

https://dspace.org/

Techniques to improve iterative decoding of linear block codes

Genga, Yuval Odhiambo

2019-10

https://hdl.handle.net/10539/29047

Genga, Yuval Odhiambo (2019) Techniques to improve iterative decoding of linear block codes, University of the Witwatersrand, Johannesburg, <http://hdl.handle.net/10539/29047>

Downloaded from a DSpace repository. https://dspace.org/

University of the Witwatersrand

Doctoral Thesis

Techniques to Improve Iterative Decoding of Linear Block Codes

Author:
Yuval Odhiambo Genga

A Thesis submitted in fulfilment of the requirements for the degree of Doctor of Philosophy

in the

Centre for Telecommunications Access and Services
School of Electrical and Information Engineering

October 2019



The financial assistance of the Center for Telecommunications Access and Services (CeTAS) towards this research is hereby acknowledged. Opinions expressed and conclusions arrived at, are those of the author and are not necessarily to be attributed to the CeTAS.

Declaration

I, Yuval Odhiambo Genga, declare that this Thesis titled, 'Techniques to Improve Iterative Decoding of Linear Block Codes" and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a research degree at this University.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this Thesis is entirely my own work.
- I have acknowledged all main sources of help.

Signed: _____

Date: _____

UNIVERSITY OF THE WITWATERSRAND

Abstract

Engineering and the Built Environment
School of Electrical and Information Engineering

Doctor of Philosophy

Techniques to Improve Iterative Decoding of Linear Block Codes

by Yuval Odhiambo Genga

Supervisor: Olutayo Oyerinde and Jaco Versfeld

In the field of forward error correction, the development of decoding algorithms with a high error correction performance and tolerable complexity has been of great interest for the reliable transmission of data through a noisy channel. The focus of the work done in this thesis is to exploit techniques used in forward error correction in the development of an iterative soft-decision decoding approach that yields a high performance in terms of error correction and a tolerable computational complexity cost when compared to existing decoding algorithms. The decoding technique developed in this research takes advantage of the systematic structure exhibited by linear block codes to implement an information set decoding approach to correct errors in the received vector outputted from the channel. The proposed decoding approach improves the iterative performance of the algorithm as the decoder is only required to detect and correct a subset of the symbols from the received vector. These symbols are referred to as the information set. The information set, which matches the length of the message, is then used decode the entire codeword.

The decoding approach presented in the thesis is tested on both Reed Solomon and Low Density Parity Check codes. The implementation of the decoder varies for both the linear block codes due to the different structural properties of the codes.

Reed Solomon codes have the advantage of having a row rank inverse property which enables the construction of a partial systematic structure using any set of columns in the parity check matrix. This property provides a more direct implementation for finding the information set required by the decoder based on the soft reliability information. However, the dense structure of the parity check matrix of Reed Solomon codes presents challenges in terms of error detection and correction for the proposed decoding approach. To counter this problem, a bit-level implementation of the decoding technique for Reed Solomon codes is presented in the thesis.

The presentation of the parity check matrix extension technique is also proposed in the thesis. This technique involves the addition of low weight codewords from the dual code, that match the minimum distance of the code, to the parity check matrix during the decoding process. This helps add sparsity to the symbol-level implementation of the proposed decoder. This sparsity helps with the efficient exchange of the soft information during the message passing stage of the proposed decoder.

Most high performance Low Density Parity Check codes proposed in literature lack a systematic structure. This presents a challenge for the proposed decoding approach in obtaining the information set. A systematic construction for a Quasi-Cyclic Low Density Parity Check code is also presented in this thesis so as to allow for the information set decoding. The proposed construction is able to match the error correction performance of a high performance Quasi-Cyclic Low Density Parity Check matrix design, while having the benefit of a low complexity construction for the encoder.

In addition, this thesis also proposes a stopping condition for iterative decoding algorithms based on the information set decoding technique. This stopping condition is applied to other high performance iterative decoding algorithms for both Reed Solomon codes and Low Density Parity Check codes so as to improve the iterative performance. This improves on the overall efficiency of the decoding algorithms.

I dedicate this to God

Who has not only give me the ability to come this far, but has been with me through it all.

I dedicate this to my Mother

Mrs Florence Auma Genga, who always believed even when I did not.

I dedicate this to my Father

Prof Riewa Onyango Genga, who has sacrificed it all to give me a my siblings the gift of education.

Acknowledgements

I would like to take this opportunity, with deepest gratitude, to thank both my supervisors Prof O. Olutayo and Prof. D.J.J. Versfeld for the advice and counsel that they have provided through this period. It would have literally been impossible for me to take this work to completion without their incredible support and encouragement. I shall be forever grateful.

I would also like to thank the CeTAS program for the assistance they provided, in terms of finance and the lab equipment.

I would like to express my sincere thanks to all my friends who have patiently extended all kind of help for accomplishing of this undertaking.

Most importantly, I would like to thank my family for being there for me through it all. I thank them for all the financial and moral support they gave me. To them I am forever grateful.

Contents

Declaration i

Abstract ii

Dedication iv

Acknowledgements v

Contents vi

List of Figures x

List of Tables xiii

Abbreviations xiv

Publications xvi

1 Introduction 1
- 1.1 Introduction . . . 1
- 1.2 Motivation and Problem Statement . . . 3
- 1.3 Research Objectives . . . 5
- 1.4 Organisation of the Thesis . . . 6

2 Basic Forward Error Correction Principles Important to the Research 8
- 2.1 Introduction . . . 8
- 2.2 Forward Error Correction Overview . . . 9
  - 2.2.1 GF (2^m) arithmetic . . . 13
- 2.3 Reed-Solomon Codes . . . 14
  - 2.3.1 Definition of Reed-Solomon Codes . . . 14
- 2.4 Low Density Parity Check (LDPC) Codes . . . 16
  - 2.4.1 Representation of LDPC codes . . . 16
- 2.5 Decoding Techniques Used in Error Correction of Linear Block Codes 17
  - 2.5.1 Symbols-Level and Bit-Level Decoding . . . 17
  - 2.5.2 Information Set Decoding . . . 18
  - 2.5.3 Hard-Decision Decoding Techniques . . . 18
    - 2.5.3.1 Reed-Solomon Codes . . . 18
    - 2.5.3.2 LDPC Codes . . . 19

2.5.4 Soft-Decision Decoding Techniques 19
2.6 Iterative Soft-Decision Decoders 20
2.6.1 Belief Propagation 20
2.6.1.1 The Belief Propagation Algorithm for LDPC codes 20
2.6.1.2 The Adaptive Belief Propagation Algorithm 23
2.6.2 The Parity Check Transformation Algorithm 26
2.7 Algorithm Complexity 28
2.8 Importance of the Systematic Structure of the H Matrix in this Research for the Decoding Process 29
2.9 Framework for Simulations 31
2.9.1 The Channel Model 32
2.9.2 Obtaining Bit Probabilities and Creation of the Reliability matrix using a Binary Phase Shift Key (BPSK) Modulation Scheme 33
2.9.3 Obtaining Bit Probabilities and Creation of the Reliability matrix using a 16-QAM scheme 34
2.9.3.1 Gray mapping for the 16-QAM constellations . . . 35
2.9.3.2 Bit Probabilities for a 16-QAM Modulation Scheme 36
2.9.3.3 Creating the Reliability Matrix (β) at A Symbol-Level Using a 16-QAM Modulation Scheme 39
2.9.4 Simulation Model 40
2.9.4.1 Symbol Error Probability 40
2.9.4.2 Simulation setup 42
2.10 Summary 44

3 Literature Review 46
3.1 Introduction 46
3.2 Information Set Decoding 46
3.3 Reed-Solomon Decoders 52
3.4 Bit-level Iterative Soft-decision Decoding of Reed Solomon Codes . 54
3.5 Symbol-level Iterative Soft-decision Decoding on a Dense H matrix of Reed Solomon Codes Based on Information Set Decoding Techniques 61
3.6 Information Set Decoding on a systematic Binary QC-LDPC code . 64

4 Bit-level Implementation of the Proposed Soft-Decision Decoder for Reed-Solomon Codes 68
4.1 Introduction 68
4.2 Preliminary Tests 70
4.2.1 Bit-level Implementation of the PTA Algorithm 70
4.2.1.1 Modifications in the H transformation step 71
4.2.1.2 Modifications to the correction step 72
4.2.2 Results and Analysis 73
4.2.2.1 Algorithm Analysis 73
4.2.2.2 Performance Analysis of Half Rate codes 75

4.2.2.3 Performance Analysis of High Rate codes 77
4.2.3 Conclusions From the Preliminary Results 78
4.3 The Proposed Decoding Scheme 78
4.3.1 The Decoding Condition and Thresholds 82
4.4 Results and Analysis 84
4.4.1 Analysis of the Proposed Bit-Level Decoding Algorithm . . . 84
4.4.2 Performance Analysis of the Proposed Bit-Level Decoding Algorithm 87
4.5 Complexity Analysis 95
4.5.1 Time Complexity 95
4.5.2 Complexity measured in Terms of Number of Operations . . 100
4.6 The Decoding Condition Used as an Additional Stopping Criteria for Bit-Level RS Decoders 103
4.6.1 Bit-Level Implementation of the Decoding Condition with the PTA decoder 105
4.6.2 Bit-Level Implementation of the Decoding Condition with the ABP decoder 110
4.7 Conclusion 113

5 The Symbol-Level Implementation of the Proposed-Decoding Algorithm for Reed-Solomon codes 115
5.1 Introduction 115
5.2 Preliminary Tests 116
5.2.1 The PTA Extended Algorithim 117
5.2.2 Simulation and Results 119
5.2.2.1 Half rate simulations 119
5.2.2.2 High rate simulations 122
5.2.3 Conclusions From the Preliminary Results 123
5.3 Implimentation of the Proposed Iterative Decoding Technique at a Symbol-Level 124
5.4 The Stopping Criteria for the Algorithm 129
5.5 Results and Analysis 130
5.5.1 Analysis of the Proposed Symbol-Level Decoding Algorithm 130
5.5.2 Performance Analysis of the Proposed Symbol-Level Decoding Algorithm 134
5.5.3 Half Rate Codes 134
5.5.4 High Rate Codes 136
5.6 Complexity Analysis 139
5.6.1 Time Complexity 139
5.6.2 Complexity measured in Terms of Number of Operations . . 144
5.7 The Decoding Condition Used as an Additional Stopping Criteria for the Symbol-Level PTA Decoder 147
5.8 Conclusion 151

6 The Decoding condition applied to Binary QC-LDPC codes 152

6.1 The Proposed Systematic Construction for QC-LDPC Codes 152
6.1.1 General Encoding of QC-LDPC codes 153
6.1.2 The Proposed Row Reduction Technique 154
6.1.3 The Systematic QC-LDPC Parity check Matrix 155
6.1.4 The Generator matrix (Systematic Encoder) 156
6.1.5 Performance Analysis 157
6.1.5.1 Rank Analysis 157
6.1.5.2 BER Performance of the Proposed Systematic QC-LDPC code Compared to a High Performance code Construction 159
6.1.5.3 Complexity Analysis 161
6.1.6 Performance Analysis with QC-LDPC codes used in Practical Applications 163
6.1.6.1 Performance against the IEEE802.16-2012 QC-LDPC codes 163
6.1.6.2 Performance against the ITU-T G.9960 QC-LDPC codes 165
6.2 Decoding of QC-LDPC using the Decoding condition 167
6.2.1 QC-LDPC codes Results 167
6.3 Conclusion 168

7 Conclusion and Future Work **170**
7.1 Conclusion 170
7.2 Future Work 174

Bibliography **176**

List of Figures

2.1 16-QAM constellation plot with gray coded mapping. 36
2.2 16-QAM alphabet and the corresponding decision regions. 41
2.3 Uncoded simulation graphs using different types soft information . . 44

4.1 Performance comparison of the PTA$_{bl}$ based on the different δ implementations. 74
4.2 Performance comparison of the PTA$_{bl}$ based on the value of δ used. 75
4.3 Performance of the PTA$_{bl}$ compared to the symbol-level PTA for a $(15, 7)$ RS code 76
4.4 Performance of the PTA$_{bl}$ compared to the symbol-level PTA for a $(15, 11)$ code 77
4.5 Performance comparison of the kBD based on different τ values in terms of BER for a (15,7) RS code. 85
4.6 Performance comparison of the kBD based on different τ values in terms of number of iterations for a (15,7) RS code. 85
4.7 Performance comparison of the kBD based on different ρ values in terms of BER for a (15,7) RS code. 86
4.8 Performance comparison of the kBD based on different ρ values in terms of average number of iterations for a (15,7) RS code. 86
4.9 Performance comparison of the ABP, PTA$_{bl}$, kBD and kBD$_{nt}$ in terms of BER for a (15,7) RS code. 88
4.10 Performance comparison of the ABP, PTA$_{bl}$, kBD and kBD$_{nt}$ in terms of average number of iterations for a (15,7) RS code.. 88
4.11 Performance comparison of the kBD based on different τ values in terms of BER for a (15,11) RS code. 90
4.12 Performance comparison of the kBD based on different ρ values in terms of average number of iterations for a (15,11) RS code. 91
4.13 Performance comparison of the ABP, PTA$_{bl}$, kBD and kBD$_{nt}$ in terms of BER for a (15,11) RS code.. 92
4.14 Performance comparison of the ABP, PTA$_{bl}$, kBD and kBD$_{nt}$ in terms of average number of iterations for a (15,11) RS code.. 92
4.15 Performance comparison of the ABP, PTA$_{bl}$, kBD and kBD$_{nt}$ for a $(31, 25)$ RS code. 94
4.16 Complexity comparisons for the bit-level decoders applied to a (15,7) RS code over an AWGN channel using a 16-QAM modulation scheme. 101
4.17 Complexity comparisons for the bit-level decoders applied to a (15,11) RS code over an AWGN channel using a 16-QAM modulation scheme. 102

4.18 SER performance of PTA_{bl} vs the PTA_{bl} implemented with the decoding condition defined by different values of τ for a $(15, 7)$ RS code. 106
4.19 Iterative performance of PTA_{bl} vs the PTA_{bl} implemented with the decoding condition defined by different values of τ for a (15×7) RS code. 106
4.20 SER performance of PTA_{bl} vs the PTA_{bl} implemented with the decoding condition defined by different values of τ for a $(15, 11)$ RS code. 108
4.21 Iterative performance of PTA_{bl} vs the PTA_{bl} implemented with the decoding condition defined by different values of τ for a $(15, 11)$ RS code. 109
4.22 SER performance of ABP vs the ABP implemented with the decoding condition defined by different values of f^{ι} for a $(15, 7)$ RS code. 111
4.23 Iterative performance of ABP vs the ABP implemented with the decoding condition defined by different values of f^{ι} for a $(15, 7)$ RS code. 111
4.24 SER performance of ABP vs the ABP implemented with the decoding condition defined by different values of f^{ι} for a $(15, 11)$ RS code. 112
4.25 Iterative performance of ABP vs the ABP implemented with the decoding condition defined by different values of f^{ι} for a $(15, 11)$ RS code. 113

5.1 SER Performance comparison of the PTA extended algorithm with 2, 3 and 4 H matrices used to decode the received vector for a (15,7) RS codes. 120
5.2 Iterative performance comparison of the PTA extended algorithm with 2, 3 and 4 H matrices used to decode the received vector for a (15,7) RS codes. 120
5.3 Performance comparison of the PTA and the PTAe_3 in terms of SER for a $(15, 7)$ RS code. 121
5.4 Performance comparison of the PTA and the PTAe_3 in terms of average number of iterations for a $(15, 7)$ RS code. 121
5.5 Performance comparison of the PTA and the PTAe_3 in terms of SER for a $(15, 11)$ RS code. 122
5.6 Performance comparison of the PTA and the PTAe_3 in terms of average number of iterations for a $(15, 11)$ RS code. 123
5.7 SER performance comparison of the kSD for a $(15, 7)$ RS code over AWGN channel for different values of ε. 131
5.8 Iterative performance comparison of the kSD for a $(15, 7)$ RS code over AWGN channel for different values of ε. 132
5.9 SER performance comparison of the kSD for a $(15, 11)$ RS code over AWGN channel for different values of ε. 133

5.10 Iterative performance comparison of the kSD for a $(15, 11)$ RS code over AWGN channel for different values of ε. 134
5.11 SER performance comparison of the kSD for a $(15, 7)$ RS code over AWGN channel. 135
5.12 Iterative performance comparison of the kSD for a $(15, 7)$ RS code over AWGN channel. 136
5.13 SER performance comparison of the kSD for a $(15, 11)$ RS code over AWGN channel. 137
5.14 Iterative performance comparison of the kSD for a $(15, 11)$ RS code over AWGN channel. 137
5.15 Performance comparison of the SYM ABP and the kSD for a $(63, 55)$ RS code. 138
5.16 Complexity comparisons for the symbol-level decoders applied to a (15,7) RS code over an AWGN channel using a 16-QAM modulation scheme. 145
5.17 Complexity comparisons for the symbol-level decoders applied to a (15,11) RS code over an AWGN channel using a 16-QAM modulation scheme. 146
5.18 SER perfomance of PTAe_3 vs PTAe_3 implemented with the decoding condition defined by $\tau = 3$. 148
5.19 Iterative performance of PTAe_3 vs PTAe_3 implemented with the decoding condition defined by $\tau = 3$. 149
5.20 SER performance of PTAe_3 vs PTAe_3 implemented with the decoding condition defined by $\tau = 1$. 150
5.21 Iterative performance of PTAe_3 vs PTAe_3 implemented with the decoding condition defined by $\tau = 1$. 150

6.1 Performance comparison of the proposed systematic $(806, 620)$ QC-LDPC code against the non-systematic $(744, 573)$ QC-LDPC code using the SPA decoder. 160
6.2 Performance comparison of the proposed systematic $(899, 806)$ QC-LDPC code against the non-systematic $(899, 813)$ QC-LDPC code using the MSA decoder. 161
6.3 The proposed systematic $(651, 527)$ QC-LDPC code against the IEEE802.11-2012 $(648, 540)$ QC-LDPC code using the SPA decoder. 164
6.4 The proposed systematic $(1260, 945)$ QC-LDPC code against the IEEE802.11-2012 $(1296, 972)$ QC-LDPC code using the SPA decoder. 164
6.5 The proposed systematic $(1134, 945)$ QC-LDPC code against the ITU-T G.9960 $(1152, 960)$ QC-LDPC code using the SPA decoder. . 166
6.6 The proposed systematic $(1449, 966)$ QC-LDPC code against the ITU-T G.9960 $(1440, 960)$ QC-LDPC code using the SPA decoder. . 166
6.7 BER perfomance of SPA vs SPA implemented with the decoding condition defined by $\tau = 2$. 168
6.8 Iterative performance of SPA vs SPA implemented with the decoding condition defined by $\tau = 2$. 169

List of Tables

2.1 Bit representation for the In phase and Quadrature components in the 16-QAM modulation scheme 35
2.2 Bit mapping used to create symbol positions on 16-QAM constellations. 35

4.1 Summary of the overall complexity for the ABP decoder. 95
4.2 Summary of the overall complexity for PTA$_{bl}$ decoder. 96
4.3 Summary of the overall complexity for the kBD decoder. 97
4.4 Summary of the overall complexity of the kBD$_{nt}$ decoder. 98
4.5 Summary of the computational complexity for the ABP decoder. . . 99
4.6 Summary of the computational complexity for PTA$_{bl}$ decoder. . . . 99
4.7 Summary of the computational complexity for the kBD and the kBD$_{nt}$ decoder. 99

5.1 Summary of the overall complexity for the symbol-level PTA decoder. 140
5.2 Summary of the overall complexity for the symbol-level ABP decoder. 141
5.3 Summary of the overall complexity for the kSD algorithm. 142
5.4 Summary of the computational complexity for the symbol-level PTA decoder. 143
5.5 Summary of the computational complexity for the symbol-level ABP decoder. 143
5.6 Summary of the computational complexity for the kSD algorithm. . 144

6.1 Summary of the complexity in the encoder construction. 162

Abbreviations

ABP	**A**daptive **B**elief **P**ropagation
ABP-ASD	**A**daptive **B**elief **P**ropagation **A**lgebraic **S**oft **D**ecoding
ABP-BM	**A**daptive **B**elief **P**ropagation **B**erlekamp **M**assey
ABP-KV	**A**daptive **B**elief **P**ropagation **K**oetter **V**ardy
AWGN	**A**dditive **W**hite **G**aussian **N**oise
BER	**B**it **E**rror **R**ate
BCH	**B**ose-**C**haudhuri-**H**ocquenghem(BCH)
BM	**B**erlekamp **M**assey
BP	**B**elief **P**ropagation
BPSK	**B**inary **P**hase **S**hift **K**eying
FEC	**F**orward **E**rror **C**orrection
GF	**G**alois **F**ield
GS	**G**uruswami-**S**udan
HDPC	**H**igh **D**ensity **P**arity **C**heck
k**BD**	k-**B**it **D**ecoder
k**SD**	k-**S**ymbol **D**ecoder
KV	**K**oetter-**V**ardy
LDPC	**L**ow **P**arity **D**ensity **C**heck
MAP	**M**aximum **a** **P**osteriori
MDS	**M**aximum **D**istance **S**eparable
ML	**M**aximum **L**ikelihood
PAM	**P**ulse **A**mplitude **M**odulation
PTA	**P**arity check matrix **T**ranformations **A**lgorithm
PTA$_{bl}$	**B**it **L**evel **P**arity check matrix **T**ranformations **A**lgorithm
PTAe	**E**xtended **P**arity check matrix **T**ranformations **A**lgorithm

PSK	**P**hase **S**hift **K**eying
QAM	**Q**uadrature **A**mplitude **M**odulation
QC-LDPC	**Q**uasi **C**yclic **L**ow **P**arity **D**ensity **C**heck
SPA	**S**um **P**roduct **A**lgorithm
RS	**R**eed **S**olomon
SNR	**S**ignal-to-**N**oise **R**atio
SER	**S**ymbol **E**rror **R**ate

Publications

Overlapping chapter	Peer-reviewed conference and journal papers directly related to this Thesis
[4]	**Yuval Genga**, Olutayo Oyerinde and Jaco Versfeld, "Bit Level Implementation of the PTA Algorithm for Reed-Solomon Codes", *Global Wireless Summit (GWS)*, pp. 39-43, 2017.
[4]	**Yuval Genga** , Olutayo Oyerinde and Jaco Versfeld, "Improved Iterative Convergence Rate for Soft-Decision Bit-Level Reed-Solomon Decoders Using Information Set Decoding ", *IEEE VTS Wireless African Conference (WAC)*, Accepted and Presented on August 19 2019. (**Won the Best Paper Award**)
[5]	**Yuval Genga**, Olutayo Oyerinde and Jaco Versfeld, "Improving the Decoding Performance of the PTA Algorithm for RS Codes Through the Extension of the Parity Check Matrix", *AFRICON*, pp. 171-174, 2017.
[6]	**Yuval Genga**, Olayinka Ogundile, Olutayo Oyerinde and Jaco Versfeld, "A Low Complexity Approach to the Construction of Systematic Quasi-Cyclic LDPC Codes ", *AFRICON*, pp. 167-170, 2017.
[4]	**Yuval Genga**, Olutayo Oyerinde and Jaco Versfeld, "Iterative Soft-Input Soft-Output Bit-Level Reed-Solomon Decoder Based on Information Set Decoding", under review for publication in the Elsevier Computer and Electrical Engineering Journal.
[5]	**Yuval Genga**, Olutayo Oyerinde and Jaco Versfeld, "Symbol-Level Soft-Input Iterative Decoding of Reed-Solomon Codes Based on Information Set Decoding using Parity Check Matrix Extensions", In preparation for submission to the IEEE Access journal.

CHAPTER 1

Introduction

1.1 Introduction

Forward error correction is an error control strategy used in communication systems, for data sent from the transmitter to the receiver. Forward error correction decoding techniques are used to detect and correct errors at the receiver which are introduced during transmission through a noisy channel. Error correction at the receiver is very important as it saves on power that would be wasted on retransmission, especially with the constant, rapidly increasing exchange of data in today's world.

There exist two main classes of codes used for the purposes of error correction; linear block codes and convolution codes. Linear block codes are favoured for practical applications due to their advantageous linear properties, easy encoding procedures and efficient decoding algorithms. Examples of popular linear block codes used in error correction include Bose-Chaudhuri-Hocquenghem(BCH)[1], Reed-Solomon codes [2], Low Density Parity Check codes [3] among others.

Reed-Solomon (RS) codes are a popular class of non-binary linear block code [2] that were developed by Irving S.Reed and Gustave Solomon [2]. This class of code has a wide range of applications that include telecommunications, storage devices and even space technology. These codes are largely popular due to their favourable algebraic properties which enables good error detection and correction with implementable decoders. Based on the good error detection performance of this class of code, there has been a lot of research on the performance analysis based on efficient encoding, decoding and overall design.

There are many techniques used in the decoding of RS codes. Most of these decoding techniques are either classified as hard-decision decoders or soft-decision decoders. Hard-decision decoders that decode up to the minimum distance, such as the Berlekamp-Massey (BM) algorithm [4] and the Euclidean algorithm [5], have been shown to efficiently perform error correction on the received vector of RS codes that are outputted from noisy channels. However, if soft information is present, a decoder that utilises this performs better [6]. Examples of the RS decoders that utilise soft-decision decoding techniques include the widely used Koetter and Vardy (KV) algorithm [7], the Adaptive Belief Propagation (ABP) algorithm and the Parity check Transformation algorithm (PTA) [8]. These soft-decision decoders give significant gains over hard-decision decoders, but this comes at the cost of computational complexity.

This research focuses on the development of an iterative soft-input soft-output decoding algorithm that is able to yield a good error correction performance while running at a relatively lower complexity when compared to existing high performing decoders. The decoding approach presented in this research takes advantage of low complexity techniques to detect and correct an information set [9] [10] of symbol or bits during the iterative decoding process based on the soft information. The information set is made up of a subset of symbols or bits in the received vector. The length of the information set varies based on the rate of the code. The information set obtained by the proposed algorithm is used to decode the entire received vector. This decoding approach aids in increasing the efficiency of the decoding process by reducing the decoder complexity. This is because it only has to successfully decode a subset of symbols in the received vector as opposed to all of the symbols in the received vector. To measure the performance of the proposed algorithm presented in this research, simulations are run with the results compared to the ABP [6] and the PTA [8] decoders which are considered as high performing iterative soft-input soft-output decoding algorithms for RS codes. The decoding approach presented in this research is also be implemented as an additional stopping criteria for both the probability domain[1] [3] and the log domain [11] sum product algorithm (SPA) when applied to Quasi-Cyclic Low Density Parity Check

Codes (QC-LDPC).

1.2 Motivation and Problem Statement

Soft-decision decoders used for RS codes give a good error correction performance but at the cost of some form of complexity. The ABP algorithm gives a significant gain compared to hard-decision decoders and the widely used KV algorithm, but this comes at the cost of a high computational complexity [6], [12]. The PTA algorithm has also been shown to be a good decoder while outperforming the KV algorithm [8]. This is at the cost of the algorithm running numerous iterations [13], therefore making the PTA a computationally intensive algorithm.

A similar case is also seen in the decoding of the Low Density Parity Check (LDPC) codes decoded with the belief propagation algorithm. LPDC codes [14] have been shown in literature to attain a near Shannon limit performance [3] but this is at the cost of computation complexity [11]. Part of the reason decoders based on belief propagation work so well with LDPC codes is due to the sparse nature of parity check matrices.

This research focuses on the development of a low complexity iterative soft-input soft-output decoding approach for RS codes and a subclass of LDPC codes referred to as Quasi-Cyclic Low Density Parity Check (QC-LDPC) codes. QC-LDPC codes are selected for this research largely due their simple implementation by use of shift registers and their good bit error rate (BER) performance when transmitted through a noisy channel [15].

One of the hindering factors of the implementation of the decoder for QC-LDPC codes is the lack of parity check matrices defined by a systematic structure for the class of code. Construction of most QC-LDPC codes focuses on ensuring the cycles on the Tanner graphs do not have cycles smaller than a length of 4. Having cycles of length 4 in the Tanner graphs that define the parity check matrix hinders the exchange of soft information during the decoding of LDPC codes [15]. This leads to a loss in terms of error correction performance of the LDPC code [1] [11]. Due to the focus on the length of cycles on the Tanner graph, research

on systematic construction of QC-LDPC codes is not common. Also, systematic constructions of QC-LDPC are inhibited due to most QC-LDPC codes not having a parity check matrices defined by a full rank. This is because of the large size of the parity check matrix.

A parity check matrix with a systematic structure is important for the development of the proposed iterative soft-input soft-output decoding approach. This is because the systematic structure is required for the use of information set decoding. Part of this research, therefore, focuses on the construction of a QC-LDPC code defined by a parity check matrix with a systematic structure. This contribution not only benefits the implementation of the proposed iterative decoder, but also provides an option for error correction application with a QC-LDPC code that has the benefits of a systematic structure. The benefits of a systematic structure includes obtaining decoded information that does not need to be converted into user data at the receiver and a simple encoder construction.

The "complexity vs performance" tradeoff is quite common in the field of telecommunications when it comes to the selection of appropriate decoding algorithms. This research focuses on the development of a decoding technique that iteratively utilizes the soft information outputted from the noisy channel to yield a high error correction performance at a reduced computational complexity cost. The development of a reduced complexity decoding scheme is done with the aim of eliminating the current "complexity against performance" trade-off that is used during the decoding process. This would enable efficient use of error correction codes for a wider range of applications in high speed data communication and digital storage systems.

Research Question

Based on the highlighted problem statement this research attempts to answer the following question.

- *What forward error correction techniques can be applied to linear block codes to reduce complexity while maintaining a good error detection and correction performance during the iterative decoding processes?*

Based on the implementation of the proposed techniques, this research also additionally attempts to answer the following questions

- *What performance is attained from the proposed iterative decoding processes when implemented for different code rates?*
- *What performance is attained from the proposed iterative decoding processes when applied to both a bit and symbol-level?*
- *Is it possible to improve the iterative convergence rate of the ABP [6], PTA [8] and the SPA [1] [11] decoders while maintaining the error correction performance using information set decoding techniques as an additional stopping condition?*

1.3 Research Objectives

The main objective of this research is the development of a decoding approach that is able to maintain high error correction performance while working at a complexity that is considered to be tolerable. This is done for the sole purpose of increasing the efficiency of the decoding process of linear block codes at the receiver for the wide range of application in different communication systems.
The main research contributions presented in this research are as follows:

1. Development of an iterative soft-input soft-output information set decoding algorithm that utilises the systematic structure of the parity check matrix of a RS code working at a bit-level with a reasonable computational complexity.
2. Development of an iterative soft-input soft-output information set decoding algorithm that is applied to the symbol-level dense parity check matrix of a RS code with a reasonable computational complexity.

3. Development of a systematic QC-LDPC code with a reasonable construction complexity.

4. Improve the iterative convergence rate, without a significant addition to complexity, of the ABP, PTA and Belief propagation algorithms through the use of an additional stopping criteria.

1.4 Organisation of the Thesis

The rest of the thesis is structured as follows:
In Chapter 2 a basic overview of the forward error correction principles used in the research is presented. The literature presented in this chapter also covers the different decoding techniques used in the error detection and correction of RS and LDPC codes. The basic idea of how the systematic structure of the code is used to obtain the information set required by the decoder is proposed in this chapter. The simulation setup used to arrive at the conclusion presented in this research is described in Chapter 2 as well.

Chapter 3 presents an overview of the related works in literature, mainly focusing on iterative soft-input soft-output decoder implementations on RS codes at a bit-level and a symbol-level. This chapter also provides a literature survey on the construction of systematic QC-LDPC codes.

Chapter 4 proposes the implementation of the proposed decoder at a bit-level when applied to RS codes. The implementation of an additional stopping condition which yields an improved iterative convergence rate during the decoding process when applied to the ABP and the bit-level PTA is also presented in this Chapter.

Chapter 5 presents the implementation of iterative soft-decision information set decoding on a RS code at the symbol-level. This implementation is made possible with the aid of parity check matrix extensions. Work on an additional stopping

condition is also carried out in the section when applied to iteratve soft-decision symbol-level decoders.

Chapter 6 looks into investigating the implementation of information set decoding techniques as an additional stopping criteria for binary systematic QC-LDPC codes. A systematic approach to the construction of QC-LDPC codes is also presented in this chapter. An analysis in terms of performance and construction complexity of the proposed decoding approach is investigated in this chapter as well.

Finally, the contributions presented in the thesis are summarised and concluded in Chapter 7.

CHAPTER 2

Basic Forward Error Correction Principles Important to the Research

2.1 Introduction

The purpose of this chapter is to introduce and establish notation used in Forward Error Correction related to this research. This chapter presents the definitions of Reed-Solomon (RS) codes and Low Density Parity Check (LDPC) codes. This chapter also give a brief overview of the different iterative decoding techniques used for both RS and LDPC codes. The importance of the systematic structure of linear block codes and how it assists in the decoding process is also discussed in this chapter.

The rest of the chapter is structured as follows; The basic concepts of forward error correction used in the research are presented in Section 2.2. Reed-Solomon codes and LDPC codes are defined in Section 2.3 and 2.4 receptively. The various decoding techniques used in error correction of linear block codes are discussed in Section 2.5. High performance error correction iterative soft-decision decoding algorithms are presented in Section 2.6. The importance of the systematic structure of linear block codes when applied to this research is presented in Section 2.8. The simulation setup that is used to test the error correction performance of the proposed iterative decoder is described in Section 2.9. A summary to this chapter is given in Section 2.10.

2.2 Forward Error Correction Overview

In telecommunications, forward error correction is the study of detecting and correcting the errors found in data transmitted over an unreliable noisy communication channel. The message to be transmitted is encoded by the sender through the addition of redundant symbols to form the codeword. This redundancy is utilized in the decoding process by enabling the receiver to detect a specified number of errors that may be found in the received message outputted from the noisy channel. Error correcting codes are often referred to as linear codes. Important forward error correction principles are now defined:

- **Galois Field(GF):** A field is a set of elements where by the addition, subtraction, multiplication and division operations can be performed on any number of elements in the set with the results being an element in the same set. A Galois Field (GF) is a field that contains a finite number of elements. This field is usually denoted as GF(q^m), where m is a whole, non-negative number and q is a prime number. All arithmetic calculations are performed using modulo-q operations. In forward error correction, all the possible symbols of a transmitted codeword c are elements in a the field GF(q^m).

- **The primitive element of the field:** For the field GF(q^m), there exists at least one special element referred to as the primitive element and is denoted as α. The primitive element is defined as any nonzero element in the field with the order $(q^m - 1)$. This means that any nonzero element in the field GF(q^m) can be represented as a power of α [16].

- **Linear block codes:** There are a subclass of linear codes that encode data in blocks. These codes are denoted as (n, k) linear block codes, because a message with k symbols is input into the encoder and a codeword of n symbols is obtained at the output of the encoder, where $n > k$ through the addition of parity symbols. This codeword is what is transmitted across the noisy channel and received at the demodulator. For the field GF(q^m), a (n, k) linear block code $\mathcal{C}$ contains a set of $(q^m)^k$ codewords[1]. Examples of

linear block codes include Hamming Codes, Reed-Solomon Codes and Low Density Party Check codes. The parameters used to define linear block codes are as follows:

Definition 2.1: For a (n,k) linear block code $\mathcal{C}$, k denotes the number of information symbols in the message.

Definition 2.2: For a (n,k) linear block code $\mathcal{C}$, n denotes the total length of a codeword c.

Definition 2.3: The parity symbols of a linear block code are defined as the number of redundancy symbols per transmitted codeword. The number of parity symbols are expressed as:

$$n - k. \tag{2.1}$$

Definition 2.4: The code rate $\mathcal{R}$ of a block code is the ratio of the information symbols k to the codeword symbols n. It is represented as

$$\mathcal{R} = \frac{k}{n}. \tag{2.2}$$

From (2.2), it can be seen that more information symbols can be transmitted for higher code rates.

Definition 2.5: The Hamming distance between two codewords of the same length is defined as the total number of positions in which the two codewords differ. The hamming distance between codewords c_1 and c_2 of the code $\mathcal{C}$ is represented as

$$d_h(c_1, c_2) \tag{2.3}$$

Definition 2.6: The minimum Hamming distance, $d_{\min}$, of a linear block code is expressed as [17]:

$$d_{\min} = \min\{d_h(c_1, c_2) : c_1, c_2 \in \mathcal{C}, c_1 \neq c_2\}, \tag{2.4}$$

The minimum distance $d_{\min}$ of a linear block code can be used to determine

the number of errors that the code can detect or correct. In general, a linear block code can reliably detect ϵ errors where:

$$\epsilon \leq d_{\min} - 1. \tag{2.5}$$

Definition 2.7: The Euclidean distance is defined as the sum of the squared distance between two points in signal space. For example, the Euclidean distance between the two signal points a_1 and a_2 in one dimensional is computed as [11]

$$d_E(a_1, a_2) = \sqrt{\sum_{j}^{n} (a_{1_j} - a_{2_j})^2}, \tag{2.6}$$

where $1 \leq j \leq n$.

Definition 2.8: A Maximum Distance Separable (MDS) code is a special class of linear block codes whose minimum distance, $d_{\min}$, can be defined as

$$d_{\min} = n - k + 1. \tag{2.7}$$

MDS codes are able to correct the largest number of errors for a given code based on the code rate $\mathcal{R}$. MDS codes also have the property

$$n - k = 2\lambda, \tag{2.8}$$

where λ represents the minimum number of correctable errors per transmitted codeword, and is represented as

$$\lambda = \left[\frac{d_{\min} - 1}{2} \right]. \tag{2.9}$$

The condition defined by (2.9) is referred to as the Singleton bound [18]. Reed-Solomon codes form the most essential class of MDS codes [17].

- **The Generator matrix(G):** This is a matrix whose rows are defined by a set of k linearly independent codewords of the (n, k) linear block code $\mathcal{C}$,

and has the dimensions $k \times n$. The Generator matrix is used in the encoding process by adding parity symbols to a message u (of length k), to obtain the codeword c (of length n) as shown in (2.10)

$$c = u \times G. \tag{2.10}$$

- **The Parity check matrix (H):** An (n, k) linear block code $\mathcal{C}$ exists in a k-dimensional vector subspace, because it is made up of $(q^m)^k$ codewords. Therefore, there exist an $(n-k)$-dimensional dual space to $\mathcal{C}$ [1]. The parity check matrix H, with the dimensions $(n-k) \times n$, is defined as the dual code of the generator matrix G. This means that G and H must satisfy

$$G \times H^\top = 0, \tag{2.11}$$

where $H^\top$ is the transpose of the matrix H.

- **The Syndrome (S):** For a given codeword c to exist in the k-dimensional subspace of the code $\mathcal{C}$, it must satisfy

$$S = c \times H^\top, \tag{2.12}$$

where S is a zero vector of length $(n-k)$ if and only if c is a valid codeword of the code $\mathcal{C}$. The syndrome calculation in (2.12) can also be represented in terms of the scalar product as shown in (2.13) [17]

$$S_i = c \cdot H_i, \tag{2.13}$$

where $0 \leq i \leq (n-k)$ represents the rows of H. During iterative decoding processes, the syndrome is often calculated with the received vector to check if it is a valid codeword.

- **The systematic structure:** The systematic structure of the G and the H matrices is of the form shown in (2.14) and (2.15) receptively

$$G = [I \quad Q] \quad \text{or} \quad G = [Q \quad I], \tag{2.14}$$

$$H = [Q^\top \quad I] \quad \text{or} \quad H = [I \quad Q^\top], \tag{2.15}$$

  where Q represents the parity submatrix of dimensions $k \times (n-k)$. The submatrix I is an identity matrix of dimension $k \times k$ for G and $(n-k) \times (n-k)$ for H. This structure is very important for the implementation of information set decoding 2.8.

2.2.1 GF (2^m) arithmetic

The decoding technique developed in this research works based on the additive inverse property of the field GF (2^m) [17]; For any element α^z in the field, where α represents the primitive element of the field and z is either $-\infty$ or $0 \leq z \leq (2^m - 2)$,

$$\alpha^z + \alpha^z = 0. \tag{2.16}$$

The other important GF (2^m),but basic, arithmetic expressions required for this research include; for any element in the field α^z when multiplied by the 0 symbol $(\alpha^{-\infty})$ in the field gives a 0, as shown in (2.17). Also any element in the field α^z when multiplied to the multiplication identity 1 (α^0) in the field gives the same element α^z, as shown in (2.18)

$$\alpha^z \times 0 = 0, \tag{2.17}$$

$$\alpha^z \times 1 = \alpha^z. \tag{2.18}$$

2.3 Reed-Solomon Codes

Reed-Solomon (RS) codes are a popular set of cyclic linear block codes that are often used in practical systems. More than 50 years after they were developed by Irving S. Reed and Gustave Solomon [2] RS codes are still one of the more popular class of linear block codes used in error correction. RS codes are largely popular due to their favourable algebraic properties. They can also be defined as Non-binary Bose, Chaudhuri and Hocquenghem (BCH) codes and are widely used in correcting random symbol errors, burst errors, and erasures in digital communication and data storage systems [17].

2.3.1 Definition of Reed-Solomon Codes

The Generator polynomial and the Parity Check polynomial

A (n, k) RS code can be defined by the generator polynomial $g(x)$ and the parity check (dual code) polynomial $h(x)$.
The generator polynomial of a RS code over GF(q^m) of length $n = q^m - 1$ has a degree of $(n - k)$ and can be expressed as [16]

$$g(x) = \prod_{i=k}^{n-1}(x - \alpha^{-i}). \tag{2.19}$$

The parity check polynomial $h(x)$ is obtained using a similar calculation as that of the generator polynomial $g(x)$. The parity check polynomial $h(x)$ has a degree k and can be calculated using (2.20)[16]

$$h(x) = \prod_{i=0}^{k-1}(x - \alpha^{-i}). \tag{2.20}$$

From (2.19) and (2.20) it is important to note that the roots for the polynomials $g(x)$ and $h(x)$ are disjoint. This means all the possible roots α^i are involved in the calculation for both polynomials.

The Generator Matrix and the Parity Check Matrix

From (2.19), the generator polynomial $g(x)$ can be rewritten as

$$g(x) = (x-\alpha^{-k})(x-\alpha^{-(k+1)})...(x-\alpha^{-(n-1)}), \quad (2.21)$$

$$g(x) = g_0 + g_1 x + ... + g_{(n-k)}x^{(n-k)}. \quad (2.22)$$

Based on the properties of cyclic codes, the generator matrix G of a (n,k) RS code can be created by cyclically shifting the coefficients of the polynomial k times to give the form shown in (2.23) [16]

$$G = \begin{bmatrix} g_0 & g_1 & g_2 & ... & g_{(n-k)} & 0 & 0 & ... & 0 \\ 0 & g_0 & g_1 & g_2 & ... & g_{(n-k)} & 0 & ... & 0 \\ 0 & 0 & g_0 & g_1 & g_2 & ... & g_{(n-k)} & ... & 0 \\ \vdots & & & & \vdots & & & & \vdots \\ 0 & 0 & ... & 0 & g_0 & g_1 & g_2 & ... & g_{(n-k)} \end{bmatrix}. \quad (2.23)$$

The parity check matrix H can also be obtained in a similar way to that of the generator matrix. Firstly, the equation for parity check polynomial is expanded from the form shown in (2.20)

$$h(x) = (x-\alpha^{0})(x-\alpha^{-1})...(x-\alpha^{-(k-1)}), \quad (2.24)$$

$$h(x) = h_0 + h_1 x + ... + h_k x^k. \quad (2.25)$$

The parity check matrix is then created by shifting the coefficients of $h(x)$ $(n-k)$ times as shown (2.26) [16]

$$H = \begin{bmatrix} h_k & h_{(k-1)} & h_{(k-2)} & ... & h_0 & 0 & 0 & ... & 0 \\ 0 & h_k & h_{(k-1)} & h_{(k-2)} & ... & h_0 & 0 & ... & 0 \\ 0 & 0 & h_k & h_{(k-1)} & h_{(k-2)} & ... & h_0 & ... & 0 \\ \vdots & & & \vdots & & & & & \vdots \\ 0 & 0 & ... & 0 & h_k & h_{(k-1)} & h_{(k-2)} & ... & h_0 \end{bmatrix}. \quad (2.26)$$

2.4 Low Density Parity Check (LDPC) Codes

Low Density Parity Check (LDPC) codes get their name from the low density of non zero elements in their H matrices. Due to this fact, the different types of LDPC codes are completely defined by the sparse nature of their respective H matrices. This class of codes were discovered by Gallager in his PhD thesis [14] in 1963 but due to their computational and implementation complexity at that time they were mostly ignored. Research on LDPC codes was resurrected in the mid 1990's when they were popularized by Davey and MacKay who proved these codes had a near Shannon limit performance, while decoding using probabilistic soft-decision decoding algorithms, and other advantages of block codes with low density (sparse) parity check matrices[19]. Since then there has been a lot of research on the performance analysis based on construction, efficient encoding, decoding and overall design of these codes for various applications in digital communication and storage devices.

2.4.1 Representation of LDPC codes

When representing LDPC codes, there are 2 categories that are considered. An LDPC code can either be a regular LDPC code or an irregular LDPC code. A regular LDPC code is a LDPC code whose parity check matrix is characterised by having the same the column weight and the same row weight. On the contrary if the parity check matrix of the LDPC code has different row weights and different column weights, then the code refereed to as an irregular LDPC code. For both regular and irregular LDPC code constructions the following structural constratints are imposed on the structural property on the parity check matrix: no two rows or columns have more than one position in common that contains a non zero element [11]. This structural property is refereed to as the row-column constraint. This constraints ensures that a girth of length 4 is not present in the Tanner graph representation of the LDPC codes [20]. Having a girth of length 4

negatively affects the error correction performance of the LDPC code. This is because it increases the chances of the algorithm getting stuck at pseudo-equilibrium points during the exchange of the soft information in the iterative decoding process [1].

2.5 Decoding Techniques Used in Error Correction of Linear Block Codes

2.5.1 Symbols-Level and Bit-Level Decoding

Soft-decision decoders for RS codes working in the field GF(2^m), can be divided into two categories. They can either be symbol-level decoders or bit-level decoders. Most soft-decision decoders for RS codes work on a symbol-level. Symbol-level decoders work on the $2^m - 1$ symbols of the elements in the field during the decoding process, with the arithmetic being done in the field GF(q^m). Examples of soft-decision decoding algorithms implemented on a symbol-level can be found in [7, 8, 19]

When working in the field GF(2^m) each symbol can be expressed in the form of m-bits. Due to the field GF(2^m) being an extension field of GF(2), calculations with the bit representation of the code are carried out using the modulo 2 operations. Bit-level decoders work by first converting a codeword of length n into its bit form. This yields a vector of length nm, where $nm = n \times m$. The codeword in its bit form is then used in the decoding process. The conversion of the symbols to bits is not only performed on the codeword but also on the H matrix by means of a binary image expansion [6] [21].

- ***Definition 2.9***: In order to understand the binary expansion its important to note that each element α^z of the field GF(2^m) can be replaced by a corresponding $m \times m$ binary matrix B^z, where B is denotes a companion matrix of the primitive polynomial which creates the field GF(2) [21]. The binary image of the H matrix is obtained by replacing each element in the

matrix with their corresponding companion matrix to give a binary matrix with the dimensions $(nm - km) \times nm$ where $km = k \times m$.

Research performed on decoding algorithms at a bit-level can found in [6, 12, 22–26].

2.5.2 Information Set Decoding

Information set decoding is a decoding technique that was developed by Prange [9] and was used in the decoding of Cyclic codes. This decoding approach works by using a set of k linearly independent bits from the received vector to construct a unique codeword of the code [10]. The k linearly linearly independent bits are referred to as the information set with remaining $(n - k)$ bits being erased. The information set is used to reconstruct the $(n - k)$ bits by reencoding the information set and using the new codeword to locate the errors in the received vector [10]. Research carried out on the information set decoding approach has led to different implementations and modifications for its applications on linear block codes [27–32].

2.5.3 Hard-Decision Decoding Techniques

2.5.3.1 Reed-Solomon Codes

A typical hard-decision decoder for RS codes can correct errors up to:

$$\lambda = \left\lfloor \frac{d_{\min} - 1}{2} \right\rfloor = \left\lfloor \frac{n - k}{2} \right\rfloor. \tag{2.27}$$

Decoding algorithms for RS codes that maintain the λ error correcting bounds are given the name Minimum distance decoding (MDD) algorithms [17]. Hard-decision decoding algorithms for RS codes based on the minimum distance include

the the Euclidean algorithm [5], Berlekamp-Massey (BM) algorithm [4] [33] and the Berlekamp-Welch algorithm [34].

2.5.3.2 LDPC Codes

Hard-decision decoding algorithms for binary LDPC codes are generally referred to as bit-flipping algorithms. The basis of the bit-flipping algorithms is that an incorrect bit in a codeword is contained in a large number of parity-check equations with no other incorrect bits. Each of these correct bits are used to calculate the value of the incorrect bit. The sparse nature of the H matrix assists in spreading out the bits into checks so as to reduce the chances of two parity-check equations containing the same set of codeword bits [35]. However, the bit-flipping algorithm is not perfect as the occurrence of an overlap in the parity check equations proves to be detrimental to the algorithm.

2.5.4 Soft-Decision Decoding Techniques

Most of the widely used high performance decoding algorithm utilize soft information obtained from the noisy channel to perform error correction of the received codeword. The soft-decision decoding techniques used on the received information include,

1. **Maximum Likelihood (ML) decoding:** This technique works by using a received vector r obtained by transmitting a codeword c through a noisy channel, and maximizing the probability $P(r|c)$ to obtain the decoded codeword $\widehat{c}$ as shown in (2.28)[11][35].

$$\widehat{c} = \text{argmax}_{c \epsilon \mathcal{C}} P(r|c). \tag{2.28}$$

2. **Maximum a posteriori (MAP) decoding:** This technique works by maximizing the a posteriori probability (APP) of the codeword c. The decoding works by selecting the codeword with the maximum APP probability

as shown in (2.29) [11][35].

$$\widehat{c} = \text{argmax}_{c \epsilon \mathcal{C}} P(c|r). \tag{2.29}$$

3. **Iterative decoding:** This technique decodes in an iterative manner to yield the precise approximations of $P(c_j|r_j)$ (where $1 \leq j \leq n$) by using repeated operations. These operations can involve either ML, MAP or other processes to decode the received vector[35].

In general for both cases of RS and LDPC codes hard-decision decoders have been shown to yield losses, in terms of error correction, when compared to soft-decision decoding techniques [1, 6, 8]. For example, the loss incurred by hard-decision decoders when compared to optimal soft-decision decoding is believed to be 2-3dB [6]. For this reason, soft-decision decoding algorithms for RS codes has been of particular interest of research for both practical and theoretical reasons [6, 7, 25, 26, 36–39]. Based on the gain yielded by soft-decision decoders when compared to hard-decision decoders, this research will focus on soft-decision decoding techniques and how to improve on them.

2.6 Iterative Soft-Decision Decoders

2.6.1 Belief Propagation

2.6.1.1 The Belief Propagation Algorithm for LDPC codes

The belief propagation (BP) decoding algorithm for LDPC codes is a symbol by symbol soft-in soft-out process and is also commonly referred to as the Sum product algorithm (SPA). The underlying structure that enables the efficient decoding of LDPC codes with the BP algorithms is the fact that the H matrix of an LDPC code can be represented in the form of Tanner graph [1]. In the Tanner graph representation, each column defines a variable node and each row is used to define a check node. The BP algorithm involves iteratively processing the received symbols

to improve the reliability of each decoded code based on each parity check sums computed from the hard-decisions of the received symbols based on the sparse parity check matrix H of an LDPC code. The computed reliability measures of code symbols at the end of each decoding iteration are used as input for the next iteration. This process is repeated until a certain stopping condition is attained. There are two main versions of the BP algorithm used for the decoding of binary LDPC codes are

1. The Probability-domain BP algorithm
2. The Log-domain BP algorithm

The main difference in the two versions of the BP algorithm is in the type of soft information fed into the decoder. As their names suggest, the probability-domain BP handles soft information in the form of bit probabilities while the log-domain BP algorithm handles soft information in the form of log likelihood ratios (LLR). It is important to note that this research focuses on the decoding of binary LDPC codes. Therefore, the binary version of the BP algorithms are now described.

The probability-domain BP algorithm has 2 main stages, the check node update stage and the variable node update stage. For $1 \leq i \leq (n-k)$, $1 \leq j \leq n$ and $a \in$ GF (2), the probabilities p_{ij}^a and e_{ij}^a are iteratively updated during these 2 stages. The notation p_{ij}^a is used to represent the probability that bit j of a is a, given the information calculated based on all the check nodes except for the check node i. The notation e_{ij}^a represents the probability that the syndrome check at row i of H is satisfied if bit j of a is considered fixed at a. It is important to note the values of the probabilities p_{ij}^a and e_{ij}^a are associated with each nonzero element in the H matrix[3]. The pseudo code run for iteration f of the probability-domain BP algorithm can be seen in Algorithm 1[1] [3].

The log-domain BP algorithm works in a similar way to the probability-domain BP algorithm, as it utilises both the check node update and the variable node update stages to correct the extrinsic and intrinsic information respectively. The pseudo code run for iteration f of the log-domain BP algorithm is presented in Algorithm 2 [11].

Algorithm 1: The Probability-Domain BP algorithm

Input: The received vector, r, from the channel. The probabilities $P(cb_j = 1|r_j)$ and $P(cb_j = 0|r_j)$. The maximum number of iterations f_{max} is set. Set $f = 1$.

Output: The decoded vector $\widehat{r}$.

Initialize: the probabilities p_{ij}^1 and p_{ij}^0 are set to $P(cb_j = 1|r_j)$ and $P(cb_j = 0|r_j)$ respectively for all (i, j) with $H_{i,j} = 1$.

repeat

- *Check Node Update*:
  The extrinsic information, e_{ij}, is computed as
  $$\delta e_{ij} = \prod_{j'=1, j' \neq j, H_{(i,j')}=1}^{n} \delta p_{ij'},$$
  where $\delta p_{ij} = p_{ij}^1 - p_{ij}^0$.

  Set $e_{ij}^0 = \frac{1}{2}(1 + \delta e_{ij})$ and $e_{ij}^1 = \frac{1}{2}(1 - \delta e_{ij})$

- *The Variable Node Update*:
  The intrinsic information is computed as
  $$p_{ij}^0 = \varrho_{ij} P(cb_j = 0|r_j) \prod_{i'=1, i' \neq i, H_{(i',j)}=1}^{(n-k)} e_{i'j}^0,$$

  $$p_{ij}^1 = \varrho_{ij} P(cb_j = 1|r_j) \prod_{i'=1, i' \neq i, H_{(i',j)}=1}^{(n-k)} e_{i'j}^1,$$
  where $\varrho_{ij} = p_{ij}^0 + p_{ij}^1$.

- *Updating the "Pseudoposterior" Probabilities:*
  the "pseudoposterior" probabilities, p_j^a, are given by
  $$p_j^0 = \varrho_j P(cb_j = 0|r_j) \prod_{i=1, H_{(i,j)}=1}^{(n-k)} e_{ij}^0,$$

  $$p_j^1 = \varrho_j P(cb_j = 0|r_j) \prod_{i=1, H_{(i,j)}=1}^{(n-k)} e_{ij}^0,$$
  where $\varrho_j = p_j^0 + p_j^1$.

- *The Hard-Decision Vector*
  The hard-decision vector, $\widehat{r}$, is computed as
  $$\widehat{r}_j = \begin{cases} 1 & \text{if } p_j^1 > p_j^0, \\ 0 & \text{if } p_j^1 < p_j^0. \end{cases}$$

- *Syndrome Check:*
  $S = \widehat{r} \times H^\top$.

- *Stopping Criteria:*
  if $S = 0$ **then**
  | **break**
  else
  | $f = f + 1$
  end

until $f = f_{max}$

Algorithm 2: The Log-Domain Belief Propagation Algorithm

Input: The received vector, r, from the channel. The probabilities $P(cb_j = 1|r_j)$ and $P(cb_j = 0|r_j)$. The maximum number of iterations f_{max} is set. Set $f = 1$.

Output: The decoded vector $\widehat{r}$.

Initialize: Obtain the LLR values, $L_j = log\frac{P(cb_j=1|r_j)}{P(cb_j=0|r_j)}$, based on the channel model. Set $L(p_{ij})^{(1)} = L_j$ for all (i, j) with $H_{i,j} = 1$.

repeat

- *The Check Node Update*:
  The extrinsic information, $L(e_{ij})$, is updated based on the LLR values.
  $$L(e_{ij})^{(f)} = 2\tanh^{-1}\left(\prod_{j'=1,j'\neq j,H_{(i,j')}=1}^{n}\tanh\left(\frac{L(p_{ij'})^{(f)}}{2}\right)\right).$$

- *The Variable Node Update*:
  The LLR values L_j are updated based on the extrinsic information,
  $$L(p_{ij})^{(f+1)} = L_j + \sum_{i'=1,i'\neq i,H_{(i',j)}=1}^{(n-k)} L(e_{ij})^{(f)},$$

- *Total LLR values:*
  $$L(T)_j = L_j + \sum_{i=1,H_{(i,j)}=1}^{(n-k)} L(e_{ij})^{(f)},$$

- *The Hard-Decision Vector* $\widehat{r}$
  $$\widehat{r} = \begin{cases} 1 & \text{if } L(T)_j > 0, \\ 0 & \text{if } L(T)_j < 0. \end{cases}$$

- *Syndrome Check*
  $S = \widehat{r} \times H^{\top}$.

- *Stopping Criteria:*
  if $S = 0$ **then**
  | **break**
  else
  | $f = f + 1$
  end

until $f = f_{max}$

2.6.1.2 The Adaptive Belief Propagation Algorithm

The Adaptive Belief Propagation (ABP) algorithm [6], is an extension of the belief propagation algorithm for low density parity check (LDPC) codes adapted for linear block codes defined by High Density Parity Check (HDPC) matrices (like RS codes)[40]. The ABP is a MAP decoder that can be applied at both a bit and

symbol-level for a class of nonbinary codes. In [6], the ABP algorithm was shown to significantly outperform the widely used Koetter and Vardy (KV) [7] algorithm which is considered to be a high performance soft-decision decoder for RS codes.

The bit-level ABP algorithm

To understand how the bit-level ABP algorithm works consider an (n, k) RS codeword, c. The codeword is expressed in bit form to give the vector cb. The H matrix, with the dimensions $(n-k) \times n$, has a binary image expansion performed on it to give the matrix $\mathcal{H}$. The vector cb is then transmitted through a noisy channel after being mapped onto signals using a selected modulation scheme. At the output of the channel a received vector r is obtained and is used to create an LLR vector L,

$$L(cb_j) = log\frac{P(cb_j = 1|r_j)}{P(cb_j = 0|r_j)}, \tag{2.30}$$

where $1 \leq j \leq nm$. The vector L is then fed into the ABP algorithm. For the iteration f, the ABP decoding algorithm is summarized as follows:

1. Find the absolute values of L to obtain the vector $|L|$
2. The values of the vector $|L|$ are sorted in ascending order and the swapped indices noted
3. Based on the swapped order of the indices from the sorting of the vector $|L|$, the matrix $\mathcal{H}$ is row reduced such that the least values of $|L|$ will match the identity submatrix.
4. For iteration f, belief propagation is now carried out on the vector L using the row reduced matrix $\mathcal{H}$. This is carried out to generate the extrinsic LLR vector L_e,

$$L_e(cb_j)^{(f)} = \sum_{i=1,\mathcal{H}_{ij}=1}^{(nm-km)} 2\tanh^{-1}\left(\prod_{j'=1,j'\neq j,\mathcal{H}_{i,j'}=1}^{nm} \tanh\left(\frac{L^{(f)}(cb_{j'})}{2}\right)\right). \quad (2.31)$$

5. The LLR bit reliabilities are then updated using

$$L(cb_j)^{(f+1)} = L(cb_j)^{(f)} + \eta L_e(cb_j)^{(f)}, \quad (2.32)$$

   where η is the damping coefficient and is of the value $0 < \eta \leq 1$

6. Hard-decision detection is performed on the vector $L^{(f+1)}$ based on (2.33). The hard-decision vector is then used to calculate the syndrome.

$$\widehat{r} = \begin{cases} 1 & \text{if } L(cb_j)^{(f+1)} > 0, \\ 0 & \text{if } L(cb_j)^{(f+1)} < 0. \end{cases} \quad (2.33)$$

7. The iterative process continues until the syndrome is satisfied ($S = 0$) or until the point when the maximum number of pre set iterations is reached.

The symbol-level ABP algorithm

The symbol-level ABP decoder works in very similar way to the bit-level ABP, with the main difference being in how the parity check matrix is adapted. The bit-level ABP performs its adaptation on the binary image of the parity check matrix, $\mathcal{H}$, based on the bit reliability information. This is different from the symbol-level ABP which performs it adaptation on the parity check matrix, H, based on the symbol reliability information. Once the H matrix is adapted, the binary image expansion is carried out to create the matrix $\mathcal{H}$. However, it is important to note that the adaptation of the matrix H is performed once at the initialisation of the algorithm before (2.31) and (2.32) are iteratively repeated, using the bit LLR values and the matrix $\mathcal{H}$, until the syndrome is satisfied or until the point when the maximum number of pre set iterations is reached.

Due to the symbol-level ABP only adapting the H matrix once, it is less complex

when compared to the bit-level ABP which transforms the binary matrix $\mathcal{H}$ every iteration based on the soft information in the vector L^f. However, the reduced complexity of the symbol-level ABP comes at the cost of a reduced performance in terms of error correction when compared to the bit-level ABP as seen in [6].

2.6.2 The Parity Check Transformation Algorithm

The Parity check Transformation Algorithm (PTA) algorithm is a symbol-wise soft-decision decoder that utilizes the reliable information from the channel to transform the H matrix by performing row operations. The PTA is an iterative decoder that works by repeatedly correcting reliability values to obtain the decoded message. In [8] [41], The PTA was shown to outperform the widely used KV algorithm. However, these gains in decoding performance of the PTA come at the cost of the algorithm running numerous iterations during the decoding of the received vector, especially for low SNR cases.

To understand how the PTA algorithm works, consider a codeword c of an (n, k) RS code whose symbols are transmitted across a noisy channel. The received vector r is outputted from the channel and is used to create a reliability matrix β. The matrix β is then fed into the PTA decoder. The PTA decoder works iteratively and performs the following steps during each iteration.

1. *Sorting the Symbol Reliability:* A vector containing the maximum reliabilities in each column of the matrix β is obtained as shown in [8] and is represented as.

$$A = [A_1, A_2, ..., A_j] \tag{2.34}$$

   where $A_j = \text{argmax}(\beta_j)$. The reliability vector is then sorted in ascending order with the original indices being noted and stored in the vector Y as shown in (2.35).

$$Y = [Y_1, Y_2, ..., Y_n] \tag{2.35}$$

   The k highest values of the vector Y are used to represent the reliable symbols and their indices and stored in the vector $\mathcal{K}$. The remaining $(n - k)$ values

are assumed to be unreliable and the indices are stored in the vector $\mathcal{U}$ as shown in (2.36)

$$\begin{aligned} \mathcal{U} &= [Y_1, I_2, ..., Y_{(n-k)}], \\ \mathcal{K} &= [Y_{(n-k)+1}, ..., Y_n]. \end{aligned} \tag{2.36}$$

2. *Transforming the H matrix:* The H matrix is then transformed by performing row reduction operations, such that the identity submatrix matches the indices in $\mathcal{U}$ and the dense parity submatrix matches the indices in $\mathcal{K}$. The transformed parity check matrix is denoted as $H^{'}$.

3. *Finding the syndrome checks:* Hard-decision detection is carried out on the matrix β to obtain the vector $\widehat{c}$. The syndrome vector, S, is then obtained by finding the scalar product of $\widehat{c}$ and the rows $H^{'}$ as shown in (2.37)

$$S_i = \widehat{c} \cdot H_i^{'}, \tag{2.37}$$

where $1 \leq i \leq (n-k)$ and $H_i^{'}$ represents the rows of $H^{'}$.

4. *The Correction Step:* For each corresponding row of $H^{'}$ if $S_i \neq 0$, the participating symbol reliabilities indexed by $\mathcal{U}$ and $\mathcal{K}$ are subtracted by a correction factor, δ and $\delta/2$ respectively. This is performed to reduce the chances of these indices being selected and stored in the vector A for the next iteration. For the case when $S_i = 0$ on a given row of $H^{'}$, the participating symbol reliabilities indexed by $\mathcal{U}$ and $\mathcal{K}$ added by δ and $\delta/2$ respectively. This reinforces the selection of these reliabilities to be store in the vector A for the next iteration. The corrections on the selected symbol reliabilities carried out during each syndrome check are summarized in (2.38).

$$A_j = \begin{cases} A_{\mathcal{U}_i} + \delta & \text{if } S_i = 0, \\ A_{\mathcal{K}} + \delta/2, & \\ & \\ A_{\mathcal{U}_i} - \delta & \text{if } S_i \neq 0. \\ A_{\mathcal{K}} - \delta/2, & \end{cases} \tag{2.38}$$

The corrected reliabilities in the vector A are then updated to their corresponding positions in β, in preparation for the next iteration.

5. *Stopping Condition:* The procedure is repeated until all the values in the syndrome vector $S = 0$ or a negative value is updated to the matrix β.

2.7 Algorithm Complexity

The complexity analysis for the algorithms is represented as a measure of the computational complexity cost. The computational cost is expressed in terms of the *Time complexity*. The time complexity provides a good measure of the complexity cost as an order of the rate at which the number of operations increase as the input size gets bigger. This order as referred to is the big O notation ($\mathcal{O}$). The use of the big O notation in reading the time complexity provides a measure asymptotic efficiency. That is, big O notation is a measure of how the running time of an algorithm increases with the size of the input in the limit, as the size of the input increases infinitely [42]. This means the time complexity is a representation of the worst-case number of operations required by an algorithm to compute a given function. For example, assume an algorithm with the total number of operations given in (2.39)

$$(M \times N^3) + (N^2 \div N) + (M \times N) + 9. \tag{2.39}$$

From the definition of the worst-case number of operations, the time complexity of the algorithm can then be represented as the order

$$\mathcal{O}(M \times N^3). \tag{2.40}$$

This is because, as the term N^3 grows to infinity, the total complexity of the algorithm represented in (2.39) is dominated by the function represented by $(M \times N^3)$. When comparing algorithms, the best option is often the one that is asymptotically more efficient for all but very small inputs [42]. In FER, the complexity cost of the decoding algorithms is often a measure of the time complexity[12, 43, 44].

When analysing the complexity cost, noting the number of 'additions/subtractions' $(+/-)$ and the 'multiplication/divisions' $(\times/\div)$ can also be used to measure if the algorithm is computationally intense. Algorithms with more $(\times/\div)$ than $(+/-)$ are considered to have a higher computational complexity cost.
Additionally, this research will also measure the complexity cost as a function of the total number of the operations and the average number of iterations required to decode to supplement the time complexity analysis. The extra analysis is carried out because this research aims to study the effect of a reduced convergence rate to the complexity cost of iterative decoding algorithms.

2.8 Importance of the Systematic Structure of the H Matrix in this Research for the Decoding Process

To better understand the foundation of the decoder, the importance of the systematic structure to the decoding scheme and how it relates to (2.16) is discussed. Assume the H matrix for a given code in the systematic form is expressed as

$$H = [I \;\; Q], \tag{2.41}$$

and has the form shown in (2.42)

$$H = \begin{bmatrix} \alpha^0 & 0 & 0 & 0 & 0 & \alpha^8 & \alpha^{14} & \alpha^3 \\ 0 & \alpha^0 & 0 & 0 & 0 & \alpha^4 & \alpha^1 & \alpha^5 \\ 0 & 0 & \alpha^0 & 0 & 0 & \alpha^2 & \alpha^4 & \alpha^8 \\ 0 & 0 & 0 & \alpha^0 & 0 & \alpha^3 & \alpha^8 & \alpha^2 \\ 0 & 0 & 0 & 0 & \alpha^0 & \alpha^5 & \alpha^4 & \alpha^1 \end{bmatrix}. \tag{2.42}$$

A valid codeword c is defined by the matrix H is presented in 2.43

$$c = [c_1,\ c_2,\ c_3,\ c_4\ ,\ c_5,\ c_6,\ c_7,\ c_8]. \tag{2.43}$$

Because c is a valid codeword, the syndrome S is a zero vector and is found by getting the product with the transpose of H as shown in (2.44)

$$S = c \times H^{\top}. \tag{2.44}$$

Equation (2.44) can also be found by getting the scalar product of c and H as shown in (2.45) [17].

$$S_i = c \cdot H_i = \sum_{j=0}^{n} c_j \times H_{i,j} = 0, \tag{2.45}$$

where $0 \leq i \leq (n-k)$ and $0 \leq j \leq n$ represent the rows and columns of H. The expression in (2.45) can be rewritten as shown in (2.46)

$$S = [c_1, c_2, c_3, c_4, c_5, c_6, c_7, c_8] \cdot \begin{bmatrix} \alpha^0 & 0 & 0 & 0 & 0 & \alpha^8 & \alpha^{14} & \alpha^3 \\ 0 & \alpha^0 & 0 & 0 & 0 & \alpha^4 & \alpha^1 & \alpha^5 \\ 0 & 0 & \alpha^0 & 0 & 0 & \alpha^2 & \alpha^4 & \alpha^8 \\ 0 & 0 & 0 & \alpha^0 & 0 & \alpha^3 & \alpha^8 & \alpha^2 \\ 0 & 0 & 0 & 0 & \alpha^0 & \alpha^5 & \alpha^4 & \alpha^1 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}. \tag{2.46}$$

From (2.17), it can be seen that anything multiplied by a 0 in the field gives a 0. Therefore, if the symbols that do not multiply with the 0 found in the identity submatrix in the scalar product calculations are considered, the solution for the syndrome will still be the same. It can also be seen from (2.18) that any of the symbols of c that will multiply with α^0 along the diagonal of the identity submatrix will remain the same.

For each row H_i, the symbols c_i that is multiplied with a non zero element in H is referred to as a participating symbol. Therefore, the syndrome calculation in (2.46) can be found using only the participating symbols for the corresponding

indices for each row H_i as seen in (2.47)

$$[c_1, c_2, c_3, c_4, c_5, c_6, c_7, c_8] \cdot \begin{bmatrix} \alpha^0 & 0 & 0 & 0 & 0 & \alpha^8 & \alpha^{14} & \alpha^3 \\ 0 & \alpha^0 & 0 & 0 & 0 & \alpha^4 & \alpha^1 & \alpha^5 \\ 0 & 0 & \alpha^0 & 0 & 0 & \alpha^2 & \alpha^4 & \alpha^8 \\ 0 & 0 & 0 & \alpha^0 & 0 & \alpha^3 & \alpha^8 & \alpha^2 \\ 0 & 0 & 0 & 0 & \alpha^0 & \alpha^5 & \alpha^4 & \alpha^1 \end{bmatrix} \begin{array}{l} = c_1 + (\alpha^8 \cdot c_6) + (\alpha^{14} \cdot c_7) + (\alpha^3 \cdot c_8) = 0 \\ = c_2 + (\alpha^4 \cdot c_6) + (\alpha^1 \cdot c_7) + (\alpha^5 \cdot c_8) = 0 \\ = c_3 + (\alpha^2 \cdot c_6) + (\alpha^4 \cdot c_7) + (\alpha^8 \cdot c_8) = 0 \\ = c_4 + (\alpha^3 \cdot c_6) + (\alpha^8 \cdot c_7) + (\alpha^2 \cdot c_8) = 0 \\ = c_5 + (\alpha^5 \cdot c_6) + (\alpha^4 \cdot c_7) + (\alpha^1 \cdot c_8) = 0 \end{array}, \tag{2.47}$$

Applying (2.16) to each of the row equations shown in (2.47),

$$\begin{aligned} c_1 = &\ (\alpha^8 \cdot c_6) + (\alpha^{14} \cdot c_7) + (\alpha^3 \cdot c_8), \\ c_2 = &\ (\alpha^4 \cdot c_6) + (\alpha^1 \cdot c_7) + (\alpha^5 \cdot c_8), \\ c_3 = &\ (\alpha^2 \cdot c_6) + (\alpha^4 \cdot c_7) + (\alpha^8 \cdot c_8), \\ c_4 = &\ (\alpha^3 \cdot c_6) + (\alpha^8 \cdot c_7) + (\alpha^2 \cdot c_8), \\ c_5 = &\ (\alpha^5 \cdot c_6) + (\alpha^4 \cdot c_7) + (\alpha^1 \cdot c_8). \end{aligned} \tag{2.48}$$

In summary, because of the systematic structure of H, it is possible to find an information set of symbols that match the identity submatrix by finding the scalar product of the rows in H and the corresponding participating symbols indices of c.

The decoding approach presented in this Section is important to this research as it provides the basis on how the information set is used in the decoding process. However, for this decoding approach to work all the symbols in the information set have to be correct. This is very hard to determine due to the complete dense structure of the parity submatrix in H. This means that a wrong symbol in the information set $[c_6, c_7, c_8]$ would incorrectly decode the subset $[c_1, c_2, c_3, c_4, c_5]$ using (2.48).

2.9 Framework for Simulations

This section provides the scientific methods and approaches used in performing the research simulations and presentation of results. This Section also looks into the approaches used in obtaining the soft information that are derived from the

received vector at the output of the channel. The approaches to obtaining the soft information differ depending the channel model and the modulation scheme used. Obtaining of the soft information is very crucial for this research as this is what is fed into the decoding algorithms.

2.9.1 The Channel Model

Simulations tested for all the algorithms are run for both RS and QC-LDPC codes transmitted through an AWGN channel. With this regard, this section covers the channel model used in this research.

For an AWGN channel, the general expression for the received coded sequence can be obtained as shown in (2.49)

$$r = v + \mathcal{N}, \tag{2.49}$$

where v represents the transmitted (modulated) coded sequence of length n with the constellation representation of the symbols or bits and $\mathcal{N}$ represents additive white noise. The received vector r has the same length as v.

When transmitting signals through an AWGN channel using a $\mathcal{M}$-QAM modulation scheme (2.49) can be rewritten as

$$r = v\Gamma + N, \tag{2.50}$$

where Γ represents the normalising factor and is obtained as shown in (2.51)

$$\Gamma = \frac{1}{\sqrt{\frac{2}{3}(\mathcal{M}-1)}}. \tag{2.51}$$

For an Additive White Gaussian noise, the probability distribution function can be summarised as in (2.52)[1]

$$P(v_j = a|r_j) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{\frac{-(r_j - v_j)^2}{2\sigma^2}}, \tag{2.52}$$

where $a \in \{0, 1\}$, $0 \leq j \leq n$, the variance $\sigma = \frac{N_0}{2}$ and each element v_j is independent, identically distributed with zero mean. N_0 represents the noise spectral density.

2.9.2 Obtaining Bit Probabilities and Creation of the Reliability matrix using a Binary Phase Shift Key (BPSK) Modulation Scheme

In this section the approach used to obtain the reliabilities for the binary codewords, used in the QC-LDPC tests, transmitted using a BPSK modulation scheme is discussed. The technique used to obtain the bit probabilities is similar to the approach presented in [1]. Consider a binary codeword c of length n of the form.

$$c = [c_1, c_2, ..., c_n] \tag{2.53}$$

The codeword c is then mapped onto a binary phase shift keyed (BPSK) signal constellation to get the transmission vector v which also has the length n. For $1 \leq j \leq n$, the transmission signal vector v can be represented as a bipolar sequence. That is, if the bit $c_j = 1$ then it is mapped onto $v_j = 1$ and if the bit $c_j = 0$ then it is mapped onto $v_j = -1$. The received vector r is obtained from v after being transmitted through an AWGN using (2.49) and is represented as

$$r = [r_1, r_2, ..., r_n] \tag{2.54}$$

The channel posterior probability that the bit $c_j = 1$ given the information in r_j can be obtained by rewriting (2.52) to give

$$P(c_j = 1|r_j) = \frac{1}{1 + \exp(-2r_j/\sigma^2)} \tag{2.55}$$

It is assumed that $P(c_j = 1) = P(c_j = 0) = \frac{1}{2}$ [1]. Therefore, the channel posterior probability that the bit $c_j = 0$ given the information in r_j can be calculated using

(2.56).

$$P(c_j = 0|r_j) = 1 - P(c_j = 1|r_j) \tag{2.56}$$

The probabilities can also be used to find the Log Likelihood Ratios (LLR). The general equation to obtain the LLR value is expressed as (2.57) [11]

$$LLR_{(c_j)} = \log\left(\frac{P(c_j = 1|r_j)}{P(c_j = 0|r_j)}\right). \tag{2.57}$$

Using (2.55) and (2.56), The LLR values obtained from the calculation in (2.57) can be rewritten as (2.58)

$$LLR_{(c_j)} = \frac{2r_j}{\sigma^2}. \tag{2.58}$$

2.9.3 Obtaining Bit Probabilities and Creation of the Reliability matrix using a 16-QAM scheme

The RS codes used for the simulations are created in the field GF(2^4). The codewords generated in this field are of length 15 and are considered to be short codes. The main reason this research focuses on short codes is so that comprehensive tests can be run on the algorithm, under different conditions, with the available computational power.

The experimental setup for the simulations involves transmission of a codeword through an Additive White Gaussian Noise (AWGN) channel using a Quadrature Amplitude Modulation (QAM) scheme. The main reason the QAM modulation scheme is selected for this research is because it is more power efficient when compared to the Pulse Amplitude Modulation (PAM) and Phase Key Modulation (PSK) scheme [45]. It is also important to note that $\mathcal{M}$-QAM yields a better performance for the same SNR value per symbol when compared to $\mathcal{M}$-PSK [45] [17]. Rectangular $\mathcal{M}$-QAM signals are also used in the mapping of the symbol constellations because, for $\mathcal{M} \geq 16$, the average transmitted power required to attain a given minimum distance is only slightly greater than the average power required for the best $\mathcal{M}$-QAM signal constellation [45].

2.9.3.1 Gray mapping for the 16-QAM constellations

Each constellation point on a 16-QAM modulation scheme is used to represent an element in the field GF(2^4). Each element in this field can be represented using four bits which are denoted as b_0,b_1,b_2,b_3. The constellations are mapped in such a way that the bits b_0,b_1 represent the points on the real axis and the bits b_2,b_3 represents the points on the imaginary axis. The representation for the mapping can be seen in Table 2.1.

TABLE 2.1: Bit representation for the In phase and Quadrature components in the 16-QAM modulation scheme .

b_0	b_1	$\mathcal{I}$
0	0	1
0	1	-1
1	0	3
1	1	-3

b_2	b_3	$\mathcal{Q}$
0	0	1
0	1	-1
1	0	3
1	1	-3

where $\mathcal{I}$ represent in phase (real) components and $\mathcal{Q}$ represents quadrature (imaginary) components. This representation is used to create a constellation mapping points as shown in Table 2.2.

TABLE 2.2: Bit mapping used to create symbol positions on 16-QAM constellations.

symbol	b_0, b_1, b_2, b_3	constellation mapping	symbol	b_0, b_1, b_2, b_3	constellation mapping
0	0 0 0 0	1+1j	8	1 0 0 0	3+1j
1	0 0 0 1	1-1j	9	1 0 0 1	3-1j
2	0 0 1 0	1+3j	10	1 0 1 0	3+3j
3	0 0 1 1	1-3j	11	1 0 1 1	3-3j
4	0 1 0 0	-1+1j	12	1 1 0 0	-3+1j
5	0 1 0 1	-1-1j	13	1 1 0 1	-3-1j
6	0 1 1 0	-1+3j	14	1 1 1 0	-3+3j
7	0 1 1 1	-1-3j	15	1 1 1 1	-3-3j

The Gray code constellation mapping for the 16-QAM modulation scheme based on Table 2.1 and Table 2.2 is illustrated in Fig 2.1.

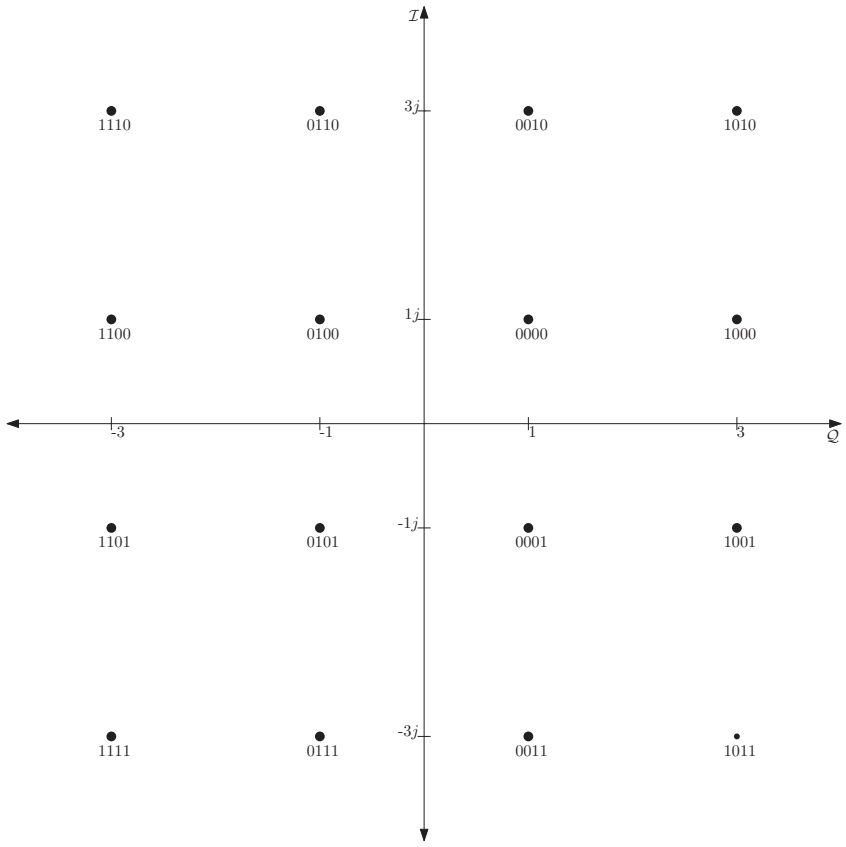


FIGURE 2.1: 16-QAM constellation plot with gray coded mapping.

2.9.3.2 Bit Probabilities for a 16-QAM Modulation Scheme

Once the RS codeword is transmitted using a 16-QAM modulation scheme through an AWGN channel, a received vector r is obtained using (2.52). Each element r_j of the vector r has the form shown in (2.59)

$$r_j = r_{re_j} + r_{im_j}, \tag{2.59}$$

where r_{re_j} represents the real component of the element r_j and r_{im_j} represents the imaginary component of each r_j.

From Table 2.1 and Fig. 2.1 it can be seen from the 16-QAM constellation mapping that the bit b_0 toggles from 0 to 1 along the real axis, i.e when

- $b_0 = 0$, toggles between 1 and -1 on the real axis of constellation mappings

- $b_0 = 1$, toggles between 3 and -3 on the real axis of constellation mappings

From this, the conditional probability of that $b_0 = 0$ given the received element r_j can be derived from (2.52) using the values -1 and 1 respectively and is given as

$$P(b_{(j,0)} = 0|r_j) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{\frac{-(rre_j - 1\Gamma)^2}{2\sigma^2}} + \frac{1}{\sqrt{2\pi\sigma^2}} e^{\frac{-(rre_j + 1\Gamma)^2}{2\sigma^2}} \tag{2.60}$$

Applying (2.60) to when $b_0 = 1$ for the values 3 and -3 respectively gives

$$P(b_{(j,0)} = 1|r_j) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{\frac{-(rre_j - 3\Gamma)^2}{2\sigma^2}} + \frac{1}{\sqrt{2\pi\sigma^2}} e^{\frac{-(rre_j + 3\Gamma)^2}{2\sigma^2}} \tag{2.61}$$

Using the approach in (2.60) and (2.61), the equation for the bit probabilities for the other bits can be calculated. Firstly, the positions in which the respective bits toggle on the constellation mappings from the Fig. 2.1 and Table 2.1 are determined. Once the positions are obtained, the conditional probabilities can be calculated as follows:
for b_1, when

- $b_1 = 0$, toggles between 1 and 3 on the real axis of constellation mappings
- $b_1 = 1$, toggles between -1 and -3 on the real axis of constellation mappings

$$P(b_{(j,1)} = 0|r_j) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{\frac{-(rre_j - 1\Gamma)^2}{2\sigma^2}} + \frac{1}{\sqrt{2\pi\sigma^2}} e^{\frac{-(rre_j - 3\Gamma)^2}{2\sigma^2}} \tag{2.62}$$

$$P(b_{(j,1)} = 0|r_j) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{\frac{-(rre_j + 1\Gamma)^2}{2\sigma^2}} + \frac{1}{\sqrt{2\pi\sigma^2}} e^{\frac{-(rre_j + 3\Gamma)^2}{2\sigma^2}} \tag{2.63}$$

for b_2, when

- $b_2 = 0$, toggles between 1 and -1 on the imaginary axis of constellation mappings
- $b_2 = 1$, toggles between 3 and -3 on the imaginary axis of constellation mappings

$$P(b_{(j,2)}=0|r_j)=\frac{1}{\sqrt{2\pi\sigma^2}}e^{\frac{-(r_{im_j}-1\Gamma)^2}{2\sigma^2}}+\frac{1}{\sqrt{2\pi\sigma^2}}e^{\frac{-(r_{im_j}+1\Gamma)^2}{2\sigma^2}} \tag{2.64}$$

$$P(b_{(j,2)}=0|r_j)=\frac{1}{\sqrt{2\pi\sigma^2}}e^{\frac{-(r_{im_j}-3\Gamma)^2}{2\sigma^2}}+\frac{1}{\sqrt{2\pi\sigma^2}}e^{\frac{-(r_{im_j}+3\Gamma)^2}{2\sigma^2}} \tag{2.65}$$

for b_3, when

- $b_3 = 0$, toggles between 1 and 3 on the imaginary axis of constellation mappings
- $b_3 = 1$, toggles between -1 and -3 on the imaginary axis of constellation mappings

$$P(b_{(j,3)}=0|r_j)=\frac{1}{\sqrt{2\pi\sigma^2}}e^{\frac{-(r_{im_j}-1\Gamma)^2}{2\sigma^2}}+\frac{1}{\sqrt{2\pi\sigma^2}}e^{\frac{-(r_{im_j}-3\Gamma)^2}{2\sigma^2}} \tag{2.66}$$

$$P(b_{(j,3)}=0|r_j)=\frac{1}{\sqrt{2\pi\sigma^2}}e^{\frac{-(r_{im_j}+1\Gamma)^2}{2\sigma^2}}+\frac{1}{\sqrt{2\pi\sigma^2}}e^{\frac{-(r_{im_j}+3\Gamma)^2}{2\sigma^2}} \tag{2.67}$$

It is important to note that the positions for b_0 and b_2 are identical except for the fact that b_0 is in the real axis and b_2 is in the imaginary axis. The same applies for b_1 and b_3. Therefore, the conditional probabilities for these bits have the same equations that differ only by using the real and imaginary components depending on the bit.

The Bit-Level β Matrix for a 16-QAM Modulation Scheme

The bit probabilities are used to create the bit-level β matrix used for implementation for both the proposed decoder and the bit-level PTA algorithm. The bit-level β matrix for the 16-QAM modulation scheme, where $n = 15$, is expressed as shown in (2.68)

$$\beta=\begin{bmatrix} P(b_{(0,0)}=0|r_1), & P(b_{(0,1)}=0|r_1), & ..., & P(b_{(15,2)}=0|r_{15}), & P(b_{(15,3)}=0|r_{15}), \\ P(b_{(0,0)}=1|r_1), & P(b_{(0,1)}=1|r_1), & ..., & P(b_{(15,2)}=1|r_{15}), & P(b_{(15,3)}=0|r_{15}) \end{bmatrix}. \tag{2.68}$$

The Bit-Level LLR Values for a 16-QAM Modulation Scheme

Using the equations for the positions of the bits and (2.57), the full list of general equations for the LLRs used in the implementation of the ABP algorithm are given as

$$LLR(b_{(j,0)}) = ln\left(\frac{e^{\frac{-(r_{re_j}+3\Gamma)^2}{2\sigma^2}} + e^{\frac{-(r_{re_j}-3\Gamma)^2}{2\sigma^2}}}{e^{\frac{-(r_{re_j}+1\Gamma)^2}{2\sigma^2}} + e^{\frac{-(r_{re_j}-1\Gamma)^2}{2\sigma^2}}}\right), \tag{2.69}$$

$$LLR(b_{(j,1)}) = ln\left(\frac{e^{\frac{-(r_{re_j}+1\Gamma)^2}{2\sigma^2}} + e^{\frac{-(r_{re_j}+3\Gamma)^2}{2\sigma^2}}}{e^{\frac{-(r_{re_j}-1\Gamma)^2}{2\sigma^2}} + e^{\frac{-(r_{re_j}-3\Gamma)^2}{2\sigma^2}}}\right), \tag{2.70}$$

$$LLR(b_{(j,2)}) = ln\left(\frac{e^{\frac{-(r_{im_j}+3\Gamma)^2}{2\sigma^2}} + e^{\frac{-(r_{im_j}-3\Gamma)^2}{2\sigma^2}}}{e^{\frac{-(r_{im_j}+1\Gamma)^2}{2\sigma^2}} + e^{\frac{-(r_{im_j}-1\Gamma)^2}{2\sigma^2}}}\right), \tag{2.71}$$

$$LLR(b_{(j,3)}) = ln\left(\frac{e^{\frac{-(r_{im_j}+1\Gamma)^2}{2\sigma^2}} + e^{\frac{-(r_{im_j}+3\Gamma)^2}{2\sigma^2}}}{e^{\frac{-(r_{im_j}-1\Gamma)^2}{2\sigma^2}} + e^{\frac{-(r_{im_j}-3\Gamma)^2}{2\sigma^2}}}\right). \tag{2.72}$$

2.9.3.3 Creating the Reliability Matrix (β) at A Symbol-Level Using a 16-QAM Modulation Scheme

The construction of the reliability matrix for a RS code used in the symbol-level 16-QAM decoder implementations and simulations is now discussed. The Distance Metric (DM) presented in [46] is used. Assume an (n,k) RS code C mapped onto a 16-QAM modulation scheme to get the transmission signal v. The signal v is transmitted through a noisy channel to obtain the vector r as shown in (2.50). The first step to creating the reliability matrix involve using the Euclidean distance equation shown in (2.73) to create the matrix B.

$$B = \sqrt{(real(r_j) - real(v_{\alpha^z})\Gamma)^2 + (imag(r_j) - imag(v_{\alpha^z})\Gamma)^2}, \tag{2.73}$$

where B is a matrix with the dimensions $2^m \times n$ and v_{α^z} represents the transmission constellation for the GF(2^m) symbol α^z. The notation $z = -\infty$ or $0 \leq z \leq 2^m - 2$

and $0 \leq j \leq n$ are used to represent the rows and columns of B respectively. The reliability matrix β is then obtained by applying (2.74)

$$\beta = \exp(-B). \tag{2.74}$$

2.9.4 Simulation Model

2.9.4.1 Symbol Error Probability

In spite of the high performance error correction implementations at the demodulator, some errors can still go undetected. In this section, the probability of error for a 16-QAM modulation is discussed [47] [48].

For QAM schemes with Gray mapping in Gaussian noise, the probabilities can be expressed using the error function. The error function, erfc(y), for an AWGN channel can be represented as [16] [11]

$$\text{erfc}(y) = \frac{1}{2\pi} \int_{y}^{\infty} e^{\mu^2/2} d\mu. \tag{2.75}$$

Consider the 16-QAM alphabet based on Table 2.2 shown in Fig. 2.2

From Fig .2.2, there are 3 distinct decision regions. The first decision region of the four inside points (o_0, o_1, o_4, o_5) are considered. In this case, the probability of error is

$$P(\text{error}|o_{\hat{i}} \text{ in the inside}) = 2\text{erfc}\left(\Gamma\sqrt{\frac{E_s}{N_0}}\right) - \text{erfc}^2\left(\Gamma\sqrt{\frac{E_s}{N_0}}\right), \tag{2.76}$$

where $0 \leq \hat{i} \leq 15$. The second decision region are points in the corner regions $(o_{10}, o_{11}, o_{14}, o_{15})$. The probability of error in this region is expressed as

$$P(\text{error}|o_{\hat{i}} \text{ in the corner}) = \text{erfc}\left(\Gamma\sqrt{\frac{E_s}{N_0}}\right) - \frac{1}{4}\text{erfc}^2\left(\Gamma\sqrt{\frac{E_s}{N_0}}\right). \tag{2.77}$$

The third decision region are located at the points that are not in the inside or at the corner $(o_2, o_3, o_6, o_7, o_8, o_9, o_{12}, o_{13})$. The probability of error in this region

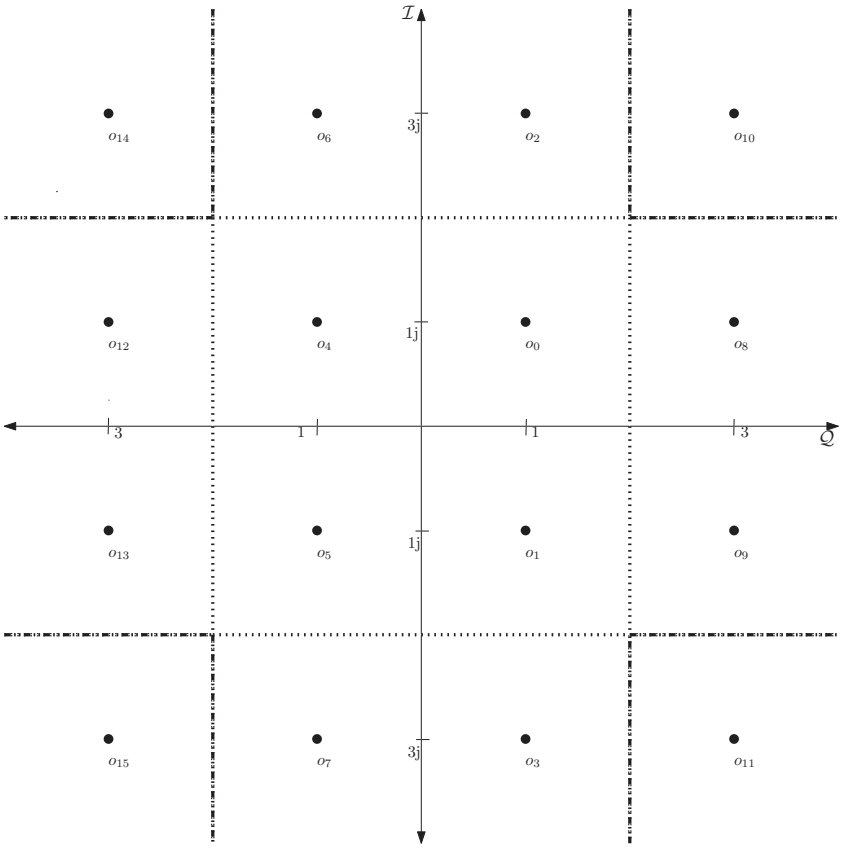


FIGURE 2.2: 16-QAM alphabet and the corresponding decision regions.

is computed as

$$P(\text{error}|o_{\hat{i}} \text{ not inside or corner}) = \frac{3}{2}\text{erfc}\left(\sqrt{\frac{E_s}{10N_0}}\right) - \frac{1}{2}\text{erfc}^2\left(\Gamma\sqrt{\frac{E_s}{N_0}}\right). \quad (2.78)$$

All the symbols (4 in the middle, 4 in the corner and the remaining 8) are assumed to be equally likely [48]. Therefore, the total probability of symbol error for a 16-QAM modulation scheme is represented as [47]

$$\begin{aligned} P(\text{error}|16-QAM) = (4 \times P(\text{error}|o_{\hat{i}} \text{ in the inside})) + (4 \times P(\text{error}|o_{\hat{i}} \text{ in the corner})) \\ + (8 \times P(\text{error}|o_{\hat{i}} \text{ not inside or corner})). \end{aligned} \quad (2.79)$$

To obtain the joint symbol error rate, the average error is calculated as shown in

$$\frac{P(\text{error}|16-QAM)}{16} \quad (2.80)$$

The probability of symbol error in (2.80) can be further reduced to [47] [48]

$$P(\text{error}|16-QAM) = 2\left(1-\frac{1}{\sqrt{16}}\right)\text{erfc}\left(\Gamma\sqrt{\frac{E_s}{N_0}}\right) - \left(1-\frac{2}{\sqrt{16}}+\frac{1}{16}\right)\text{erfc}^2\left(\Gamma\frac{E_s}{N_0}\right) \quad (2.81)$$

2.9.4.2 Simulation setup

The error correction performance are measured in terms of Symbol Error Rate (SER) and are tested across a range of signal to noise ratios (SNR). The SNR values are measured in terms E_s/N_0 for the SER simulations, where E_s represents the average energy per transmitted symbol. The value of $N_0 = 1$ for all the simulations and represents the noise spectral density.

In order to test the simulation set up, 10,000 random uncoded codewords with a length of 15 symbols each are transmitted through a 16QAM modulation scheme. The soft information is decoded using hard decision detection based on maximum likelihood. This test is carried out to ascertain if the results run using the simulation setup match the theoretical results obtained using (2.81). The received vector outputted from the channel is used to compute the different representations of soft information computed presented in Section 2.9.3.2 and Section 2.9.3.3. The pseudo code of the simulation set for the uncoded transmission is found in Algorithm 3.

Algorithm 3: Pseudo code for the symbol error probability (Uncoded) curve

Initialize: Set SNR = 0. Set $c_{\text{count}} = 0$, where c_{count} represents the codeword count. Create vector $Nerr$. The vector $Nerr$ has the length of the SNR range (from 0 to 20)

Input: Empty vector $Nerr$.

Output: Vector $Nerr$ filled with the total number of errors for each SNR value.

repeat

  repeat

  - *Generate codeword*
    A random codeword, c, of length 15 in the field GF(2^4) is created.
  - *Modulation*:
    The 15 symbols in c are mapped to their respective constellation points based on the Gray mapping for a 16-QAM modulation scheme illustrated in Fig .2.1
  - *Transmission*:
    The modulated signal is then transmitted through an AWGN channel using (2.50)
  - *Output of the channel*
    The received vector, r, is then used to create
    1. The bit-level probability matrix,β_1, as shown in (2.68).
    2. The LLR values, L, as shown in (2.69), (2.70), (2.71), (2.72).
    3. The symbol level reliability matrix,β_2, as shown (2.74)
  - *The Hard-Decision Vector*
    The hard-decision vector is computed as:
    - For β_1 and β_2 use (2.28)
    - For L use (2.33)
  - *Error check*
    The hamming distance is computed for β_1, L and β_2 as $d_h(cb, \widehat{cb})$, $d_h(L, \widehat{L})$, $d_h(c, \widehat{c})$ respectively
  - *Storing the errors:*
    $Nerr_{\beta_1}(\text{SNR}) = Nerr_{\beta_1}(\text{SNR}) + d_h(cb, \widehat{cb})$.
    $Nerr_{L}(\text{SNR}) = Nerr_{L}(\text{SNR}) + d_h(L, \widehat{L})$.
    $Nerr_{\beta_2}(\text{SNR}) = Nerr_{\beta_2}(\text{SNR}) + d_h(c, \widehat{c})$.

  until $c_{count} = 10000$

  SNR = SNR+1

until $SNR = 20$

The outcome for the tests comparing the simulated results for the uncoded codewords generated using Algorithm 3 with the theoretical limit for the probability of symbol error can be seen Fig. 2.3.

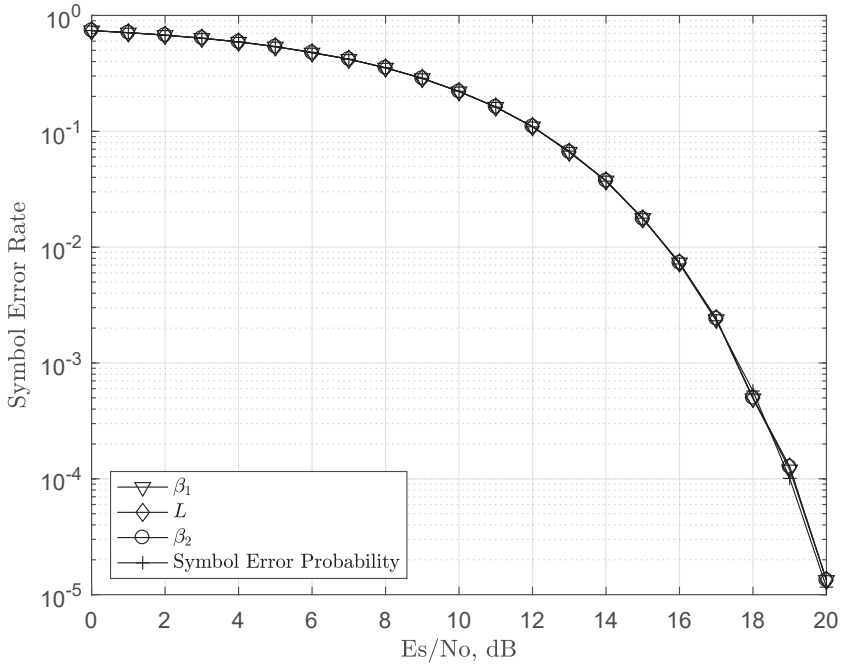


FIGURE 2.3: Uncoded simulation graphs using different types soft information

It can be seen in from Fig. 2.3 that the uncoded symbol error graphs, for the different types of the soft information, generated from the simulation setup matches the theoretical graph for the uncoded symbol error probability. The results in Fig. 2.3 show that the simulation setup presented in this section is suitable for the carrying out the tests to measure the error correction performance.

2.10 Summary

This chapter provides the core pillars and foundations which are used in the development of the ideas and contributions presented in the following chapters. It is important to also note Section 2.8 focuses on the importance of the systematic structure of linear block codes and how it applies in obtaining the information set for the proposed decoding technique. Section 2.9 looks into the setup used in

running of the simulations and obtaining the results for tests carried out during the research. This section is important because the conclusions arrived at during the research are closely linked to the results obtained from the simulations run on this setup.

CHAPTER 3

Literature Review

3.1 Introduction

This chapter presents the literature survey that has be carried out on iterative decoding schemes when applied to RS codes on both a bit-level and symbol-level. The literature survey mainly focuses on relevant papers with regards to the implementations of the proposed iterative soft-input soft-output decoding schemes when applied to RS codes. Literature survey on the systematic construction of QC-LDPC codes is also carried out in this chapter.

3.2 Information Set Decoding

Information set decoding was first presented by Prange [9] for the decoding of Cyclic codes. Since then, the algorithm has had different modifications for application in error correction of linear block codes [10, 28–30] and Cryptography [27, 31]. In all its forms, information set decoding uses k linearly independent bits or symbols from the received vector to re-encode a unique codeword during the decoding process [10]. The k symbols or bits used to obtain the unique codeword are referred to as the information set. In most cases, information set decoders make use of either the G or the H matrix to re-encode the unique codeword.
In the original implementation of the information set decoding approach presented in [9], Prange assumes a k-dimensional subspace that exists in the n-dimensional vector space of the code $\mathcal{C}$ over the finite field GF(q^m). This k-dimensional subspace is denoted as $\mathcal{C}_k$. When a codeword c of the code $\mathcal{C}$ is transmitted through

a noisy channel, a received vector r is obtained at the output of the channel. The information set is then defined as the any set of k symbols of $\mathcal{C}_k$ that match a subvector of symbols obtained from the hard-decision vector $\widehat{r}$. The information set decoding technique is used to construct the closest codeword $\widehat{c}$ from the subvector. If the distance $d_h(\widehat{r}, \widehat{c})$ is small enough then $\widehat{c}$ is taken as the decoded codeword. This hard-decision implementation of obtaining the information sets was used in the decoding of Cyclic codes.

In [49] the approach to information set decoding is updated to work with soft information. The decoding technique presented in [49] incorporates Ordered Statistics Decoding (OSD) and reprocessing to obtain a list of candidate codewords from the information sets. The OSD approach used in [49] involves sorting the probabilities of the received vector outputted from the channel in an ordered sequence. The sequence of the soft information in the received vector is sorted in a decreasing order. The G matrix is then modified based on the ordered sequence to form the matrix G^+. The k symbols indexed by the most reliable soft information are set to match the linearly independent columns (identity submatrix) of G^+. The first candidate codeword, $\widehat{c}$, is then computed using the information set obtained from hard decisions made on the k symbol indexes matching most reliable information and the matrix G^+. This codeword is considered to be a hard-decision decoded estimate. Reprocessing is carried out on $\widehat{c}$ to obtain the rest of the candidate codewords required for the list decoding stage of the decoder. For $i < \ell < k$, order-ℓ reprocessing is defined as all possible ℓ permutations of the information set of bits in $\widehat{c}$. Each permutation of the information set is used in the reconstruction of a candidate codeword. The squared Euclidean distance is then obtained between ordered received vector and the candidate codeword. The candidate codeword with the smallest distance is taken as the decoded codeword. In [49], this decoding technique is said to yield a good error correction performance when applied to different classes of binary codes including Golay codes, Reed-Muller codes and extended BCH codes. However, the complexity of the decoding process increases with larger values of k. This is because the decoder requires $\sum_{\imath=0}^{\ell} \binom{k}{\imath}$ candidate computations in order to carry out the order-ℓ decoding [49].

Research carried out in [28] builds on the work performed in [49] to improve on the order reprocessing technique used to obtain the information set. The process of computing the first candidate codeword, $\widehat{c}$, is identical to the OSD method presented in [49]. The full list of candidate codewords are then obtained through the construction of different information sets by replacing subsets of positions in the most reliable indexes with positions outside the original information set. This decoding technique is shown shown to yield a performance gain in terms of error correction when applied to extended BCH codes and compared to the Chase decoder in [28]. The performance gain in error correction is more pronounced when the larger order reprocessing versions of the algorithm are used. This decoding approach experiences similar complexity issues as the decoder presented in [49], especially with the larger order reprocessing versions of the decoder. The larger order reprocessing versions of the decoder have a higher complexity cost because more subsets are used to construct more candidate codewords from the different information sets.

A different approach to decoding with information sets using soft information is presented in [50]. For this implementation, the $(n-k)$ set of most plausible error patterns are represented in the form of ellipsoids in Hamming spaces. Once the erroneous positions are identified, the information set of k bits are then used to re-encode a candidate codeword in each trial of the decoding process . Finally, the candidate codeword that is closest to the received vector is selected as the decoded codeword. For the set of reliabilities $p = [p_1, ..., p_n]$ obtained from the received vector, the ellipsoid $\Theta(p, \xi)$ of radius ξ is defined as

$$\Theta(p, \xi) = \{a' \in \Theta_2^n | \sum_j p_j a_j'^2 \leq \xi\}. \tag{3.1}$$

In [50] a minimal ellipsoidal covering $T(n, n-k, \Theta(p, \xi))$ that covers the ellipsoidal $\Theta(p, \xi)$ of exponential size 2^{n-k} using the least possible number of $(n-k)$ subsets is designed, where $T(n, n-k, \Theta(p, \xi))$ represents a covering set of vectors with a minimum weight of $(n-k)$. This decoding approach has an exponential complexity

$2^{n\partial}$. The exponent ∂ is defined as

$$\partial = (1 - \mathcal{R})(1 - \mathbb{H}(\frac{\gamma}{1 - \mathcal{R}})), \tag{3.2}$$

where $\mathcal{R}$ is the code rate, $\mathbb{H}$ is the entropy and γ is the relative Gilbert-Varshamov distance [16, 51, 52]. The exponential growth in complexity leads to a much slower decoding time especially for codes with higher values of k [32].

A sub-optimum soft-decoding for linear block codes that is based on information set decoding is proposed in [32]. This decoding approach is based on the information set decoding technique presented in [53]. The decoder works by re-encoding a list of candidate codewords using matrix G^+. Matrix G^+ is obtained by adapting the G matrix based on bit reliabilities in the information set. In the list decoding stage of this algorithm, a candidate codeword is assumed to be correct if it is determined to be in the Voronoi region. The Voronoi region is defined as the set of vectors that are closest to the transmitted codeword. The candidate codewords are determined to be in the Voronoi region based on the stopping condition computed in (3.3)

$$\sum_{j' \in \mathbb{Q}} \Upsilon'_j \leq 0, \tag{3.3}$$

where j' represents the indexes of the vectors (Υ) in the Voronoi region and $\mathbb{Q}$ represents the vectors, Υ, of d_{min} indexed by j'. The list of codewords are generated by adding erasures to the information set. The erasures are introduced by flipping each of the bits in the information set once . After every bit flip, the altered information set used is to re-encode a new candidate codeword. This decoding approach is shown to yield a performance that matches the Viterbi algorithm [17] in terms of bit error rate (BER) when applied to Golay codes. The error correction performance tests for the decoder proposed in [32] are only consider for coderates not higher than $\frac{1}{2}$. This is because for larger coderates the Voronoi region is larger. This results in in an increase in the combination computations in the information sets required during decoding. The complexity cost of the proposed method in [32] can be represented as a linear function of k [32]. Reduced complexity variants are also proposed in [32], but this operate at a reduced performance in terms of

error-correction when compared to the original implementation.

In [30] a FPGA implementation for information set decoding is presented when applied to RS codes. Besides the FPGA implimnetation, part of the novelty of the approach proposed in [30] is that information set decoding is applied to a class of nobinary codes. Most of the implementations presented in literature, to that point, had applied the information set decoding techniques to binary codes. However, it is important to note the decoding method presented in [30] is not applied at the symbol-level but instead implemented at the bit-level. The decoder is able to work at the bit-level by performing a binary image expansion [21] on the H matrix and converting the RS code into its bit form. The proposed decoding method in [30] is based on a modified OSD implementation of the information set decoding technique used in [54]. The list of candidate codewords from the OSD are obtained using order-1 reprocessing [49]. This involves flipping every bit in the information set once to obtain the different subvectors, of length $k \times m$, used in the re-encoding of the candidate codewords. The main difference with the OSD approach presented in [30] when compared to the approach used in [49], is that the re-encoding of the candidate codewords is carried out based on H instead of G. The proposed OSD algorithm is further modified to enable an efficient hardware implementation and to reduce the complexity cost of the decoder. The modified algorithm determines if the next bit is to be flipped based on the syndrome weight. A threshold is set on the syndrome weight in the initialisation of the OSD algorithm to assist in the decoding process. Instead of computing the hamming distance, the decoded codeword is determined by selecting the candidate codeword that yields a syndrome weight below the set threshold value. The modified OSD algorithm is shown in [30] is able to yield a performance gain in terms of frame error rate (FER) when compared to the hard-decison decoder based on HDD IP core for RS codes [55].

Information set decoders when applied to linear block codes, as presented in the literature, utilise list decoding techniques to obtain the decoded vector. Therefore, in spite of information set decoders being generally less complex than other decoding techniques [10] [56], the candidate codewords used in the list decoding are

obtained by performing combinatorial computations which add to the complexity of the decoding approach. This is especially true for larger values of k [28, 32, 49]. Also the use of list decoding ensures that information set decoding algorithms that utilise soft information are soft-input hard-output algorithms. Work in this research proposes the use of iterative decoding to obtaining the decoded vector using the information set, based on a soft-input soft-output approach. The soft information is corrected based on a message passing algorithm that assists in identifying the incorrect symbols in the information set. This approach reduces the complexity cost of the decoder because no combinatorial computations are require to create a large list of candidate codewords. Instead, the symbols in error are corrected directly based on the extrinsic information.

The proposed information set decoding approach presented in section 2.8 is able to take advantage of systematic structure presented by the H matrix of the code at the symbol-level. However, the complete dense structure of parity submatrix of H ensures all the symbols of the received vector that form the information set participate in every syndrome check. This adds to the difficulty in detecting and correcting of errors in the information set of symbols used in the decoding process. Due to the challenges presented by the dense structure of the H matrix at the symbol-level, the proposed decoding approach will be first implemented at the bit-level so as to take advantage of the sparse structure of the binary image of the H matrix.

Research on information set decoding techniques that iteratively utilise soft information input into the decoder with the ability to output the corrected soft information has not been explored in literature. With regards to this, the implementation of a soft-input soft-output iterative decoder for RS codes, based on information set decoding is presented in this research. This decoding approach is proposed with the aim of taking of advantage of the reduced complexity that comes with an information set decoding implementation [10] [56]. The proposed approach takes advantage of the systematic structure created from the H matrix to iteratively correct the soft information during the decoding of the received vector with the aid of information set decoding techniques.

3.3 Reed-Solomon Decoders

Reed-Solomon (RS) codes are a class of non-binary Bose-Chaudhari-Hocquenghem (BCH) which were developed in 1960 by Irving S. Reed and Gustave Solomon [2]. RS codes have played an important role in the telecommunications revolution. Reed-Solomon codes posses favourable algebraic properties which has allowed for numerous applications in telecommunications and storage devices. Development of numerous decoding schemes to help improve the error correction performance of this class of code for its various applications has been of much interest.

Decoding schemes for RS codes fall into two categories. They can either be hard decision decoders or soft decision decoders. Hard decision decoders, based on the minimum distance principle, have proved to work efficiently with this class of code due to the fact that RS codes meet the Singleton bound [17]. Some of the standard and widely used hard-decision decoders for RS codes include the Berlekamp-Massey (BM) algorithm [4][45][33], the Euclidean algorithm [5] and the Berlekamp-Welch algorithm [34].

The BM algorithm is a popular hard-decision decoder for RS codes with numerous research works on the decoding of RS code being dedicated to improving the performance of this algorithm [57–63]. Some of the improvements into the BM decoder are implemented in [62] with the aim of reducing the complexity and the latency of the decoder. The proposed modifications were applied to full RS codes and and also adopted to shortened RS codes. The reverse BM decoding algorithm is presented in [62] with the main of aim solving the partial-inverse problem. The partial-inverse problem occurs as a result of the alternative key equation which generalise naturally to polynomial remainder codes [57], [60]. The solution to the partial-inverse problem presented in [57] is an algorithm that coincides with the BM algorithm except that it processes the syndrome in an inverse order, hence the name reverse BM decoder. Work carried out in [63] gives a more detailed derivation of the partial-inverse problem and the reverse BM decoding algorithm. More recent work carried out in [64] presents a different viewpoint of the BM

algorithm which is more similar to the Euclidean algorithm [5]. The proposed approach is simpler than the Massey formulation and can be used to better explain the BM algorithm in course presentations. In spite of the efficient implementation of the hard-decision decoders for RS codes like the BM algorithm, whenever soft information from the channel is used in decoding, a significant performance gain is exhibited when compared to hard-decision decoding techniques [1, 6, 8]. With this regard, work carried out in this research focuses on soft-decision decoding techniques.

Algebraic soft-decision decoder like the Guruswami and Sudan (GS) [65], and its modification proposed by Koetter and Vardy (KV) [7], are able to correct errors beyond the Singleton bound [18]. However this performance comes at the cost of an increased computational complexity [41]. There are several techniques that have been proposed in literature with the aim of reducing the computational complexity cost of algebraic soft-decision decoders [44, 66–68]. This complexity can become prohibitively large when greater coding gains are required [6].

The Chase decoder [69] has also been extensively modified in literature [67, 70–74] with aim of improving the performance of the decoder. The most recent work on the Chase decoder is presented in [43] and is based of the work carried out in [72]. This modification of the Chase decoder is a soft-input soft-output and it involves implementation of the test vector generation using multivariate Bernoulli (Multinoulli) distribution [75]. This version of the Chase decoder obtains the required parameters by exploiting the void space between constellation points and the q-ary modulation. The decoder implements soft-decision techniques but also employs the use the hard decision BM decoder in the algorithm. This makes the algorithm a double decoder which adds to the overall complexity. The modified Chase decoding algorithm in [43] is shown to yield a good error correction performance, however this is at the expense of complexity. The decoder has an exponential complexity which can be prohibitively large when applying it to large codes. This is in part to the algorithm employing list decoding techniques and search algorithms to obtain the decoded vector. The complexity for the symbol-level version of the algorithm can be reduced to a polynomial complexity. However this is only under

the condition when the K-D tree algorithm [76] is adopted to generate unique test-vectors.

Iterative soft-decision decoders are able to fully utilise the soft information by continuously applying corrections to the received vector outputted from the channel [16]. This enables the iterative soft-decision decoders to converge to the most likely codeword. Due to iterative decoders being able to converge to a codeword with a good error correction performance and a reasonable complexity, when compared Algebraic and list decoding techniques while handling soft-information, work presented in this research will focus on iterative decoding techniques. Iterative soft-decision decoders for RS codes include the Adaptive Belief Propagation algorithm (plus its various modifications) and the Parity check Transformation Algorithm. Iterative decoders for RS codes are covered in Section 3.4 and Section 3.5.

3.4 Bit-level Iterative Soft-decision Decoding of Reed Solomon Codes

Implementation of soft-decision decoding techniques for RS codes has been an area of active research for a long time [6–8, 12, 25, 26, 39, 77–81]. Soft-decision decoders for RS codes working in the field GF(2^m), where m is any positive integer, can be divided into two categories. They can either be symbol-level decoders or bit-level decoders. Most soft-decision decoders for RS codes work on a symbol-level. Some of the high performing symbol-level decoders include the widely used Koetter and Vardy (KV) algorithm [7] and the Parity check Transformation algorithm (PTA) [41] [8]. Previous research works on soft-decision decoders have shown gains, in terms of error correcting performance, attained when working on a bit-level compared to symbol-level [6, 12, 24, 82], however this is often at the cost of a higher computational complexity. In addition to the good error correction performance, the bit-level implementation of RS codes allows for the code to be used in situations where decoders are required to make use of bit based soft information [22]. Soft-input soft-output bit-level decoding of RS codes is first proposed in [22]. The

research work in [22] was motivated by the gains achieved in terms of error correction performance attained by using soft-decision decoding over hard-decision decoding [6, 45, 83, 84]. In spite of the gains in error correction, the complexity cost of soft-decision decoding at that time made utilisation of soft information during the decoding process prohibitive. The aim of the bit-level implementation was to develop an efficient decoder for RS codes that could make full use of bit-based soft-information, which was the main drawback of RS decoding algorithms for their applications in satellite telecommunications at the time [85]. The bit-level implementation in [22] is made possible through the decomposition of RS codes into BCH sub-field subcodes which are glued together using glue vectors. In spite of the decomposition of the RS codes significantly reducing the trellis complexity of maximum likelihood decoding, the complexity still increases exponentially with code length and d_{min} [6]. This makes the decoding approach presented in [22] infeasible for practicable long codes.

In [6], a highly successful implementation of an iterative soft-input soft-output bit-level decoding algorithm is presented and referred to as the Adaptive Belief Propagation (ABP) algorithm. The ABP algorithm is shown in [6] to yield significant gains in terms of error correction when compared to other widely used RS decoders. Based on this, all implementations of iterative bit-level soft-decision decoding algorithms for RS codes are in some way a variation of the ABP algorithm. This section, therefore, gives a review in to the literature on the different bit-level implementations of the widely used ABP decoding algorithm.

The ABP algorithm is first presented in [6] by Jing Jiang and Krishna R. Narayanan with the aim of effectively applying soft-decision iterative decoding techniques to linear block codes defined by a H matrix with a dense structure. The iterative decoder selected for this implementation was the belief propagation algorithm due to its near shannon limit performance when applied to LDPC codes [3] [86]. The dense structure of the RS code H matrix presented implementation challenges for the decoder. This is because the belief propagation algorithm is designed to take advantage of the sparse structure presented by the H matrix of the LDPC code so as to decode the received vector efficiently [87]. Before the belief propagation

algorithm is used in the decoding of RS codes, a binary image expansion [21] is performed on the H matrix so as to make it sparse. In addition to working at a bit level, the ABP decoder also iteratively adapts the binary image of the H matrix based on the bit reliability information to improve the error correcting performance of the algorithm. The adaptation is done in such a way that the bits considered to be unreliable match the sparse identity submatrix of the adapted binary image of the H matrix. Adapting the matrix in this way prevents the decoder from getting stuck at psuedo-equilibrium points [6] due to the reduced use of unreliable information in the message passing stage of the decoder. This results in a significant improvement in the convergence behaviour of the bit-level iterative decoder.

The generic form of the ABP decoder presented in [6] works by first converting the bit reliabilities of the codeword into their corresponding Log Likelihood Ratios (LLR). The H matrix in its binary image form is then adapted based on these LLR values using Gaussian row reduction techniques. Belief propagation is then applied to the soft information based on the adapted H matrix, as presented in (2.31),(2.32) and (2.33), to decode the received vector. This ABP implementation proposed in [6] has been shown to yield a significant gain when compared to widely used RS decoders including the KV algorithm, BM algorithm and the Algebraic hard-decision decoder. However, the gains achieved by the ABP algorithm come at the cost of a high computational complexity [6, 88]. The complete process of decoding the received vector using the generic form of the ABP algorithm is explained in Section 2.6.1.2.

Further modifications are made to the ABP decoding algorithm in [6] with the aim of either improving the error correcting capability or to reduce the computational complexity cost of the decoder. A variation to improve the error correcting performance of the ABP decoder based on preprocessing [89] [90] is one the modifications proposed in [6]. This technique is similar to the reprocessing carried out in [28] for the OSD algorithm in that it utilises different groupings of the unreliable bits to assist in the decoding of the received vector. That is, the algorithm switches some bits that are considered reliable with the unreliable bits near the

boundary of the two sets of bits during the decoding process. The preprocessing approach used in obtaining the different groupings of the reliable bits for the ABP decoder is presented in [89]. The matrix adaptation in then carried out on the binary image of H based on the new groupings of the bit reliabilities. For each iteration the decoder is run, a new estimated codeword may be obtained due to the difference in the adapted binary H matrix used. Similar to the OSD decoder presented in [28, 30, 49, 54], all the estimate codewords are kept in a list and finally the most likelihood codeword is selected as the decode codeword [6]. In addition, the algorithm also works alongside a hard-decision decoder. The hard-decision decoder corrects the vectors outputted from the ABP decoder for every iteration. The estimated codewords outputted from the hard-decision decoder are also added to the list of possible decoded codewords. This version of the algorithm is represented as ABP-HDD(f_1, f_2). The notation f_1 refers to the maximum number of iterations required to decode by a single grouping of the bit reliabilities, while the notation f_2 refers to the number of decoding rounds based on the different groupings. This implementation of the ABP results in the decoder running for a maximum of $(f_1 \times f_2)$ iterations. This implementation of the algorithm is shown in [6] to yield the best error correction performance of all the versions of the ABP decoder presented in the paper. However, this version is significantly more complex due to the numerous iterations required during the decoding of the received vector and the use of double decoding techniques through the implementation of a hard-decision decoder. The numerous iterations increases the complexity of the ABP decoder through the additional computations carried out, in each iteration, for the Gaussian elimination process and belief propagation operations. The extra computations required in the hard-decision decoding also contributes to the increased computational complexity cost.

Another modification presented in [6] is the partial reliable bit updating version of the ABP algorithm. This version of the algorithm is presented with the aim of reducing the computational complexity cost of the ABP decoder. The main difference between this version of the ABP and the generic implementation of the decoder is seen in the bit updating stage. This version of the ABP updates the

all the unreliable bits and a subset of the reliable bits close to boundary with the unreliable bits. That is, a specified number of bits that are considered to be the most reliable, and further away from the away from the boundary with the unreliable bits, are not updated during the decoding process. This is carried out because the main floating point operating complexity comes from the computation (2.31), which calculates the extrinsic information in the reliable part based on the more dense parity submatrix in the binary image of H [6]. Therefore, the extrinsic information of the set of bits further away from the boundary with unreliable bits is not computed. This results in a reduced cost in computational complexity of the ABP. However, it is important to note the reduced computational complexity cost achieved by this implementation results in a performance loss in terms of error correction when compared to the generic version of the ABP as shown in [6].

The use of the ABP algorithm is also incorporated with a hard-decision decoder in [6] and [81] so as to improve the error correction performance of the decoder. The version of the ABP in [6] considers an implementation where the ABP runs until the maximum number of iterations, f_{max}, is reached in order to obtain a list of candidate codewords outputted from the hard-decision decoder in each iteration. This is carried out because the codeword returned from hard-decision decoding may differ from the codeword obtained from the maximum likelihood decoding approach utilised by the ABP algorithm. After all the iterations are run, the ABP decoder selects the most likely codeword from the full list of candidate codewords. This decoding approach has the advantage of being unable to perform worse than either the ABP or the selected hard-decision decoder. This decoding approach is implemented in [91] using the ABP and the Berlekamp-Massey (BM) algorithm [4] and denoted as the ABP-BM decoder.

In [81] a simplified Chase type decoder [69] is cascaded to the ABP algorithm in each iteration of the decoding process. One Chase type decoder is run before the ABP such that if it succeeds to find a valid codeword, the iterative process of ABP algorithm can be avoided. The second Chase type decoder is run after each ABP update and acts as a stopping condition for the iterative decoding process when a valid codeword is obtained. The additional stopping criteria defined by the

Chase type decoder improves the convergence behaviour of the ABP algorithm. The ABP-Chase decoder is shown in [81] to outperform the ABP-BM decoder in terms of error correction capability. The decoding approach presented by the ABP-Chase decoder is still more complex than the generic ABP implementation. This is because it is essentially a double decoding scheme. Therefore, the computations carried out by the Chase decoder add to the total complexity cost incurred by the ABP algorithm.

A hybrid implementation of the ABP decoder with Algebraic Soft Decoding (ASD) is presented in [91]. The algorithm utilises a list decoding approach and is denoted as ABP-ASD. The ABP-ASD decoder works by generating a list of candidate codewords at the end of each iteration. If a decoding success is achieved by the ABP-ASD at the end of an iteration, the candidate codeword is added to a global list of codewords. The decoder stops when the number of iterations required to decode the received vector $f_1 = f_{max}$. If there are multiple codewords in the global list, then the list decoder outputs the codeword with the shortest Euclidean distance from the received vector. The preprocessing approach presented in [6],[89] can also be used to improve the performance of the decoder. The preprocessing approach allows for f_2 parallel ABP-ASD decoders, each performing list decoding, to form at most $f_1 \times f_2$ codewords. In [91], the algebraic soft-decision decoder used alongside the ABP is the KV algorithm. The KV algorithm acts as the multiplicity assignment scheme for the ABP-ASD decoder. That is, the soft information after the ABP update in each iteration is used by the KV algorithm to make the multiplicity matrix. The multiplicity matrix is then fed in to the modified Guruswami Sudan (GS) algorithm [65] [92] to obtain the candidate codewords used in the list decoding stage of the ABP-ASD decoder.The ABP-ASD decoder employs a similar double decoding approach to that presented in [81]. This increases the computational complexity of the decoder as the computations of both the ABP and the KV decoder are considered in the total computational complexity cost.

A reduced complexity version of the ABP-ASD is also proposed in [91]. This version works in such a way that the complexity cost of the Gaussian elimination

process is only significantly experienced in the first iteration. However, the complexity incurred by utilizing KV decoding after each ABP iteration is extremely high [88] [93]. The ABP-ASD is shown in [91] to yield an error correction performance gain when compared to the ABP-BM decoder.

Bit-level adoptive belief propagation techniques are explored for double and triple parity check RS codes in [94]. The work derives parity-check equations that posses a specific GF(2) structure. This inherent GF(2) structure, obtained from the derivations, enables representation of a compound binary Hamming code from the non-binary parity-check equations. This approach enables Belief-Propagation decoding of $(n, n-2, 3)$ and $(n, n-3, 4)$ RS codes. Results run in [94] show a favourable performance of the decoder when compared to the BM and the KV decoder. However, this approach is only effective for the derived expressions for parity check equations with the roots $\alpha^{z^*}(0 < z^* < 3)$ over GF(2^m). This is because these parity check equations are viewed as compound Hamming codes over GF(2). When applied to non-binary cyclic codes with roots more than 3, the decoding performance is not as good as expected [94].

Research carried out in [25] proposes an ABP implementation with Informed Dynamic Scheduling (ISD) [95–97]. This version of the algorithm is based on the Residual Belief propagation (RBP) [95] algorithm. The dynamic scheduling is done with the aim of ensuring the least reliable bits receive more updates while simultaneously limiting their unreliable influence. This is achieved by updating to the least reliable bits using the important messages. The importance of the message is measured according to a metric referred to as the residual [25] [95]. This decoding technique is shown to yield a good error correcting performance in [25]. However, the algorithm requires numerous iterations and utilises double decoding techniques through the implementation of a non-binary algebraic hard-decision decoder in the iterative decoding process. This approach, therefore, adds to the complexity cost of the modified ABP decoder.

Iterative soft-input soft-output decoders for RS codes working at a bit-level that are not based on belief propagation have not been investigated in literature. One of the contributions presented in this research is the implementation of a bit-level

iterative decoder, for RS codes, that corrects the soft information outputted form the noisy channel without the use of the belief propagation algorithm. Implementation of belief propagation is avoided in the proposed decoding technique due to the high computational complexity cost presented when using the ABP algorithm [6, 88, 98, 99]. The decoder proposed in this research is a message passing algorithm, that takes advantage of the sparse structure presented by the binary image of the H matrix to decode the received vector. Information set decoding is implemented in the proposed bit-level decoding approach with the aim of improving the iterative convergence rate during the decoding process. An improved iterative convergence rate contributes to a reduction in the computational complexity cost as it reduces the total number of operations carried out during the entire decoding process of the received vector.

3.5 Symbol-level Iterative Soft-decision Decoding on a Dense H matrix of Reed Solomon Codes Based on Information Set Decoding Techniques

Work in this research focuses on the development of a decoding schemes that utilise information set decoding techniques to increase the efficiency of the decoder. This is mainly achieved through an improved iterative convergence rate while detecting and correcting errors in the received vector. Symbol-wise iterative decoding for linear block codes is based on the use of reliability information at the input (soft-input) to produce soft-information at the output (soft-output) [16]. For RS code there exist soft-input hard-output iterative decoders such as the ones presented in [77, 100]. However, its often desirable to have the soft information about the reliability of the decisions [16]. This is especially true for iterative decoders because for each iteration, after the first iterations, new reliability information based on the extrinsic information is incorporated during the decoding process [16].

A soft-input soft-output iterative decoder is presented in [101] for a concatenated scheme [102], with a Convolution inner code and a RS outer code. The iterative decoder combines the iterative decoding scheme for block Turbo codes [103] and

the iterative decoding scheme for a Convolution Turbo code [104]. This decoding approach is able yield a coding gain of 6.6dB for a BER of 10^{-5} while transmitting through an AWGN channel using a binary modulation scheme. It is important to note the complexity of this decoding approach increases with $d_{(min)}$. This increases the cost in computational complexity when decoding codes of a longer length. Also, it is worth noting this decoder implementation has only been applied to concatenated RS codes and not general RS codes.

Iterative soft-input soft-output decoding techniques for RS codes is difficult to implement [26] due to the dense structure of the symbol-level H matrix [6]. The first successfully iterative soft-input soft-output decoding technique for RS codes was the ABP decoder [81]. This was mainly due to the algorithm being implemented on a bit-level, which takes advantage of the sparse structure presented by the binary image of the H matrix. Based on the successful implementation, most iterative soft-decision decoders for RS codes are in some way a variation of the ABP algorithm and work on a bit-level [6, 12, 25, 26, 81]. These variants have been able to yield a favourable error correcting performance when compared to the ABP algorithm, however this is at the cost of a high computational complexity [6, 81, 88]. Working on a bit-level, with a larger H matrix, and applying Gaussian row reduction operations in each iteration contribute to the computational complexity cost of the algorithm [81].

In an attempt to avoid the computational complexity cost of iterative decoding attained from working at a bit-level with soft information, research into the implementation of iterative symbol-level decoders lead to the development of the PTA algorithm [8]. The decoder works by transforming the H matrix of the RS code based on the soft symbol reliability information. The reliability information is then corrected based on the syndrome check equations so as increase the likelihood of the correct symbol being selected. This process is repeated iteratively until the most likely decoded vector is obtained. The main novelty of the algorithm is that it is able to correct the soft information outputted from the channel based on the dense symbol-level H matrix. The PTA algorithm has been shown to outperform the KV decoder and the BM algorithm [8] [41]. However, due to the same dense

structure of the H matrix, the algorithm requires numerous iterations during the decoding process to achieve these gains in the error correction performance [13]. The numerous iterations required during the decoding process add to the overall computational complexity cost of the PTA decoder.

The dense structure of the H matrix has also presented a challenge in the symbol-level implementation of the information set decoding technique proposed in this research. The complete dense structure of the parity submatrix in the H matrix makes it difficult to detect and correct any error that may occur in the information set of symbols. Obtaining the correct information set of symbol is very important because even one wrong symbol in the information set will lead to incorrect decoding of the entire group of symbols outside the information set. A symbol-level implementation of the proposed decoding approach is crucial to this research due the reduced computational complexity cost attained from working at a symbol-level when compared to working at a bit-level.

Due to the challenges presented from having a dense structure in the G and H matrices of nonbinary codes, like in the case of RS codes, investigation into the implementation of information set decoding using iterative soft-decision techniques at the symbol-level (nonbinary) has not been fully explored in past literature. Work done in this research presents a decoding approach that allows for iterative soft-input soft-output decoding of the information set on a dense symbol-level H matrix without the use of the binary image of the code. The proposed decoding technique extends the parity check matrix with aim of adding sparsity to the dense H matrix at the symbol-level. The extensions to the parity check matrix are made up of low weight parity check equations that are added to the original symbol-level H matrix. The weight of all the parity check equations, added to the H matrix, matches the minimum distance of the code. Each extension to the parity check matrix has the same dimensions as the original H matrix. This enables the columns of the parity check extensions to be arranged in such a way that each extension has a different information set. The use of different information sets in each extension, coupled with the added sparsity of the extended H matrix, allows for efficient exchange of the soft information during the decoding of the

received vector. This results in the reduction of the number of iterations required in the decoding process, which in turn yields a lower computational complexity cost for the proposed decoder. Therefore the use of the extended matrix allows for a suitable implementation of the proposed information set decoding technique, which results in the realisation of a symbol-level soft-input soft-output iterative decoder for RS codes.

3.6 Information Set Decoding on a systematic Binary QC-LDPC code

Low density parity check (LDPC) codes are a class of linear block codes with implementable decoders that provide a near Shannon limit performance, on a large set of data transmission and data-storage channels. After the discovery of these codes by Gallager in 1962 [14], these codes were largely ignored for many years due to their computational and implementation complexity. In 1981, work done by Tanner on generalized LDPC codes developed a graphical representation of the codes [105]. This study was important as it has greatly helped in the understanding of decoding LPDC codes. Research on LDPC codes was resurrected in the mid 1990's when they were popularized by Davey and MacKay who proved these codes had a near Shannon limit performance and other advantages of block codes with low density (sparse) parity check matrices[19]. Since then there has been a lot of research on the performance analysis based on construction, efficient encoding, decoding and overall design of these codes for various applications in digital communication and storage devices.

LDPC codes can be constructed in either one of 2 ways. The LDPC constructions are based of random constructions or based on structured algebraic construction [106]. LDPC codes based on random construction techniques [3, 107–110] have been shown to yield a good error correction performance in literature. However, they are not easy to implement in hardware due to considerable storage space needed for irregular connections between the variable and the check nodes in the Tanner graph [106]. As a result structured constructions are favoured for practical

applications.

One of the frequently used subclasses of non-random LDPC codes are the quasi cyclic (QC) LDPC codes. This is largely due to their simple implementation by use of shift registers and their good bit error rate (BER) performance when transmitted through a noisy channel [15]. QC-LDPC codes have the advantage of yielding a performance comparable to randomly constructed LDPC codes while having a structure that easily allows for hardware implementation [111]. The quasi-cyclic structure possessed by this class of LDPC codes also enables parallel decoding which allows for the tradeoff between the decoding throughput and implementation complexity [111]. As a result, much of the research on QC-LDPC codes has focused on the different constructions of parity check matrices and more efficient implementations for different applications [11, 15, 112–115].

Some of the commonly used QC-LDPC codes in literature are the codes defined by the construction based on field partitions [116]. This construction works by partitioning the elements of the field GF(2^m) into two disjoint subsets. The elements of the base matrix is the calculated by getting the sum of the two subsets as shown in [20]. The base matrix is then expanded to form the H matrix by performing a q-ary (where $q = 2^m$) matrix expansion to create circulant matrices from the each element of the base matrix [11] [15]. QC-LDPC codes have also been constructed based on the minimum-weight codewords of an RS Code with two information symbols [117]. As the name suggests, this construction techniques uses algebraic methods to construct the base matrix based on the minimum-weight of two Reed-Solomon codewords over GF(2^m) with two information symbols. The base matrix is formed in a $(2^m - 1) \times (2^m - 1)$ square matrix with zeros along the diagonal. The rate and the length of the QC-LDPC code can then be adjusted by an appropriate selection of a submatrix, starting from the first row and column, to perform the q-ary matrix expansion. Progressive Edge Growth (PEG) techniques have also been used in the construction of QC-LDPC codes [118]. The PEG algorithm is a technique of LDPC code construction which looks to optimize the girth during the edge growth [119] [120]. The PEG algorithm is able to generate large cycles each time a new edge is place on to the Tanner graph, however it cant guarantee

if this is the best possible tanner graph [118]. Improvements are made to the PEG algorithm for implementation with QC-LDPC codes in [118]. These modifications reduce the presence of error floors in the performance of the QC-LDPC by efficiently eliminating short cycles in the Tanner graph. An algebraic approach based on the conjugates of primitive elements over finite fields is proposed in [115] to construct QC-LDPC codes. This construction works by creating the base matrix finding the conjugates of the root of the primitive polynomial of degree m in the filed GF(2^m). Research in [115], proves that the conjugates root of the polynomial of degree m are also root of the same polynomial and are all elements in the field. The base matrix is a square matrix, of dimension $e^* \times e^*$ (where $e^* \leq m$) filled with the conjugates of the root for the primitive polynomial.

These QC-LDPC constructions, like most QC-LDPC codes, are defined by parity check matrices that have a non-systematic form therefore losing some of the advantages that come with this structure. This is due to the fact that the H matrix does not posses a full rank. The advantages of having a systematic structure is that the decoded information does not need to be converted into user data at the receiver. This ensures less encoding complexity as it only involves the adding of a matching identity submatrix to the transposed parity submatrix.

Research carried out on the construction of LDPC codes defined by a H matrix with a systematic structure is presented in, but not limited to, [121–123]. However these constructions do not have the quasi-cyclic structure, and therefore do not possess the advantages that come with this structure.

Examples of QC-LDPC codes with a full rank include the codes that form the basis of communication system standards like the IEEE802.11-2012, IEEE802.16-2009 and ITU-T G.9960 [124]. This codes however do not posses a systematic structure. This means Gaussian elimination can be used in the encoding process but it tends to give a more dense generator matrix that is not in circulant form. This presents implementation challenges for codes of a longer length [125]. In [15], Gaussian row reduction is used to encode and also maintain the circulant and sparse structure of the code. However, with this technique, a systematic encoder is only applicable for cases of a LDPC code defined by a parity check matrix with a full row rank.

Work carried in [125] presents an efficient practical implementation of a QC-LDPC code constructed using the approach presented in [15].

Work in [126] presents the construction of QC-LDPC code with a H matrix that can be used to generate a systematic codeword. However, the H matrix for the QC-LDPC code presented in this research is a regular LDPC code with a node degree 2 on the Tanner graph. This type of construction is not suitable for the information set decoding approach proposed in this research. This is because the information set decoding approach requires a node of degree 1 on the Tanner graph for the identity submatrix. A node degree of 1 ensures that the bits matching the identity submatrix do not actively participate in the extrinsic information calculations. This justifies the erasing of these bits when using the information set to re-encode the codeword. Also, the construction presented in [126] is limited to a degree 2 at each node. This presents a problem when more connections are required at each node of the Tanner graph.

Work carried out in this research presents the construction of a QC-LDPC code defined by a parity check matrix with a systematic-circulant (SC) form. This is achieved by using the underlying idea presented in [8] [127], to obtain the generalized form of the reduced complexity row reduction technique. This construction gives all the advantages of having a systematic structure while maintaining the sparseness and quasi cyclic properties of the LDPC code. Information set decoding techniques are applied to the proposed systematic construction of the QC-LDPC code through the aid of an additional stopping condition during the iterative decoding process. This is done to reduce the number of iterations required during decoding of the received vector.

CHAPTER 4

Bit-level Implementation of the Proposed Soft-Decision Decoder for Reed-Solomon Codes

4.1 Introduction

This chapter provides the techniques employed in the development of the proposed bit-level implementation for the iterative soft-input soft-output information set decoding algorithm. From Subsection 2.8, it can be seen that an information set made up of any k symbols from the received vector can be used to successfully re-encode the decoded codeword by taking advantage of the systematic structure of the H matrix. This decoding approach especially lends itself to RS codes. This is because a rearranged systematic structure can be obtained using any set of columns of the H matrix due to their full row rank inverse properties [127]. However, this technique can only successfully decode the entire received vector if and only if all the symbols in the information set are correct. If even one of the symbols in the information set is wrong, the decoder will fail to correctly decode the received vector. The challenge in correcting the information set symbols, for RS codes, is largely a results of the complete dense structure of the parity submatrix of H. Therefore, the proposed decoding algorithm is implemented at a bit-level so to take advantage of the sparse structure attained from the binary image of the H matrix. Preliminary tests to assist in design and implementation of the proposed decoder are run on the PTA algorithm at a bit-level.

Decoding the received vector using the information set decoding approach enables the proposed algorithm to run less iterations. This is because the proposed algorithm does not have to ensure all the bits are correct before it can break the

iterative decoding process. As a result, the time required to decode is reduced which in turn helps improve the efficiency of proposed algorithm during the decoding process. The proposed decoding algorithm is able to effectively implement the information set decoding technique with the aid of an additional stopping criteria that is referred to as the decoding condition which is defined in this chapter. This chapter also presents the implementation of an additional stopping criteria based on information set decoding techniques to the ABP and the bit level version of the PTA decoder. The additional stopping condition assists in reducing the iterations required to decode the received vector. This results in a reduced complexity version of the soft-decision bit-level decoding algorithms. The reduced complexity is achieved while maintaining the error correction performance of the decoders.

The proposed iterative decoder is benchmarked against the generic version of the ABP decoder as opposed to the ABP-HDD(f_1, f_2) modification [6]. This is due to the high complexity cost of the ABP-HDD(f_1, f_2). The ABP-HDD(f_1, f_2) always runs for a total of ($f_1 \times f_2$) iterations even when under higher SNR conditions. The algorithm works more like a list decoder instead of an iterative decoder while using double decoding techniques through the addition of a hard-decision decoder. This means the ABP-HDD($f1, f2$) does not necessarily converge to a codeword in the same way the generic version of the ABP does. As a result benchmarking with the ABP-HDD($f1, f2$) will always give a better iterative performance, for the proposed iterative decoder, resulting in a less computational complexity cost especially for the higher SNR values. The generic version of the ABP works as an iterative decoder and is believed to give a fair comparison in terms of the complexity analysis and error correction performance.

The rest of the work carried out in this chapter is structured as follow: Section 4.2 presents the preliminary test run on a modified version of the PTA algorithm that works on a bit-level. The preliminary tests are used to aid in the implementation of the proposed bit-level decoder developed in this research. A detailed description on how the proposed bit-level decoder works is given in Section 4.3. An analysis

of the proposed bit-level algorithm and simulation results are presented in Section 4.4. A complexity analysis of the proposed bit-level decoder is investigated in Section 4.5. The proposed technique used in the implementation of the information set decoding approach is proposed in Section 4.6 and Section 4.7 gives the conclusion to the findings obtained in this chapter.

4.2 Preliminary Tests

The preliminary design and implementation of the proposed decoding algorithm is presented in this section. The preliminary design is aided through the modification of the PTA decoder to enable a bit-level adaptation of algorithm. Simulations are run to test for the performance difference, in terms of error correction, of the PTA algorithm when applied to the bit-level sparse H matrix compared to the dense H matrix at the symbol-level. Performance gains at the bit-level would provide an additional motivation for the bit-level implementation of the proposed decoder beyond just the current implementation limitations attained by working with the dense H matrix at the symbol-level.

The performance of the bit-level implementation for the PTA algorithm in this chapter is measured in terms of Symbol Error Rate (SER) for a range of different Signal to Noise Ratio (SNR) values. An analysis with the aim of determining the optimum performance conditions of the proposed bit-level implementation of the PTA algorithm is also carried out in this chapter.

4.2.1 Bit-level Implementation of the PTA Algorithm

Modifications applied to the PTA algorithm to enable the bit-level implementation are now discussed. For purposes of notation, the PTA algorithm implemented at the bit-level is denoted to as PTA_{bl}.

It is also important to note, for the field $\text{GF}(2^m)$, a symbol-level H matrix with dimensions $(n-k) \times n$ will have a binary image matrix $\mathcal{H}$ with the dimension of $M \times N$, where $M = (nm - km)$ and $N = nm$.

4.2.1.1 Modifications in the H transformation step

As seen in Section 2.6.2, the PTA decodes the received vector by performing corrections to the reliabilities in the matrix β. Therefore, the algorithm does not directly utilize the algebraic properties of the RS code. However, it does take advantage of the structural properties of the H matrix. This is because the H matrix of an (n, k) RS code has a set of $(n-k)$ independent columns regardless of the order of columns selected [127] [128]. This assists during the transformation step of the PTA decoding process. After a binary image expansion is performed on H, the new binary matrix $\mathcal{H}$ lacks this property. However, the new binary matrix $\mathcal{H}$ still possess a full row rank. This means that for matrix $\mathcal{H}$, with the dimensions $M \times N$, there exists a total number of M independent columns which can be obtained through row reduction. However, there is no guarantee that these M columns will run consecutively to form an exact identity matrix after the row reduction is carried out. With regard to this, the modification to the H transformation step can be summarized as follows:

- The received vector is first converted into its bit form based on a selected modulation scheme. The β matrix is now constructed with only 2 rows to represent either a "0" or a "1" for the hard-decision detection. For a 16-QAM modulation scheme with Gray mapping, this process of obtaining the β matrix is explained in Section 2.9.3. To ensure that each column in β adds up to one as in the case of [8], the value of the reliabilities are scaled to obtain the matrix $\beta^{(scaled)}$ as shown in (4.1)

$$\beta_{(g,j)}^{(\text{scaled})} = \frac{\beta_{(g,j)}}{\sum \beta_j}, \tag{4.1}$$

  where $1 \leq g \leq 2$, $1 \leq j \leq N$ and $\beta_{(g,j)}$ denotes the entry in g^{th} row and the j^{th} column of β. The vector A is obtained in the same way as (2.34) and shown in [8]. The vector A is then sorted in ascending order. The rearranged indexes are noted as well and stored in the vector Y as shown in (2.35).

- Matrix $\mathcal{H}$ is then transformed based on the rearranged indices in the vector Y. The independent columns in matrix $\mathcal{H}$ are represented by the columns that match the identity submatrix after the transformation. These columns are now used to represent the indexes $\mathcal{U}$ and the remaining indexes are used to represent the $\mathcal{K}$ indexes.

4.2.1.2 Modifications to the correction step

Working at a bit-level means working with a binary parity check matrix that is more sparse than a regular H matrix for an RS code. For the symbol-level H matrix of a RS code, the identity submatrix is largely sparse with nonzeros only along the diagonal. The parity submatrix is completely dense with no zero elements. As a result of the contrasting differences in sparsity of the submatrices in the H matrix, in its partial systematic transformed state, the symbol-level PTA uses different values of δ in the correction step for each submatrix as shown in (2.38). This enables larger corrections to be made to the indices of the identity submatrix because each index only participates in one syndrome check equation during the syndrome calculation. This is opposed to the indexes matching the parity submatrix which require smaller corrections due to these indexes participating in all the syndrome checks equations in the syndrome calculations.

When working at a bit-level, the matrix $\mathcal{H}$ is considered to be sparse in both the identity and parity submatrices in its transformed state. This ensures that not all the bits, in both the identity and parity submatrices, participate in every syndrome check calculation. Therefore, the need for small corrections in the parity submatrix is eliminated. Therefore, the PTA_{bl} is implemented using a single δ value to correct the indices of the reliabilities that match both the sparse identity and the relatively sparse parity submatrices during each iteration of the decoding process. The modified correction step is now summarised as

$$A_j = \begin{cases} A_{t_i} + \delta & \text{if } S_i = 0, \\ \\ A_{t_i} - \delta & \text{if } S_i \neq 0. \end{cases} \tag{4.2}$$

where t_i represents a subvector in Y of indexes matching bits in the i^{th} row of $\mathcal{H}^+$ that participate in the syndrome check equation. The bit indexes represented by t_i can be expressed as

$$t_i = \{N : \mathcal{H}^+_{(i,N)} = 1\}. \tag{4.3}$$

4.2.2 Results and Analysis

Simulation results for the PTA$_{bl}$ algorithm are presented in this section. These simulations are carried out to obtain optimum performance conditions of the algorithm. The performance of the PTA$_{bl}$ decoder is also compared against symbol-level implementation of the PTA .

4.2.2.1 Algorithm Analysis

The performance analysis of the PTA$_{bl}$ decoder is first investigated based on the new implementation using a single δ to correct the reliabilities. The performance of this implementation will be compared to the original case when different δ values are used to correct the reliabilities matching the participating $\mathcal{U}$ and $\mathcal{K}$ indices in each syndrome check at a bit-level. A $(15, 7)$ RS code is used in this simulation, with the symbols being transmitted through an AWGN channel using a 16-QAM modulation scheme with Gray mapping. A value of $\delta = 0.001$ is used with both PTA$_{bl}$ implementations. Results for this simulation are shown in Fig. 4.1.

From Fig. 4.1 it can be seen that the single δ implementation performs better than the original implementation that requires different δ values for correcting the $\mathcal{U}$ and $\mathcal{K}$ indices. The PTA$_{bl}$ implemented with a single δ yields a gain of about 0.5dB at an SER of 10^{-3} when compared to the PTA$_{bl}$ with original δ implementation. This result supports the hypothesis presented in section 4.2.1.2 that, to obtain an optimum performance, a single δ value is required to correct the reliabilities due to the sparse nature of both the identity and the parity submatrix of $\mathcal{H}$. For the rest of the simulations the PTA$_{bl}$ decoder is implemented with a single δ.

A further investigation on the performance of the PTA$_{bl}$ based on the value of δ is

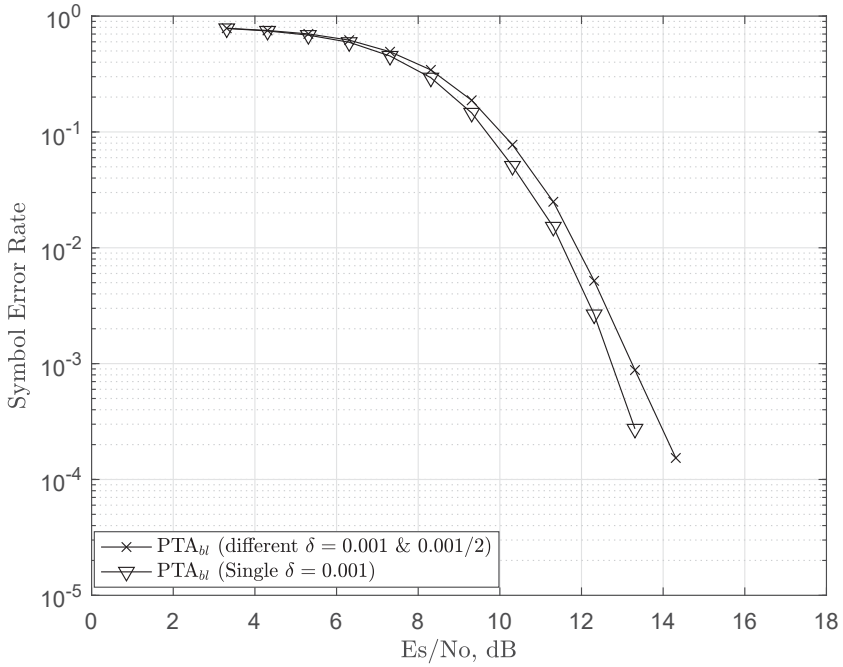


FIGURE 4.1: Performance comparison of the PTA$_{bl}$ based on the different δ implementations.

conducted. A similar test was performed in [8] for the symbol-level PTA. This is carried out to study the effect that the size of δ has on the case of the sparse binary $\mathcal{H}$ matrix. A $(15, 7)$ RS code is again transmitted through an AWGN channel using a 16-QAM modulation scheme with Gray mapping. These results can be seen in Fig. 4.2.

Fig. 4.2 shows that values of $\delta < 0.01$ all yield a very similar performance. This gives a different result when compared to the symbol-level implementation in [8], which shows an increase in error correction performance the more the values of δ decreases. The reason why larger values of δ, for the bit-level case, work the same as the optimum value of 0.001 shown in [8] is once again due the sparse structure of matrix $\mathcal{H}$. For the symbol-level PTA, all the indices matching $\mathcal{K}$ participate in every check, due to the complete dense structure of the parity submatrix. This means that the $\mathcal{K}$ indices are corrected based on (2.38), regardless if they are right or wrong, depending on the result of the syndrome check equation for each row of H. Therefore, a smaller value of δ is required for the correction step to obtain a more accurate result. This is because the small values of δ ensure the additions

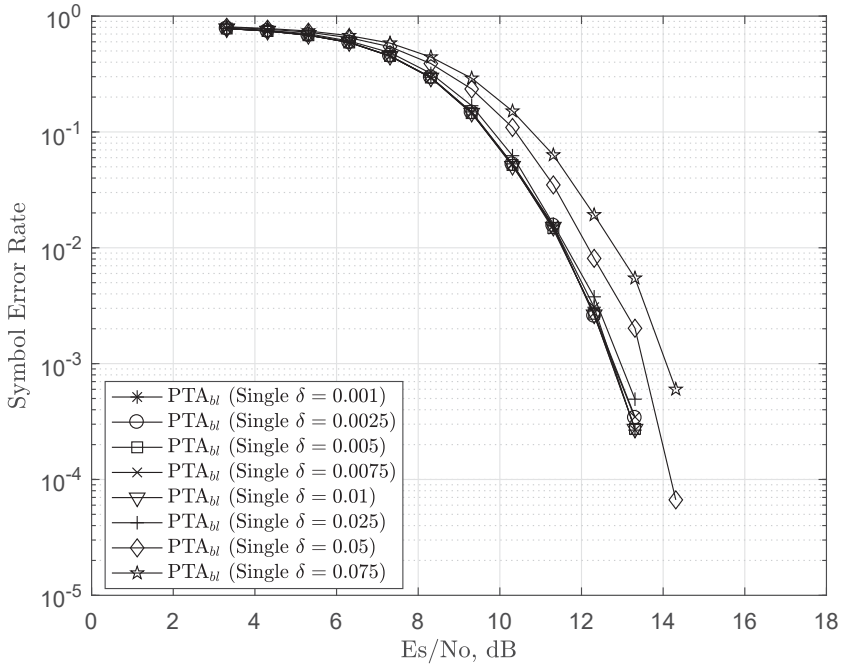


FIGURE 4.2: Performance comparison of the PTA$_{bl}$ based on the value of δ used.

and subtractions made during correction do not largely change the values of the reliabilities which could cause the decoder to select wrong symbols that also satisfy the syndrome checks. This is not the case for the binary image of the H matrix. As mentioned in section 4.2.1.2, the sparse structure of $\mathcal{H}$ ensures that not all the indices matching $\mathcal{K}$ take part in every syndrome check calculation. This means that only the participating $\mathcal{K}$ indices are corrected during each syndrome check. Therefore, larger value of δ, not exceeding 0.01, can also be used in the decoding process for the PTA$_{bl}$ and still yield an optimum performance. This proves to be advantageous as larger values of δ enable the PTA to decode the received vector with less iterations [8].

4.2.2.2 Performance Analysis of Half Rate codes

A performance comparison between the PTA algorithm when implemented on a bit-level and symbol-level is now presented. The same simulation parameters are used as in the case for Fig. 4.1 and Fig. 4.2. The PTA$_{bl}$ is run with a correction

factor of $\delta = 0.01$ while the symbol-level PTA is implemented with a correction factor of $\delta = 0.001$ as used in [8]. The results of the simulation are shown in Fig. 4.3.

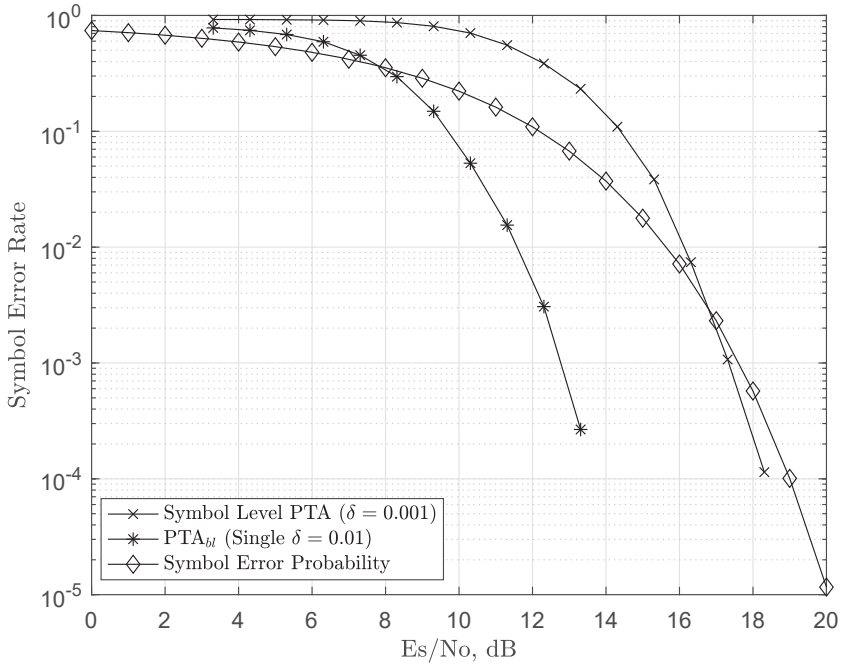


FIGURE 4.3: Performance of the PTA$_{bl}$ compared to the symbol-level PTA for a $(15, 7)$ RS code

From Fig. 4.3, it can be seen that a significant gain is experienced when the PTA is implemented on a bit-level when compared to that of the symbol-level. The PTA$_{bl}$ algorithm outperforms the symbol-level PTA by well over 4dB at a SER of 10^{-3}. Working at a bit-level means, that for the same field, the decoder uses a larger parity check matrix to decode a longer received vector than the case of a symbol-level implementation. That is, an increased complexity cost is inured when using the PTA algorithm to decode the received vector at a bit-level when compared to the symbol-level.

4.2.2.3 Performance Analysis of High Rate codes

An additional simulation is carried out to test the performance of the PTA$_{bl}$ algorithm under high rate conditions. For this simulation, a $(15, 11)$ RS code is once again transmitted using a rectangular 16-QAM modulation scheme with Gray mapping through an AWGN channel. The PTA$_{bl}$ is still corrected using a single value of $\delta = 0.01$, while being compared in terms of SER performance against the symbol-level PTA running under the same conditions as those used for the half rate codes. The results for these simulations are presented in Fig. 4.4.

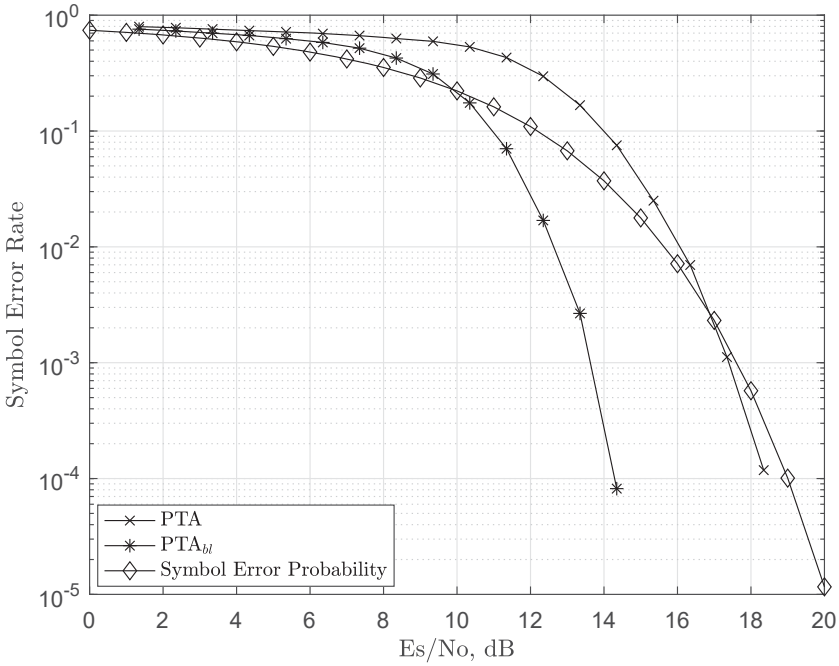


FIGURE 4.4: Performance of the PTA$_{bl}$ compared to the symbol-level PTA for a $(15, 11)$ code

Fig. 4.4 shows that the PTA$_{bl}$ algorithm yields a SER performance gain of about 3dB when compared to the symbol-level PTA algorithm at an SER value of 10^{-3}. The gains obtained by working at a bit-level, once again, come at the cost of computational complexity when compared to the symbol-level implementation.

4.2.3 Conclusions From the Preliminary Results

The bit-level implementation of the PTA decoder significantly outperforms the symbol-level implementation of the PTA. The algorithm takes advantage of the sparsity provided by working at a bit-level to facilitate more efficient message passing of the soft information. However, this improved performance of the bit-level implementation of the PTA decoder comes at the expense of additional complexity when compared to the symbol-level implementation. This is because, for the same field, to the bit-level implementation of the PTA decodes a longer received vector with the aid of a larger parity check matrix when compared to the symbol-level implementation.

Due to the improved performance of the PTA algorithm at the bit-level, the basic implementation and design of iteratively obtaining the information set required in the decoding process by the proposed decoding algorithm is based on a similar approach.

It is important to note that the performance gain obtained when comparing the PTA_{bl} and the symbol-level PTA is not considered a fair comparison due to the algorithms working on different levels. That is, bit-level decoders use binary schemes to perform error correction as opposed to the symbol-level decoder which use non-binary algorithms. Further simulations performed in this research compare bit-level decoder separately from the symbol-level implementations for the purpose of giving a fair comparison in the results obtained.

4.3 The Proposed Decoding Scheme

To understand how the proposed decoder works, the relevant notation is first established. The decoder works on a bit-level, this means a binary image expansion is performed on H to obtain $\mathcal{H}$. For ease of notation, $M = (nm - km)$, $N = nm$ and $K = km$.

A RS codeword c in the extension field is now considered. The codeword c is transmitted using a selected modulation scheme through a noisy channel. The

soft information received at the output of the channel is then used to create the reliability matrix β. The matrix β has the dimensions $2 \times N$, where the 1^{st} and 2^{nd} row of β represent the reliability of the selected bit index being a "0" and a "1" respectively as shown in (2.68).

Once the matrix β is created, each of the columns is then scaled using (4.1). This is carried out to ensure that the reliabilities in each column of β add up to 1 before being fed into the decoder. For each iteration of the proposed algorithm, the decoding process can be summarised by the following steps:

1. *Finding the maximum bit reliabilities:* The maximum reliabilities in each column of β are identified and arranged in the vector A as shown in (4.4)

$$A = [A_1, A_2, ..., A_N] \tag{4.4}$$

where $A_j = \text{argmax}(\beta_j)$, $0 \leq j \leq N$ and β_j represents each column of the matrix β. These reliabilities are then sorted in ascending order. The original indices of the reliabilities are identified as well and stored in terms of their ascending order in the vector Y

$$Y = [\ Y_1\ ,\ Y_2\ ,\ ...\ ,\ Y_N\ ]. \tag{4.5}$$

The K highest values of the A are considered to be the most reliable. For this reason, row reduction is performed on the rearranged columns of $\mathcal{H}$ based on the indices of Y. An approach similar to information set decoding technique presented in [30] is applied in that the indices of the K highest reliabilities match the parity submatrix obtained from the row reduction of the now quasi-systematic structure of $\mathcal{H}$. The matrix obtained from the row reduction of the rearranged columns of $\mathcal{H}$ is denoted as $\mathcal{H}^+$.

It is important to note that the matrix $\mathcal{H}$ has a full row rank. This means that there exists a total of M independent columns present in $\mathcal{H}$. However, as noted in section 4.2.1.1, there is no guarantee that the M least reliable indices found in Y will match these columns during row reduction. This means that not all the most reliable K indices will match the parity submatrix for

every row reduction operation performed on $\mathcal{H}$. When this happens, any M bit indexes that matches an identity submatrix of $\mathcal{H}^+$ are considered to be unreliable, and any K bit indexes that match the parity submatrix of $\mathcal{H}^+$ are considered to be reliable.

2. *Hard-decision detection and the Syndrome check:* Hard-decision detection is then carried out on β, using (2.28), to obtain the vector $\widehat{cb}$. The syndrome is then calculated similar to (2.13) by getting the scalar product of the vector $\widehat{cb}$ and each row of $\mathcal{H}^+$ as shown in (4.6)

$$S_i = \widehat{cb} \cdot \mathcal{H}_i^+, \tag{4.6}$$

where $1 \leq i \leq M$ is used to denote each row of $\mathcal{H}^+$ and each value of the syndrome vector, S. Due to the decoder working on a bit-level, (2.18) and (2.45) can be used to rewrite the syndrome calculation in (4.6) to the form shown in (4.7)

$$S_i = \sum \widehat{cb}_{t_i}, \tag{4.7}$$

where t_i represents all the indices of the participating bits of $\widehat{cb}$ in the i^{th} syndrome check equation and is expressed as shown in (4.3)

3. *Obtaining the votes:* During the syndrome calculation, votes are cast for each bit. Each bit gets a vote of either being a "0" or a "1" based on the extrinsic information. This means, for each row, all the participating bits except the one being investigated take part in the vote. Based on the vote, the syndrome calculation in (4.7) is rewritten as

$$S_i = \widehat{cb}_y + \sum \widehat{cb}_{t'}, \tag{4.8}$$

where $y \in t_i$ represents the index of the bit being voted for. The subvector t' represents the set of indices in t_i without the bit index y. Assuming the set of bits $\widehat{cb}_{t'_i}$ are all correct and $S_i = 0$, the calculation of votes can be derived

from the extrinsic information using (2.16) as

$$\widehat{cb}_y = \sum \widehat{cb}_{t'}. \tag{4.9}$$

From (4.9), $\widehat{cb}_y$ is found to be either a "0" or "1". This counts as a vote for the bit $\widehat{cb}_y$. This process is repeated for all participating bits of $\widehat{cb}$ in every row of $\mathcal{H}^+$, with the votes being stored in the vector V as seen in (4.10)

$$V = \begin{bmatrix} V_{0,1}, & V_{0,2}, & \dots & V_{0,N} \\ V_{1,1}, & V_{1,2}, & \dots & V_{1,N} \end{bmatrix}, \tag{4.10}$$

where $V_{0,j}$ and $V_{1,j}$ represents the total votes each bit index gets for being a "0" and a "1" respectively, for all the rows of $\mathcal{H}^+$,

4. *Obtaining the confidence rating:* The first step to obtaining the confidence rating of each bit is to get the voting ratios, ϑ, as shown in (4.11)

$$\vartheta = \begin{bmatrix} \frac{V_{0,1}}{V_{T,0}}, & \frac{V_{0,1}}{V_{T,1}}, & \dots & \frac{V_{0,N}}{V_{T,N}} \\ \frac{V_{1,1}}{V_{T,1}}, & \frac{V_{1,2}}{V_{T,2}}, & \dots & \frac{V_{1,N}}{V_{T,N}} \end{bmatrix}, \tag{4.11}$$

where each $V_{T,j}$ represents the total number of votes each bit index gets and it is computed as

$$V_{T,j} = V_{0,j} + V_{1,j}. \tag{4.12}$$

The confidence rating, Λ, for each bit is then calculated by dividing the voting ratios in ϑ by a value of ρ as shown in (4.13)

$$\Lambda = \frac{\vartheta}{\rho}, \tag{4.13}$$

where ρ represents the divisor which is a constant predefined value input during the initialisation of the algorithm. The values in the matrix Λ can be represented as

$$\Lambda = \begin{bmatrix} \Lambda_{0,1}, & \Lambda_{0,2}, & \dots & \Lambda_{0,N} \\ \Lambda_{1,0}, & \Lambda_{1,2}, & \dots & \Lambda_{1,N} \end{bmatrix}, \tag{4.14}$$

5. *Updating β:* The values in Λ represent the level of confidence that a bit is correct and should be updated in β. The update works by adding the indexed confidence ratio to the corresponding reliability in β based on the vote in (4.9) for being either a "0" or "1". The update can be summarized as follows

$$\beta^{(f+1)}_{(\widehat{cb}_y,j)} = \beta^{f}_{(\widehat{cb}_y,j)} + \Lambda_{(\widehat{cb}_y,j)}, \tag{4.15}$$

where f is used to denote the current iteration number of the decoder. The value of $\widehat{cb}_y$ is either a "0" or a "1". This value is used to identify which of the reliabilities in β, indexed at the j^{th} column, to update with the corresponding confidence rating. The notation y represents the index of the bit considered during (4.9). All the reliabilities in β are updated based on their participation in each row of the matrix $\mathcal{H}^+$. After all the N indices are updated, $\beta^{(f+1)}$ is scaled as shown in (4.1) to ensure all the columns add up to 1 in preparation for the next iteration.

4.3.1 The Decoding Condition and Thresholds

The proposed algorithm works iteratively and has two stopping conditions. The first is when $S = 0$. The second is when the decoding condition is met. The decoding condition is based on information set decoding. That is, it makes use of a set of K bits to re-encode the decoded codeword based on the rearranged systematic structure of $\mathcal{H}^+$.

The decoding condition is met if the algorithm can ascertain that the information set of K bits with indices matching the parity submatrix is correct. If the information set is determined to be correct, then the remaining M bits that match the identity matrix can be decoded.

In order to understand how this works, consider the syndrome equation between $\widehat{cb}$ and the rearranged systematic matrix $\mathcal{H}^+$ in the form shown in (4.16).

$$S = (\widehat{cb}_{Y_M} \times I^{\top}_{\mathcal{H}^+}) + (\widehat{cb}_{Y_K} \times Q^{\top}_{\mathcal{H}^+}), \tag{4.16}$$

where $I_{\mathcal{H}^+}^{\top}$ and $Q_{\mathcal{H}^+}^{\top}$ match the transpose of the column indices of identity and parity submatrices of $\mathcal{H}^+$, while $\widehat{cb}_{Y_M}$ and $\widehat{cb}_{Y_K}$ match the indices of $\widehat{cb}$ that correspond to the columns of the identity and parity submatrices of $\mathcal{H}^+$ respectively. Due to $I_{\mathcal{H}^+}$ being an identity matrix, (4.16) can be reduced further as shown in (4.17).

$$S = \widehat{cb}_{Y_M} + (\widehat{cb}_{Y_K} \times Q_{\mathcal{H}^+}^{\top}), \tag{4.17}$$

If the $\widehat{cb}_{Y_K}$ indices are correct, the bit values of $\widehat{cb}_{Y_M}$ can be obtained by assuming $S = 0$ and using (2.16) to rewrite (4.17). This yields the expression in (4.18).

$$\widehat{cb}_{Y_M} = \widehat{cb}_{Y_K} \times Q_{\mathcal{H}^+}^{\top}. \tag{4.18}$$

Due to the decoder working on a bit-level, (2.18) and (2.45) can once again be used to reduce (4.18) by considering only the participating indices to give

$$\widehat{cb}_{Y_{M_i}} = \sum \widehat{cb}_{t_{K_i}} \tag{4.19}$$

where $\widehat{cb}_{Y_{M_i}}$ and $\widehat{cb}_{t_{K_i}}$ are the participating bits of $\widehat{cb}_{Y_M}$ and $\widehat{cb}_{Y_K}$ in the i^{th} row of $\mathcal{H}^+$ respectively.

The algorithm is able to determine if the information set with the K bits is correct by setting a threshold. The threshold, τ, is defined as the minimum number of syndrome check equations each of the K bits in the information set should satisfy for (4.19) to be applied in the decoding of the received vector. The higher the value of τ, the more confidence the algorithm has that the most reliable K indices are correct. However, this comes at the expense of the algorithm running more iterations.

The main advantage of using the decoding condition as a stopping criteria is that the algorithm is able to use K bits to decode an entire received vector of length N. This assists in reducing the number of iterations required to decode the received vector, because the algorithm does not have to confirm if every single bit is correct before it can break the iterative decoding process.

4.4 Results and Analysis

In this section, simulation results for the proposed iterative decoder and its variants are presented. For ease of notation, the proposed decoder is referred to as the k-Bit Decoding algorithm and is denoted as the kBD algorithm.

4.4.1 Analysis of the Proposed Bit-Level Decoding Algorithm

Simulations are run to find an optimum performance conditions for the kBD algorithm using different values of τ and ρ. The performance of the kBD algorithm using different values of τ is first investigated. The performance of the kBD is measured in terms of error correction and iterative convergence rate. A nearly half rate $(15, 7)$ RS code is used in this simulation, with the symbols being transmitted through an AWGN channel using a 16-QAM modulation scheme with Gray mapping. The value of $\rho = 50$ is used for these simulations. Results for the simulations in terms of Bit Error Rate (BER) and the average number of iterations are shown in Fig. 4.5 and Fig. 4.6 respectively.

From Fig. 4.5 it can be seen that the error correction performance of the kBD algorithm works best when $\tau \geq 7$. It can also be seen from Fig. 4.6 that working with $\tau = 7$ presents a more efficient kBD algorithm. This is because it requires less than half of the average number of iterations used by kBD with $\tau = 10$ during the decoding process, to achieve a similar BER performance.

Tests to determine the optimum value of ρ are also carried out. Similar conditions are used for transmission of the $(15, 7)$ RS code. The kBD decoder is implemented with a $\tau = 7$. The results for these simulations are presented in Fig. 4.7 and Fig. 4.8.

The kBD with values of $\rho \geq 50$ are able to achieve a slightly higher gain in terms of BER when compared to the kBD with values of $\rho \leq 30$ as seen in Fig. 4.7. The kBD with $\rho = 50$ is shown to be an efficient decoder by requiring less iterations to yield a comparable BER performance to the other versions of the algorithm,

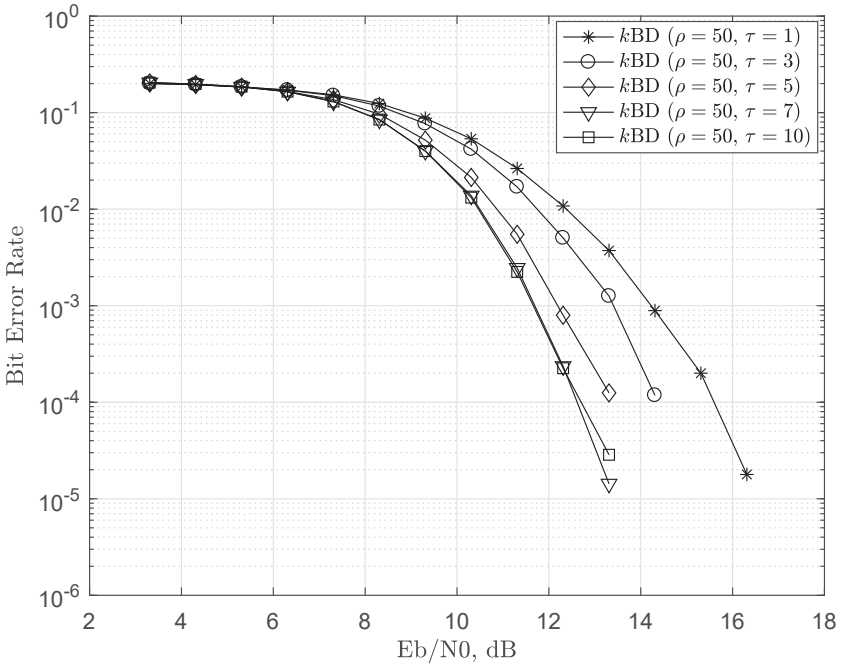


FIGURE 4.5: Performance comparison of the kBD based on different τ values in terms of BER for a (15,7) RS code.

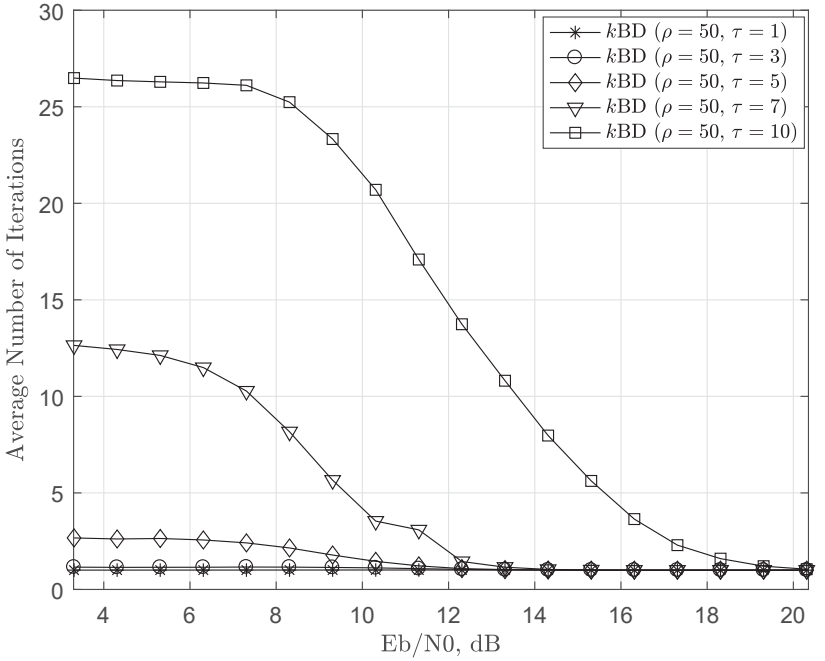


FIGURE 4.6: Performance comparison of the kBD based on different τ values in terms of number of iterations for a (15,7) RS code.

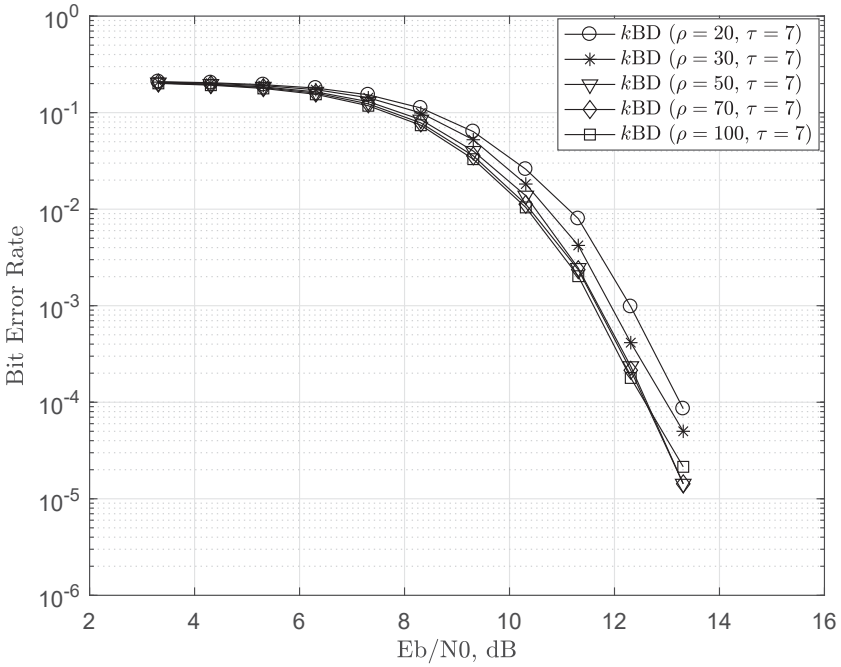


FIGURE 4.7: Performance comparison of the kBD based on different ρ values in terms of BER for a (15,7) RS code.

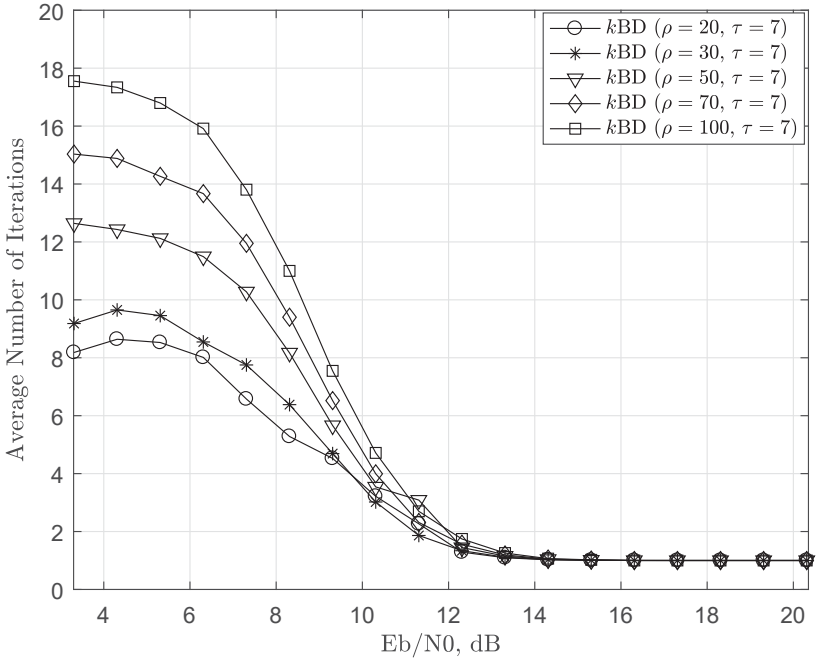


FIGURE 4.8: Performance comparison of the kBD based on different ρ values in terms of average number of iterations for a (15,7) RS code.

with values of $\rho > 50$, as seen in Fig. 4.8. Based on this, the kBD with $\rho = 50$ and $\tau = 7$ is used to benchmark the performance of the proposed algorithm with other iterative soft-input soft-output bit-level decoders.

4.4.2 Performance Analysis of the Proposed Bit-Level Decoding Algorithm

Half rate codes

In this section the performance of the kBD algorithm is benchmarked against the PTA$_{bl}$ decoder and the ABP algorithm. Simulations are run on a nearly half rate $(15, 7)$ RS code. The ABP is simulated with the value of $\eta = 0.05$ and is set to run with a the maximum number of 20 iterations [6]. The PTA$_{bl}$ is run with a value of $\delta = 0.01$, as this value yields an optimum performance as presented in Fig 4.2. For these simulation a version of the kBD algorithm with a lower computational complexity is tested as well. This version of the kBD only performs Gaussian row reduction once on $\mathcal{H}$, in the first iteration, to obtain $\mathcal{H}^+$. The same $\mathcal{H}^+$ is then used throughout the entire iterative decoding of the received vector. This version of the kBD is referred to as the none transform version of the kBD. This is because it does not iteratively perform Gaussian row reduction operations to transform $\mathcal{H}^+$, based on the soft bit reliabilities, during decoding of the received vector. As a result of the reduced Gaussian row reduction operations, this version of the kBD has a lower computational complexity cost than the original implementation of the kBD algorithm. For the ease of notation, the none transform version of the kBD is denoted as kBD$_{nt}$.

The same simulation parameters for encoding, modulation and transmission as those used obtain the results in Fig. 4.5, are utilised for this set of simulations. The results for these simulations are presented in Fig. 4.9 and Fig. 4.10.

It can be seen from Fig. 4.9 that the kBD algorithm experiences a gain of about 0.6dB when compared to the ABP algorithm with $\eta = 0.05$ at a BER of 10^{-3}. The kBD decoder also outperforms the PTA$_{bl}$ decoder with $\delta = 0.01$, but with a

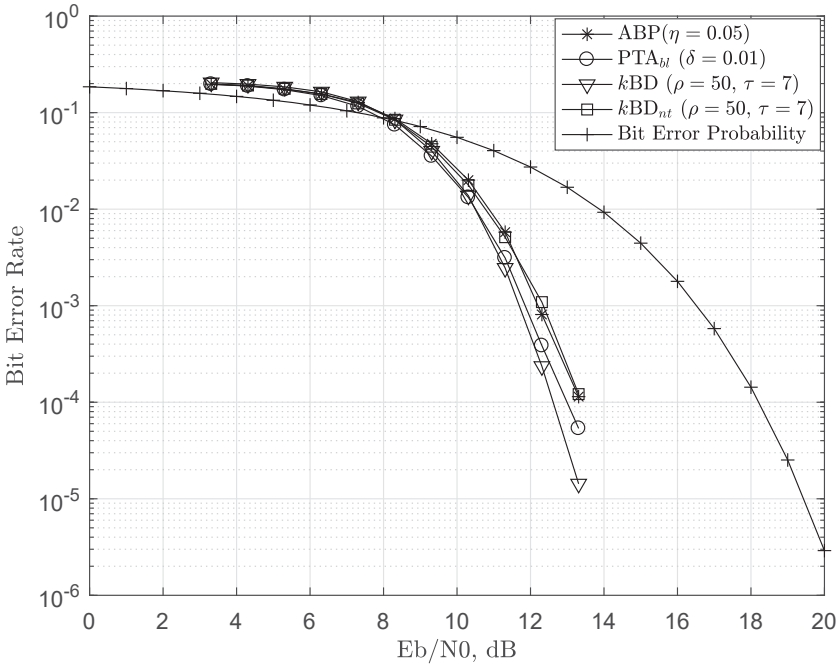


FIGURE 4.9: Performance comparison of the ABP, PTA$_{bl}$, kBD and kBD$_{nt}$ in terms of BER for a (15,7) RS code.

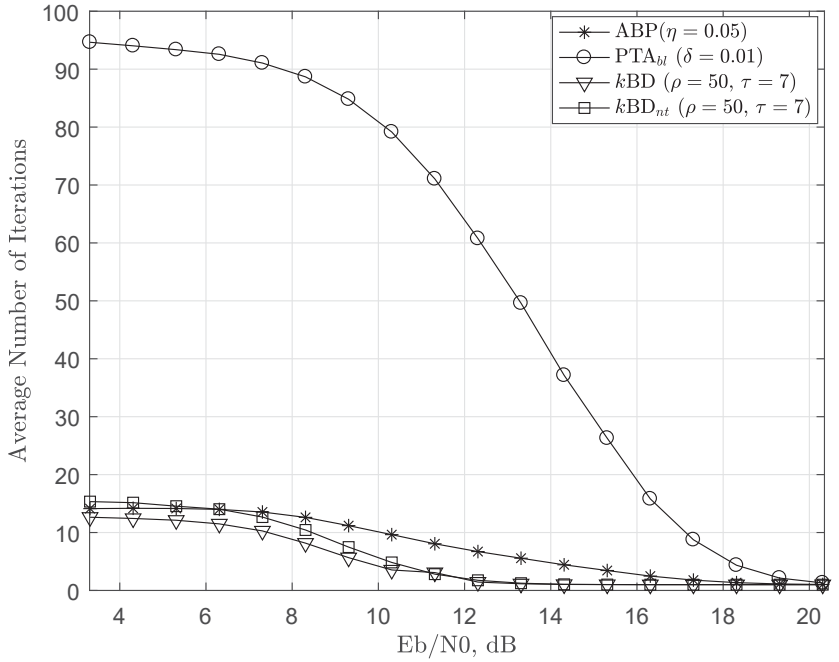


FIGURE 4.10: Performance comparison of the ABP, PTA$_{bl}$, kBD and kBD$_{nt}$ in terms of average number of iterations for a (15,7) RS code..

smaller gain of about 0.15dB for the same BER value.

The $k\text{BD}_{nt}$ algorithm, being less complex due to its lack of iterative $\mathcal{H}^+$ transformations, yields a similar performance to that of the ABP algorithm with $\eta = 0.05$. However, the $k\text{BD}_{nt}$ decoder is outperformed by both the kBD and the PTA_{bl} algorithm by about 0.63dB and 0.42dB respectively for a BER of 10^{-3}. It is important to note that the PTA_{bl} algorithm is significantly more computationally complex than all the bit-level decoders used in the simulation. This is because it performs Gaussian row reduction operations to transform the matrix $\mathcal{H}$ for each of the numerous iterations required during the decoding process, as seen in Fig. 4.10. This provides justification for the use of the $k\text{BD}_{nt}$ over the PTA_{bl} decoder whenever a complexity-performance tradeoff is required between the two algorithms.

In Fig. 4.10, the iterative converegnce rate of both version of the kBD only differ slightly from the ABP algorithm for the low SNR values. However, as the SNR values increase the iterative converge for both versions of the kBD reduces faster when compared to the ABP decoder.

From Fig. 4.9 and Fig. 4.10, the kBD is seen to be a good performing iterative bit-level soft-decision decoding algorithm. This is because it is able to achieve gains in BER performance, while running for fewer iterations than all other bit-level soft-decision decoding algorithms used in the $(15, 7)$ RS code simulations.

High rate codes

Additional simulations are carried out to test the performance of the proposed variations of the kBD algorithms under high rate conditions.Working at a high rate means working with a $\mathcal{H}$ matrix with less rows when compared to a half rate code. That is, there are fewer syndrome check equations than in the case for the $(15, 7)$ RS code. This means the each of the K bits in the information set participate in less syndrome check equations. H. With respect to this, a new set of simulations to determine an optimum value for τ is carried out. For these simulation, a $(15, 11)$ RS code is once again transmitted through an AWGN channel using a 16-QAM modulation scheme with Gray mapping. The results for

these simulations are displayed in Fig. 4.11 and Fig. 4.12.

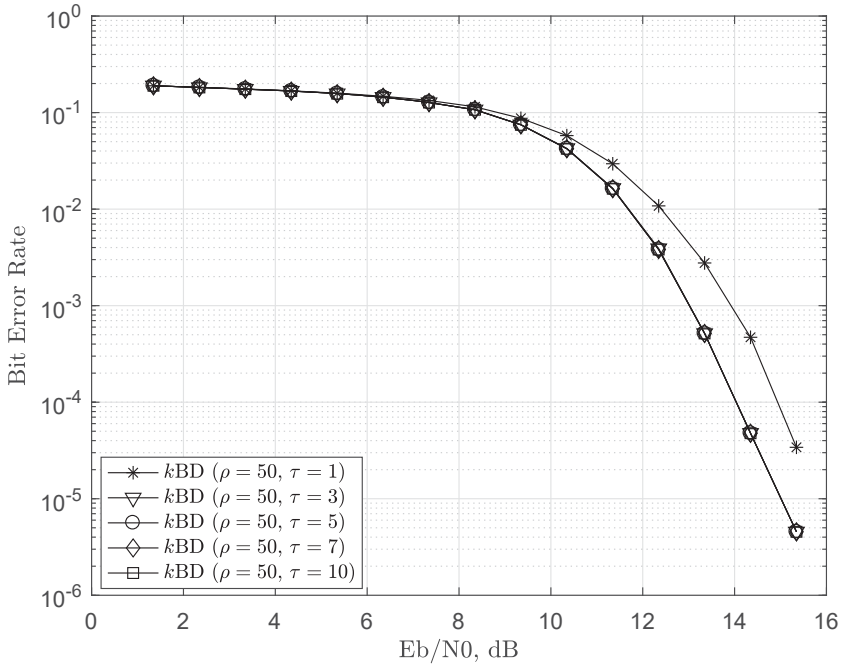


FIGURE 4.11: Performance comparison of the kBD based on different τ values in terms of BER for a (15,11) RS code.

It can be seen from Fig. 4.11 that the BER performance of the algorithm is the same for values of $\tau \geq 3$. The main difference in the performance of the algorithm can be seen in Fig. 4.12. The kBD with $\tau = 3$ requires less than half of the average number of iterations to decode the received vector, when compared to the kBD algorithm with values of $\tau \geq 5$. As mentioned in section 4.3.1, the reason why the kBD with $\tau = 3$ requires less iterations than values of $\tau > 3$, is because the algorithm only has to ensure that each bit in the information set, $\widehat{cb}_{Y_K}$, satisfies 3 syndrome check equations as opposed to the cases when $\tau > 3$. This enables the kBD algorithm to meet the decoding condition with fewer iterations than when $\tau \geq 5$. Also, due to the $(15, 11)$ RS code having a $\mathcal{H}$ matrix with less syndrome check equations, the value of $\tau = 3$ when applied to the kBD is sufficient to give an optimum BER performance as opposed to the case of the $(15, 7)$ RS code where $\tau = 7$ is required for an optimum performance. Hence, the kBD algorithm with $\rho = 50$ and $\tau = 3$ is used when benchmarking the performance of the decoder with the ABP and the PTA$_{bl}$ for high rate codes. No modifications are made to

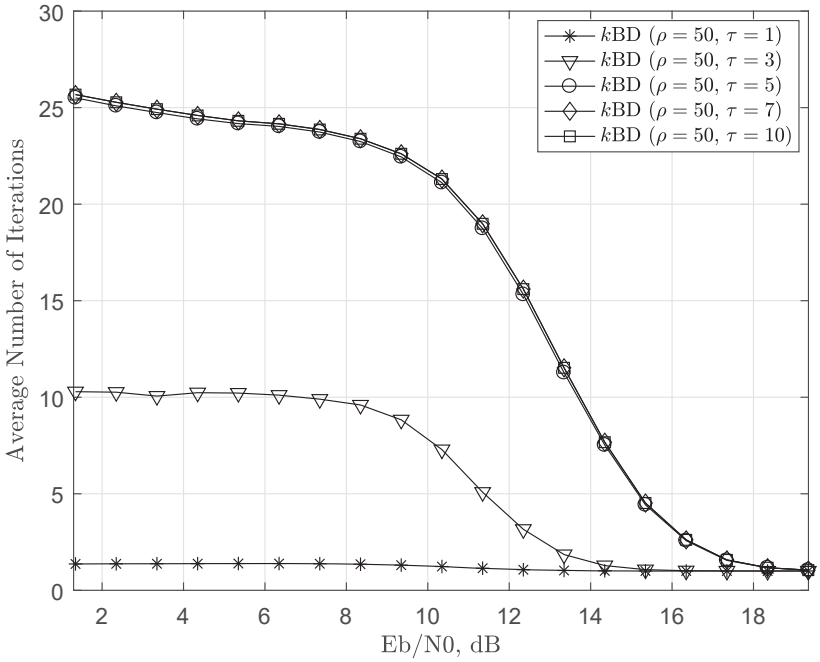


FIGURE 4.12: Performance comparison of the kBD based on different ρ values in terms of average number of iterations for a (15,11) RS code.

the implementations of the PTA$_{bl}$ and the ABP algorithm from the half rate case. All the algorithms are run under the same conditions as the half rate case. The results for this set of simulations can be seen in Fig. 4.13 and Fig. 4.14.

From the results presented in Fig. 4.13, the performance of the kBD algorithm matches the performance of the PTA$_{bl}$ decoder. This performance is achieved by the kBD algorithm while running at a lower average number of iterations than the PTA$_{bl}$, during the decoding process, as seen in Fig. 4.14. The kBD also compares favourably to the ABP algorithm by yielding a slight BER performance gain at an BER value of 10^{-5}. The kBD$_{nt}$ algorithm is also only slightly outperformed by less than 0.1dB at a BER of 10^{-5} when compared to the PTA$_{bl}$ decoder and achieves a similar BER performance to that of the ABP algorithm, while running at an average number iterations that is less than both algorithms. This is important note because the kBD$_{nt}$ algorithm does not require row reduction operations to be performed iteratively, as in the case of the other bit-level decoders used in the simulations. This justifies the use of the kBD$_{nt}$, when selecting a bit-level soft-input soft-output decoder, in the case of a complexity-performance tradeoff. This

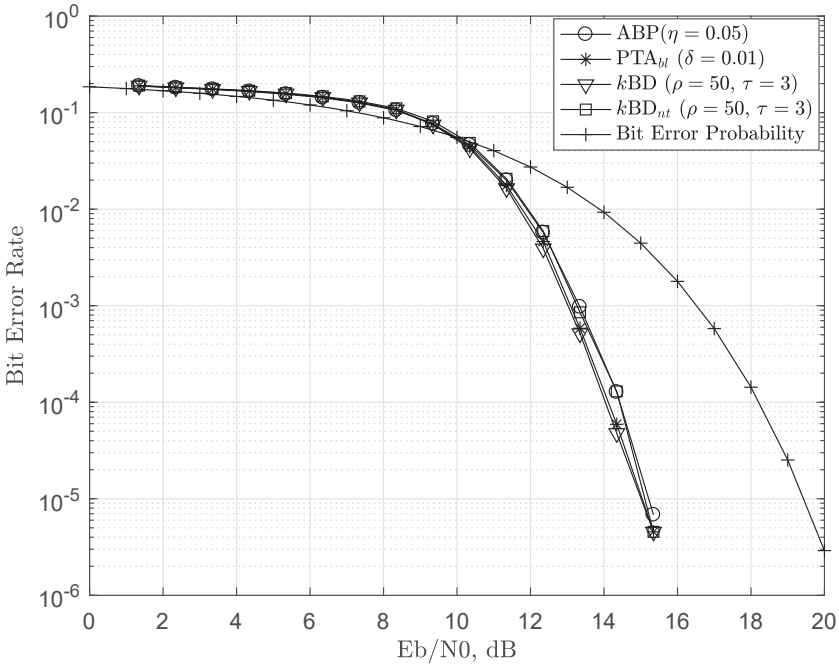


FIGURE 4.13: Performance comparison of the ABP, PTA$_{bl}$, kBD and kBD$_{nt}$ in terms of BER for a (15,11) RS code..

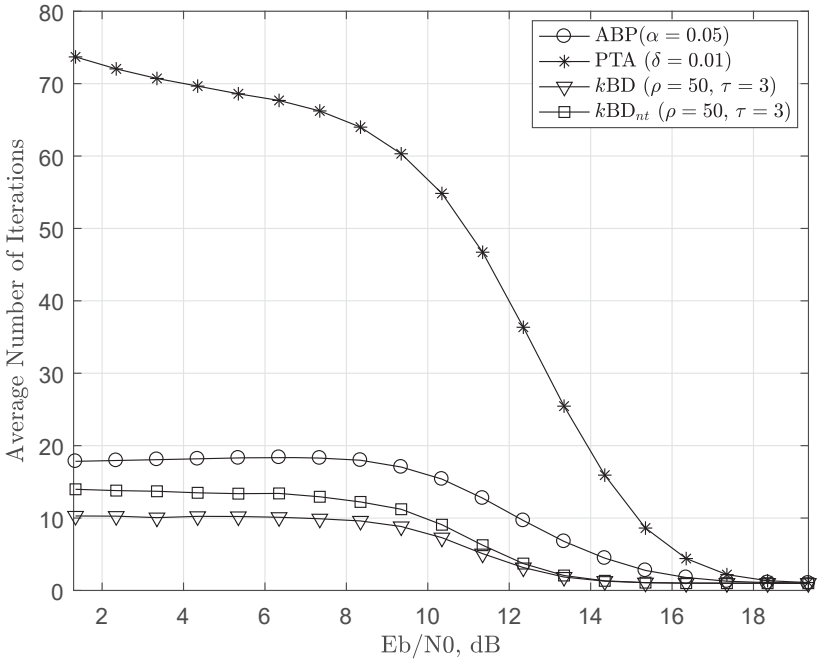


FIGURE 4.14: Performance comparison of the ABP, PTA$_{bl}$, kBD and kBD$_{nt}$ in terms of average number of iterations for a (15,11) RS code..

is because the $k\text{BD}_{nt}$ operates at a lower computational complexity cost while yielding a good BER performance, when compared to the other bit-level decoders used in the simulations.

The algorithm is also run for a $(31, 25)$ RS code so as to benchmark the proposed decoder against the results presented in [6]. For these simulations, the proposed algorithm is also tested against one of the modification of the ABP decoder refereed to as the ABP-HDD(20,1) algorithm. This version of the ABP works alongside a 'genie aided' hard decision decoder (HDD) [6]. This version of the ABP work iteratively however, it does not converge to a codeword. Instead it runs for all the 20 iterations while outputting a codeword that is fed into the HDD with a genie aided stopping condition. The ABP in this implementation therefore works in a similar way to a list decoder. The ABP only selects the most likely codeword if the HDD is not able to obtain the decoded vector from the list of codewords generated from each of the 20 iterations. The result of the ABP-HDD(20,1) is described as 'optimistic' in [6] due to the inclusion of the genie aided HDD. This is because the genie aided stopping criteria already knows the correct codeword and breaks the decoding process once the correct codeword is obtained instead of letting the iterative algorithm converge to the most likely codeword[6]. The genie aided HDD is added to the algorithm to prevent the decoder from running all 20 iterations and therefore speeding up the decoding process. This is because the algorithm is a double decoder. This means there is a significant increased complexity when compared to the generic version of the ABP. This is especially due to the algorithm running all 20 iterations during each decoding process, even for higher SNR values when less iterations are required. As highlighted in Section 4.1, for reasons of complexity and also the use of a genie aided stopping criteria, the ABP-HDD was not used in the other benchmarking simulations.

The simulations are tested using BPSK modulation and the RS code is transmitted through an AWGN channel so as to obtain the results in a similar way to [6] [43]. The Maximum Likelihood (ML) lower bound is obtained using the technique presented in [16] [129]. The results for these simulations are presented in Fig. 4.15.

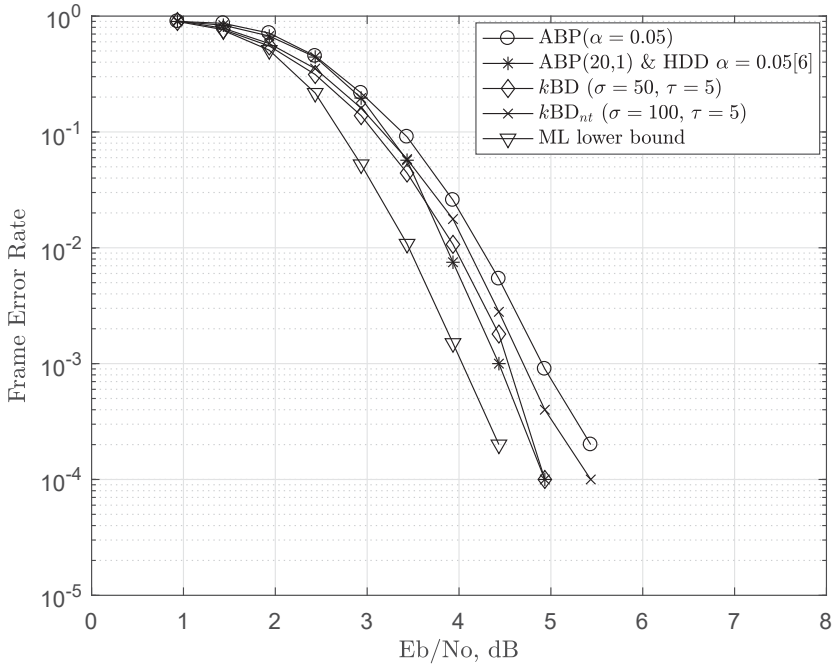


FIGURE 4.15: Performance comparison of the ABP, PTA$_{bl}$, kBD and kBD$_{nt}$ for a $(31, 25)$ RS code.

From the results in Fig. 4.15 it can be seen that the error correction performance of the kBD and kBD$_{nt}$ are quite favourable when compared to the generic ABP decoder. The kBD algorithm outperforms the generic version of the ABP by about 0.5dB at an FER of 10^{-3}. The less comples kBD$_{nt}$ yields a gain of about 0.25dB when compared to the generic form of the algorithm. However, the kBD has a loss of about 0.6dB when compared to the ML lower bound graph and a loss of less than 0.1dB when compared to the ABP-HDD(20,1) [6], for the same FER value. However it is important to note that the results for the ABP-HDD(20,1) are optimistic, due to the genie aided stopping condition, while still being a more complex decoder.

4.5 Complexity Analysis

4.5.1 Time Complexity

In this section, the computational complexity cost for the kBD decoding technique is compared to that of the ABP and the PTA$_{bl}$ algorithms. The complexity cost of the bit-level soft-input soft-output decoders is measured in terms of the total number of operations carried out for the average number of iterations required in the entire decoding process. In order to better represent the computation calculations, notation is established. The average row weight and column weight of the participating bits indexed by t are denoted as W_r and W_c respectively. The average row weight of indices that match the parity submatrix of $\mathcal{H}^+$ are denoted using W_k. The notation W_k is also used to denote the average row weight for computations that obtain the extrinsic bit information. The computational complexity cost for a single iteration of the ABP, the PTA$_{bl}$, kBD and the kBD$_{nt}$ decoders are all summarised in the Table 4.1, Table 4.2, Table 4.3 and Table 4.4 respectively.

TABLE 4.1: Summary of the overall complexity for the ABP decoder.

Decoding Stage	**Stage description**	**Number of Operations**
Obtaining the vector $\|L\|$	Finding absolute values for L	N
Sorting of $\|L\|$	$\|L\|$ sorted in ascending order	N
Adapting the $\mathcal{H}$ matrix	Row reduction of $\mathcal{H}$	$M^2 \times N$
Obtaining the vector L_e	Calculating the extrinsic information as shown in (2.31)	$(M \times W_k^2) + (N \times W_c)$
Updating the vector L	Adding $L_e(cb_j)$ to $L(cb_j)$	N
Obtaining the hard-decision vector $\widehat{cb}$	Assigning one of the binary values to each $L(cb_j)$	N
Syndrome Check	Performing the calculation represented in (4.7)	$M \times W_r$
Overall Complexity of the ABP decoder		
Total = $N + N + (M^2 \times N) + (M \times W_k^2) + (N \times W_c) + N + N + (M \times W_r)$		
Time complexity = $\mathcal{O}(M \times W_k^2)$ [12]		

TABLE 4.2: Summary of the overall complexity for PTA$_{bl}$ decoder.

Decoding Stage	**Stage description**	**Number of Operations**
Finding the reliability vector	Searching for maximum values in β	N
Sorting of reliabilities	Reliabilities sorted in ascending order	N
Transforming matrix $\mathcal{H}$	Row reduction of $\mathcal{H}$	$M^2 \times N$
Obtaining the hard-decision vector $\widehat{cb}$	Assigning one of the binary values in β	N
Syndrome Check	Performing the calculation represented in (4.7)	$M \times W_r$
Correction step	Updating the β reliabilities based on each Syndrome check	$M \times W_r$
Scaling reliabilities in β	If $S \neq 0$, (4.1) is applied in preparation for the next iteration	$(N + 2N)$

Overall Complexity of the PTA$_{bl}$ decoder

$$\text{Total} = N + N + (M^2 \times N) + N + (M \times W_r) + \underbrace{(M \times W_r) + (N + 2N)}_{\text{if } S \neq 0}$$

Time complexity = $\mathcal{O}(M^2 \times N)$

TABLE 4.3: Summary of the overall complexity for the kBD decoder.

Decoding Stage	**Stage description**	**Number of Operations**
Finding the vector $\beta_{\max}$	Searching for maximum values in β	N
Sorting the reliabilities	Reliabilities sorted in ascending order	N
Transforming matrix $\mathcal{H}$	Row reduction of $\mathcal{H}$	$M^2 \times N$
Obtaining the hard-decision vector $\widehat{cb}$	Assigning one of the binary values in β	N
Syndrome Check and vote tallying	Applying (4.7) and (4.9)	$(M \times W_r) + (M \times W_k^2)$
Obtaining the bit confidence rating	Applying (4.12),(4.11) and (4.13)	$(N + 2N + 2N)$
Updating β	Bit reliabilities are updated in β	$M \times W_r$
Checking for decoding condition	Checking if τ is met by the informational set	K
Scaling reliabilities in β	If $S \neq 0$ or τ is not met, (4.1) is applied in preparation for the next iteration	$(N + 2N)$
Applying the Decoding condition	Applying (4.19)	$M \times W_k$ - only applied in the final iteration if $S \neq 0$ when the threshold τ is met

Overall Complexity of the kBD decoder

Total $= N + N + (M^2 \times N) + N + (M \times W_r) + (M \times W_k^2) + (N + 2N + 2N) + (M \times W_r) + K + \underbrace{(N + 2N)}_{\text{if } S \neq 0 \text{ or } \tau \text{ is not met}} + \underbrace{(M \times W_k)}_{\text{last iteration only}}$

Time complexity $= \mathcal{O}(M \times W_k^2)$

TABLE 4.4: Summary of the overall complexity of the $k\text{BD}_{nt}$ decoder.

Decoding Stage	**Stage description**	**Number of Operations**
Finding the vector $\beta_{\max}$	Searching for maximum values in β	N
Sorting the reliabilities	Reliabilities sorted in ascending order	N
Transforming matrix $\mathcal{H}$	Row reduction of $\mathcal{H}$	$M^2 \times N$ - only applied in the first iteration.
Obtaining the hard-decision vector $\widehat{cb}$	Assigning one of the binary values in β	N
Syndrome Check and vote tallying	Applying (4.7) and (4.9)	$(M \times W_r) + (M \times W_k^2)$
Obtaining the bit confidence rating	Applying (4.12),(4.11) and (4.13)	$(N + 2N + 2N)$
Updating β	Bit reliabilities are updated in β	$(M \times W_r)$
Checking for decoding condition	Checking if the τ is met by informational set	K
Scaling reliabilities in β	If $S \neq 0$ or τ is not met, (4.1) is applied in preparation for the next iteration	$(N + 2N)$
Applying the Decoding condition	If τ is met, (4.19) is computed	$(M \times W_k)$ - only applied in the final iteration if $S \neq 0$ when the threshold τ is met

Overall Complexity of the $k\text{BD}_{nt}$ decoder

$$\text{Total} = N + N + \overbrace{(M^2 \times N)}^{1^{st}\text{ iteration only}} + N + (M \times W_r) + (M \times W_k^2) + (N + 2N + 2N) + (N \times W_r) + K + \underbrace{(N + 2N)}_{\text{if } S \neq 0 \text{ or } \tau \text{ is not met}} + \underbrace{(M \times W_k)}_{\text{last iteration only}}$$

$$\text{Time complexity} = \mathcal{O}(M \times W_k^2)$$

To create a clear perspective of the complexities of the algorithms, tables to note the computations that involve 'additions/subtraction', 'multiplications/divisions' and 'other' operations are created for each algorithm. The tables only considers the operations that are unique to at least one algorithm. For example, operations like Gaussian elimination and the syndrome check are present in all algorithms and are therefore ignored in this analysis. The tables with this analysis are presented in Table 4.5, Table 4.6, and Table 4.7 respectively.

TABLE 4.5: Summary of the computational complexity for the ABP decoder.

Decoding Stage	$+/-$	$\times/\div$	other
Obtaining the vector $\|L\|$		N	
Obtaining the vector L_e	$(N \times W_c)$	$(M \times W_k^2)$	
Updating the vector L	N	N	

TABLE 4.6: Summary of the computational complexity for PTA_{bl} decoder.

Decoding Stage	$+/-$	$\times/\div$	other
Finding the reliability vector			N
Correction step	$M \times W_r$		
Scaling reliabilities in β	N		$2N$

TABLE 4.7: Summary of the computational complexity for the kBD and the $k\text{BD}_{nt}$ decoder.

Decoding Stage	$+/-$	$\times/\div$	other
Finding the vector $\beta_{\max}$			N
vote tallying	$(M \times W_k^2)$		
Obtaining the bit confidence rating	N	$(2N + 2N)$	
Updating β	$M \times W_r$		
Checking for decoding condition			K
Scaling reliabilities in β	N	$2N$	
Applying the Decoding condition	$M \times W_k$		

From Table. 4.1, Table. 4.3 and Table. 4.4, it can be seen that the ABP decoder and both versions of the kBD algorithm all have the same time complexity. This is due both algorithms calculating the extrinsic information. However from computational complexities presented in Table. 4.5 and Table. 4.7, the extrinsic

information computation for the ABP requires $(M \times W_k^2)$ multiplications as seen in (2.31) while both version of the kBD use $(M \times W_k^2)$ additions as shown in (4.9).

4.5.2 Complexity measured in Terms of Number of Operations

Research carried out in this thesis also attempted to visually represent the complexities by the decoders. The complexities graphs plotted are based on the of total number of operations as a function of the average number of iterations run by the decoders for each SNR value. This additional set of complexity analysis simulations are carried out so as to investigate the effect of the iterative performance on the complexity cost of the algorithms. The total number operations are obtained from the equations given in Table 4.1, Table 4.2, Table 4.3 and Table 4.4 and multiplied by the average number of iterations run by each algorithm for the different SNR values shown in Fig. 4.10 and Fig. 4.14.

From the simulation performed for the (15,7) RS codes, it is found that $W_r = 14.74$, $W_c = 7.86$, $W_k = 13.74$, $M = 32$ and $N = 60$. The results using these values can be seen in Fig. 4.16.

It can be seen from Fig. 4.16 that both variants of the kBD algorithm require less operations than the ABP with $\eta = 0.05$ and the PTA$_{bl}$ decoder with $\delta = 0.01$. As expected, the kBD$_{nt}$ is the least complex of all the algorithms for the $(15, 7)$ RS code. It is important to note that the kBD$_{nt}$ algorithm is able to exhibiting a comparable BER performance to the high performance ABP algorithm, while being significantly less complex, as shown in Fig. 4.9 and Fig. 4.16 for the nearly half rate RS code. The kBD is also shown to yield a tolerable complexity when compared to both the ABP and the PTA$_{bl}$. For the low SNR values, the computational complexity cost of the kBD is comparable to the ABP. However, as the values of the SNR increase, the complexity cost of the kBD reduces significantly faster than the ABP decoder. From Table 4.2, the PTA$_{bl}$ appears to be the least complex bit-level decoder for operations carried out in a single iteration. However, the numerous iterations required by the PTA$_{bl}$ to decode the received vector,

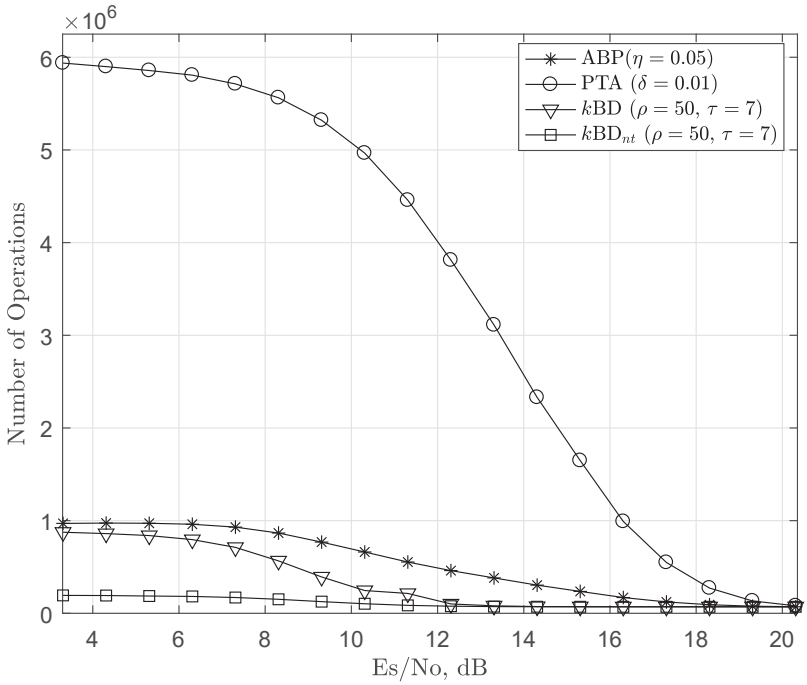


FIGURE 4.16: Complexity comparisons for the bit-level decoders applied to a (15,7) RS code over an AWGN channel using a 16-QAM modulation scheme.

considerably add to the computational complexity cost as seen from Fig. 4.16. This makes the PTA$_{bl}$ decoder significantly more complex than the other bit-level decoders used in the simulations.

The low computational complexity cost of the kBD and the kBD$_{nt}$, when compared to the ABP and the PTA$_{bl}$, is largely attributed to the iterative convergence rate of the decoder. The information set decoding based stopping criteria ensures the algorithm is able to converge to a codeword with less iterations. This is because the algorithm is only required to decode K bits. This is not the case for the ABP and the PTA$_{bl}$ which have to decode the entire codeword of N bits before the iterative decoding process can break.

Simulations are also run for the high rate (15,11) RS code to compare the computational complexity cost for the bit-level decoders. The results for these simulations can be seen in Fig. 4.17. For these simulations $W_r = 22.48$, $W_c = 5.99$, $W_k = 21.48$, $M = 16$ and $N = 60$.

Similar to the results in Fig. 4.16, the variants of the kBD still perform less operations than the ABP and the PTA$_{bl}$ during decoding of the received vector as

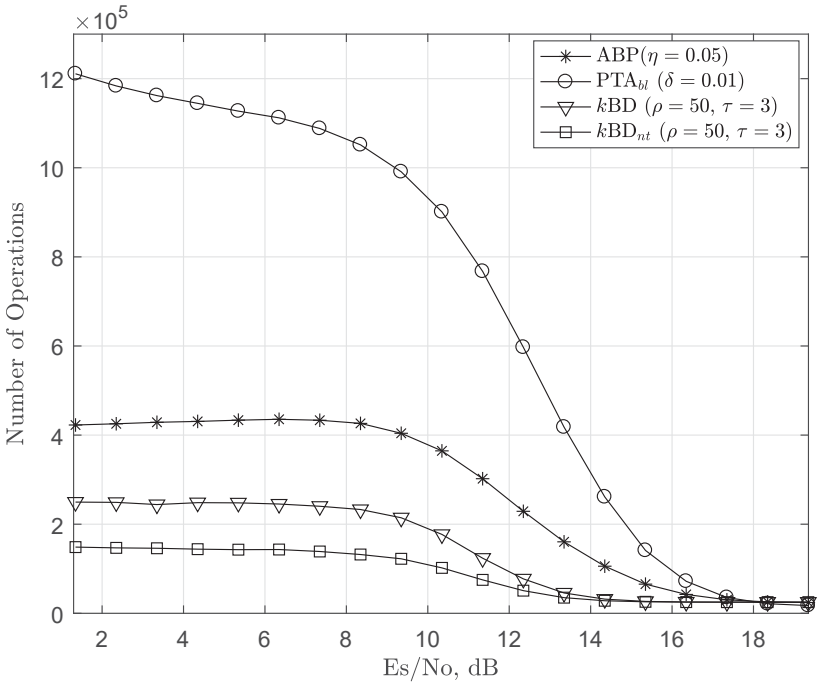


FIGURE 4.17: Complexity comparisons for the bit-level decoders applied to a (15,11) RS code over an AWGN channel using a 16-QAM modulation scheme.

seen in Fig. 4.17. However, there is a larger performance difference in the number of operations run between the kBD and the ABP for the $(15, 11)$ RS code, when compared to the $(15, 7)$ RS code. The main reason the kBD carries out much less operations when compared to the ABP for the $(15, 11)$ RS code than the $(15, 7)$ RS code, is due to the use of a smaller value of τ. The smaller value of τ ensures the decoding condition is met much quicker. This reduces the number of iterations used to correct the bits in the received vector, which in turn reduces the number of operations required during the decoding process. Also, it is important to note that the ABP requires more iterations to decode the received vector for the $(15, 11)$ RS code when compared to the $(15, 7)$ RS code. This is because the high rate code has less rows and a larger parity submatrix in $\mathcal{H}$. This makes the matrix $\mathcal{H}$ more dense which affects the iterative convergence rate of the belief propagation algorithm. The results for the $k\text{BD}_{nt}$ are again quite favourable as in the case of the $(15, 7)$ RS code. This is because the significantly more complex PTA_{bl} decoder only outperforms $k\text{BD}_{nt}$ by less than 0.2dB in terms of error correction. The $k\text{BD}_{nt}$ algorithm also matches the error correction performance of the

ABP, which is also has a higher computational complex cost as seen in Fig. 4.13. As highlighted in section 4.4.2, this justifies the use of the algorithm whenever a performance-complexity tradeoff is required.

4.6 The Decoding Condition Used as an Additional Stopping Criteria for Bit-Level RS Decoders

Work carried out in this section focuses on the implementation of the decoding condition applied to bit-level iterative decoders for RS codes. The decoding condition is incorporated into the decoding algorithms to serve as an additional stopping criteria. This is performed with the aim of improving the iterative decoding convergence rate of the soft-decision bit-level iterative decoders.

The decoding algorithm proposed in this research, is able to obtain the information set of bits by rearranging the columns of the $\mathcal{H}$ matrix and using that to decode the received vector. This approach of rearranging the columns of $\mathcal{H}$ to decoding based on the bit reliability is similar to that of the ABP and the PTA$_{bl}$ decoders. With regards to this, the implementation of the decoding condition works in such a way that the existing decoders only have to find the information set based on bits determined to be the most reliable. The stopping criteria defined by the decoding condition is then used to find the remaining bits that are considered unreliable as shown in (4.19). The decoders ascertain if indeed the information set of reliable bits are correct by working iteratively. At this point (2.16) is applied and the unreliable bits decoded. Work carried out in this section investigates the implementation of the decoding condition at the bit-level for the ABP and the PTA$_{bl}$ decoders.

The main problem in the implementation of the decoding condition is in determining if the reliable bits are indeed correct. The existing way of knowing this is if the reliable bits satisfy a minimum number of syndrome check equations defined by the threshold τ as presented in section 4.3.1. Due to the decoders working at a bit level, all arithmetic operations yield only two possible solutions; either a "0"or a "1". This increases the chances of getting a "0" from the syndrome check

calculations, therefore, giving a false positive for the incorrect bits. To remedy this situation, both the ABP and the PTA$_{bl}$ are periodically run for a specified number of iterations before testing if the decoding condition is met.

For the decoding condition to be used as an additional stopping criteria on both the PTA$_{bl}$ decoder and the ABP algorithm, the pseudo code presented in Algorithm 4 is implemented:

Algorithm 4: Implementation of the Decoding Condition on the PTA$_{bl}$ and ABP Algorithms

---

Input: The received vector, r, from the channel is used to obtain the bit probabilities as shown in section 2.9.3.2

Output: The decoded vector $\widehat{cb}$.

Initialize: Set the value of τ. Set the value of f_{max}. Set $f^{\iota} = f^*$, where f^{ι} represents the number of predefined loops (f^*) the decoder runs before the decoding condition is checked.

Initialize: Setting up the soft-information:

- For the PTA$_{bl}$, the bit reliabilities are used to create β as shown in (2.68)
- For the ABP, the bit probilites are used to obtain the LLR values as presented in (2.69), (2.70), (2.71), (2.72).

repeat

  Repeat

    •*Decoding*:
    Soft-information is fed into decoder and corrected.

  until $f = f^{\iota}$

  •*Decoding condition check*:
  Note the values of S_{Y_K}, where S_{Y_K} represents the number of syndrome checks satisfied by each bit in the information set, $\widehat{cb}_{Y_K}$, defined in (4.16).

  if $S_{Y_K} = \tau$ **then**
    compute (4.19)
  else
    $f^{\iota} = f + f^*$
  end

until $f = f_{max}$

4.6.1 Bit-Level Implementation of the Decoding Condition with the PTA decoder

This section looks at the implementation of the decoding condition as an additional stopping criteria for the PTA_{bl} decoder. This set of simulations are run to test the improvement in the iterative convergence rate when the decoding condition is applied to the PTA_{bl} decoder. This is carried out with the aim of improving the efficiency of the algorithm.

Half Rate RS code Simulations

The first set of simulations are performed on a nearly half rate $(15, 7)$ RS code. For the benchmark simulations, the performance of the PTA_{bl} with a set number of iterations $f^{\iota} = 5$ is compared to PTA_{bl} running with no modifications. The PTA_{bl} with $f^{\iota} = 5$ is run with different values of τ to assist in finding the version of the algorithm that performs best when compared to the original PTA_{bl}. The simulations are run by transmitting the $(15, 7)$ RS code through an AWGN channel using a 16-QAM modulation scheme with Gray mapping. The results for these simulations are presented in Fig. 4.18 and Fig. 4.19.

From Fig. 4.18 it can be seen that the SER performance of the modified version of the PTA_{bl} algorithm, when compared to the original PTA_{bl}, yields a similar result while having $f^{\iota} = 5$ and with $\tau = 1$, $\tau = 3$ and $\tau = 5$. The difference in performance can be seen in the average number of iterations each algorithm requires to decode the received vector, as shown in Fig. 4.19. The orginal version of the PTA_{bl} algorithm requires an average of about 94 iterations at its worst iterative performance to attain the SER performance shown in Fig. 4.18.

The implementation of the decoding condition as an additional stopping condition significantly improves the iterative performance of the PTA_{bl} algorithm as seen Fig. 4.19. This is because the decoder only has to successfully decode the information set based on the set value of τ instead of decoding the entire received

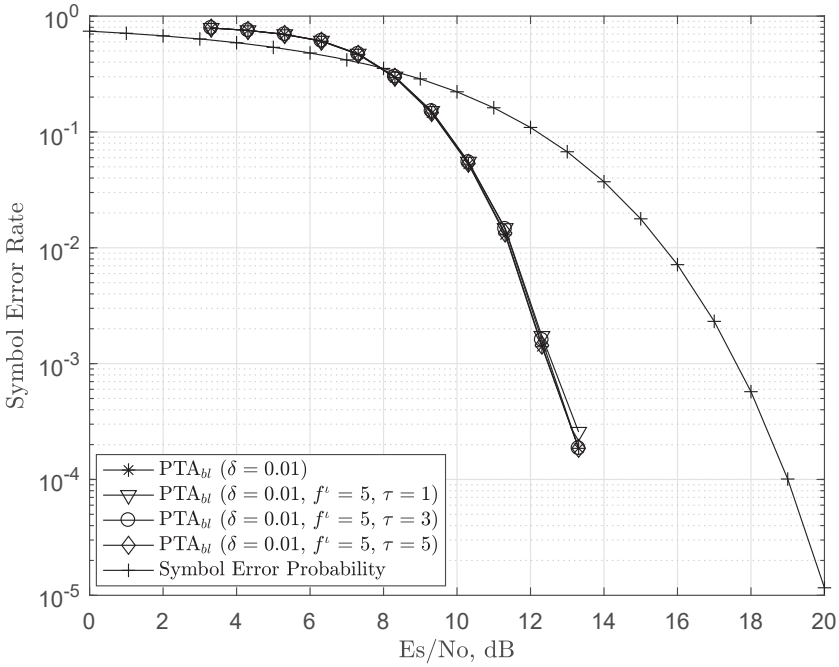


FIGURE 4.18: SER performance of PTA$_{bl}$ vs the PTA$_{bl}$ implemented with the decoding condition defined by different values of τ for a $(15, 7)$ RS code.

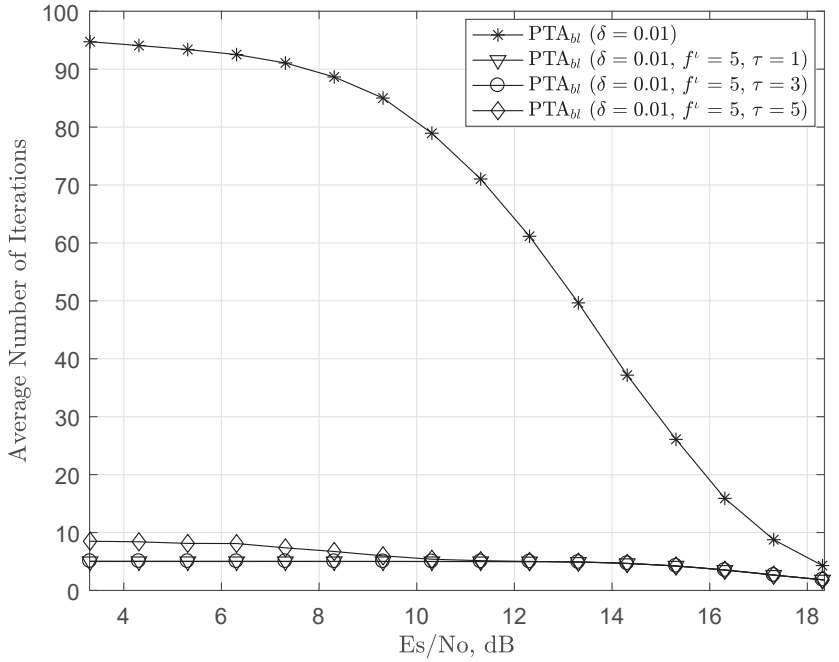


FIGURE 4.19: Iterative performance of PTA$_{bl}$ vs the PTA$_{bl}$ implemented with the decoding condition defined by different values of τ for a (15×7) RS code.

vector. Sorting the bits of the received vector such that the parity submatrix matches the most reliable bits ensures that the most reliable bits, that make up the information set, are responsible for most of the exchange of the soft information as they participate in multiple syndrome check equations in each iteration. This is opposed to the bits that are considered to be unreliable which will only participate in one syndrome check equation in every iteration due to the fact the unreliable bits match the identity submatrix. This makes the decoder require more iterations to correct the unreliable bits when the decoding condition is not applied. The decoding condition eliminates the need to successfully correct the unreliable bits due to the information set decoding approach applied while decoding the received vector. This results in less iterations required in the decoding precess.

The PTA_{bl} with $f^{\iota} = 5$ and $\tau = 3$ is the version of the proposed algorithm implementation that yields the best iterative convergence performance. This is because the algorithm requires an average of about 5 iterations at its worst performance to give the same SER performance as that of the original PTA_{bl} algorithm. The PTA_{bl} algorithms having $f^{\iota} = 5$ with $\tau = 5$ requires an average of about 8 iterations the worst iterative convergence performance in order to match the SER performance of the original PTA_{bl}. While yielding the same iterative convergence performance to that of the PTA_{bl} with $f^{\iota} = 5$ and $\tau = 3$, the PTA_{bl} with $f^{\iota} = 5$ and $\tau = 1$ incurs a slight performance loss when compared to the original PTA_{bl} decoder in terms of SER.

The PTA_{bl} algorithm with $f^{\iota} = 5$ and $\tau = 3$ is able to achieve these gain in terms of a reduced number of iterations due to the small value of τ required for the decoding condition to be met. This performance exhibited by the PTA_{bl} algorithm with $f^{\iota} = 5$ and $\tau = 3$ show that the PTA_{bl} decoder requires an average of about 5 iterations to decode its most reliable information from the received vector. Applying the decoding condition, therefore, saves the algorithm from running an additional average of about 89 iterations in order to decode the remaining unreliable bits from the received vector. This significantly reduces the latency required by the PTA_{bl} to decode the received vector outputted from the channel, which in turn increases the efficiency of the algorithm.

High Rate RS code Simulations

Simulations are run for a high rate $(15, 11)$ RS code. The $(15, 11)$ RS code is run under the same conditions as with the case of the $(15, 7)$ RS code. The set of results for this simulations are presented in Fig. 4.20 and Fig. 4.21.

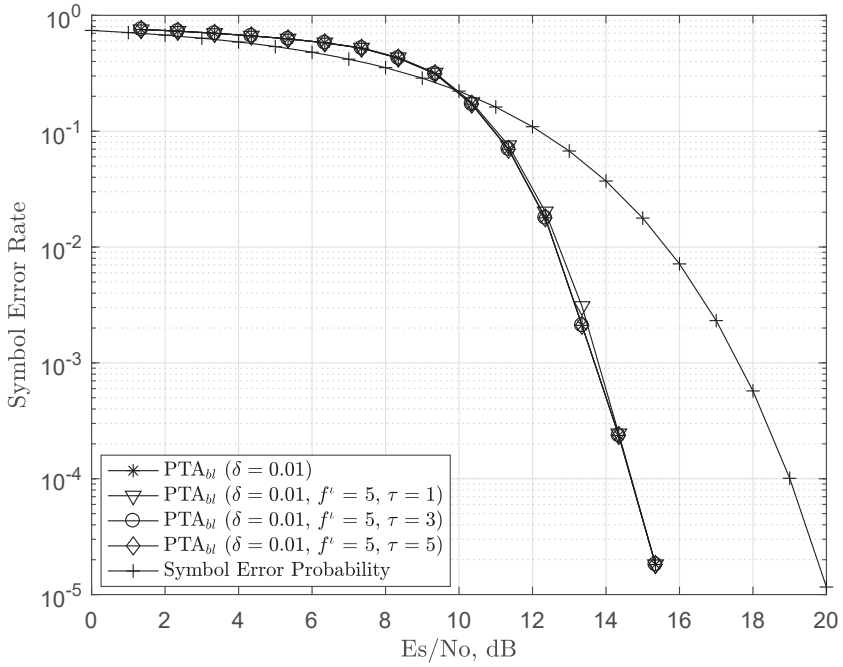


FIGURE 4.20: SER performance of PTA$_{bl}$ vs the PTA$_{bl}$ implemented with the decoding condition defined by different values of τ for a $(15, 11)$ RS code.

From Fig. 4.20, the SER performance for all the decoding algorithms is the same. Once again, the difference in performance of the algorithms is seen in terms of the average number of iterations required to decode the received vector. From Fig. 4.20 and Fig. 4.21 it can be seen that the PTA$_{bl}$ algorithm with $f^{\iota} = 5$ and $\tau = 1$ is the most efficient of the version of the PTA$_{bl}$ algorithm. The algorithm requires an average of just over 5 iterations at its worst performance to decode the received vector, while yielding a performance similar to all variants of the PTA$_{bl}$ including the original version. The iterative performance PTA$_{bl}$ algorithm with $f^{\iota} = 5$ and $\tau = 1$ is significantly better than all the other versions of the

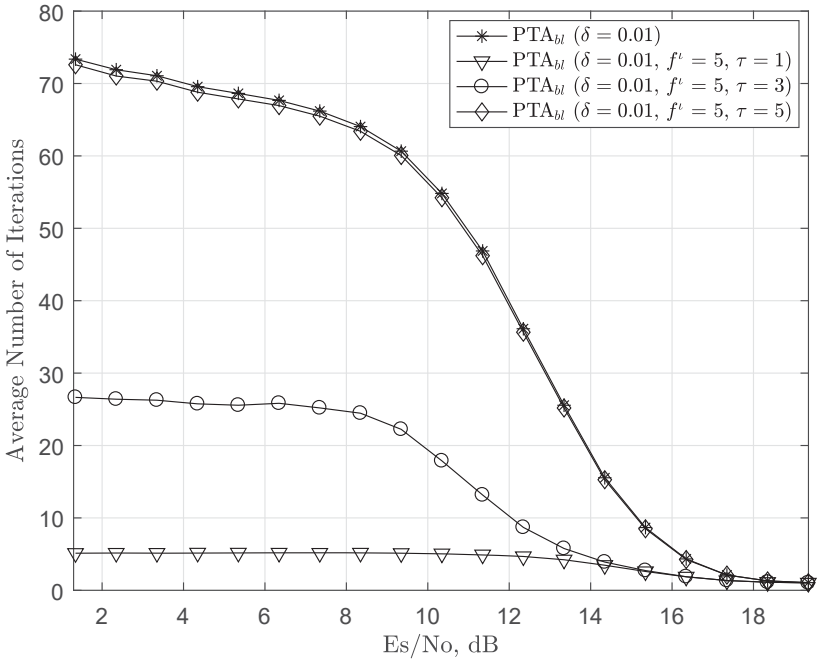


FIGURE 4.21: Iterative performance of PTA$_{bl}$ vs the PTA$_{bl}$ implemented with the decoding condition defined by different values of τ for a $(15, 11)$ RS code.

PTA$_{bl}$, with next best performing algorithm in the simulation being the PTA$_{bl}$ with $f^{\iota} = 5$ and $\tau = 3$ running at an average of just over 19 iterations. The PTA$_{bl}$ with $f^{\iota} = 5$ and $\tau = 5$ requires an average of just over 47 iterations at its worst iterative convergence performance to decode the received vector. The iterative performance of the PTA$_{bl}$ with $f^{\iota} = 5$ and $\tau = 5$ is very close to that of original PTA$_{bl}$, which requires an average of just over 48 iterations at its worst performance to decode the received vector.

For this high rate case, the value of the threshold can be dropped to $\tau = 1$ and still have a high accuracy in terms of error detection due to more bit indexes of the received vector matching the parity submatrix of $\mathcal{H}^{+}$. This means more bits matching the vector indexes of the most reliable information participate in the different parity check equations. Having more bit indexes that are considered to be reliable participating in fewer syndrome check equations due to the higher rate reduces the chances of all the reliable bits meeting the condition of $\tau = 1$, unless these bits are actually correct.

4.6.2 Bit-Level Implementation of the Decoding Condition with the ABP decoder

Similar simulations to test the performance of the decoding condition with the bit-level ABP algorithm are also run. The simulations are run for a nearly half rate $(15, 7)$ RS code and a high rate $(15, 11)$ RS code. Both codes are transmitted through a AWGN channel using a 16-QAM modulation scheme with Gray mapping.

Half Rate RS code Simulations

The first set of simulations for the bit-level ABP decoding algorithm implemented with the decoding condition are run on a nearly half rate $(15, 7)$ RS code. All the versions of the ABP algorithm with the decoding condition have the set threshold $\tau = 3$. However, the value of f^ι is varied. The variations of the ABP algorithms having a decoding condition with a set $\tau = 3$ are run with $f^\iota = 3$, $f^\iota = 5$ and $f^\iota = 7$. All the version of the ABP algorithms are run with the maximum number of iterations set to 20. The results for this simulations can be seen in Fig. 4.22 and Fig. 4.23.

The SER performance of the ABP with the decoding condition that has $f^\iota \geq 5$ and $\tau = 3$ matches the SER performance of the original ABP implementation as seen in Fig. 4.22. It is also noted that the use of the decoding condition as an additional stopping criteria with $f^\iota = 5$ and $\tau = 3$ significantly improves the iterative performance while maintaining the SER performance of the ABP algorithms as seen from Fig. 4.23. The efficiency of the algorithm with $f^\iota = 5$ can be seen as it runs an average of 5 iterations at its worst iterative performance while the original implementation of the ABP algorithm runs for an average of about 14 iterations at its worst.

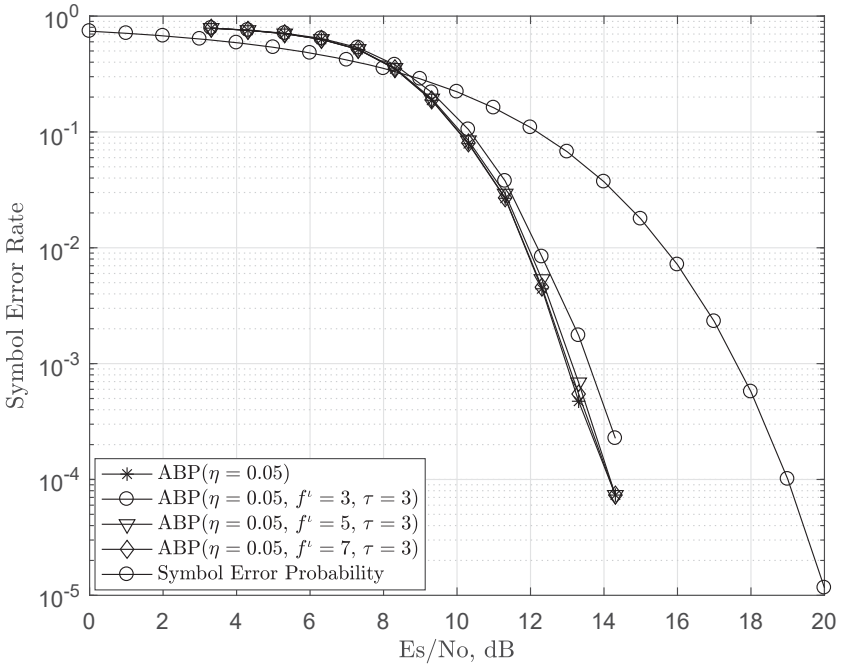


FIGURE 4.22: SER performance of ABP vs the ABP implemented with the decoding condition defined by different values of f^ι for a $(15, 7)$ RS code.

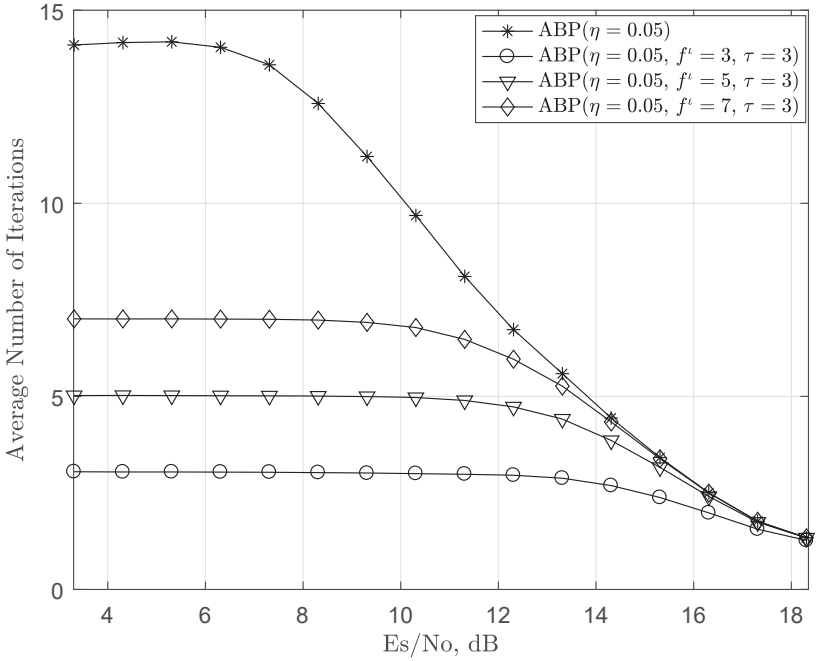


FIGURE 4.23: Iterative performance of ABP vs the ABP implemented with the decoding condition defined by different values of f^ι for a $(15, 7)$ RS code.

High Rate RS code Simulations

Simulations are also run for a high rate $(15, 11)$codes. The code is transmitted under similar conditions through an AWGN channel as the $(15, 7)$ RS code case. Due to H matrix of the higher code have less rows, the value of the $\tau = 3$. Similar to the nearly half rate case, the variations of the ABP algorithm with decoding condition are run with $f^{\iota} = 3$, $f^{\iota} = 5$ and $f^{\iota} = 7$.

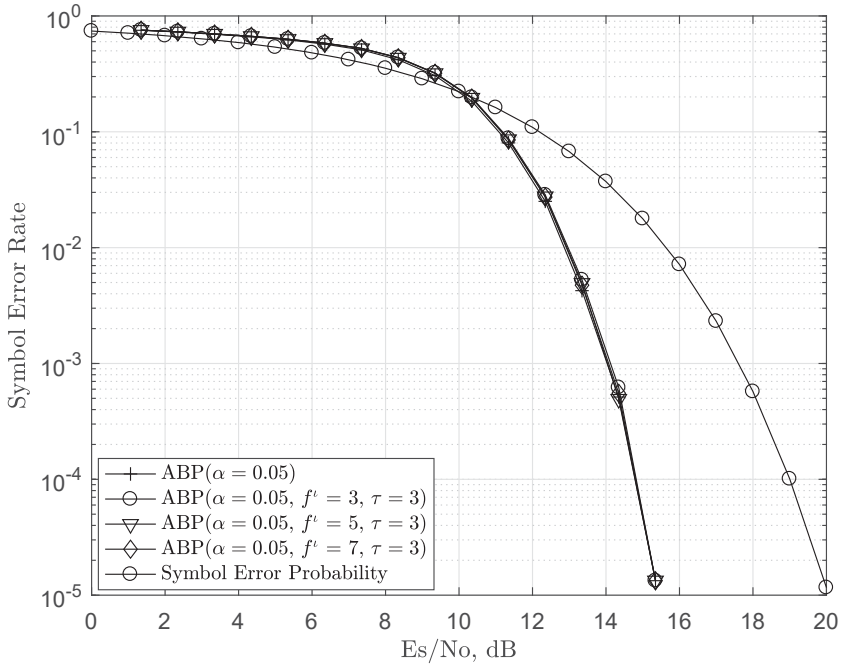


FIGURE 4.24: SER performance of ABP vs the ABP implemented with the decoding condition defined by different values of f^{ι} for a $(15, 11)$ RS code.

All the 3 variations of the ABP algorithm with decoding condition match the performance of the ABP algorithm. This is important to note because it means the the ABP with $f^{\iota} = 3$ which averages the least number of iterations when decoding yields a similar performance to that of the original implementation of the ABP algorithm. This provides an improved efficiency of the algorithm as the ABP algorithm with $f^{\iota} = 3$ runs for an average of about 11 iterations at its worst iterative convergence performance when compared to the ABP without the decoding condition implementation that runs for about 18 iterations at its worst iterative convergence performance.

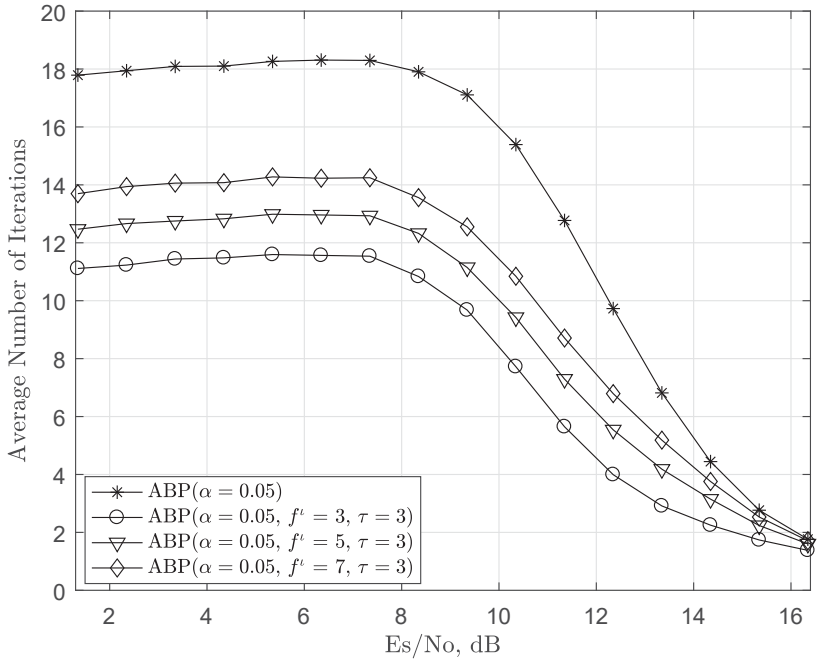


FIGURE 4.25: Iterative performance of ABP vs the ABP implemented with the decoding condition defined by different values of f^{ι} for a $(15, 11)$ RS code.

4.7 Conclusion

In this chapter an iterative soft-input soft-output bit-level information set decoder is proposed for RS codes. The algorithm has two variants which both achieve gains in terms of the algorithms ability to converge to the decoded codeword when compared to the high performing bit-level RS decoders, while yielding a good SER performances. The ability of the proposed decoder to require less iterations during the decoding process largely contributes to the algorithm a relatively lower computational complexity. For the $(15, 7)$ RS code transmitted using a 16-QAM modulation scheme with Gray mapping, the kBD outperforms the ABP with $\eta = 0.05$ by a gain of about 0.6dB and yields a gain of about 0.25dB when compared to the PTA$_{bl}$ with $\delta = 0.01$. The kBD$_{nt}$ is outperformed by the PTA$_{bl}$ with $\delta = 0.01$ by about 0.35dB and yields a similar SER performance to that of the ABP with $\eta = 0.05$. However, the PTA$_{bl}$ with $\delta = 0.01$ requires significantly more iterations to attain this slight gain over the kBD$_{nt}$. Due to the iterative performance of the PTA$_{bl}$ with $\delta = 0.01$, the complexity of the algorithm is much higher than the other

iterative soft-decision algorithms used in the simulations. This justifes the use of the $k\text{BD}_{nt}$ decoding algorithm over the PTA_{bl} decoder whenever a performance-complexity tradeoff is required.

For the $(15, 11)$ RS code, similar results are yielded in terms of SER performance by the kBD algorithm and the PTA_{bl} decoder. The performance of the kBD once again comes at a significantly lower complexity than the PTA_{bl} decoder. This is largely due to the numerous iterations required by the PTA_{bl} decoder. In spite of the of the $k\text{BD}_{nt}$ yielding a similar performance to the ABP algorithm, it is slightly outperformed by about 0.2dB when compared to both the kBD and the PTA_{bl} decoders. However, this error correction performance of the $k\text{BD}_{nt}$ decoder is considered favourable predominately because it is the least computationally complex algorithm of the decoders used in the simulations.

The decoding condition as an additional stopping criteria for the ABP, and the PTA_{bl} decoder is seen to be an an efficient decoding implementation by maintaining the high performance of these algorithms while reducing the number of iterations required to decode the received vector. This justifies the use decoding condition as an additional stopping criteria whenever a reduced complexity implementation of the bit-level decoders is required.

CHAPTER 5

The Symbol-Level Implementation of the Proposed-Decoding Algorithm for Reed-Solomon codes

5.1 Introduction

This chapter proposes the implementation of an information set decoding algorithm at a symbol-level. Implementation at a symbol-level was previously inhibited by the dense structure parity check submatrix in H. The dense structure in the parity check submatrix guarantees all the symbols in the information set participated in every syndrome check equation. This makes it hard for the proposed decoder to detect and correct errors in the information set required to decode the received vector. This is because if there is even one error in the information set then most, if not all, syndrome check equations will not be satisfied. The research carried out in this chapter proposes the integration of the H matrix extensions in the implementation of the decoding algorithm with aim of solving this problem. The use of the H matrix extensions enables the decoder to work with multiple parity check matrices, each with different groupings of column indices. This enables the decoder to obtain different information sets from each H matrix extension. The decoder is then able to detect the correct information set based on the number of satisfied syndrome check equations from the each syndrome vector obtained from the different H matrix extensions. That is, the number of satisfied syndrome check equations in the different syndrome vectors acts as a measure of confidence in the selection of the correct information set. Similar to Chapter 4, preliminary

tests to assist in design and implementation of the proposed symbol-level information set decoder are run on the PTA.

The rest of the chapter is structure as follows; Section 5.2 presents the preliminary design implementation and analysis of the proposed parity check extensions applied the PTA decoder. The symbol-level implementation of the proposed information set decoder is explained in Section 5.3. The stopping criteria for the decoder is described in 5.4. The performance of the symbol-level implementation of the information set decoder, in terms of SER and iterative convergence rate, is investigated in Section 5.5. A complexity analysis of the symbol-level information set decoder is carried out in Section 5.6. Section 5.7 proposes the inclusion of the decoding condition as an additional stopping criteria for the PTA decoder. Finally, a conclusion to the chapter can be found in Section 5.8.

5.2 Preliminary Tests

This section presents the preliminary results of a proposed decoding technique used in symbol-level decoding. The proposed technique is referred to as the parity check matrix extension. This technique is implemented by the addition of parity check equations with the aim of improving the iterative performance of the proposed decoding algorithm. The additional parity check equations are low weight equations that match the minimum distance of the RS code. Adding low weight equations adds to the sparsity of the symbol-level H matrix. This allows for a more efficient exchange of soft information during the decoding process when compared to using the original H matrix. Similar to Chapter 4, the decoding technique is first applied to the PTA decoder with aim of performing a preliminary analysis.

The parity check equations used to extend the H matrix are implemented through the addition of more parity check matrices that exist in the same dual space of the transmitted RS code. The systematic structure of the additional H matrices is also utilised to ensure the weight of the parity check equations, that make up the extension matrices, meet the minimum distance so as to make the full extension of the H matrix as sparse as possible at the symbol-level. The H matrix extension

decoding technique is proposed with the main aim of creating a sparse matrix that can assist in the detection and correction of errors in symbols with indices matching the parity submatrix from the different H matrices. With regard to this, the performance of the symbol-level PTA decoder with the H matrix extensions is paramount to the research because it provides the preliminary design and performance of the implementation of the decoding algorithm proposed in this research at the symbol-level.

5.2.1 The PTA Extended Algorithim

To reduce the number of iterations required to decode the received vector by the PTA algorithm, more corrections have to be carried out on the reliable information during each iteration. For such an implementation to work, the parity check matrix is extended to accommodate more corrections and efficient of the soft information. This research extends the H matrix in such a way that the correction step involves satisfying the syndrome check equations using more than one H matrix. Work on the correction step is performed through an extension of the H matrix by using 2 parity check matrices, 3 parity check matrices and 4 parity check matrices. A maximum of 4 parity check matrices is selected so as to prevent complexity issues that may arise from using a very large H matrix in the decoding process.
To establish notation, The new matrix obtained from the original transformation is referred to as h_1. The index vectors $\mathcal{U}$ and $\mathcal{K}$ used in the transformation process, obtained from sorting the symbol reliabilities in Section 2.6.2, are referred to as $\mathcal{U}_1$ and $\mathcal{K}_1$. The extension of the H matrix used to enable more corrections for the PTA decoder is implemented as follows:

1. *2nd Parity Check Matrices, h_2:* To obtain the second parity check matrix, h_2, different indexing vectors are used. The new indexing vectors are obtained by swapping the last index in $\mathcal{U}_1$ and with the first index in $\mathcal{K}_1$, to form the

vectors $\mathcal{U}_2$ and $\mathcal{K}_2$ which can be shown as

$$\begin{aligned} \mathcal{U}_2 &= [Y_1, ..., Y_{(n-k)-1}, Y_{(n-k)+1}], \\ \mathcal{K}_2 &= [Y_{(n-k)}, Y_{(n-k)+2}, ..., Y_k]. \end{aligned} \tag{5.1}$$

The original parity check matrix, H, is then transformed based on these new vectors to form the matrix h_2. Therefore, the matrix h_2 is a function of the matrix H and the vectors $\mathcal{U}_2$ and $\mathcal{K}_2$ and can be expressed as

$$h_2 = \mathcal{F}(H, ([\mathcal{U}_2\ \mathcal{K}_2])). \tag{5.2}$$

2. *3rd Parity Check Matrices, h_3:* The implementation is similar to that of the h_2 with new index vectors $\mathcal{U}_3$ and $\mathcal{K}_3$ obtained by swapping the second last index in $\mathcal{U}_1$ with the first index in $\mathcal{K}_1$ and is noted as

$$\begin{aligned} \mathcal{U}_3 &= [Y_1, Y_2, ..., Y_{(n-k)-2}, Y_{(n-k)}, Y_{(n-k)+1}], \\ \mathcal{K}_3 &= [Y_{(n-k)-1}, Y_{(n-k)+2}, Y_{(n-k)+3}, ..., Y_k]. \end{aligned} \tag{5.3}$$

The matrix h_3 is also formed as a function of the matrix H and the two index vectors $\mathcal{U}_3$ and $\mathcal{K}_3$ and can be displayed as

$$h_3 = \mathcal{F}(H, [\mathcal{U}_3\ \mathcal{K}_3])). \tag{5.4}$$

3. *4th Parity Check Matrices, h_4:* The matrix h_4 is formed using the new index vector $\mathcal{U}_4$ and $\mathcal{K}_4$, which are obtained by swapping the third last index in $\mathcal{U}_1$ and the first index in $\mathcal{K}_1$ and is shown as

$$\begin{aligned} \mathcal{U}_4 &= [Y_1, Y_2, ..., Y_{(n-k)-2}, Y_{(n-k)}, Y_{(n-k)+1}], \\ \mathcal{K}_4 &= [Y_{(n-k)-3}, Y_{(n-k)+2}, Y_{(n-k)+3}, ..., Y_k]. \end{aligned} \tag{5.5}$$

The matrix h_4 is a function of the H matrix and the index vectors is expressed as

$$h_4 = \mathcal{F}(H, ([\mathcal{U}_4\ \mathcal{K}_4])). \tag{5.6}$$

Depending on the number of H matrices required by the PTA extension in the correction step of the algorithm, syndrome check equations are computed on a matrix of the form displayed in (5.16)

$$H_{\varepsilon} = \begin{bmatrix} h_1 \\ \vdots \\ h_{\varepsilon} \end{bmatrix}, \tag{5.7}$$

where ε represents the number of H matrices used in the extension.

5.2.2 Simulation and Results

5.2.2.1 Half rate simulations

The 3 types of extensions of the PTA were run on a $(15, 7)$ RS code. The code was transmitted through an AWGN channel using a 16-QAM modulation scheme with Gray mapping. PTAe_2, PTAe_3 and PTAe_4 are used to represent the PTA extension that use 2, 3 and 4 H matrices to decode the received vector respectively. The value of the correction factor, δ, used is 0.001. This is because the PTA performs at its best with this condition [8]. The performance of the 3 types of extension were compared against each other to determine the best performing algorithm in terms of SER and average number of iterations. These results can be seen in Fig. 5.1 and Fig. 5.2.

From Fig. 5.1, it can be seen that PTAe_3 and PTAe_2 slightly outperforms the PTAe_4 implementations at an SER value of 10^{-3} with a gain of about 0.4dB and 0.17dB respectively. The iterative performance of the PTAe_3 is significantly better than that of the PTAe_2 as seen in Fig 5.2. The PTAe_4 runs for the least average number of iterations, however it is outperformed in terms of SER by the 2 other implementations of the PTA extensions. The result shown in Fig. 5.1 and Fig. 5.2 is used as justification for the use of the PTAe_3 in the benchmark simulations against the PTA decoder. The performance is also tested in terms of SER and average number of iterations. These results can be seen in Fig. 5.3 and Fig. 5.4

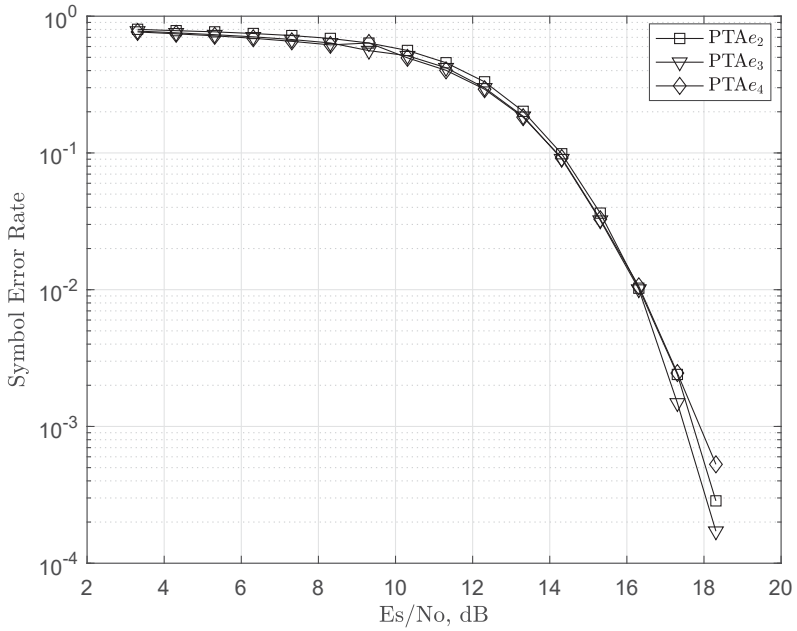


FIGURE 5.1: SER Performance comparison of the PTA extended algorithm with 2, 3 and 4 H matrices used to decode the received vector for a (15,7) RS codes.

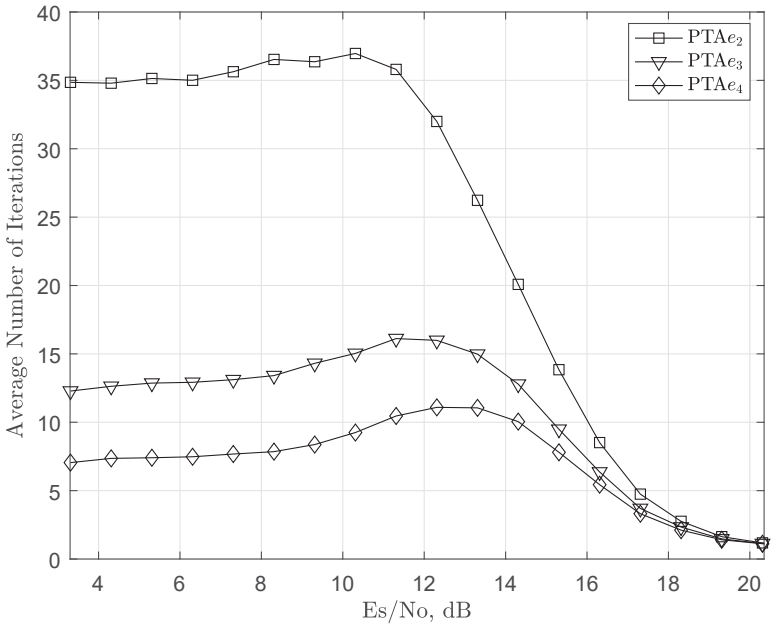


FIGURE 5.2: Iterative performance comparison of the PTA extended algorithm with 2, 3 and 4 H matrices used to decode the received vector for a (15,7) RS codes.

respectively.

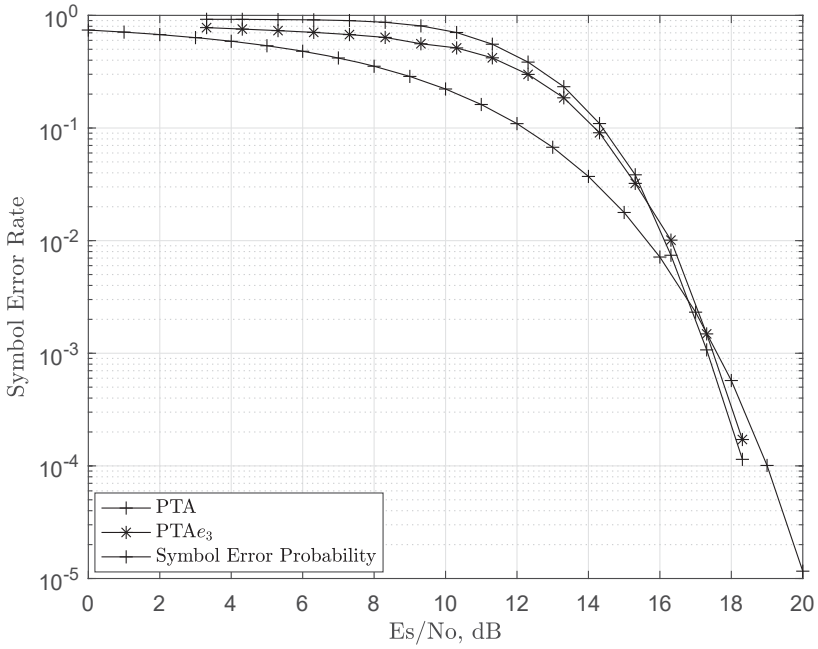


FIGURE 5.3: Performance comparison of the PTA and the PTAe_3 in terms of SER for a $(15, 7)$ RS code.

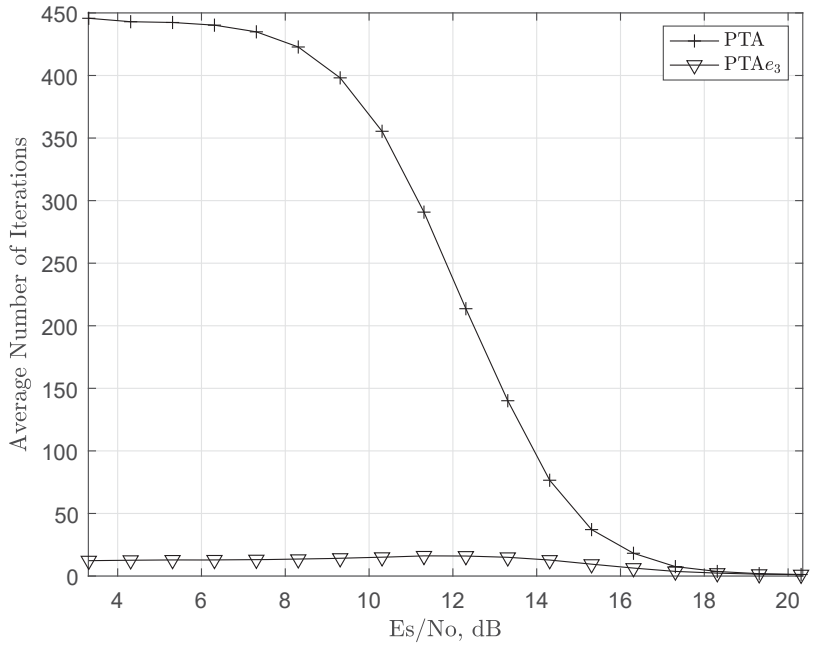


FIGURE 5.4: Performance comparison of the PTA and the PTAe_3 in terms of average number of iterations for a $(15, 7)$ RS code.

From Fig. 5.3 it can be seen the PTAe_3 gives a similar SER performance when compared to the PTA algorithm, with a slight gain of 0.15dB yielded by the PTA decoder. In terms of average number of iterations, a more significant gain is experienced as seen in Fig. 5.4. The PTAe_3 requires an average of about 13 iterations, at it worst iterative convergence performance, to decode the received vector compared to the PTA algorithm which runs for slightly less than 450 iterations at its worst iterative convergence performance. This is a significant improvement in the efficiency of the decoder.

5.2.2.2 High rate simulations

An additional simulation is performed for a high rate $(15, 11)$ RS code. This is carried out in order to test if the efficiency of this implementation holds up at higher code rates. The code is transmitted using similar parameters as those used for the $(15, 7)$ RS code. The results for this simulation with regards to symbol error performance are presented in Fig. 5.5 and the results for average iteration count can be seen in Fig. 5.6.

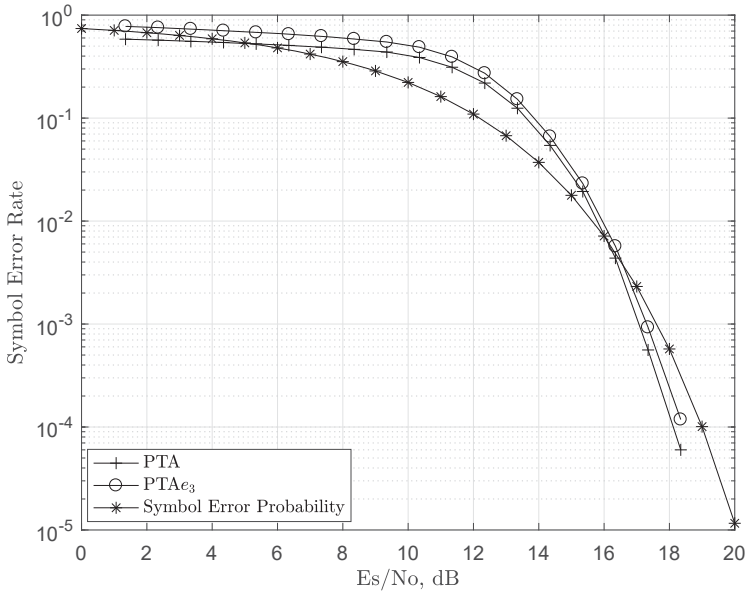


FIGURE 5.5: Performance comparison of the PTA and the PTAe_3 in terms of SER for a $(15, 11)$ RS code.

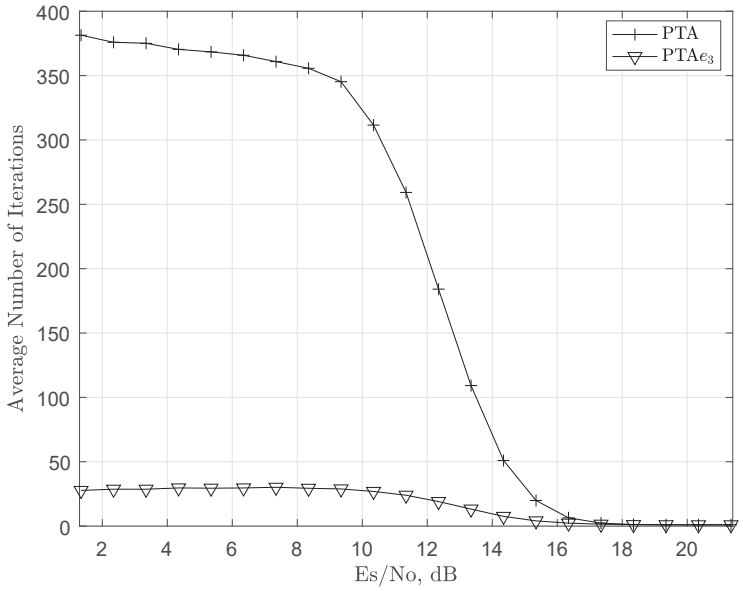


FIGURE 5.6: Performance comparison of the PTA and the PTAe_3 in terms of average number of iterations for a $(15, 11)$ RS code.

For this high rate code, it is observed that the gain for the PTAe_3 in terms of SER yields a similar performance to that of the original PTA implementation, with a slight gain of 0.25dB achieved by the PTA decoder. Similar to the half rate case, the significant reduction exhibited in the average number of iterations required to decode the received vector is maintained by the PTAe_3 for high rate codes. Therefore, it can be inferred that the efficiency of the PTAe_3 algorithm is maintained even with an increased code rate.

5.2.3 Conclusions From the Preliminary Results

The implementation of the extension of the parity check matrix improves the iterative performance of the symbol-level PTA decoder. The addition of the sparse parity check vectors assists in the correction of the wrong symbols through more efficient message passing at the symbol-level. The parity check extension decoding technique enables the PTA algorithm to work more efficiently by yielding a similar SER performance and while significantly reducing the average number of iterations.

The results obtained from the H matrix extensions with the symbol-level PTA provide the basis of the design and implementation for the proposed decoder. The low weight parity check equations used in the H matrix extensions are used to assist the proposed decoder implementation at the symbol-level due to the improved message passing of the soft information and the iterative performance.

5.3 Implimentation of the Proposed Iterative Decoding Technique at a Symbol-Level

To better highlight the limitations presented from a direct implementation of the proposed decoder in Chapter 4 at a symbol-level, a basic explanation of how the decoder works is now presented. Consider an (n, k) RS code $\mathcal{C}$, in the field GF(2^m), defined by a systematic H matrix with the form.

$$H = [I \;\; Q] \tag{5.8}$$

Any codeword c of $\mathcal{C}$ can be divided into two parts as shown in (5.9)

$$\begin{aligned} c_{\mathcal{U}} &= [\ \widehat{c}_1\ ,\ \widehat{c}_2\ , ...,\ \widehat{c}_{(n-k)}\ ], \\ c_{\mathcal{K}} &= [\ \widehat{c}_{(n-k)+1}\ , ...,\ \widehat{c}_n\ ]. \end{aligned} \tag{5.9}$$

The syndrome calculation presented in (2.44) and (2.45) can be rewritten as

$$S = (c_{\mathcal{U}} \times I^{\top}) + (c_{\mathcal{K}} \times Q^{\top}), \tag{5.10}$$

Submatrix I is an identity matrix, therefore (5.10) can be rewritten as

$$S = c_{\mathcal{U}} + (c_{\mathcal{K}} \times Q^{\top}). \tag{5.11}$$

Assuming c to be a valid codeword, then the syndrome vector $S = 0$. This means (2.16) can be used to represent (5.11) as

$$c_{\mathcal{U}} = (c_{\mathcal{K}} \times Q^{\top}) \tag{5.12}$$

In a bid to apply (5.12) to the proposed decoder, all the k symbols in $c_{\mathcal{K}}$ have to be correct. This presents a challenge in the proposed decoding approach. This is because, for RS codes, the parity submatrix Q is completely dense meaning all the symbols in $c_{\mathcal{K}}$ will participate in each syndrome check equation for each row of H. This makes it difficult to detect and correct the symbols that are wrong in $c_{\mathcal{K}}$. For the bit-level implementation, due to the sparse nature of the parity submatrix, different parity check equations containing different bit indexes of the received vector enables an effective application of the proposed decoding algorithm. A similar approach is required at the symbol-level to enable the implementation of the proposed decoding technique. To allow for different symbol indexes that match the parity submatrix at the symbol-level, more than one information set is required for the decoding of the received vector. To facilitate the implementation of more than one information set, parity check equations containing different symbol indexes need to be used in the decoding process. The additional parity check equations, having a weight matching the minimum distance of the code, are added to H using the parity check matrix extension approach presented in Section 5.2.1. In order to understand how the proposed decoder works, consider the same (n, k) RS codeword c in the field GF(2^m) transmitted through a noisy channel to obtain the received vector r. The vector r is then used to create the reliability matrix β using the distance metric method computed using (2.73) and (2.74) [46]. It is important to note that each row in β represents an element in the field as illustrated in [8]. The β matrix is then fed into the decoder. The steps involved in decoding for the proposed algorithm can be summarised as follows:

1. *Determining the symbol reliabilities*: This stage involves storing the maximum reliabilities in each column of β in the vector A. This is expressed as

$$A = [A_1, A_2, ..., A_j] \tag{5.13}$$

where $A_j = \text{argmax}(\beta_j)$, $0 \leq j \leq n$ and β_j represents each column of the matrix β. The reliabilities in A are then sorted in ascending order with indexes of the k highest reliabilities being stored in the vector $\mathcal{K}$. The

remaining $(n-k)$ indexes are considered to be unreliable and are stored in the vector $\mathcal{U}$. These vectors are shown in (5.14)

$$\begin{aligned} \mathcal{U} &= [Y_1, Y_2, ..., Y_{(n-k)}], \\ \mathcal{K} &= [Y_{(n-k)+1}, .., Y_n]. \end{aligned} \tag{5.14}$$

where Y_j represents the symbol indexes.

2. *The H matrix extension*: The columns of the matrix H are then rearranged using row reduction techniques to ensure that columns with the indexes $\mathcal{K}$ match the parity submatrix. A computationally efficient way of doing this can be seen in [127] and [130]. The matrix obtained from the first row reduction is denoted as h_1. The matrix h_1 represents the first matrix extension. Other H matrix extensions are obtained by using the general formula shown in (5.15) to rearrange the columns of the corresponding matrix extensions based on swapping the positions of the symbol index from their respective vectors as presented in Section 5.2.1.

$$\begin{aligned} \mathcal{U}_x &= [Y_1, Y_2, ..., Y_{((n-k)-(x-1))}, ..., Y_{(n-k)+1}], \\ \mathcal{K}_x &= [Y_{((n-k)-(x-2))}, Y_{(n-k)+2}, ..., Y_n]. \end{aligned} \tag{5.15}$$

where $1 \leq x \leq \varepsilon$ and ε represents the predefined number of parity check matrix extensions. It is important to note when $x = 1$ the index vectors $\mathcal{U}_1$ and $\mathcal{K}_1$ are obtained using (5.14). All other index vectors $\mathcal{U}_x$ and $\mathcal{K}_x$, where $x \geq 2$, used for the additional extensions are obtained using (5.15). The matrix extensions obtained from different index vectors are denoted as h_x. The complete extended parity check matrix is denoted as H_ε and is expressed as

$$H_\varepsilon = \begin{bmatrix} h_1 \\ \vdots \\ h_\varepsilon \end{bmatrix}. \tag{5.16}$$

3. *The syndrome check*: The hard-decision vector $\widehat{c}$ is obtained from β and is used to get the syndrome, S, using the same approach shown in [8]. Each

extension in H_ε has a syndrome vector denoted as s_x. The syndrome equation for each of the vectors s_x is calculated by getting the product between $\widehat{c}$ and H_ε as expressed in (5.17)

$$\begin{bmatrix} s_1 \\ \vdots \\ s_\varepsilon \end{bmatrix} = \widehat{c} \times \begin{bmatrix} h_1^\top \\ \vdots \\ h_\varepsilon^\top \end{bmatrix}. \tag{5.17}$$

4. *Symbol voting ratios*: Voting for each symbol is performed during the syndrome calculation. All symbols, except the symbol being investigated, participate in the vote. That is,the syndrome equation in (5.17) can also be expressed as

$$s_{x_i} = \widehat{c} \cdot h_{x_i}, \tag{5.18}$$

where $1 \leq i \leq (n-k)$ and is used to represent the row number in each parity check matrix extension. Due to the rearranged systematic structure of the parity check matrix extensions of H_ε, only one symbol index of $\mathcal{U}_x$ participates in each row check. Therefore, (5.18) can be reduced to the form shown in (5.19)

$$s_{x_i} = \widehat{c}_{t_i} \cdot h_{x_{(i,t_i)}}, \tag{5.19}$$

where t_i represents the indices of the participating symbols of $\widehat{c}$ in the i^{th} syndrome check equation of s_x and has a length of $k+1$. The indices denoted by the vector t_i can be written as

$$t_i = \{n : h_{x_{(i,n)}} \neq 0\}. \tag{5.20}$$

To apply the vote, it is assumed all the symbols except the one being voted for are correct. Hence, its presumed that $s_{x_i} = 0$, which means (2.16) can be used to rewrite (5.19) as form of the extrinsic information. This yields the expression given as

$$\widehat{c}_y \cdot h_{x_{(i,y)}} = \widehat{c}_{t'} \cdot h_{x_{(i,t')}}, \tag{5.21}$$

where $y \in t_i$ represents the symbol index that gets the vote. The subvector t' represents all the indexes in the vector t_i without the bit index y. To get the symbol vote for $\widehat{c}_y$, (5.21) is rewritten as

$$\widehat{c}_y = \frac{\widehat{c}_{t'} \cdot h_{x_{(i,t')}}}{h_{x_{(i,y)}}}. \tag{5.22}$$

The actual value of $\widehat{c}_y$ obtained from (5.22) is an element in the field GF(2^m) and is counted as a single vote for the respective element. This is repeated for all participating symbols in all rows of H_ε. The votes are tallied for each element in the field and stored in the matrix V as shown

$$V = \begin{bmatrix} \alpha^{-\infty}, & \alpha^0, & \alpha^1 & ... & \alpha^{2^m-2} \\ V_{\alpha^{-\infty}}, & V_{\alpha^0}, & V_{\alpha^1} & ... & V_{\alpha^{2^m-2}} \end{bmatrix}. \tag{5.23}$$

The first row of V represents each element in the field GF(2^m). The second row of the matrix V has the values V_{α^z}, which are used to represent the total number of votes each of the corresponding elements get. The vote ratios are then calculated as seen in (5.24), and denoted using ϑ.

$$\vartheta = \begin{bmatrix} \frac{V_{\alpha^{-\infty}}}{V_T}, & \frac{V_{\alpha^0}}{V_T}, & \frac{V_{\alpha^1}}{V_T}, & ... & \frac{V_{\alpha^{2^m-2}}}{V_T} \end{bmatrix}, \tag{5.24}$$

where V_T represents the total number of votes cast and is expressed as

$$V_T = V_{\alpha^{-\infty}} + V_{\alpha^0} + ... + V_{\alpha^{2^m-2}}. \tag{5.25}$$

5. *Obtaining the confidence rating*: The algorithm uses the confidence ratios as a measure of confidence that a symbol is correct based on the number of votes it gets. The confidence rating of each symbol is stored in the vector Λ and is calculated as

$$\Lambda = \frac{\vartheta}{\rho}, \tag{5.26}$$

where ρ represents a constant predefined value input during the initialisation of the algorithm. The confidence ratios for each symbol are represented as

$$\Lambda = \left[\begin{array}{cccc} \Lambda_{\alpha^{-\infty}}, & \Lambda_{\alpha^0}, & \ldots, & \Lambda_{\alpha^{(2^m-2)}} \end{array} \right]. \tag{5.27}$$

6. *Updating* β: The confidence ratios are used to update the reliabilities in the matrix β. The update is based on the vote each symbol index gets in the voting stage. The update is performed by adding the confidence ratio of the symbol that wins the vote to its corresponding reliability in β. This update is performed on each of the participating symbols for every row of H_ε as shown.

$$\beta^{(f+1)}_{(\widehat{c}_y,j)} = \beta^{(f)}_{(\widehat{c}_y,j)} + \Lambda_{\widehat{c}_y}, \tag{5.28}$$

where f represents the current iteration number and $\widehat{c}_y$ is used to represent the symbol obtained from (5.22) as indexed in the matrix β. The matrix β is then scaled to ensure each column adds up to 1.

5.4 The Stopping Criteria for the Algorithm

The proposed algorithm repeats the steps in Section 5.3 until the stopping criteria is met. Similar to the bit-level implementation, there are 2 stopping conditions defined for the proposed decoder. The first is when all the elements in the syndrome vector $S = 0$. The second stopping criteria is when the decoding condition is met. The decoding condition is defined as the moment when the algorithm determines that the k symbols matching the parity submatrix of any of the matrices h_x are correct. Once the proposed symbol-level decoding algorithm is able to deduce the information set of k symbols are correct, the entire codeword can be decoded based on these symbols using (5.12).

The proposed decoding algorithm is able to ascertain if the k information set symbols are correct by setting a threshold, τ, on the total number of syndrome check equations satisfied by a single parity check matrix extension h_x. This means, for example, if $\tau = 4$ then only 4 elements in a single syndrome vector s_x should be

equal to 0 for the decoding condition to be met. The remaining $(n-k)$ symbols matching the identity submatrix are then decoded using (5.12) based on the parity check matrix extension h_x that matches the syndrome vector s_x.

Meeting the decoding condition enables the proposed decoder to run less iterations. This is because the algorithm only has to ensure that the k information set symbols are correct compared to decoding the entire received codeword of length n.

It is important to note that the value of τ is once again used as a measure of confidence that the k symbols matching the parity submatrix are correct. Lower values of τ ensure that decoding condition is met early, therefore reducing the number of iterations required to decode the received vector. However, this reduces the error correction performance of the algorithm because there is a higher chance that the information set used to decode the remaining $(n-k)$ symbols may contain errors.

5.5 Results and Analysis

5.5.1 Analysis of the Proposed Symbol-Level Decoding Algorithm

In this section, simulations are performed on the proposed decoder to determine its optimum conditions based on the different number of parity check matrix extensions. The results for the simulations are quantified in terms of Symbol Error Rate (SER) and average number of iterations required by the algorithms to decode the received vector. For the purposes of notation the proposed decoder will be referred to as the k-Symbol Decoder and denoted as kSD. Similar to Section 5.3, the number of parity check matrix extensions used with the kSD algorithm are denoted as ε.

The first set of tests involve running simulations for a nearly half rate $(15, 7)$ RS code. The 3 variations used for the kSD run in the simulations for this code include the decoder running with $\varepsilon = 2$, $\varepsilon = 3$ and $\varepsilon = 4$. All the variations of the kSD are run with $\rho = 50$ and $\tau = 4$. The code is transmitted through an AWGN

channel using a 16-QAM modulation scheme with Gray mapping. The results for these simulations are presented in Fig. 5.7 and Fig. 5.8.

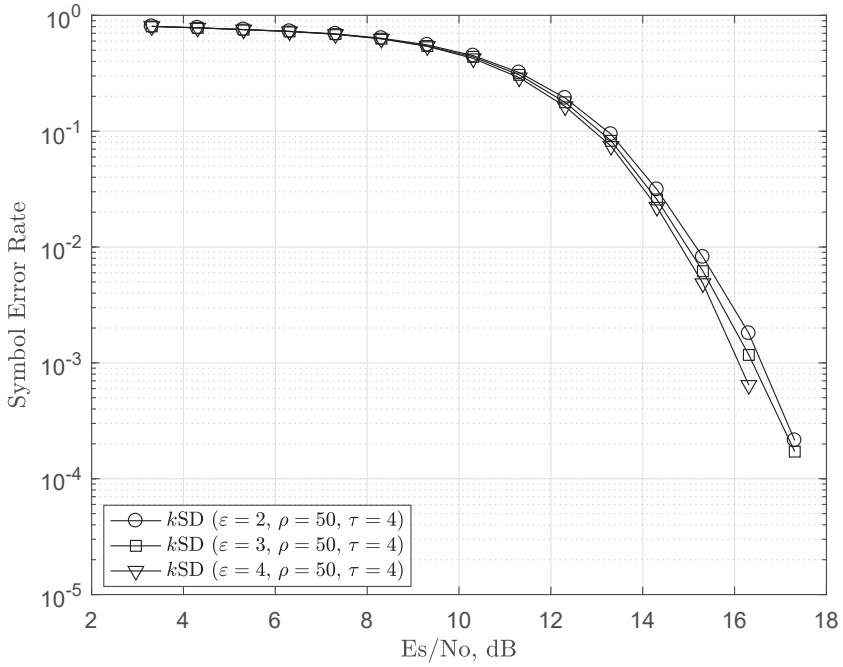


FIGURE 5.7: SER performance comparison of the kSD for a $(15, 7)$ RS code over AWGN channel for different values of ε.

It can be seen from Fig. 5.7 that the performance of the algorithm improves with the number of extensions used. The kSD with $\varepsilon = 4$ is the best performing decoder out of the 3 algorithms. At an SER value of 10^{-3}, the kSD with $\varepsilon = 4$ yields gains of about 0.3dB and 0.5dB when compared to the kSD with $\varepsilon = 3$ and $\varepsilon = 2$ respectively. The kSD algorithm with $\varepsilon = 4$ also runs for the least average number of iterations when compared to the kSD with $\varepsilon = 3$ and $\varepsilon = 2$ as seen from Fig. 5.8. The kSD with $\varepsilon = 4$ run for an average number of just over 15 iterations at its worst performance. The kSD with $\varepsilon = 3$ runs for just over an average number of 18 iterations at its worst iterative convergence performance while the kSD with $\varepsilon = 2$ runs for an average number of about 25 iterations at its worst iterative convergence performance.

The proposed algorithm with $\varepsilon = 4$ is able yield a good performance in terms of SER and average number of iterations because there are more combinations of symbol indexes of the vector $\mathcal{K}_x$ used with the different parity check matrix

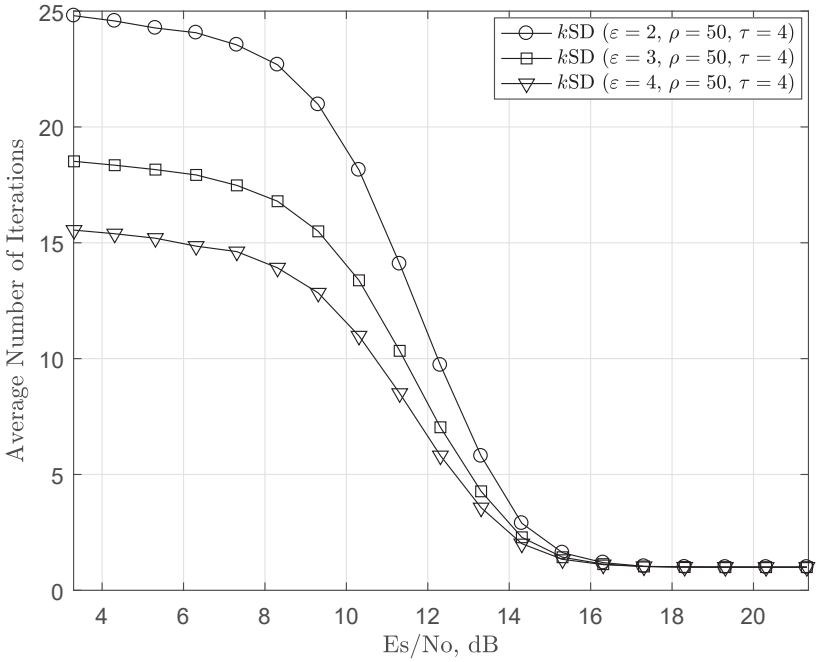


FIGURE 5.8: Iterative performance comparison of the kSD for a $(15, 7)$ RS code over AWGN channel for different values of ε.

extensions h_x. This gives the algorithm more parity check equations which yields more syndrome vectors, s_x, that are used to check if the threshold is met. Using more syndrome vectors increase the chances of the decoding condition being met earlier, therefore reducing the average number of iterations required to decode the received vector. Also due to the larger size of H_ε when $\varepsilon = 4$, the algorithm is able to make more updates to the β matrix in a single iteration which helps the algorithm converge to a codeword quicker.

Due to the performance exhibited by the kSD with $\varepsilon = 4$, $\rho = 50$ and $\tau = 4$, this version of the algorithm is used in the benchmark test for the nearly half rate case. Simulations are also run for a high rate $(15, 11)$ RS code to test for the optimum conditions for the proposed algorithm. Working with higher rates means working with a H matrix that has less rows. Therefore, the length of each vector s_x obtained from the syndrome calculation in (5.17) is smaller. With regards to this, a smaller value of τ has to be used. For this set of simulations the kSD is run with $\tau = 2$. The proposed algorithm is run under the same conditions with the same variants of the kSD algorithm as the nearly half rate case. The results for this

simulations can be seen in Fig. 5.9 and Fig. 5.10.

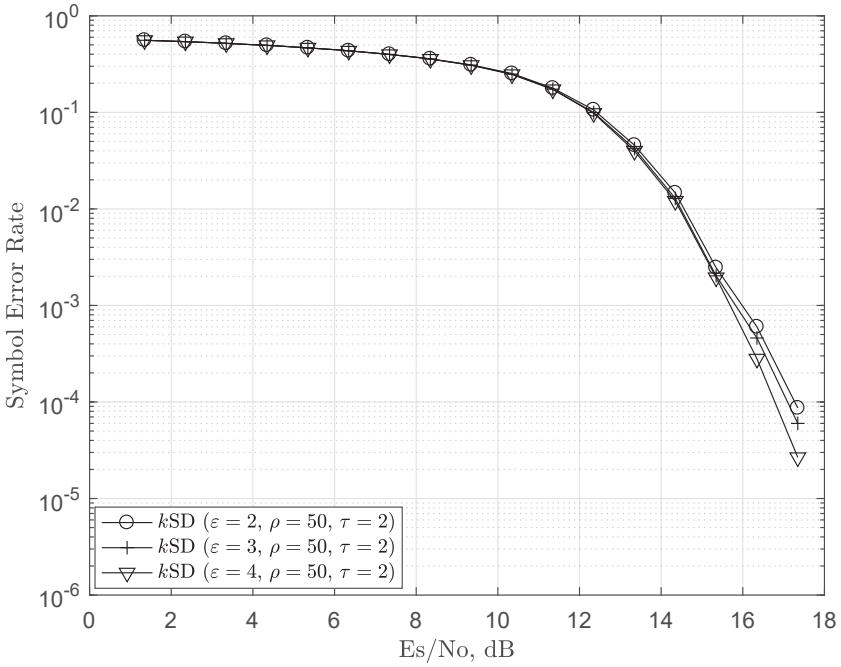


FIGURE 5.9: SER performance comparison of the kSD for a $(15, 11)$ RS code over AWGN channel for different values of ε.

Similar to the nearly half rate case, the proposed algorithm with $\varepsilon = 4$ is seen to yield a better performance gain in terms of SER when compared to the kSD with $\varepsilon = 3$ and $\varepsilon = 2$ as seen in Fig. 5.9. The kSD gives a coding gain of about 0.3dB when compared to the proposed decoder with the $\varepsilon = 3$ and a coding gain of about 0.12dB when compared to kSD with $\varepsilon = 2$. From Fig. 5.10, the kSD with $\varepsilon = 4$ is seen to run for an average number of just over 8 iterations at its worst iterative convergence performance. This iterative performance is better than the kSD with $\varepsilon = 3$ and $\varepsilon = 2$, which run for an average number of just over 10 and just over 14 iterations respectively. Based on this result, the kSD with $\varepsilon = 4$ is used in the benchmarking simulations for the high rate codes.

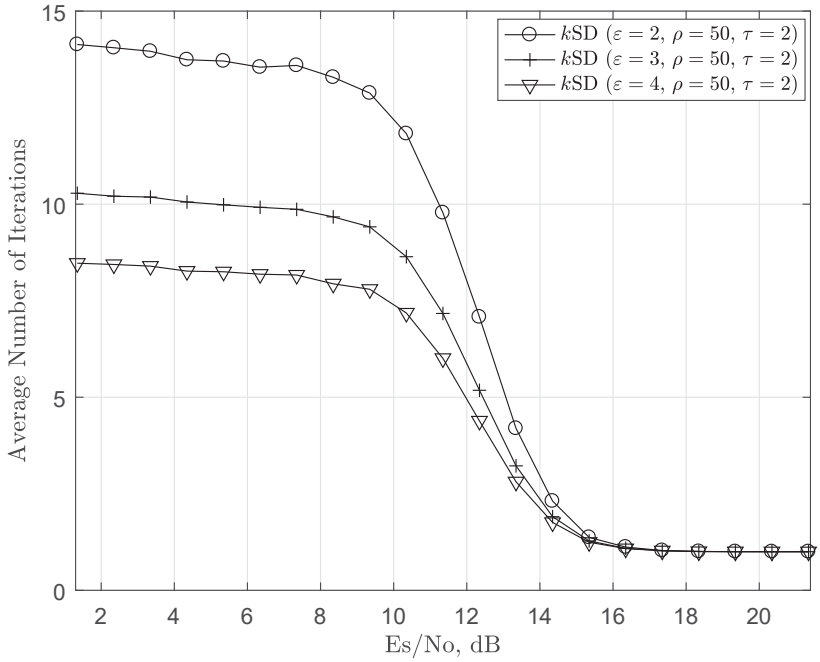


FIGURE 5.10: Iterative performance comparison of the kSD for a $(15, 11)$ RS code over AWGN channel for different values of ε.

5.5.2 Performance Analysis of the Proposed Symbol-Level Decoding Algorithm

In this section the performance for the proposed decoder is benchmarked against other iterative symbol-level decoders. The symbol-level implementations for the decoding algorithms are used in these simulations for the purposes of giving a fair comparison. The decoders used in the simulations are the symbol-level ABP algorithm and the PTA decoder.

5.5.3 Half Rate Codes

For these simulations the symbol-level ABP algorithm is run with a value of $\eta = 0.05$ and the maximum number of iterations is set to 20 at the initialisation of the algorithm [6], while the PTA runs with a value of $\delta = 0.001$ [8]. The simulations are run under similar conditions as the previous tests for the kSD algorithm. The

results for these simulations are presented in Fig. 5.11 and Fig. 5.12.

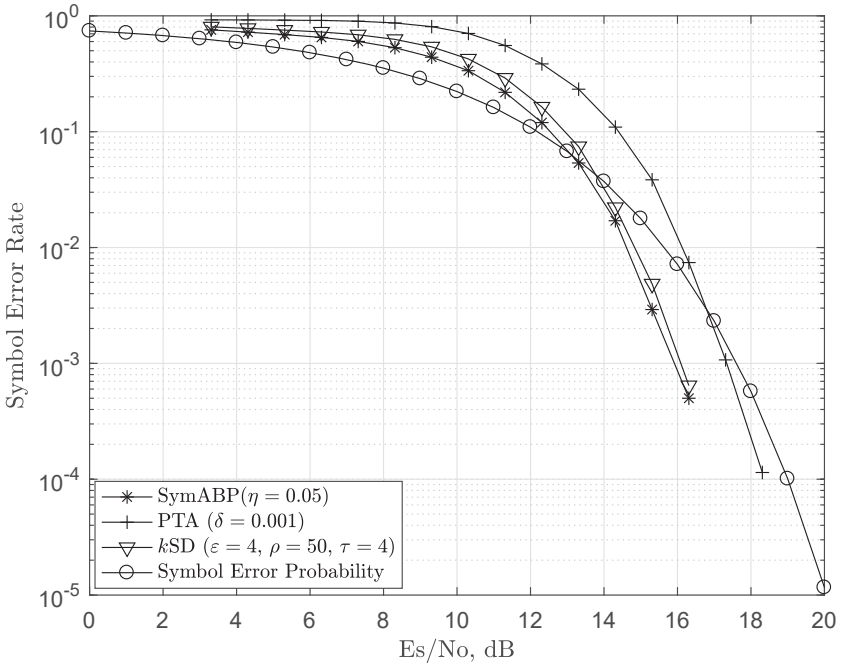


FIGURE 5.11: SER performance comparison of the kSD for a $(15, 7)$ RS code over AWGN channel.

From Fig. 5.11, it can be seen that the kSD with $\varepsilon = 4$ is slightly outperformed by about 0.15dB when compared to the symbol-level ABP algorithm at an SER value of 10^{-3}. The kSD algorithm, when compared to the PTA decoder, is able to yield a gain of about 1.25dB at the same SER value. From Fig. 5.12, it can be seen that the PTA algorithm requires numerous iterations to converge especially for the lower SNR regions. The PTA runs for an average of about 445 iterations at its worst performance which is a large number of iterations when compared to the other iterative symbol-level decoders. The symbol-level ABP runs a maximum of 20 iterations. This is because that is the cap set at the initialisation of the symbol-level ABP algorithm on the maximum number of iterations. The kSD with $\varepsilon = 4$ is able to yield a better convergence rate than all the algorithms having a maximum of an average of just over 15 iterations at its worst iterative convergence performance.

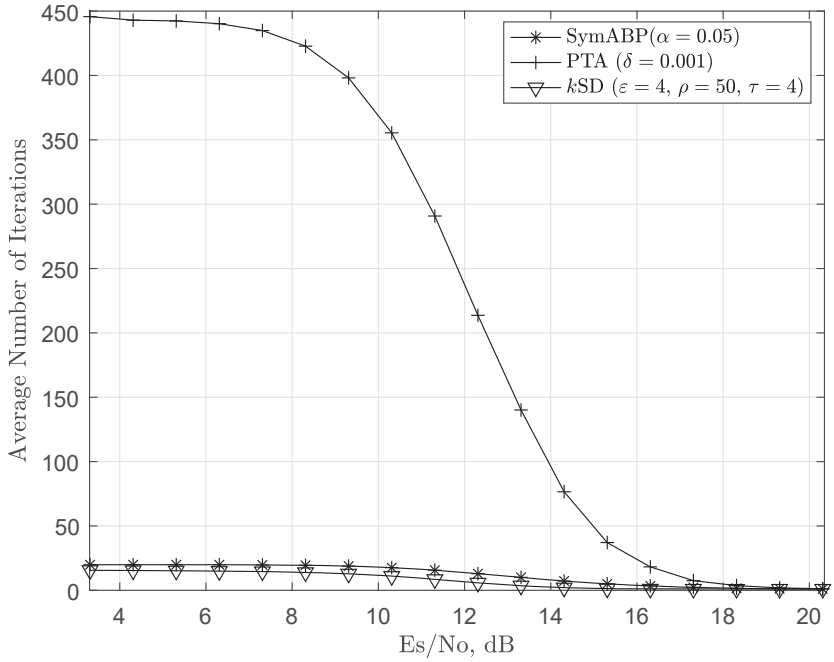


FIGURE 5.12: Iterative performance comparison of the kSD for a $(15, 7)$ RS code over AWGN channel.

5.5.4 High Rate Codes

In this section, the performance of the proposed algorithm for high rate codes is investigated. For these simulation a $(15, 11)$ RS code is transmitted using a 16-QAM modulation scheme with Gray mapping through an AWGN channel. The simulation conditions for the algorithms remain the same as the case for the $(15, 7)$ RS code. The results for these simulations can be seen in Fig. 5.13 and Fig. 5.14. The symbol-level ABP is shown to be the best performing algorithm in Fig. 5.13 as it yields a gain of 0.2dB when compared to the kSD with $\varepsilon = 4$ at the SER value of 10^{-4}. When compared to the PTA, the kSD with $\varepsilon = 4$ gives a gain of 1.3dB at the SER value of 10^{-3}. Similarly to the case of the $(15, 7)$ RS codes, the PTA algorithm requires the most average number of iterations to converge to a codeword during the decoding process with an average of about 380 iterations at its worst iterative convergence performance as seen in Fig. 5.14. The symbol-level ABP once again has a maximum average number of 20 iterations as it is capped at this number during the initialisation of the algorithm. The kSD with $\varepsilon = 4$

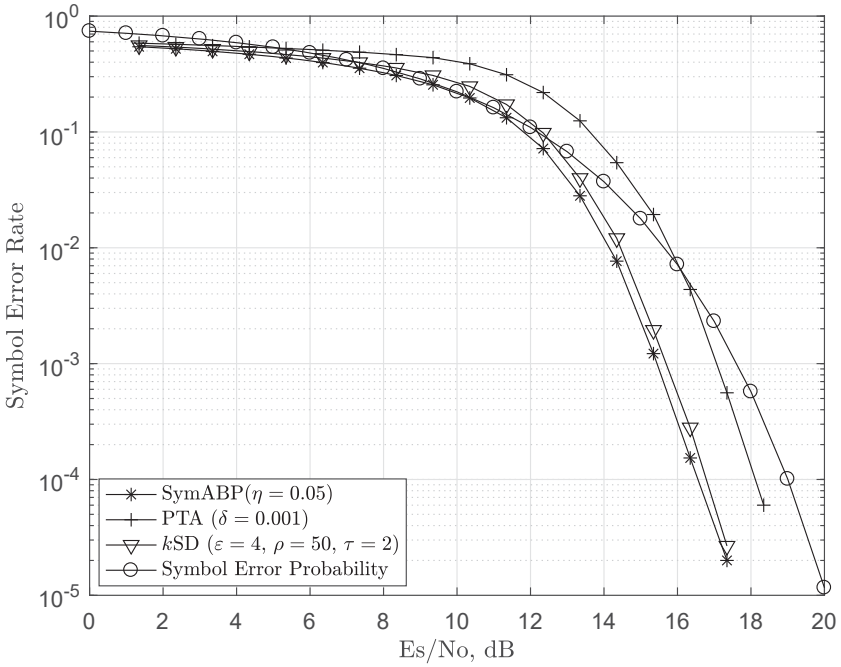


FIGURE 5.13: SER performance comparison of the kSD for a $(15, 11)$ RS code over AWGN channel.

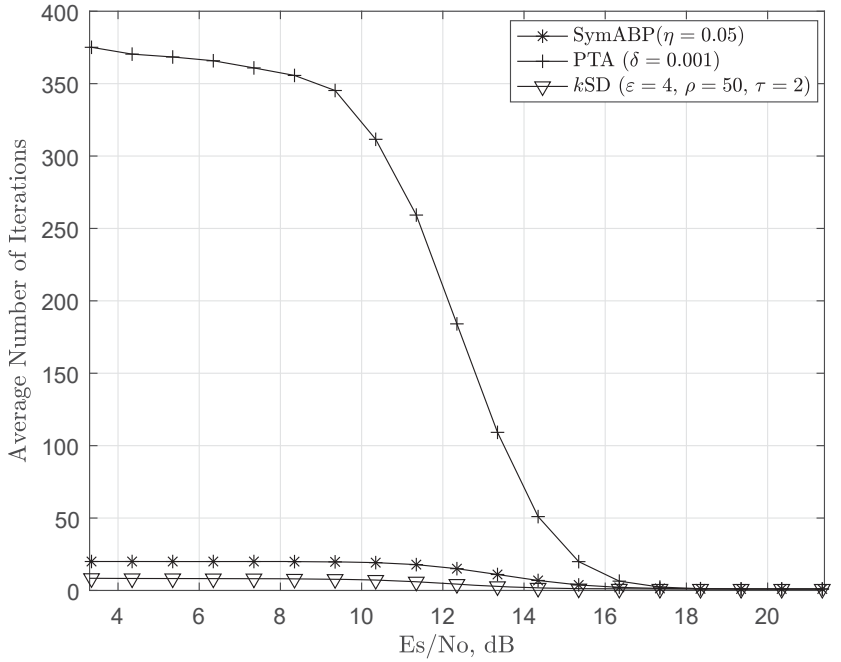


FIGURE 5.14: Iterative performance comparison of the kSD for a $(15, 11)$ RS code over AWGN channel.

requires the least iterations to decode the received vector by taking an average of about only requiring about 8 iterations at its worst performance.

Simulations to better benchmark the kSD against work carried in literature are run on a longer $(63, 55)$ RS code [6]. The kSD is benchmarked against the symbol-level ABP and the symbol-level ABP-HDD. The symbol-level ABP-HDD works in with a similar genie aided stopping criteria which speeds up the decoding process and yields an optimistic result. The simulations are under BPSK modulation conditions and an AWGN transmission. The results for these simulation are presented in Fig. 5.15.

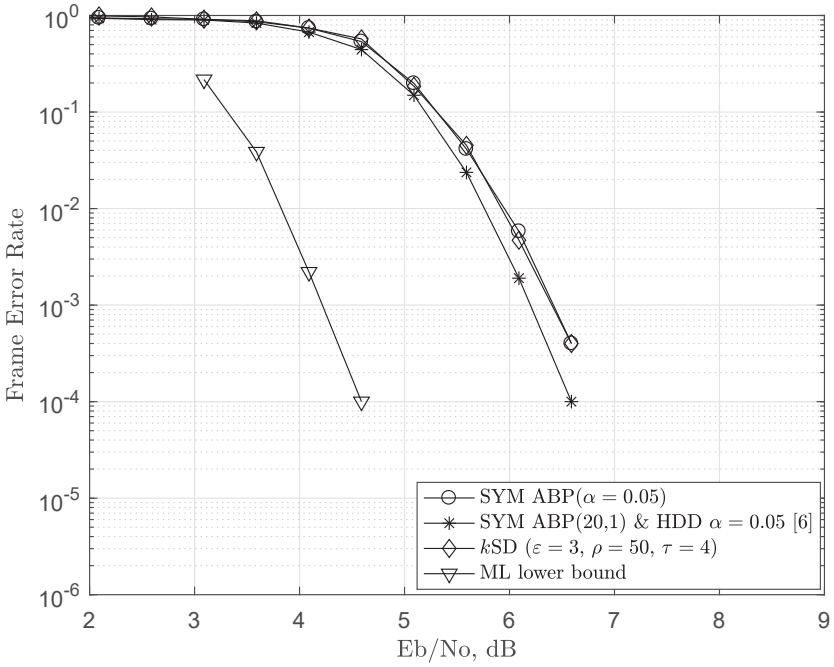


FIGURE 5.15: Performance comparison of the SYM ABP and the kSD for a $(63, 55)$ RS code.

From Fig. 5.15, it can be seen that the kSD is able to yield a performance that is similar to the generic version of the symbol-level ABP. The kSD incurs a loss of symbol-level ABP-HDD of 0.25dB. A bigger loss, of about 2dB, is seen when the kSD is compared to the ML lower bound. However, it is important to note that this bound is set for bit-level decoders which explains the large gain. This is similar to the result in Fig. 4.3 and Fig 4.4 when the PTA_{bl} is compared to the symbol-level PTA.

5.6 Complexity Analysis

5.6.1 Time Complexity

The complexity of the proposed algorithm when compared to the other iterative symbol-level soft-decision decoders is studied in this section. The notation used in the representation of the complexities is as follows

- n = Length of the codeword/ Number of columns in H
- k = The number of information symbols/ The number of columns found in the parity submatrix in H
- $(n-k)$ = Number of rows in H
- ε = The number of extension used
- t = The number of participating symbols in each row. Can be calculated as $k+1$
- W_c = The average column weight of the binary image expansion of H
- W_k = The average row weight of bits in the binary image of H used in the calculation of the extrinsic information.

The complexity of the algorithms are measured in terms of the total number of operations performed in each iteration. The overall complexity for the symbol-level PTA, symbol-level ABP and the kSD algorithm are summarised in the Table 5.1, Table 5.2 and Table 5.3 respectively.

TABLE 5.1: Summary of the overall complexity for the symbol-level PTA decoder.

Decoding Stage	**Stage description**	**Number of Operations**
Finding the reliability vector	Searching for maximum values in β	n
Sorting of reliabilities	Reliabilities sorted in ascending order	n
Transforming matrix H	Row reduction of H	$((n-k)^2 \times n)$
Obtaining the hard-decision vector $\widehat{c}$	Assigning a symbol value from the soft information in β	n
Syndrome Check	Performing the calculation $\widehat{c} \cdot H$	$((n-k) \times n)$
Correction step	Updating the β reliabilities based on each Syndrome check equation	$((n-k) \times t)$
Overall Complexity of the symbol-level PTA decoder		
Total $= n + n + ((n-k)^2 \times n) + n + ((n-k) \times n) + ((n-k) \times t)$		
Time Complexity $= \mathcal{O}((n-k)^2 \times n)$		

TABLE 5.2: Summary of the overall complexity for the symbol-level ABP decoder.

Decoding Stage	**Stage description**	**Number of Operations**
Obtaining the reliability vector	Searching for maximum values in β	n (1st iteration only)
Sorting of the symbol reliabilities	Reliabilities sorted in ascending order	n (1st iteration only)
Adapting the matrix H	Row reduction of H	$((n-k)^2 \times n)$ (1st iteration only)
Obtaining the binary image of adapted H	Replacing each symbol in H with the corresponding binary companion matrix	$((n-k) \times n)$ (1st iteration only)
Obtaining the bit LLR values from the received vector	Applying the approach presented in Section 2.9.3.2	N (1st iteration only)
Obtaining the vector L_e	Calculating the extrinsic information as shown in (2.31)	$(M \times W_k^2) + \mathcal{O}(N \times W_c)$
Updating the vector L	Adding $L_e(cb_j)$ to $L(cb_j)$	N
Obtaining the hard-decision vector $\widehat{cb}$	Assigning one of the binary values to each $L(cb_j)$	N
Syndrome Check	Performing the calculation $\widehat{cb} \cdot \mathcal{H}$	$(M \times N)$

Overall Complexity of the symbol-level ABP decoder

Total = $\overbrace{n + n + ((n-k)^2 \times n) + ((n-k) \times n) + N}^{\text{Only included in the } 1^{st} \text{ iteration}} + (M \times W_k^2) + (N \times W_c) + N + N + (M \times N)$

Time Complexity = $\mathcal{O}(M \times W_k^2)$

TABLE 5.3: Summary of the overall complexity for the kSD algorithm.

Decoding Stage	**Stage description**	**Number of Operations**
Finding the vector $\beta_{\max}$	Searching for maximum values in β	n
Sorting the reliabilities	Reliabilities sorted in ascending order	n
Obtaining the vectors $\mathcal{U}_x$ and $\mathcal{K}_x$	Swapping of 2 bit indexes using (5.15)	$((\varepsilon - 1) \times 2)$
Transforming matrix H_ε	Row reduction of each H_x	$((n-k)^2 \times \varepsilon) \times n)$
Obtaining the hard-decision vector $\widehat{c}$	Assigning a symbol value from the soft information in β	n
Syndrome Check	Applying (5.17)	$(((n-k) \times \varepsilon) \times n)$
Calculating the symbol votes	Applying (5.22)	$(((n-k) \times \varepsilon) \times k^2)$
Finding the symbol voting ratios	Summing the votes and dividing each vote by the total as shown in (5.23)and (5.24)	$(2^m + 2^m)$
Obtaining the bit confidence rating	Dividing the voting ratios by the ρ as shown in (5.26)	(2^m)
Updating β	Bit reliabilities are updated in β	$(((n-k) \times \varepsilon) \times t)$
Scaling reliabilities in β	If $S \neq 0$ or τ is not met, (4.1) is applied in preparation for the next iteration	$(2^m \times n) + (2^m \times n)$
Checking for decoding condition	if the threshold τ is met by the informational set	τ
Applying the Decoding condition	Applying (4.19)	$((n-k) \times k)$ - only applied in the final iteration if $S \neq 0$ when the threshold τ is met

Overall Complexity of the kSD algorithm

$$\text{Total} = n+n+((\varepsilon-1)\times 2)+(((n-k)^2\times\varepsilon)\times n)+n+(((n-k)\times\varepsilon)\times n)+(((n-k)\times\varepsilon)\times k^2)+(2^m+2^m)+2^m+(((n-k)\times\varepsilon)\times t)+\underbrace{((2^m\times n)+(2^m\times n))}_{\text{if } S\neq 0 \text{ or } \tau \text{ is not met}}+\tau+\underbrace{((n-k)\times k)}_{\text{last iteration only}}$$

$$\text{Time Complexity} = \mathcal{O}((((n-k)\times\varepsilon)\times k^2))$$

Similar to the bit level case, we summarize the computational complexity for the symbol-level iterative decoders by noting their 'additions/subtraction', 'multiplications/divisions' and 'other' operations. This can be seen in Table 5.4, Table 5.5 and Table 5.6. Once again, the tables are only filled with the operations unique to at least one of the decoders.

TABLE 5.4: Summary of the computational complexity for the symbol-level PTA decoder.

Decoding Stage	$+/-$	$\times/\div$	other
Transforming matrix H (Based on matrix multiplication [8])	$((n-k)^2 \times n)$	$((n-k)^2) \times n)$	
Obtaining the hard-decision vector $\widehat{c}$			n
Syndrome Check	$((n-k) \times n)$	$((n-k) \times n)$	
Correction step	$((n-k) \times t)$		

TABLE 5.5: Summary of the computational complexity for the symbol-level ABP decoder.

Decoding Stage	$+/-$	$\times/\div$	other
Adapting the matrix H (Based on Gaussian row reduction [131])	$(2n^3+3n^2-5n)/6$	$(2n^3+3n^2-5n)/6+$ $n(n+1)/2$	
Obtaining the binary image of adapted H			$((n-k) \times n)$
Obtaining the bit LLR values from the received vector		N	
Obtaining the vector L_e	$(N \times W_c)$	$(M \times W_k^2)$	
Updating the vector L	N		
Obtaining the hard-decision vector $\widehat{cb}$			N
Syndrome Check	$(M \times N)$	$(M \times N)$	

The time complexity of the symbol-level ABP presented in Table. 5.2 is seen to be the same as the bit-level ABP due to the extrinsic information being calculated in the same way. Unlike the kBD, the time compexity kSD decoder is dominated by a function that uses multiplications and divisions as seen in Table. 5.2, Table. 5.5 and (5.22). However the matrix dimension of the kSD are much smaller when compared to the symbol-level ABP which, in spite of adopting the H matrix at a symbol level, still performs the belief propagation on a larger H matrix at the bit-level.

TABLE 5.6: Summary of the computational complexity for the kSD algorithm.

Decoding Stage	$+/-$	$\times/\div$	other
Obtaining the vectors $\mathcal{U}_x$ and $\mathcal{K}_x$			$((\varepsilon-1)\times 2)$
Transforming matrix H_ε (Based on matrix multiplication [8])	$\varepsilon((n-k)^2 \times n)$	$\varepsilon((n-k)^2 \times n)$	
Obtaining the hard-decision vector $\widehat{c}$			n
Syndrome Check	$(((n-k)\times\varepsilon)\times n)$	$(((n-k)\times\varepsilon)\times n)$	
Calculating the symbol votes		$(((n-k)\times\varepsilon)\times(t+t^2))$	
Finding the symbol voting ratios	2^m	2^m	
Obtaining the symbol confidence rating		2^m	
Updating β	$(((n-k)\times\varepsilon)\times t)$		
Scaling reliabilities in β	$(2^m \times n)$	$(2^m \times n)$	
Checking for decoding condition			τn
Applying the Decoding condition	$((n-k)^2 \times k)$	$((n-k)^2 \times k)$	

5.6.2 Complexity measured in Terms of Number of Operations

In order to vividly differentiate the complexities in the algorithms, simulation are run to generate graphs similar to those in Section. 4.5. Similar to Section 4.5.2, these simulations assist in providing a study on the effect of the number of iterations on the complexity cost of the decoding algorithms. The graphs generated using the complexity equations presented in Table 5.1, Table 5.2 and Table 5.3 multiplied by the average number iterations used to plot the graphs in Fig. 5.11 and Fig. 5.13.

Before the graphs are generated, the actual values for each of the notation used is determined. For the $(15, 7)$ Rs code $n = 15$, $k = 7$, $(n-k) = 8$, $\varepsilon = 4$, $t = 8$, $W_c = 7.86$, $W_k = 13.74$ and $\tau = 4$.

The complexity graph for the $(15, 7)$ RS code can be seen in Fig. 5.16

From Fig. 5.16, it can be seen that the performance gains made by the symbol-level ABP come at the cost of a higher computational complexity when compared

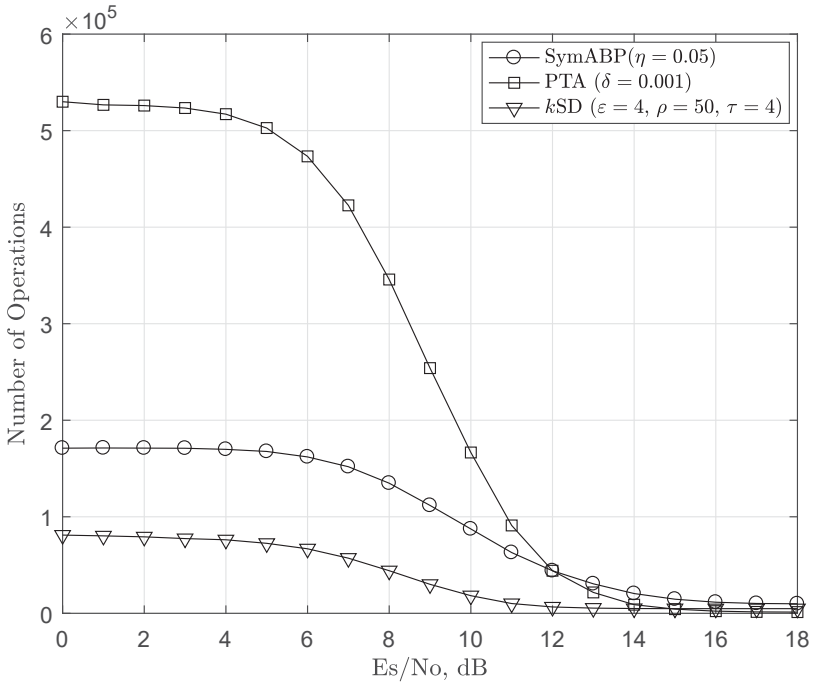


FIGURE 5.16: Complexity comparisons for the symbol-level decoders applied to a (15,7) RS code over an AWGN channel using a 16-QAM modulation scheme.

to the kSD algorithm. The PTA appears to be the least complex computationally for a single iteration as shown in Table 5.1. However, similar to the bit-level case, the numerous iterations required by the PTA to decode the received vector, as seen in Fig. 5.12, add to its total computational complexity cost. It can also be seen from Fig. 5.12 the kSD is able to perform at a complexity that is tolerable when compared to the other iterative symbol-level decoders. It is also important to note that while performing at a much lower complexity, the kSD algorithm with $\varepsilon = 4$ is outperformed by only 0.15dB by the symbol-level ABP for the $(15, 7)$ RS code.

Similar simulations are also run for the high rate $(15, 11)$ RS code to generate the complexity graphs. The actual values used for the notations used in the complexity equations are $n = 15$, $k = 11$, $(n - k) = 4$, $\varepsilon = 4$, $t = 12$, $W_c = 5.99$, $W_k = 21.48$ and $\tau = 2$. The complexity graph for the $(15, 11)$ RS code simulations can be seen in Fig. 5.17

The symbol-level ABP has the highest computational complexity cost when compared to the other iterative soft-decision decoders used in the $(15, 11)$ RS code

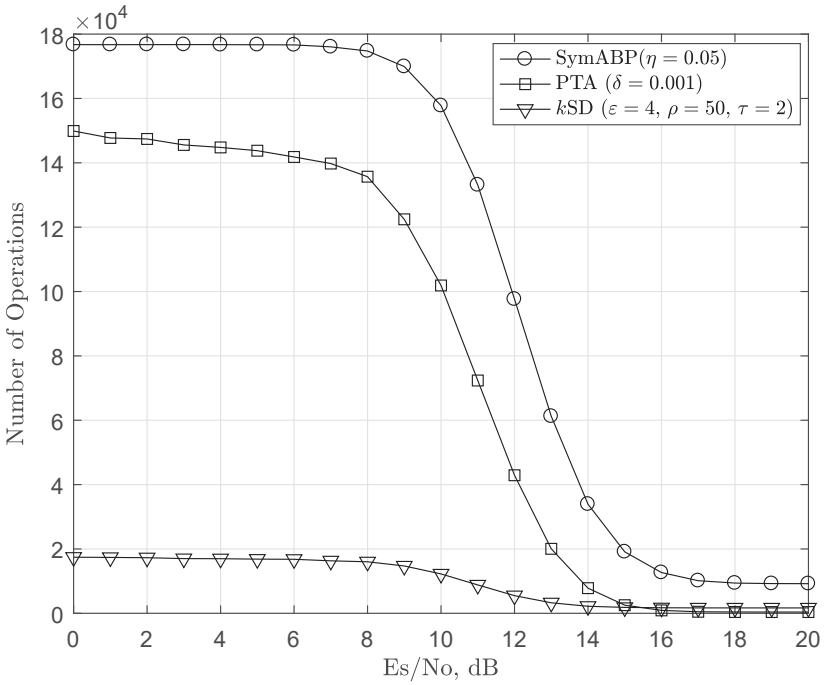


FIGURE 5.17: Complexity comparisons for the symbol-level decoders applied to a (15,11) RS code over an AWGN channel using a 16-QAM modulation scheme.

simulations as seen in Fig. 5.17. The complexity cost of the symbol-level ABP significantly increases for the $(15, 11)$ RS code when compared to the $(15, 7)$ RS code due to the binary image of the H having less rows and a smaller identity submatrix. This makes the binary H matrix more dense when compared to the nearly half rate case. The more dense structure means more bits participate in belief propagation stage of this algorithms as seen by the higher value of W_k for the $(15, 11)$ RS code when compared to the $(15, 7)$ RS code. Therefore, any calculations that requires this parameter adds to the number of operations performed by the symbol-level ABP during the decoding of the received vector. Also due to the more dense structure, the ability of the algorithm to handle the soft information during the message passing stages of the algorithm is not as efficient as for the $(15, 7)$ RS code. This causes the algorithm to run more iterations on average. This causes the algorithm to break on most cases, for the lower SNR values, due to the maximum number of iterations being reached as opposed to the decoder converging to a decoded codeword.

The PTA decoder still operates at a high computational complexity largely due to

the high number of iterations required during the decoding of the received vector. The complexity for the kSD when compared to the other symbol-level decoders is more favourable for the high rate case. The complexity performance is much less largely due to the algorithm having an improved iterative performance as seen in Fig. 5.14. The fact that the symbol-level ABP decodes the received vector using the binary image of H and the numerous iterations run by the PTA algorithm also contribute to the huge performance difference in complexity when compared to the kSD for the high rate case. Considering the kSD is outperformed by about 0.2dB for the $(15, 11)$ RS code by the symbol-level ABP, the use of the algorithm when a tradeoff is required in terms of performance and computational complexity can be justified.

5.7 The Decoding Condition Used as an Additional Stopping Criteria for the Symbol-Level PTA Decoder

One of the main novelties of the PTA algorithm is that it is able to work iteratively with the soft information on a dense parity check matrix of an RS code. Working with the dense H matrix poses a huge problem with the implementation of the decoding condition on the PTA algorithm. This is because all the symbols indices that match the parity submatrix participate in all the syndrome check equations, unlike the bit-level case. This creates an issue in trying to determine a threshold for the decoding condition to be met. To counter this problem the extended matrix approach defined in section 5.2.1 is used. With the matrix extension used, the threshold τ, is now defined as the minimum number of syndrome check equations that should be satisfied by a single extension for the decoding condition to be applied as described in section 5.4.

Similar simulations to those run in section 5.2.2 and section 5.5 are done with the same conditions. From section 5.2.2 the extension of the PTA with 3 parity check matrices, PTAe_3, was found to be the best performing decoder when both the error correction and iterative convergence rate are considered. In this section, the decoding condition is applied to the PTAe_3 with a $\tau = 3$ and compared to the

normal PTAe_3 algorithm in terms of SER and the average number of iterations required to decode the received vector.

The first set of simulations are run for a (15,7) RS code. This set of simulations can be seen in Fig. 5.18 and Fig. 5.19.

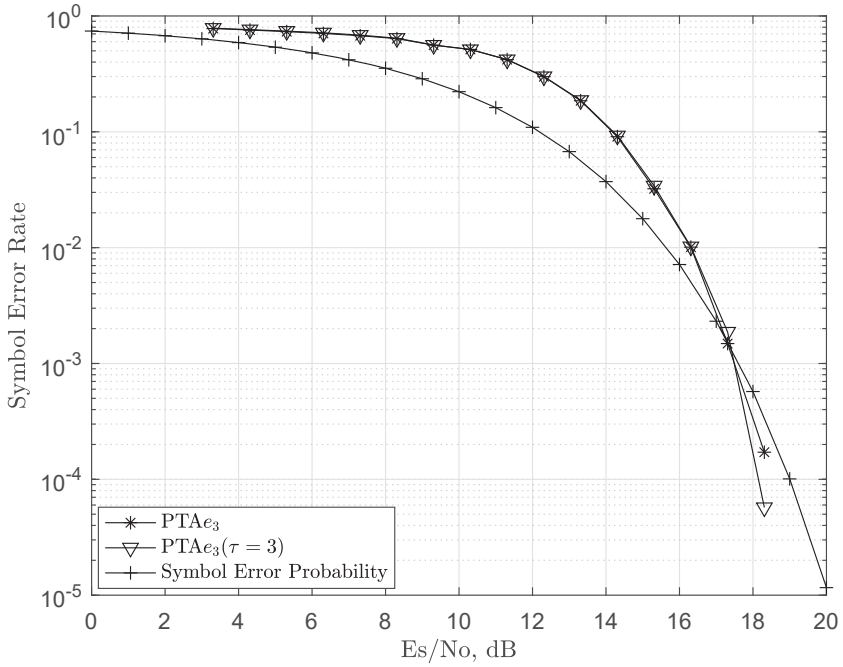


FIGURE 5.18: SER perfomance of PTAe_3 vs PTAe_3 implemented with the decoding condition defined by $\tau = 3$.

From Fig. 5.18 it can be seen that the inclusion of the decoding condition to the PTAe_3 is able to yield a similar error correction performance when compared to the normal PTAe_3 algorithm at an SER value of 10^{-3}. This SER performance yielded by the PTAe_3 using the decoding condition is attained while running at less iterations as shown in Fig. 5.19. The PTAe_3 with the decoding condition requires an average of about 10 iterations at its worst iterative convergence performance to decode the received vector, while the original implementation of the PTAe_3 algorithm requires an average of about 16 iterations at its worst performance to decode the received vector. Including the decoding condition to the PTAe_3 increases the overall efficiency of the algorithm because the SER performance is maintained while the iterative convergence performance of the algorithm is improved.

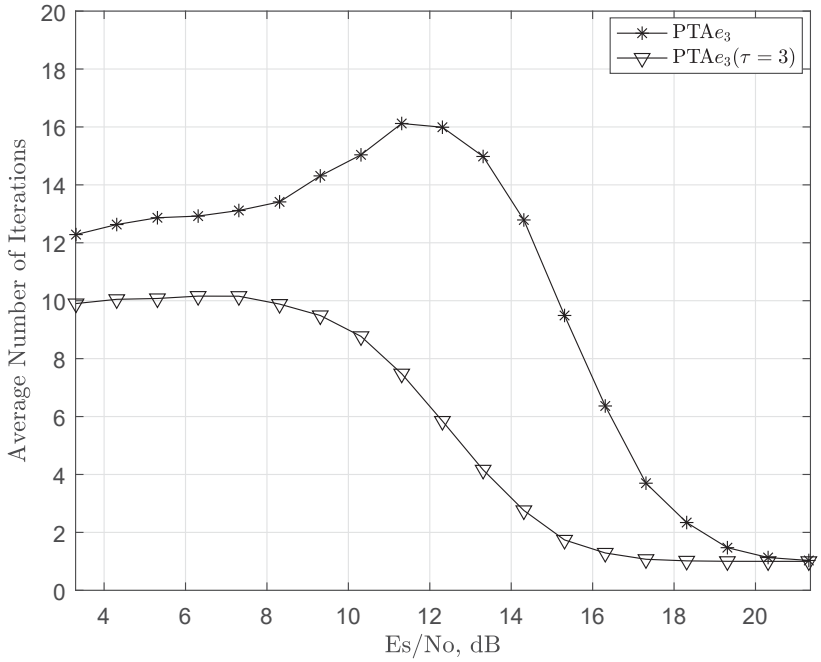


FIGURE 5.19: Iterative performance of PTAe_3 vs PTAe_3 implemented with the decoding condition defined by $\tau = 3$.

Simulations for high rate $(15, 11)$ RS code are also carried out. For a $(15, 11)$ RS code the H matrix is much smaller. The H matrix has only 4 rows. Having only 4 rows in the H matrix means there is only a maximum of 4 syndrome check equations that can be satisfied compared to a maximum of 8 for the (15,7) RS code. Due to the reduced number of syndrome check equations that can be satisfied, the value of τ has to be reduced as well. With regards to this, these simulation are run using PTAe_3 implemented with the decoding condition having $\tau = 1$. The results for these simulations can be seen in Fig. 5.20 and Fig. 5.21.

Similar results are seen in Fig. 5.20 and Fig. 5.21 as those from the (15,7) RS code simulations. The PTAe_3 using the the decoding condition as an additional stopping criteria yields a similar SER performance to that of the original implementation of the PTAe_3 decoder at a SER of 10^{-3}. It is important to note that the PTAe_3 implemented with the decoding condition runs for significantly less iterations when compared to the original implementation of the PTAe_3 decoder. From Fig 5.21, the original implementation of the PTAe_3 runs an average of about 30

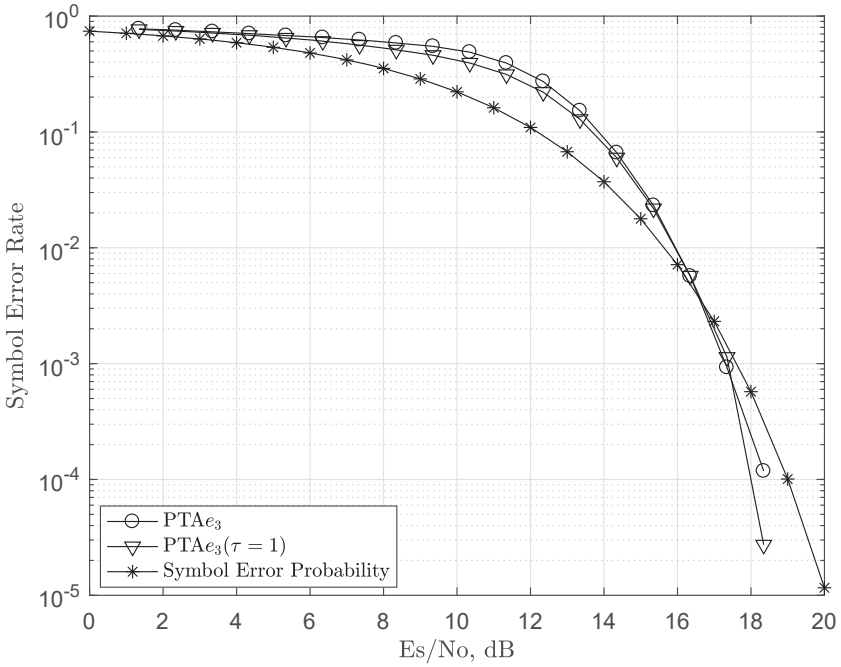


FIGURE 5.20: SER performance of PTAe_3 vs PTAe_3 implemented with the decoding condition defined by $\tau = 1$.

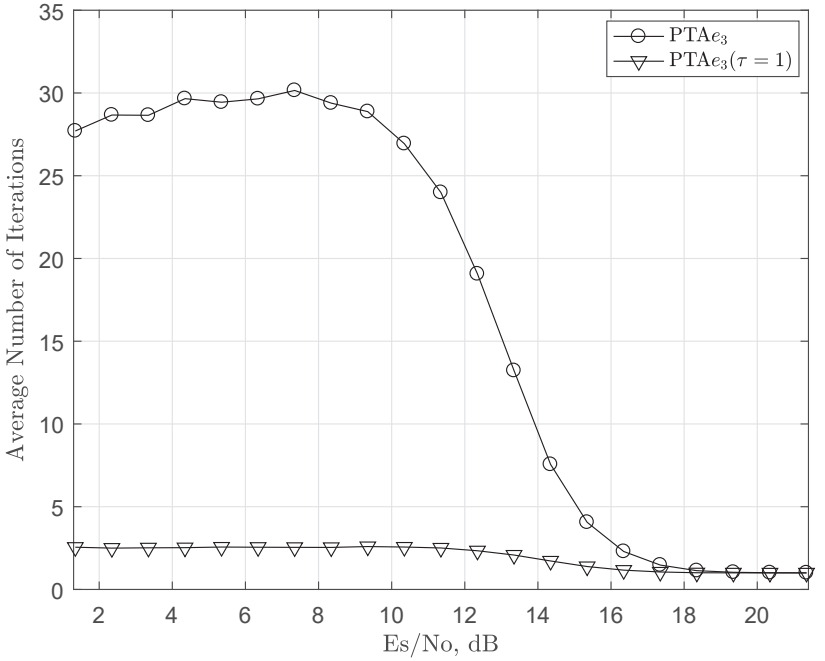


FIGURE 5.21: Iterative performance of PTAe_3 vs PTAe_3 implemented with the decoding condition defined by $\tau = 1$.

iteration to decode the received vector while the PTAe_3 implemented with decoding condition having $\tau = 1$ runs for less than an average of less than 3 iterations at it worst iterative convergence performance during the decoding process. Using a value of $\tau = 1$ enables the decoding condition to met with less iterations, thus improving the iterative convergence rate of the decoder while obtaining the decoded vector.

5.8 Conclusion

An iterative symbol-level implementation of the proposed soft-decision information set decoder is presented in this chapter. The proposed symbol-level implementation of the algorithm has several variations depending on the number of parity check matrix extensions used. The parity check matrix extensions, included to the kSD, add to the number of parity check equations used with the received vector during decoding. Rearranging the columns of the different parity check matrix extensions based on the reliability information helps in identifying errors that may be located in the different information sets that could be previously not be found. Using the parity check matrix extensions and setting a threshold on the number of satisfied checks by the k information set symbols improves the iterative performance of the algorithm while yielding a good SER performance, therefore making the decoder more efficient. Adding more parity check matrix extensions increase the complexity of the proposed decoder for a single iteration. However, the overall SER and iterative performance is improved. This provides a justifiable tradeoff when choosing the number of parity check matrix extensions to use during decoding.

The decoding condition is also implemented on a symbol-level when applied to the PTA algorithm. Similar to the bit-level case, the implementation is able to maintain the SER performance while improving on the iterative convergence rate of the soft-decision decoder. This improves on the efficiency while reducing the computational complexity cost of the algorithm.

CHAPTER 6

The Decoding condition applied to Binary QC-LDPC codes

6.1 The Proposed Systematic Construction for QC-LDPC Codes

Implementation of the decoding condition with LDPC codes is different from that of RS codes. The is because for RS codes row reduction operations are performed in each iteration so as to set the most reliable k symbols to match the parity submatrix of the transformed H matrix. This cannot be directly implemented for LDPC codes because the row reduction operations will make the H matrix lose its sparse structure. The sparse structure of LDPC codes is important for decoding as it helps the efficient exchange of information between nodes so as to get reliable information on the symbols [1]. This means that a systematic structure has to be obtained from the H matrix without losing the sparse nature of the QC-LDPC code. This presents a challenge in the implementation of the proposed decoding approach as most QC-LDPC codes are not defined by a H matrix with a systematic structure. Therefore, the construction of a QC-LDPC code defined by a H matrix with a systematic structure is required for the implementation of the decoding condition on this class of code. The proposed systematic QC-LDPC code construction is made possible by using a row reduction technique that is different from the usual Gaussian operations.

6.1.1 General Encoding of QC-LDPC codes

To understand how the proposed systematic QC-LDPC H matrix is constructed, a general approach of encoding for QC-LDPC codes [15] is first presented. This method is highly favoured as it maintains the quasi cyclic structure for both the parity check matrix (H) and the generator matrix (G).

Notation used for the construction of the systematic QC-LDPC code is first established. The encoding process is performed on a parity check matrix made up of a set of $(n-k) \times n$ sparse circulants with the from of H_{qc}:

$$H_{qc} = \begin{bmatrix} C_{0,0} & C_{0,1} & \dots & C_{0,(n-1)} \\ C_{1,0} & C_{1,1} & \dots & C_{1,(n-1)} \\ \vdots & & \vdots & \\ C_{((n-k)-1),0} & C_{((n-k)-1),1} & \dots & C_{((n-k)-1),(n-1)} \end{bmatrix}, \tag{6.1}$$

where $C_{i,j}$ is a square circulant or zero submatrix, for $1 \leq i \leq (n-k)$ and $1 \leq j \leq n$. A circulant matrix is a matrix in which each row is a right cyclic shift of the row above it [132]. From this definition, it can be seen that a circulant matrix can be completely characterized and constructed using only its first row. This row is referred to as the generator of the circulant matrix.

The encoding technique proposed in [15] creates a generator matrix characterized by $k \times n$ sparse circulants, in the form of G_{qc}:

$$G_{qc} = \begin{bmatrix} I_{0,0} & \dots & O_{0,(n-k-1)} & G_{0,(n-k)} & \dots & G_{0,(n-1)} \\ O_{1,0} & \dots & O_{1,(n-k-1)} & G_{1,(n-k)} & \dots & G_{1,(n-1)} \\ \vdots & \ddots & \vdots & \vdots & \vdots & \vdots \\ O_{(k-1),0} & \dots & I_{(k-1),(n-k-1)} & G_{(k-1),(n-k)} & \dots & G_{(k-1),n-1} \end{bmatrix}, \tag{6.2}$$

where I is a square identity submatrix and O is square zero submatrix. $G_{i',j}$ represents the new circulant submatrices obtained from the encoding process, where $1 \leq i' \leq k$ and $1 \leq j \leq n$. This systematic form of G_{qc} is only obtained when the

matrix H_{qc} is full rank [15].

The encoding process involves finding the generator of the circulant $G_{i',j}$ by the use of Gaussian row reduction operations as shown in [15]. This row reduction computations are carried out k times as this is the number of row blocks of the matrix G_{qc}.

For non full rank matrices an additional submatrix Q is added to the matrix G_{qc} with the same number of rows as the dependent columns of the matrix H_{qc}. This involves additional Gaussian row reduction operations and also causes the matrix G_{qc} to lose its systematic structure.

This encoding technique is efficient when used to maintain the quasi cyclic structure of QC-LDPC codes. However, due to the numerous row reduction computations required it can become quite complex for larger codes.

6.1.2 The Proposed Row Reduction Technique

The proposed row reduction method used to give the systematic structure of the G matrix is based on the underlying idea in [127]. To better explain how this technique works, an example of a matrix X with dimensions $(n-k) \times n$ is considered. In order to perform row reduction on the matrix X, a $(n-k) \times (n-k)$ submatrix X_I is formed with the first $(n-k)$ columns of X as:

$$X = [X_I \mid X_Q], \tag{6.3}$$

where X_Q represents the $(n-k) \times k$ submatrix formed using the remaining k columns of X. If the columns of X_I are independent, the reduced row echelon form (X_{rref}) of X can be obtained by finding the inverse of matrix X_I and multiply it by X:

$$X_{\text{rref}} = [\ X_I\ ]^{-1} \cdot [\ X\ ]. \tag{6.4}$$

This row reduction technique can be applied to any set of $(n-k)$ independent columns of the matrix X. The columns of the identity submatrix of X_{rref} will match the order of the $(n-k)$ independent columns used in the submatrix X_I.

6.1.3 The Systematic QC-LDPC Parity check Matrix

LDPC codes are defined by their parity check matrices, H. QC-LDPC codes are defined by a H matrix made up of an array of sparse circulant square submatrices [133]. For the field GF(q^m), construction of a QC-LDPC H matrix is based on replacement of elements in a $(n-k)\times n$ base matrix B with $b'\times b'$ sparse circulant submatrices, where $b'=(q^m-1)$. This gives a $(n-k)b'\times nb'$ parity check matrix [134]. For the construction proposed in this section, the base matrix is defined as a $(n-k)\times n$ Vandermonde matrix, where $(n-k)<n$ and $(n-k)+n\leq q^m$, is of the form shown in (6.5),

$$B=\begin{bmatrix} 1 & \alpha^1 & \alpha^2 & ... & \alpha^{(n-1)} \\ 1 & \alpha^2 & \alpha^{2(2)} & ... & \alpha^{2(n-1)} \\ \vdots & \vdots & \vdots & & \vdots \\ 1 & \alpha^{(n-k)} & \alpha^{(n-k)(2)} & ... & \alpha^{(n-k)(n-1)} \end{bmatrix}, \tag{6.5}$$

and α represents the primitive element of the field.

The reason a Vandermonde matrix is selected is because of its full row rank inverse properties [135]. The proposed row reduction technique is then used to reduce the last $(n-k)\times(n-k)$ submatrix of the base matrix. This will give a base matrix of the form:

$$B=[B_Q \mid B_I], \tag{6.6}$$

where B_Q is a $(n-k)\times k$ submatrix and B_I is a $(n-k)\times(n-k)$ identity submatrix. A q-ary matrix dispersion [134] is then performed on the base matrix B to give a parity check matrix H of the form shown in (6.7), where $C_{i,j}$ and $D_{i,j}$ represents a q-ary $b'\times b'$ α *multiplied circulant permutation matrix* [134]. $O_{i,j}$ represents a

zero matrix.

$$H = \begin{bmatrix} C_{0,0} & \cdots & C_{0,(k-1)} & D_{0,(k)} & \cdots & O_{0,(n-1)} \\ C_{1,0} & \cdots & C_{1,(k-1)} & O_{1,(k)} & \cdots & O_{1,(n-1)} \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ C_{((n-k)-1),0} & \cdots & C_{((n-k)-1),(k-1)} & O_{((n-k)-1),(k)} & \cdots & D_{((n-k)-1),(n-1)} \end{bmatrix} \tag{6.7}$$

Due to the $(n-k)\times(n-k)$ identity submatrix in B the last $(n-k)b'\times(n-k)b'$ submatrix of H is a diagonal matrix. D represents the diagonal submatrices in H. The proposed row reduction technique is then used to convert the diagonal $(n-k)b'\times(n-k)b'$ submatrix of H into an identity matrix. The matrix H is now of the form shown in (6.8), where $I^*_{i,j}$ represent an identity submatrices and $C^*_{i,j}$ represents the new circulant submatrices obtained from the row reduction.

$$H = \begin{bmatrix} C^*_{0,0} & \cdots & C^*_{0,(k-1)} & I^*_{0,(k)} & \cdots & O_{0,(n-1)} \\ C^*_{1,0} & \cdots & C^*_{1,(k-1)} & O_{1,(k)} & \cdots & O_{1,(n-1)} \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ C^*_{((n-k)-1),0} & \cdots & C^*_{((n-k)-1),(k-1)} & O_{((n-k)-1),(k)} & \cdots & I^*_{((n-k)-1),(n-1)} \end{bmatrix}, \tag{6.8}$$

This construction technique can be applied to both cases of binary and nonbinary QC-LDPC codes.

6.1.4 The Generator matrix (Systematic Encoder)

The above matrix H is in systematic form and can be expressed as:

$$H = [C^* \; I^*_{(n-k)b'}]. \tag{6.9}$$

Therefore the generator matrix can easily be created by extracting the parity submatrix, C^*, of the H matrix and finding its transpose. This submatrix is then dimensionally matched to a corresponding identity matrix to give the form

$$G = [I^*_{kb'} \; C^{*T}]. \tag{6.10}$$

The G matrix has the dimensions $kb' \times nb'$, and the expanded form is shown in (6.11)

$$G = \begin{bmatrix} I^*_{0,0} & \dots & O_{0,(n-k-1)} & C^{*T}_{0,(n-k)} & \dots & C^{*T}_{0,(n-1)} \\ O_{1,0} & \dots & O_{1,(n-k-1)} & C^{*T}_{1,(n-k)} & \dots & C^{*T}_{1,(n-1)} \\ \vdots & \ddots & \vdots & \vdots & \vdots & \vdots \\ O_{(k-1),0} & \dots & I^*_{(k-1),(n-k-1)} & C^{*T}_{(k-1),(n-k)} & \dots & C^{*T}_{(k-1),(n-1)} \end{bmatrix} \tag{6.11}$$

6.1.5 Performance Analysis

In this section, the performance of the proposed structure with the construction of a QC-LDPC code based on field partitions[20] is compared. This construction is selected because it is a high-rate, high performance QC-LDPC code [116]. The general technique used to create the G matrix for QC-LDPC codes shown in [15] is applied to this construction. A rank analysis is carried out on all the QC-LDPC codes to determine the rate of the code and assist in the construction of the G matrix. The performance will be measured in terms of BER comparison and a complexity analysis.

6.1.5.1 Rank Analysis

LDPC codes are differentiated from other classes of linear block codes due to the fact that they are defined by their pariy check matrices. This means is that the parity check matrix is first constructed before the generator matrix can be obtained. Due to the long length and large size of LDPC codes, this class of code is mostly defined by parity check matrices without a full rank i.e the H matrix has parity check rows that are not independent. This makes finding the encoder complicated due to the use of complex procedures like Gaussian eliminations being employed [20, 125, 136, 137]. Without knowing the rank of the H matrix, important encoding properties such as the rate of the code cannot be determined [15].

An efficient way of calculating the rank of a H matrix for a QC-LDPC code with the dimensions $(M \times N)$ is presented in [20] and is expressed as

$$rank(H) = 3^m - 2^m - \sum_{\hbar=\hbar_0+1}^{m-1} \binom{m}{\hbar} (2^\hbar - M), \tag{6.12}$$

where $\hbar_0$ represents the largest integer such that $2_0^\hbar \leq M$. The rate of the QC-LDPC code is then obtained as

$$R = \frac{(N-M) + (M - rank(H))}{N}, \tag{6.13}$$

which can be further reduced to

$$R = \frac{(N - rank(H))}{N}. \tag{6.14}$$

The systematic QC-LDPC construction of the H matrix proposed in Section 6.1.3 has the advantage of always having a full rank due to the presence of the identity submatrix. This means the rank of the QC-LDPC code is the always the same as the number of rows. Therefore, (6.12) does not need to be computed during the calculation of the rank. This makes it easy to determine the rate of the QC-LDPC code by directly applying (6.15)

$$R = \frac{(N-M)}{N} \tag{6.15}$$

This also makes for a more direct construction of the G matrix as shown in Section 6.1.4. This aids in the reduction of the complexity in the encoding process for the proposed systematic QC-LDPC code.

For purposes of giving a fair comparison, a rank analysis is performed on all the QC-LDPC codes used in the comparison tests so as to run simulations with the same rate.

6.1.5.2 BER Performance of the Proposed Systematic QC-LDPC code Compared to a High Performance code Construction

The construction of the base matrix of the QC-LDPC code are carried out in the field GF(32). The construction based on field partitions has a base matrix with the dimensions (8×24). Performing a q-ary matrix dispersion over GF(2) yields a regular quasi-cyclic H matrix with the dimensions $(248, 744)$ and a girth of at least 6 in the Tanner graph. After performing a rank analysis using (6.12), the rank of H is determined to be 171. This means that the H matrix for the QC-LDPC code for the construction based on field partitions has 77 dependant columns. We are now able to determine, using (6.14), that the regular QC-LDPC code has a G matrix with the dimensions (573×744) and a rate of 0.77. To achieve the same rate, the proposed construction has a base matrix set with the dimensions (6×26). Performing a q-ary matrix dispersion over GF(2) on the base matrix of the proposed construction yields a systematic QC-LDPC H matrix with the dimensions (186×806) and a girth of at least 6 in the Tanner graph. Due to the systematic structure, the proposed H matrix has a full rank. This means we can directly apply (6.15) to determine that the proposed construction yields a systematic irregular $(806, 620)$ QC-LDPC code with a rate of 0.77.

The codes are modulated using a BPSK modulation scheme and transmitted through an AWGN channel. The decoding algorithm used is the probability based sum product algorithm (SPA) [138] with maximum iterations set to 50. The equation used to compute the bit error probability in the AWGN channel is presented in (6.16) [45] [139].

$$P(error|BPSK) = \frac{1}{2}\text{erfc}\left(\sqrt{\frac{E_b}{N_0}}\right) \tag{6.16}$$

The results for these simulations can be seen in Fig. 6.1.

It can be seen from Fig. 6.1 that the proposed systematic construction matches the performance of the construction based on field partitions for low SNR values, while slightly performing better for higher SNR values.

An additional simulation for BER performance test was run using the less complex

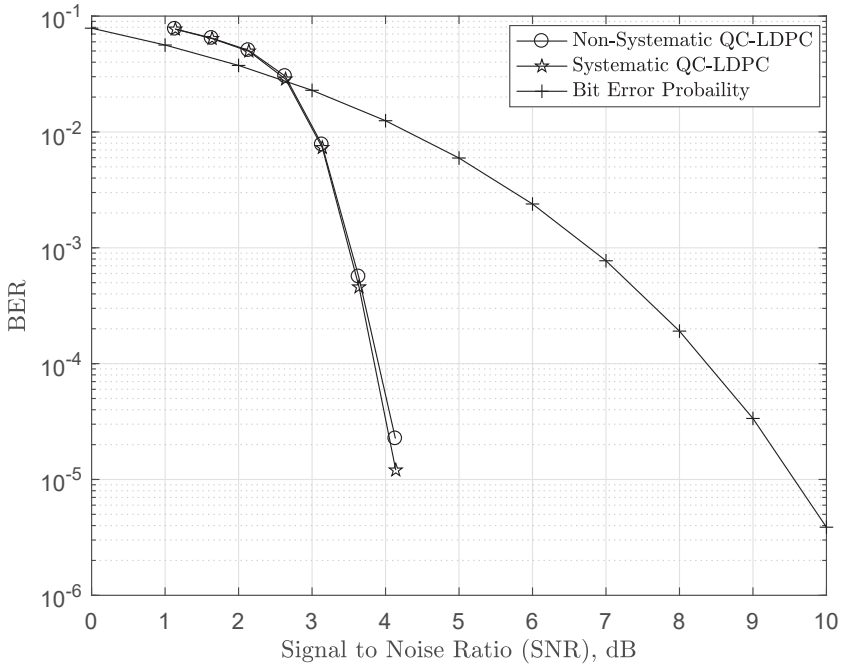


FIGURE 6.1: Performance comparison of the proposed systematic $(806, 620)$ QC-LDPC code against the non-systematic $(744, 573)$ QC-LDPC code using the SPA decoder.

log based Min Sum Algorithm (MSA) [11]. This is carried out to test the implementation and performance of a different decoder on the new construction. This simulations are run on a code rate of 0.9. For this rate to be achieved, in the field GF(32), a base matrix with the dimensions (3×29) is used for both QC-LDPC constructions. After performing a q-ary matrix dispersion over GF(2), the non systematic H matrix will have the dimensions (93×899) and a girth of at least 6 in the Tanner graph. The rank of the matrix is determined to be 83 after computing (6.12). Applying (6.14), we are able to ascertain that the (3×29) base matrix constructed using field partitions defines a regular $(899, 813)$ QC-LDPC code. Due to the base matrices having the same dimensions, the proposed construction will yield a systematic quasi-cyclic H matrix with the dimensions (93×899). The systematic H matrix has a full rank and a girth of at least 6 in the Tanner graph. After computing (6.15), the base matrix for the proposed construction is determined to define an irregular $(899, 806)$ QC-LDPC code.

The codes are again transmitted through an AWGN using a BPSK modulation

scheme. The results for this simulation can be seen in Fig 6.2.

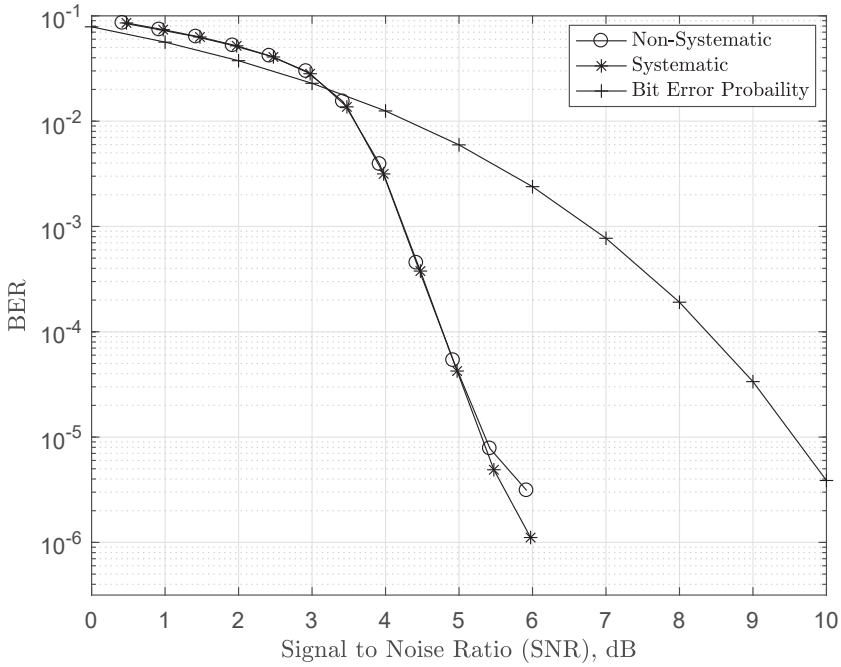


FIGURE 6.2: Performance comparison of the proposed systematic $(899, 806)$ QC-LDPC code against the non-systematic $(899, 813)$ QC-LDPC code using the MSA decoder.

From Fig 6.2, it can be seen that similar BER performance is obtained using both decoders. This is because the new construction matches the performance of the non-systematic QC-LDPC code for low SNR cases while exhibiting a slight gain for the higher SNR cases. It is important to note an error floor appears in BER curves in Fig. 6.2 due to the relatively short length of the LDPC codes used for the simulations [140] [141].

6.1.5.3 Complexity Analysis

The proposed construction does not only give the advantage of a systematic structure, but also offers a reduced computational complexity cost while constructing the encoder when compared to the technique used for general construction of the G matrix shown in [15]. This section focuses on the complexity analysis of the 2 constructions of the G matrix used for the simulations.

The G matrix of the proposed construction is dominated by a complexity of $\mathcal{O}(mp \times kp)$. This is because it involves the transposing of the parity submatrix and the addition of the identity submatrix to form the systematic G matrix. The high complexity cost of the QC-LDPC code based on the additive fields is fully experienced in the general process for the construction of the G matrix, which requires row reduction operations on each of the k row blocks of H. In [142] the overall complexity of the Gaussian techniques, for a rectangular matrix with the dimensions of $((n-k) \times n)$, is shown to be $\mathcal{O}((n-k)^2 \times n)$. Therefore, each of the k Gaussian row reductions required for the construction of the encoder yields a complexity of $\mathcal{O}((nb' - kb')^2 \times ((nb' - kb') + 1))$. Additional row reductions are also performed to obtain the submatrix Q [15], because the parity check matrix does not have a full rank. This yields a similar complexity of $\mathcal{O}((nb' - kb')^2 \times ((nb' - kb') + 1))$.

The complexity analysis for the two constructions of the G matrix are summarized in Table 6.1. Construction 1 is used to describe the proposed systematic construction while Construction 2 is used to describe the construction based on additive fields.

TABLE 6.1: Summary of the complexity in the encoder construction.

Construction	Encoding Computation	Time Complexity
Construction 1	Transposing	$\mathcal{O}((nb' - kb') \times kb')$
Construction 2	k row reductions for the row blocks of H	$\mathcal{O}((nb' - kb')^2 \times ((nb' - kb') + 1))$
	Additional row reductions to obtain submatrix Q	$\mathcal{O}((nb' - kb')^2 \times ((nb' - kb') + 1))$

It can be seen from Table 6.1 that the proposed construction exhibits less complexity in the construction of the G matrix due to the fact that its most complex procedure is the transposing of the parity submatrix H. Construction 2 however requires several row reduction procedures. These row reduction operations can make the Construction of the G matrix quite complex when dealing with larger H matrices.

6.1.6 Performance Analysis with QC-LDPC codes used in Practical Applications

To further test the performance of the systematic construction, simulations are run against full rank QC-LDPC codes used in WiMAX applications. The QC-LDPC codes used in the section are the IEEE802.11-2012 codes and the ITU-T G.9960 codes. These QC-LDPC codes are defined by expansion matrices and not base matrices. The expansion matrix is a matrix that maps the indexes where the cyclic shifts starts for each square circulant submatrix that makes up the QC-LDPC H matrix [124]. The expansion matrices for these codes are obtained from [124]. The simulation are run for different rates and lengths specific to the expansion matrices.

6.1.6.1 Performance against the IEEE802.16-2012 QC-LDPC codes

The first set of simulation are run for a IEEE802.11-2012 code with a rate of $\mathcal{R} = \frac{5}{6}$ and length of 648, $\mathcal{R} = \frac{3}{4}$ and length 1296. To obtain a near match in length and rate, the proposed construction is defined by a base matrix with the dimensions (4×21) created in the field GF(32). This base matrix is used to construct a binary systematic $(651, 527)$ QC-LDPC code. The results for these simulations can be seen in Fig. 6.3. Simulations are also run against the IEEE802.11-2012 code with a rates of $\mathcal{R} = \frac{3}{4}$ and length of 1296. The proposed construction uses a base matrix with the dimensions (5×20) constructed in the field GF(64) to obtain a near match in the length and rate of the code, by creating a binary $(1260, 945)$ QC-LDPC code with $\mathcal{R} = \frac{3}{4}$. The results for this simulations are presented in Fig. 6.4.

For the comparison test with the IEEE802.11-2012 QC-LDPC of rate $\mathcal{R} = \frac{5}{6}$ the Proposed systematic construction is seen in Fig. 6.3 to have a good error correction performance for the low SNR values. However, the code appears to suffer from an error floor as the SNR values increase. From Fig. 6.4 the proposed systematic

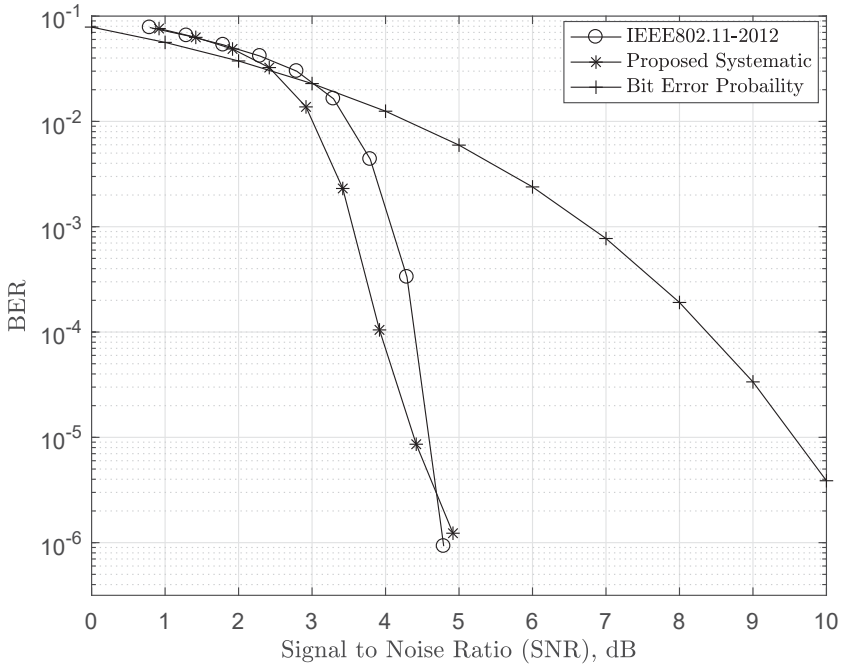


FIGURE 6.3: The proposed systematic $(651, 527)$ QC-LDPC code against the IEEE802.11-2012 $(648, 540)$ QC-LDPC code using the SPA decoder.

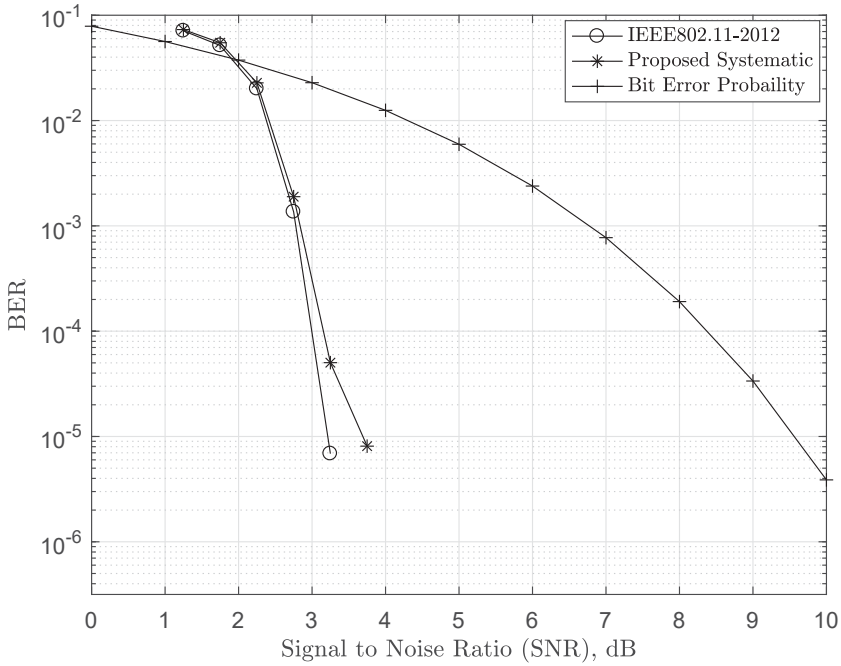


FIGURE 6.4: The proposed systematic $(1260, 945)$ QC-LDPC code against the IEEE802.11-2012 $(1296, 972)$ QC-LDPC code using the SPA decoder.

construction can be seen to yield a comparable performance to the IEEE802.11-2012 QC-LDPC code for the lower SNR values. As the SNR value increases, an error floor performance is once again exhibited by the proposed systematic construction. The presence of the error floor results in the IEEE802.11-2012 QC-LDPC code yielding a better error correction performance when compared to the propose systematic construction for both the $\mathcal{R} = \frac{5}{6}$ and $\mathcal{R} = \frac{3}{4}$. As stated in Section 6.1.5.2, The error floors are attributed to working with a relatively short LDPC code length. Short length LDPC code have a large presence of stopping sets and trapping sets [140] which contribute to error floors. Techniques similar to the work carried out in [141, 143, 144] can be explored in future research, so as to reduce the presence of trapping sets in the proposed irregular QC-LDPC code. Also, masking techniques [106, 111, 145, 146] can be explored to maintain the systematic structure of the base matrix while reducing the trapping sets and stopping sets in the expanded H matrix.

6.1.6.2 Performance against the ITU-T G.9960 QC-LDPC codes

The proposed systematic construction is also tested against ITU-T G.9960 QC-LDPC code with $\mathcal{R} = \frac{5}{6}$ and a length of 1152. To obtain a near match in rate and length of the ITU-T G.9960 code, a base matrix of dimension (3×18) constructed in the field GF(64) is used to create a binary systematic $(1134, 945)$ QC-LDPC code. Results for these simulations can be found in Fig. 6.5. Simulations are also run on the longer ITU-T G.9960 of length 1440 and $\mathcal{R} = \frac{2}{3}$. The proposed construction utilises a base matrix of dimensions (8×23) to create a binary systematic $(1449, 966)$ QC-LDPC code. The results for these simulations can be seen in Fig. 6.6.

From Fig. 6.5, the BER performance is comparable to that of the ITU-T G.9960 for the lower SNR regions but suffers from error floors again in the higher SNR areas. However, the proposed construction performs better with the longer length of 1449 as seen in Fig. 6.6. The error correction performance of the proposed

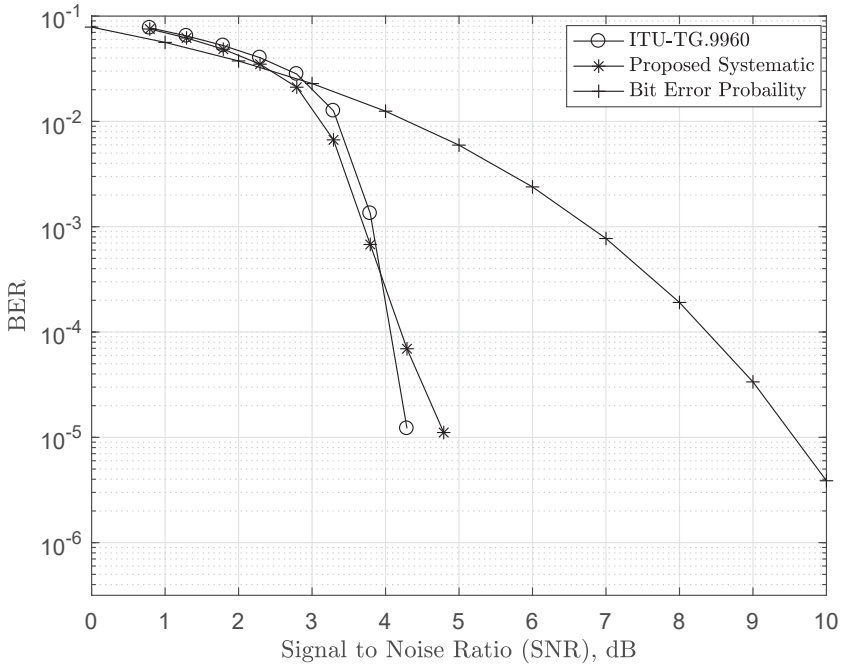


FIGURE 6.5: The proposed systematic $(1134, 945)$ QC-LDPC code against the ITU-T G.9960 $(1152, 960)$ QC-LDPC code using the SPA decoder.

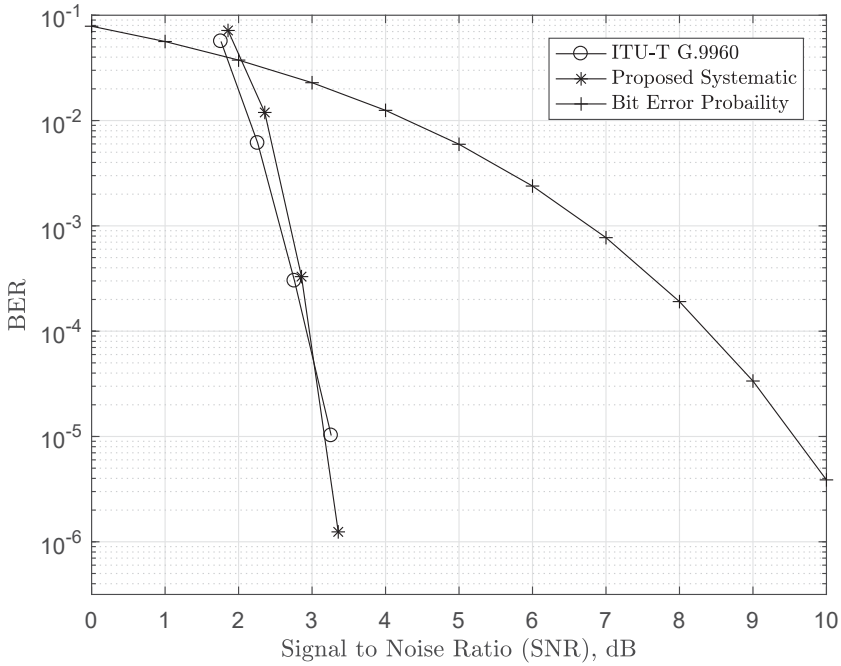


FIGURE 6.6: The proposed systematic $(1449, 966)$ QC-LDPC code against the ITU-T G.9960 $(1440, 960)$ QC-LDPC code using the SPA decoder.

systematic QC-LDPC code is similar to that of the ITU-T G.9960 code with the length 1440 and $\mathcal{R} = \frac{2}{3}$.

6.2 Decoding of QC-LDPC using the Decoding condition

The decoding condition is used alongside the probability based SPA algorithm during the decoding process. The pseudo code used in the decoding of the systematic sparse H matrix for the QC-LDPC code with the decoding condition is presented in Algorithm 5.

Algorithm 5: Implementation of the Decoding Condition on binary QC-LDPC codes

Input: The received vector, r, outputed from the channel
Output: The decoded vector $\widehat{r}$.
Initialize: Set the value f_{max}, Set the value of τ
Initialize: Setting up the soft-information: Bit probabilities are obtained as shown in (2.55)

repeat
- •*Decoding*:
  Soft bit information is fed into the SPA decoder.
- •*Decoding condition check*:
  Note the values of S_Q, where S_Q represents the number of syndrome check equations satisfied by each bit in the information set, indexed by the bits matching the parity submatrix.

  if $S_Q = \tau$ **then**
    compute (4.19)
  else
    $f = f + 1$
  end

until $f = f_{max}$

6.2.1 QC-LDPC codes Results

The simulations were run on a binary (135,30) QC-LDPC code. The bits were transmitted across an AWGN channel using a BPSK modulation scheme. For

these simulations the decoding condition is used alongside the SPA decoder with maximum iterations $f_{max} = 20$. The test are run against the SPA vs the SPA with the decoding condition to note any performance difference. The SPA has to meet the threshold, $\tau = 4$, before the decoding condition can be used to decode the received vector. This results can be seen in Fig 6.7 and Fig 6.8.

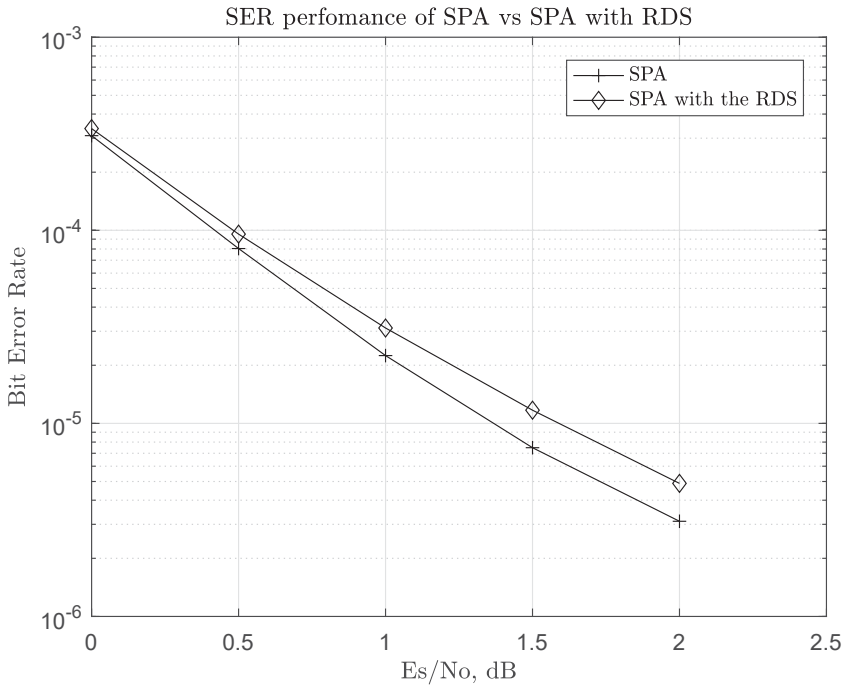


FIGURE 6.7: BER perfomance of SPA vs SPA implemented with the decoding condition defined by $\tau = 2$.

From Fig 6.7 and Fig 6.8 it can be seen that the SPA used with the decoding condition runs slightly less iterations when compared to the SPA alone, however this is at a reduced BER performance. The gain attained by the SPA algorithm when used alone is not greater than 0.5dB when compared to the SPA with the decoding condition. Similar error floors to those in Fig. 6.2 are seen in this simulation due the short LDPC code used.

6.3 Conclusion

A systematic QC-LDPC code construction is presented in this chapter. The systematic construction of the H matrix has the advantage of having a tolerable

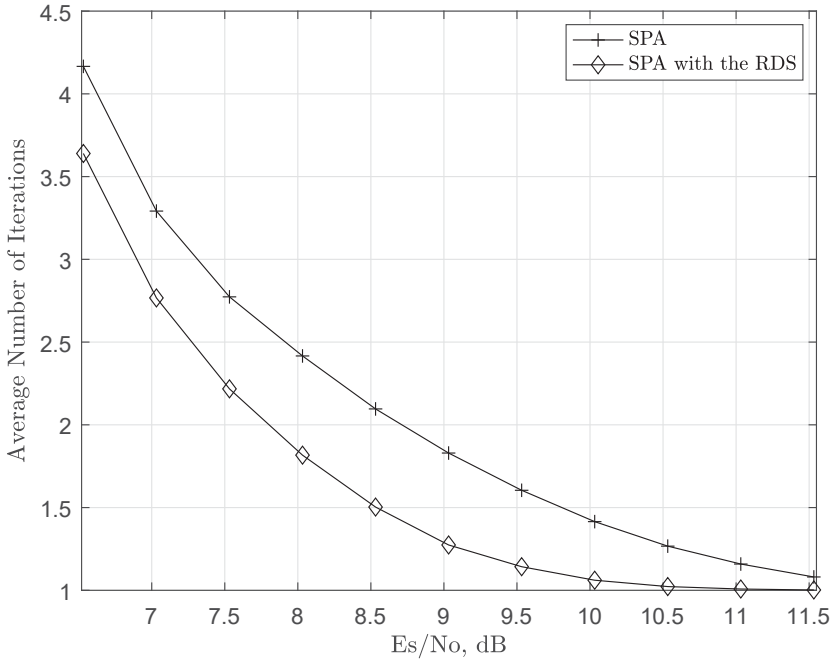


FIGURE 6.8: Iterative performance of SPA vs SPA implemented with the decoding condition defined by $\tau = 2$.

computational complexity cost while still having all the benefits of working with a systematic code. From the simulations run it is shown, that in addition to having a less complex encoder construction, the proposed systematic QC-LDPC code also yields a good BER performance. However, the code also appears to exhibit an error floor performance. This can be attributted to the fact relatively short QC-LDPC codes have been used during the simulations. Future research can utilise similar techniques as those presented in [141, 143, 144] to reduce the presence of the error floors

The use of the decoding condition, as an additional stopping criteria to the short QC-LDPC codes, is able to slightly reduce the iterative convergence rate of the SPA decoder without a significant loss in error correction performance. However, the implementation of the decoding condition on the binary QC-LDPC codes still requires more work in order to confidently justify its use with the SPA decoder whenever a performance-complexity tradeoff is required.

CHAPTER 7

Conclusion and Future Work

7.1 Conclusion

The "Complexity vs Performance" tradeoff off is common compromise made in the field of Forward Error Correction (FEC), especially in the selection of an appropriate decoding scheme. Therefore, the main objective of this research is the development of a decoding approach that utilises various FEC techniques that are able to work at a low complexity while maintaining a good error correction performance for codes transmitted through noisy channels. This goal is accomplished through the development of an iterative soft-input soft-output decoding algorithm that is implementable on both a bit and a symbol-level as presented in Chapter 4 and Chapter 5 respectively.

The bit-level implementation of the proposed decoder is able to take advantage of the sparsity of the parity check matrix presented at this level to efficiently decode the information set required for the decoding of the received vector. The algorithm works as a simpler form of a message passing algorithm that is able to asses if a bit is correct by using extrinsic information to apply votes to each bit. The algorithm is also able to improve on its iterative convergence rate by using an additional stopping criteria referred to as the decoding condition. The decoding condition is applied only when a set threshold, on the minimum number of syndrome check equations each bit in the information set should satisfy, is met. The threshold is a measure of confidence that the proposed decoder uses to determine if indeed the information is without error. The value of the threshold determines how quick the decoder converges to a codeword when decoding the received vector. The smaller

the threshold the quicker the algorithm converges to the decoded codeword. However, the performance of the algorithm worsens when a small threshold value is used. This is especially true for the half rate to lower rate codes. With higher rate codes, lower thresholds can be used because more bits participate in less syndrome check equations. This means chances of a bit satisfying a syndrome check equation a reduced. Therefore, if a bit satisfies a syndrome check equation for a high rate RS code, then there is a high probability the bit is correct.

A low complexity implementation of the proposed algorithm at a bit-level is also presented in Chapter 4. The low complexity yielded by this version of the proposed algorithm is largely due to the fact that the algorithm does not require iterative transformations of the bit-level parity check matrix during the decoding process. This variant of the algorithm is referred to as the no transform version of the proposed decoder.

The bit-level implementation of the proposed decoder is also shown to exhibit a good SER performance when compared to other high performance iterative bit-level RS decoders. The iterative bit-level decoders used in the SER test are the Adaptive Belief Propagation (ABP) algorithm and the bit-level implementation of the Parity check Transformation Algorithm (PTA). The performance testS were run for a nearly half rate $(15, 7)$ RS code and high rate $(15, 11)$ RS code. The bit-level implementation of the proposed decoder is able to yield a gain when compared to the algorithms used in the benchmark tests. The algorithm is able to run with a low iterative convergence rate which results in the algorithm running at a complexity lower than that of the PTA and the ABP algorithm. The performance of the no transform version of the proposed decoder for the $(15, 7)$ RS code in terms of SER is also very positive. The no transform version of the proposed algorithm is shown to match the high performance of the ABP algorithm with $\eta = 0.05$. For the high rate $(15, 11)$ RS code, the proposed algorithm matches the performance of both the bit-level PTA decoder. The difference in performance is seen in the complexity as the proposed algorithm is significantly less complex than the bit-level PTA with $\delta = 0.01$ and the ABP with $\eta = 0.05$. The performance of the no transform version of the proposed algorithm is maintained for the high rate

case as the algorithm matches the SER performance of the ABP with $\eta = 0.05$. The no transform version of the algorithm runs at a lower complexity than all of the algorithms used in the benchmark simulations which would justify the use of the algorithm without the need for a "Complexity vs Performance" tradeoff when compared to the ABP. Based on the complexity analysis, SER and iterative convergence rate simulations run in Chapter 4, the bit-level implementation for the proposed decoding approach and the no transform version of the proposed decoding approach met the objective of this research.

The symbol-level implementation of the proposed decoding algorithm is presented in Chapter 5. The limitations presented with obtaining the correct information set is addressed with addition of low weight parity check equations in the decoding process. The parity check equations added to the H matrix have the lowest possible weight that can be achieved at the symbol-level which corresponds minimum distance of the (n, k) RS code. The parity check equations added through a technique referred to as the parity check matrix extension. Each parity check extension matches the size of the original H matrix. The parity check matrix extensions are able to add some sparsity to the matrix H_ε which enables efficient exchange of information during the voting stage of the decoding process. The decoding condition is also used as an additional stopping criteria for the proposed decoding algorithm. The implementation of the threshold used by the decoding condition is slightly different from the bit-level case. The threshold for the symbol-level case is defined as the minimum number of syndrome check equations satisfied in the syndrome vector S. Once the specified number of syndrome check equations are satisfied, the decoding condition assumes the information set is correct and decodes the remaining the $(n-k)$ symbols. Similar to the bit-level case, the threshold is used as a measure of confidence by the decoding condition if the information set correct. Smaller values of the threshold have a better iterative performance but this may affect the error correction performance of the proposed symbol-level decoder. The performance of the symbol-level implementation of the proposed algorithm is benchmarked against the symbol-level ABP algorithm and the symbol-level implementation of the PTA. The simulations are run for a nearly half rate $(15, 7)$

RS code and a high rate $(15, 11)$ RS code. For the nearly half rate $(15, 7)$ RS code, the symbol-level implementation of the proposed algorithm is able to match the performance of the symbol-level ABP algorithm while outperforming the the symbol-level PTA by over 1dB. The algorithm is able to achieve this performance while being significantly less complex than all algorithms used in the benchmark tests as shown in the complexity analysis. For the high rate $(15, 11)$ RS code, the symbol-level implementation of the proposed decoder is slightly outperformed by the symbol-level ABP by about 0.2dB. The symbol-level yields a better SER performance when compared to the symbol-level PTA. The proposed algorithm at a symbol-level yields a gain of 1.3dB when compared to the symbol-level PTA. This SER performance of the symbol-level implementation of the algorithm is once again achieved while running at a significantly lower complexity when compared to the other algorithm used in the benchmark simulations. Based on the SER performance and the complexity analysis presented in Chapter 5, the symbol-level implementation developed for the proposed decoding approach meet the objective of this research.

Another objective of this research is to develop a decoding approach that can improve the iterative performance of existing decoders. Work carried out to meet this objective is performed in Chapter 4 and Chapter 5 for RS codes. The decoding condition is applied as additional stopping condition for the bit-level PTA, PTAe and the bit-level ABP algorithm. The decoding condition can be applied to these iterative decoder due to the row rank inverse properties of the H matrix of RS codes. The decoding condition is able to improve the iterative convergence rate of the these decoders. The improved iterative convergence rate does not came at the expense of a reduced SER performance. For the bit-level implementations of the PTA and the ABP, the use of the decoding condition is able to maintain the SER performance while running at less iterations. For the symbol-level implementation of the PTAe, the use of the decoding condition is able to maintain the SER performance of the algorithm while reducing the number of iterations for the case of the $(15, 7)$ RS code. For the $(15, 11)$ RS code, the use of the decoding condition with the PTAe once again matches the performance while significantly reducing

the number of iterations required to decode the received vector. From Chapter 4 and 5, it can be seen that the iterative convergence rate of the ABP and PTA decoders are improved, by the addition of the decoding condition, without an any significant loss in error correction performance. This increases the efficiency of these iterative soft-decision decoders. The use of the decoding condition as an additional stopping criteria also meets the main objective of this research which is to reduce complexity while maintaining error correction performance of the iterative decoding algorithms.

A systematic construction for QC-LDPC codes is also proposed in this research. This was one of the objectives for this research. This was largely in part due to the importance of the systematic structure for the implementation of the decoding condition. The systematic construction for the QC-LDC code is presented in chapter 6 and has the benefit of having a less complex G matrix construction. The BER performance of the systematic QC-LDPC code is also presented in this chapter. The construction is shown to perform favourably when compared to a high performance non full rank QC-LDPC construction and full rank constructions used in WiMAX applications. However the construction experiences error floors for some code rates and code lengths. More work needs to be done to reduce on this in order to better improve the performance of the overall decoding process. In Chapter 6, the decoding condition is also implemented on Belief Propagation (BP) algorithm. A comparable BER performance is exhibited by the SPA with the decoding condition to that of the original implementation of the SPA. However, the iterative performance is only slightly improved.

7.2 Future Work

Work in this research focused on the development of a high performance decoding approach that runs at a relatively low complexity. The decoding approach work well, however improvements can still be made. For the symbol-level case, a more

elegant way to obtain the low weight parity check equations used in the extensions can be devised. There are several further potential techniques that can be investigated in the selection of the index vectors used to rearrange the matrices h_x such as the use of reprocessing and preprocessing techniques. This would help in the selection of a more diverse set of vectors used in obtaining the parity check matrix extensions. This will help improve the performance and efficiency of the proposed algorithm without a significant increase in complexity.

Research can also be carried out to test the coding gain the bit-level implementation of the proposed algorithm can attain from using low weight parity check equations instead of the original H matrix as in the case for the parity check extensions. To further improve the error correction performance of the proposed bit-level decoder, at the expense of an increased computational complexity cost, double decoding techniques can also be explored. The use of a hard-decision decoder, in a similar way to the implementation with soft-decision decoders presented in [6, 69, 91], can also be investigated when applied to the proposed decoding approach.

For the LDPC implementation, a better implementation of the decoding condition is still lacking. The current method does little to improve on the iterative performance of the algorithm. Research carried out to improve this is sill required. Also techniques similar to work carried out in [141, 143, 144] should be explored with the aim of reducing the error floor performance of the proposed systematic QC-LDPC code. Also Masking techniques [106, 111, 145, 146] can also be studied to yield a low error floors while decoding. This would aid in significantly improving the error correction performance of the of the proposed construction.

References

[1] T. K. Moon, *"Error Correction Coding: Mathematical Methods and Algorithms"*. John Wiley & Sons, 2005, no. ISBN 0-471-64800-0.

[2] I. S. Reed and G. Solomon, "Polynomial Codes over Certain Finite Fields," *J. Soc.Ind. Appl. Maths.*, 1960.

[3] D. MacKey and R. M. Neil, "Near Shannon Limit Performance of Low Density Parity Check Codes," *Electro. Lett*, 1996.

[4] J. Massey, "Shift-Register Synthesis and BCH Decoding," *Information Theory, IEEE Transactions on*, vol. 15, no. 1, pp. 122–127, Jan 1969.

[5] S. H. Y. Sugiyama, M. Kasahara and T. Namekawa, "A Method for Solving Key Equation for Decoding Goppa Codes," *Information and Control*, 1975.

[6] J. Jiang and K. R. Narayanan, "Iterative Soft-Input Soft-Output Decoding of Reed–Solomon Codes by Adapting the Parity-Check Matrix," *IEEE Transactions on Information Theory*, vol. 52, no. 8, pp. 3746–3756, Aug 2006.

[7] R. Koetter and A. Vardy, "Algebraic Soft-Decision Decoding of Reed-Solomon Codes," *Information Theory, IEEE Transactions on*, 2003.

[8] O. Ogundile, Y. Genga, and D. Versfeld, "Symbol Level Iterative Soft Decision Decoder for Reed-Solomon Codes Based on Parity-Check Equations," *Electronics Letters*, vol. 51, no. 17, pp. 1332–1333, Aug. 2015.

[9] E. Prange, "The Use of Information Sets in Decoding Cyclic Codes," *IRE Transactions on Information Theory*, vol. 8, no. 5, pp. 5–9, September 1962.

[10] J. T. Coffey and R. M. Goodman, "The Complexity of Information Set Decoding," *IEEE Transactions on Information Theory*, vol. 36, no. 5, pp. 1031–1037, Sep 1990.

[11] S. Lin and W. Ryan, *"Channel Codes Classical and Modern"*. Cambridge University Press, 2009, no. ISBN-13 978-0-511-64182-4.

[12] L. Chen, "Iterative Soft Decoding of Reed-Solomon Convolutional Concatenated Codes," *IEEE Transactions on Communications*, vol. 61, no. 10, pp. 4076–4085, October 2013.

[13] Y. Genga and D. Versfeld, "The Modified Soft Input Parity Check Transformation Algorithm for Reed Solomon Codes," *SAIEE African Research Jornal*, vol. 108, no. 1, pp. 24–30, Mar. 2017.

[14] R. Gallager, "Low Density Parity Check Codes," *IRE Transactions on Information*, vol. IT-8, no. 1, pp. 21–28, Jan 1962.

[15] Z. Li, L. Chen, L. Zeng, S. Lin, and W. H. Fong, "Efficient Encoding of Quasi-Cyclic Low-Density Parity-Check Codes," *IEEE Transactions on Communications*, vol. 54, no. 1, pp. 71–81, Jan 2006.

[16] M. Bossert, *Channel Coding for Telecommuincations*. John Wiley & Sons, 1999, no. ISBN 0-471-98277-6.

[17] S. Lin and D. J. Costello, *Error Control Coding*, 2nd ed. Pearson Prentice Hall, 2004, no. ISBN 0-13-042672-5.

[18] R. Singleton, "Maximum Distance Q-Nary Codes," *IEEE Transactions on Information Theory*, vol. 10, no. 2, pp. 116–118, Apr 1964.

[19] M. Davey and D. Mackay, "Low-Density Parity Check Code over GF(q)," *IEEE Communications Letters*, vol. 2, no. 6, pp. 165–167, June 1998.

[20] K. Liu, Q. Huang, S. Lin, and K. Abdel-Ghaffar, "Quasi-cyclic LDPC Codes: Construction and Rank Analysis of Their Parity-Check Matrices," in *Information Theory and Applications Workshop (ITA), 2012*, Feb 2012, pp. 227–233.

[21] F. MacWilliams and N. Sloane, *"The Theory of Error-Correcting Codes"*, 8th ed. Elsevier Science Publishers, June 1993, vol. 16, no. ISBN 0 444 85193 3, pp. 106.

[22] A. Vardy and Y. Be'ery, "Bit-Level Soft-Decision Decoding of Reed-Solomon Codes," *Communications, IEEE Transactions on*, vol. 39, no. 3, pp. 440–444, Mar 1991.

[23] O. Ur-rehman and N. Zivic, "Soft Decision Iterative Error and Erasure Decoder for Reed-Solomon Codes," *Communications, IET*, vol. 8, no. 16, pp. 2863–2870, 2014.

[24] V. T. Van, S. Mita, J. Li, C. Yuen, and Y. L. Guan, "Bit-Level Soft-Decision Decoding of Triple-Parity Reed-Solomon Codes Through Automorphism Groups," *Communications Letters, IEEE*, vol. 17, no. 3, pp. 553–556, March 2013.

[25] H. Lee, J. Wu, C. Wang, and Y. Ueng, "An Iterative Soft-Decision Decoding Algorithm for Reed-Solomon Codes," in *2017 IEEE International Symposium on Information Theory (ISIT)*, June 2017, pp. 2775–2779.

[26] X. Huang and L. Chen, "Iterative Multistage Soft Decoding of Multilevel Reed-Solomon Codes," in *2018 IEEE Information Theory Workshop (ITW)*, Nov 2018, pp. 1–5.

[27] Q. Guo, T. Johansson, E. Mårtensson, and P. Stankovski, "Information Set Decoding with Soft Information and Some Cryptographic Applications," in *2017 IEEE International Symposium on Information Theory (ISIT)*, June 2017, pp. 1793–1797.

[28] M. Fossorier and S. Lin, "Reliability-Based Information Set Decoding of Binary Linear Codes," in *Proceedings. 1998 IEEE International Symposium on Information Theory (Cat. No.98CH36252)*, Aug 1998, p. 229.

[29] L. Zhang, Z. Zhang, X. Wang, C. Zhong, and L. Ping, "Simplified Successive-Cancellation Decoding Using Information Set Reselection for Polar Codes

with Arbitrary Blocklength," *IET Communications*, vol. 9, no. 11, pp. 1380–1387, 2015.

[30] S. Scholl and N. Wehn, "Hardware Implementation of a Reed-Solomon Soft Decoder Based on Information Set Decoding," in *2014 Design, Automation Test in Europe Conference Exhibition (DATE)*, March 2014, pp. 1–6.

[31] C. Peters, "Information-Set Decoding for Linear Codes over F_q," in *Sendrier N. (eds) Post-Quantum Cryptography. PQCrypto 2010. Lecture Notes in Computer Science*, vol. 6061, 2010, pp. 81–84.

[32] G. G. de Oliveira Brante, D. Nascimento Muniz, and W. Godoy, "Information Set Based Soft-Decoding Algorithm for Block Codes," in *2010 IEEE Latin-American Conference on Communications*, Sep. 2010, pp. 1–6.

[33] E. R. Berlekamp, *Algebraic Coding Theory.* New York: McGraw-Hill, Inc., 1968.

[34] L. R. Welch and E. R. Berlekamp, "Error Correction for Algebraic Block Codes," *Patent US 4 633 470*, 1986.

[35] Sarah J. Johnson, *Iterative Error Correction: Turbo, Low-Density Parity-Check and Repeat–Accumulate Codes.* Cambridge University Press, 2010, vol. 1 pp 19-24, no. ISBN-13 978-0-511-69129-4.

[36] W. Zhang, X. Zhang, and H. Wang, "Increasing the Energy Efficiency of WSNs Using Algebraic Soft-Decision Reed-Solomon Decoders," in *2012 IEEE Asia Pacific Conference on Circuits and Systems*, Dec 2012, pp. 49–52.

[37] Q. Chen, Z. Wang, and J. Ma, "FPGA Implementation of an Interpolation Processor for Soft-Decision Decoding of Reed-Solomon Codes," in *2007 IEEE International Symposium on Circuits and Systems*, May 2007, pp. 2100–2103.

[38] H. Wang, W. Zhang, and Y. Liu, "Novel Pipelined Interpolator for Low-Complexity Chase Soft-Decision Reed-Solomon Decoder," in *2011 IEEE International Conference of Electron Devices and Solid-State Circuits*, Nov 2011, pp. 1–2.

[39] B. Liu, Y. Xie, L. Yang, and J. Yuan, "An Iterative Soft-Decision Decoding Algorithm with Dynamic Saturation for Short Reed-Solomon Codes," in *2018 IEEE Information Theory Workshop (ITW)*, Nov 2018, pp. 1–5.

[40] R. Lucas, M. Bossert, and M. Breitbach, "On Iterative Soft-Decision Decoding of Linear Binary Block Codes and Product Codes," *Selected Areas in Communications, IEEE Journal on*, vol. 16, no. 2, pp. 276–296, Feb 1998.

[41] O. O. Ogundile and D. J. J. Versfeld, "A Low Complexity Iterative Channel Estimation and Decoding Receiver Based on Reed-Solomon PTA," *IEEE Access*, vol. 4, pp. 8805–8813, 2016.

[42] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *"Introduction to Algorithms"*. The MIT Press, 2009, vol. 3^{rd} Edition, no. ISBN 978-0-262-03384-8.

[43] H. Mani and S. Hemati, "Symbol-Level Stochastic Chase Decoding of Reed-Solomon and BCH Codes," *IEEE Transactions on Communications*, vol. 67, no. 8, pp. 5241–5252, Aug 2019.

[44] Y. Cassuto, J. Bruck, and R. J. McEliece, "On the Average Complexity of Reed–Solomon List Decoders," *IEEE Transactions on Information Theory*, vol. 59, no. 4, pp. 2336–2351, April 2013.

[45] J. G. Proakis and M. Salehi, *Digital Communications*, 5th ed. McGraw-Hilll Higher Education, Nov 2008, no. ISBN 978-0-07-295716-7.

[46] O. O. Ogundile, O. O. Oyerinde, and D. J. J. Versfeld, "Decision Directed Iterative Channel Estimation and Reed–Solomon Decoding Over Flat Fading Channels," *IET Communications*, vol. 9, no. 17, pp. 2077–2084, 2015.

[47] K. SANKAR. (2012) Symbol Error rate for QAM (16, 64, 256,.., M-QAM). [Online]. Available: http://www.dsplog.com/2012/01/01/symbol-error-rate-16qam-64qam-256qam/

[48] J. Barry, E. Lee, and D. Messerschmitt, *"Digital Communication"*, 3rd ed. Springer, 2004, no. ISBN 13: 978-0-7923-7548-7.

[49] M. P. C. Fossorier and S. Lin, "Soft-Decision Decoding of Linear Block Codes Based on Ordered Statistics," *IEEE Transactions on Information Theory*, vol. 41, no. 5, pp. 1379–1396, Sep. 1995.

[50] I. Dumer, "Information-Set Soft-Decision Decoding," in *1998 Information Theory Workshop (Cat. No.98EX131)*, Jun 1998, pp. 77–78.

[51] E. N. Gilbert, "A Comparison of Signalling Alphabets," *The Bell System Technical Journal*, vol. 31, no. 3, pp. 504–522, May 1952.

[52] R. R. Varshamov, "Estimate of the Number of Signals in Error Correcting Codes," *Dokl. Acad. Nauk SSSR*, vol. 117, pp. 739 – 741, 1957.

[53] D. J. Barros, W. Godoy Jr, and E. C. wille, "A New Approach to the Information Set Decoding Algorithm," in *Computer Communications*, vol. 20, Jan. 1996, pp. 208–302.

[54] A. Ahmed, R. Koetter, and N. R. Shanbhag, "Performance Analysis of the Adaptive Parity Check Matrix Based Soft-Decision Decoding Algorithm," in *Conference Record of the Thirty-Eighth Asilomar Conference on Signals, Systems and Computers, 2004.*, vol. 2, Nov 2004, pp. 1995–1999 Vol.2.

[55] Xilinx LogiCORE IP Reed-Solomon Decoder. (March 2013). [Online]. Available: http://www.xilinx.com/products/intellectual-property/DO-DI-RSD.htm

[56] W. Sung and J. T. Coffey, "Information Set Decoding Complexity for Linear Codes in Bursty Channels with Side Information," in *Proceedings of 1995 IEEE International Symposium on Information Theory*, Sep. 1995, pp. 53–.

[57] A. Shiozaki, "Decoding of Redundant Residue Polynomial Codes Using Euclid's Algorithm," *IEEE Transactions on Information Theory*, vol. 34, no. 5, pp. 1351–1354, Sep. 1988.

[58] J. Yu and H. Loeliger, "Reverse Berlekamp-Massey Decoding," in *2013 IEEE International Symposium on Information Theory*, July 2013, pp. 1212–1216.

[59] J. Park, K. Lee, C. Choi, and H. Lee, "High-Speed Low-Complexity Reed-Solomon Decoder Using Pipelined Berlekamp-Massey Algorithm," in *2009 International SoC Design Conference (ISOCC)*, Nov 2009, pp. 452–455.

[60] J. Yu and H. Loeliger, "On Irreducible Polynomial Remainder Codes," in *2011 IEEE International Symposium on Information Theory Proceedings*, July 2011, pp. 1190–1194.

[61] M. Cheng, "Generalised Berlekamp-Massey Algorithm," *IEE Proceedings - Communications*, vol. 149, no. 4, pp. 207–210, Aug 2002.

[62] S. Srivastava, R. McSweeney, C. Spagnol, and E. Popovici, "Efficient Berlekamp-Massey Based Recursive Decoder for Reed-Solomon Codes," in *2012 28th International Conference on Microelectronics Proceedings*, May 2012, pp. 379–382.

[63] J. Yu and H. Loeliger, "Partial Inverses mod $m(x)$ and Reverse Berlekamp-Massey Decoding," *IEEE Transactions on Information Theory*, vol. 62, no. 12, pp. 6737–6756, Dec 2016.

[64] I. Ilani, "Berlekamp-Massey Algorithm: Euclid in Disguise," in *2018 IEEE International Conference on the Science of Electrical Engineering in Israel (ICSEE)*, Dec 2018, pp. 1–5.

[65] V. Guruswami and M. Sudan, "Improved Decoding of Reed-Solomon and Algebraic-Geometry Codes," *Information Theory, IEEE Transactions on*, 1999.

[66] L. Chen, R. Carrasco, and M. Johnston, "Reduced Complexity Interpolation for List Decoding Hermitian Codes," *IEEE Transactions on Wireless Communications*, vol. 7, no. 11, pp. 4353–4361, November 2008.

[67] X. Peng, W. Zhang, W. Ji, Z. Liang, and Y. Liu, "Reduced-Complexity Multiplicity Assignment Algorithm and Architecture for Low-Complexity Chase Decoder of Reed-Solomon Codes," *IEEE Communications Letters*, vol. 19, no. 11, pp. 1865–1868, Nov 2015.

[68] X. Li, W. Zhang, and Y. Liu, "Efficient Architecture for Algebraic Soft-Decision Decoding of Reed–Solomon Codes," *IET Communications*, vol. 9, no. 1, pp. 10–16, 2015.

[69] D. Chase, "Class of Algorithms for Decoding Block Codes with Channel Measurement Information," *IEEE Transactions on Information Theory*, vol. 18, no. 1, pp. 170–182, January 1972.

[70] C. Leroux, S. Hemati, S. Mannor, and W. J. Gross, "Stochastic Chase Decoding of Reed-Solomon Codes," *IEEE Communications Letters*, vol. 14, no. 9, pp. 863–865, Sep. 2010.

[71] L. Chen, S. Tang, and X. Ma, "Progressive Algebraic Soft-Decision Decoding of Reed-Solomon Codes," *IEEE Transactions on Communications*, vol. 61, no. 2, pp. 433–442, February 2013.

[72] J. Zhao, L. Chen, X. Ma, and M. Johnston, "Progressive Algebraic Chase Decoding Algorithms for Reed–Solomon Codes," *IET Communications*, vol. 10, no. 12, pp. 1416–1427, 2016.

[73] S. Chu, Y. Chen, Y. Chiu, and R. Chang, "Fast Chase Algorithms for Decoding Reed-Solomon Codes," in *2014 International Symposium on Next-Generation Electronics (ISNE)*, May 2014, pp. 1–3.

[74] J. Zhao, L. Chen, X. Ma, and M. Johnston, "Interpolation Based Progressive Algebraic Chase Decoding of Reed-Solomon Codes," in *2016 IEEE International Conference on Communications (ICC)*, May 2016, pp. 1–6.

[75] K. Murphy, *"Machine Learning: A Probabilistic Perpective"*. MIT press, 2012, vol. pp 35, no. ISBN-10: 0262018020.

[76] S. Marsland, *"Machine Learning: An Algorithmic Perspective"*. Chapman & Hall / CRC, October 2014, vol. 2^{nd} Edition, no. ISBN-10: 1420067184.

[77] J. J. Kosmach and S. B. Wicker, "Algebraic-Iterative Reed-Solomon Trellis Decoding," in *1999 IEEE 49th Vehicular Technology Conference (Cat. No.99CH36363)*, vol. 1, May 1999, pp. 269–273 vol.1.

[78] R. Kotter and A. Vardy, "Algebraic Soft-Decision Decoding of Reed-Solomon Codes," in *2000 IEEE International Symposium on Information Theory (Cat. No.00CH37060)*, June 2000, pp. 61–.

[79] W. Jin and M. Fossorier, "Towards Maximum Likelihood Soft Decision Decoding of the (255,239) Reed Solomon Code," *IEEE Transactions on Magnetics*, vol. 44, no. 3, pp. 423–428, March 2008.

[80] S. Lee and B. V. K. V. Kumar, "Soft-Decision Decoding of Reed-Solomon Codes Using Successive Error-and-Erasure Decoding," in *IEEE GLOBECOM 2008 - 2008 IEEE Global Telecommunications Conference*, Nov 2008, pp. 1–5.

[81] Y. Yang, M. Jiang, and X. Wu, "An Investigation in Iterative Decoding of Reed-Solomon codes Based on Adaptive Belief Propagation," in *2009 International Conference on Wireless Communications Signal Processing*, Nov 2009, pp. 1–5.

[82] B. Kamali and A. Aghvami, "Belief Propagation Decoding of Reed-Solomon Codes; a Bit-Level Soft Decision Decoding Algorithm," *Broadcasting, IEEE Transactions on*, vol. 51, no. 1, pp. 106–113, March 2005.

[83] U. Cheng and G. K. Huth, "Bounds on the Bit Error Probability of a Linear Cyclic Code Over GF(2^l) and its Extended Code," *IEEE Transactions on Information Theory*, vol. 34, no. 4, pp. 776–785, July 1988.

[84] S. K. Shin and P. Sweeney, "Soft Decision Decoding of Reed-Solomon Codes Using Trellis Methods," *IEE Proceedings - Communications*, vol. 141, no. 5, pp. 303–308, Oct 1994.

[85] E. Berlekamp, R. Peile, and S. Pope, "The Application of Error Control to Communications," *IEEE Communications Magazine*, vol. 25, no. 4, pp. 44–57, April 1987.

[86] S. Chung, G. D. Forney, T. J. Richardson, and R. Urbanke, "On the Design of Low-Density Parity-Check Codes within 0.0045 dB of the Shannon Limit," *IEEE Communications Letters*, vol. 5, no. 2, pp. 58–60, Feb 2001.

[87] D. J. C. MacKay, "Good Error-Correcting Codes Based on Very Sparse Matrices," *IEEE Transactions on Information Theory*, vol. 45, no. 2, pp. 399–431, Mar 1999.

[88] J. Bellorado, A. Kavcic, M. Marrow, and L. Ping, "Low-Complexity Soft-Decoding Algorithms for Reed–Solomon Codes-Part II: Soft-Input Soft-Output Iterative Decoding," *IEEE Transactions on Information Theory*, vol. 56, no. 3, pp. 960–967, March 2010.

[89] Y. Wu, H. Koetter, and C. Hadjicostis, "Soft-Decision Decoding of Linear Block Codes Using Preprocessing," in *International Symposium onInformation Theory, 2004. ISIT 2004. Proceedings.*, June 2004, pp. 259–.

[90] Y. Wu and C. N. Hadjicostis, "Soft-Decision Decoding of Linear Block Codes Using Preprocessing and Diversification," *IEEE Transactions on Information Theory*, vol. 53, no. 1, pp. 378–393, Jan 2007.

[91] M. El-Khamy and R. J. McEliece, "Iterative Algebraic Soft-Decision List Decoding of Reed-Solomon Codes," *IEEE Journal on Selected Areas in Communications*, vol. 24, no. 3, pp. 481–490, March 2006.

[92] R. McEliece, "The Guruswami-Sudan Decoding Algorithm for Reed-Solomon Codes," IPN Progress Rep.,Tech Rep. 42 - 153, 2003.

[93] J. Bellorado and A. Kavcic, "Low-Complexity Soft-Decoding Algorithms for Reed-Solomon Codes-Part I: An Algebraic Soft-In Hard-Out Chase Decoder," *IEEE Transactions on Information Theory*, vol. 56, no. 3, pp. 945–959, March 2010.

[94] X. Wang and S. Tang, "Bit-Level Soft-Decision Decoding of Double and Triple-Parity Reed-Solomon Codes Through Binary Hamming Code Constraints," *IEEE Communications Letters*, vol. 19, no. 2, pp. 135–138, Feb 2015.

[95] A. I. V. Casado, M. Griot, and R. D. Wesel, "LDPC Decoders with Informed Dynamic Scheduling," *IEEE Transactions on Communications*, vol. 58, no. 12, pp. 3470–3479, December 2010.

[96] H. Lee, Y. Ueng, S. Yeh, and W. Weng, "Two Informed Dynamic Scheduling Strategies for Iterative LDPC Decoders," *IEEE Transactions on Communications*, vol. 61, no. 3, pp. 886–896, March 2013.

[97] H. Lee and Y. Ueng, "LDPC Decoding Scheduling for Faster Convergence and Lower Error Floor," *IEEE Transactions on Communications*, vol. 62, no. 9, pp. 3104–3113, Sep. 2014.

[98] H. Xia and J. R. Cruz, "Reliability-Based Reed-Solomon Decoding for Magnetic Recording Channels," *IEEE Transactions on Magnetics*, vol. 42, no. 10, pp. 2603–2605, Oct 2006.

[99] H. Xia and J. Cruz, "Performance of Reliability-Based Iterative Soft-Decision Reed-Solomon Decoding on Magnetic Recording Channels," *IEEE Transactions on Magnetics*, vol. 43, no. 7, pp. 3320–3323, July 2007.

[100] H. Tokushige, I. Hisadomi, and T. Kasami, "On Soft-decision Iterative Decoding Algorithms by Using Algebraic Decoding for RS Codes," in *Global Telecommunications Conference, 2002. GLOBECOM '02. IEEE*, vol. 2, Nov 2002, pp. 1042–1045 vol.2.

[101] O. Aitsab and R. Pyndiah, "Performance of Concatenated Reed-Solomon/Convolutional Codes with Iterative Decoding," in *GLOBECOM 97. IEEE Global Telecommunications Conference. Conference Record*, vol. 2, Nov 1997, pp. 934–938 vol.2.

[102] G. Forney, *Concatinated Codes*. M.I.T Press, 1966.

[103] R. Pyndiah, A. Glavieux, A. Picart, and S. Jacq, "Near Optimum Decoding of Product Codes," in *1994 IEEE GLOBECOM. Communications: The Global Bridge*, Nov 1994, pp. 339–343 vol.1.

[104] C. Berrou, A. Glavieux, and P. Thitimajshima, "Near Shannon Limit Error-Correcting Coding and Decoding: Turbo-Codes. 1," in *Proceedings of ICC '93 - IEEE International Conference on Communications*, vol. 2, May 1993, pp. 1064–1070 vol.2.

[105] R. Tanner, "A Recursive Approach to Low Complexity Codes," *IEEE Transactions on Information Theory*, vol. 27, no. 5, pp. 533–547, Sep 1981.

[106] L. Wang, X. Zhang, F. Yu, Y. Fu, and Y. Wang, "QC-LDPC Codes with Girth Eight Based on Independent Row-Column Mapping Sequence," *IEEE Communications Letters*, vol. 17, no. 11, pp. 2140–2143, November 2013.

[107] D. Lin, Q. Li, and S. Li, "Semi-Random Construction of Quasi-Cyclic LDPC Codes," in *Proceedings. 2005 International Conference on Communications, Circuits and Systems, 2005.*, vol. 1, May 2005, pp. 9–13 Vol. 1.

[108] L. Ghouti and A. M. Andalusi, "High-Capacity Colour Image Watermarking Using Multi-Dimensional Fourier Transforms and Semi-Random LDPC codes," in *IET Conference on Image Processing (IPR 2012)*, July 2012, pp. 1–5.

[109] B. Ghazi and E. Lee, "LP/SDP Hierarchy Lower Bounds for Decoding Random LDPC Codes," *IEEE Transactions on Information Theory*, vol. 64, no. 6, pp. 4423–4437, June 2018.

[110] A. Dehghan and A. H. Banihashemi, "On the Tanner Graph Cycle Distribution of Random LDPC, Random Protograph-Based LDPC, and Random Quasi-Cyclic LDPC Code Ensembles," *IEEE Transactions on Information Theory*, vol. 64, no. 6, pp. 4438–4451, June 2018.

[111] G. Han, Y. L. Guan, and L. Kong, "Construction of Irregular QC-LDPC Codes via Masking with ACE Optimization," *IEEE Communications Letters*, vol. 18, no. 2, pp. 348–351, February 2014.

[112] F. Chenghua and L. Xiuli, "Construction of QC-LDPC Codes on Combinatorial Sequences," in *2013 5th International Conference on Intelligent Human-Machine Systems and Cybernetics*, vol. 2, Aug 2013, pp. 74–78.

[113] Z. Fan, W. Zhang, X. Liu, and H. Cheng, "An Improved Algorithm for Constructing QC-LDPC Codes Based on the PEG Algorithm," in *2009 Fourth International Conference on Communications and Networking in China*, Aug 2009, pp. 1–4.

[114] D. Wang, L. Wang, X. Chen, C. Ju, Z. Wang, A. Fei, H. Wang, and Q. Zhang, "Irregular QC-LDPC Based Multi-Level Coded Modulation Scheme for the Next Generation Optical Communication Systems," in *2017 Opto-Electronics and Communications Conference (OECC) and Photonics Global Conference (PGC)*, July 2017, pp. 1–2.

[115] M. Asif, W. Zhou, J. S. Ally, N. A. Khan, and Z. u. A. Akhtar, "An Algebraic Construction of Quasi-Cyclic LDPC Codes Based on the Conjugates of Primitive Elements over Finite Fields," in *2018 IEEE 18th International Conference on Communication Technology (ICCT)*, Oct 2018, pp. 115–119.

[116] J. Li, K. Liu, S. Lin, and K. Abdel-Ghaffar, "Decoding of Quasi-Cyclic LDPC Codes with Section-Wise Cyclic Structure," in *Information Theory and Applications Workshop (ITA), 2014*, Feb 2014, pp. 1–10.

[117] L. Chen, I. Djurdjevic, and J. Xu, "Construction of Quasi-Cyclic LDPC codes Based on the Minimum Weight Codewords of Reed-Solomon Codes,"

in *International Symposium on Information Theory, 2004. ISIT 2004. Proceedings.*, June 2004, p. 239.

[118] Z. Fan, W. Zhang, X. Liu, and H. Cheng, "An Improved Algorithm for Constructing QC-LDPC codes based on the PEG algorithm," in *2009 Fourth International Conference on Communications and Networking in China*, Aug 2009, pp. 1–4.

[119] Xiao-Yu Hu, E. Eleftheriou, and D. Arnold, "Progressive Edge-Growth Tanner Graphs," in *GLOBECOM'01. IEEE Global Telecommunications Conference (Cat. No.01CH37270)*, vol. 2, Nov 2001, pp. 995–1001 vol.2.

[120] Xiao-Yu Hu, E. Eleftheriou, and D. M. Arnold, "Regular and Irregular Progressive Edge-Growth Tanner Graphs," *IEEE Transactions on Information Theory*, vol. 51, no. 1, pp. 386–398, Jan 2005.

[121] S. Kim and D. Shin, "Lowering Error Floors of Systematic LDPC Codes Using Data Shortening," *IEEE Communications Letters*, vol. 17, no. 12, pp. 2348–2351, December 2013.

[122] J. Ning and J. Yuan, "Design of Systematic LDPC Codes Using Density Evolution Based on Tripartite Graph," in *2008 Australian Communications Theory Workshop*, Jan 2008, pp. 150–155.

[123] L. Grosjean, L. K. Rasmussen, R. Thobaben, and M. Skoglund, "Systematic LDPC Convolutional Codes: Asymptotic and Finite-Length Anytime Properties," *IEEE Transactions on Communications*, vol. 62, no. 12, pp. 4165–4183, Dec 2014.

[124] D. Declercq, M. Fossorier, and E. Biglieri, *"Channel Coding: Theory, Algorithms and Applications"*. Elsevier, June 2014, no. ISBN-13: 978-0-08-101330-4.

[125] C. F. Tseng and J. H. Tarng, "Low-Complexity and Piecewise Systematic Encoding of Non-Full-Rank QC-LDPC Codes," *IEEE Communications Letters*, vol. 19, no. 6, pp. 897–900, June 2015.

[126] X. Wang and D. Xie, "The Systematic Form of a Class of Regular Structured LDPC Codes," in *2013 Fourth International Conference on Emerging Intelligent Data and Web Technologies*, Sep. 2013, pp. 648–651.

[127] D. Versfeld, J. Ridley, H. Ferreira, and A. Helberg, "On Systematic Generator Matrices for Reed-Solomon Codes," *Information Theory, IEEE Transactions on*, vol. 56, no. 6, pp. 2549–2550, June 2010.

[128] D. J. J. Versfeld, H. C. Ferreira, and A. S. J. Helberg, "Efficient Packet Erasure Decoding by Transforming the Systematic Generator Matrix of an RS code," in *2008 IEEE Information Theory Workshop*, May 2008, pp. 401–405.

[129] R. Lucas and M. Bossert and M. Breitbach, "On Iterative Soft-Decision Decoding of Linear Binary Block Codes and Product Codes," *IEEE Journal on Selected Areas in Communications*, vol. 16, no. 2, pp. 276–296, Feb 1998.

[130] Y. Genga, O. Ogundile, O. Oyerinde, and J. Versfeld, "A Low Complexity Encoder Construction for Systematic Quasi-Cyclic LDPC Codes," in *2017 IEEE AFRICON*, Sept 2017, pp. 167–170.

[131] R.W. Farebrother, *Linear Least Squares Computations*. Marcel Dekker, 1988, vol. 91 pp. 11-12, no. ISBN 978-0-8247-7661-9.

[132] S. Song, B. Zhou, S. Lin, and K. Abdel-Ghaffar, "A Unified Approach to the Construction of Binary and Nonbinary Quasi-Cyclic LDPC Codes Based on Finite Fields," *IEEE Transactions on Communications*, vol. 57, no. 1, pp. 84–93, January 2009.

[133] L. Zeng, L. Lan, Y. Y. Tai, S. Song, S. Lin, and K. Abdel-Ghaffar, "Transactions Papers-Constructions of Nonbinary Quasi-Cyclic LDPC Codes: A Finite Field Approach," *IEEE Transactions on Communications*, vol. 56, no. 4, pp. 545–554, April 2008.

[134] 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 Transactions on Information Theory*, vol. 53, no. 7, pp. 2429–2458, July 2007.

[135] L. R. Turner, "Inverse of the Vandermonde Matrix with Applications," Tech. Rep., 1966.

[136] P. Yang, C. Wang, and C. Chao, "Rank Analysis of Parity-Check Matrices for Quasi-Cyclic LDPC Codes," in *2018 IEEE International Symposium on Information Theory (ISIT)*, June 2018, pp. 491–495.

[137] E. M. Gabidulin and M. Bossert, "On the Rank of LDPC Matrices Constructed by Vandermonde Matrices and RS Codes," in *2006 IEEE International Symposium on Information Theory*, July 2006, pp. 861–865.

[138] D. MacKay and R. Neal, "Near Shannon Limit Performance of Low Density Parity Check Codes," *Electronics Letters*, vol. 33, no. 6, pp. 457–458, Mar 1997.

[139] K. SANKAR. (2007) Bit Error Rate (BER) for BPSK modulation. [Online]. Available: http://www.dsplog.com/2007/08/05/bit-error-probability-for-bpsk-modulation/

[140] X. Zheng, F. C. M. Lau, C. K. Tse, and Y. He, "Construction of Short-Length LDPC Codes with Low Error Floor," in *APCCAS 2008 - 2008 IEEE Asia Pacific Conference on Circuits and Systems*, Nov 2008, pp. 1818–1821.

[141] X. Zheng, F. C. M. Lau, and C. K. Tse, "Constructing Short-Length Irregular LDPC Codes with Low Error Floor," *IEEE Transactions on Communications*, vol. 58, no. 10, pp. 2823–2834, October 2010.

[142] Y. Han and C. Hartmann, "Efficent Maximum-Likelihood Soft-Descion Decoding of Linear Block Codes Using Algorithm," Syracuse University, Syracuse, New York, Tech. Rep. SU-CIS-92-10, December 1991.

[143] X. Zhang and S. Chen, "A Two-Stage Decoding Algorithm to Lower the Error-Floors for LDPC Codes," *IEEE Communications Letters*, vol. 19, no. 4, pp. 517–520, April 2015.

[144] W. Li, J. Lin, and Z. Wang, "An Efficient Post Processing Scheme to Lower the Error Floor of LDPC Decoders," in *2017 IEEE 17th International Conference on Communication Technology (ICCT)*, Oct 2017, pp. 122–126.

[145] J. Xu, L. Chen, I. Djurdjevic, S. Lin, and K. Abdel-Ghaffar, "Construction of Regular and Irregular LDPC Codes: Geometry Decomposition and Masking," *IEEE Transactions on Information Theory*, vol. 53, no. 1, pp. 121–134, Jan 2007.

[146] T. Yeh, W. Chen, and C. Lu, "A Construction of LDPC Codes with Low Error Floors," in *2013 International Conference on Advanced Technologies for Communications (ATC 2013)*, Oct 2013, pp. 266–270.