoding algorithms for different linear block codes. We showed that the minimum amount of memory requirement for calculating the symbol's APP values of any linear block code can be achieved by straightforward implementation of the trellis, while for a high-rate code defined over  $\mathbb{Z}_q$ , this can be achieved by the APP decoding algorithm implemented based on the code's dual space. This means that for any linear block code defined over  $\mathbb{F}_q$ , regardless of its rate, the minimum amount of memory requirement for calculating the symbol's APP values can be achieved by straightforward implementation of the trellis. However, for linear block codes defined over  $\mathbb{Z}_q$ , the minimal-storage APP decoding algorithm depends on the code rate, which can be achieved by straightforward implementation of the trellis for low-rate codes, or by the APP decoding algorithm implemented based on the code's dual space for high-rate codes. Furthermore, we modified the minimal-storage APP decoding algorithm for high-rate codes defined over  $\mathbb{Z}_q$  to reduce the computational complexity of the algorithm. Based on this modification, the DFT vectors were employed to limit the repetitive calculations. We showed that this modified algorithm is valid only for the finite fields with modulo- $q$  addition and therefore, it cannot be used for

the codes that are defined over finite field of order  $2^p$  ( $p$  is any positive integer). Moreover, we showed that by using FFT algorithm for calculating the DFT vectors, this modified algorithm can be implemented more efficiently for the codes that are defined over  $\mathbb{Z}_{2^p}$ . The numerical results, as well as the complexity comparison for the proposed algorithms, were given.

## **CHAPTER 3**

### **NON-BINARY SINGLE PARITY-CHECK CODES**

#### **3.1. Introduction**

Turbo codes [4] and low-density parity-check (LDPC) codes [10,12] are the two major classes of capacity approaching codes that have attracted considerable attention. Both of these codes are constructed from concatenation of simple constituent codes and are decoded by iterative decoding algorithms that repeatedly exchange the soft information between their component codes [86]. Turbo codes are typically constructed from parallel concatenation of simple convolutional or simple block codes, while LDPC codes can be considered as multiple parallel concatenations of single parity-check (SPC) codes [87].

Besides LDPC codes, binary SPC codes are also used as constituent codes in turbo-like structures and good-performance binary codes, with relatively low-complexity decoding algorithms, are designed based on their concatenation [66-68,88,89]. It is shown that high-rate binary SPC product codes can perform as well as the same rate binary LDPC codes, while having less encoding and decoding complexity [88] and serially concatenated binary SPC codes can outperform 16-state binary turbo codes [89]. Furthermore, the simulation results in [68] shows that a binary SPC product code with length of 97751 and rate of 0.985 performs only 0.44 dB away from the Shannon-limit at a bit-error rate (BER) of  $10^{-5}$ .

SPC codes are high-rate codes and are suitable to be used as a constituent code for constructing high-rate compound codes. However, they can also be employed as constituent code for constructing lower-rate compound codes. The multidimensional product codes constructed from binary SPC codes are studied in [66] and it is shown that by increasing the number of dimensions, the performance of the resulting compound code improves, yet the rate of the compound code reduces. Furthermore, it is shown that by interleaving information between the encoding of each dimension, the performance of the resulting compound code improves [67].

Since the performance of a moderate-length LDPC code improves over higher order fields [15], a good-performance code with short or medium block size may be designed based on concatenation of non-binary SPC codes. Moreover, non-binary codes are attractive for high data-rate digital communication, where high-order modulations are widely being used and it is more convenient to use non-binary codes with appropriate alphabet size to match the constellation. Furthermore, by considering non-binary alphabets, an extra degree of freedom is added to the design parameters of a code.

In this chapter, the construction and decoding of SPC codes over non-binary finite rings are studied. The concept of finite ring is more general and it includes finite field. Since SPC codes are mainly used as constituent code in concatenated structures, soft-input soft-output (SISO) decoding algorithms for decoding SPC codes are required. In this chapter, two optimum *a posteriori* probability (APP) decoding algorithms for decoding non-binary SPC codes are presented and the computational complexity and the memory requirement for each algorithm are discussed. We show that based on our proposed decoding algorithm the decoding complexity for non-binary SPC codes defined over finite ring of integer modulo- $q$ ,  $\mathbb{Z}_q$ , is considerably small and therefore, non-binary SPC code defined over  $\mathbb{Z}_q$  is a good choice to be used as constituent code in concatenated structures.

To the extent of our knowledge, non-binary SPC codes have not been studied in the literature, however as it was mentioned before, LDPC codes can be considered as multiple parallel concatenations of SPC codes and therefore, we can say that non-binary SPC codes are not directly but rather indirectly studied through non-binary LDPC codes [15-16, 51-54]. In this chapter, we specifically study the structure and decoding of non-binary SPC codes. We show that the fast Fourier transform belief propagation (FFT-BP) decoding algorithm

[51] and subsequently, other algorithms that are derived from FFT-BP such as Log-FFT-BP [52,53] and extended min-sum (EMS) [54] are valid only for the LDPC codes defined over  $\mathbb{Z}_q$ . Furthermore, based on the structure of non-binary SPC codes, the performance improvement of non-binary LDPC codes over higher order fields, which was left as an open problem by [15], is explained.

This chapter is organized as follows. The encoding of non-binary SPC codes is discussed in Section 3.2. In Section 3.3, two optimum APP decoding algorithms for decoding non-binary SPC codes are presented and the complexity of each algorithm is discussed. The simulation results are given in Section 3.4. The final section concludes this chapter.

### 3.2. Encoding of non-binary SPC codes

Non-binary SPC codes are not directly but rather indirectly studied through non-binary LDPC codes [15-16, 51-54]. However, to the extent of our knowledge, they are not specifically studied. Non-binary codes are mostly defined over a finite field of order  $q$ ,  $\mathbb{F}_q$ , but can also be defined over finite rings or finite groups [80-81]. In this chapter, we consider non-binary SPC codes defined over  $\mathbb{Z}_q$ , as well as the non-binary SPC codes that are defined over  $\mathbb{F}_q$ .

A ring of integer modulo- $q$  is a commutative finite ring, where set  $\mathbb{Z}_q = \{0, 1, \dots, q - 1\}$  under addition modulo- $q$  is a group, but set  $\mathbb{Z}_q \setminus \{0\} = \{1, \dots, q - 1\}$  under multiplication modulo- $q$  is not a group [90]. Thus generally,  $\mathbb{Z}_q$  cannot be considered as a finite field. However, for a prime number  $q$ ,  $\mathbb{Z}_q \setminus \{0\}$  under multiplication modulo- $q$  is also a commutative group and therefore, for a prime number  $q$ ,  $\mathbb{Z}_q$  can be considered as a finite field. The addition and multiplication tables for  $\mathbb{F}_2$ ,  $\mathbb{Z}_2$ ,  $\mathbb{F}_4$  and  $\mathbb{Z}_4$  are given in Figure 3.1. It can be seen that  $\mathbb{F}_2$  tables and  $\mathbb{Z}_2$  tables are the same, while  $\mathbb{F}_4$  tables and  $\mathbb{Z}_4$  tables are different.



$\mathbb{F}_2$  Addition and multiplication tables

$\oplus$	0	1
0	0	1
1	1	0

$\otimes$	0	1
0	0	0
1	0	1

 $\mathbb{Z}_2$  Addition and multiplication tables

$\oplus$	0	1
0	0	1
1	1	0

$\otimes$	0	1
0	0	0
1	0	1

 $\mathbb{F}_4$  Addition and multiplication tables

$\oplus$	0	1	2	3
0	0	1	2	3
1	1	0	3	2
2	2	3	0	1
3	3	2	1	0

$\otimes$	0	1	2	3
0	0	0	0	0
1	0	1	2	3
2	0	2	3	1
3	0	3	1	2

 $\mathbb{Z}_4$  Addition and multiplication tables

$\oplus$	0	1	2	3
0	0	1	2	3
1	1	2	3	0
2	2	3	0	1
3	3	0	1	2

$\otimes$	0	1	2	3
0	0	0	0	0
1	0	1	2	3
2	0	2	0	2
3	0	3	2	1

Figure 3.1: Addition and multiplication of  $\mathbb{F}_2$  versus  $\mathbb{Z}_2$  and  $\mathbb{F}_4$  versus  $\mathbb{Z}_4$ 

For  $K$  message symbols  $v_i$ , where  $\{v_i \in \{0, 1, 2, \dots, q-1\} : 0 \leq i \leq K-1\}$ , a non-binary  $(K+1, K)$  SPC code is defined as  $(K+1)$ -tuples like  $(v_0, v_1, v_2, \dots, v_K)$  such that

$$v_0 \oplus v_1 \oplus v_2 \oplus \dots \oplus v_{K-1} \oplus v_K = 0 \quad (3.1)$$

where depending on the definition of the code,  $\oplus$  is the addition operation defined over  $\mathbb{Z}_q$  or  $\mathbb{F}_q$ . Both  $(\mathbb{Z}_q, \oplus)$  and  $(\mathbb{F}_q, \oplus)$  are commutative group, therefore they are close under addition operation and subsequently, we have  $v_K \in \{0, 1, 2, \dots, q-1\}$ .

### 3.3. Decoding of non-binary SPC codes

Maximum-likelihood decoding of powerful and mostly long concatenated codes is often computationally infeasible. Therefore, concatenated codes are usually decoded based

on iterative decoding of their constituent codes. Since SPC codes are typically used as constituent code in concatenated structures [66-68,88,89], they need to be decoded by SISO algorithms. In this section, two different optimum APP decoding algorithms for decoding non-binary SPC codes are presented. In the first algorithm, the symbols' APP values are calculated by straightforward implementation of the code's trellis, which can be used for any SPC code, while in the second algorithm, the APP values are calculated over the code's dual space and as it was explained in Chapter 2, it can be used only for the SPC codes defined over  $\mathbb{Z}_q$ .

Let  $C$  be a non-binary SPC code and  $\mathbf{v} = (v_0, v_1, \dots, v_K)$  be any codeword that belongs to  $C$ . The codewords are transmitted over a discrete-time, memoryless, noisy channel and the received sequence  $\mathbf{r} = (r_0, r_1, \dots, r_K)$  that corresponds to a transmitted codeword is available at the decoder. From Chapter 2, we know that the vector of log-likelihood ratios (LLR)  $L_i = \{L_{i,\mu} : 0 \leq \mu \leq q-1\}$  for the information symbols  $\{v_i : 0 \leq i \leq K-1\}$  of a non-binary code can be achieved from  $L_{i,\mu}$ , which is defined as

$$L_{i,\mu} \stackrel{\text{def}}{=} \ln \left[ \frac{\sum_{\substack{\mathbf{v} \in C \\ v_i = \mu}} Pr(\mathbf{v}|\mathbf{r})}{\sum_{\substack{\mathbf{v} \in C \\ v_i = 0}} Pr(\mathbf{v}|\mathbf{r})} \right] \quad (3.2)$$

### 3.3.1. APP decoding algorithm based on straightforward implementation

The codewords are transmitted over a memoryless channel, therefore, from Bayes' rule we have

$$Pr(\mathbf{v}|\mathbf{r}) = \frac{1}{Pr(\mathbf{r})} \prod_{m=0}^K Pr(r_m; v_m) \quad (3.3)$$

and the LLR value for the symbol  $v_i$  is calculated by

$$L_{i,\mu} = L_\mu(v_i; r_i) + \ln \left( \frac{\sum_{\substack{v \in C \\ v_i = \mu}} \prod_{\substack{m=0 \\ m \neq i}}^K e^{L_{v_m}(v_m; r_m)}}{\sum_{\substack{v \in C \\ v_i = 0}} \prod_{\substack{m=0 \\ m \neq i}}^K e^{L_{v_m}(v_m; r_m)}} \right) \quad (3.4)$$

where,  $L_\mu(v_i; r_i) \stackrel{\text{def}}{=} \ln[Pr(v_i = \mu; r_i)/Pr(v_i = 0; r_i)]$ .

Equation (3.4) is the general formula to find the APP value for the symbol  $v_i$  of an SPC code when the codewords are transmitted over a memoryless channel. It can be used for SPC codes defined over  $\mathbb{F}_q$  as well as SPC codes that are defined over  $\mathbb{Z}_q$ . Compared with the BCJR algorithm [71], which is the standard way of calculating the symbol's APP values and is realized by (2.4), the amount of memory requirement by (3.4) is significantly reduced. Based on (2.4), for calculating the symbol's APP values, all  $\{\alpha_i : 0 \leq i \leq N\}$  and  $\{\beta_i : 0 \leq i \leq N\}$  need to be calculated and stored in memory prior to be used in (2.4), which as mentioned before, requires a large amount of memory and is often prohibitive in many practical applications. However, for calculating the symbol's APP values based on (3.4), no prior calculations are required and only  $K$  memory cells for storing the information symbols are sufficient.

### 3.3.2. APP decoding algorithm based on Fourier transform

As it was mentioned in the previous chapter, for a high-rate linear block code defined over  $\mathbb{Z}_q$ , the minimal-storage APP decoding algorithm can be achieved by using the APP decoding algorithm that is implemented based on the code's dual space. Therefore, for the SPC codes defined over  $\mathbb{Z}_q$ , which are high-rate codes, the minimal-storage APP decoding algorithm is implemented based on the codes' dual space. Furthermore, we showed that by using the concept of Fourier transform, the computational complexity of the dual implementation of the APP decoding algorithm is reduced. The idea of using Fourier

transform for decoding codes can also be found in decoding of the LDPC codes [51-54]. However, in LDPC codes, the Fourier transform is used to convert the convolution operation into multiplication operation and thus simplify the decoding algorithm while in this section, we use the concept of Fourier transform to avoid the repetitive calculation and reduce the computational complexity of the decoding algorithm. Let  $C'$  be the dual code of the code  $C$  and  $\mathbf{v}' = (v'_0, v'_1, \dots, v'_K)$  be any codeword that belongs to  $C'$ . Therefore, for any  $(K+1)$ -tuple  $\mathbf{u} = (u_0, u_1, \dots, u_K)$  defined over  $\mathbb{Z}_q$  we can write

$$\sum_{\mathbf{v}' \in C'} e^{j\frac{2\pi}{q}\mathbf{u} \odot \mathbf{v}'} = \sum_{\mathbf{v}' \in C'} e^{j\frac{2\pi}{q}((u_0 \otimes v'_0) \oplus (u_1 \otimes v'_1) \oplus \dots \oplus (u_K \otimes v'_K))} \quad (3.5)$$

where  $\odot$  denotes the inner product between the vectors defined over  $\mathbb{Z}_q$  and the addition and multiplication over  $\mathbb{Z}_q$  are denoted by  $\oplus$  and  $\otimes$  respectively. Since  $\oplus$  is defined as modulo- $q$  addition, we have

$$\sum_{\mathbf{v}' \in C'} e^{j\frac{2\pi}{q}\mathbf{u} \odot \mathbf{v}'} = \sum_{\mathbf{v}' \in C'} \prod_{m=0}^K e^{j\frac{2\pi}{q}(u_m \otimes v'_m)} \quad (3.6)$$

and since  $C$  is a non-binary  $(K+1, K)$  SPC code, the dual code for that, which is  $C'$ , is a repetition code of length  $K+1$  that is defined over  $\mathbb{Z}_q$ . The inner product between any codeword in  $C$  and any codeword in  $C'$  based on orthogonality is zero. Therefore,

$$\sum_{\mathbf{v}' \in C'} e^{j\frac{2\pi}{q}\mathbf{u} \odot \mathbf{v}'} = \sum_{s=0}^{q-1} \prod_{m=0}^K e^{j\frac{2\pi}{q}(u_m \otimes s)} = \begin{cases} q^{N-K}; & \mathbf{u} \in C \\ 0 & ; \text{ otherwise} \end{cases} \quad (3.7)$$

Let  $\mathbf{V}^N$  be the vector space of all  $(K+1)$ -tuples defined over  $\mathbb{Z}_q$ . Since the codewords are transmitted over a memoryless channel, from Bayes' rule and by using (3.7) we have

$$\begin{aligned} \sum_{\substack{\mathbf{v} \in \mathcal{C} \\ v_i = \mu}} Pr(\mathbf{v}|\mathbf{r}) &= \frac{1}{q^{N-K} Pr(\mathbf{r})} \sum_{\substack{\mathbf{u} \in \mathbf{V}^N \\ u_i = \mu}} \left[ Pr(\mathbf{r}; \mathbf{u}) \sum_{v' \in \mathcal{C}'} e^{j \frac{2\pi}{q} \mathbf{u} \odot \mathbf{v}'} \right] \\ &= \frac{1}{q^{N-K} Pr(\mathbf{r})} \sum_{s=0}^{q-1} \left[ Pr(r_i; \mu) e^{j \frac{2\pi}{q} (\mu \otimes s)} \prod_{\substack{m=0 \\ m \neq i}}^K \sum_{n=0}^{q-1} \left( Pr(r_m; n) e^{j \frac{2\pi}{q} (n \otimes s)} \right) \right] \end{aligned} \quad (3.8)$$

which by substituting (3.8) in (3.2),  $L_{i,\mu}$  can be calculated as

$$L_{i,\mu} = L_\mu(v_i; r_i) + \ln \left( \frac{\sum_{s=0}^{q-1} \left[ \left( e^{j \frac{2\pi}{q} (\mu \otimes s)} \right) \left( \prod_{m=0, m \neq i}^K \sum_{n=0}^{q-1} \left[ Pr(r_m; n) e^{j \frac{2\pi}{q} (n \otimes s)} \right] \right) \right]}{\sum_{s=0}^{q-1} \left[ \prod_{m=0}^K \sum_{n=0}^{q-1} \left[ Pr(r_m; n) e^{j \frac{2\pi}{q} (n \otimes s)} \right] \right]} \right) \quad (3.9)$$

The discrete Fourier transform (DFT) and the inverse DFT over  $\mathbb{Z}_q$  is defined as

$$F(s) = \mathcal{F}_s\{f(n)\} \stackrel{\text{def}}{=} \sum_{n=0}^{q-1} f(n) e^{j \frac{2\pi}{q} n \otimes s} \quad (3.10)$$

$$f(n) = \mathcal{F}_n^{-1}\{F(s)\} \stackrel{\text{def}}{=} \frac{1}{q} \sum_{s=0}^{q-1} F(s) e^{j \frac{2\pi}{q} (-n) \otimes s} \quad (3.11)$$

where  $-n$  is the additive inverse of  $n$  over  $\mathbb{Z}_q$ . Therefore, by using the concept of DFT we have

$$\begin{aligned}
 L_{i,\mu} &= L_\mu(v_i; r_i) + \ln \left( \frac{\mathcal{F}_{n=-\mu}^{-1} \left\{ \prod_{\substack{m=0 \\ m \neq i}}^K \mathcal{F}_s \{ Pr(r_m; n) \} \right\}}{\mathcal{F}_{n=0}^{-1} \left\{ \prod_{\substack{m=0 \\ m \neq i}}^K \mathcal{F}_s \{ Pr(r_m; n) \} \right\}} \right) \\
 &= L_\mu(v_i; r_i) + \ln \left( \frac{\mathcal{F}_{n=-\mu}^{-1} \left\{ \prod_{\substack{m=0 \\ m \neq i}}^K \mathcal{F}_s \{ e^{L_n(v_m; r_m)} \} \right\}}{\mathcal{F}_{n=0}^{-1} \left\{ \prod_{\substack{m=0 \\ m \neq i}}^K \mathcal{F}_s \{ e^{L_n(v_m; r_m)} \} \right\}} \right) \tag{3.12}
 \end{aligned}$$

$\prod(\cdot)$  in (3.12) represents the term-by-term product of the vectors.

As it is observed, for each received symbol  $r_m$ , a  $q$ -dimensional vector  $\{e^{L_n(v_m; r_m)} : 0 \leq n \leq q-1\}$  is considered, where the Fourier transforms of these vectors have to be calculated and stored in memory prior to be used in (3.12). However, for all information symbols  $v_i$ ,  $0 \leq i \leq K-1$ , and throughout the computation of  $L_i = \{L_{i,\mu}\}$ , only the Fourier transform for each vector  $\{e^{L_n(v_m; r_m)}\}$  has to be calculated. Therefore, the repetitive calculations can be avoided by storing the Fourier transform vectors and subsequently, the computational complexity can be reduced.

Although implementing (3.12) requires more memory than (3.4), compared with the BCJR algorithm [71], which is the standard way of calculating the symbol's APP values and is realized by (2.4), the amount of memory that is required by (3.12) is significantly reduced. Based on (2.4), for calculating the symbol's APP values, all  $\{\alpha_i : 0 \leq i \leq N\}$  and  $\{\beta_i : 0 \leq i \leq N\}$  have to be calculated and stored in memory prior to be used in (2.4). This requires a large amount of memory and it is often prohibitive in many practical applications. However, for calculating the symbol's APP values based on (3.12), only  $K$  DFT vectors need to be calculated and stored prior to be used in (3.12). Each DFT vector has  $q$

arrays and since the arrays are complex, two memory cells for storing each array are required. Therefore, the total memory requirement of (3.12) is  $2Kq$  memory cells.

Although the amount of memory requirement of (3.12) is more than (3.4), it has less complexity compared with (3.4). This arises from the amount of calculations that are required by each algorithm for computation of  $L_{i,\mu}$ . As it is seen from (3.4), part of calculations that are required for computation of  $L_{i,\mu}$  have to be repeated for all codewords and therefore, the number of calculations are in the order of  $\mathcal{O}(q^{K-1})$ . However, based on (3.12) and by using the stored DFT vectors, the computation complexity is in the order of  $\mathcal{O}(qK)$ .

Furthermore, despite the differences between the approaches that we consider for deriving (3.12) and the one that is considered in [52] for the row-step of the FFT-BP algorithm, they both result in the same formula, which is (3.12). This can be explained by the structure of LDPC codes, which are constructed as parallel concatenation of SPC codes. However, it can be concluded that the FFT-BP decoding algorithm and subsequently, other algorithms that are derived from FFT-BP such as Log-FFT-BP [52,53] and EMS [54] are valid only for the LDPC codes defined over  $\mathbb{Z}_q$  and implementing them for the non-binary LDPC codes defined over  $\mathbb{F}_q$  is not mathematically correct.

### 3.3.3. Complexity comparison

In this section, we compare the complexity of the two proposed algorithms. The complexity of an algorithm can be measured by the amount of memory requirement and the number of real operations required for computation of  $L_{i,\mu}$ . The complexity comparison between the two algorithms is given in Table 3.1. These numbers are calculated by considering that each complex addition is equivalent to two real additions and each complex multiplication is equivalent to four real multiplications and two real additions. The operations over  $\mathbb{Z}_q$  are neglected and a lookup table is used for reading the logarithmic values. Moreover, the division operation is considered as a multiplication operation.

Table 3.1  
Complexity comparison between the two proposed algorithms

	Direct Implementation	DFT Based
Memory Cell	$K$	$2Kq$
Addition	$2q^{K-1} - 1$	$4(K + 1)q^2 - 4q + 1$
Multiplication	$2(K - 1)q^{K-1} + 1$	$4(K + 1)q^2 + 4(K - 1)q + 1$

For computation of  $L_{i,\mu}$  based on DFT based algorithm,  $K$  DFT vectors and one inverse DFT vector have to be calculated. For reducing the computational complexity, the DFT vectors are required to be stored in memory prior to being used in (3.12). Each DFT vector has  $q$  complex arrays and storing each array, requires two memory cells. For calculating each DFT or inverse DFT vector,  $q(q - 1)$  complex additions and  $q^2$  complex multiplications are required. Moreover, for the SPC codes defined over  $\mathbb{Z}_{2^p}$  ( $p$  is any positive integer), the fast Fourier transform (FFT) algorithm can be used for calculating the Fourier and inverse Fourier transforms. The computational complexity for calculating the Fourier transform of  $\{e^{L_n(v_m; r_m)}\}$  is  $\mathcal{O}(q^2)$  while by using an FFT algorithm, the same results are obtained by only  $\mathcal{O}(q \log_2(q))$  operations. Therefore, by using FFT instead of DFT, the overall computational complexity is reduced further.

As it is observed, the computational complexity of the APP decoding algorithm based on Fourier transform is dominated by  $\mathcal{O}(Kq^2)$  operations and it requires  $2Kq$  memory cells, which are reasonably small numbers. Therefore, non-binary SPC codes defined over  $\mathbb{Z}_q$  are good options to be used as constituent codes in concatenated structures.

### 3.4. Simulation results

In this section, the simulation results for non-binary SPC codes are presented. Similar to [15, 16], it is assumed that all codewords are modulated by binary phase-shift keying (BPSK) modulation scheme and are transmitted over a binary input additive white Gaussian



noise (BI-AWGN) channel with double-sided noise power spectral density of  $\frac{N_0}{2}$ . Therefore, the symbols need to be mapped to bipolar sequences as

$$v_i \rightarrow \bar{\mathbf{c}}_i = (c_i^1, c_i^2, \dots, c_i^p) \rightarrow \bar{\mathbf{v}}_i = (v_i^1, v_i^2, \dots, v_i^p) \quad (3.13)$$

where  $\bar{\mathbf{c}}_i$  is the corresponding binary sequence for the symbol  $v_i$  such that  $\{c_i^m \in \{0,1\} : 1 \leq m \leq p\}$  and  $\bar{\mathbf{v}}_i$  is the corresponding bipolar sequence for the symbol  $v_i$  such that  $\{v_i^m = 2c_i^m - 1 : 1 \leq m \leq p\}$  and the length of both sequences are equal to  $p = \log_2 q$ . For example:

$$v_i = 9 \rightarrow \bar{\mathbf{c}}_i = (1,0,0,1) \rightarrow \bar{\mathbf{v}}_i = (1,-1,-1,1)$$

We assume that at the receiver side, the sequence  $\mathbf{r} = (\bar{\mathbf{r}}_0, \bar{\mathbf{r}}_2, \dots, \bar{\mathbf{r}}_{N-1})$  is received. The noise vector is represented by  $\boldsymbol{\eta} = (\bar{\boldsymbol{\eta}}_0, \bar{\boldsymbol{\eta}}_2, \dots, \bar{\boldsymbol{\eta}}_{N-1})$ , where  $\bar{\boldsymbol{\eta}}_i = (\eta_i^1, \eta_i^2, \dots, \eta_i^p)$  is the corresponding noise sequence for the bipolar sequence of the symbol  $v_i$ . Therefore,  $\mathbf{r} = \mathbf{v} + \boldsymbol{\eta}$  and  $\bar{\mathbf{r}}_i = \bar{\mathbf{v}}_i + \bar{\boldsymbol{\eta}}_i$ . Since the codewords are transmitted over an AWGN channel, the noise is independent from the transmitted symbol and  $\{\eta_i^m : 0 \leq i \leq N-1, 1 \leq m \leq p\}$  are zero-mean, identically, independently distributed (i.i.d.) Gaussian random variables with variance  $\sigma^2 = \frac{N_0}{2}$ . Therefore, the LLR for the joint random variable  $(v_i; r_i)$  is calculated as

$$L_\mu(v_i; r_i) = L_\mu(v_i) + \ln \left( \frac{\exp\left(-\frac{\|\bar{\mathbf{r}}_i - \bar{\boldsymbol{\mu}}\|^2}{2\sigma^2}\right)}{\exp\left(-\frac{\|\bar{\mathbf{r}}_i - \bar{\mathbf{0}}\|^2}{2\sigma^2}\right)} \right) \quad (3.14)$$

where  $\bar{\boldsymbol{\mu}}$  and  $\bar{\mathbf{0}}$  are the corresponding bipolar sequences for  $\mu$  and 0 respectively and  $\|\mathbf{X}\|$  denotes the norm of the vector  $\mathbf{X}$ .

$$\|\bar{\mathbf{r}}_i - \bar{\boldsymbol{\mu}}\|^2 = \|\bar{\mathbf{r}}_i\|^2 + \|\bar{\boldsymbol{\mu}}\|^2 - 2(\bar{\mathbf{r}}_i \odot \bar{\boldsymbol{\mu}}) \quad (3.15)$$

where  $\odot$  represents the inner product between the two vectors. Since the energy of all symbols are equal,  $\|\bar{\boldsymbol{\mu}}\|^2 = \|\bar{\mathbf{0}}\|^2$  and after cancelling out the common factors we have

$$L_\mu(v_i; r_i) = L_\mu(v_i) + \sum_{j=1}^p \frac{(\mu^j + 1)r_i^j}{\sigma^2} \quad (3.16)$$

where  $\mu^j$  is the  $j^{\text{th}}$  component of  $\bar{\boldsymbol{\mu}}$  and  $r_i^j$  is the  $j^{\text{th}}$  component of  $\bar{\mathbf{r}}_i$ . Moreover, for equiprobable information symbols,  $L_\mu(v_i) = 0$ . Therefore, LLR value that is given by (3.4), provided that the information symbols are equiprobable, can be realized as

$$L_{i,\mu} = \sum_{j=1}^p \frac{(\mu^j + 1)r_i^j}{\sigma^2} + \ln \left( \frac{\sum_{\substack{v \in C \\ v_i = \mu}} \exp \left( \sum_{\substack{m=0 \\ m \neq i}}^K \sum_{j=1}^p \frac{(v_m^j + 1)r_m^j}{\sigma^2} \right)}{\sum_{\substack{v \in C \\ v_i = 0}} \exp \left( \sum_{\substack{m=0 \\ m \neq i}}^K \sum_{j=1}^p \frac{(v_m^j + 1)r_m^j}{\sigma^2} \right)} \right) \quad (3.17)$$

and the LLR value based on Fourier transform that is given in (3.12), provided that the information symbols are equiprobable, can be realized as

$$L_{i,\mu} = \sum_{j=1}^p \frac{(\mu^j + 1)r_i^j}{\sigma^2} + \ln \left( \frac{\mathcal{F}_{n=-\mu}^{-1} \left\{ \prod_{\substack{m=0 \\ m \neq i}}^K \mathcal{F}_s \left\{ \exp \left( \sum_{j=1}^p \frac{(n^j + 1)r_m^j}{\sigma^2} \right) \right\} \right\}}{\mathcal{F}_{n=0}^{-1} \left\{ \prod_{\substack{m=0 \\ m \neq i}}^K \mathcal{F}_s \left\{ \exp \left( \sum_{j=1}^p \frac{(n^j + 1)r_m^j}{\sigma^2} \right) \right\} \right\}} \right) \quad (3.18)$$

In Figure 3.2, the performance of the (5,4) SPC code defined over different field orders is presented and is compared with the asymptotic bound for the (5,4) SPC code, which is calculated in Chapter 5. These codes are decoded with the straightforward implementation of the APP decoding algorithm that is given by (3.17). As it is observed, the performance of the SPC codes defined over finite field is independent of the order of the field, which can be explained by the minimum distance property of the SPC codes that is presented in Chapter 5. However, the simulation result shows a different outcome for the SPC codes defined over  $\mathbb{Z}_q$ .

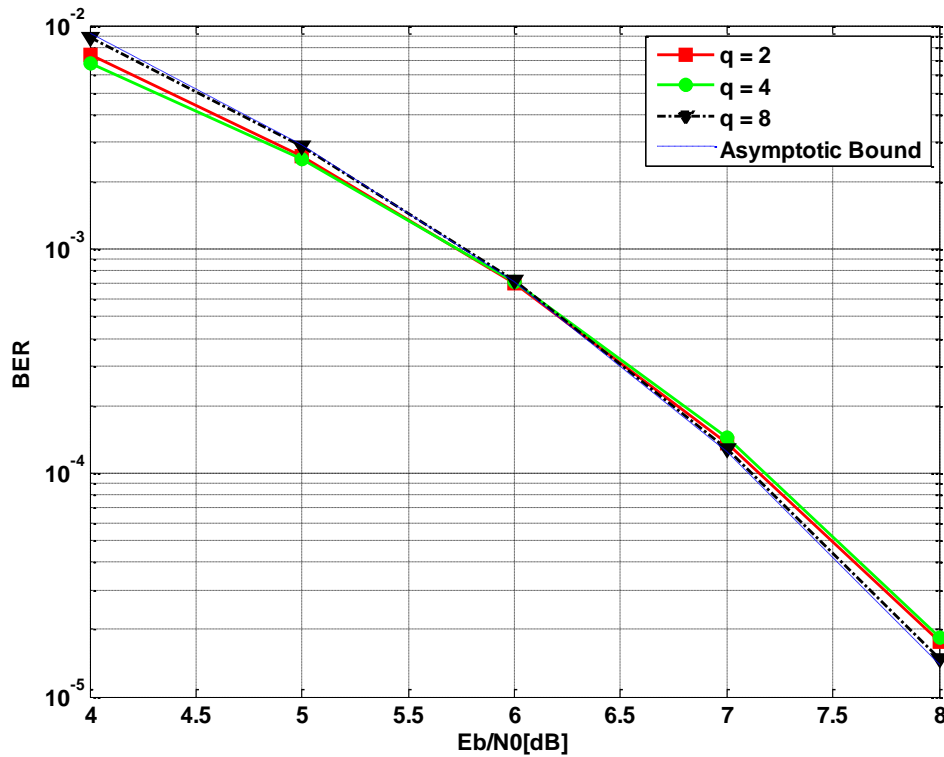


Figure 3.2: Performance of the (5,4) SPC code defined over  $\mathbb{F}_q$

In Figure 3.3, the performance of the (5,4) SPC code defined over different ring of integer modulo- $q$  is presented and the performance is compared with the asymptotic bound. These codes are decoded with the Fourier based APP decoding algorithm, which is given by

(3.18). Since both proposed decoding algorithms are optimum, the performance of an SPC code defined over  $\mathbb{Z}_q$  has to be the same under both decoding algorithms. It can be observed that unlike SPC codes defined over finite fields, the performance of the (5,4) SPC codes defined over ring of integer modulo- $q$  improves with higher order rings. This can be explained by the minimum distance property of the SPC codes that is presented in Chapter 5. Furthermore, since LDPC codes are considered as parallel concatenation of SPC codes, the performance improvement of SPC codes over higher order rings can be considered as an explanation for the performance improvement of non-binary LDPC codes over higher order fields, which was left as an open problem by [15]. However, it is important to note that non-binary LDPC codes actually perform better over higher order rings and not over higher order fields.

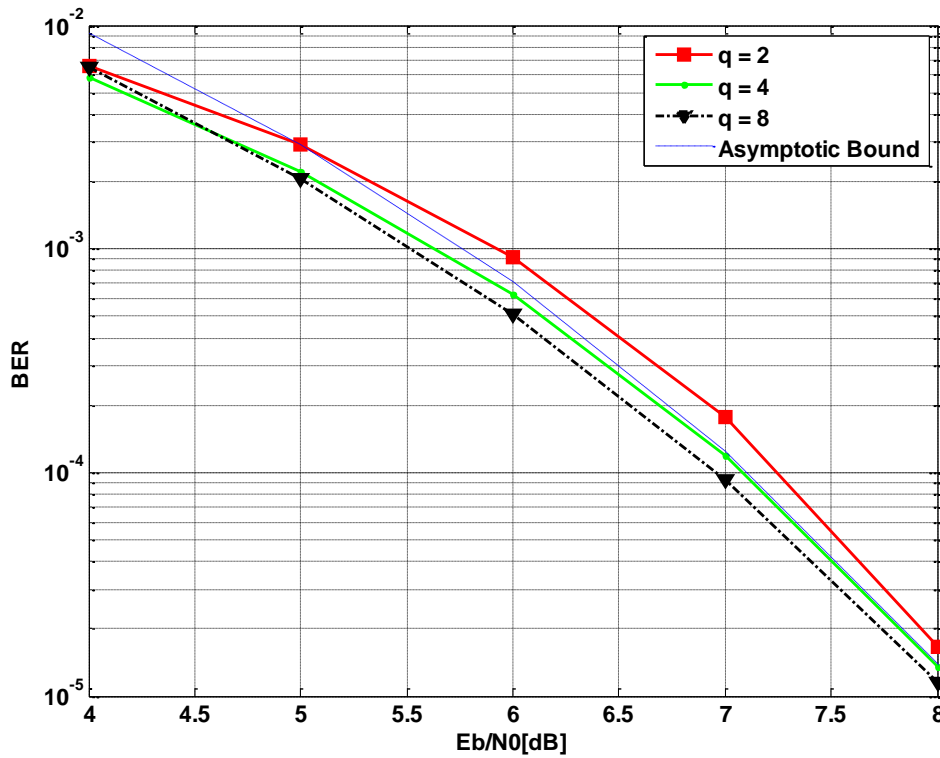


Figure 3.3: Performance of the (5,4) SPC code defined over  $\mathbb{Z}_q$

In Figure 3.4, the performance of the (5,4) SPC code defined over  $\mathbb{F}_8$  is compared with the performance of the (5,4) SPC code that is defined over  $\mathbb{Z}_8$ . As it is observed, for  $\{q = 2^p : p > 1\}$  and for the small signal-to-noise-ratios (SNR), an SPC code defined over  $\mathbb{Z}_q$  performs better than the same length SPC code defined over  $\mathbb{F}_q$ . However for the large SNRs, the performance of both codes approaches the asymptotic bound, since both codes have the same asymptotic bound, which is independent of the field and ring order, as it is given in Chapter 5. Therefore, SPC codes defined over  $\mathbb{Z}_q$  are better choices to be used as constituent code for the compound codes that need to perform at lower SNRs.

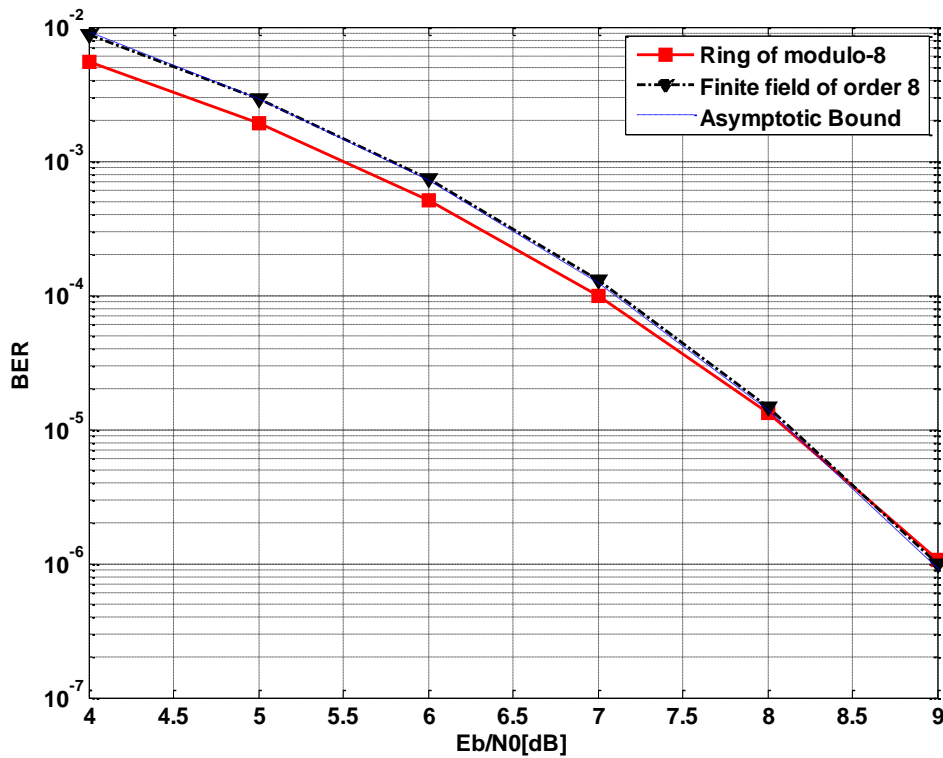


Figure 3.4: Performance of the (5,4) SPC code defined over  $\mathbb{F}_8$  and  $\mathbb{Z}_8$

As it was mentioned before, since both decoding algorithms are optimum, the decoding algorithm should not have any effect on the performance of the code. In Figure 3.5,

the performance of the (5,4) SPC code defined over  $\mathbb{Z}_8$  that is separately decoded by the two proposed decoding algorithms are compared and as it was expected, the code performance under different optimum decoding algorithms is the same. However, as it is shown in Figure 3.6, the APP decoding algorithm based on Fourier transform can be implemented only for the SPC codes defined over  $\mathbb{Z}_q$  and as it is observed, the (5,4) SPC code that is defined over  $\mathbb{F}_8$  and is decoded by the APP decoding algorithm based on Fourier transform, performs far away from its asymptotic bound.

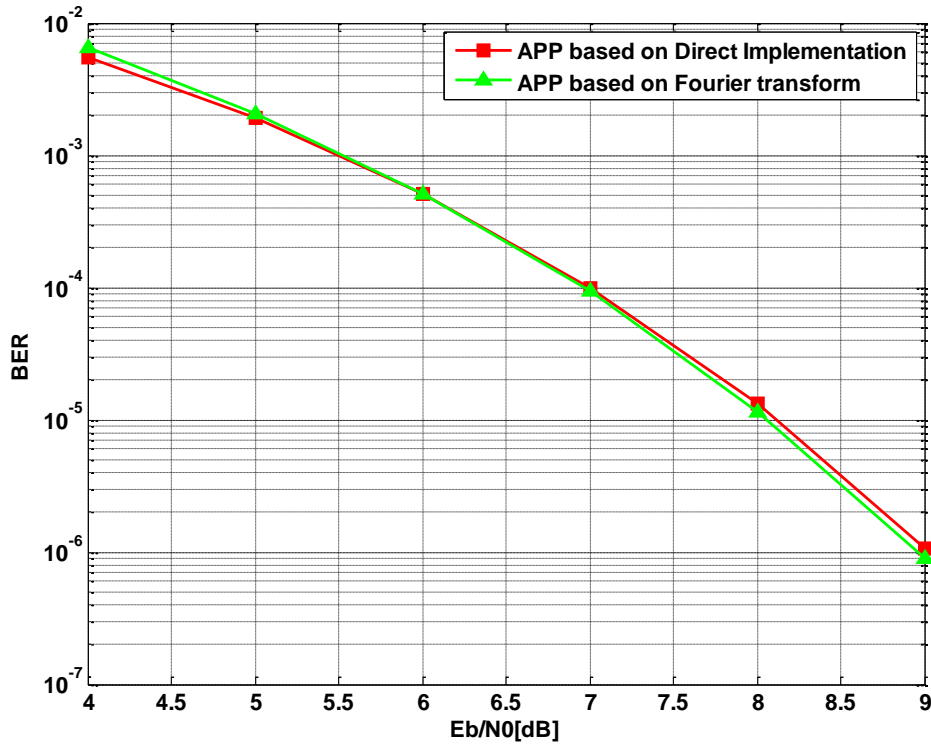


Figure 3.5: Performance of the (5,4) SPC code defined over  $\mathbb{Z}_8$  that is decoded with different decoding algorithms

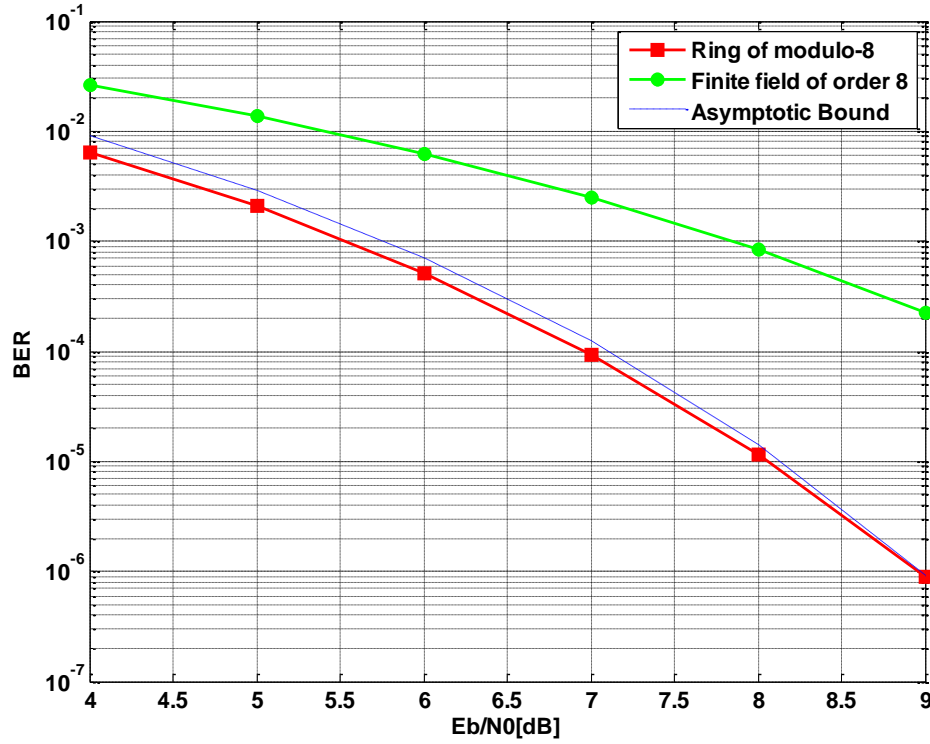


Figure 3.6: Performance of the (5,4) SPC code defined over  $\mathbb{F}_8$  and  $\mathbb{Z}_8$  that are decoded with the APP decoding algorithm based on Fourier transform

In Table 3.2, the complexity comparison for decoding the (5,4) SPC code defined over  $\mathbb{Z}_8$  that is decoded by different proposed APP decoding algorithms is given. The FFT algorithm is used for calculating the Fourier and inverse Fourier transforms in the Fourier based APP decoding algorithm. As it is observed, with a manageable increase in the memory requirement, the computational complexity is reduced by 75%.

Table 3.2  
Complexity comparison between the two proposed algorithms  
for decoding the (5,4) SPC code defined over  $\mathbb{Z}_8$

	Direct Implementation	DFT Based
Memory Cell	4	64
Addition	1023	449
Multiplication	3073	577

### 3.5. Conclusion

In this chapter, we studied the encoding and decoding of SPC codes over  $\mathbb{F}_q$  and  $\mathbb{Z}_q$ . Two different optimum APP decoding algorithms for decoding non-binary SPC codes were proposed. The first algorithm was based on straightforward implementation of the code's trellis which can be used for any SPC code, while in the second algorithm the APP values were calculated over the code's dual space by using Fourier transforms which can be implemented only for the SPC code defined over  $\mathbb{Z}_q$ .

Furthermore, we showed that despite the differences between the approaches that we considered for deriving DFT-based APP decoding algorithm and the one that is considered for the row-step of the FFT-BP algorithm, they both resulted in the same formula. The reason for this similarity was given and it was concluded that the FFT-BP decoding algorithm and subsequently, other algorithms that are derived from FFT-BP such as Log-FFT-BP and EMS are valid only for the LDPC codes defined over  $\mathbb{Z}_q$  and implementing them for the non-binary LDPC codes defined over  $\mathbb{F}_q$  are not mathematically correct.

Finally, simulation results were presented and it was observed that the performance of SPC codes defined over  $\mathbb{F}_q$  is independent of the field order, while the performance of SPC codes defined over  $\mathbb{Z}_q$  improves over a higher order ring. Therefore, the SPC codes defined over  $\mathbb{Z}_q$  can be considered as a good option to be used as constituent code in concatenated structures.



## CHAPTER 4

### SINGLE PARITY-CHECK TURBO PRODUCT CODES

#### 4.1. Introduction

The first idea for constructing long and powerful codes from combination of short and simple codes dates back to the invention of product codes by Elias in 1954 [63]. Product codes are efficient for controlling both random-error and burst-error patterns [69] and are suggested for applications with high coding rate requirements such as submarine cables, optical transport networks and networks at 100Gbit/sec [70].

Product codes can be constructed from any binary or non-binary, block or convolutional constituent codes in multiple dimensions. However, common choices are the two or three-dimensional product codes that are constructed from Bose–Chaudhuri–Hochquenghem (BCH) codes, binary single parity-check (SPC) codes or Hamming codes. A  $d$ -dimensional product code can be considered as a structured interleaved code where each of its information symbols (bits) is employed in  $d$  constituent codes. Despite the simplicity and weak performance of SPC codes, multidimensional binary SPC product codes have shown good performance under iterative decoding [66]. It is shown that a high-rate binary SPC product code exhibits similar performance compared with the same rate binary low-density parity-check code, yet having less encoding and decoding complexity [88]. Moreover, the

simulation results in [68] show that a binary SPC product code with length of 97751 and rate of 0.985 performs only 0.44 dB away from the Shannon-limit at a bit-error rate (BER) of  $10^{-5}$ .

The first soft decoding algorithm for a product code is given by Battail [64] and later Hagenauer *et al.* [65] showed that the turbo decoding principles can be used for decoding a product code. A turbo product (TP) code is a product code that is decoded by a soft-input soft-output (SISO) iterative decoding algorithm and it is shown [66-70] that they can achieve good performance.

High data-rate digital communication systems require high-order modulations and it is more convenient to use non-binary codes with appropriate alphabet size to match the constellation. Furthermore, by considering non-binary alphabets, an extra degree of freedom is added to the design parameters of a code. In [91], other advantages for non-binary turbo codes over their binary counterparts are also mentioned, including better convergence under iterative decoding, larger minimum distance, less sensitivity towards puncturing patterns and robustness for the flaws of the component-decoding algorithm.

As it was mentioned in the previous chapter, low-density parity-check (LDPC) codes can be considered as multiple parallel concatenations of SPC codes and therefore, non-binary LDPC codes can be considered as non-binary turbo-like codes. However, beside non-binary LDPC codes [15,16], the vast majority of turbo and turbo-like codes are binary codes. This mainly arises from the computational complexity and the amount of memory storage, which is required for optimal soft decoding of non-binary codes. Furthermore, employing sub-optimal decoding algorithms degrades the bit (or symbol) error rate performance of a code.

The first investigation on non-binary turbo-like codes was reported in [92] where Reed-Solomon (RS) codes were used as constituent code in a two-dimensional product structure and a sub-optimal SISO iterative decoding algorithm was employed for decoding the resulted product codes. However, due to the simplicity of the sub-optimum component decoder that was employed for decoding the RS component codes, the performance of the resulting product codes was far away from the channel capacity. The performance of the two-dimensional RS product codes can be improved by using a better sub-optimum component decoder [93]. Furthermore, it is shown in [94] that compared with the two-dimensional TP codes constructed from RS component codes, the two-dimensional TP codes

that are constructed from extended RS code can perform better under sub-optimum iterative decoding algorithms.

Turbo codes [4] were originally constructed as parallel concatenation of two interleaved binary convolutional codes. Furthermore, the feasibility of the extension of turbo codes to higher order finite fields was studied in [91,95-96] and non-binary turbo codes were constructed from parallel concatenation of convolutional component codes where each convolutional code has  $m$  ( $m \geq 2$ ) inputs. In [97], the performance of the parallel and serial non-binary turbo codes is compared and it is shown that the serial concatenated convolutional turbo codes have lower error floor compared with their parallel counterparts. Nevertheless, the research in [95,96] shows that the non-binary turbo codes do not perform better over higher order fields, yet the complexity of non-binary turbo codes increases over higher order fields.

Non-binary turbo codes are studied over finite ring of integer modulo- $q$ ,  $\mathbb{Z}_q$ , and it is shown that they have good performance over additive white Gaussian noise (AWGN) channel [81,98-100]. The simulation results in [100] shows that a  $\frac{1}{2}$  rate non-binary turbo code defined over  $\mathbb{Z}_3$  has 0.5 dB gain at BER of  $10^{-4}$  over its binary counterpart, while, a  $\frac{1}{3}$  rate non-binary turbo code defined over  $\mathbb{Z}_3$  has 0.1 dB loss compared with the same rate binary turbo code. The good-performance non-binary turbo codes can be designed over higher order rings of integer modulo- $q$ , yet all designed good-performance turbo codes have code rates less than  $\frac{1}{2}$  [81,98-100].

The results in [92-94] show that the high-rate non-binary turbo-like codes can be constructed based on two-dimensional product structure. Furthermore, it is shown that the performance of a moderate-length LDPC code improves over higher order fields [15,16] and since LDPC codes are constructed as parallel concatenation of SPC codes, it can be concluded that codes which are designed based on concatenation of non-binary SPC codes may perform better for short-length or medium-length block size. Moreover, the research on generalized low-density (GLD) and generalized irregular low-density (GILD) codes [60-62] shows that the parallel-concatenated SPC codes with denser parity-check matrices are better choices for designing small or medium length good-performance codes. Therefore, in this chapter, we study the two-dimensional, non-binary SPC turbo product (2D-SPC-TP) codes,

which are the two-dimensional SPC (2D-SPC) product codes that are decoded by SISO iterative decoding algorithms.

In [101], the maximum *a posteriori* probability (MAP) decoding of the non-binary 2D-SPC product codes is studied. In this chapter, different SISO iterative decoding algorithms for decoding the non-binary 2D-SPC product codes are presented and the simulation result shows that the performance of the 2D-SPC product codes is improved under iterative decoding. We consider non-binary SPC constituent codes that are defined over  $\mathbb{Z}_q$  as well as non-binary SPC constituent codes that are defined over finite field of order  $q$ ,  $\mathbb{F}_q$ . Different iterative decoding algorithms for decoding the 2D-SPC-TP codes are presented and the performance of the 2D-SPC-TP codes over an AWGN channel is studied. Simulation results show that regardless of the field's order, the performance of the 2D-SPC-TP codes defined over  $\mathbb{F}_{2^p}$  ( $p$  is any positive integer) remains the same for different field orders, yet the performance of the 2D-SPC-TP codes defined over  $\mathbb{Z}_{2^p}$  improves over higher order rings. This performance improvement is explained by analysis of the weight distribution that is given in Chapter 5.

The rest of the chapter is organized as follows. In Section 4.2, the structure of product codes is presented. In 4.3, different SISO iterative decoding algorithms for decoding the non-binary SPC product codes are presented and the complexity of each algorithm is discussed. In Section 4.4, the simulation results are presented. Finally in Section 4.5 a conclusion for this chapter is given.

## 4.2. The structure of product codes

In general, for constructing any product code, first, the message sequence is arranged in a hypercube of dimension  $d$ , with  $\{K_1, K_2, \dots, K_d\}$  symbols in each dimension. Then, each dimension is encoded with a constituent code like  $C^i$ , where  $C^i$  is a systematic linear block code with length of  $N_i$  and dimension of  $K_i$ . For a non-binary product code, each code symbol belongs to set  $\{0, 1, 2, \dots, q-1\}$ . It is obvious that the number of message symbols for the resulted product code is

$$K = \prod_{i=1}^d k_i \quad (4.1)$$

and the length of the resulted product code is

$$N = \prod_{i=1}^d N_i \quad (4.2)$$

Thus, the code rate of the resulted product code is

$$R = \frac{K}{N} = \frac{\prod_{i=1}^d k_i}{\prod_{i=1}^d N_i} = \prod_{i=1}^d \frac{k_i}{N_i} = \prod_{i=1}^d R_i \quad (4.3)$$

where  $R_i$  is the rate of the constituent code that is employed in the  $i^{\text{th}}$  dimension of the product code. Moreover, it can be proved [63] that the minimum distance of a product code is equal to

$$d_{\min} = \prod_{i=1}^d d_{\min_i} \quad (4.4)$$

where  $d_{\min_i}$  is the minimum distance of the constituent code employed in the  $i^{\text{th}}$  dimension. This product code is denoted by  $(N, K, d_{\min})_q$ . A two-dimensional product code, as it is depicted in Figure 4.1, is consisted of a data block, a set of parity-check symbols on each row, a set of parity-check symbols on each column, and a set of parity-check symbols on the parity-checks.

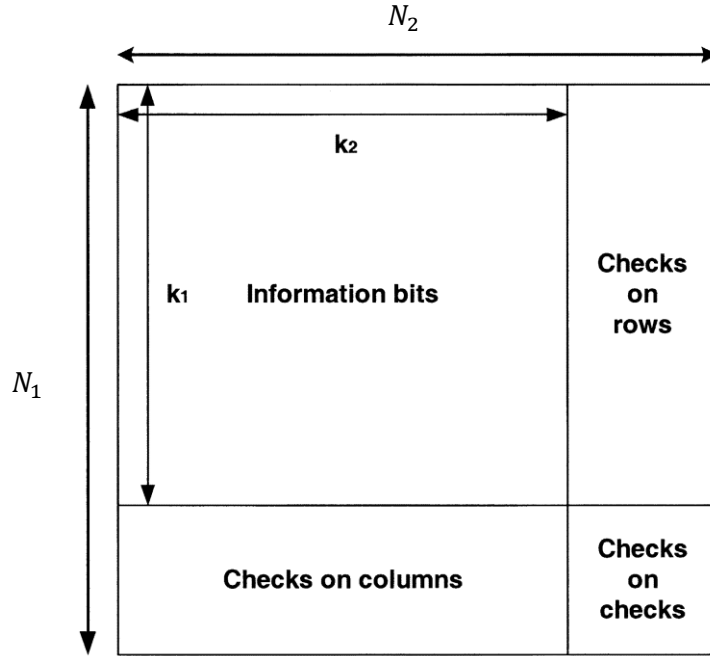


Figure 4.1: A general structure for a two-dimensional product code

SPC product codes are constructed from combination of SPC codes. A  $d$ -dimensional SPC product code that is constructed from the same length  $(K + 1, K)$  SPC codes encodes  $K^d$  information symbols and its code rate is

$$R = \left( \frac{K}{K + 1} \right)^d \quad (4.5)$$

which by tending  $K$  towards infinity the code rate approaches one. Therefore, non-binary SPC product codes can be considered as an error-correcting scheme for the applications where high-rate non-binary codes are required. In this chapter, we only consider the two-dimensional product codes that are constructed from the same length non-binary

$(K + 1, K)$  SPC codes and their check on check symbol is punctured. This means that every row and column of message data has only one parity-check symbol. The minimum Hamming distance of this code is three and it is depicted in Figure 4.2. We denote this code as  $(K^2 + 2K, K^2, 3)_q$ .

$u_{1,1}$	$u_{1,2}$	$\dots$	$\dots$	$u_{1,K}$	$P_1^-$
$u_{2,1}$	$u_{2,2}$	$\dots$	$\dots$	$u_{2,K}$	$P_2^-$
$u_{3,1}$	$u_{3,2}$	$\dots$	$\dots$	$u_{3,K}$	$P_3^-$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	
$u_{K,1}$	$u_{K,2}$	$\dots$	$\dots$	$u_{K,K}$	$P_K^-$
$P_1^l$	$P_2^l$	$\dots$	$\dots$	$P_K^l$	

Figure 4.2: Two-dimensional SPC product code

$\{u_{i,j} \in \{0,1,2,\dots,q-1\} : 1 \leq i \leq K, 1 \leq j \leq K\}$  are the message symbols,  $\{p_i^- ; 1 \leq i \leq K\}$  are the row parity-check symbols and  $\{p_j^l ; 1 \leq j \leq K\}$  are the column parity-check symbols. The parity-check equations are given by

$$u_{i,1} \oplus u_{i,2} \oplus \dots \oplus u_{i,K} \oplus p_i^- = 0 \quad 1 \leq i \leq K \quad (4.6)$$

$$u_{1,j} \oplus u_{2,j} \oplus \dots \oplus u_{K,j} \oplus p_j^l = 0 \quad 1 \leq j \leq K \quad (4.7)$$

If the  $\oplus$  operation is defined over  $\mathbb{F}_q$  then the resulting SPC product code is defined over  $\mathbb{F}_q$ , and if the  $\oplus$  operation is defined over  $\mathbb{Z}_q$  then the resulted SPC product code is defined over  $\mathbb{Z}_q$ . Both  $(\mathbb{Z}_q, \oplus)$  and  $(\mathbb{F}_q, \oplus)$  are commutative group, therefore, they are close

under addition operation and subsequently, we have  $\{p_i^- \in \{0,1,2,\dots,q-1\} : 1 \leq i \leq K\}$  and  $\{p_j^\perp \in \{0,1,2,\dots,q-1\} : 1 \leq j \leq K\}$ .

### 4.3. Turbo decoding of non-binary SPC code

Let  $C$  be a  $(K^2 + 2K, K^2, 3)_q$  2D-SPC product code as it is depicted in Figure 4.2 and  $\mathbf{v} = (v_0, v_1, \dots, v_{K^2+2K-1})$  be any codeword of code  $C$ . The codewords are transmitted over a discrete-time, memoryless, noisy channel and the soft decision received sequence  $\mathbf{r} = (r_0, r_1, \dots, r_{K^2+2K-1})$  corresponding to a transmitted codeword is available at the decoder. Without loss of generality, all codewords are assumed to be in systematic format such that the first  $K^2$  symbols of each codeword are information symbols and the remaining  $2K$  symbols are the parity-check symbols. Each information symbol  $\{v_i : 0 \leq i \leq K^2 - 1\}$  involves a horizontal  $(K+1, K)$  SPC constituent code that is denoted by  $C^-$  and a vertical  $(K+1, K)$  SPC constituent code that is denoted by  $C^\perp$ . Let  $\mathbf{v}^- = (v_0^-, v_1^-, \dots, v_K^-)$  be any codeword of code  $C^-$  where its  $l^{\text{th}}$  ( $0 \leq l < K$ ) element is the information symbol  $v_i$ ,  $v_l^- = v_i$ , and  $\mathbf{v}^\perp = (v_0^\perp, v_1^\perp, \dots, v_K^\perp)$  be any codeword of code  $C^\perp$  where its  $n^{\text{th}}$  ( $0 \leq n < K$ ) element is the information symbol  $v_i$ ,  $v_n^\perp = v_i$ . Moreover, the soft decision received sequence that corresponds to  $\mathbf{v}^-$  is denoted by  $\mathbf{r}^- = (r_0^-, r_1^-, \dots, r_K^-)$  and the soft decision received sequence that corresponds to  $\mathbf{v}^\perp$  is denoted by  $\mathbf{r}^\perp = (r_0^\perp, r_1^\perp, \dots, r_K^\perp)$  and it is obvious that by knowing  $\mathbf{r}$ , the values for  $\mathbf{r}^-$  and  $\mathbf{r}^\perp$  are known.

From Chapter 2, we know that vector of log-likelihood ratios (LLR)  $L_i = \{L_{i,\mu} : 0 \leq \mu \leq q-1\}$  for the information symbols  $\{v_i : 0 \leq i \leq K^2 - 1\}$  of code  $C$  can be calculated from  $L_{i,\mu}$  that is defined as

$$L_{i,\mu} \stackrel{\text{def}}{=} \ln \left[ \frac{\sum_{\substack{\mathbf{v} \in C \\ v_i = \mu}} Pr(\mathbf{v}|\mathbf{r})}{\sum_{\substack{\mathbf{v} \in C \\ v_i = 0}} Pr(\mathbf{v}|\mathbf{r})} \right] \quad (4.8)$$



and the hard decision is made by  $\hat{v}_i = \arg \max_{\mu \in \{0,1,\dots,q-1\}} \{L_{i,\mu}\}$ . Since  $C$  is defined as the  $(K^2 + 2K, K^2, 3)_q$  2D-SPC product code and  $v_i$  involves in only two parity-check equations, we have

$$L_{i,\mu} = \ln \left[ \frac{\sum_{\substack{\mathbf{v}^- \in C^- \\ v_i^- = \mu}} \sum_{\substack{\mathbf{v}^l \in C^l \\ v_n^l = \mu}} Pr(\mathbf{v}^-; \mathbf{v}^l | \mathbf{r})}{\sum_{\substack{\mathbf{v}^- \in C^- \\ v_i^- = 0}} \sum_{\substack{\mathbf{v}^l \in C^l \\ v_n^l = 0}} Pr(\mathbf{v}^-; \mathbf{v}^l | \mathbf{r})} \right] \quad (4.9)$$

The information symbols are mutually independent and equiprobable, therefore, for any  $\mathbf{v}^- \in C^-$  and  $\mathbf{v}^l \in C^l$  the sequence of random variables  $(\mathbf{v}^- | \mathbf{r}; v_i^-)$  is independent from the sequence of random variables  $(\mathbf{v}^l | \mathbf{r}; v_n^l)$ . Therefore,

$$L_{i,\mu} = \ln \left[ \frac{\sum_{\substack{\mathbf{v}^- \in C^- \\ v_i^- = \mu}} Pr(\mathbf{v}^- | \mathbf{r}^-)}{\sum_{\substack{\mathbf{v}^- \in C^- \\ v_i^- = 0}} Pr(\mathbf{v}^- | \mathbf{r}^-)} \right] + \ln \left[ \frac{\sum_{\substack{\mathbf{v}^l \in C^l \\ v_n^l = \mu}} Pr(\mathbf{v}^l | \mathbf{r}^l)}{\sum_{\substack{\mathbf{v}^l \in C^l \\ v_n^l = 0}} Pr(\mathbf{v}^l | \mathbf{r}^l)} \right] \quad (4.10)$$

The MAP decision for the symbol  $v_i$  of code  $C$  can be made by  $\hat{v}_i = \arg \max_{\mu \in \{0,1,\dots,q-1\}} \{L_{i,\mu}\}$  and since the information symbols are assumed to be equiprobable, the MAP decision is equivalent to the maximum-likelihood decision for the symbol  $v_i$ .

The block diagram of the turbo decoder for the 2D-SPC-TP codes is shown in Figure 4.3. In a turbo decoder, the soft-output value for each dimension is calculated from the channel information and the extrinsic information shared by the other dimension. This means that the soft-output value of one dimension is shared with the other dimension so that it can be used as extrinsic information in the next iteration. Once both dimensions are decoded, the decoding iteration is complete. This algorithm can be repeated as many times as required. However, after the first iteration, the parity-check equations are statistically dependent and

therefore, (4.10) is no longer valid and the maximum value of the summation of a symbol's LLR values over its two dimensions cannot be considered as the maximum-likelihood value for that symbol. However, the comparison between the simulation result and the asymptotic performance bound for the maximum-likelihood decoding of a 2D-SPC product code shows that the performance of a 2D-SPC product code is improved by iterative decoding algorithm.

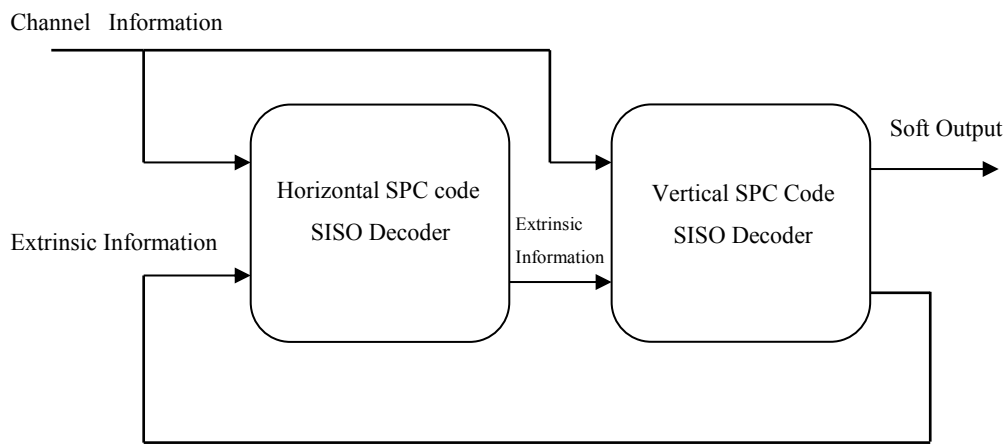


Figure 4.3: Turbo decoder for the 2D-SPC-TP code

In this section, three different iterative decoding algorithms for decoding the 2D-SPC-TP codes are presented. These algorithms differ in the way that the SISO component decoders are realized. In the first algorithm, which in this chapter is referred to as Algorithm I, the SISO component decoders are constructed based on direct implementation of the optimal APP algorithm. In the second algorithm, which we call it as Modified-Algorithm I (MA-I), the SISO component decoders are constructed based on the sub-optimal max-log-APP decoding algorithm. Finally, the SISO component decoders for the third algorithm, which in this chapter we call it as Algorithm II, are constructed from the optimal APP decoding algorithm based on Fourier transform. Both Algorithm I and algorithm MA-I can be used for any 2D-SPC-TP codes, while the Algorithm II can be implemented only for the 2D-SPC-TP codes defined over  $\mathbb{Z}_q$ .

### 4.3.1. Algorithm I

In this algorithm, the SISO decoders are implemented based on direct implementation of the APP algorithm, therefore this algorithm can be used for decoding any non-binary SPC constituent code. From Chapter 3 we know that the LLR value for the information symbol  $\{v_i : 0 \leq i \leq K^2 - 1\}$  can be calculated from

$$L_{i,\mu} = L_\mu(v_i) + L_\mu(v_i|r_i) + \ln \left( \frac{\sum_{\substack{v \in C \\ v_i = \mu}} \prod_{\substack{m=0 \\ m \neq i}}^K e^{(L_{v_m}(v_m) + L_{v_m}(v_m|r_m))}}{\sum_{\substack{v \in C \\ v_i = 0}} \prod_{\substack{m=0 \\ m \neq i}}^K e^{(L_{v_m}(v_m) + L_{v_m}(v_m|r_m))}} \right) \quad (4.11)$$

Therefore, the LLR value for the horizontal dimension is calculated by

$$L_{i,\mu}^- = L_\mu(v_i) + L_\mu(v_i|r_i) + \ln \left( \frac{\sum_{\substack{v^- \in C^- \\ v_i^- = \mu}} \prod_{\substack{m=0 \\ m \neq l}}^K e^{(L_{v_m^-}(v_m^-) + L_{v_m^-}(v_m^-|r_m^-))}}{\sum_{\substack{v^- \in C^- \\ v_i^- = 0}} \prod_{\substack{m=0 \\ m \neq l}}^K e^{(L_{v_m^-}(v_m^-) + L_{v_m^-}(v_m^-|r_m^-))}} \right) \quad (4.12)$$

and the LLR value for the vertical dimension is calculated by

$$L_{i,\mu}^{\downarrow} = L_\mu(v_i) + L_\mu(v_i|r_i) + \ln \left( \frac{\sum_{\substack{v \in C \\ v_n^{\downarrow} = \mu}} \prod_{\substack{m=0 \\ m \neq n}}^K e^{(L_{v_m^{\downarrow}}(v_m^{\downarrow}) + L_{v_m^{\downarrow}}(v_m^{\downarrow}|r_m^{\downarrow}))}}{\sum_{\substack{v \in C \\ v_n^{\downarrow} = 0}} \prod_{\substack{m=0 \\ m \neq n}}^K e^{(L_{v_m^{\downarrow}}(v_m^{\downarrow}) + L_{v_m^{\downarrow}}(v_m^{\downarrow}|r_m^{\downarrow}))}} \right) \quad (4.13)$$

and an iterative decoding algorithm can be implemented as follows:

1. **Initialization:** The algorithm begins by calculating  $L_{i,\mu}^-$  and  $L_{i,\mu}^l$  from (4.12) and (4.13) respectively. Since the information symbols are assumed to be equiprobable, the *a priori* LLR values for all symbols are taken as zero in the calculations of the first iteration. Therefore,  $\{L_{v_m^l}(v_m^l) = L_{v_m^-}(v_m^-) = 0 : 0 \leq m \leq K\}$  and  $\{L_\mu(v_i) = 0 : 0 \leq i \leq K^2 - 1, 0 \leq \mu \leq q - 1\}$ .
2. **Decode each dimension:** From the second iteration onwards, the *a priori* LLR values for the symbols of one dimension are substituted by the previously calculated LLR values for those particular symbols from the other dimension. Therefore,  $L_{i,\mu}^-$  is calculated by using the extrinsic information provided by the vertical dimension and  $L_{i,\mu}^l$  is calculated by using the extrinsic information provided by the horizontal dimension. The extrinsic information provided by the vertical dimension for each iteration are the values of  $\{L_{i,\mu}^l : 0 \leq i \leq K^2 - 1, 0 \leq \mu \leq q - 1\}$ , which are calculated in the previous iteration and are substituted for  $\{L_{v_m^-}(v_m^-) : 0 \leq m \leq K\}$  in the current iteration in order to calculate  $L_{i,\mu}^-$ . Similarly, the extrinsic information provided by the horizontal dimension for each iteration are the values of  $\{L_{i,\mu}^- : 0 \leq i \leq K^2 - 1, 0 \leq \mu \leq q - 1\}$ , which are calculated in the previous iteration and are substituted for  $\{L_{v_m^l}(v_m^l) : 0 \leq m \leq K\}$  in the current iteration in order to calculate  $L_{i,\mu}^l$ .
3. **Iteration:** Step 2 can be repeated as many times as required and eventually a hard decision for the information symbol  $\{v_i : 0 \leq i \leq K^2 - 1\}$  is made by 
$$\hat{v}_i = \arg \max_{\mu \in \{0,1,\dots,q-1\}} \{L_{i,\mu}^- + L_{i,\mu}^l\}.$$

#### 4.3.2. Modified-Algorithm I

The SISO decoders for Algorithm I are implemented based on direct implementation of the APP algorithm. For reducing the computational complexity of the APP decoding algorithm, the component decoders can be constructed based on sub-optimal APP algorithms

such as max-log-MAP [65,72], but employing the sub-optimal decoding algorithms, degrades the bit (or symbol) error rate performance of a code. However, this is not always the situation and as it is shown in [49], the performance of LDPC codes whose Tanner graph contains many short cycles improves using sub-optimum decoding algorithms.

The SISO decoders for MA-I are implemented based on max-log-APP algorithm, which is a sub-optimal algorithm. Therefore, the computational complexity of MA-I is less than the computational complexity of algorithm I. Since the Tanner graph for 2D-SPC-TP codes contains many short cycles, the performance of the 2D-SPC-TP codes might be improved using this sub-optimum component decoder. The simulation results, which are presented in the next section, show that 2D-SPC-TP codes perform better using MA-I decoding algorithm. Similar to algorithm I, the MA-I can also be used for any non-binary SPC code. The LLR value for the horizontal dimension is calculated by

$$L_{i,\mu}^- = L_\mu(v_i) + L_\mu(v_i|r_i) + \ln \left( \frac{\max_{\substack{v^- \in C^- \\ v_l^- = \mu}} \left\{ \prod_{\substack{m=0 \\ m \neq l}}^K e^{(L_{v_m^-}(v_m^-) + L_{v_m^-}(v_m^-|r_m^-))} \right\}}{\max_{\substack{v^- \in C^- \\ v_l^- = 0}} \left\{ \prod_{\substack{m=0 \\ m \neq l}}^K e^{(L_{v_m^-}(v_m^-) + L_{v_m^-}(v_m^-|r_m^-))} \right\}} \right) \quad (4.14)$$

and the LLR value for the vertical dimension is calculated by

$$L_{i,\mu}^l = L_\mu(v_i) + L_\mu(v_i|r_i) + \ln \left( \frac{\max_{\substack{v^l \in C^l \\ v_n^l = \mu}} \left\{ \prod_{\substack{m=0 \\ m \neq n}}^K e^{(L_{v_m^l}(v_m^l) + L_{v_m^l}(v_m^l|r_m^l))} \right\}}{\max_{\substack{v^l \in C^l \\ v_n^l = 0}} \left\{ \prod_{\substack{m=0 \\ m \neq n}}^K e^{(L_{v_m^l}(v_m^l) + L_{v_m^l}(v_m^l|r_m^l))} \right\}} \right) \quad (4.15)$$

and an iterative decoding algorithm can be implemented as follows:

1. **Initialization:** The algorithm begins by calculating  $L_{i,\mu}^-$  and  $L_{i,\mu}^|$  from (4.14) and (4.15) respectively. Since the information symbols are assumed to be equiprobable, the *a priori* LLR values for all symbols are taken as zero in the calculations of the first iteration. Therefore,  $\{L_{v_m^|}(v_m^|) = L_{v_m^-}(v_m^-) = 0 : 0 \leq m \leq K\}$  and  $\{L_\mu(v_i) = 0 : 0 \leq i \leq K^2 - 1, 0 \leq \mu \leq q - 1\}$ .
2. **Decode each dimension:** From the second iteration onwards, the *a priori* LLR values for the symbols of one dimension are substituted by the previously calculated LLR values for those particular symbols from the other dimension. Therefore,  $L_{i,\mu}^-$  is calculated using the extrinsic information provided by the vertical dimension and  $L_{i,\mu}^|$  is calculated using the extrinsic information provided by the horizontal dimension. The extrinsic information provided by the vertical dimension for each iteration are the values of  $\{L_{i,\mu}^| : 0 \leq i \leq K^2 - 1, 0 \leq \mu \leq q - 1\}$ , which are calculated in the previous iteration and are substituted for  $\{L_{v_m^-}(v_m^-) : 0 \leq m \leq K\}$  in the current iteration in order to calculate  $L_{i,\mu}^-$ . Similarly, the extrinsic information provided by the horizontal dimension for each iteration are the values of  $\{L_{i,\mu}^- : 0 \leq i \leq K^2 - 1, 0 \leq \mu \leq q - 1\}$ , which are calculated in the previous iteration and are substituted for  $\{L_{v_m^|}(v_m^|) : 0 \leq m \leq K\}$  in the current iteration in order to calculate  $L_{i,\mu}^|$ .
3. **Iteration:** Step 2 can be repeated as many times as required and eventually a hard decision for the information symbol  $\{v_i : 0 \leq i \leq K^2 - 1\}$  is made by  $\hat{v}_i = \arg \max_{\mu \in \{0,1,\dots,q-1\}} \{L_{i,\mu}^- + L_{i,\mu}^|\}$ .

#### 4.3.3. Algorithm II

In this algorithm, the SISO decoders are constructed from the optimal APP decoding algorithm based on Fourier transform. As it was mentioned in Chapter 2, for a high-rate linear block code defined over  $\mathbb{Z}_q$ , the minimal-storage APP decoding algorithm can be achieved by using the APP decoding algorithm that is implemented based on the code's dual space. Therefore, for the SPC codes defined over  $\mathbb{Z}_q$ , which are high-rate codes, the

minimal-storage APP decoding algorithm is implemented based on the codes' dual space. Furthermore, we showed that by using the concept of Fourier transform, the computational complexity of the dual implementation of the APP decoding algorithm is reduced. The idea of using Fourier transform for decoding codes can also be found in decoding of the LDPC codes [51-54]. However, in LDPC codes, the Fourier transform is used to convert the convolution operation into multiplication operation and thus, simplify the decoding algorithm, while in this section, we use the concept of Fourier transform to avoid the repetitive calculation and reduce the computational complexity of the decoding algorithm. This algorithm can be used only for decoding the non-binary SPC codes that are defined over  $\mathbb{Z}_q$ .

From Chapter 3 we know that the LLR value based on Fourier transform for the horizontal dimension is calculated by

$$L_{i,\mu}^- = L_\mu(v_i) + L_\mu(v_i|r_i) + \ln \left( \frac{\mathcal{F}_{\alpha=-\mu}^{-1} \left\{ \prod_{m=0}^K \mathcal{F}_s \left\{ e^{(L_\alpha(v_m^-) + L_\alpha(v_m^-|r_m^-))} \right\} \right\}}{\mathcal{F}_{\alpha=0}^{-1} \left\{ \prod_{m=0}^K \mathcal{F}_s \left\{ e^{(L_\alpha(v_m^-) + L_\alpha(v_m^-|r_m^-))} \right\} \right\}} \right) \quad (4.16)$$

and the LLR value for the vertical dimension is calculated by

$$L_{i,\mu}^| = L_\mu(v_i) + L_\mu(v_i|r_i) + \ln \left( \frac{\mathcal{F}_{\alpha=-\mu}^{-1} \left\{ \prod_{m=0}^K \mathcal{F}_s \left\{ e^{(L_\alpha(v_m^|) + L_\alpha(v_m^|r_m^|))} \right\} \right\}}{\mathcal{F}_{\alpha=0}^{-1} \left\{ \prod_{m=0}^K \mathcal{F}_s \left\{ e^{(L_\alpha(v_m^|) + L_\alpha(v_m^|r_m^|))} \right\} \right\}} \right) \quad (4.17)$$

and an iterative decoding algorithm can be implemented as follows:

1. **Initialization:** The algorithm begins by calculating  $L_{i,\mu}^-$  and  $L_{i,\mu}^|$  from (4.16) and (4.17) respectively. Since the information symbols are assumed to be equiprobable,

the *a priori* LLR values for all symbols are taken as zero in the calculations of the first iteration. Therefore,  $\{L_\alpha(v_m^l) = L_\alpha(v_m^-) = 0 : 0 \leq m \leq K, 0 \leq \alpha \leq q-1\}$  and  $\{L_\mu(v_i) = 0 : 0 \leq i \leq K^2 - 1, 0 \leq \mu \leq q-1\}$ .

2. **Decode each dimension:** From the second iteration onwards, the *a priori* LLR values for the symbols of one dimension are substituted by the previously calculated LLR values for those particular symbols from the other dimension. Therefore,  $L_{i,\mu}^-$  is calculated using the extrinsic information provided by the vertical dimension and  $L_{i,\mu}^l$  is calculated using the extrinsic information provided by the horizontal dimension. The extrinsic information provided by the vertical dimension for each iteration are the values of  $\{L_{i,\mu}^l : 0 \leq i \leq K^2 - 1, 0 \leq \mu \leq q-1\}$ , which are calculated in the previous iteration and are substituted for  $\{L_{v_m^-}(v_m^-) : 0 \leq m \leq K\}$  in the current iteration in order to calculate  $L_{i,\mu}^-$ . Similarly, the extrinsic information provided by the horizontal dimension for each iteration are the values of  $\{L_{i,\mu}^- : 0 \leq i \leq K^2 - 1, 0 \leq \mu \leq q-1\}$ , which are calculated in the previous iteration and are substituted for  $\{L_{v_m^l}(v_m^l) : 0 \leq m \leq K\}$  in the current iteration in order to calculate  $L_{i,\mu}^l$ .
3. **Iteration:** Step 2 can be repeated as many times as required and eventually a hard decision for the information symbol  $\{v_i : 0 \leq i \leq K^2 - 1\}$  is made by

$$\hat{v}_i = \arg \max_{\mu \in \{0,1,\dots,q-1\}} \{L_{i,\mu}^- + L_{i,\mu}^l\}.$$

#### 4.4. Simulation results

In this section, the simulation results for the 2D-SPC-TP codes are presented. It is assumed that all codewords are modulated by binary phase-shift keying (BPSK) modulation scheme and are transmitted over a binary input AWGN channel with double-sided noise power spectral density of  $\frac{N_0}{2}$ . Therefore, the symbols need to be mapped to bipolar sequences as



$$v_i \rightarrow \bar{c}_i = (c_i^1, c_i^2, \dots, c_i^p) \rightarrow \bar{v}_i = (v_i^1, v_i^2, \dots, v_i^p) \quad (4.18)$$

where  $\bar{c}_i$  is the corresponding binary sequence for the symbol  $v_i$  such that  $\{c_i^m \in \{0,1\} : 1 \leq m \leq p\}$  and  $\bar{v}_i$  is the corresponding bipolar sequence for the symbol  $v_i$  such that  $\{v_i^m = 2c_i^m - 1 : 1 \leq m \leq p\}$  and the length of both sequences are equal to  $p = \log_2 q$ . For example:

$$v_i = 9 \rightarrow \bar{c}_i = (1,0,0,1) \rightarrow \bar{v}_i = (1,-1,-1,1)$$

We assume that at the receiver side, the sequence  $\mathbf{r} = (\bar{r}_0, \bar{r}_2, \dots, \bar{r}_{K^2+2K-1})$  is received. The noise vector is represented by  $\boldsymbol{\eta} = (\bar{\eta}_0, \bar{\eta}_2, \dots, \bar{\eta}_{K^2+2K-1})$ , where  $\bar{\eta}_i = (\eta_i^1, \eta_i^2, \dots, \eta_i^p)$  is the corresponding noise sequence for the bipolar sequence of the symbol  $v_i$ . Therefore,  $\mathbf{r} = \mathbf{v} + \boldsymbol{\eta}$  and  $\bar{\mathbf{r}}_i = \bar{\mathbf{v}}_i + \bar{\boldsymbol{\eta}}_i$ . Since the codewords are transmitted over an AWGN channel, the noise is independent from transmitted symbol and  $\{\eta_i^m : 0 \leq i \leq N-1, 1 \leq m \leq p\}$  are zero-mean, identically, independent distributed (i.i.d.) Gaussian random variables with variance  $\sigma^2 = \frac{N_0}{2}$ . Therefore, the LLR for the conditional random variable  $(v_i|r_i)$  is calculated as

$$L_\mu(v_i|r_i) = \ln \left( \frac{\exp\left(-\frac{\|\bar{\mathbf{r}}_i - \bar{\boldsymbol{\mu}}\|^2}{2\sigma^2}\right)}{\exp\left(-\frac{\|\bar{\mathbf{r}}_i - \bar{\mathbf{0}}\|^2}{2\sigma^2}\right)} \right) \quad (4.19)$$

where  $\bar{\boldsymbol{\mu}}$  and  $\bar{\mathbf{0}}$  are the corresponding bipolar sequences for  $\mu$  and 0 respectively and  $\|\mathbf{X}\|$  denotes the norm of the vector  $\mathbf{X}$ .

$$\|\bar{\mathbf{r}}_i - \bar{\boldsymbol{\mu}}\|^2 = \|\bar{\mathbf{r}}_i\|^2 + \|\bar{\boldsymbol{\mu}}\|^2 - 2(\bar{\mathbf{r}}_i \odot \bar{\boldsymbol{\mu}}) \quad (4.20)$$

where  $\odot$  represents the inner product between the two vectors. Since the energy of all symbols are equal,  $\|\bar{\mu}\|^2 = \|\bar{\mathbf{0}}\|^2$  and after cancelling out the common factors we have

$$L_{\mu}(v_i|r_i) = \sum_{j=1}^p \frac{(\mu^j + 1)r_i^j}{\sigma^2} \quad (4.21)$$

where  $\mu^j$  is the  $j^{\text{th}}$  component of  $\bar{\mu}$  and  $r_i^j$  is the  $j^{\text{th}}$  component of  $\bar{r}_i$ . The formula given in (4.21), is used to calculate the LLR values of the conditional random variables in (4.12)-(4.17).

The performance of the  $(8,4,3)_8$  2D-SPC-TP code that is defined over  $\mathbb{Z}_8$  and is decoded by Algorithm I, is depicted in Figure 4.4 and it is shown that the iterative decoding improves the performance of the code. This improvement can be explained from the mutual information that exists between the two dimensions. However, as it is observed, the code performance converges after two iterations and further iterations after convergence has less effect on the performance of the code. This is because after certain number of iterations, the two dimensions become so correlated that the mutual information between them has less effect on their decoding outcomes. Therefore, having more iteration after convergence does not improve the performance of the code and is not required. The effect of number of iterations on algorithm MA- I and Algorithm II is shown in Figures 4.5 and 4.6 respectively and as it is seen, two iterations for decoding a 2D-SPC-TP code is sufficient.

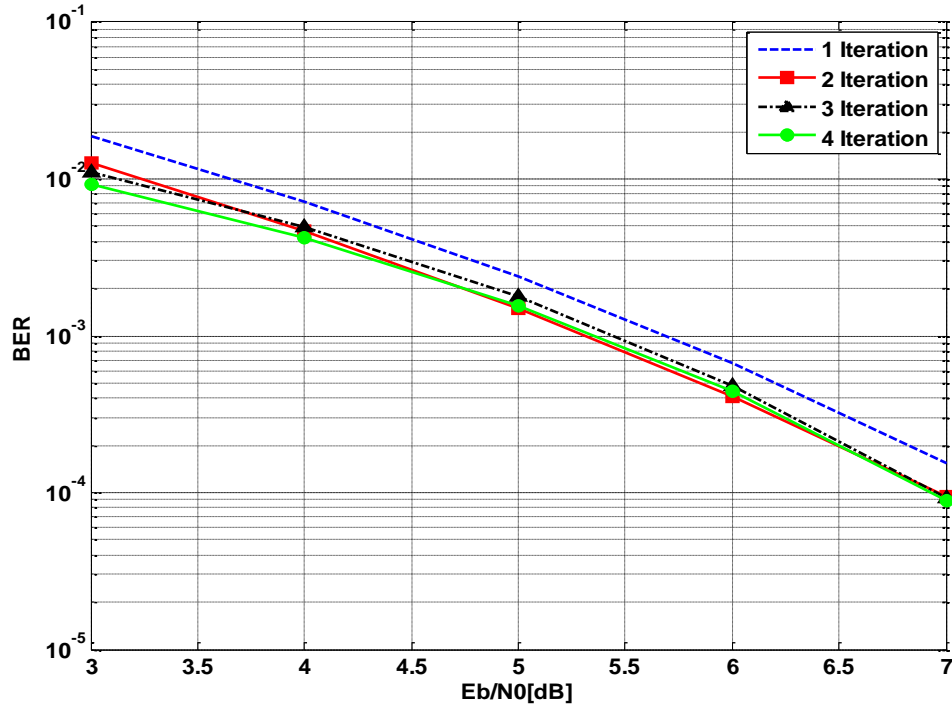


Figure 4.4: Effect of iteration on the performance of the  $(8,4,3)_8$  2D-SPC-TP code that is defined over  $\mathbb{Z}_8$  and is decoded by Algorithm I

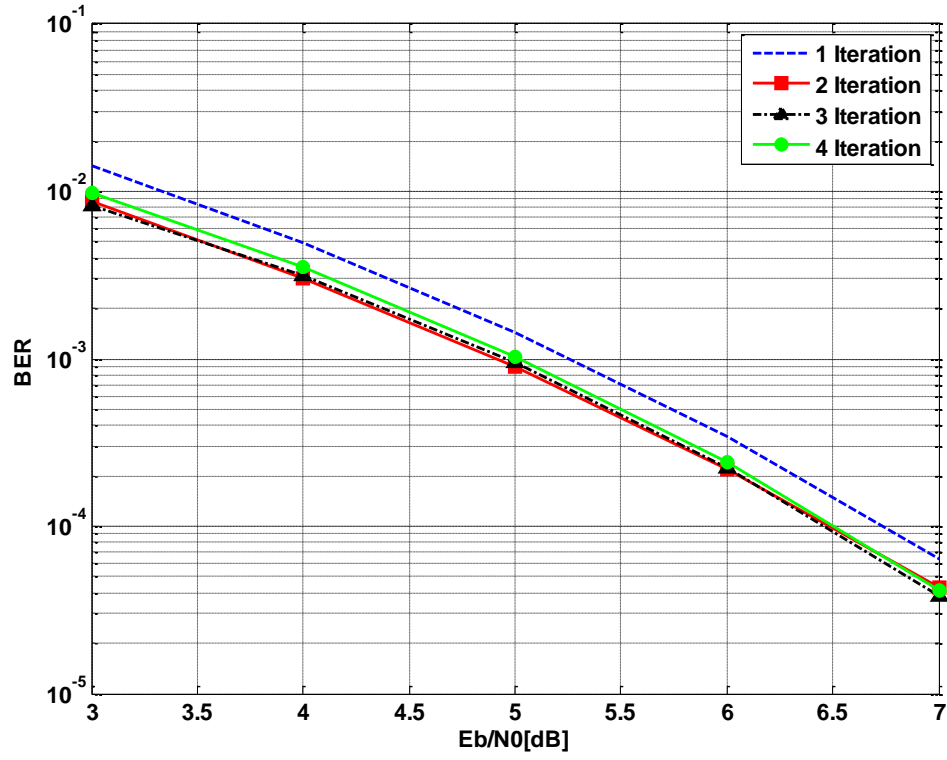


Figure 4.5: Effect of iteration on the performance of the  $(8,4,3)_8$  2D-SPC-TP code that is defined over  $\mathbb{Z}_8$  and is decoded by algorithm MA-I

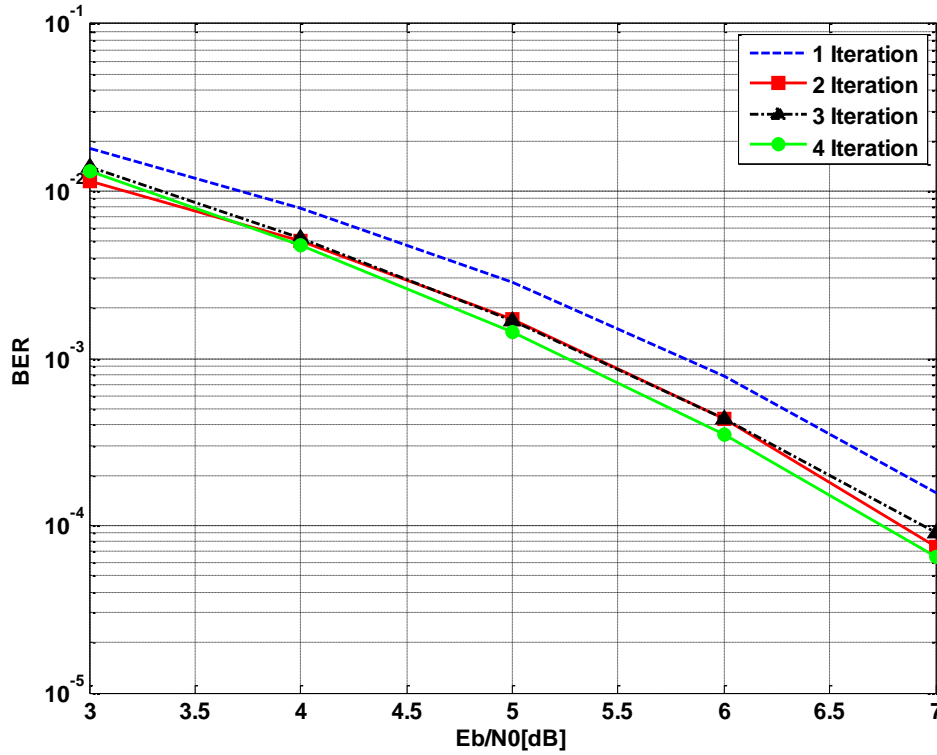


Figure 4.6: Effect of iteration on the performance of the  $(8,4,3)_8$  2D-SPC-TP code that is defined over  $\mathbb{Z}_8$  and is decoded by Algorithm II

The simulation results for the performance of a 2D-SPC-TP code under different iterative decoding algorithms are presented in Figures 4.7 and 4.8. In Figure 4.7, the performance of the  $(63,49,3)_4$  2D-SPC-TP code that is defined over  $\mathbb{F}_4$  and is separately decoded by Algorithm I and algorithm MA-I, is presented. By comparing the code performance under different decoding algorithms, the decoders' performance can be compared. The code performance is also compared with the maximum-likelihood decoding asymptotic bound for the  $(63,49,3)_4$  2D-SPC product code, which is calculated in Chapter 5. As it is seen, Algorithm I performs better than algorithm MA-I for the small signal-to-noise-ratios (SNR); and for large SNRs, turbo decoders in general outperform the maximum-likelihood decoders.

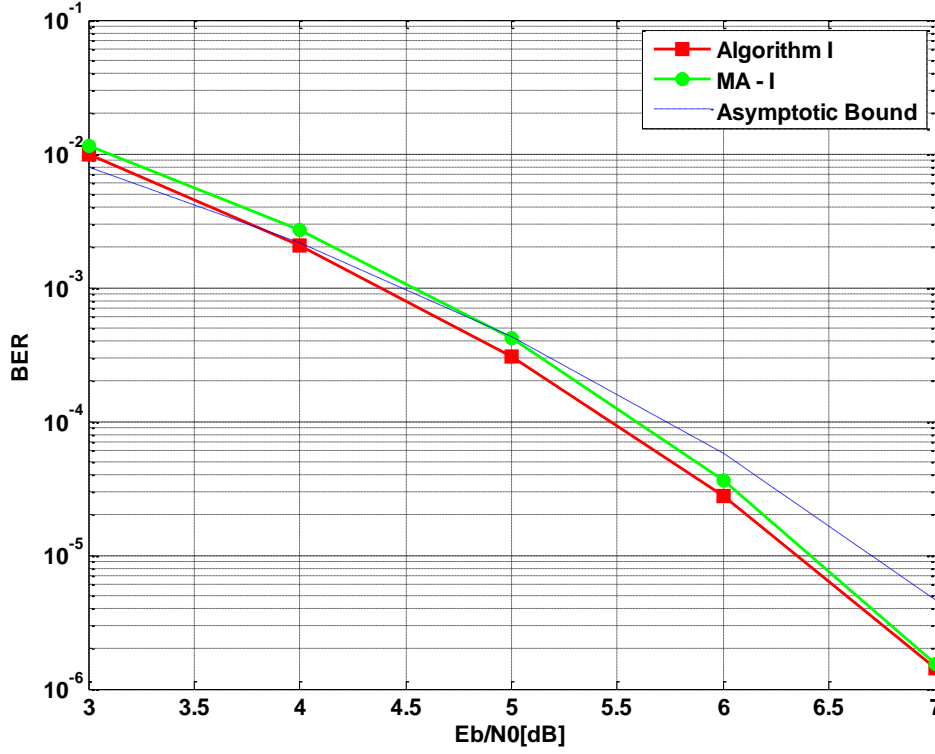


Figure 4.7: The performance of the  $(63,49,3)_4$  2D-SPC-TP code that is defined over  $\mathbb{F}_4$  and is decoded by different iterative decoding algorithms after two iterations

In Figure 4.8, the performance of the  $(63,49,3)_4$  2D-SPC-TP code that is defined over  $\mathbb{Z}_4$  and is separately decoded by Algorithm I, algorithm MA- I and Algorithm II is presented. Similar to the 2D-SPC-TP codes defined over  $\mathbb{F}_q$ , for large SNRs, turbo decoders perform better than the maximum-likelihood decoder for decoding the 2D-SPC codes defined over  $\mathbb{Z}_q$ . Also it can be seen that the performance of the code under decoding Algorithm I and decoding Algorithm II, which both have optimal SISO component decoders, is the same and despite using suboptimal SISO component decoders in algorithm MA- I, the code performance under this decoding algorithm improves. However, as it is shown in Chapter 3, the computational complexity of algorithm MA- I which is almost the same as the computational complexity of Algorithm I is much greater than the computational complexity of Algorithm II.

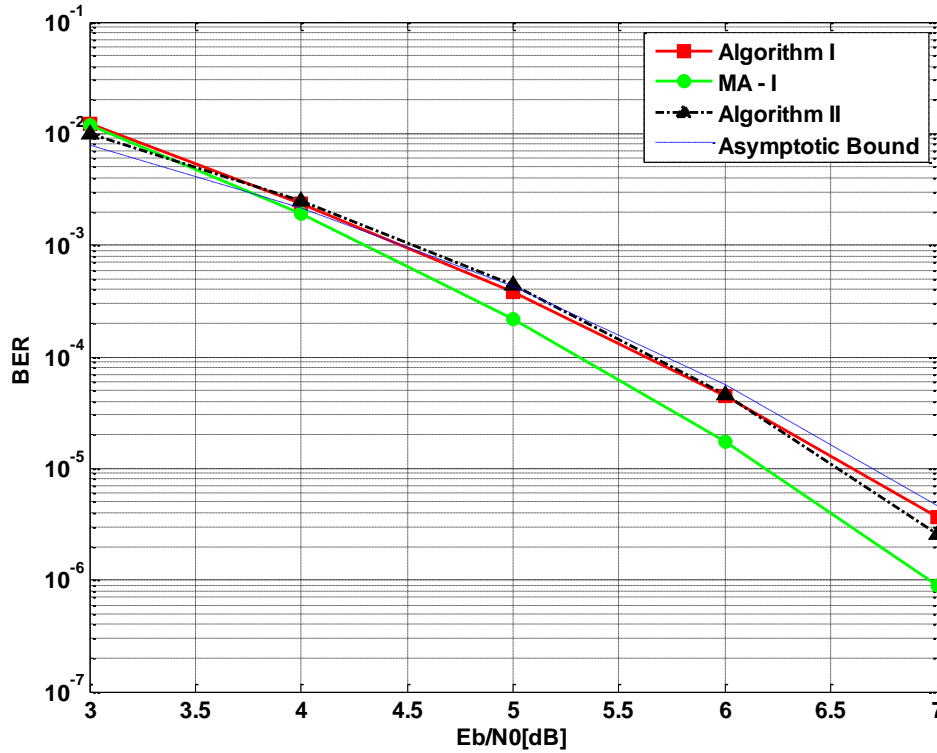


Figure 4.8: The performance of the  $(63,49,3)_4$  2D-SPC-TP code that is defined over  $\mathbb{Z}_4$  and is decoded by different iterative decoding algorithms after two iterations

In Figures 4.7 and 4.8, the performance of the  $(63,49,3)_4$  2D-SPC-TP under different iterative decoding algorithms is compared with the maximum-likelihood decoding asymptotic bound for the  $(63,49,3)_4$  2D-SPC product code. However, the comparison between the simulation results and the asymptotic performance bound for the maximum-likelihood decoding of a 2D-SPC product code shows that the performance of a 2D-SPC product code under iterative decoding algorithm does not approach the asymptotic bound, yet it is improved by iterative decoding algorithm. This can be explained from the fact that after the first iteration, the parity-check equations are statistically dependent and therefore, the summation of a symbol's LLR values over the two dimensions of the 2D-SPC codes cannot be considered as the maximum-likelihood value for that symbol. This is shown in Figure 4.9, by comparing the performance of the  $(63,49,3)_4$  2D-SPC-TP code that is defined

over  $\mathbb{Z}_4$  and is decoded by MA-I iterative decoding algorithm. As it is seen, the performance of the code which is decoded with only one iteration, approaches the maximum-likelihood asymptotic bound, while the performance of the code, which is decoded with two iterations diverges from the asymptotic bound.

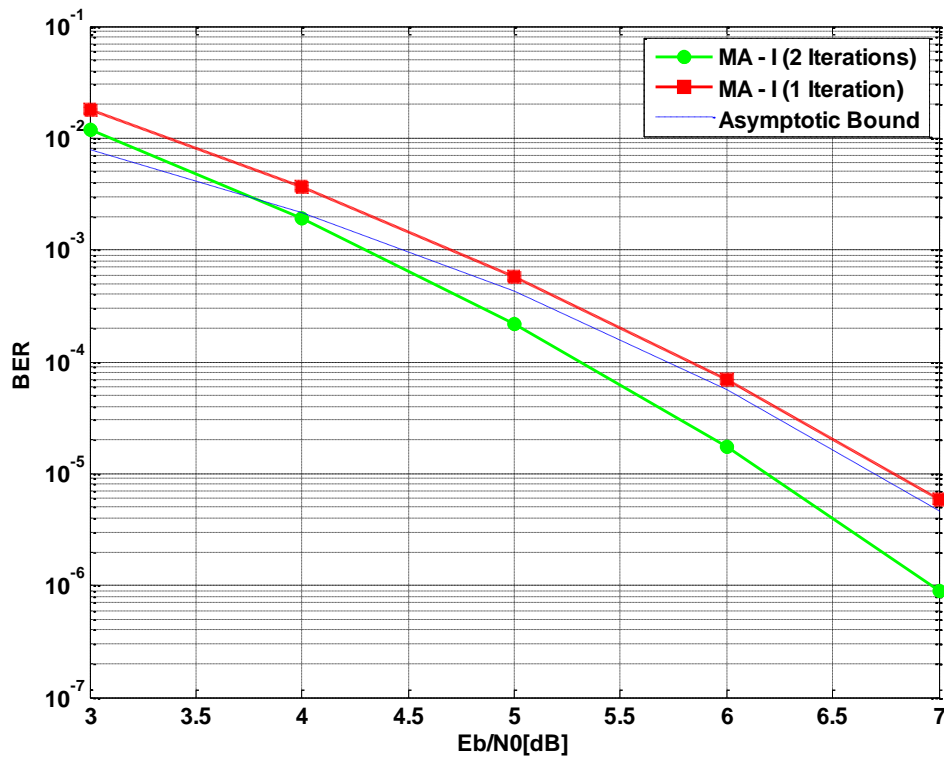


Figure 4.9: The performance of the  $(63,49,3)_4$  2D-SPC-TP code that is defined over  $\mathbb{Z}_4$  and is decoded by algorithm MA-I with different number of iterations

The performance of the  $(63,49,3)_q$  2D-SPC-TP code that is defined over different field orders is presented in Figure 4.10. The results are compared with the asymptotic bound of the maximum-likelihood decoding algorithm for  $(63,49,3)_q$  2D-SPC product code. These codes are decoded by algorithm MA-I after two iterations. As it is observed, the performance of the  $(63,49,3)_q$  2D-SPC-TP codes defined over  $\mathbb{F}_q$  is almost the same for different field orders. This is because the addition over  $\mathbb{F}_{2^p}$  is a bitwise operation and a 2D-SPC-TP code defined over  $\mathbb{F}_{2^p}$  performs like a  $p$ -fold binary 2D-SPC-TP code, where each binary code is



separately implemented over an AWGN channel with double-sided noise power spectral density of  $\frac{N_0}{2}$ . However, the simulation result shows different outcome for the  $(63,49,3)_q$  2D-SPC-TP codes defined over  $\mathbb{Z}_q$ .

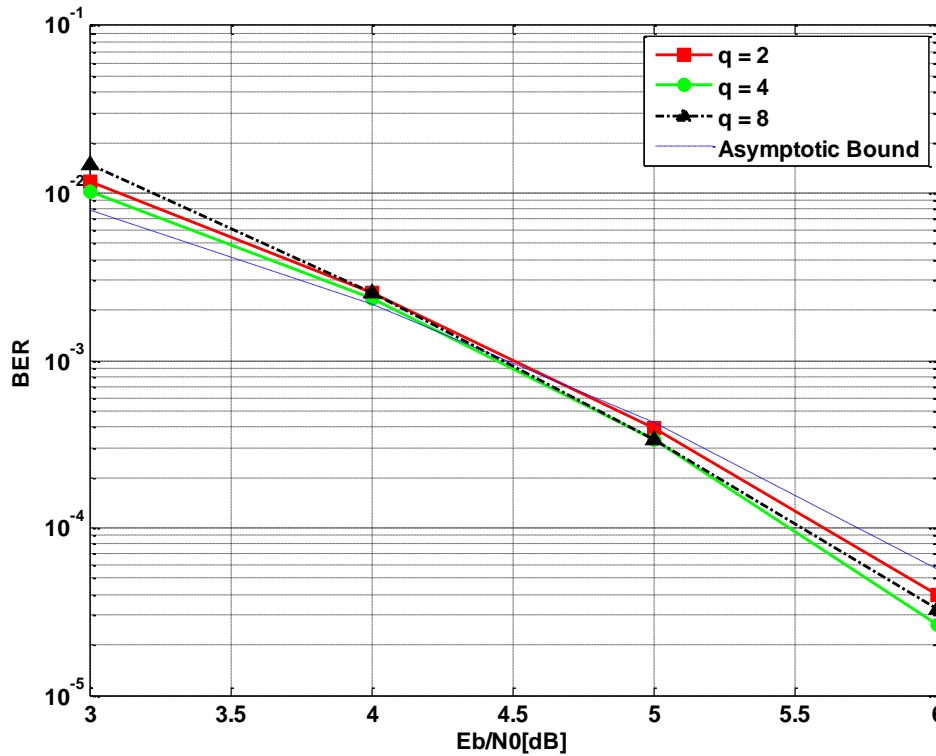


Figure 4.10: The performance of the  $(63,49,3)_q$  2D-SPC-TP code that is defined over  $\mathbb{F}_q$  and is decoded by algorithm MA-I after two iterations

The performance of the  $(63,49,3)_q$  2D-SPC-TP code defined over different ring of integer modulo- $q$  is presented in Figure 4.11 and the performance is compared with the asymptotic bound for the maximum-likelihood decoding algorithm for  $(63,49,3)_q$  2D-SPC product code. These codes are decoded by algorithm MA-I after two iterations. It can be observed that unlike the  $(63,49,3)_q$  2D-SPC-TP codes defined over finite fields, the performance of the  $(63,49,3)_q$  2D-SPC-TP codes defined over ring of integer modulo- $q$

improves over higher order rings. This can be explained by the minimum distance property of the 2D-SPC product codes that is presented in Chapter 5.

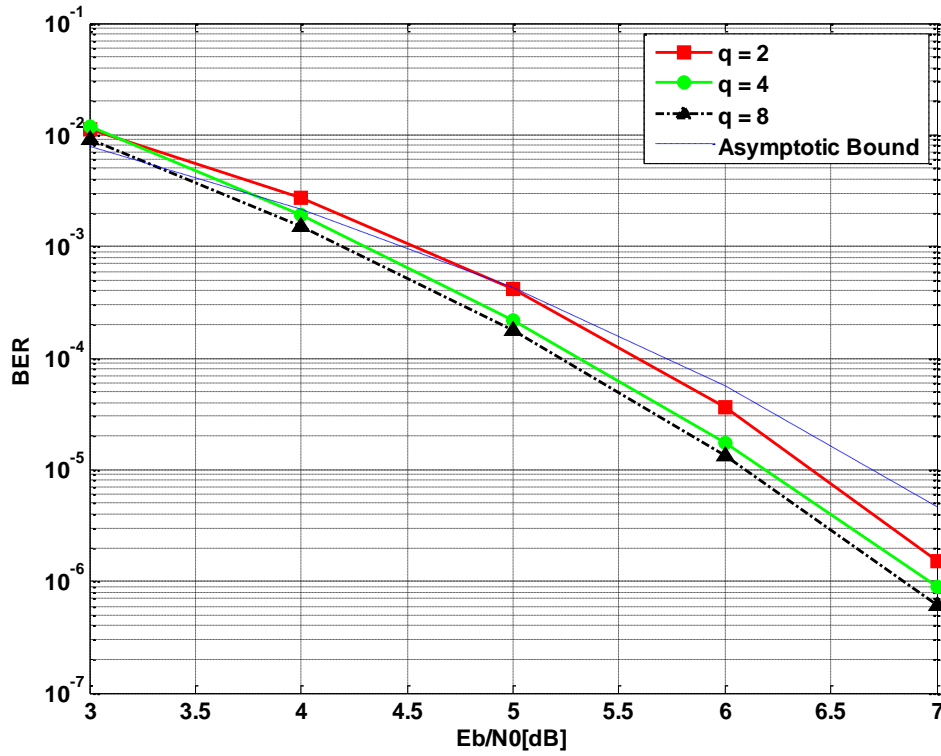


Figure 4.11: The performance of the  $(63,49,3)_q$  2D-SPC-TP code that is defined over  $\mathbb{Z}_q$  and is decoded by algorithm MA-I after two iterations

In Figure 4.12, the performance of the  $(63,49,3)_8$  2D-SPC-TP code defined over  $\mathbb{F}_8$  is compared with the performance of the  $(63,49,3)_8$  2D-SPC-TP code that is defined over  $\mathbb{Z}_8$ . Both codes are decoded by algorithm MA-I after two iterations. As it is observed, the 2D-SPC-TP code defined over  $\mathbb{Z}_8$  performs better than the same 2D-SPC-TP code that is defined over  $\mathbb{F}_8$ . Therefore, the 2D-SPC-TP codes over  $\mathbb{Z}_q$  are not just better choices from the decoding perspective but also their performance is better than the same codes defined over  $\mathbb{F}_q$ . The performance of both codes is compared with the asymptotic bound for the

maximum-likelihood decoding algorithm. It is shown in Chapter 5 that the  $(63,49,3)_q$  code defined over  $\mathbb{Z}_q$  and the  $(63,49,3)_q$  code defined over  $\mathbb{F}_q$  have the same asymptotic bound for the maximum-likelihood decoding algorithm.

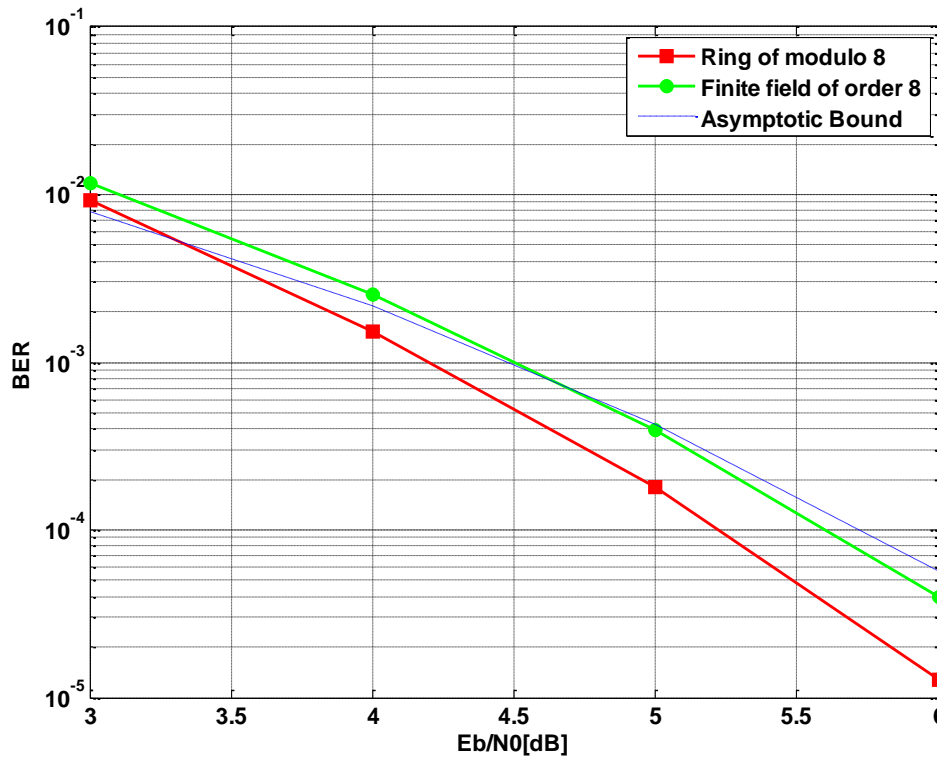


Figure 4.12: Comparison between the performance of the  $(63,49,3)_8$  2D-SPC-TP code defined over  $\mathbb{Z}_8$  and  $\mathbb{F}_8$ . The codes are decoded by algorithm MA-I after two iterations

It is known from (4.5) that by increasing the length of the constituent SPC codes, the rate of 2D-SPC-TP code becomes closer to one. In Table 4.1, the parameters for three different 2D-SPC-TP codes are given and in Figure 4.13, the performance of these codes is compared with each other. All the 2D-SPC-TP codes are defined over  $\mathbb{Z}_4$  and are decoded by algorithm MA-I after two iterations.

Table 4.1  
Code parameters for different rate 2D-SPC-TP codes

SPC Constituent Code	Resulted 2D-SPC-TP Code	Code Rate
(4,3)	$(15,9,3)_4$	0.6
(5,4)	$(24,16,3)_4$	0.67
(8,7)	$(63,49,3)_4$	0.78

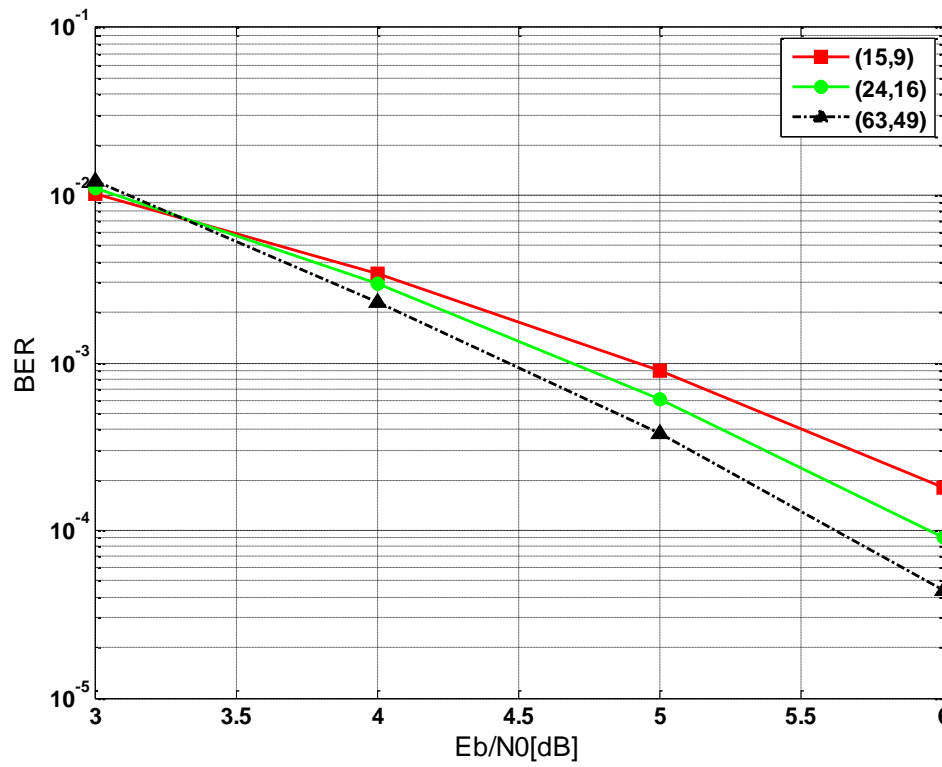


Figure 4.13: Comparison between the performance of different rate 2D-SPC-TP codes defined over  $\mathbb{Z}_4$  and decoded by algorithm MA-I after two iterations

As it is seen, by increasing the length of the 2D-SPC-TP code and subsequently increasing its coding rate, the performance of the code is improved. This can be explained with the maximum-likelihood decoding asymptotic bound for the 2D-SPC product code, which improves with increasing the code rate. Furthermore, the simulation results show that compared with maximum-likelihood asymptotic bound, the performance of a 2D-SPC product code improves under iterative decoding algorithm. Therefore, it can be concluded that the performance of a non-binary 2D-SPC-TP code improves by increasing its code rate and the non-binary 2D-SPC-TP codes have the potential to be used for the applications, which high-coding rate non-binary codes are required.

#### 4.5. Conclusion

In this chapter, we studied the two-dimensional, non-binary SPC turbo product (2D-SPC-TP) codes. A turbo product code is a product code that is decoded by a SISO iterative decoding algorithm. For constructing the 2D-SPC-TP codes, we considered the non-binary SPC constituent codes that are defined over,  $\mathbb{Z}_q$ , as well as the non-binary SPC constituent codes that are defined over  $\mathbb{F}_q$ .

Furthermore, three different iterative decoding algorithms for decoding the 2D-SPC-TP codes were presented and the performance of the codes over an AWGN channel was studied. It was shown that after the first iteration, the iterative decoders cannot be considered as a maximum-likelihood decoder, however, the comparison between the simulation result and the asymptotic performance bound for the maximum-likelihood decoding of a 2D-SPC product code shows that the performance of a 2D-SPC product code is improved by iterative decoding algorithm. The three proposed iterative decoding algorithms differ in the way that the SISO component decoders are realized. In Algorithm I, the SISO component decoders are constructed based on direct implementation of the optimal APP algorithm. In Modified-Algorithm I (MA-I), the SISO component decoders are constructed based on the sub-optimal max-log-APP decoding algorithm and finally, the SISO component decoders for Algorithm II, are constructed from the optimal APP decoding algorithm based on Fourier transform. Both Algorithm I and algorithm MA-I can be used

for any 2D-SPC-TP codes, while the Algorithm II can be implemented only for the 2D-SPC-TP codes defined over  $\mathbb{Z}_q$ .

The simulation results showed that regardless of the field's order, the performance of the 2D-SPC-TP codes defined over  $\mathbb{F}_{2^p}$  remains the same for different field orders, yet the performance of the 2D-SPC-TP codes defined over  $\mathbb{Z}_{2^p}$  improves over higher order rings. Moreover, it was shown that the non-binary 2D-SPC-TP codes have the potential to be used for the applications, where high-coding rate non-binary codes are required.

## **CHAPTER 5**

### **PERFORMANCE ANALYSIS**

#### **5.1. Introduction**

In this chapter, the performance analysis for single parity-check (SPC) codes and two-dimensional SPC (2D-SPC) product codes is discussed. The performance of a code is measured by the probability that a received sequence is decoded to a codeword different from the transmitted codeword. Therefore, the code performance is related to the distance property between codewords. In a linear block code, it can be shown that all codewords have the same set of distances from each other [85]. Therefore, the distance property between any selected codeword and the rest of codewords of a linear block code is the same as the distance property between the all-zero codeword and other codewords of that code.

This chapter is organized as follows. In Section 5.2 the performance bounds for SPC codes and 2D-SPC product codes are presented. We then in Section 5.3 study the minimum distance property of non-binary SPC codes and non-binary 2D-SPC product codes. The final section concludes this chapter.

## 5.2. Performance bound

For calculating the performance bound of a non-binary linear block code, we can assume that an error occurs when the all-zero codeword,  $\mathbf{0}$ , is transmitted and at the receiver, it is decoded to a non-zero codeword  $\hat{\mathbf{v}}_m$ . This probability is shown with  $P_{\mathbf{0} \rightarrow \hat{\mathbf{v}}_m}$ . From the union bound, we have

$$P_e \leq \sum_{\substack{\hat{\mathbf{v}}_m \in C \\ \hat{\mathbf{v}}_m \neq \mathbf{0}}} P_{\mathbf{0} \rightarrow \hat{\mathbf{v}}_m} \quad (5.1)$$

where  $P_e$  is the symbol-error rate. In [85], the theoretical upper bound for the symbol-error probability of a code, when the codewords are transmitted over a binary input memoryless channel and the received sequences are coherently detected and decoded by a maximum-likelihood decoding algorithm is given as

$$P_e \leq \sum_{d=d_{min}}^N \sum_{l=1}^K \frac{l}{K} A_{l,d} D\left(\frac{R\mathcal{E}_b}{N_0}, d\right) \quad (5.2)$$

where

$N$  is the code length.

$K$  is the code dimension.

$d_{min}$  is the code minimum Hamming distance.

$A_{l,d}$  is the number of codewords with output weight  $d$  that are associated with an input sequence of weight  $l$ .



$R$  is the code rate.

$\frac{\mathcal{E}_b}{N_0}$  is the signal to noise ratio per bit.

$D(\cdot)$  is the pairwise symbol-error probability function.

In [85], the pairwise bit-error probability function for two binary codewords that are different in  $h$  positions is given as (5.3). It is assumed that the codewords are modulated by binary phase-shift keying (BPSK) modulation scheme and are transmitted over an additive white Gaussian noise (AWGN) channel with double-sided noise power spectral density of  $\frac{N_0}{2}$ .

$$D\left(\frac{R\mathcal{E}_b}{N_0}, h\right) = Q\left(\sqrt{\frac{2Rh\mathcal{E}_b}{N_0}}\right) \quad (5.3)$$

SPC codes and 2D-SPC product codes are linear block codes and their performance bounds can be calculated from (5.2). In this research, we consider SPC codes and 2D-SPC product codes that are defined either over finite field of order  $q, \mathbb{F}_q$ , or over finite ring of integer modulo- $q, \mathbb{Z}_q$ . However, we do not make any use of non-binary modulation techniques. Therefore, the codewords' symbols are mapped to binary sequences and subsequently are modulated by BPSK modulation scheme. The codewords are then transmitted over a binary input AWGN channel with double-sided noise power spectral density of  $\frac{N_0}{2}$ . Therefore, under these conditions, the corresponding pairwise bit-error probability function,  $D(\cdot)$ , for the all-zero codeword and an arbitrary codeword  $\mathbf{v} = (v_0, v_1, \dots, v_{N-1})$  that has  $h$  non-zero elements is equal to

$$D\left(\frac{RE_b}{N_0}, h\right) = Q\left(\sqrt{\frac{2RE_b}{N_0} \sum_{i=1}^h \omega_H(\bar{\mathbf{v}}_i)}\right) \quad (5.4)$$

where  $\omega_H(\cdot)$  represents the Hamming weight of a binary sequence and  $\bar{\mathbf{v}}_i$  denotes the binary sequence corresponded to the symbol  $v_i$ . By substituting (5.4) in (5.2) we have

$$P_b \leq \sum_{d=d_{\min}}^N \sum_{l=1}^K \frac{\omega_H(\bar{\mathbf{v}}_l)}{qK} A_{l,d} Q\left(\sqrt{\frac{2RE_b}{N_0} \sum_{i=1}^d \omega_H(\bar{\mathbf{v}}_i)}\right) \quad (5.5)$$

From (5.5) the upper bound for the bit-error rate (BER) of non-binary SPC codes and non-binary 2D-SPC product codes can be calculated. Let  $\mathcal{C}$  be a non-binary  $(K+1, K)$  SPC code that is either defined over finite field of order  $q, \mathbb{F}_q$ , or defined over finite ring of integer modulo- $q, \mathbb{Z}_q$ . The minimum distance of code  $\mathcal{C}$  is two and it occurs in the codewords with only one non-zero information symbol and subsequently, one non-zero parity-check symbol. Therefore, we have

$$P_b \leq \sum_{d=2}^{K+1} \sum_{l=1}^{d-1} \frac{\omega_H(\bar{\mathbf{v}}_l)}{qK} A_{d-1,d} Q\left(\sqrt{2\left(\frac{K}{K+1}\right) \frac{E_b}{N_0} \sum_{i=1}^d \omega_H(\bar{\mathbf{v}}_i)}\right) \quad (5.6)$$

For the large signal-to-noise-ratios (SNR), the term corresponding to the minimum distance becomes the dominating term and therefore,

$$\begin{aligned}
P_b &\leq \frac{\omega_H(\bar{\mathbf{v}}_1)}{qK} \binom{K+1}{2} Q \left( \sqrt{2 \left( \frac{K}{K+1} \right) \frac{\mathcal{E}_b}{N_0} (\omega_H(\bar{\mathbf{v}}_1) + \omega_H(\bar{\mathbf{v}}_2))} \right) \\
&\leq \frac{\omega_H(\bar{\mathbf{v}}_1)}{qK} \frac{K(K+1)}{2} Q \left( \sqrt{2 \left( \frac{K}{K+1} \right) \frac{\mathcal{E}_b}{N_0} (\omega_H(\bar{\mathbf{v}}_1) + \omega_H(\bar{\mathbf{v}}_2))} \right) \\
&\leq \frac{\omega_H(\bar{\mathbf{s}})(K+1)}{2q} Q \left( \sqrt{2 \left( \frac{K}{K+1} \right) \frac{\mathcal{E}_b}{N_0} (\omega_H(\bar{\mathbf{s}}) + \omega_H(-\bar{\mathbf{s}}))} \right) \\
&\leq KQ \left( \sqrt{2 \left( \frac{K}{K+1} \right) \frac{\mathcal{E}_b}{N_0} \min_{s \in \{1,2,\dots,q-1\}} (\omega_H(\bar{\mathbf{s}}) + \omega_H(-\bar{\mathbf{s}}))} \right) \tag{5.7}
\end{aligned}$$

where, depending on the definition of the code,  $\oplus$  is the addition defined over  $\mathbb{F}_q$  or  $\mathbb{Z}_q$  then  $-\mathbf{s}$  is the additive inverse of  $\mathbf{s}$  such that  $-\mathbf{s} \oplus \mathbf{s} = 0$ . For non-binary SPC codes  $\min_{s \in \{1,2,\dots,q-1\}} (\omega_H(\bar{\mathbf{s}}) + \omega_H(-\bar{\mathbf{s}})) = 2$ ; for example for an SPC code defined over  $\mathbb{F}_{2^p}$  ( $p$  is any positive integer) this may occur for  $s = 1$  and for an SPC code defined over  $\mathbb{Z}_{2^p}$  this may occur for  $s = \frac{q}{2}$ . Therefore, the BER asymptotic bound for non-binary SPC codes can be calculated as

$$P_b \approx KQ \left( \sqrt{4 \left( \frac{K}{K+1} \right) \frac{\mathcal{E}_b}{N_0}} \right) \tag{5.8}$$

For 2D-SPC product codes, the asymptotic bound can be calculated in a similar way. Let  $\mathcal{C}$  be a  $(K^2 + 2K, K^2, 3)_q$  2D-SPC product codes that is constructed from the same length non-binary  $(K+1, K)$  SPC codes and the check on check symbol is punctured.  $\mathcal{C}$  is either defined over  $\mathbb{F}_q$  or defined over  $\mathbb{Z}_q$  and the minimum Hamming distance for  $\mathcal{C}$  is 3. Therefore, from (5.5) we have

$$P_b \leq \sum_{d=3}^{K^2+2K} \sum_{l=1}^{d-2} \frac{\omega_H(\bar{\mathbf{v}}_l)}{qK} A_{d-2,d} Q \left( \sqrt{2 \left( \frac{K^2}{K^2+2K} \right) \frac{\mathcal{E}_b}{N_0} \sum_{i=1}^d \omega_H(\bar{\mathbf{v}}_i)} \right) \quad (5.9)$$

and for the large SNRs we have

$$\begin{aligned} P_b &\leq \frac{\omega_H(\bar{\mathbf{v}}_l)}{qK} K^2 Q \left( \sqrt{2 \left( \frac{K^2}{K^2+2K} \right) \frac{\mathcal{E}_b}{N_0} (\omega_H(\bar{\mathbf{s}}) + \omega_H(\overline{-\mathbf{s}}) + \omega_H(\overline{-\mathbf{s}}))} \right) \\ &\leq KQ \left( \sqrt{2 \left( \frac{K^2}{K^2+2K} \right) \frac{\mathcal{E}_b}{N_0} \min_{s \in \{1,2,\dots,q-1\}} (\omega_H(\bar{\mathbf{s}}) + \omega_H(\overline{-\mathbf{s}}) + \omega_H(\overline{-\mathbf{s}}))} \right) \end{aligned} \quad (5.10)$$

where  $\min_{s \in \{1,2,\dots,q-1\}} (\omega_H(\bar{\mathbf{s}}) + \omega_H(\overline{-\mathbf{s}}) + \omega_H(\overline{-\mathbf{s}})) = 3$ . Therefore, the BER asymptotic bound for a 2D-SPC product code that is constructed from the same length  $(K+1, K)$  SPC codes and its check on check symbol is punctured is calculated as

$$P_b \approx KQ \left( \sqrt{6 \left( \frac{K^2}{K^2+2K} \right) \frac{\mathcal{E}_b}{N_0}} \right) \quad (5.11)$$

### 5.3. Minimum distance property

As previously mentioned, the performance of a code is related to its distance property. In this section, we study the minimum distance property of non-binary SPC codes and non-

binary 2D-SPC product codes. Since in this research we do not make any use of non-binary modulation techniques, the minimum distances are considered for the codes where the symbols are replaced by corresponding binary sequences. Therefore, as it was discussed in the previous section, the number of non-zero bits in a minimum-weight SPC codeword is  $\omega_H(\bar{s}) + \omega_H(\overline{-s})$  and the number of non-zero bits in a minimum-weight 2D-SPC codeword is  $\omega_H(\bar{s}) + \omega_H(\overline{-s}) + \omega_H(\overline{-\bar{s}})$ , where  $s \in \{1, 2, \dots, q-1\}$ . The bit weight spectral for the minimum-weight codeword of non-binary SPC codes and the minimum-weight codeword of non-binary 2D-SPC product codes are depicted in Figures 5.1 and 5.2 respectively, which for each code, the bit weight spectral for the code when it is defined over  $\mathbb{F}_q$  is compared with the bit weight spectral for that code when it is defined over  $\mathbb{Z}_q$ .

As it is seen, by defining the codes over  $\mathbb{Z}_q$ , the weight spectral becomes denser in the middle part. This means that compared with the codes defined over  $\mathbb{F}_q$ , the lower-weight codewords of the codes defined over  $\mathbb{Z}_q$  are shifting towards higher-weight codewords. Similar phenomenon occurs in turbo codes and in the literature about turbo codes; it has been termed as spectral thinning.

The spectral thinning in non-binary SPC codes and non-binary 2D-SPC product codes has no effect on the minimum distance of the code, however, it reduces the multiplicities of the low-weight codewords. It is seen from Figures 5.1 and 5.2 that the spectral thinning becomes more vivid over higher order rings. Therefore, we expect that non-binary SPC codes and non-binary 2D-SPC product codes defined over  $\mathbb{Z}_q$  have better performance compared with the same codes defined over a finite field of order  $q$  and by increasing the order of the ring, the performance of the code improves. This is also confirmed by the simulation results that were presented in Chapter 3 and Chapter 4.

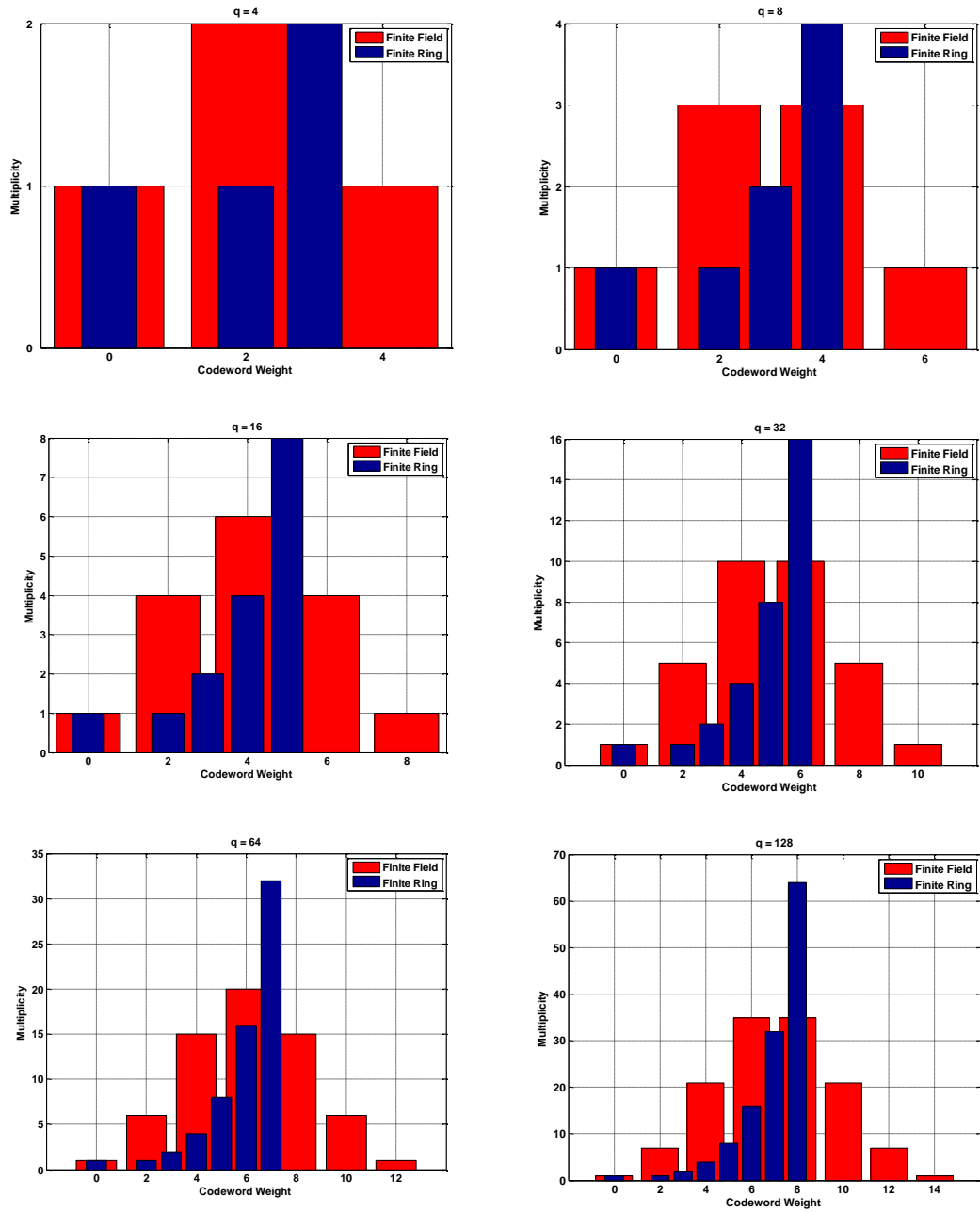


Figure 5.1: Comparison between the bit weight spectral for a minimum-weight codeword of an SPC code defined over  $\mathbb{F}_q$  and  $\mathbb{Z}_q$ .  $q = (4, 8, 16, 32, 64, 128)$

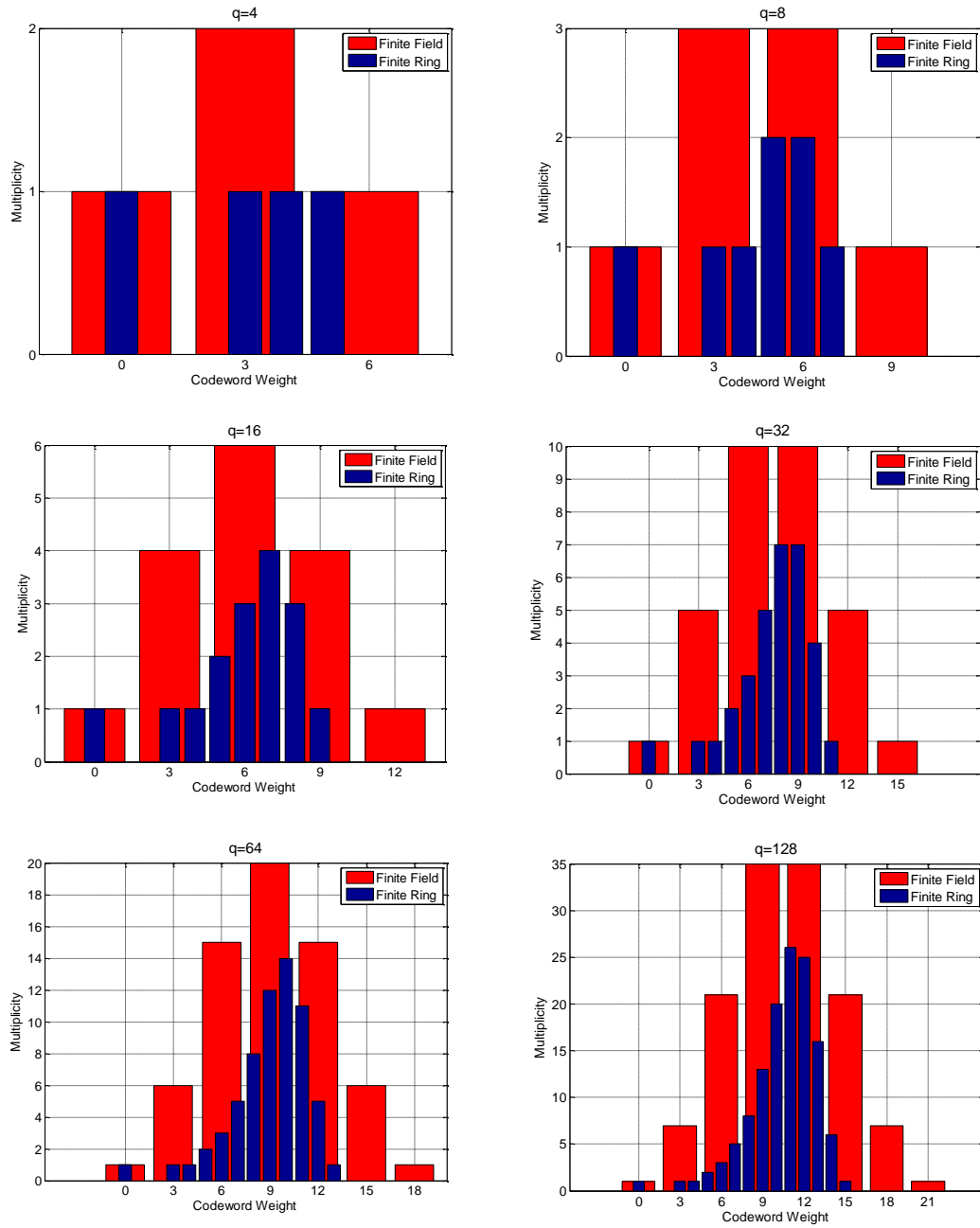


Figure 5.2: Comparison between the bit weight spectral for a minimum-weight codeword of a 2D-SPC code defined over  $\mathbb{F}_q$  and  $\mathbb{Z}_q$ .  $q = (4, 8, 16, 32, 64, 128)$

## 5.4. Conclusion

In this chapter, the performance analysis for SPC codes and 2D-SPC product codes is discussed. We showed that the performance of a code is related to the distance property between codewords. It is shown that the performance asymptotic bound for a non-binary SPC code defined over  $\mathbb{F}_q$  is the same as that for a non-binary SPC code defined over  $\mathbb{Z}_q$ . Moreover, it is shown that the performance asymptotic bound for a non-binary 2D-SPC code is also the same as that of the codes defined over  $\mathbb{F}_q$  and the codes defined over  $\mathbb{Z}_q$ . Furthermore, we studied the minimum distance property of non-binary SPC codes and non-binary 2D-SPC product codes and we showed that by defining the codes over  $\mathbb{Z}_q$ , the weight spectral becomes denser in the middle part and compared with the codes defined over  $\mathbb{F}_q$ , the lower-weight codewords of the codes defined over  $\mathbb{Z}_q$  are shifting towards higher-weight codewords. This phenomenon becomes more vivid over higher order rings and therefore, the performance improvement of the SPC and 2D-SPC product codes over higher order rings can be explained by this phenomenon.



## CHAPTER 6

### CONCLUSION

In this thesis a method for designing moderate to high coding rate, non-binary compound codes was proposed. These compound codes were constructed from a combination of non-binary single parity-check (SPC) codes and were iteratively decoded based on soft-input soft-output (SISO) decoding of their constituent codes.

The major forward error correcting codes used in the wireless communication systems were presented in Chapter 1. Turbo codes and turbo-like codes as the two important classes of capacity-approaching codes were briefly introduced and the reasons for their extraordinary performance were discussed. It became clear that low-complexity, high-rate coding schemes are essential for mobile communication systems and we showed that the existing good-performance low-density parity-check (LDPC) and turbo codes are mostly low-rate and, due to their long block size, are computationally complicated codes to be encoded and decoded. Therefore, designing small or medium block size good-performance, high-rate codes for mobile communication systems is required. Furthermore, different coding schemes were discussed, and it was explained that non-binary SPC concatenated codes with average-density parity-check matrices have the potential to be considered as the coding schemes for mobile communication systems. The motivation for this research was given and an overview of the thesis was presented. The original contributions of this thesis and the publications resulting from this research were listed.

In Chapter 2, the optimal *a posteriori* probability (APP) decoding algorithms were studied. Since turbo-like compound codes are decoded iteratively and the soft reliability information is repeatedly exchanged between their constituent codes, designing SISO decoding algorithms for decoding the constituent codes is required. An optimal SISO component decoder is constructed from an optimal APP decoder. However, the

computational complexity and the amount of memory requirement for calculating the symbol's APP values, based on standard ways, are often prohibitive in many practical applications. Therefore, we presented some modifications on the APP decoding algorithm to reduce its amount of memory requirement and computational complexity, and to make it a more feasible algorithm to be used for decoding non-binary codes. SPC codes are high-rate codes therefore we concentrated more on optimizing the optimal decoding algorithm for a high-rate code. We showed that by defining codes over finite ring of integer modulo- $q$ ,  $\mathbb{Z}_q$ , the amount of memory requirement for decoding a high-rate code could be minimized. Moreover, we used the concept of Fourier transform to reduce the computational complexity of the minimal-storage APP decoding algorithm for a high-rate code defined over  $\mathbb{Z}_q$ . Based on this algorithm, the discrete Fourier transform (DFT) vectors are employed to limit the repetitive calculations. Compared with the minimal-storage APP decoding algorithm for a high-rate code defined over  $\mathbb{Z}_q$ , the memory requirement of the DFT based APP decoding algorithm is slightly increased, but its computational complexity is reduced by the factor of  $q$ .

Despite the simplicity and weak performance of SPC codes, many good-performance codes are constructed from concatenation of binary SPC codes. These concatenated codes are decoded by SISO iterative decoding algorithms that repeatedly exchange the soft information between the SPC constituent codes. Non-binary codes can also be constructed from concatenation of non-binary SPC constituent codes. Therefore, implementing SISO decoding algorithms for decoding non-binary SPC codes is required. In Chapter 3, the structure and decoding of non-binary SPC codes were studied. We considered the non-binary SPC codes that are defined over finite field of order  $q$ ,  $\mathbb{F}_q$ , as well as the non-binary SPC codes that are defined over  $\mathbb{Z}_q$ . Moreover, two different optimum APP decoding algorithms for decoding non-binary SPC codes were presented and it was shown that the computational complexity and the amount of memory requirement for APP decoding of an SPC code defined over  $\mathbb{Z}_q$  is reasonably small. Therefore, non-binary SPC codes defined over  $\mathbb{Z}_q$  are good options to be used as constituent codes in concatenated structures.

In Chapter 4, the two-dimensional SPC (2D-SPC) product codes that are decoded by SISO iterative decoding algorithms were studied and we referred to them as the two-dimensional, non-binary SPC turbo product (2D-SPC-TP) codes. We considered the non-

binary SPC constituent codes that are defined over  $\mathbb{Z}_q$ , as well as the non-binary SPC constituent codes that are defined over  $\mathbb{F}_q$ . Three different iterative decoding algorithms for decoding the 2D-SPC-TP codes were presented and the performance of the codes over an AWGN channel was studied. The simulation result showed that the performance of the 2D-SPC product code is improved by iterative decoding. Moreover, it was shown that regardless of the field's order, the performance of the 2D-SPC-TP codes defined over  $\mathbb{F}_{2^p}$  remains the same, yet the performance of the 2D-SPC-TP codes defined over  $\mathbb{Z}_{2^p}$  improves over higher order rings. Furthermore, we showed that the performance of the 2D-SPC-TP codes defined over  $\mathbb{Z}_{2^p}$  improves with code-rate and therefore, the non-binary 2D-SPC-TP codes have the potential to be used for the applications, where high-coding rate non-binary codes are required.

In Chapter 5, the performance analysis for SPC codes and 2D-SPC product codes was discussed and we showed that the bit-error rate (BER) asymptotic bound for a non-binary SPC code defined over  $\mathbb{F}_q$  is the same as that for a non-binary SPC code defined over  $\mathbb{Z}_q$ . Moreover, it was shown that the BER asymptotic bound for a non-binary 2D-SPC code is also the same for the codes defined over  $\mathbb{F}_q$  and the codes defined over  $\mathbb{Z}_q$ . Furthermore, we studied the minimum distance property of non-binary SPC codes and non-binary 2D-SPC product codes and we showed that by defining the codes over  $\mathbb{Z}_q$ , the weight spectral becomes denser in the middle part, which compared with the codes defined over  $\mathbb{F}_q$ , the lower-weight codewords of the codes defined over  $\mathbb{Z}_q$  are shifting towards higher-weight codewords. This phenomenon becomes more vivid over higher order rings, which explains the performance improvement for the SPC and 2D-SPC product codes over higher order rings.

In general, two issues are important in designing compound codes. One is the codeword-weight-distribution of the code, and the other is the structure of the decoder. The multiplicity of the minimum weight codewords in a well-designed compound code needs to be reduced and the SISO iterative decoding algorithm needs to be simple so that the decoding latency can be avoided in the system. We showed that these two conditions could be more satisfied by using SPC constituent codes defined over  $\mathbb{Z}_q$  compared with using SPC constituent codes that are defined over  $\mathbb{F}_q$ .

## REFERENCES

- [1] C. E. Shannon, "A mathematical theory of communication," *Bell Syst. Tech. J.*, Vol. 27, pp. 379-423, 1948
- [2] R. W. Hamming, "Error detecting and error correcting codes," *Bell Syst. Tech. J.*, Vol. 29, pp.147-150, 1950.
- [3] D. J. Costello, Jr., J. Hagenauer, H. Imai, and S. B. Wicker, "Application of error-control coding," *IEEE Trans. Inf. Theory*, vol. 44, no. 10, pp. 2531-2560, Oct. 1998.
- [4] C. Berrou, A. Glavieux and P. Thitimajshima, "Near Shannon-limit error-correcting coding and decoding: turbo codes," in *Proc. 1993 IEEE Int. Commun. Conf.*, Geneva, Switzerland, May 1993, pp.1046-1070.
- [5] P.H. Siegel, D. Divsalar, E. Eleftheriou, J. Hagenauer and D. Rowitch, "The turbo principle: from theory to practice," *IEEE Journal on Selected Areas in Communications*, vol. 19, no. 5, May 2001.
- [6] G.D. Forney, Jr, "Concatenated codes," Cambridge, MA : MIT Press, 1966.
- [7] S. Benedetto and G. Montorsi, "Average performance of parallel concatenated block codes," *IET Electronics Letters*, vol. 31, no. 2, pp.156-158, Feb. 1995.
- [8] S. Benedetto, D. Divsalar, G. Montorsi, and F. Pollara, "Serial concatenation of interleaved codes: performance analysis, design and iterative decoding," *IEEE Trans. Inf. Theory*, vol. 44, no. 5, pp. 909-926, May 1998.
- [9] D. Divsalar and F. Pollara, "Serial and hybrid concatenated codes with applications," in *Proc. 1st Intl. Symp. Turbo Codes and related topics*. Sep. 1997.

- [10] R. G. Gallager, "Low-density parity-check codes," *IRE Trans. Inf. Theory*, vol. IT-8, pp. 21-28, Jan. 1962.
- [11] R. M. Tanner, "A recursive approach to low complexity codes," *IEEE Trans. Inf. Theory*, vol. IT-27, pp. 533-547, Sep. 1981.
- [12] D. J. C. MacKay and R. M. Neal, "Near Shannon limit performance of low-density parity-check codes," *Electronics Lett.*, vol. 32, no. 8, pp. 1645-1646, Aug. 1996.
- [13] D. J. C. MacKay, "Good error correcting codes based on very sparse matrices," *IEEE Trans. Inf. Theory*, vol. 45, no. 3, pp. 399-431, Mar. 1999.
- [14] M. G. Luby, M. Mitzenmacher, M. A. Shkrollahi and D. A. Spielman, "Improved low-density parity-check codes using irregular graphs," *IEEE Trans. Inf. Theory*, vol. 47, no. 2, pp. 585-598, Feb. 2001.
- [15] M. C. Davey and D. J. C. MacKay, "Low-density parity-check codes over  $GF(q)$ ," *Proc. IEEE Inform. Theory Workshop*, Jun 1998, pp. 70-71.
- [16] M. C. Davey and D. MacKay, "Low-density parity-check codes over  $GF(q)$ ," *IEEE Commun. Lett.*, vol. 2, no. 6, pp. 165-167, Jun 1998.
- [17] T. Richardson and R. Urbanke, "The capacity of low-density parity-check codes under message-passing decoding," *IEEE trans. Inf. Theory*, vol. 47, no. 2, pp. 599-618, Feb. 2001.
- [18] T. Richardson, A. Shokrollahi, and R. Urbanke, "Design of capacity-approaching low-density parity check codes," *IEEE trans. Inf. Theory*, vol. 47, no. 2, pp. 619-637, Feb. 2001.
- [19] S. Y. Chung, G. D. Forney, T. 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, no. 2, pp. 58-60, Feb. 2001.
- [20] X. Y. Hu, E. Eleftheriou, and D. M. Arnold, "Regular and irregular progressive edge-growth Tanner graphs," *IEEE Trans. Inform. Theory*, vol. 51, no. 1, pp. 386-398, Jan. 2005.

- [21] Y. Kou, S. Lin, and M. P. C. Fossorier, "Low-density parity-check codes based on finite geometries: A rediscovery and new results," *IEEE Trans. Inf. Theory*, vol. 47, no. 11, pp. 2711-2736, Nov. 2001.
- [22] S. J. Johson and S. R. Weller, "A family of irregular LDPC codes with low encoding complexity," *IEEE Commun. Lett.*, vol. 7, pp. 79-81, Feb. 2003.
- [23] I. Djurdjevic, J. Xu, K. Abdel-Ghaffar, and S. Lin, "Construction of low-density parity-check code based on Reed-Solomon code with two information symbols," *IEEE Commun. Lett.*, vol. 7, pp. 317-319, Jul. 2003.
- [24] 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 Trans. Inf. Theory*, vol. 50, no. 6, pp. 1269-1279, Jun. 2004.
- [25] B. Vasic and O. Milenkovic, "Combinatorial construction of low-density parity-check codes for iterative decoding," *IEEE Trans. Inf. Theory*, vol. 50, no. 6, pp. 1156-1176, Jun. 2004.
- [26] B. Ammar, B. Honary, Y. Kou, J. Xu, and S. Lin, "Construction of low-density parity-check codes based on balanced incomplete block designs," *IEEE Trans. Inf. Theory*, vol. 50, no. 6, pp. 1257-1268, Jun. 2004.
- [27] N. Miladovic and M. Fossorier, "Systematic recursive construction of LDPC codes," *IEEE Commun. Lett.*, vol. 8, pp. 302-304, May 2004.
- [28] J. Xu, L. Chen, I. Djurdjevic, S. Lin and K. Abdel-Ghaffar, "Construction of regular and irregular LDPC codes: geometry decomposition and masking," *IEEE Trans. Inf. Theory*, vol. 53, no. 1, pp. 121-134, Jan. 2007
- [29] L. Chen, J. Xu, I. Djurdjevic, and S. Lin, "Near-Shannon quasi-cyclic low-density parity-check codes," *IEEE Trans. Commun.*, vol. 52, pp. 1038-1042, Jul. 2004.
- [30] Z. Li, L. Chen, L. Zeng, S. Lin and W. H. Fong, "Efficient encoding of quasi-cyclic low-density parity-check codes," *IEEE Trans. Commun.*, vol. 54, pp. 71-81, Jan. 2006.

- [31] L. Lan, L. Zeng, Y. Y. Tai, L. Chen, S. Lin and K. Abdel-Ghaffar, "Construction of quasi-cyclic LDPC codes for AWGN and binary erasure channels: a finite field approach," *IEEE Trans. Inf. Theory*, vol. 53, no. 7, pp. 2429-2458, Jul. 2007.
- [32] Y. Y. Tai, L. Lan, Li. Zeng, S. Lin and K. Abdel-Ghaffar, "Algebraic construction of quasi-cyclic LDPC codes for the AWGN and erasure channels," *IEEE Trans. Commun.*, vol. 54, no. 10, pp. 1765-1774, Oct. 2006.
- [33] L. Zhang, Q. Huang, S. Lin and K. Abdel-Ghaffar, I.F. Blake, "Quasi-cyclic LDPC codes: an algebraic construction, rank analysis, and codes on Latin squares," *IEEE Trans. Commun.*, vol. 58, no. 11, pp. 3126-3139, Nov. 2010.
- [34] H. Chun-Ming, H. Jen-Fa and Y. Chao-Chin, "Construction of quasi-cyclic LDPC codes from quadratic congruencies," *IEEE Commun. Lett.*, vol. 12, no. 4, pp. 313-315, Apr. 2008.
- [35] J. Kang, Q. Huang, L. Zhang, B. Zhou and S. Lin, "Quasi-cyclic LDPC codes: an algebraic construction," *IEEE Trans. Commun.*, vol. 58, no. 5 pp. 1383-1396, May 2010.
- [36] C. Chen, B. Bai and X. Wang, "Construction of nonbinary quasi-cyclic LDPC cycle codes based on singel perfect difference set," *IEEE Commun. Lett.*, vol. 14, no. 2, pp. 181-183, Feb. 2010.
- [37] B. Zhou, J. Kang, Y. Y. Tai, S. Lin, and Z. Ding, "High performance non-binary quasi-cyclic LDPC codes on Euclidean geometries," *IEEE Trans. Commun.*, vol. 57, no. 5, pp. 1298-1311, May 2009.
- [38] L. Zeng, L. Lan; Y. Y. Tai, B. Zhou, S. Lin and K. Abdel-Ghaffar, "Construction of nonbinary cyclic, quasi-cyclic and regular LDPC codes: a finite geometry approach," *IEEE Trans. Commun.*, vol. 56, no. 3 pp. 378-387, Mar. 2008.
- [39] B. Zhou, J. Kang, S. Song, S. Lin, K. Abdel-Ghaffar and M. Xu, "Construction of non-binary quasi-cyclic LDPC codes by arrays and array dispersions," *IEEE Trans. Commun.*, vol. 57, no. 6 pp. 1952-1662, Jun 2009.

- [40] X. Y. Hu and E. Eleftheriou, "Binary representation of cycle Tanner-graph  $GF(2^q)$  codes," *The Proc. IEEE Intern. Conf. on Commun.*, Paris, France, pp. 528-532, Jun 2004.
- [41] S.Y. Chung, T. Richardson and R. Urbanke, "Analysis of sum-product decoding of low-density parity-check codes using a Gaussian approximation," *IEEE Trans. Inf. Theory*, vol. 47, no. 2, pp. 657-670, Feb. 2001.
- [42] G. Li, I.J. Fair, and W.A. Krzymien, "Density evolution for nonbinary LDPC codes under Gaussian approximation," *IEEE Trans. Inf. Theory*, vol. 55, no. 3, Mar. 2009.
- [43] A. Ashikhmin, G. Kramer and S. T. Brink, "Extrinsic Information transfer functions: model and erasure channel properties," *IEEE Trans. Inf. Theory*, vol. 50, no. 11, Nov. 2004.
- [44] G. J. Byers and F. Takawira, "Exit charts for non-binary LDPC codes," in *Proc. Int. Conf. Communication (ICC'05)*, Seoul, Korea, pp. 652-657, May 2005.
- [45] V. Rathi and R. Urbanke, "Density evolution, thresholds and the stability condition for non-binary LDPC codes," *IEE Proceedings*, vol. 152, no. 6, pp. 1069-1074, Dec. 2005.
- [46] A. Bennatan and D. Burshtein, "Design and analysis of nonbinary LDPC codes for arbitrary discrete-memoryless channels," *IEEE Trans. on Inf. Theory*, vol. 52, no. 2, pp. 549-583, Feb. 2006.
- [47] J. Pearl, "Probabilistic reasoning in intelligent systems: network of plausible inference," Morgan Kauffmann Publishers, 1988.
- [48] F. R. Kschischang, B. J. Frey and H. A. Loeliger, "Factor graphs and the sum-product algorithm," *IEEE Trans. Inf. Theory*, vol. 47, pp. 498-519, Feb. 2001.
- [49] M.P.C. Fossorier, M. Mihaljevic and H. Imai, "Reduced complexity iterative decoding of low-density parity-check codes based on belief propagation," *IEEE Trans. Commun.*, vol. 47, no. 5, pp. 673-680, May 1999.



- [50] J. Chen, A. Dholakia, E. Eleftheriou, M.P.C. Fossorier and X.Y. Hu, "Reduced-complexity decoding of LDPC codes," *IEEE Trans. Commun.*, vol. 53, no. 8, pp. 1288-1299, Aug. 2005.
- [51] D. J. C. MacKay and M. C. Davey, "Evaluation of Gallager codes of short block length and high rate applications," in *Proc. IMA International Conf. Mathematics its Applications: Codes, Systems Graphical Models*, pp. 113-130, Springer-Verlag, New York, 2000.
- [52] H. Song and J. R. Cruz, "Reduced-complexity decoding of Q-ary LDPC codes for magnetic recording," *IEEE Trans. Magn.*, vol. 39, no.3, pp. 1081-1087, Mar. 2003.
- [53] L. Barnault and D. Declercq, "Fast decoding algorithm for LDPC over  $GF(2^q)$ ," *IEEE Inf. Th. Workshop*, Paris, France, Mar. 2003.
- [54] D. Declercq and M. Fossorier, "Decoding algorithms for nonbinary LDPC codes over  $GF(q)$ ," *IEEE Trans. Commun.*, vol. 55, no. 4, pp. 633-643, Apr. 2007.
- [55] A. Voicila, D. Declercq, F. Verdier, M. Fossorier and P. Urad, "Low-complexity decoding for non-binary LDPC codes in high order fields," *IEEE Trans. Commun.*, vol. 58, pp. 1365-1375, no.5, May 2010.
- [56] ETSI EN 302 307 (V1.1.2), "Digital video broadcasting (DVB); second generation framing structure, channel coding and modulation systems for broadcasting, interactive Services, news gathering and other broadband satellite applications," *European Telecommunications Standards Institute (ETSI)*, Jun 2006.
- [57] 802.16E-2005&802.16/COR1 IEEE standard for local and metropolitan area networks part 16: air interface for fixed and mobile broadband wireless access systems amendment for physical and medium access control layers for combined fixed and mobile operation in licensed bands, 2/2006.
- [58] V. Oksman and S. Galli, "G.hn: the new ITU-T home networking standard," *IEEE Commun. Magazine*, vol. 47, no. 10, pp. 138-145. Oct. 2009.

- [59] IEEE standard for information technology-telecommunications and information exchange between systems-local and metropolitan area networks-specific requirements part 3: carrier sense multiple access with collision detection (CSMA/CD) access method and physical layer specifications, Sep. 2006, IEEE Std. 802.3an.
- [60] M. Lentmaier and K. Sh. Zigangirov, "On generalized low-density parity-check codes based on Hamming component codes", *IEEE Commun. Lett.*, vol. 3, no. 8, pp. 248-250, Aug. 1999.
- [61] J. Boutros, O. Pothier, and G. Zemor, "Generalized low density (Tanner) codes," in *Proc. IEEE ICC 99*, Houston, Texas, Jul 1999, pp. 441-445.
- [62] T. M. N. Ngatched and F. Takawira, "An ensemble of iteratively decodable codes constructed based on a superposition method," *IEEE Trans. Commun.*, vol. 54, no. 11, Nov. 2006.
- [63] P. Elias, "Error free coding," *IRE Trans. Inf. Theory*, vol. 4, pp.29-37, Sep. 1954.
- [64] G. Battail, "Building long codes by combination of simple ones, thanks to weighted-output decoding," in *Proc. URSIISSE 1989*, Erlangen, Germany, Sep. 1989, pp. 634-637.
- [65] J. Hagenauer, E. Offer and L. Papke, "Iterative decoding of binary block and convolutional code," *IEEE Trans. Inf. Theory*, vol. 42, no. 2, pp. 429-445, Mar. 1996
- [66] D. M. Rankin and T. A. Gulliver, "Single parity check product codes," *IEEE Trans on Commun*, vol. 49, pp.1354-1362, Aug. 2001.
- [67] H. Xu and F. Takawira, "A new structure of single parity check product codes," *SAIEE Africa Research Journal*, vol.97, No.2, pp.132-135, Jun 2006.
- [68] A. Shiozaki, M. Kishimoto and G. Maruoka, "Close-to-capacity performance of extended single parity check product codes," *Electronics Letters*, vol.47, pp. 34-35, Jan. 2011.

- [69] H. Burton and E. Weldon, Jr., "Cyclic product codes," *IEEE Trans on Inform Theory*, Vol. 11, pp. 433-439, Jul 1965.
- [70] J. Justesen, "Performance of product codes and related structures with iterated decoding," *IEEE Trans on Commun*, vol. 59, pp.407-415, Feb. 2011.
- [71] L. Bahl, J. Cocke, F. Jelinek, and J. Raviv, "Optimal decoding of linear codes for minimizing symbol error rate," *IEEE Trans. Inf. Theory*, vol. 20, no. 2, pp. 284-287, Mar. 1974.
- [72] R. Lucas, M. Bossert, and M. Breitbart, "On iterative soft decision decoding of linear binary block codes and product codes," *IEEE J. Select. Areas Commun*, vol. 16, No. 2, pp.276-298, Feb. 1998.
- [73] R. J. McEliece, "On the BCJR trellis for linear block codes", *IEEE Trans. Inf. Theory*, vol. 42, no. 4, pp. 1072-1092, Jul. 1996.
- [74] Y. Lin, S. Lin, and M.P.C. Fossorier, "MAP algorithms for decoding linear block codes based on sectionalized trellis diagrams," *IEEE Trans. Commun*, pp.577-587, vol. 18, no. 4, Apr. 2000.
- [75] T. Johansson and K. Zigangirov, "A Simple one sweep algorithm for optimal APP symbol decoding of linear block code," *IEEE Trans. Inf. Theory*, vol. 44, no. 7, pp. 3124-3129, Nov. 1998.
- [76] A. Trofimov and T. Johansson, "A memory-efficient optimal APP symbol-decoding algorithm for linear block codes", *IEEE Trans. Commun*, vol. 52, no. 9, pp.1429-1434, Sep. 2004.
- [77] I. Lee, "Modification of the MAP algorithm for memory savings," *IEEE Trans. Signal Processing*, vol. 53, no. 3, pp.1147-1150, Mar. 2005.
- [78] A. Worm, H. Michel, and N. Wehn, "Power minimization by optimizing data-transfers in turbo decoders," *Kleinbeubacher Berichte*, Band 43, pp. 343-350, Sep. 1999.

- [79] M. L. Vallejo, S. A. Mujtaba, and I. Lee, "A low-power architecture for maximum a posteriori turbo-decoding," in *proc. 36<sup>th</sup> Asilomar Conf.*, Nov. 2002, pp. 47-51.
- [80] D. Sridhara and T. E. Fuja, "LDPC codes over rings for PSK modulation," *IEEE Trans. Inf. Theory*, vol. 51, pp. 3209-3220, Sep. 2005.
- [81] A. C. Reid, T. A. Gulliver, and D. P. Taylor, "Rate-1/2 component codes for nonbinary turbo codes," *IEEE Trans on Commun*, vol. 53, pp. 1417-1422, Sep. 2005.
- [82] J. Berkmann, "On turbo decoding of nonbinary codes," *IEEE Commun. Lett.*, vol. 2, pp. 94-96, Apr. 1998.
- [83] J. Hagenauer and P. Hoeher, "A Viterbi algorithm with soft-decision outputs and its applications," in *Proc. Globecom*, Dallas, TX, Nov. 1989, pp.1680-1686.
- [84] C. R. Hartmann and L. D. Rudolph, "An optimum symbol-by-symbol decoding rule for linear codes," *IEEE Trans. Inform. Theory*, vol. 22, no. 5, pp. 514-517, Sep. 1976.
- [85] J. G. Proakis and M. Salehi, "Digital communication", 5<sup>th</sup> ed., McGraw Hill, 2008.
- [86] K. Gracie and M. Hamon, "Turbo and turbo-like codes: principles and applications in telecommunications," *Proceeding of the IEEE*, vol.95, pp. 1228-1254, Jun. 2007.
- [87] T. R. Oenning and J. Moon, "A low density generator matrix interpretation of parallel concatenated single bit parity codes," *IEEE Trans on Magnetism*, vol. 37, pp.737-741, Mar. 2001.
- [88] J. L. Krishna, R. Narayanan, E. Kurtas, and C. N. Georghiades, "On the performance of high-rate TPC/SPC codes and LDPC codes over partial response channels," *IEEE Trans on Commun*, vol. 50, pp.723-734, May 2002.
- [89] J. S. K. Tee, D. P. Taylor and P. A. Martin, "Multiple serial and parallel concatenated single parity-check codes," *IEEE Trans on Commun*, vol. 51, pp. 1666-1675, Oct. 2003.

- [90] J. A. Gallian, "Contemporary abstract algebra", 3<sup>rd</sup> ed., Lexington, MA: D. C. Heath and Company, 1994.
- [91] C. Berrou, M. Jezequel, C. Douillard and S. Kerouedan, "The advantages of non-binary turbo codes," in *Proc. Information Theory Workshop, ITW 2001*, pp. 61-63, 2001.
- [92] O. Aitsab and R. Pyndiah, "Performance of Reed-Solomon block turbo codes," in *Proc. IEEE Global Telecommun. Conf. 1996*, vol. 1-3, London, U.K., Nov., pp. 121-125, 1996.
- [93] P. Sweeney and S. Wesemeyer, "Iterative soft-decision decoding of linear codes," *Inst. Electr. Eng. Proc. Commun.*, vol. 147, no. 3, pp. 133-136, Jun. 2000.
- [94] R. Zhou, R. Le Bidan, Ramesh Pyndiah and A. Goalic, "Low-complexity high-rate Reed-Solomon block turbo codes," *IEEE Trans on Commun*, vol. 55, pp. 1656-1660, Sep. 2007.
- [95] G. S. White and D. J. Costello, Jr, "Construction and performance of q-ary turbo codes for use with M-ary modulation techniques," in *Proc. Conf. Inf. Sci. Syst.*, Mar. 1999.
- [96] M. B. Shoemake, C. Heefard and E. Rossin, "Turbo codes for high order constellation," in *Proc. Information Theory Workshop, ITW 1998*, Kilarney, Irland, 1998.
- [97] A. Ghrayeb and T. Abualrub, "Asympyoyic performance comparison of concatenated (turbo) codes over GF(4)," *Int. J. Commun. Syst.*, vol. 17, pp. 479-490, 2004.
- [98] A. Ruscitto and E. M. Biglieri, "Joint source and channel coding using turbo codes over rings," *IEEE Trans. Commun.*, vol. 46 , pp. 981-984, Aug. 1998.
- [99] M. Xiao and T.M. Aulin, "Serially concatenated continuous phase modulation with convolutional codes over rings," *IEEE Trans. Commun.*, vol. 54, pp. 1387-1396, Aug. 2006.

- [100] J. da Silva Barros and R. Baladini Filho, “Turbo codes with symmetric and asymmetric component codes defined over finite fields of integers,” *IET Commun.*, vol. 3, pp. 1800–1807, Apr. 2009.
- [101] M. Caldera and H. J. Zepernick, “APP decoding of nonbinary SPC product codes over discrete memoryless channels,” *10<sup>th</sup> International Conference on Telecommunication, ICT2003*, vol. 2, pp. 1167-1170, 2003.