

https://dspace.org/

Threshold based multi-bit flipping decoding of binary LDPC codes

Masunda, Kennedy Tohwechipi Fudu

2017

https://hdl.handle.net/10539/24242

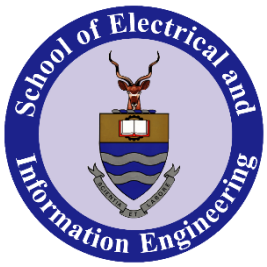
Masunda, Kennedy Tohwechipi Fudu (2017) Threshold based multi-bit flipping decoding of binary LDPC codes, University of the Witwatersrand, <https://hdl.handle.net/10539/24242>

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

School of Electrical and Information Engineering

University of the Witwatersrand

Private Bag 3, 2050, Johannesburg, South Africa



Threshold based multi-bit flipping decoding of binary LDPC codes.

Kennedy Masunda

Student Number: 514285

kennedy.masunda@students.wits.ac.za

Supervisor: Prof Fambirai Takawira

Submitted in fulfilment of the academic requirements for the Master of Science in Engineering in Electrical and Information Engineering degree in the School of Electrical and Information Engineering at the University of Witwatersrand, Johannesburg, South Africa.

August 2017

As the candidate's supervisor, I have approved this dissertation for submission.

Signed: _____

Date: _____

Name: Prof. Fambirai Takawira.

Declaration

I, KENNEDY. T. F. MASUNDA declare that

i. The research reported in this dissertation, except where otherwise indicated, and is my original work.

ii. This dissertation has not been submitted for any degree or examination at any other university.

iii. This dissertation does not contain other persons' data, pictures, graphs or other information, unless specifically acknowledged as being sourced from other persons.

iv. This dissertation does not contain other persons' writing, unless specifically acknowledged as being sourced from other researchers. Where other written sources have been quoted, then:

   a. their words have been re-written but the general information attributed to them has been referenced;

   b. where their exact words have been used, their writing has been placed inside quotation marks, and referenced.

v. Where I have reproduced a publication of which I am an author, co-author or editor, I have indicated in detail which part of the publication was actually written by myself alone and have fully referenced such publications.

vi. This dissertation does not contain text, graphics or tables copied and pasted from the Internet, unless specifically acknowledged, and the source being detailed in the dissertation and in the list of References sections.

Signed: _____

Date: _____

Name: Kennedy Masunda

Abstract

There has been a surge in the demand of high speed reliable communication infrastructure in the last few decades. Advanced technology, namely the internet has transformed the way people live and how they interact with their environment. The Internet of Things (IoT) has been a very big phenomenon and continues to transform infrastructure in the home and work place. All these developments are underpinned by the availability of cost-effective, reliable and error free communication services.

A perfect and reliable communication channel through which to transmit information does not exist. Telecommunication channels are often characterised by random noise and unpredictable disturbances that distort information or result in the loss of information. The need for reliable error-free communication has resulted in advanced research work in the field of Forward Error Correction (FEC).

Low density parity check (LDPC) codes, discovered by Gallager in 1963 provide excellent error correction performance which is close to the vaunted Shannon limit when used with long block codes and decoded with the sum-product algorithm (SPA). However, long block code lengths increase the decoding complexity exponentially and this problem is exacerbated by the intrinsic complexity of the SPA and its approximate derivatives. This makes it impossible for the SPA to be implemented in any practical communication device. Bit flipping LDPC decoders, whose error correction performance pales in comparison to the SPA have been devised to counter the disadvantages of the SPA. Even though, the bit flipping algorithms do not perform as well as the SPA, their exceeding low complexity makes them attractive for practical implementation in high speed communication devices. Thus, a lot of research has gone into the design and development of improved bit flipping algorithms.

This research work analyses and focusses on the design of improved multi-bit flipping algorithms which converge faster than single-bit flipping algorithms. The aim of the research is to devise methods with which to obtain thresholds that can be used to determine erroneous sections of a given codeword so that they can be corrected.

Two algorithms that use multi-thresholds are developed during the course of this research. The first algorithm uses multiple adaptive thresholds while the second algorithm uses multiple near optimal SNR dependant fixed thresholds to identify erroneous bits in a codeword. Both algorithms use soft information modification to further improve the decoding performance. Simulations show that the use of multiple adaptive or near optimal SNR dependant fixed

thresholds improves the bit error rate (BER) and frame error rate (FER) correcting performance and also decreases the average number of iterations (ANI) required for convergence.

The proposed algorithms are also investigated in terms of quantisation for practical applications in communication devices. Simulations show that the bit length of the quantizer as well as the quantization strategy (uniform or non-uniform quantization) is very important as it affects the decoding performance of the algorithms significantly.

Acknowledgements

First of all I would like to thank the Lord and my saviour Jesus Christ for seeing me through this very hectic but productive portion of my academic journey. I would also want to show deep appreciation and gratitude to my irreplaceable supervisor Professor Fambirai Takawira for being an unending source of invaluable guidance, information and support throughout the course of this research. There where so many brick walls and pitfalls but you assisted in navigating through them all.

Secondly I would like to show my gratitude to the Wits School of Electrical and Information Engineering for all the support and coffee. It was not going to be easy to navigate through this minefield without your companionship and camaraderie. I would also like to thank the Wits CeTAS, Prof Takawira and the Wits Financial Aid Office for sponsoring this research. It would not have been possible without these reliable sources of funds.

Last but not least, I would like to express my profound gratitude to my friends and family, particularly Mr James K Masunda for constantly encouraging me and supporting me through the good and bad times as this research progressed. All of you made this a bearable and worthwhile experience.

Many Thanks.

Preface

This dissertation is a compilation of the research work completed by Mr Kennedy T. F. Masunda under the supervision of Prof. Fambirai Takawira at the School of Electrical and Information Engineering at the University of the Witwatersrand. The work involves the study of multi-bit flipping algorithms for LDPC codes. The topic of discussion is the improvement of these algorithms by attempting to optimise the way in which flipping thresholds are determined.

Part of the results obtained in the research work have been submitted at the SATNAC 2016 conference held at George in South Africa. Some of the results will be submitted in an upcoming journal.

The dissertation in its entirety is a product of the author's work apart from the referenced material.

Table of Contents

Declaration iii
Abstract iv
Acknowledgements vi
Preface vii
Table of Contents viii
Table of Figures xi
List of Acronyms xv
Glossary of Nomenclature xvii
1. Introduction 1
1.1 Forward Error Correction 1
1.2 Research Motivation 3
1.3 Research Contributions 4
1.4 Overview of Dissertation 5
1.5 Publications 6
2. Low Density Parity Check Codes 7
2.1 Introduction 7
2.1.1 The Parity Check Matrix 7
2.1.2 The Tanner Graph 8
2.1.3 The Generator Matrix 9
2.1.4 Types of LDPC Codes 10
2.2 Encoding 14
2.2.1 Generator Matrix Encoder 14
2.2.2 Linear Time Encoding 15
2.3 Decoding 16
2.3.1 Belief Propagation 17
2.3.2 Approximate Belief Propagation 18
2.3.3 Bit flipping Algorithms 18
3. Bit Flipping decoding of LDPC Codes 20
3.1 Single Bit Flipping Algorithms 21
3.1.1 Gallager Bit Flipping Decoding Algorithm (GBF) (1962) 21
3.1.2 Weighted Bit Flipping Decoding Algorithm (WBF) (2001) 21
3.1.3 Bootstrapped Weighted Bit Flipping Decoding Algorithm (B-WBF) (2002) 22
3.1.4 Modified Weighted Bit Flipping Decoding Algorithm (M-WBF) (2004) 23
3.1.5 Improved Modified Weighted Bit Flipping Decoding Algorithm (IMWBF) (2005) ... 23
3.1.6 Reliability Ratio Based Weighted Bit Flipping Algorithm (RR-WBF) (2005) 24
3.1.7 Channel Independent Weighted Bit Flipping Decoding Algorithm (CI-WBF) (2012) 24

3.1.8 Modified Channel Independent Weighted Bit Flipping Decoding Algorithm (MCI-WBF) (2013) 25
3.1.9 An Iterative Bit Flipping based Decoding Algorithm (2015) 25
3.1.10 A Self-Normalized Weighted Bit-Flipping Decoding Algorithm (2016) 26
3.1.11 Hybrid iterative Gradient Descent Bit-Flipping algorithm (HGDBF) algorithm (2016) 27
3.1.12 Weighted Bit-Flipping Decoding for Product LDPC Codes (2016) 28
3.2 Multi-bit Flipping Algorithms 29
3.2.1 Gradient Descent Bit Flipping Algorithm (2008) 29
3.2.2 Candidate bit based bit-flipping decoding algorithm (CBBF) (2009) 31
3.2.3 Modified I-WBF (MBF) (2009) 32
3.2.4 Soft-Bit-Flipping (SBF) Decoder for Geometric LDPC Codes (2010) 33
3.2.5 Combined Modified Weighted Bit Flipping Algorithm (CM-WBF) (2012) 34
3.2.6 An Adaptive Weighted Multi-bit Flipping Algorithm (AWMBF) (2013) 34
3.2.7 Two-Staged Weighted Bit Flipping Decoding Algorithm (2015) 35
3.2.8 Mixed Modified Weighted Bit Flipping Decoding Algorithm (MM-WBF) (2015) 36
3.2.9 Noise-Aided Gradient Descent Bit-Flipping Decoder (2016) 37
3.3 Threshold Based decoding 37
3.4 Channel Information modification 38
4. Adaptive Multi-Threshold Multi-bit Flipping Algorithm (AMTMBF) 41
4.1 Introduction 41
4.2 Adaptive Threshold Scheme 42
4.2.1 Convergence comparison – Single bit flipping vs Adaptive threshold multi-bit flipping 43
4.2.2 Convergence comparison – Existing and proposed adaptive threshold techniques 44
4.2.3 Adaptive Multi Threshold 45
4.3 Channel Information Modification Scheme 49
4.3.1 The proposed scheme 50
4.3.2 Determining the enhancement value $|\boldsymbol{\delta}|$ 51
4.4 Proposed Decoding Algorithm 52
4.5 Simulation Results 54
4.6 Discussion 61
5. Near Optimal SNR Dependent Threshold Multi-bit Flipping Algorithm (NOSMBF) 63
5.1 Introduction 63
5.2 SNR Dependent Threshold Scheme 64
5.2.1 Primary Flipping Thresholds 65
5.2.2 Secondary Flipping Threshold 68
5.2.3 Distribution of the inversion function values in a Random codeword 72

5.3 Channel Information Modification Scheme 74
5.3.1 The proposed scheme 74
5.3.2 Determining the enhancement value $|\boldsymbol{\delta}|$ 75
5.4 Proposed Decoding Algorithm 76
5.5 Simulation Results 78
5.6 Discussion 85
6. Quantisation of Bit Flipping algorithms 87
6.1 Introduction 87
6.2 Simulation Results 93
6.2.1 Quantisation Schemes 93
6.2.2 Uniform Quantization 93
6.2.3 Non-Uniform Quantization 95
6.2.4 Comparative analysis of the AMTMBF with existing algorithms 98
6.2.5 Comparative analysis of the NOSMBF with existing algorithms 102
6.3 Discussion 105
7. Comparison of the AMTMBF to the NOSMBF 107
7.1 Introduction 107
7.2 Simulation Results 107
7.3 Discussion 111
8. Conclusion 113
8.1 Validity of results 113
8.2 Future Work 114
References 117

Table of Figures

Figure 2. 1 The Tanner Graph....8
Figure 2. 2 Repeat-Accumulate (RA), irregular Repeat-Accumulate (IRA) and extended irregular Repeat-Accumulate (eIRA) code encoders....14

Figure 4. 1: Average number of iterations (ANI) comparison between IMWBF and AWMBF....44
Figure 4. 2: Flipping threshold analysis for AWMBF at varying SNR....45
Figure 4. 3: Average number of iterations (ANI) comparison between AMTMBF and AWMBF....47
Figure 4. 4: Determining enhancement value |δ| for AMTMBF....51
Figure 4. 5: Enhancement values investigation....52
Figure 4. 6: AMTMBF BER analysis for different LDPC codes....54
Figure 4. 7: AMTMBF FER analysis for different LDPC codes....55
Figure 4. 8: AMTMBF FER analysis for different LDPC codes....56
Figure 4. 9: AMTMBF vs IMWBF BER performance analysis....57
Figure 4. 10: AMTMBF vs IMWBF FER performance analysis....57
Figure 4. 11: AMTMBF vs AWMBF BER performance analysis....58
Figure 4. 12: AMTMBF vs AMWBF FER performance analysis....58
Figure 4. 13: AMTMBF vs SBF BER performance analysis....59
Figure 4. 14: AMTMBF vs SBF FER performance analysis....59
Figure 4. 15: AMTMBF vs IMWBF vs AWMBF vs SBF BER performance analysis....60
Figure 4. 16: AMTMBF vs IMWBF vs AWMBF vs SBF FER performance analysis....60
Figure 4. 17: AMTMBF vs IMWBF vs AWMBF vs SBF ANI performance analysis....61

Figure 5. 1: Determining the flipping threshold at varying SNR values for PEGReg (1008, 504)....66
Figure 5. 2: Determining the flipping threshold at varying SNR values for PEGReg (504,252)....67
Figure 5. 3: Flipping thresholds for varying LDPC codes....67
Figure 5. 4: Determining the strengthening threshold for PEGReg (1008, 504)....68
Figure 5. 5: Determining the strengthening threshold for PEGReg (504, 252)....69

Figure 5. 6: Strengthening thresholds for varying LDPC codes 69
Figure 5. 7: Approximate flipping thresholds 71
Figure 5. 8: Approximate strengthening thresholds 71
Figure 5. 9: Flipping threshold analysis 73
Figure 5. 10: Strengthening threshold analysis 73
Figure 5. 11: Determining the optimal enhancement value |δ| 75
Figure 5. 12: Enhancement values investigation 76
Figure 5. 13: NOSMBF BER analysis for different LDPC codes 78
Figure 5. 14: NOSMBF FER analysis for different LDPC codes 79
Figure 5. 15: NOSMBF ANI analysis for different LDPC codes 80
Figure 5. 16: NOSMBF vs IMWBF BER performance analysis 81
Figure 5. 17: NOSMBF vs IMWBF FER performance analysis 81
Figure 5. 18: NOSMBF vs AWMBF BER performance analysis 82
Figure 5. 19: NOSMBF vs AWMBF FER performance analysis 82
Figure 5. 20: NOSMBF vs SBF BER performance analysis 83
Figure 5. 21: NOSMBF vs SBF FER performance analysis 83
Figure 5. 22: NOSMBF vs IMWBF vs AWMBF vs SBF ANI performance analysis 84
Figure 5. 23: NOSMBF vs IMWBF vs AWMBF vs SBF FER performance analysis 84
Figure 5. 24: NOSMBF vs IMWBF vs AWMBF vs SBF ANI performance analysis 85

Figure 6. 1: Quantization boundaries (qi), levels (ri), and regions (Ri) for a symmetric 2-bit quantizer 89
Figure 6. 2: AMTMBF BER uniform quantization 94
Figure 6. 3: AMTMBF FER uniform quantization 94
Figure 6. 4: NOSMBF BER uniform quantization 95
Figure 6. 5: NOSMBF FER uniform quantization 95
Figure 6. 6: AMTMBF BER non-uniform quantization 96
Figure 6. 7: AMTMBF FER uniform quantization 96
Figure 6. 8: NOSMBF BER non-uniform quantization 97
Figure 6. 9: NOSMBF FER non-uniform quantization 97
Figure 6. 10: 6 bit uniform quantization BER performance analysis – AMTMBF vs IMWBF vs SBF vs AWMBF 98

Figure 6. 11: 6 bit uniform quantization FER performance analysis – AMTMBF vs IMWBF vs SBF vs AWMBF 98
Figure 6. 12: 8 bit uniform quantization BER performance analysis – AMTMBF vs IMWBF vs SBF vs AWMBF 99
Figure 6. 13: 8 bit uniform quantization FER performance analysis – AMTMBF vs IMWBF vs SBF vs AWMBF 99
Figure 6. 14: 6 bit non-uniform quantization BER performance analysis – AMTMBF vs IMWBF vs SBF vs AWMBF 100
Figure 6. 15: 6 bit non-uniform quantization FER performance analysis – AMTMBF vs IMWBF vs SBF vs AWMBF 100
Figure 6. 16: 8 bit non-uniform quantization BER performance analysis – AMTMBF vs IMWBF vs SBF vs AWMBF 101
Figure 6. 17: 8 bit non-uniform quantization BER performance analysis – AMTMBF vs IMWBF vs SBF vs AWMBF 101
Figure 6. 18: 6 bit uniform quantization BER performance analysis – NOSMBF vs IMWBF vs SBF vs AWMBF 102
Figure 6. 19: 6 bit uniform quantization BER performance analysis – NOSMBF vs IMWBF vs SBF vs AWMBF 102
Figure 6. 20: 8 bit uniform quantization BER performance analysis – NOSMBF vs IMWBF vs SBF vs AWMBF 103
Figure 6. 21: 8 bit uniform quantization FER performance analysis – NOSMBF vs IMWBF vs SBF vs AWMBF 103
Figure 6. 22: 6 bit non-uniform quantization BER performance analysis – NOSMBF vs IMWBF vs SBF vs AWMBF 104
Figure 6. 23: 6 bit uniform quantization FER performance analysis – NOSMBF vs IMWBF vs SBF vs AWMBF 104
Figure 6. 24: 8 bit uniform quantization BER performance analysis – NOSMBF vs IMWBF vs SBF vs AWMBF 105
Figure 6. 25: 8 bit uniform quantization BER performance analysis – NOSMBF vs IMWBF vs SBF vs AWMBF 105

Figure 7. 1: Proposed algorithms BER performance analysis 107
Figure 7. 2: Proposed algorithms FER performance analysis 108

Figure 7. 3: Proposed algorithms ANI performance analysis.....108
Figure 7. 4: Proposed algorithms uniform quantization BER performance analysis 109
Figure 7. 5: Proposed algorithms uniform quantization FER performance analysis.....110
Figure 7. 6: Proposed algorithms non-uniform quantization BER performance analysis 110
Figure 7. 7: Proposed algorithms uniform quantization FER performance analysis.....111

Figure 8. 1: Verification of Simulated results using the GDBF and PEGReg (1008, 504)...113

List of Acronyms

1G:	First Generation
2G:	Second Generation
3G:	Third Generation
4G:	Fourth Generation
5G:	Fifth Generation
AMPS:	Advanced Mobile Phone System
AMTMBF:	Adaptive Multi-threshold Multi-bit Flipping
ANI:	Average Number of Iterations
AWGN:	Additive White Gaussian Noise
AWMBF:	Adaptive Weighted Multi-bit Flipping
BEC:	Binary Erasure Channel
BER:	Bit Error Rate
BF:	Bit Flipping
BP:	Belief Propagation
BPSK	Binary Phase Shift Keying
BSC:	Binary Symmetric Channel
B-WBF:	Bootstrapped - Weighted Bit Flipping
CDMA:	Code Division Multiple Access
CI-WBF:	Channel Independent - Weighted Bit Flipping
CM – WBF:	Combined Modified - Weighted Bit Flipping
DVB-S2:	Digital Video Broadcasting - Satellite - Second Generation
eIRA:	Extended Irregular Repeat Accumulate
ETACS:	European Total Access Communication Systems
FDMA:	Frequency Division Multiple Access
FEC:	Forward Error Channel
FER:	Frame Error Rate
FG – LDPC:	Finite Geometry - Low Density Parity Check Codes
FM:	Frequency Modulation
GBF:	Gallager Bit Flipping
GDBF:	Gradient Descent Bit Flipping
GSM:	Global System for Mobile Communications
HGDBF:	Hybrid Gradient Descent Bit Flipping

IMWBF:	Improved Modified Weighted Bit Flipping
IoT:	Internet of Things
IRA:	Irregular Repeat Accumulate
LDPC:	Low Density Parity Check Codes
LTE	Long Term Evolution
MBF:	Modified Weighted Bit Flipping
MCI-WBF:	Modified Channel Independent - Weighted Bit Flipping
MLD:	Maximum Likelihood Decoder
MM – WBF:	Mixed Modified - Weighted Bit Flipping
MSA:	Min Sum Algorithm
MWBF:	Modified Weighted Bit Flipping
NOSMBF:	Near Optimal SNR dependant threshold Multi-bit Flipping
RA:	Repeat Accumulate
RR-WBF:	Reliability Ratio - Based Weighted Bit Flipping
SBF:	Soft Bit Flipping
SNR:	Signal to Noise Ratio
SNWBF:	Self-Normalized Weighted Bit-Flipping
SPA:	Sum Product Algorithm
TDMA:	Time Division Multiple Access
WBF:	Weighted Bit Flipping

Glossary of Nomenclature

A^T	Transverse of matrix A
$\boldsymbol{c}$	Valid codeword
c_m	Check node m
c_n	Valid codeword bit n
$\boldsymbol{E}$	Inversion function values vector
$\boldsymbol{e}$	Error vector
E_n	Inversion function values vector element for codeword bit n
$\boldsymbol{G}$	Generator matrix
$\boldsymbol{H}$	Parity check matrix
$H(x)$	Hamming weight of vector x
$\boldsymbol{h_m}$	Parity check matrix row m
I_K	Size K identity matrix
$m \in N(n)$	Check nodes converging on variable node n
$n \in M(m)$	Variable nodes converging on check node m
$P(c_n=0 \mid \boldsymbol{y})$	Probability of codeword bit = 0 given a received soft information vector
$P(c_n=1 \mid \boldsymbol{y})$	Probability of codeword bit = 1 given a received soft information vector
$\boldsymbol{s}$	Syndrome vector
s_m	Syndrome bit m
$\boldsymbol{u}$	Message vector u
u_n	Message bit n
v_n	Variable node n
w_c	Column weight
w_r	Row weight
$\boldsymbol{y}$	Soft information vector
y_n	Soft information vector element n
$\boldsymbol{z}$	Received hard decision codeword
z_n	Hard decision codeword bit n
α	Weighting factor
γ	Threshold coefficients

e	Total number of errors

1. Introduction

1.1 Forward Error Correction

In information theory and telecommunications, Forward Error Correction (FEC) is a method used to regulate the transmission of errors in unreliable noisy channels. The first error correcting codes, the Hamming Codes where invented by Richard Hamming in the early 1950s [1].

There are several FEC methods but they all essentially function by adding redundant information to form what is called a codeword. This information, usually added to the message via an algorithm, is used to identify and correct errors. Systematic codewords are a result of redundant information being appended at the beginning or end of the message while the message itself remains unchanged. When the message itself is modified, as is the case with simple repetition code, the resulting codeword is non-systematic.

The two main types of Forward Error Correction codes in existence are linear block codes and convolutional codes. Linear block codes such as BCH, LDPC and Reed Solomon codes use fixed length symbol or bit patterns while convolutional codes are of arbitrary length.

Claude Shannon's acclaimed *A Mathematical Theory of Communication* introduces the channel coding theorem which states that reliable communication is possible in a noisy channel provided that the rate does not exceed that channel's capacity [2]. It also alludes to the existence of error-correcting codes that can reduce error probability to any desired level without specifying what the particular error codes are.

This research is carried out with low density parity check (LDPC) codes discovered by Robert Gallager [3] in the early 1960s but were forgotten for nearly thirty years until interest was rekindled by MacKay et al upon discovery that the performance of these codes approaches the Shannon limit. Binary symbols are encoded into fixed frame size codewords, modulated via the Binary Phase-shift keying (BPSK) scheme, transmitted through an additive white Gaussian noise (AWGN) channel and then decoded at the receiver via an LDPC decoder. This research work focusses on the decoding process with the aim to improve it and making it more efficient. This aim can be achieved by improving the design of LDPC decoders.

FEC has increasingly found application in recent communication standards which has resulted in massive improvements in communication. However, history shows that this has not always

been the case. First generation (1G) mobile networks were deployed in the early 1980s [4]. These were analog networks which made use of FDMA and FM modulation. 1G standards such as AMPS and ETACS have no documented FEC technologies incorporated [4].

As mobile wireless communication improved, second generation (2G) networks came to the forefront in the late 1980s to remove the capacity limitations that were prevalent in the 1G mobile system. 2G mobile systems used time division multiple access (TDMA) and code division multiple access (CDMA) digital technologies. Standards such as the Global System for Mobile Communications (GSM) used concatenated cyclic and convolution codes together with an interleaver to protect frames against burst errors, for forward error correction [5]. This made the 2G network more reliable than the precursor 1G network.

The evolution of 2G mobile networks resulted in 3G mobile networks. 3G networks are faster and can handle more complex multimedia services seamlessly. 3G has found applications in Video Conferencing, Telemedicine, Global Positioning Systems, Mobile TV and Video on demand services. Forward error correction in standards such as the Multi-Carrier Code Division Multiple Access (MC-CDMA) used in 3G use concatenated turbo and LDPC codes to eliminate transmission errors [6].

4G technology was developed to meet the shortcomings of 3G. It is estimated that the customer requirements will far outstrip the services available on 3G networks [7]. 4G has the ability to handle high volumes of multimedia data seamlessly at speeds up to 100 Mbps [7]. While 4G is a major improvement to 3G it is not a complete overhaul but a complement to 3G in which pre-existing technology is amalgamated with new to improve performance [7]. The physical layer of 4G systems in some cases, uses concatenated Reed-Solomon/LDPC codes for forward error correction [8].

The evolution of wireless mobile technology and the rediscovery of LDPC codes have seen the rise in popularity of LDPC codes as evidenced by its inclusion in major communication standards that have been described above. LDPC codes have also been included in the FEC schemes for Wi-Fi and Wi-Max standards such as the IEEE 802.16e [9].

The Digital Video Broadcasting - Satellite - Second Generation (DVB-S2) standard for video broadcasting applications uses a concatenation of BCH and LDPC codes. This second generation standard replaces the first DVB-S which used a Reed Solomon and Convolutional code concatenation [10].

1.2 Research Motivation

The world is rapidly becoming a global digital village due to rapid evolution of technology and with the advent of the Internet-of-Things (IoT), connectivity has become of paramount importance. That means the establishment of reliable and low-cost communication systems for mobile devices and microsystems on larger mechanical machinery. A major aspect of a reliable communication system is the requirement to transmit correct information which requires effective error correction schemes. A second important aspect of a reliable communication system is power consumption in the process of transmitting and receiving information. The overall complexity of encoding and decoding algorithms plays a pivotal role in the amount of energy consumed by a communications system. This means, the less complex the algorithm the more desirable it is. A third critical aspect is the speed at which information is transmitted and the less complex a decoder, the higher the throughput.

LDPC codes discovered by Gallager in 1960 [3] have some of the most effective error correcting capabilities rivalling even those of Reed-Solomon codes. The biggest drawback of using LDPC codes is that the most effective decoding algorithm, the Belief Propagation (BP) Sum-product algorithm (SPA) is too complex to implement practically and therefore not suitable for use in high-speed mobile devices.

Apart from the SPA and its approximations, there exists another set of LDPC decoders known as bit flipping (BF) decoders. Within the family of BF decoders, there is a subset of sub-optimal LDPC decoders which comprises of multi-bit flipping algorithms worthy of attention in the goal to devise high-speed, low cost communication devices. The family of bit flipping decoders generally performs worse than the family of BP-SPA algorithms but they offer the attractive quality of low complexity.

This dissertation focusses on the design and development of low-complexity low-cost multi-bit LDPC algorithms for the purpose of improving the existing decoders. The success of this research points to improved understanding of multi-bit flipping LDPC decoders which will improve future designs. The performance of the proposed decoder/algorithms will be based on the following metrics, the bit error rate (BER), frame error rate (FER),average number of iterations (ANI) which give an idea of complexity of the algorithms.

A study into the practical implementation of a real decoder is also important to investigate the effects of approximating algorithm values by quantizing the input the same way a practical decoder does.

1.3 Research Contributions

The research compilation has made the following contributions:

In Chapter 4, the Adaptive Multi-Threshold Multi-bit Flipping (AMTMBF) is proposed following the observation that there exists very few prominent algorithms with state-dependant adaptive threshold multi-bit flipping algorithm namely the AWMBF discussed in [11]. This chapter makes the following contributions in this regard:

1. The main contribution is the development of a scheme by which multiple decoding state based adaptive thresholds can be incorporated into a multi-bit flipping algorithm. This scheme makes use of the inversion function result vector as well as the syndrome to determine the appropriate threshold for bit flipping. Simulation of this scheme shows an improved performance over existing algorithms with similar characteristics.
2. The second contribution is the discovery of the role played by soft variable node modification in the decoding of LDPC codes. It can be observed through simulations that variable node (VN) soft information modification improves the decoding performance of LDPC decoders.

In Chapter 5, the Near Optimal SNR Dependent Threshold Multi-bit Flipping (NOSMBF) is proposed following the observation that, there is very little mention in literature on how optimal flipping thresholds for a multi-bit flipping algorithm can be obtained. In this chapter, the following contributions are made in this regard:

1. The main contribution is the development of SNR dependant threshold scheme for LDPC multi-bit flipping algorithms. This scheme differs from [12] in which a multi-bit flipping threshold scheme is devised based on the probability density function of the inversion function in which the thresholds obtained are constant throughout the whole SNR range. It is observed that the SNR dependant threshold scheme improves the performance of multi-bit flipping algorithms.
2. The second contribution is the discovery that there exists a linear relationship between the SNR and the near-optimal thresholds for LDPC multi-bit flipping algorithms. In [12], there is no established relationship between flipping parameters and channel state.
3. Finally, the research provides an empirical approach for determining the near optimal flipping thresholds while in [11] flipping thresholds are obtained from formulae and in [12] they are obtained from statistical analysis of bit reliability information.

In Chapter 6 is an analysis of the quantized versions of both algorithms. Both algorithms where simulated using 6 and 8 bit uniform and non-uniform quantization. The following research contributions where made:

1. The non-uniform quantization scheme provides better performance for adaptive and near-optimal threshold algorithms.
2. Adaptive thresholds are highly susceptible to both uniform and non-uniform quantization and therefore need larger word lengths to achieve performance close to the unquantized version.
3. While the near optimal thresholds are susceptible to uniform quantization, they perform very well with 8 bit non-uniform quantization.

In Chapter 7 is a comparison of both algorithms to determine the best performing threshold scheme. In this chapter, both quantized and unquantized versions where compared and the following research contribution was made:

1. The near optimal SNR dependant threshold scheme performs better than the adaptive threshold scheme in all respects.

1.4 Overview of Dissertation

This section of the document introduces the work while the rest of the dissertation is organised in this manner. The following chapter, Chapter 2 introduces the linear block codes known as Low Density Parity Check (LDPC) codes. Therein lies a full discussion of various types of LDPC codes including the encoding and the decoding process. Chapter 3 discusses the main focus of this research which is the bit-flipping decoding of LDPC codes. Several single and multi-bit flipping algorithms or decoders are discussed at length citing their strengths and weaknesses.

Chapter 4 is a proposal for a new multi-bit flipping LDPC algorithm known as the Adaptive Multi-Threshold Multi-bit Flipping (AMTMBF) algorithm. This algorithm uses multiple thresholds that adapt to the state of the algorithm to choose bits to flip or enhance while offering reduced complexity. The algorithm is analysed, simulated and the results are discussed.

Chapter 5 introduces a second multi-threshold multi-bit flipping algorithm known as the Near Optimal SNR Dependent Threshold Multi-bit Flipping Algorithm (NOSMBF). The research on this algorithm focusses on the determination of near optimal flipping thresholds based on

the codeword error patterns, channel noise and inversion function values. The algorithm is also simulated, analysed and the results are discussed.

Chapter 6 focusses on the issues surrounding the implementation of a practical decoder by introducing the concept of quantization. The quantized versions of the AMTMBF and the NOSMBF are simulated against their non-quantized versions. The results are analysed and discussed.

Chapter 7 is a comparison of the performance of both algorithms. Both the quantized and unquantized versions of the AMTMBF and the NOSMBF are compared to determine which algorithm performs the best. The results of this simulation are presented and discussed.

The last section concludes the research dissertation.

1.5 Publications

Certain parts of this dissertation have been presented at the following publications and conferences:

Kennedy Masunda, Fambirai Takawira, *"A Multi Adaptive Threshold Weighted Multi-bit Flipping Algorithm for LDPC codes"*, In proceedings of SATNAC 2016, George, South Africa, September 2016.

Kennedy Masunda, Fambirai Takawira, *"A Near Optimal SNR Dependant Threshold Weighted Multi-bit Flipping Algorithm for LDPC codes"*, To be submitted, 2017.

2. Low Density Parity Check Codes

2.1 Introduction

Low-density parity-check codes (LDPC) codes are a very important class of forward error-correction codes with the ability to perform close to the Shannon limit on various storage and data transmission channels. These codes can be implemented as both binary and non-binary codes but this research focusses only on binary LDPC codes. There are various constructions of LDPC codes such as Gallager Codes, MacKay Codes, Array Codes, Irregular Codes, Combinatorial Codes, Finite Geometry Codes and Repeat Accumulate Codes.

LDPC codes are generally expressed as a large two dimensional sparse matrix and it is this sparseness that gives rise to the codes' exceptional error correcting capabilities [13]. These codes can also be expressed in form of a bi-partite graph known as the Tanner Graph [14]. The parity-check matrix elements are mostly zeros whilst the non-zero elements are ones [15].

Messages to be transmitted through a channel can be encoded using a generator matrix derived from the parity-check matrix or by permuting the parity-check matrix in such a way that it can be used for more efficient almost linear-time encoding.

Decoding a received message can be achieved in many ways ranging from the very effective but highly complex belief propagation sum product algorithm to the less efficient but very low complexity hard decision bit flipping algorithm.

2.1.1 The Parity Check Matrix

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

A *Low Density Parity Check Code* is a special class of linear block codes in which a large sparse two dimensional $M \; x \; N$ parity check matrix $\boldsymbol{H}$ such as the (12, 6) matrix above is used. For LDPC code $\boldsymbol{C}$ with M rows and N columns, let $\boldsymbol{H}$ be the parity-check matrix. Subsequently, $\boldsymbol{h_1}, \boldsymbol{h_2}, \ldots, \boldsymbol{h_M}$ denote the individual rows of $\boldsymbol{H}$, where $\boldsymbol{h} = (h_{m,1}, h_{m,2}, \ldots, h_{m,N})$ for $1 \leq m \leq$

M. It is due to the sparsity of the matrices that the code has excellent error correcting capabilities. An arbitrary data stream is split into $\boldsymbol{u} = (u_1, u_2, \ldots, u_K)$ vectors of length K bits. The vector $\boldsymbol{u}$ is encoded using code $\boldsymbol{C}$ into a codeword $\boldsymbol{c} = (c_1, c_2, \ldots, c_N)$ N bits long in a systematic or non-systematic fashion in which $M = N - K$ redundant bits are added to $\boldsymbol{u}$, for error checking and correcting purposes. These extra M bits are referred to as parity check bits. The rate of the code is defined as $R = {}^{K}/_{N} = 1 - {}^{w_c}/_{w_r}$

Regular LDPC codes have a constant row weight w_r and column weight w_c for all rows and columns in the matrix respectively. The row and column weights are related in this manner $w_r = w_c(N/M)$ and $w_c << m$. However, irregular codes which are discussed in Section 2.1.4, have varying row and column weights throughout the matrix.

In the decoding phase, the validity of each received codeword is checked using the parity check matrix $\boldsymbol{H}$. Every bit is checked by the parity checksum in which it participates to ascertain $\boldsymbol{c_i h_{mi}} = 0$. This check tests the zero-parity check constraint. In the event that this check is not satisfied, it then follows that the given bit is most likely to be in error and the opposite is true. The equation $\boldsymbol{s} = \boldsymbol{cH^T}$ produces $\boldsymbol{s}$ which is known as the syndrome of the received vector. The received vector can be confirmed a codeword if and only if the syndrome $\boldsymbol{s} = 0$.

2.1.2 The Tanner Graph

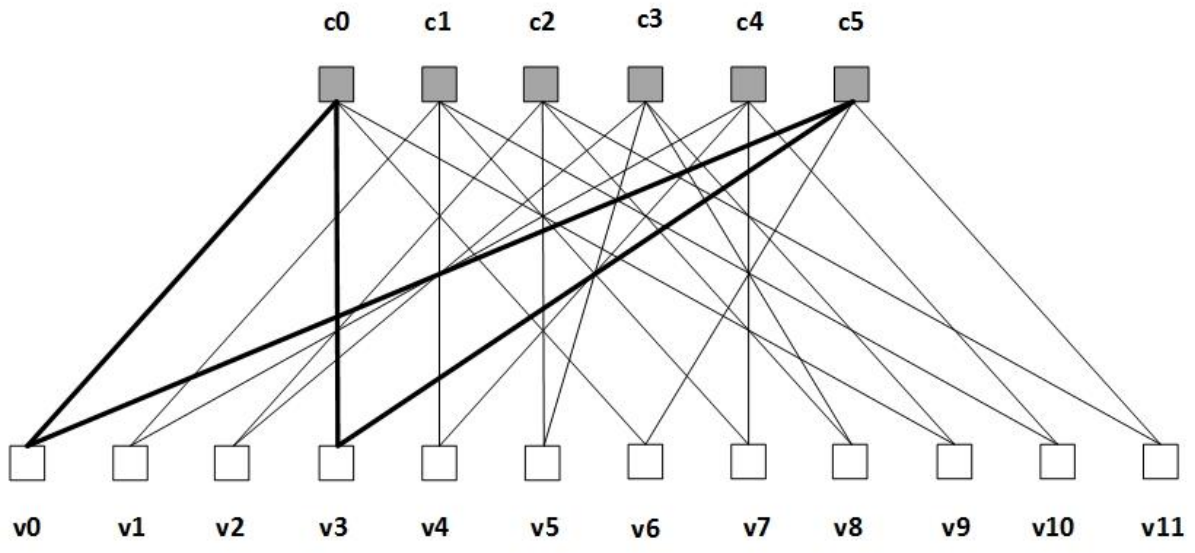


Figure 2. 1 The Tanner Graph

In 1981, Michael Tanner introduced the Tanner graph, shown in Figure 2.1, which is a bipartite graph that is useful for graphically representing parity-check matrixes [14]. The Tanner graph has two sets of nodes: check nodes, labelled c0 to c5 as well as variable nodes labelled v0 to v11. The number of check nodes and variable nodes is equivalent to the number of rows and columns of a given matrix respectively. The lines connecting the check nodes and variable nodes are known as edges.

The Tanner graph became such a critical tool in the study of LDPC codes because it facilitates the study of LDPC message passing decoders such as the sum-product algorithm [3]. It is also very useful when analysing the potential weaknesses of the parity-check matrix that can result in poor decoding performance. Figure 2.1 shows the Tanner graph for the matrix $\boldsymbol{H}$ in Section 2.1.1. The bold four edges connecting v0, c0, v3 and c5 form what is known as a short cycle which can also have six edges. Short cycles are not desirable because they degrade the decoding performance of a code. It is evident that short cycles are difficult to detect without the Tanner graph.

2.1.3 The Generator Matrix

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

The matrix $\boldsymbol{G}$ shown above, for the matrix described in Section 2.1.1, is referred to as the generator matrix which is used to generate codewords from a given bit message data tuple in a systematic or non-systematic manner. The matrix G produces systematic codewords which means, redundant parity check bits in this particular scenario are appended at end of the message data tuple [16]. The generator matrix is constructed by performing Gauss-Jordan elimination on matrix H so as to get

$$H = [A, I_{n-k}] \tag{2.1}$$

in which case $\boldsymbol{I_{n-k}}$ is an $(n-k)$ identity matrix whilst $\boldsymbol{A}$ is a $(n-k) \times k$ sub-matrix. It then follows that, the matrix G is obtained as

$$G = [I_k, A^T] \tag{2.2}$$

The process of encoding using the generator matrix is described in Section 2.2.1.

2.1.4 Types of LDPC Codes

The construction of LDPC codes is not merely random, there is a method in the way LDPC codes are constructed and different methods produce codes with different properties.

I. Gallager Codes

In 1962, when Robert Gallager discovered LDPC codes, he also developed a construction technique for pseudo-random codes that are now referred to as Gallager codes [3]. This is the original LDPC code with a regular H matrix made up of H_n submatrices that are structured in this manner

$$H = \begin{bmatrix} H_1 \\ H_2 \\ H_3 \\ \ddots \\ H_{w_c} \end{bmatrix} \tag{2.3}$$

Let w_c and β be two integers such that $w_c > 1$ and $\beta > 1$ such that the submatrix H_n with column weight and row weight 1 and w_r respectively, is of size $\beta \times \beta w_r$. The primary submatrix H_1 is in the form: for $j = 1,2,3, \ldots, \beta$ and all the 1's are in the j-th row in columns spanning from $1 + (j-1)w_r$ to jw_r. The remaining submatrices are column permutations of the primary submatrix. This code design is not guaranteed to be devoid of short cycles, however, these can be avoided by computer optimised design of the parity-check matrix. In [3] it is shown that these codes have remarkable distance properties given $w_c \geq 3$ and $w_r > w_c$. These codes give rise to low complexity encoders.

II. MacKay Codes

Gallager codes were further extended by MacKay who also learned independently the near capacity performance of these codes in [13]. A library of codes designed and developed by McKay which includes Progressive Edge Growth (PEG) codes can be found at [17]. MacKay developed algorithms to pseudo-randomly create sparse parity-check matrices and these are shown in [13]. The least complex of the algorithms involves, generating random weight w_c columns and w_r rows ensuring that no two columns overlap more than once. The more complex algorithms involve actively terminating short cycles and splitting the parity check as follows

$$H = [H_1 \quad H_2] \tag{2.4}$$

such that H_2 is either invertible or in the very least full rank.

As with Gallager codes, Mackay codes do not facilitate the development of low-complexity decoders [13]. The encoding process for these codes is mostly carried out via the generator matrix and it is explained in Section 2.2.1 why this is a computationally expensive process.

III. Array Codes

Array codes are a product of Fan, Eleftheriou and Olcer [18] who devised an LDPC parity-check matrix arrangement

$$H = \begin{bmatrix} I & I & I & \cdots & I & I & \cdots & I \\ 0 & I & \alpha & \cdots & \alpha^{m-2} & \alpha^{m-1} & \cdots & \alpha^{r-2} \\ 0 & 0 & I & \cdots & \alpha^{2(m-3)} & \alpha^{2(m-2)} & \cdots & \alpha^{2(r-3)} \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \cdots & \vdots \\ 0 & 0 & \cdots & 0 & I & \alpha^{m-1} & \cdots & \alpha^{(m-1)(r-m)} \end{bmatrix}$$

in which integers m and r are such that $m \leq p$ and $r \leq p$ where p is a prime number. I and O matrices are the size $p \times p$ identity and null matrices while the α matrix is a size $p \times p$ single right or left cyclic shift permutation matrix.

The major advantages of these codes is that they enable linear encoding [18] and they have very low error floors but have very specific code lengths and rates due to the design of H.

IV. Irregular Codes

In contrast to regular codes, irregular codes defined by Richardson et al [15] and Luby et al [19] have a non-constant variable and check node distributions. Irregular code ensembles are characterized by degree distribution polynomials

$$\lambda(x) = \lambda_1 x + \lambda_2 x^2 + \lambda_3 x^3 + \cdots + \lambda_{d_v} x^{d_v} \tag{2.1}$$

and

$$\rho(x) = \rho_{d_0} x^{d_0} + \rho_{d_0+1} x^{d_0+1} + \rho_{d_0+2} x^{d_0+2} + \cdots + \rho_{d_c} x^{d_c} \tag{2.2}$$

for variable and check nodes respectively where d_v is the maximum variable node degree, d_0 and d_c is the minimum and maximum check node degree for a given ensemble. The polynomials are optimised via density evolution [19]. Irregular codes have been proven to outperform regular codes for very long codes but exhibit worse performance for short to medium codes. This is due to the density evolution algorithm being optimised for codes length $N \rightarrow \infty$.

V. Combinatorial Codes

Given that LDPC codes can be constructed using random methods, researchers have produced Steiner and Kirkman triples based systems by successfully applying combinatorial mathematics to the design of optimal codes given some specific design constraints [20]. The problem can be expressed as a combinatorics problem of designing a size $(N - K) \times N$ $\boldsymbol{H}$ matrix with no short cycles of length four or six [20].

VI. Finite Geometry Codes

In [21] several codes have been designed using techniques based on finite geometries (FG-LDPC codes). These codes fall into a category of block codes referred to as quasi-cyclic or cyclic codes whose main advantage is apparent in the ease with which FG-LDPC decoders can be implemented via shift registers. Short length FG-LDPC codes generally outperform short length pseudo-random codes [21].

A major drawback of FG-LDPC codes is high row and column weight of the *H* matrices which compromises the sparsity of these matrices and therefore increases the iterative message passing decoder complexity. It is also not possible to construct FG-LDPC codes of arbitrary length or rate. This requires puncturing and code shortening [21].

VII. Repeat Accumulate Codes

In [22] codes with serial turbo and low density parity-check code features were proposed and these codes are referred to as *repeat-accumulate* (RA) codes because of the way in which these codes are encoded as shown in Figure 2.2. In the RA encoder, bits are repeated two or three times, permuted and then channelled through a differential encoder or accumulator. Due to the repetition, these codes are characteristically low rate but have been shown to perform very close to capacity [22].

In *Irregular repeat-accumulate* (IRA) codes the bits are repeated a variable number of times. In this set up, the generator matrix is added. These codes outperform RA codes and also facilitate higher rates [23]. The obvious drawback of IRA codes is that they are non-systematic and making them systematic greatly lowers the code rate [23].

Extended Irregular repeat-accumulate (eIRA) codes can be encoded in a more efficient manner giving rise to low and high rate codes which are also systematic [24].

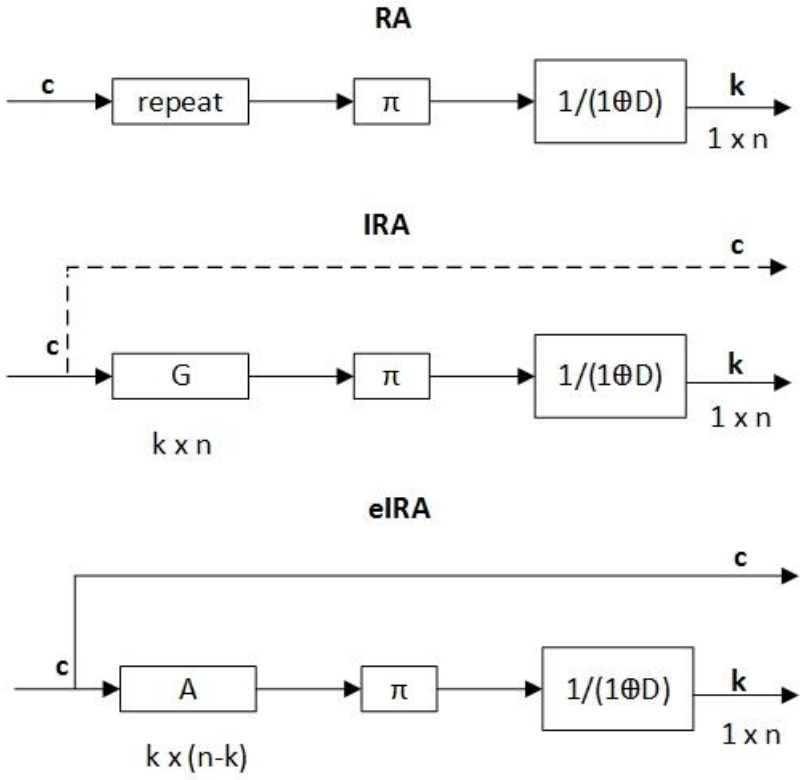


Figure 2. 2 Repeat-Accumulate (RA), irregular Repeat-Accumulate (IRA) and extended irregular Repeat-Accumulate (eIRA) code encoders

2.2 Encoding

A binary message block $\boldsymbol{u}$ of length K is encoded by adding redundant bits to produce codeword $\boldsymbol{c}$ of length N. There are two prominent methods of encoding of which the first methods involves the use of a generator matrix described in Section 2.1.4 while the second method involves the use of a permuted parity-check matrix to perform linear time encoding as described below. Both methods involve the use of a parity check matrix in one form or another.

2.2.1 Generator Matrix Encoder

The generator matrix $\boldsymbol{G}$ is first described in section 2.1.3 and is constructed by performing Gauss-Jordan elimination on a parity-check matrix $\boldsymbol{H}$ to obtain Equation 2.3.

$$\boldsymbol{H} = [\boldsymbol{A}, \boldsymbol{I_{N-K}}] \tag{2.3}$$

such that $\boldsymbol{I_{N-K}}$ is a size $(N-K)$ identity matrix and $\boldsymbol{A}$ is a $(N-K) \times K$ sub-matrix. The matrix $\boldsymbol{G}$ is obtained as shown in Equation 2.4.

$$\boldsymbol{G} = [\boldsymbol{I_K}, \boldsymbol{A^T}] \tag{2.4}$$

A valid generator matrix is such that

$$\boldsymbol{GH^T} = \boldsymbol{0} \tag{2.5}$$

because a valid generator matrix G has a row space orthogonal to the parity-check matrix $\boldsymbol{H}$ as shown in Equation 2.5. A given information vector $\boldsymbol{u}$ is multiplied to the generator matrix G to produce a codeword $\boldsymbol{c}$ in the manner shown in Equation 2.6.

$$\boldsymbol{c} = \boldsymbol{uG} \tag{2.6}$$

A valid codeword is also orthogonal to the parity-check matrix as shown in Equation 2.7.

$$\boldsymbol{cH^T} = \boldsymbol{0} \tag{2.7}$$

The generator matrix as opposed to the parity-check matrix is no longer guaranteed a sparse matrix and therefore the encoder for a length N codeword is of the order N^2 which makes the encoder prohibitively complex for very long codes. This problem is addressed by the use of the linear time encoding technique [15].

2.2.2 Linear Time Encoding

Linear time encoding was introduced in [15] by Richardson and Urbanke as a solution to the exceeding complexity associated with generator matrix encoding. In this approach, the parity-check matrix H undergoes row and column permutations that will result in an *approximate lower triangular form* parity-check matrix H_T

$$H_T = \begin{bmatrix} A & B & T \\ C & D & E \end{bmatrix} \tag{2.8}$$

The matrix T is in lower triangular form while the matrices C, D and E are known as the gap of H_T. Next step is to obtain Equation 2.9.

$$\begin{bmatrix} I & 0 \\ ET^{-1} & I \end{bmatrix} \begin{bmatrix} A & B & T \\ C & D & E \end{bmatrix} = \begin{bmatrix} A & B & T \\ -ET^{-1}A + C & -ET^{-1}B + D & 0 \end{bmatrix} \quad (2.9)$$

The term $-ET^{-1}A + C$ must be non-singular. The last step is to solve for $\boldsymbol{p_1}$ and $\boldsymbol{p_2}$ in

$$A\boldsymbol{u} + B\boldsymbol{p_1} + T\boldsymbol{p_2} = \mathbf{0} \quad (2.10)$$

and

$$(-ET^{-1}A + C)\boldsymbol{u} + (-ET^{-1}B + D)\boldsymbol{p_1} + T\boldsymbol{p_2} = \mathbf{0} \quad (2.11)$$

Such that a valid code word $\boldsymbol{c} = (\boldsymbol{u}, \boldsymbol{p_1}, \boldsymbol{p_2})$. If the sparseness of the submatrices is maintained, the encoder complexity can be kept significantly lower than generator matrix encoding.

2.3 Decoding

The maximum likelihood decoder (MLD) would be the most suitable decoder for LDPC codes and a thorough mathematical treatment of such a decoder has been presented by Gallager in his iconic PhD thesis [3]. However such a decoder is impossible to implement practically especially for the long codes required for LDPC codes to perform close to the Shannon limit. That being the case, Gallager went on to propose two major ways of iteratively decoding LDPC codes which are discussed below.

The following are a few terms to consider as they are valid preliminaries in almost all of the decoding scenarios for LDPC decoding algorithms. Suppose an LDPC code $\boldsymbol{C}$ is used for error control over a Binary Input - Additive White Gaussian Noise (BI-AWGN) with zero mean and power spectral density $N_0/2$. We assume that we are using binary-phase shift keying (BPSK) signalling with unit energy. A codeword $\boldsymbol{c} = (c_1, c_2, \dots, c_N)$ where $c_i \in (0,1)$ is mapped into a bipolar sequence $\boldsymbol{x} = (x_1, x_2, \dots, x_N)$ before its transmission where $x_i = (2c_i - 1)$ mapping $v_i \in (0,1)$ to $x_i \in (-1,1)$ (In some cases, $c_i \in (0,1)$ is mapped into $x_i \in (-1,1)$ via $x_i = (1 - 2c_i)$) where $1 \leq i \leq N$. Let $\boldsymbol{y} = (y_1, y_2, \dots, y_N)$ be the soft decision sequence received at the input of the receiver for $1 \leq i \leq N$, $y_i = x_i + n_i$ where n_i is a Gaussian random variable with zero mean and a variance of $N_0/2$.

2.3.1 Belief Propagation

The most famous decoding algorithm is the iterative *sum product algorithm* (SPA) which is also known as soft-decision decoding algorithm and makes use of received channel information and the parity check matrix to calculate the probabilities of a received bit being correct based on the information received from the channel via a process known as belief propagation. This is the best known LDPC decoder whose performance nears the Shannon limit for long codewords. Even though the SPA is a very effective decoder, it is very computationally complex due to the numerous divisions and multiplications needed per iteration. This makes the SPA very difficult to practically implement in hardware because multiplications and divisions are expensive to implement on logic circuits [25]. A marginally less complex version of the SPA has also been devised and is known as the log likelihood ratio version of the SPA. The functions of log-likelihood version of the SPA are shown below.

$$L(c_n|\boldsymbol{y}) = log\left(\frac{P(c=0|\boldsymbol{y})}{P(c=1|\boldsymbol{y})}\right) = \frac{2y}{\delta^2} \tag{2.12}$$

$$L(r_{nm}) = \prod_{n' \in V_m \backslash n} sign[L(q_{mn})]_{n'm} . \emptyset\left(\sum_{n' \in V_m \backslash n} \emptyset([|L(q_{nm})|]_{n'm})\right) \tag{2.13}$$

$$\emptyset(x) = -log[tanh(x/2)] = log\left(\frac{e^x + 1}{e^x - 1}\right) \tag{2.14}$$

$$L(q_{nm}) = L(c_i) + \sum_{m' \in C_n \backslash m} L(r_{m'n}) \tag{2.15}$$

The Equation 2.12 is the initialisation of the SPA where y is the received channel information while δ is the AWGN noise variation. c is the associated codeword bit, while the Equation 2.13 is the check node update equation with $L(r_{mn})$ as variable node to check node message for variable node n and check node m. Equation 2.15 is the variable node update and $L(q_{nm})$ is the check node to variable node message [3]. The term $\emptyset(x)$ in Equation 2.15 is described in Equation 2.14.

2.3.2 Approximate Belief Propagation

Due to the very complex nature of the SPA and its log likelihood version, several algorithms such as the Normalized BP-based Min-Sum algorithm (MSA) and the Offset BP-Based algorithm which are approximate versions of the SPA were proposed. These algorithms are not as complex to implement practically as the SPA but they have Bit Error Rate (BER) performance which is close to that of the SPA [26].

$$L(r_{mn}) = \prod_{n' \in V_m \backslash n} sign[L(q_{nm})]_{n'm} . min_{n' \in V_m \backslash n} |L(q_{nm})| \tag{2.16}$$

$$L(r_{mn}) = \prod_{n' \in V_m \backslash n} sign[L(q_{nm})]_{n'm} . \frac{min_{n' \in V_m \backslash n} |L(q_{nm})|}{\alpha} \tag{2.17}$$

$$L(r_{mn}) = \prod_{n' \in V_m \backslash n} sign[L(q_{nm})]_{n'm} . max\{min_{n' \in V_m \backslash n} |L(q_{nm})| - \beta, 0\} \tag{2.18}$$

The variable node update remains the same but the check node update does away with the computationally intensive Equation 2.13. Equation 2.16 is known as the Min Sum Algorithm (MSA). Equation 2.17 is the normalized MSA where α is the normalizing factor while Equation 2.18 is known as the offset MSA where β is referred to as the offset factor. While these approximate algorithms are notably less complex when compared to the SPA, they are still complex to a degree that discourages wide adaptation and therefore more attention has been shifted to another category of algorithms known as bit flipping (BF) decoders.

2.3.3 Bit flipping Algorithms

Bit flipping (BF) algorithms are a family of iterative algorithms that are used to perform hard decision iterative decoding of LDPC codes. They are derived from Gallager's original BF algorithm devised in the 1960s [3]. Certain algorithms in this family referred to as Single-bit flipping algorithms (SBF), flip one bit per iteration whilst some of them flip more than one bit per iteration and are thus referred to as Multiple/Multi-bit flipping algorithms (MBF). The decoding performance of the BF algorithms in terms of BER, FER and number of iterations is

inferior to that of the soft decision Sum-Product Algorithm (SPA). The SPA and its approximate versions are not used in many applications because of the complexity and therefore much attention has been given to sub-optimal but significantly less complex decoding algorithms based on the SPA such as the soft decision MSA algorithms [26] and several variations of the hard decision BF algorithm.

Almost all bit flipping algorithms follow Gallager's general BF algorithm which makes use of the syndrome vector as described in the following steps [3]:

Step 1: Compute the syndrome for the received codeword of size *N*. If all the parity-check equations are satisfied then decoding can stop otherwise proceed to step 2.

Step 2: For each bit in the codeword, compute the number of parity-check equations that have not been satisfied and denote for each bit as f_i , $i = 1,2, \dots, N$.

Step 3: Identify the set $\boldsymbol{\mu}$ of bits that have the largest f_i measured against some predefined threshold f_x. (One for SPF and a set of bits for MBF).

Step 4: Flip the bits in set $\boldsymbol{\mu}$.

Step 5: Repeat the steps 1 to 4 until all the parity check equations have been satisfied in which case we stop at Step 1, or until a predetermined maximum iteration number has been reached.

Most of the bit flipping algorithms discussed in literature follow Gallager's general BF algorithms as shown in the next chapter.

3. Bit Flipping decoding of LDPC Codes

Various parts of the original bit flipping algorithm discussed in Section 2.3.3 have been modified to come up with different versions of BF algorithms. Improvement in decoding performance of BF algorithms is obtained by incorporating certain soft decision aspects such as those of the SPA into the hard decision BF algorithm. This means that apart from hard decision decoding, the algorithms also make use of soft channel information to establish a metric by which the reliability of the hard decision is measured.

Consider $\boldsymbol{H}$ to be the parity-check matrix for LDPC code $\boldsymbol{C}$ with N columns and M rows. Let $\boldsymbol{h_1}, \boldsymbol{h_2}, \dots, \boldsymbol{h_M}$ represent rows of $\boldsymbol{H}$, in which $\boldsymbol{h_m} = (h_{m,1}, h_{m,2}, \dots, h_{m,N})$ for $1 \leq m \leq M$. It can then be shown that the received hard decision codeword $\boldsymbol{z}$ has the syndrome **s** where

$$\boldsymbol{s} = (s_1, s_2, \dots, s_M) = \boldsymbol{z}.\boldsymbol{H^T} \tag{3.1}$$

The syndrome component s_m is given by Equation 3.1

$$s_m = \boldsymbol{z}.\boldsymbol{h_m} = \sum_{l=1}^{N} z_l h_{m,l} \tag{3.2}$$

The vector $\boldsymbol{z}$ received is a codeword if and only if $\boldsymbol{s} = \boldsymbol{0}$. If $\boldsymbol{s} \neq \boldsymbol{0}$ then it means that errors have been detected. The check nodes joining on a variable bit n are signified by $\boldsymbol{M}(n) = \{m | h_{m,n} = 1\}$, whereas the set of variable nodes joining on a check m are noted as $\boldsymbol{N}(m) = \{n | h_{m,n} = 1\}$.

Let

$$e = (e_1, e_2, \dots, e_N) = (c_1, c_2, \dots, c_N) + (z_1, z_2, \dots, z_N) \tag{3.3}$$

The vector $\boldsymbol{e}$ is the error pattern that appears in $\boldsymbol{z}$ when compared with the corresponding valid codeword $\boldsymbol{c}$. It follows then that the error pattern found in vector $\boldsymbol{e}$ and the syndrome $\boldsymbol{s}$ are related as shown

$$\boldsymbol{s} = \boldsymbol{e}.\boldsymbol{H^T} \tag{3.4}$$

where

$$s_m = \boldsymbol{e}.\boldsymbol{h_m} = \sum_{n=1}^{N} e_n h_{m,n} \tag{3.5}$$

for $1 \leq m \leq M$.

3.1 Single Bit Flipping Algorithms

There are numerous variations of the SBF decoding algorithms documented. However, they all have a common feature, which is the use of the syndrome vector $\boldsymbol{s}$ to obtain a metric by which to flip a single perceived erroneous bit per single iteration. What differs in each algorithm is how the metric is obtained as explained below.

3.1.1 Gallager Bit Flipping Decoding Algorithm (GBF) (1962)

This is the original SBF decoding algorithm on which most of the algorithms are based. The algorithm is basically the exact one described in Section 2.3.3 [3].

3.1.2 Weighted Bit Flipping Decoding Algorithm (WBF) (2001)

In this algorithm [27] the hard decision vector $\boldsymbol{z}$ is obtained from the soft information vector $\boldsymbol{y}$ received from the communication channel. We define a reliability measure $|y_n|$ of the received symbol y_n. The larger $|y_n|$ is, the more reliable the hard decision is considered to be. From the parity check matrix $\boldsymbol{H}$, where $\boldsymbol{h_m} = (h_{m,1}, h_{m,2}, \dots, h_{m,N})$ for $1 \leq m \leq M$. We define

$$|y_n|_{min} \triangleq \{min\{|y_n|\}: 1 \leq n \leq N, h_{m,n} = 1\} \tag{3.6}$$

The metric, also known as the inversion function, with which to flip bits believed to be in error is calculated as follows

$$E_n \triangleq \sum_{m \in M(n)} (2s_m - 1)\,|y_n|_{min} \tag{3.7}$$

Where the set of variable nodes sums orthogonal to the syndrome bit s_m is $m \in M(n)$. The metric E_n is a weighted checksum orthogonal on the codeword bit positioned at n.

The general algorithm is modified as follows:

Step 2: Find the bit position n which has the largest E_n.

Step 3: Flip bit z_n in position n found in step 2.

3.1.3 Bootstrapped Weighted Bit Flipping Decoding Algorithm (B-WBF) (2002)

Consider the received vector $\boldsymbol{y}$. A value y_n is considered unreliable if $|y_n| < \alpha$, for a predetermined α which is optimized by simulation. A check node connected to an unreliable variable node is considered reliable if all the variable nodes adjacent to it are reliable. Unreliable y_n are replaced by improved values y'_n as shown in Equation 3.8. This new value is obtained from reliable check nodes which obtain information from reliable variable nodes [28].

$$y'_n = y_n + \sum_{m \in N(m)} \prod_{n \in M(n) \backslash n} sgn(y_n).min_{n \in M(n) \backslash n} |y_n| \tag{3.8}$$

The term $m \in N(m)$ denotes the reliable check nodes adjacent to variable node n. The term $n \in M(n) \backslash n$ represents all the variable nodes connected to check node m apart from n. All reliable variable nodes or unreliable variable nodes with unreliable check nodes remain unchanged such that $y'_n = y_n$.

When vector $\boldsymbol{y'}$ has been obtained, conventional WBF can commence as described in Section 3.1.2. The optimal value for α is dependent on the SNR and the code as well as its tanner graph. The BWBF shows a considerable BER improvement as well as reduced iterations when compared to the WBF [29]. The number of iterations is halved and a performance improvement of about 1.5 dB at a BER of 10^{-5}.

3.1.4 Modified Weighted Bit Flipping Decoding Algorithm (M-WBF) (2004)

The basic framework for this algorithm is similar to Section 3.1.2 apart from the calculation of the metric E_n which is calculated as follows:

$$E_n \triangleq \sum_{m \in M(n)} (2s_m - 1) \left| y_{n \in N(m)} \right|_{min} - \alpha |y_n| \tag{3.9}$$

The first term on the right indicates to what extend the erroneous bit should be flipped while the second term indicates to what extend the bit value should be maintained. The weighting factor α is obtained through simulation [30]. It is a real number $\alpha \geq 0$. The term α is SNR and code dependant. It is shown in [30] that the MWBF performs considerably better than the WBF at SNRs of particular interest. With an optimal α the M-WBF requires about half the iterations required for the standard WBF and offers about 0.5 dB improvement at a BER of 10^{-5}.

3.1.5 Improved Modified Weighted Bit Flipping Decoding Algorithm (IMWBF) (2005)

This algorithm is a slight modification of the algorithm in Section 3.1.4. The soft information from the bit whose metric is being calculated is removed from the metric computation such that in Equation 3.8 becomes:

$$\left| y_{n \in N(m)/n} \right|_{min} \triangleq \{ min\{|y_i|\} : 1 \leq i \leq N, h_{j,i} = 1, n \neq i \} \tag{3.10}$$

And (3.9) becomes:

$$E_n \triangleq \sum_{m \in M(n)} (2s_m - 1) \left| y_{n \in N(m)/n} \right|_{min} - \alpha |y_n| \tag{3.11}$$

Analysis of the weighting also shows that α decreases as the SNR increases and also increases as the column weight of the codes increases. However, α is unaffected by the length of the codes. In [31] it is also demonstrated how to optimise α.

Simulation BER curves show that the IMWBF with an optimised α value performs better than the MWBF for both Gallager and Finite Geometry codes [31].

3.1.6 Reliability Ratio Based Weighted Bit Flipping Algorithm (RR-WBF) (2005)

This algorithm introduces a different way of calculating the bit flipping function by including what is termed a reliability ratio [32].

$$R_{mn} = \beta \frac{|y_n|}{|y_{n\in N(m)}^{max}|} \tag{3.4}$$

In this algorithm, $|y_{n\in N(m)}^{max}|$ is the highest soft magnitude of the variable nodes adjacent to check node m. The term β is a normalization factor to ensure $\sum_{n\in M(n)} R_{mn} = 1$. The new function therefore becomes

$$E_n \triangleq \frac{\sum_{m\,\in\, M(n)}(2s_m - 1)\,|y_{n\in N(m)/n}|_{min}}{R_{mn}} \tag{3.5}$$

Simulations in [32] have shown that the RR-WBF has performance improvement of about 1 to 2 dB at a BER of 10^{-5} over the IMWBF. The algorithm has a complexity similar to that of the IMWBF and there is no need for offline pre-processing.

3.1.7 Channel Independent Weighted Bit Flipping Decoding Algorithm (CI-WBF) (2012)

The CI-WBF modifies the method of computing the weighting function and also does away with the channel information term that is present in the IM-WBF. The weighting term is calculated as shown in Equation 3.13.

$$w_{mn} = \frac{1}{|y_n|} \sum_{n\in M(n)\backslash n} |y_n| \tag{3.14}$$

The weighting function is incorporated into the metric formula as shown

$$E_n \triangleq \sum_{m \in M(n)} (2s_m - 1)\, w_{mn} \tag{3.15}$$

Simulations show that the CI-WBF algorithm has a 1.6 dB coding gain improvement at a BER of 10^{-5} over the WBF and also reduces the number of iterations by 53% without increasing complexity [33].

3.1.8 Modified Channel Independent Weighted Bit Flipping Decoding Algorithm (MCI-WBF) (2013)

This algorithm is also known as the Offset Channel Independent Weighted Bit Flipping Decoding Algorithm (OCIWBF). The basic structure of the CI-WBF is retained apart from the weighting function in which an offset factor $\gamma \geq 0$ is added to improve the accuracy of the CI-WBF. The weighting factor becomes:

$$w_{mn} = \frac{1}{|y_n|} max\left(\sum_{n \in M(n) \backslash n} |y_n| - min(\gamma, 0) \right) \tag{3.16}$$

The offset factor γ is found through Monte-Carlo simulation because theoretical determination of this term is difficult to accomplish. The offset is either a 0 or γ depending on which one is smaller. Computational complexity is only slightly increased at initialisation. Simulations in [34] show that, the OCI-WBF has a 0.45 dB performance gain at a BER of 10^{-5} over the CI-WBF.

3.1.9 An Iterative Bit Flipping based Decoding Algorithm (2015)

The algorithm presented in [35] is a modification of the original weighted bit flipping (WBF) described in [35]. The algorithm is initialised in the exact same way as the WBF, however, the difference, lies in the manner in which the inversion function is calculated.

The proposed algorithm, the recursive weighted bit flipping algorithm (RECWBF) exploits the correlation between iteratively decoded and received channel vectors. The inversion function is:

$$E_{n,k} = \sum_{m \in M(n)} \emptyset_m 2(s_m - 1) + \sum_{t=1}^{k-1} (\lambda)^{k-t} v_n^{k-1} y_n + (\gamma)^k v_n^k y_n \qquad (3.17)$$

Where

$$\emptyset_i = min_{n \in M(n)} |y_n| \qquad (3.18)$$

and the terms λ and γ are known as the forgetting coefficients. The proposed algorithm introduces a new term $v_n^k y_n$ from the preceding iteration to prevent fluctuations in the inversion function values which is a model for bits which are most likely to be flipped. This modification is said to aid in the rapid convergence of the decoder subsequently improving the decoding performance.

Simulations of the RECWBF with PEGReg (1008, 504) and Gallager (273,191) show that the algorithm shows a performance improvement of at least 1.5 dB when compared to the WBF, GDWBF and IERRWBF [35]. There is no theoretical determination of the forgetting factors λ and γ and therefore have been determined via exhaustive simulations.

3.1.10 A Self-Normalized Weighted Bit-Flipping Decoding Algorithm (2016)

The Weighted Bit-flipping (WBF) algorithm is the blue print for the algorithm proposed in [36]. The so-called Self-Normalized Weighted Bit-Flipping Decoding Algorithm (SNWBF) operates as an iterative single bit flipping algorithm for bits that have the highest inversion function value defined as:

$$E_n = \sum_{m \in M(n)} (2s_m - 1) \emptyset_{m,n} \qquad (3.19)$$

Where

$$\emptyset_{m,n} = {min_{i \in N(m)} |y_i| \Big/ |y_n|} \qquad (3.20)$$

The main modification is the introduction of the normalizing absolute variable node channel value $|y_n|$ in the weighting factor which represents the reliability of the received signal y_n . When compared to the IMWBF, the SNWBF does away with the weighting factor because the optimal weighting factor cannot be determined theoretically and so needs extensive simulations to obtain. This is inefficient and the weighting factor varies with changing conditions such as SNR and code structure [36]. The normalization procedure is important so as to take into account the influence of the reliability of the channel information and the hard decision bit. A 2-bit and 3-bit quantized version of the algorithm has also been developed. However, this quantised algorithm does not provide any performance improvement, but it does reduce the complexity.

The SNWBF is only marginally more complex than both the WBF and the IMWBF. Simulations with finite geometry codes (FG-LDPC) codes reveals that both the quantized and unquantized SNWBF algorithm outperforms both the WBF and IMWBF algorithm at high SNR. The proposed algorithm also converges faster and therefore requires fewer iterations to achieve better BER performance when compared to existing aforementioned algorithms.

3.1.11 Hybrid iterative Gradient Descent Bit-Flipping algorithm (HGDBF) algorithm (2016)

The Hybrid iterative Gradient Descent Bit-Flipping algorithm (HGDBF) algorithm is a modified version of the original GDBF. The paper considers the inversion function modified by Li et al [37] in which a zero-mean Gaussian random term q_n and a weighting factor ω to improve performance are introduced:

$$E_n = \hat{z}_n y_n + \omega \sum_{n \in M(n)} (1 - 2s_k) + q_n \tag{3.21}$$

However, the HGDBF is modified by introducing a weighting to the single bit GDBF. This is done on the variable nodes by adding a weighted inversion function value so as to reflect the check node transactions on the variable node in question. The weighting factor takes into account the one-sided power spectral density of the additive white Gaussian Noise (AWGN) associated with the codeword as shown

$$\alpha = \eta \cdot \frac{N_0}{E_b} \tag{3.22}$$

The optimal value for the term η is obtained by simulation and the term $\frac{N_0}{E_b}$ is the inverse of the SNR. The term $\hat{u}_n$ is the hard decision bit term. The final variable node update equation looks like:

$$y_n^{k+1} = y_n^k + \hat{u}_n \cdot \alpha \cdot E_n^k \tag{3.23}$$

Simulations carried out in [37] reveal that the HGDBF has a 0.4 dB gain over the GDBF. It is suggested that improved performance can also be obtained by applying [38] to the noisy GDBF as well as the multi-bit GDBF.

3.1.12 Weighted Bit-Flipping Decoding for Product LDPC Codes (2016)

Unlike most algorithm described in this section, the Page-Based Weighted Bit-Flipping (P-WBF) and its simplified version, the Row-Column Weighted Bit-Flipping (RC-WBF) decoding algorithm are designed specifically for product LDPC codes for 2D codewords described in [38].

The P-WBF has an inversion function for a 2D codeword described by:

$$E_{n_{i,j}} = \frac{1}{\left|y_{n_{i,j}}\right|}\left[\sum_{m\in M_{row}(j)}\left(2s_{m_j}-1\right)\left(\sum_{j\in N_{row}(m)}\left|y_{i,j}\right|\right) + \sum_{m\in M_{col}(i)}\left(2s_{m_i}-1\right)\left(\sum_{i\in N_{col}(m)}\left|y_{i,j}\right|\right)\right] \tag{3.24}$$

The algorithm flips the bit with the highest $E_{n_{i,j}}$ for variable node $y_{n_{i,j}}$ where i and j represents the row and column position respectively. The biggest drawback of the P-WBF is that its complexity increases quadratically as the codeword size increases. This gives rise to the Row-Column Weighted Bit-Flipping (RC-WBF) decoding algorithm which has a linear complexity because it decodes the 2D codeword on column and row directions separately. The RC-WBF inversion function of the RC-WBF is

$$E_n = \frac{1}{|y_n|}\left[\sum_{m\in M_r(j)}(2s_m - 1)\left(\sum_{j\in N_r(m)}|y_n|\right)\right] \tag{3.25}$$

where r is either row or column.

The RC-WBF exhibits what is known as the V-Shape property [38] in which the algorithm reaches the lowest BER at some optimal iteration number but then the BER increases as the iterations go beyond the optimal number of iterations. Thus, a threshold obtained from exhaustive simulations, must be set to ensure optimum BER.

Simulations show that, the P-WBF performs better than the RC-WBF however, the RC-WBF has a lower complexity. Both algorithms have a BER performance that is still inferior to the Belief Propagation SPA decoder [38].

3.2 Multi-bit Flipping Algorithms

The bit-flipping algorithms in this category have the similar characteristics to the single-bit flipping algorithms apart from the fact that they flip multiple bits as opposed to flipping one bit per iteration. The general algorithm framework is the same as the one described in Section 2.3.3 with a few modifications to step 2 and step 3.

3.2.1 Gradient Descent Bit Flipping Algorithm (2008)

This algorithm will have a few different preliminaries defined as follows H is the binary $M\ x\ N$ matrix and $N > M \geq 1$. We have binary linear code $C \triangleq \{\boldsymbol{c} \in F_2^n : H\boldsymbol{c} = 0\}$. Conversion is through $\tilde{C} \triangleq \{(1-2c_1, 1-2c_2, \dots, 1-2c_N : \boldsymbol{c} \in C)\},\ C \in \{0,1\},\ \tilde{C} \in \{+1,-1\}$. The AWGN effects are as described in 1.1. We have the parity condition $\prod_{j\in N(i)} x_j = 1, (\forall i \in [1,N])$ is equivalent to $(x_1, x_2, \dots, x_N) \in \tilde{C}$. The i −th bipolar syndrome of x is the value $\prod_{j\in N(i)} x_j \in \{+1,-1\}$.

Then we have $N(i) \triangleq \{j \in [1,N] : h_{ij} = 1\}$, and $M(j) \triangleq \{i \in [1,M] : h_{ij} = 1\}$ in which case $N(i)$ and $M(j) (i \in [1,M], j \in [1,N])$.

We introduce the objective function which is based on the MLD correlation decoding rule:

$$f(x) \triangleq \sum_{i=1}^{N} x_j y_j + \sum_{i=1}^{M} \prod_{j \in N(i)} x_j \tag{3.26}$$

The first term is a correlation between the received word and the bipolar codeword which requires maximisation. The sum of the bipolar syndrome of x. If and only if $\boldsymbol{x} \in \tilde{C}$, the second term has its maximum value.

The single BF algorithm based function is shown below:

$$\Delta_k^{(GD)} \triangleq x_k y_k + \sum_{i \in M(k)} \prod_{j \in N(i)} x_j \tag{3.27}$$

The GDBF is a variable size bit-flipping algorithm and uses Equation 3.26) and Equation 3.27 to decode by modifying the bit flipping step 3 and step 4 as shown below [39]. We introduce new parameters θwhich is a negative real number named the inversion threshold. A secondary parameter μ defined in step 3 is a binary variable called the mode flag responsible for selecting the mode in which the algorithm is operating.

Step 3 – Evaluate the value of the objective function, and let $f_1 := f(\boldsymbol{x})$. If $\mu = 0$, then execute Sub-step 3-1 (multi-bit mode), else execute Sub-step 3-2 (single-bit mode).

Step 3.1 Flip all bits satisfying

$$\Delta_k^{(GD)} < \theta \ \ (k \in [1, N]) \tag{3.28}$$

Evaluate the value of the objective function again, and let$f_2 := f(\boldsymbol{x})$. If $f_1 > f_2$then make $\mu = 1$.

Step 3.2 Flip a single bit at the jth position, where

$$j \triangleq argmin_{k \in [1,n]} \Delta_k^{(GD)} \tag{3.29}$$

In some cases the algorithm can be trapped in a local maxima which is not a codeword. When this occurs, there is a further modification which called the escape process described in [39].

Simulations presented in [39] show that the Multi-GDBF algorithm has a 1.6 dB performance improvement over the MWBF. The Multi-GDBF algorithm with the escape mechanism adds another 1.5 dB improvement.

3.2.2 Candidate bit based bit-flipping decoding algorithm (CBBF) (2009)

The multi-bit flipping Candidate bit based bit-flipping (CBBF) decoding algorithm is proposed and described at length in [40]. This algorithm works on the premise that an unreliable bit z_n has u_n number of unsatisfied parity check equations. If $u_n > \delta$ where the term δ is some fixed threshold, then is considered to have a high probability of being in error. The bits to be flipped are those where $u_n = u_{max}$ where the term $u_{max} = max(u_n)$.

The m_{th} parity check may have a total c_m number of candidate bits and the lower the number of c_m the more likely, the selected candidate bits are likely to be in error. A new term, the flipping criterion or metric

$$r_n = \sum_{m \in M(n)} c_m - 1 \tag{3.30}$$

Equation 3.30 is introduced such that $u_n = u_{max}$ and $u_l = min(r_i)$ both contribute to unsatisfied parity checks.

The initialisation of the algorithm is similar to prior algorithms apart from the computation of u_n and r_n. The flipping criteria however is such that, flip all the bits in which $u_n = u_{max}$ and $u_l = min(r_n)$.

The CBBF is seen to perform better than the Gallager BF and the soft information WBF. Performance of the CBBF approaches that of the IMWBF but does not surpass it. In comparison to the single bit flipping WBF and IMWBF algorithms, the CBBF is marginally less complex and also converges a lot faster because it flips multiple bits at a time. While most bit flipping algorithms oscillate, the CBBF has a terminating condition in which the algorithm terminates if $u_{max} < \delta$.

3.2.3 Modified I-WBF (MBF) (2009)

In this algorithm, we receive the vector $\boldsymbol{y}$ with soft information from the AWGN. For each output $y_n, 0 \leq n \leq N$ we define a log-likelihood ratio (LLR):

$$L_n \triangleq ln\left(\frac{P(c_n = 1|y_n)}{P(c_n = 0|y_n)}\right) \tag{3.31}$$

The larger the reliability $|L_n|$ the higher the reliability of the hard decision of $z_n^{(0)}$ at iteration 0. For each parity check sum, i.e. the rows in $\boldsymbol{H}$, $1 \leq m \leq M$ compute "lower check reliability" value and the "upper check reliability value" as shown in (3.32)

$$l_m \triangleq min_{l \in M(m)}|L_m|, \qquad u_j \triangleq max_{l \in M(m)}|L_m| \tag{3.32}$$

The cumulative metric by which we flip a bit at position l in iteration k is obtained as follows:

$$\emptyset_n^{(k)} \triangleq \sum_{m \in M(n)} \emptyset_{n,m}^{(k)}, 1 \leq n \leq N \tag{3.33}$$

For each check $m \in M(n)$

$$\emptyset_n^{(k)} \triangleq \begin{cases} |L_n| - l_m/2\,, & if\ s_m^{(k-1)} = 0 \\ |L_n| - (u_m + l_m/2), & if\ s_m^{(k-1)} = 1 \end{cases} \tag{3.34}$$

The bits to flip are the ones with the least cumulative metric. We therefore calculate the number of bits p to flip using as follows:

$$p = \left\lfloor {w_H(\boldsymbol{s}^k)}/{w_C} \right\rfloor \tag{3.35}$$

$w_H(\boldsymbol{a})$ is the Hamming Weight of the vector $\boldsymbol{a}$ while $\lfloor x \rfloor$ represents the largest integer that is equal to or less than x. We then proceed to find the set of p bits $D^k = \left\{v_1^{(k)}, v_2^{(k)}, \dots, v_p^{(k)}\right\}$ with the least value of $\emptyset_n^{(k)}$.

The algorithm also has a loop detection feature. Consider a situation in which a decoded vector $\boldsymbol{z}^{(\boldsymbol{k_0})}$ decoded at iteration k_0 and another $\boldsymbol{z}^{(k)}$ one decoded at iteration k, where $k_0 \leq k$. Then $\boldsymbol{z}^{(k_0+1)} = \boldsymbol{z}^{(k+1)}$ because $\boldsymbol{D}^{(k_0+1)} = \boldsymbol{D}^{(k+1)}$ and this is a clear indication that an infinity loop has occurred. The loop detection feature is discussed here [41].

Simulations indicate that the MBF has a 0.25 dB performance improvement at a BER of 10^{-5} compared to the IMWBF. A few observations have been listed in [41] concerning the MBF:

- The standard algorithm is known to experience decoding failure due to infinite loop traps
- The MFB has fewer infinite traps due to its ability to flip multiple bits at the same time.
- Most bits flipped at each iteration correspond to actual errors. If new errors are introduced they are subsequently corrected in preceding iterations.
- The number of bits flipped by the MBF decrease as the iterations increase.

3.2.4 Soft-Bit-Flipping (SBF) Decoder for Geometric LDPC Codes (2010)

The Soft-Bit-Flipping (SBF) decoder presented in [12] is a multi-bit flipping algorithm designed specifically for geometric codes and VLSI high speed, high throughput applications. The SBF uses a modified version of the IMWBF inversion function [31]. The SBF inversion function does away with IMWBF marginalization which increases complexity and is not suitable for high speed applications but rather employs pseudo-marginalization like the MWBF.

$$E_n \triangleq \sum_{m \in \boldsymbol{M}(n)} (2s_m - 1)\, min_{i \in \boldsymbol{N}(m)} |y_i| - \alpha . |y_n| \tag{3.36}$$

The SBF also uses a multi-threshold technique first developed in [12] to determine the flipping strategy for each bit dependant on the associated inversion function value. The values b_1, b_2 and b_3, which are the flipping thresholds are obtained by investigating the distribution of the inversion function values probability density function and are such that $b_1 < b_2 < b_3$. Instead of performing a hard flip on the codeword, the SBF performs what are referred to as soft flips on VN channel values as follows:

Strong flip:

If $\quad b_3 \leq E_n^{(k)}, y_n^{(k+1)} = sgn\left(\left(\left|y_n^{(k)}\right| - 2.5\right).y_n^{(k)}\right).max\left(1, \left|y_n^{(k)}\right| - 2\right)$

Weak flip:

If $\quad b_2 \leq E_n^{(k)} < b_3, y_n^{(k+1)} = sgn\left(\left(\left|y_n^{(k)}\right| - 1.5\right).y_n^{(k)}\right).\max(1, \left|y_n^{(k)}\right| - 1)$

Maintain:

If $b_1 \leq E_n^{(k)} < b_2$, $y_n^{(k+1)} = sgn\left(y_n^{(k)}\right).\left|y_n^{(k)}\right|$

Strengthen:

If $E_n^{(k)} < b_1$, $y_n^{(k+1)} = sgn\left(y_n^{(k)}\right).min\left(2^q, \left|y_n^{(k)}\right|\right)$

where k is the iteration number.

Simulation and implementation as presented in [12], shows that hardware complexity is reduced significantly when compared to belief propagation algorithms. The algorithm also has an improved BER performance and convergence speed when compared to the IMWBF [31].

3.2.5 Combined Modified Weighted Bit Flipping Algorithm (CM-WBF) (2012)

This algorithm makes use of two algorithms to flip erroneous bits. Both algorithms independently select perceived erroneous bits based on their inversion metric calculations. The two algorithms to a large extend select different bits to flip per iteration. Simulations show that the two algorithms select different bits at least 60 % of the time.

If both algorithms select different bits then both bits can be flipped otherwise flip the bit that the first algorithm chose.

In [42] the first algorithm is the MWBF as explained in Section 3.1.4 while the second algorithm is the as described in explained in [43].

In comparison to other WBF algorithms, the CM-WBF has a higher complexity per iteration. The CM-WBF algorithm has also been observed to have fewer iterations, about a 50% fewer iterations as well as a 0.2 dB and 0.1 dB improvement over the LC-WBF and M-WBF respectively on irregular LDPC codes.

3.2.6 An Adaptive Weighted Multi-bit Flipping Algorithm (AWMBF) (2013)

This algorithm makes use of the IMWBF as described in Section 3.1.5 as well as a bit-flipping threshold $E_{th}$to flip multiple bits per iteration. In this algorithm, the bipolar encoder is defined as $x_n = 1 - 2c_n$. The inversion function E_n of the IMWBF is also defined in Section 3.1.5.

However, in this case, the bit with the highest E_n is the one most likely to be incorrect. Designate the highest $E_n$to E_{max}. A set of the least reliable bits is identified using the following criterion.

$$LRP(E) = \{n | E_n \geq E_{th}, n \in [1, N]\} \tag{3.37}$$

The flipping threshold is defined as follows

$$E_{th} \triangleq E_{max} - |E_{max}| \left(1 - \frac{1}{M} \sum_{m=1}^{M} s_m\right) \tag{3.38}$$

The set of bits $n^* \in LRP(E)$ are all flipped in parallel.

Simulation results presented in [11] show that, the AWMBF performs better than the IMWBF with fewer iterations and achieves approximately 0.9 dB performance gain over the IMWBF at a BER of 10^{-5} [11].

3.2.7 Two-Staged Weighted Bit Flipping Decoding Algorithm (2015)

In [44] is presented the Two-Staged Weighted Bit Flipping Decoding Algorithm (TSWBF) aimed at mitigating the error floors prevalent in many bit flipping decoders. Achieving this objective will enable the implementation of LDPC decoders in low power communication devices.

The proposed algorithm is a hybrid of the parallel IMWBF (PWBF) [45] (which has an inversion function identical to the IMWBF in [45]) and the Gallager Bit Flipping (BF) algorithm. The hybrid system is meant to mitigate the high error floors that appear at the high SNR regions of the PWBF decoder [45].

It has been investigated and been found out that the PWBF decoding fails when there is a bad node and this happens when a variable ω_S is greater than a value T_S. The TSWBF algorithm is designed such that it switches to the BF when $\omega_S < T_S$ .The switching of the TSWBF is described as follows:

Generate the syndrome $\boldsymbol{s}$

$$If\ (\boldsymbol{s} = \emptyset)\ or\ (k > K_{Max})$$

Terminate TSWBF and output codeword;

Else If $0 < \omega_S < T_S$

Activate Bit Flipping (BF) Algorithm decoding

Else If $T_S \leq \omega_S$

Activate Two-Staged Weighted Bit Flipping (TSWBF) Algorithm decoding

End If

The computational complexity of the TSWBF is slightly more than that of the PWBF. The term cannot be obtained by theoretical means and so can only be obtained by exhaustive simulations. The TSWBF decoder is simulated against the PWBF and the BP algorithm for EG, PG, McKay and PEG codes and is shown to reduce the error floors prevalent in the PWBF. As expected, the TSWBF algorithm does not perform better than the BP algorithm, however it does reduce the performance gap between the BP and the PWBF algorithm [44].

3.2.8 Mixed Modified Weighted Bit Flipping Decoding Algorithm (MM-WBF) (2015)

This algorithm makes use of two algorithms to flip erroneous bits. One algorithm acts as the main algorithm while the other acts as an auxiliary algorithm. The main algorithm has to be the best performing algorithm available while the auxiliary algorithm is the second best and this is determined by simulations. The two algorithms must also to a large extend select different bits per iteration.

Certain rules apply when flipping bits. First, if both algorithms select different bits then both bits can be flipped otherwise flip the bit that the main algorithm chose. Secondly, the auxiliary algorithm may not flip bits that have already been flipped by the main algorithm.

In [32] the main algorithm is the RR-WBF as explained in Section 3.1.6 while the second algorithm is the IMWBF as explained in Section 3.1.5.

In comparison to other WBF algorithms, the MM-WBF has a higher complexity per iteration. The MM-WBF algorithm has also been observed to have fewer iterations and a 0.4 dB and 0.6 dB improvement over the RR-WBF and IM-WBF respectively on irregular LDPC codes.

3.2.9 Noise-Aided Gradient Descent Bit-Flipping Decoder (2016)

In the literature [46], a Noise-Added Gradient Descent Bit-Flipping Decoder is proposed as a further enhancement to the existing GDBF described in [39]. The algorithm has been designed for both the BSC channel and the AWGN channel. Two important concepts intended to improve GDBF decoding are presented namely the introduction of noise to VN flipping matric calculation and iterative scheduling.

It is demonstrated in simulations that adding noise to GDBF decoders in form of random perturbations enables the decoders to escape error prone structures of fixed points exhibited by trapping sets that result in decoding failure.

The iterative scheduling modification of the noisy GDBF algorithms enhances the random nature of the VN perturbations which further improves the performance of the decoder to approach the performance of a typical MLD decoder when the algorithm runs for a large number of iterations.

This novel noise addition to GDBF decoder design is expected to improve in performance to match the decoding performance of soft decision belief propagation (BP) decoders [46].

3.3 Threshold Based decoding

The syndrome is an indication of the correctness of a received codeword. The parity check equations converging on a codeword, in which the syndrome bits participate, also gives a measure of the probability of the correctness of a variable node hard decision. This characteristic of the LDPC codes has given rise to various bit flipping algorithms which make use of parity check equation based inversion functions.

The simplest algorithms [29] only flip one bit per iteration. In the WBF [47] single bit flipping algorithm, the inversion function is defined in Equation 3.7. The bit with the highest probability of error has the highest value of E_n. The single bit flipping GDBF algorithm on the other hand operates in the opposite manner. The objective function described in Equation 3.26 provides a measure of the correctness of the hard decision. However, in this case, the variable node with the smallest value of Δ_n is considered to be the least reliable bit and should therefore be corrected.

The single bit flipping concept can be further enhanced and applied to multi-bit flipping algorithms. Weighted Bit Flipping algorithms such as the WBF, MWBF and IMWBF operate on the premise that the higher the value of E_n of a particular bit, the more likely that bit is in error. Gradient Descent Bit Flipping (GDBF) algorithms operate on the premise that the smaller value of Δ_n, the higher the probability of error.

In order to correct multiple bits at the same time, it can be assumed that for WBF based decoders, the higher values of E_n contain a significant number of erroneous bits and the least of the Δ_n values are comprised of erroneous bits for GDBF based decoders. This reasoning has brought about what are known as thresholds for multi-bit flipping decoders which are used as a flipping criterion depending on the algorithm in question.

The Adaptive Weighted Multi-bit Flipping Algorithm (AMWBF) uses a non-fixed threshold to correct erroneous bits. The threshold is calculated using Equation 3.38 and bits with E_n above this threshold are flipped while those below it are left unchanged. The SBF in [12] uses multiple fixed thresholds to determine how to modify the associated soft bit information. The multi-bit flipping GDBF also uses a fixed threshold to determine which bits to flip. Almost all the algorithms described in Section 3.2 of this document use some form of threshold to determine which bits to flip.

This method of flipping is very effective in speeding up the convergence of a bit flipping algorithm when it has a high chance of flipping erroneous bits correctly. However, this concept of thresholds sometimes results in the flipping of correct bits in which case new errors are introduced into the codeword.

3.4 Channel Information modification

The default operation of bit flipping algorithms is to obtain the hard decision from the received channel values upon initialization. As the algorithm iteratively decodes, certain bits which meet the flipping criterion get inverted with no modification to the associated soft channel information.

In [12] and [46] the bit flipping algorithms perform the flipping function using a different strategy which involves the modification of the soft channel information which in turn modifies the hard decision. The purpose of modifying the soft channel information is to increase the

transfer of information from variable nodes to check nodes and vice versa in the same way the sum-product algorithm passes messages for more effective error correction [3].

The multiple-step majority logic decoder by Palinki et al [48] is the first known decoder to make note of the difference in the distribution of the inversion function values and therefore modify the bits with respect to the reliability as calculated by the inversion function. The multiple-step majority logic decoder was a three stage decoder which worked as follows:

If any of the bits in a checksum calculation are erased, deactivate the check sum

If $b_1 \leq E_n^{(k)}$, Flip the bit n

If $b_2 \leq E_n^{(k)} < b_1$, Erase the bit n

If $E_n^{(k)} < b_2$, Maintain the bit n

The simulations with EG codes have shown that, the algorithm can operate at close to MLD of intermediary length MSLMLD codes despite the presence of short cycles.

The BWBF described in Section 3.1.3 also incorporates soft bit flipping decoding by initialising the variable node values according to the rule shown below

$$y_n' = y_n + \sum_{m \in N(m)} \prod_{n \in M(n) \backslash n} sgn(y_n).min_{n \in M(n) \backslash n} |y_n|$$

The aptly named Soft bit flipping (SBF) algorithm decodes codewords by manipulating the received codeword soft information as opposed to hard decision bit flipping [12]. The algorithm uses a non-marginalized IMWBF inversion function to calculate the reliability of the hard decision. While in the usual case, an LDPC decoding algorithm would flip bits that meet a given flipping criterion, the SBF, has three different thresholds b_1, b_2 and b_3 that are used to modify soft channel information as follows:

Strong flip:

If $b_3 \leq E_n^{(k)}$,

$$y_n^{(k+1)} = sgn\left(\left(\left|y_n^{(k)}\right| - 2.5\right).y_n^{(k)}\right).max\left(1, \left|y_n^{(k)}\right| - 2\right) \qquad (3.39)$$

Weak flip:

If $b_2 \leq E_n^{(k)} < b_3$,

$$y_n^{(k+1)} = sgn\left(\left(\left|y_n^{(k)}\right| - 1.5\right).y_n^{(k)}\right).\max(1, \left|y_n^{(k)}\right| - 1) \quad (3.40)$$

Maintain:

If $b_1 \leq E_n^{(k)} < b_2$,

$$y_n^{(k+1)} = sgn\left(y_n^{(k)}\right).\left|y_n^{(k)}\right| \quad (3.41)$$

Strengthen:

If $E_n^{(k)} < b_1$,

$$y_n^{(k+1)} = sgn\left(y_n^{(k)}\right).min\left(2^q, \left|y_n^{(k)}\right|\right) \quad (3.42)$$

where k is the iteration number.

The SBF decoder has been simulated with Geometric LDPC codes and has been observed to have a higher throughput and better BER performance when compared to the regular IMWBF [45].

The HGDBF algorithm presented in [46] is a hybrid iterative decoder for LDPC Codes. The HGDBF uses the latest channel information and inversion function value to update the variable node to increase the reliability of the hard decision. The inversion function for the HGDBF is described in [46]. The variable node values are iteratively updated per iteration k as follows:

$$y_n^{k+1} = y_n^k + \hat{u}_n \cdot \eta \cdot \frac{N_0}{E_b} \cdot E_n^k \quad (3.43)$$

The variable node update is also used to reduce oscillations that are prevalent in many LDPC bit flipping algorithms.

4. Adaptive Multi-Threshold Multi-bit Flipping Algorithm (AMTMBF)

4.1 Introduction

In this chapter, a new adaptive threshold based multi-bit flipping algorithm is devised. The proposed algorithm known as the Adaptive Multi-Threshold Multi-bit Flipping Algorithm (AMTMBF) is an investigation into the efficacy of the adaptive threshold strategy in the design of multiple bit decoding LDPC algorithm in terms of bit error rate (BER), average number of iterations (ANI) and complexity.

As discussed in literature, there has been extensive study into the design of single bit flipping algorithms which span from Gallager's original bit flipping algorithm [3] to the more recent single bit flipping Gradient Descend bit flipping algorithms (GDBF [39]. The most obvious drawback of single bit-flipping algorithm is the average time required for convergence which increases the complexity of the algorithm mostly because the algorithm flips a perceived incorrect bit per iteration. The second problem with single bit flipping algorithms occurs when a codeword has more errors than the maximum number of iterations, then it is impossible to correct that particular codeword even when the total number of errors is lower than the minimum distance of that particular code. Thirdly, the algorithm may flip the wrong bits thereby introducing new errors into the codeword or it may start oscillating about a set of bits until it reaches maximum number of iterations in which case a decoding failure occurs.

The weaknesses detailed above that are associated with the single bit flipping algorithms have motivated increased research into several variations of multi-bit flipping algorithms to facilitate the design and development of low cost, low complexity and high speed communication devices as shown in [12]. The major benefit of using multi-bit flipping algorithms is in the average number of iterations (ANI) required for convergence which are reduced when compared to the SBF ANI. This reduced ANI also manifests itself in the reduced complexity of the decoder since multiple bits can be corrected in fewer iterations. However, the multi-bit flipping strategy does not solve the issue of oscillations as this is also prevalent in all bit-flipping algorithms. However, a technique to curb oscillations has been presented by Ngatched et al in [41].

While, it is relatively simple to determine which bit to flip in single bit flipping algorithms where the bit with the largest or smallest inversion matric is flipped, determining which bits to

flip in multiple bit flipping algorithm is a more complex process. This process involves some form of threshold with which to identify bits with high error probability.

The AWMBF presented in [11] uses an adaptive threshold determined by the state of the algorithm as it iteratively decodes. This is very effective in curbing oscillations as the threshold is not fixed but changes as the errors are corrected in the codeword.

The SBF implemented on VLSI chips and presented in [12] makes use of multiple fixed thresholds to modify the associated soft bit information in its decoding iterations. The reason for this is so that the algorithm can take note of various degrees of reliability of all the inversion function values and adjusts each of the associated soft bit values in each individual iteration.

Weighted bit flipping algorithms incorporate weighted soft bit information to aid in the calculation of the reliability factor or inversion function which is the metric by which the reliability of the bit hard decision is measured. A bit is then flipped or left unchanged based on this metric. In a majority of these algorithms, only the hard decision bits are modified with no corresponding modification to the associated soft bit information. In the paper [12], the SBF makes use of soft information modification to enhance the effect of the weight by adjusting the signs and magnitude of the soft information to reflect the confidence in reliability of the hard decision in the calculation of the inversion function weight.

Application of this soft information modification scheme in the SBF coupled with a fixed multi-threshold scheme has improved its decoding capabilities. The adaptive threshold scheme presented in [11] has also improved the convergence and BER of the AWMBF. The AMTMBF presented in this chapter combines the multi-threshold scheme; adaptive threshold scheme and the soft information modification scheme with the aim to improve the decoding capabilities of the existing threshold based multi-bit flipping algorithms.

4.2 Adaptive Threshold Scheme

Hard decision LDPC bit-flipping algorithms determine the reliability of a given codeword bit by computing a reliability function also known as the inversion function. In single bit flipping functions, the bit with the least reliable inversion function value is flipped while a group of bits with the least reliable inversion function values are flipped in the multi-bit flipping algorithms.

The crux of the multi-bit flipping conundrum is determining which bits have the most unreliable inversion function values such that amongst the ones selected, none have been

selected in error such that the algorithm flips correct bits thereby introducing errors into the codeword. This has to be accomplished while also ensuring rapid convergence.

The adaptive threshold scheme has received very little attention in favour of fixed threshold algorithms. This is a scheme in which the threshold with which to determine bits to flip varies according to the state of the decoding algorithm. The adaptive threshold scheme has been introduced to the IMWBF to create the multi-bit flipping AWMBF [11] described in Section 3.2.6.

The AWMBF converts the reliability profile of a given codeword into two sections. The one section contains the most reliable bits (MRB) while the other contains the least reliable bits (LRB). The threshold is calculated as shown in Equation 3.38.

4.2.1 Convergence comparison – Single bit flipping vs Adaptive threshold multi-bit flipping

An important performance metric for bit flipping algorithms is the average number of iterations (ANI) required for convergence. In general, multi-bit flipping algorithms converge faster than single bit flipping algorithms because they correct more bits per iteration as opposed to one bit. However, many bit flipping algorithms are susceptible to oscillations which may take away the low ANI advantage that the multi-bit flipping algorithms have.

Figure 4.1 shows the ANI performance comparison between the IMWBF and the AWMBF using Projective Geometry Regular (PEGReg (1008, 504)) LDPC codes for an average 100 000 codewords at a maximum number of iterations equal to 100. At low SNR both algorithms have the same performance in terms of convergence. Below an SNR of 3 dB, both algorithms actually do not converge to a valid codeword therefore using up the maximum 100 iterations without converging to a valid codeword. This can be attributed to both algorithms either oscillating or introducing new errors due to flipping the wrong bits. A BER performance can be used to determine which algorithm corrects more errors. However, the paper [11], Chen et al shows that the AWMBF has better error correction performance when compared to the IMWBF.

At high SNR, the ANI performance improves with the multi-bit AWMBF taking the lead. At SNR higher than 3 dB, the inversion function value profile can clearly distinguish between the correct and wrong bits and therefore the AWMBF can identify multiple erroneous bits

accurately and correct them at once in one iteration while the IMWBF algorithm can only identify and correct one bit per iteration.

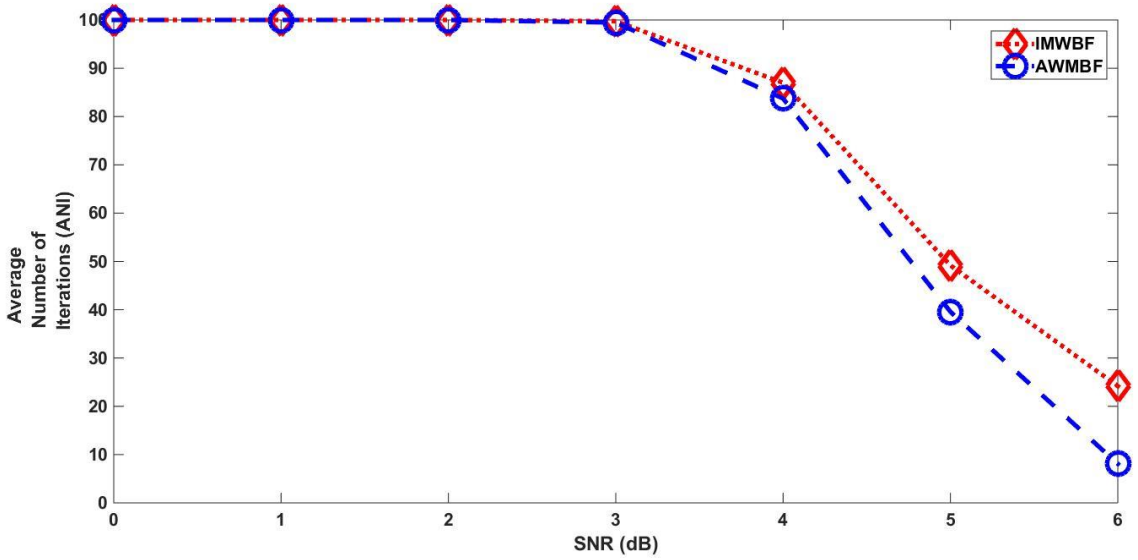


Figure 4. 1: Average number of iterations (ANI) comparison between IMWBF and AWMBF

4.2.2 Convergence comparison – Existing and proposed adaptive threshold techniques

The AWMBF algorithm uses an adaptive threshold scheme described by the equation in [11]. The equation generates the term E_{th} which divides the $\boldsymbol{E}$ tuple into two parts, MRB (Most Reliable Bits) and LRB (Least Reliable Bits) to be flipped. The value E_{th} is calculated by finding the LRB window which is dependent on E_{max}, which is the value of the highest value in the $\boldsymbol{E}$ tuple as well as the state of the syndrome vector $\boldsymbol{s}$.

$$E_{th} \triangleq E_{max} - |E_{max}|\left(1 - \frac{1}{M}\sum_{m=1}^{M} s_m\right)$$

It is clear that both the $\boldsymbol{E}$ tuple and the syndrome vector $\boldsymbol{s}$ are terms that vary with the state of the algorithm from iteration to iteration. Hence the LRB window varies as $\boldsymbol{E}$ and $\boldsymbol{s}$ change.

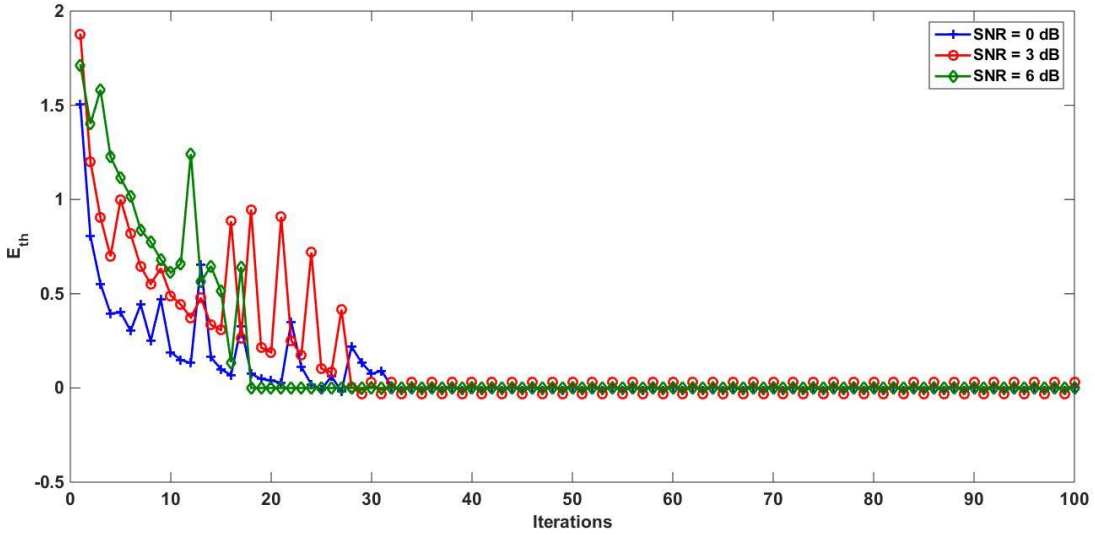


Figure 4. 2: Flipping threshold analysis for AWMBF at varying SNR

Figure 4.2 shows the variation of the threshold with iteration for a single codeword at various different signal to noise ratios. At SNR = 0 dB, there is a decrease in ANI with major spikes between iteration 12 and 32. Starting from iteration 33 to maximum iteration the threshold starts to oscillate between two levels which is a reflection of what is actually happening within the algorithm. At this point the algorithm is stuck in a continuous loop or oscillation where it keeps identifying and flipping the same set of bits without converging to a correct codeword. The similar oscillation phenomenon occurs at signal-to-noise ratio of 3 dB from iteration number 28 while major spikes are observed from iteration number 15 to 27. At 6 dB, the algorithm converges to a valid codeword after 17 iterations.

4.2.3 Adaptive Multi Threshold

Consider the tuple $\boldsymbol{E}$ which represents the inversion function calculations which reflect the reliability of the hard decision made on the codeword bits in hard decision tuple $\boldsymbol{z}$. The tuple $\boldsymbol{E}$ values span from the least reliable E_{max} to the most reliable E_{min}. The subset of the least reliable bits (LRB) spans from $E_{max} - \Delta$ to E_{max}. Where Δ is the search window which in this case Δ for the AWMBF is $\Delta(E_{max}, \boldsymbol{s})$. The syndrome $\boldsymbol{s}$ and the value E_{max} is a good indicator with which to estimate the size of the LRB window. A valid codeword has a null syndrome $\boldsymbol{s} = \emptyset$. Considering Equation 3.4, given the syndrome, we can obtain the error vector $\boldsymbol{e}$ by:

$$\boldsymbol{e} = \boldsymbol{s}.(H^T)^{-1}$$

Therefore, it can be inferred that the larger the number of non-zero terms in the syndrome, the larger the perceived number of errors $e = H(\boldsymbol{e})$, such that $e \propto H(\boldsymbol{s})$ as a hypothesis stating that the number of non-zero elements in the syndrome $\boldsymbol{s}$ is directly proportional to the number of errors in the codeword $\boldsymbol{c}$. The term $H(\boldsymbol{x})$ is the hamming weight of a given vector $\boldsymbol{x}$.

The maximum $H(\boldsymbol{s}) = M$ and therefore an estimation of the number of errors in a codeword can be determined by

$$\frac{e}{N} \cong \frac{H(\boldsymbol{s})}{M} max(\boldsymbol{E}) \tag{4.1}$$

Which is an approximate size of the window

$$\Delta(E_{max}, \boldsymbol{s}) \cong \frac{e}{N} \cong \frac{H(\boldsymbol{s})}{M} max(\boldsymbol{E}) \tag{4.2}$$

Therefore a new adaptive threshold is defined in Equation 4.3

$$E_{th} \triangleq E_{max} - \Delta(E_{max}, \boldsymbol{s}) \tag{4.3}$$

For ease of computation as the algorithm executes, Equation 4.3 can be rewritten as

$$E_{th} = E_{max} - \gamma . |E_{max}| \left(\frac{1}{M} \sum_{m=1}^{M} s_m \right) \tag{4.4}$$

where γ is a weight that can be optimised through simulations.

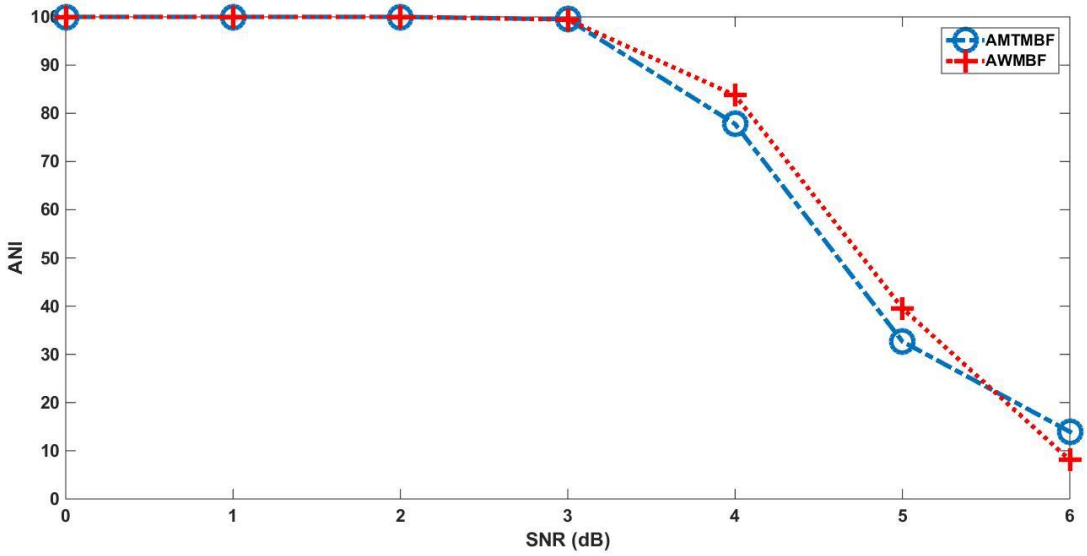


Figure 4. 3: Average number of iterations (ANI) comparison between AMTMBF and AWMBF

The proposed adaptive threshold equation with two terms is slightly different from the AWMBF adaptive threshold with three term. Figure 4.3 is a comparison between the proposed adaptive threshold and the AWMBF threshold in terms of convergence using PEGReg (1008, 504) codes. The proposed adaptive threshold scheme converges faster than the AWMBF for signal to noise ratio between 3 dBs and 5.5 dBs. Beyond 5.5 dBs, the AWMBF has better convergence.

A large majority of the existing multi-bit algorithms described in Section 3.2 are single threshold schemes which means that a single threshold is used to divide the codeword bits into two sets, the LRB bits to be flipped and the MRB bits which are usually left unchanged.

The drawback of the multi-bit flipping single threshold scheme is that the bits in each set are assumed to be of the same reliability. Take for instance, the bits in the LRB set, these bits are all inverted as they are assumed to all be in error and the MRB bits are all left untouched as they are considered to be correct. This is not very effective because, the tuple $\boldsymbol{E}$ spans from E_{min} to E_{max} and each value of E_n represents a different measure of reliability which is not taken into consideration when a single threshold is used. In [48] a multi-threshold algorithm is introduced in which different sets of bits are considered and corrected based on their perceived reliability. The SBF in [12] further develops the concept in [48] by introducing three thresholds and modifying the soft information in each bit based on the calculated reliability value.

A more effective decoder design will take into consideration the different reliability measures of all the values in $\boldsymbol{E}$ and modify the bits or associated bit information accordingly. A multi-threshold scheme is proposed as follows:

Consider an error vector $\boldsymbol{e_1}$ that can be identified via the window $\boldsymbol{\Delta_1}$ such that:

$$\Delta_1 = \frac{e_1}{N} \tag{4.5}$$

Consider also an error vector $\boldsymbol{e_2}$ which is a subset of $\boldsymbol{e_1}$ such that $\boldsymbol{e_2}$ can be extracted from $\boldsymbol{e_1}$ by:

$$e_2 = \beta_1 e_1 \tag{4.6}$$

The window Δ_2 to identify $\boldsymbol{e_2}$ can be obtained as:

$$\Delta_2 = \frac{e_2}{N} = \beta_1 \frac{e_1}{N} \tag{4.7}$$

In the same way window Δ_3 to identify $\boldsymbol{e_3}$ can be obtained as:

$$\Delta_3 = \frac{e}{N} = \beta_2 \frac{e_1}{N} \tag{4.8}$$

Therefore, the arbitrary window Δ_i to identify $\boldsymbol{e_i}$ can be obtained as:

$$\Delta_i = \frac{e_i}{N} = \beta_{i-1} \frac{e_1}{N} \tag{4.9}$$

All the possible windows for an error vector $\boldsymbol{e_N}$ can be defined as follows

$$[\Delta_1 \ , \Delta_2 \ , \Delta_3 \ , \dots , \Delta_N \ ] = \left[\frac{e_1}{N}, \beta_1 \frac{e_1}{N}, \beta_2 \frac{e_1}{N}, \dots , \beta_{N-1} \frac{e_1}{N}\right] \tag{4.10}$$

Where the values $[\beta_1, \beta_2, \dots, \beta_{N-1}]$ have to be determined through simulation.

The error vector $\boldsymbol{e_i}$ can be identified between two windows $[\Delta_i\ , \Delta_{i+1}\ ]$ such that $[\boldsymbol{e_1}, \boldsymbol{e_2}, \dots]$ can be identified from $[(\Delta_1\ , \Delta_2), (\Delta_2\ , \Delta_3), \dots]$

It is practically impossible to devise a bit flipping algorithm that makes use of all the possible thresholds without incurring prohibitive computational complexity. The AMTMBF makes use of two thresholds E_{th1} and E_{th2} such that the error vectors in the codeword $[\boldsymbol{e_1}, \boldsymbol{e_2}, \boldsymbol{e_3}]$ can be identified from $[E_n > E_{th1}, E_{th1} \geq E_n > E_{th2}, E_{th2} \geq E_n]$.

The two thresholds are obtained from the window calculation

$$\Delta_i(E_{max}, \boldsymbol{s}) \cong \frac{\boldsymbol{e}}{N} = \frac{H(\boldsymbol{s})}{M} max(\boldsymbol{E}) \tag{4.11}$$

The thresholds E_{th1} and E_{th2} uses a different scaling factor for the window. The thresholds for the AMTBF are calculated as follows:

$$E_{th1} \triangleq E_{max} - \Delta_1(E_{max}, \boldsymbol{s}) = E_{max} - \gamma_1 . |E_{max}| \left(\frac{1}{M} \sum_{m=1}^{M} s_m \right) \tag{4.12}$$

$$E_{th2} \triangleq E_{max} - \Delta_2(E_{max}, \boldsymbol{s}) = E_{max} - \gamma_2 . |E_{max}| \left(\frac{1}{M} \sum_{m=1}^{M} s_m \right) \tag{4.13}$$

where γ_1 and γ_2 are weighting factors that can be optimised through simulations.

4.3 Channel Information Modification Scheme

Weighted bit flipping algorithms depend on two sources of information to make decoding decisions. These two sources of information are the soft information obtained from the channel and the hard decision codeword derived from this soft information. Conventional bit flipping algorithms simply flip the least reliable bits of the hard decision codeword by inverting them. In literature, the SBF devised in [12] as well as the HGDBF in [46] enhance the decoding capabilities of a conventional decoder by modifying the codeword soft information based on

reliability values. This gives the algorithm more decoding states as opposed to binary states of the codeword bits. This is also effective in decoding because soft information participates in the calculation of the inversion function calculation, thus modifying it with respect to reliability measurements will influence the value of the inversion function in the next iteration. The soft information can be modified based on the various degrees of reliability in the vector $\boldsymbol{E}$.

4.3.1 The proposed scheme

The scheme used in the proposed algorithm is described as follows:

i) *Flip* – The bits in this set are considered to be the most unreliable and should therefore be inverted. However, these bits are inverted indirectly by inverting the actual soft information which will manifest as a codeword bit flip in the next iteration. The bit flip is described by:

$$y_n^{(k+1)} = (-1).sgn\left(y_n^{(k)}\right).\left|y_n^{(k)}\right| \tag{4.14}$$

ii) *Maintain* – The bits in this set are considered to be correct to an acceptable degree and should therefore remain untouched. The uncertainty of the correctness of these bits is not 0 % and therefore, the soft information remains untouched as well so that incorrect bits do not negatively affect the inversion function calculations in the next iterations. The maintain procedure is described by:

$$y_n^{(k+1)} = sgn\left(y_n^{(k)}\right).\left|y_n^{(k)}\right| \tag{4.15}$$

iii) *Strengthen* – The bits in this set are considered to be correct with the highest level of confidence. Since these bits are correct, the related soft information can be used to influence the inversion function calculations for other bits. Therefore, the value of the soft information of these bits can be enhanced or strengthened to influence the next iteration. The strengthening procedure is described by:

$$y_n^{(k+1)} = sgn\left(y_n^{(k)}\right).\left|y_n^{(k)}\right| + |\delta| \qquad (4.16)$$

4.3.2 Determining the enhancement value $|\delta|$

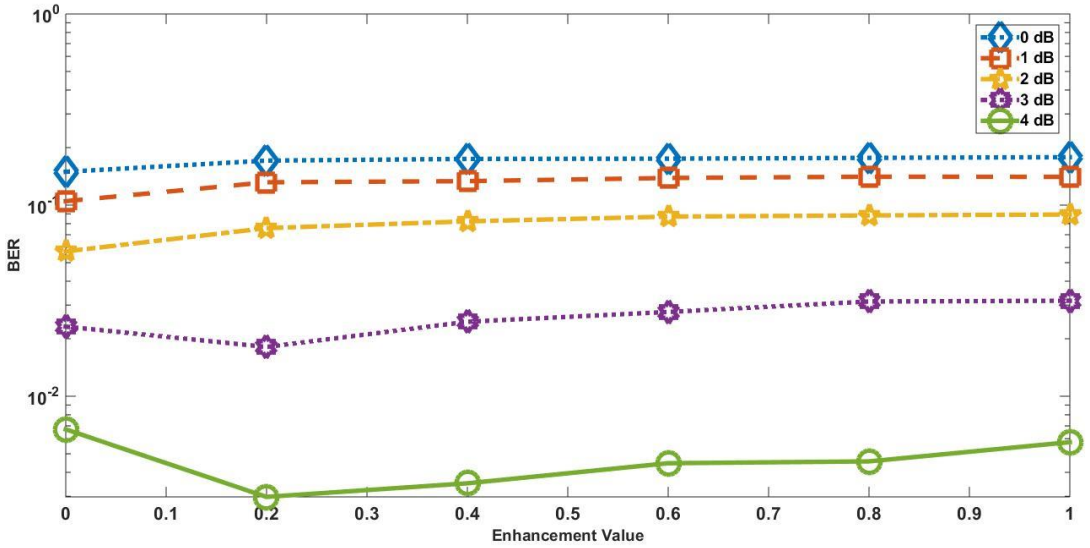


Figure 4. 4: Determining enhancement value |δ| for AMTMBF

The enhancement value vs BER plot is shown in Figure 4.4. The test values span from 0 to 1 because the BPSK modulation is of unit power and simulations have shown that values greater than one worsen the BER for all SNR values as shown in Figure 4.5.

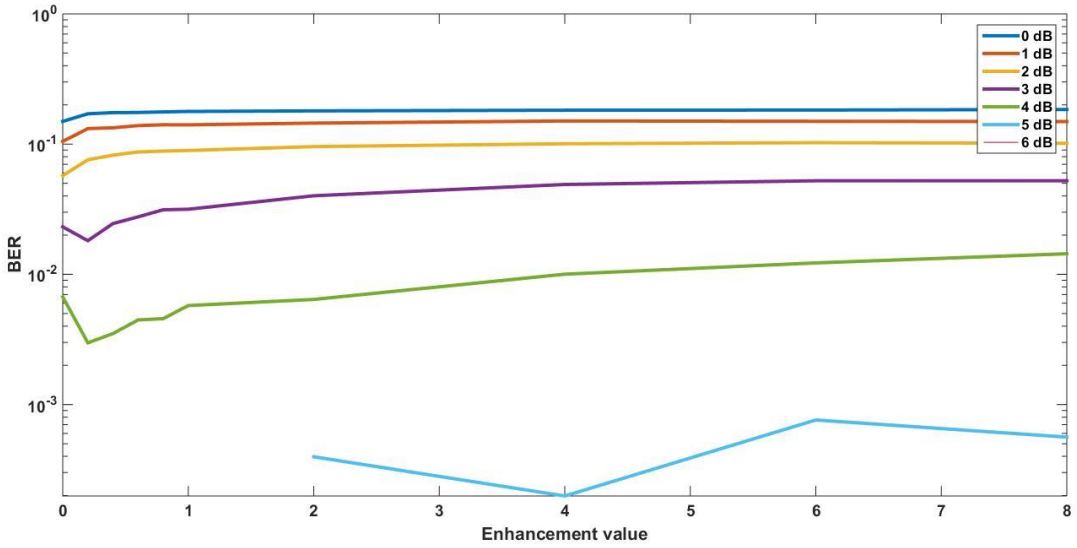


Figure 4. 5: Enhancement values investigation

It can be seen that for low SNR values, the most appropriate value of the enhancement value is 0 and for higher SNR values, the best Enhancement value is 0.2 for PEGReg (1008, 504) LDPC codes. This analysis presents a problem since the optimal enhancement value varies with signal to noise to ratio therefore, for optimal performance, the algorithm will need input from a channel estimator. However, finding an average enhancement value offers a simpler and less complex solution.

4.4 Proposed Decoding Algorithm

This section is the culmination of all the concepts discussed in this chapter so far. The proposed algorithm is known as the Adaptive Multi-Threshold Multi-bit Flipping Algorithm (AMTMBF). The algorithm uses adaptive multi-thresholds as well as soft information modification.

The full algorithm is described below:

Initialize: set $k = 0$, as well as $\boldsymbol{z}^{(\boldsymbol{0})} = \boldsymbol{z}$. For $1 \leq m \leq M$ and compute $min_{i \in \boldsymbol{N}(m) \backslash n} |y_i|$ and store.

1) While $k \leq k_{max}$, compute the syndrome $\boldsymbol{s}^{(\boldsymbol{k})} = \boldsymbol{z}^{(\boldsymbol{k})} \boldsymbol{H}^{(\boldsymbol{T})}$. If $\boldsymbol{s}^{(\boldsymbol{k})} = \boldsymbol{0}$, terminate algorithm and output decoded codeword $\boldsymbol{z}^{(\boldsymbol{k})}$.

2) Compute the vector $\boldsymbol{E^{(k)}} = \left(E_1^{(k)}, E_2^{(k)}, \ldots, E_N^{(k)}\right)$ based on:

$$E_n \triangleq \sum_{m \in \boldsymbol{M}(n)} (2s_m - 1)\, min_{i \in \boldsymbol{N}(m) \setminus n} |y_i| - \alpha . |y_n|$$

α is the weighting factor obtained via simulations.

3) Compute the upper flipping and lower strengthening threshold E_{th1} and E_{th2} respectively based on:

$$E_{th1} \triangleq E_{max} - \gamma_1 . |E_{max}| \left(\frac{1}{M} \sum_{m=1}^{M} s_m\right)$$

And

$$E_{th2} \triangleq E_{max} - \gamma_2 . |E_{max}| \left(\frac{1}{M} \sum_{m=1}^{M} s_m\right)$$

Where γ_1 and γ_2 are determined via simulation.

4) Obtain $\boldsymbol{y^{(k+1)}} = \left(y_1^{(k)}, y_2^{(k)}, \ldots, y_N^{(k)}\right)$ on the basis of the following:

(a) Flip:

If $E_{th1} < E_n^{(k)}$,

$$y_n^{(k+1)} = (-1) . sgn\left(y_n^{(k)}\right) . \left|y_n^{(k)}\right|$$

(b) Maintain:

If $E_{th2} < E_n^{(k)} \leq E_{th1}$,

$$y_n^{(k+1)} = sgn\left(y_n^{(k)}\right) . \left|y_n^{(k)}\right|$$

(c) Strengthen:

If $E_{th1} \leq E_n^{(k)}$,

$$y_n^{(k+1)} = sgn\left(y_n^{(k)}\right) . \left|y_n^{(k)}\right| + |\delta|$$

5) $k \leftarrow k + 1$, if $k > k_{max}$ terminate algorithm otherwise go to step 1.

4.5 Simulation Results

The Adaptive Multi-Threshold Multi-bit Flipping Algorithm (AMTMBF) was simulated through a Binary Input – Additive White Gaussian Noise (BI-AWGN) channel whose noise signal to noise values varies from 0 dBs to 6 dBs. The binary symbols where modulated using the Binary Phase Shift Keying (BPSK) modulating technique. All simulations where carried out in Matlab.

Various types of LDPC codes are used for different algorithms as described in Section 2.2.1. It is important to ascertain the type of code that produces the best performance for the AMTMBF algorithm. At the compilation of this dissertation, there is no known theoretical method of determining the most suitable type of LDPC code for a given algorithm.

The AMTMBF was simulated with a subset of some of the most effective regular codes namely the

i) Progressive Edge Growth Regular codes – PEGReg(1008,504)
ii) Progressive Edge Growth Regular codes – PEGReg(504,252)
iii) Gallager random codes – Gal (408,204)
iv) Geometric Cyclic codes – Cyclic (1008,504)

These codes where obtained from [17]

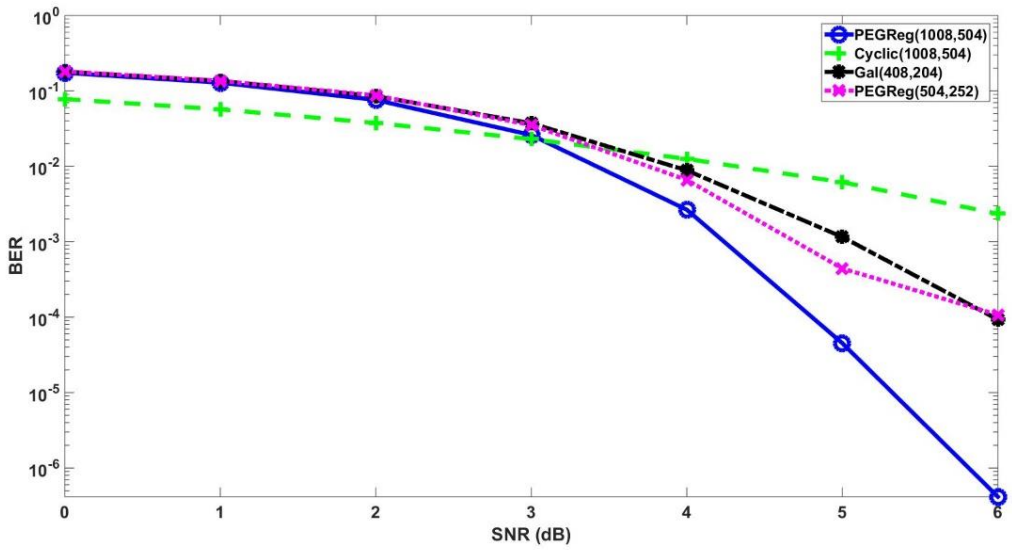


Figure 4. 6: AMTMBF BER analysis for different LDPC codes

Figure 4.6 and Figure 4.7 show the bit error rate (BER) and the frame error rate (FER) of the AMTMBF when simulated with the codes (i)-(iv). Cyclic codes provide the lowest BER and FER at low SNR but provide the worst performance at high SNR and the slope almost mirrors that of the BPSK coding scheme. The very first LDPC codes devised by Gallager are the second worst performing codes for the AMTMBF and this can be attributed to their random nature which has no specific design objective. According to [17], the best known LDPC codes are the Progressive Edge Growth (PEG) codes and the two known regular codes of size 504 and 1008 are simulated in this text. The longer PEGReg (1008, 504) codes have better performance in terms of BER and FER when compared to the shorter PEGReg (504, 252) codes. From a signal to noise ratio of 3 dBs, the PEGReg (1008, 504) codes outperform the other three codes in terms of BER. The best FER performance is also obtained from the PEGReg (1008, 504) codes from a signal to noise ratio of 4 dBs.

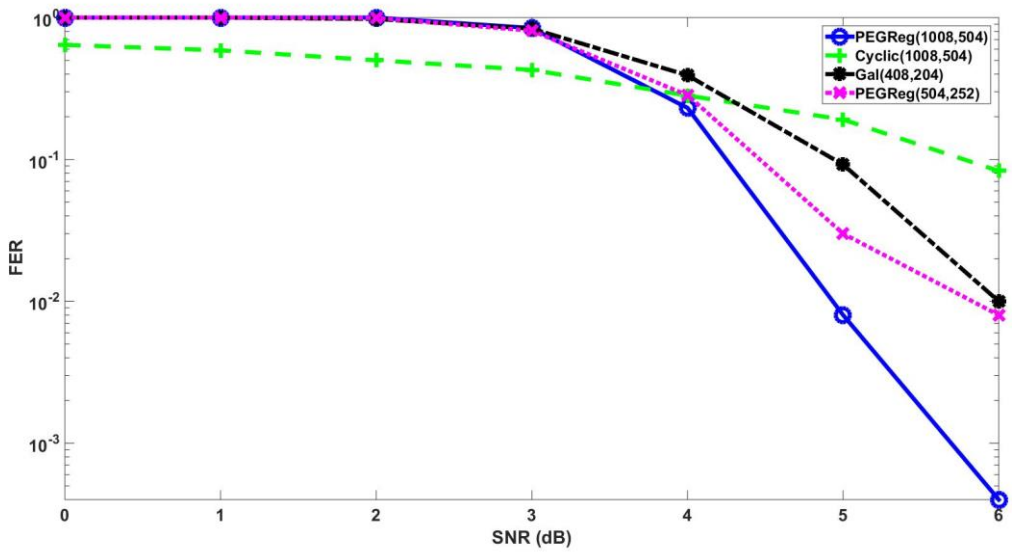


Figure 4. 7: AMTMBF FER analysis for different LDPC codes

On the other hand, the Cyclic codes seem to converge at a much faster rate when compared to the Progressive Edge Growth and Gallager codes. This results in a much lower average number of iterations (ANI) for low SNR regions. The Progressive Edge Growth and Gallager codes have a waterfall region upon transitioning from low to high SNR region. The Gal and PEG codes start to converge a lot faster than the Cyclic codes and this manifests in improved FER and BER performance when compared to the Cyclic codes.

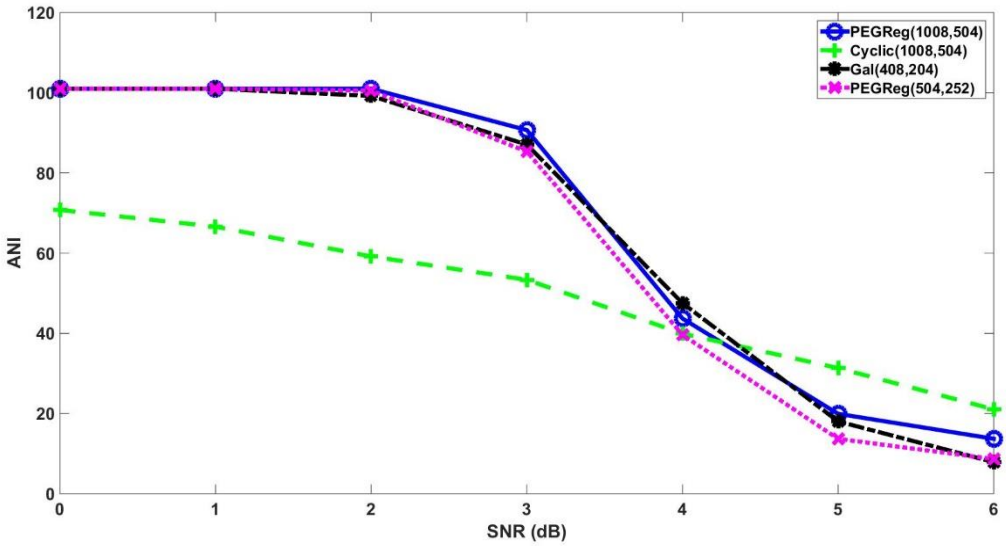


Figure 4. 8: AMTMBF FER analysis for different LDPC codes

Figure 4.6 and Figure 4.7 have shown that for the chosen subset of LDPC codes, the PEGReg (1008, 504) codes are the most suitable codes for the proposed algorithm followed by PEGReg (504, 252) then Gal (408,204) and then lastly Cyclic (1008, 504) codes in terms of error correction capabilities. The rest of the simulations in this section for the AMTMBF will make use of PEGReg (1008, 504) codes.

The proposed algorithm, the AMTMBF uses certain building blocks that exist in other algorithms that have been discussed in the literature survey namely the IMWBF, AWMBF and the SBF. The proposed algorithm is simulated against these algorithms to establish if there is any performance improvement to be gained from the concepts behind the design of the AMTMBF.

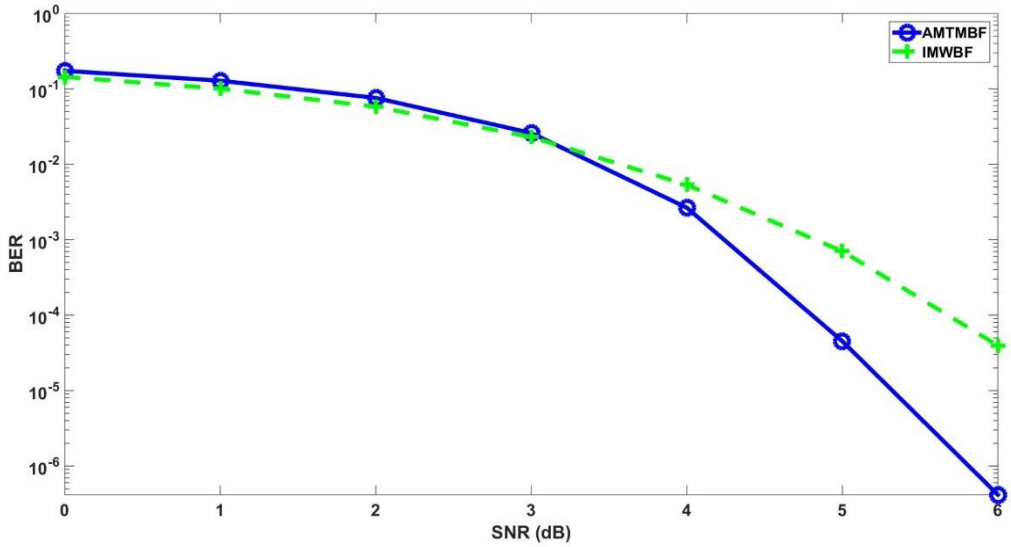


Figure 4. 9: AMTMBF vs IMWBF BER performance analysis

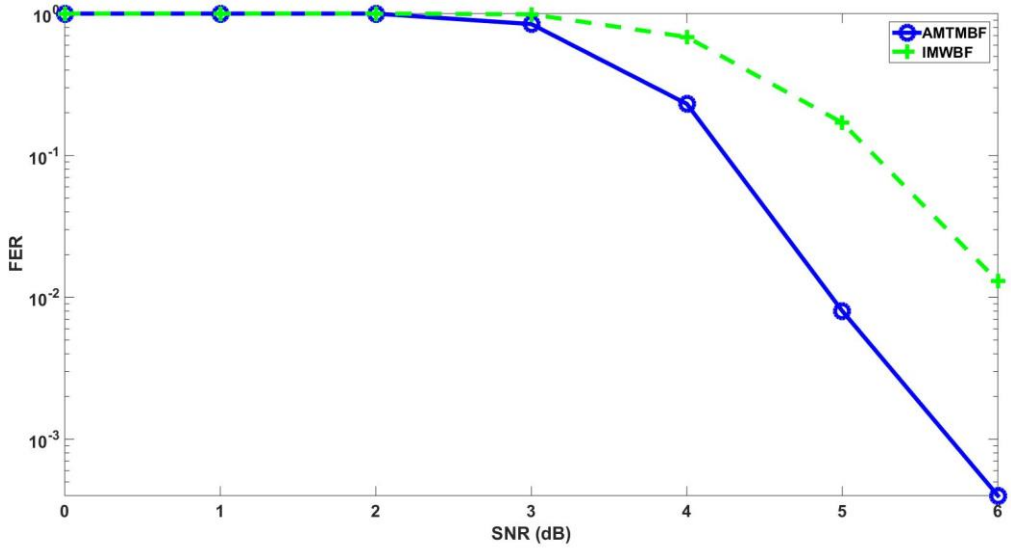


Figure 4. 10: AMTMBF vs IMWBF FER performance analysis

Figure 4.9 and Figure 4.10 depicts the BER and FER curve of the proposed multi-bit flipping AMTMBF versus the single-bit flipping IMWBF algorithm. The AMTMBF uses the IMWBF inversion function to calculate the reliability of the codeword bits. The IMWBF has a slight improvement over the AMTMBF at low SNR and this is because at low SNR values, the correct and incorrect inversion function values are indistinguishable thus the IMWBF, being a single bit flipping algorithm, the chances of erroneously inverting a correct bit are much less than the AMTMBF which may introduce more errors per iteration.

At higher SNR values, the correct and incorrect inversion values are more separated. The set of incorrect bits have a much higher probability of having the largest inversion function values while the opposite is true for the correct bits. This enables the algorithm to accurately select and correct the incorrect bits.

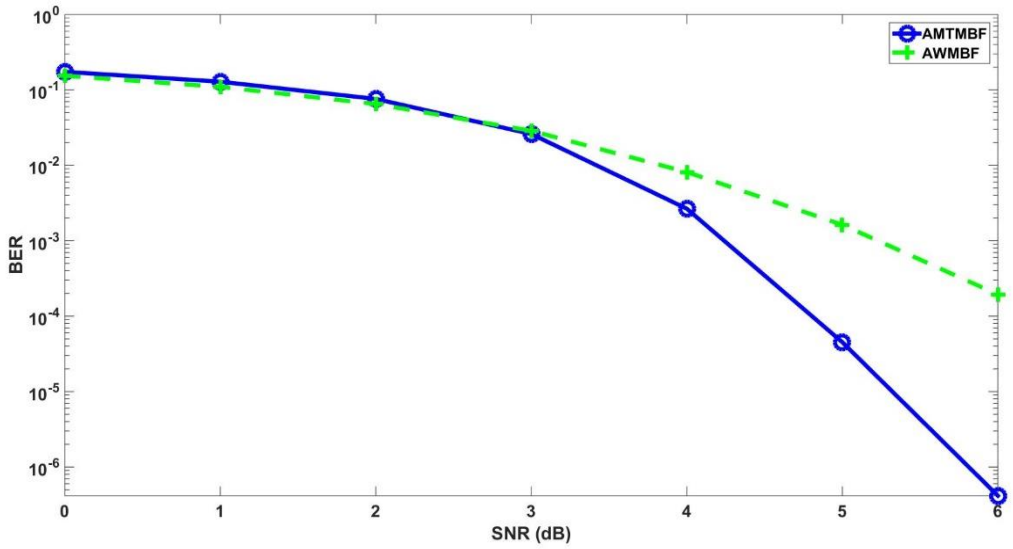


Figure 4. 11: AMTMBF vs AWMBF BER performance analysis

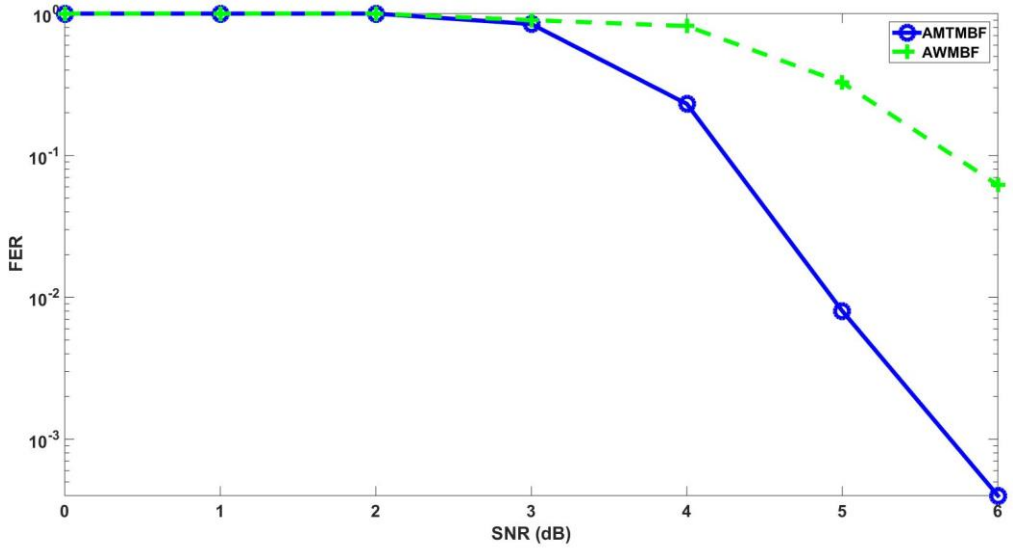


Figure 4. 12: AMTMBF vs AMWBF FER performance analysis

The AWMBF which forms the foundation of the adaptive threshold technique is simulated against the proposed AMTMBF and the results are displayed in Figure 4.11 and Figure 4.12. The AWMBF is a multi-bit, single adaptive threshold algorithm while the AMTMBF is a multi-bit, multi-adaptive threshold algorithm. At SNR values greater than 2.5 dBs the performance

gap between the two algorithms starts to widen with clear gains being observed in the performance of the AMTMBF. The AMTMBF improvements can be attributed to the use of two adaptive thresholds and the incorporation of soft information modification to reflect the level of confidence in the inversion function.

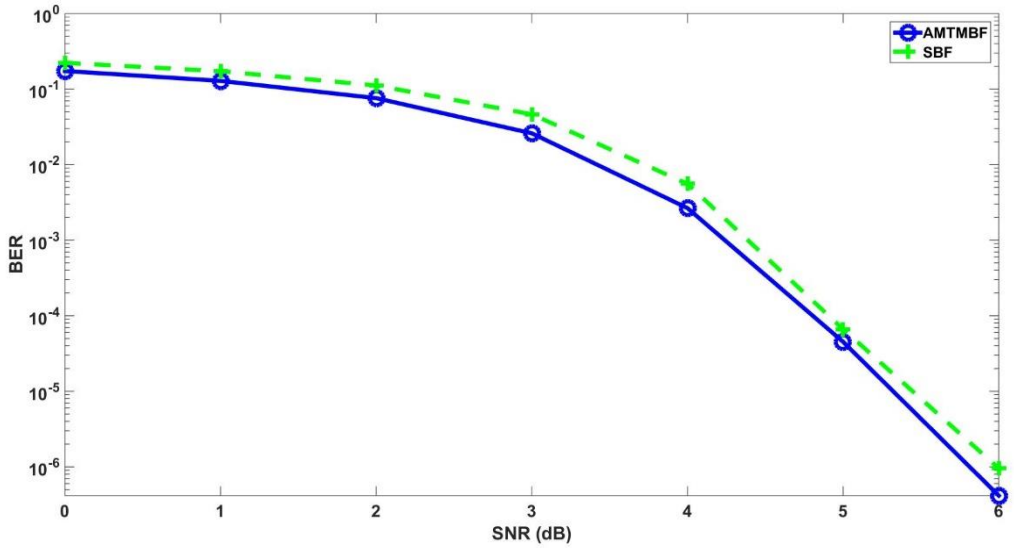


Figure 4. 13: AMTMBF vs SBF BER performance analysis

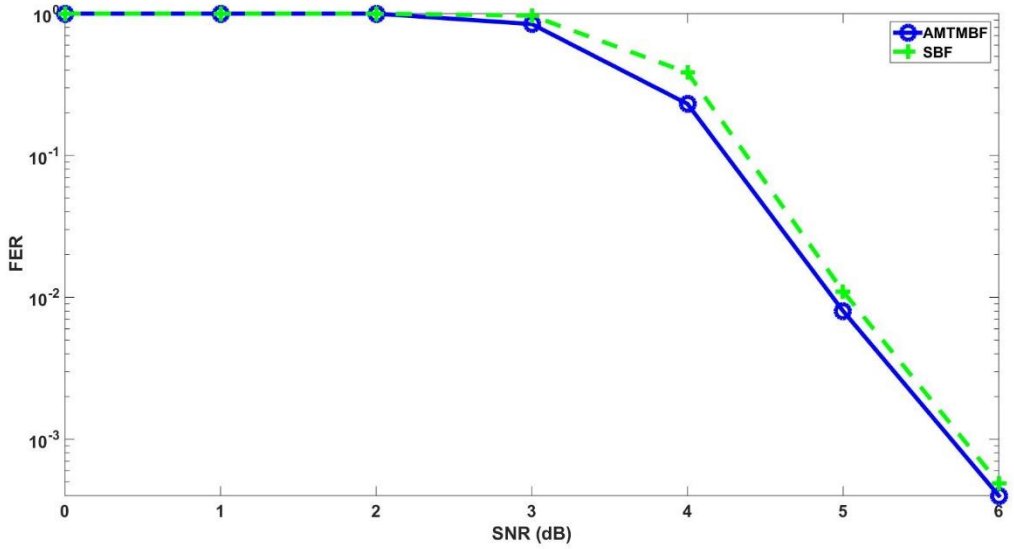


Figure 4. 14: AMTMBF vs SBF FER performance analysis

Figure 4.13 and Figure 4.14 shows the simulation comparison between the AMTMBF and the SBF. The SBF has multiple fixed thresholds but also incorporates soft information modification

to improving decoding performance. The AMTMBF has a marginal performance advantage over the multi-fixed threshold SBF in the SNR region simulated. The slight gain in performance is attributed to the multiple adaptive nature of the AMTMBF's threshold scheme which is responsible for correctly identifying erroneous bits and correcting them.

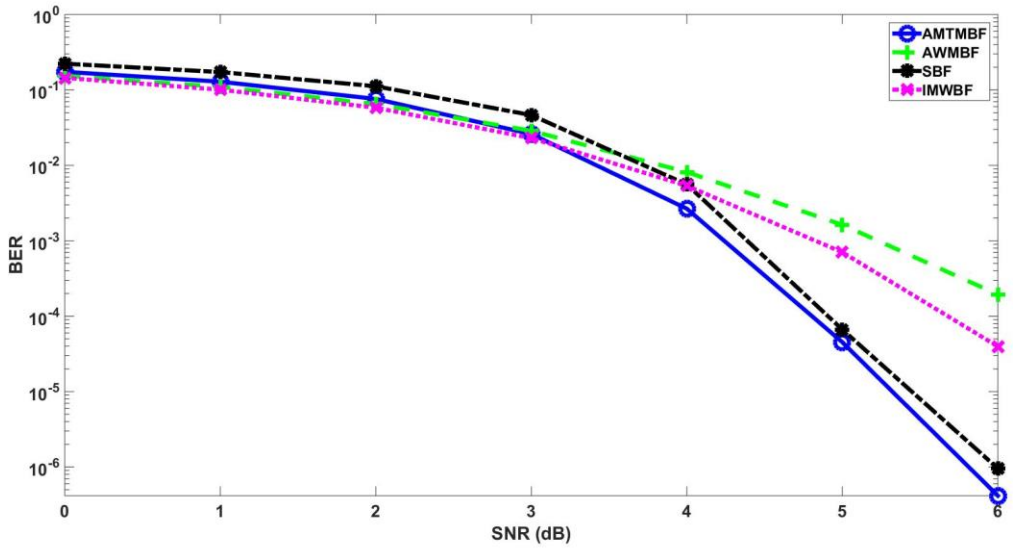


Figure 4. 15: AMTMBF vs IMWBF vs AWMBF vs SBF BER performance analysis

Figure 4.15 shows the comparative BER performance of the AMTMBF, AWMBF, SBF and the IMWBF algorithms.

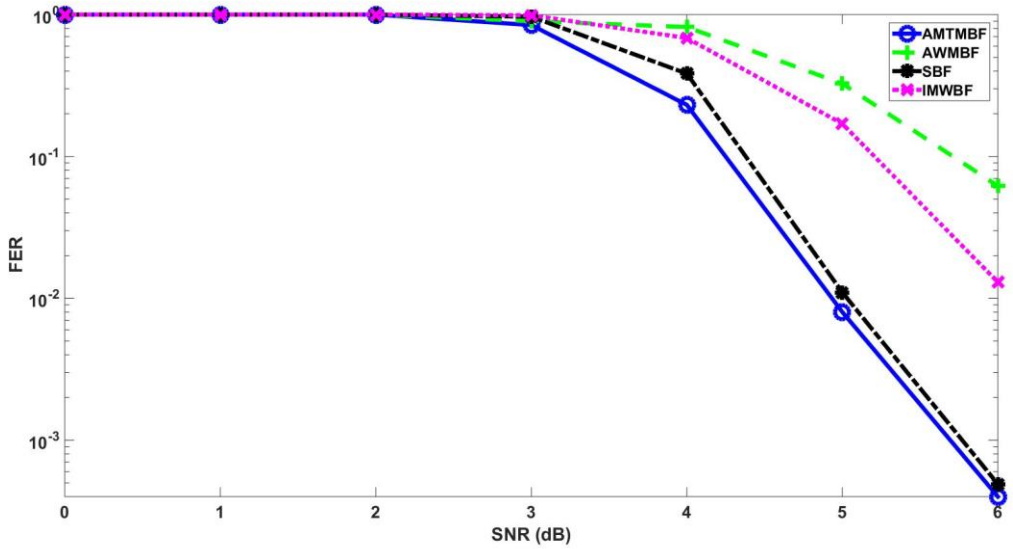


Figure 4. 16: AMTMBF vs IMWBF vs AWMBF vs SBF FER performance analysis

Figure 4.16 shows the frame error rate comparison of the AMTMBF, AWMBF, SBF and the IMWBF algorithms. The AMTMBF algorithm performs better than the IMWBF, AWMBF and the SBF for all SNR.

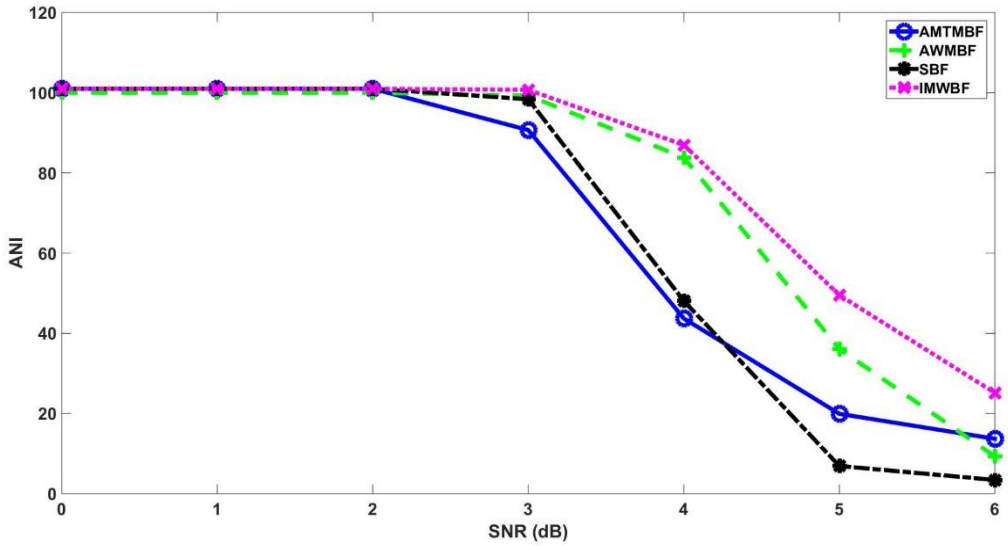


Figure 4. 17: AMTMBF vs IMWBF vs AWMBF vs SBF ANI performance analysis

The AMTMBF converges faster than the IMWBF and the AWMBF for a better part of the SNR range. Notably, at 4 dBs, the AMTMBF requires only about 50% of the iterations required by both the IMWBF and the AWMBF. The SBF however starts to converge faster than the proposed algorithm for SNR values greater than 4.2 dBs.

4.6 Discussion

The proposed algorithm, the adaptive multi-threshold multi-bit flipping (AMTMBF) algorithm incorporates several but modified building blocks from pre-existing algorithms namely the IMWBF, AWMBF and the SBF which have all been discussed in the literature survey. The AMTMBF has been simulated with various types of LDPC codes and it has been determined that regular progressive edge growth (PEGReg) codes provide the best performance.

From Section 4.5 it can be seen that the proposed algorithm shows a significant decoding performance improvement when compared to the single-bit flipping improved modified weighted bit flipping (IMWBF) algorithm. The AMTMBF uses the IMWBF inversion function to calculate the reliability of the bits therefore this improvement can be attributed to two main factors. The first factor is that the proposed algorithm is a multi-bit flipping algorithm therefore

it can correct multiple bits at the same time and therefore it converges at a much faster rate than the IMWBF. The second factor is the modification of the way, the inversion function is calculated. The inversion function of the AMTMBF incorporates modified soft information to improve the calculation of the reliability values.

The proposed algorithm also presents a significant performance advantage when compared to the adaptive weighted modified bit flipping (AWMBF) algorithm. The AMTMBF has multiple adaptive thresholds while the AWMBF only has one threshold. The AMTMBF thresholds are also determined differently from the AWMBF method. At a BER of 10^{-5} and FER of 10^{-3} the proposed algorithm had over 1 dBs performance advantage. This performance advantage was also realised in the algorithm complexity at high SNR where the AMTMBF needed about 50 % average number of iterations to converge.

The proposed algorithm has a slight performance advantage when compared to the SBF algorithm. The SBF has fixed thresholds that determine the decoding of the codewords. The SBF like the AMTMBF also incorporates soft information modification to improve decoding performance. Beyond the signal to noise ratio of 3 dBs, the AMTMBF maintains a slight performance gap ahead of the SBF. At high SNR the AMTMBF has a higher ANI because it is difficult to distinguish the correct bits from the incorrect bits due to the very small noise induced variations in the soft information. Therefore, the AMTMBF tends to "hunt" for erroneous bits whose values are similar to those of the correct bits so in some cases it erroneously flips correct bit thereby introducing errors which are later corrected in subsequent iterations resulting in a higher ANI.

In conclusion, the AMTMBF presents a strong argument for adaptive thresholds. Multi-bit, multi-adaptive threshold techniques do improve the decoding capabilities of LDPC decoding algorithms as evidenced by the simulation results in which the proposed algorithm is compared to existing algorithm whose building blocks are incorporated into the proposed algorithm. Simulations show that, the proposed algorithm performs better than the existing algorithms in terms of BER and FER. The SBF seems to be preferable in terms of ANI at high SNR.

5. Near Optimal SNR Dependent Threshold Multi-bit Flipping Algorithm (NOSMBF)

5.1 Introduction

In this chapter an SNR dependent multi-threshold, multi-bit flipping algorithm is proposed with improved decoding capabilities algorithm in terms of bit error rate (BER), average number of iterations (ANI) and complexity. The proposed algorithm is designated the Near Optimal SNR Dependent Threshold Multi-bit Flipping Algorithm (NOSMBF). The algorithm is a study into the relationship between multi-bit flipping thresholds and additive white Gaussian signal to noise ratio.

The discussion in Section 3.2 shows that the majority of the multi-bit flipping algorithms make use of fixed thresholds that are obtained from exhaustive simulation. The multi-GDBF in [39] uses two predetermined thresholds to flip bits. The main threshold θ_1 is the primary threshold, while the second threshold θ_2 is activated to extricate the algorithm from oscillations. The SBF in [12] makes use of three different thresholds which divide the reliability measurements into four sections such that the soft bit information is modified with respect to the section in which the reliability measurement falls.

As explained at length in the previous chapter, multi-bit flipping algorithms make use of thresholds to select the bits to flip or to maintain. There exist two main methods of setting thresholds, namely the adaptive threshold and the fixed threshold. The NOSMBF uses a fixed threshold for a given SNR value. This threshold is termed the near optimal threshold because it is determined by finding the threshold with the least number of errors for a particular SNR.

The study of SNR dependant thresholds also provides a heuristic approach with which to determine near optimal thresholds given for all SNR values given an LDPC parity check matrix.

The NOSMBF has two thresholds and these are the primary and the secondary thresholds. These thresholds serve different purposes in the decoding process. The primary threshold determines the set of bits to flip while the secondary threshold determines the set of bits to enhance. Both thresholds are used to determine the bits to maintain.

The NOSMBF also uses the channel soft information modification scheme since the inversion function uses a weighting factor obtained from channel soft information. This scheme has been

shown to be effective in [46] and [12]. A custom soft information modification scheme has been introduced in Section 4 and this scheme is employed in the NOSMBF.

5.2 SNR Dependent Threshold Scheme

In the single bit flipping scheme, a flipping metric based on the parity check sum and received channel information is calculated for each bit in the codeword. Depending on the manner in which the metric is calculated, the bit with the largest or smallest metric is flipped iteratively until a correct codeword is found or until the maximum number of iterations has been reached.

In a multi-bit flipping scheme, the flipping metric is also calculated but multiple bits are flipped per iteration based on a given criterion. One criterion is to make use of thresholds such that a set of bits can be flipped based on the value of their flipping metric with respect to that given threshold. Therefore, bits can be flipped or have the soft information modified based on a given threshold as shown in [12].

In [12] the high throughput soft bit flipping decoder makes use of three fixed thresholds whose values are determined by analysing the statistical distribution of the inversion function values obtained from the modified Improved Modified Weighted Bit-Flipping algorithm. In [12] correct and incorrect bit inversion function values are obtained and the probability density function (PDF) for these values is obtained. The three flipping thresholds which determine how the soft bit information is flipped are obtained by analysing the PDF.

The rational for determining thresholds is the existence of a pattern or a characteristic associated with each bit in the codeword that points to the likelihood of a bit being correct or incorrect. Therefore, certain thresholds derived mostly through exhaustive simulations as shown in [12] are derived and bits or associated channel information is modified with the aim to correct any errors in the codeword.

The flipping thresholds in this text are a development from the algorithm in Section 4 which makes use of two thresholds that are used to modify channel information. Instead of using adaptive threshold equations, this new scheme makes use of predetermined fixed thresholds as seen in [12]. These thresholds are obtained by simulating a large number of codewords with known error patterns so as to get a distribution of the inversion function values.

5.2.1 Primary Flipping Thresholds

Consider a valid codeword $\boldsymbol{c}$ of length N where $c_n \in \boldsymbol{c}$ and a hard decision length N received codeword $\boldsymbol{z}$, where $z_n \in \boldsymbol{z}$ is an Additive White Gaussian Noise corrupted version of $\boldsymbol{c}$. Using the IMWBF inversion function in Equation 3.11, the tuple $\boldsymbol{E} = \{E_1, E_2, E_3, \dots, E_N\}$ from where $E_{max} = \max(\boldsymbol{E})$ and $E_{min} = \min(\boldsymbol{E})$ are obtained. Let E_i be a value that lies between E_{max} and E_{min} such that $E_{min} \leq E_{th} \leq E_{max}$.

Consider also an all-zero codeword $\boldsymbol{p}$ where $p_n \in \boldsymbol{p}$ where p_n is flipped to a 1 if and only if the associated inversion function value $E_n \geq E_{th}$. Therefore, the codeword $\boldsymbol{p}$ is a function of E_i such that the error pattern $\boldsymbol{e}$ of the received codeword $\boldsymbol{z}$ as a function of E_i is obtained as shown in Equation 5.1 to be $\boldsymbol{e}(E_i)$. The term $\boldsymbol{p}_{\boldsymbol{n}(E_i)}$ in Equation 5.1 refers to positions in the vector $\boldsymbol{p}$ that meet the condition $\boldsymbol{p}_{n(E_i)} = \{z_n = 1, E_n \geq E_{th} | z_n = 0, E_n \geq E_{th}, 1 \leq n \leq N\}$ such that

$$\boldsymbol{e}(E_i) = (\boldsymbol{z} + \boldsymbol{p}_{\boldsymbol{n}(E_{th})} + \boldsymbol{c}) mod 2 \tag{5.1}$$

Hence, the total number of errors for a codeword in variation with E_{th} is calculated as shown in Equation 5.2 since for bit positions which meet the condition $E_n \geq E_{th}$. The relationship between the total number of errors and the error vector is $e = H(\boldsymbol{e})$, such that the new relationship is $e(E_i) = H\big(\boldsymbol{e}(E_i)\big)$.

$$e(E_i) = \sum_{n=1}^{N} \big(z_n + p_{n(E_{th})} + c_i\big) mod 2 \tag{5.2}$$

The Figure 5.1 and Figure 5.2 shows the curve of $e(E_{th})$ against E_{th} for a random codeword at varying SNR where the optimal flipping threshold E_{fth} for that codeword is the minimum of the curve, as shown.

$$E_{fth} = min\big(\varepsilon(E_{th})\big) \tag{5.3}$$

To obtain the optimal flipping threshold, E_{FTH} for a particular signal to noise ratio (SNR), a sufficiently large number of codeword curves are obtained and averaged such that if R codewords are simulated, the optimal threshold is obtained as follows:

$$E_{FTH} = \frac{1}{R}\sum_{r=1}^{R} E_{fth_r} \tag{5.4}$$

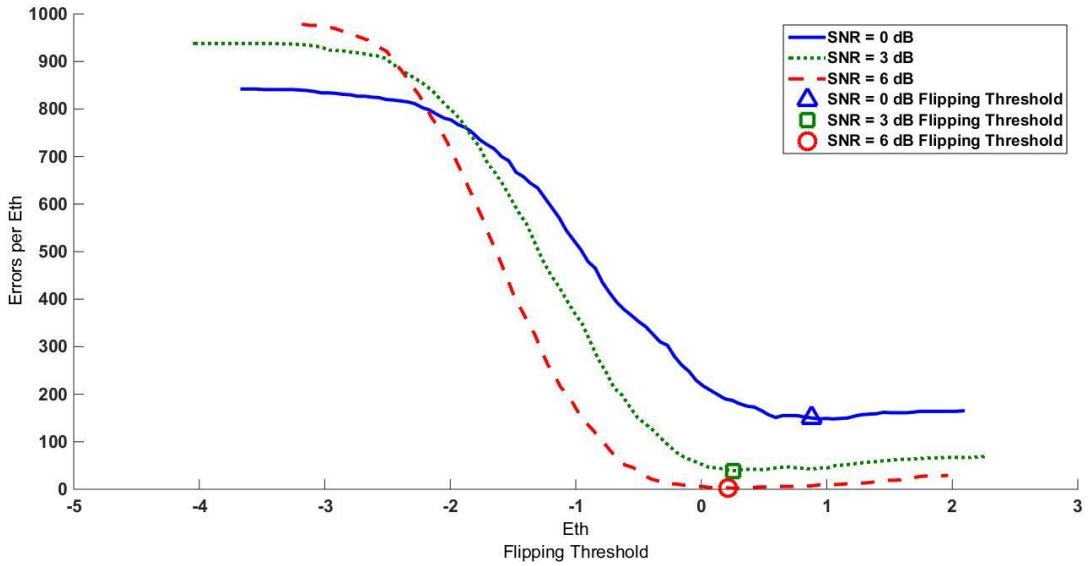


Figure 5. 1: Determining the flipping threshold at varying SNR values for PEGReg (1008, 504)

Figure 5.1 and Figure 5.2 show the flipping threshold plots for PEGReg (1008, 504) and PEGReg (504, 252) respectively for varying SNR values. The curves have similar profiles starting from E_{min} to show that flipping bits at that value of E_{th} will result in the largest number of errors. The curve gradually drops with increase in E_{th} towards E_{max} because the more you increase the value of E_{th} the more you accurately identify incorrect bits and ignore the correct ones. However, the curve has a minimum at which the most number of bits are corrected and this is referred to as the flipping threshold and moving the E_{th} beyond this point increases the number of errors because some errors will be ignored in the flipping process. At $E_{th} = E_{max}$ no bits are flipped and therefore, all the errors that where originally in the codeword remain.

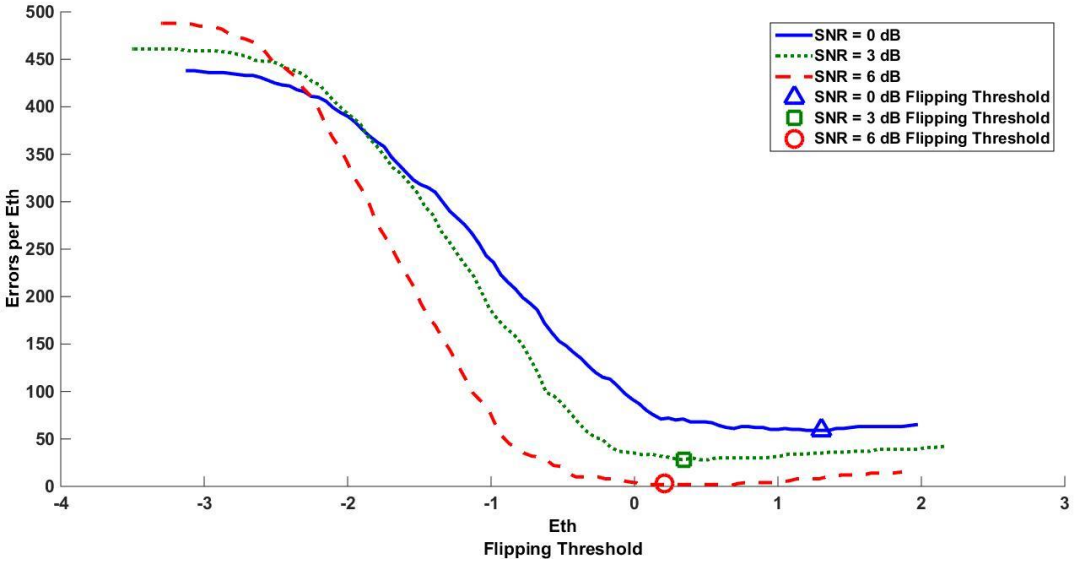


Figure 5. 2: Determining the flipping threshold at varying SNR values for PEGReg (504, 252)

The Figure 5.3 shows the flipping thresholds for PEGReg (1008, 504) and PEGReg (504, 252) obtained for SNR region ranging from 0 dBs to 6dBs. The profile of the two curves is similar and it is apparent that there is an almost linear relationship between the flipping threshold and SNR values with a negative gradient.

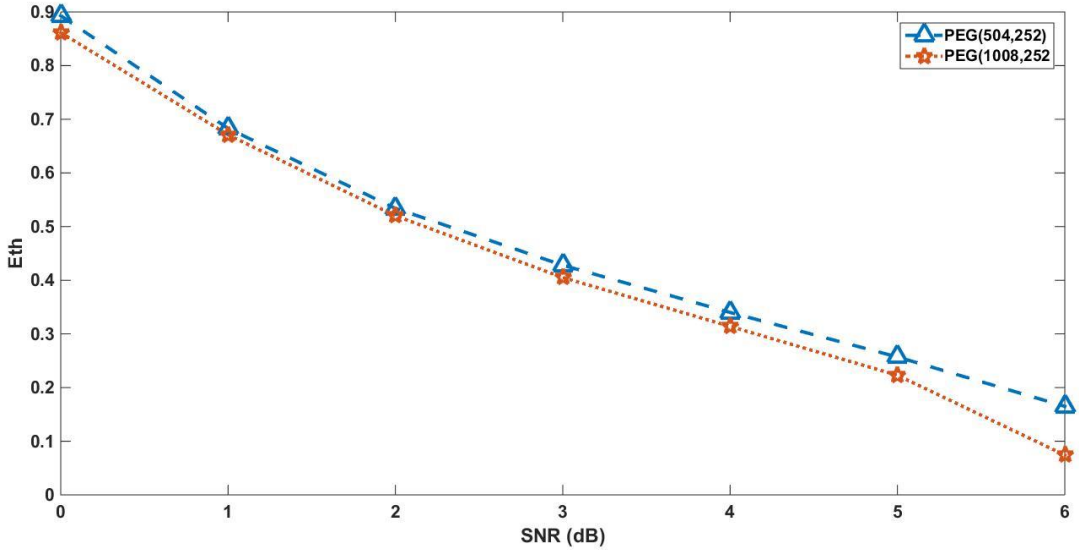


Figure 5. 3: Flipping thresholds for varying LDPC codes

5.2.2 Secondary Flipping Threshold

Given that the channel information has a significant role in determining the flipping metric, it can be observed that enhancing the value of the channel information of the bit that is perceived to be correct reduces the likelihood of a correct bit being incorrectly flipped as well as increasing the chance of an incorrect bit being identified and corrected through the metric. Therefore, a secondary threshold known as the strengthening threshold is also calculated to identify these correct bits according to these rules: In a weighted bit flipping algorithm, the channel information is involved in the computation of the inversion function value as a scaling factor [27] therefore all the channel information associated with correct bits can be scaled up in some manner to increase the impact reliable bits contribute to the computation of the inversion value [12].

Given the same codewords $\boldsymbol{c}$ and $\boldsymbol{z}$ considered in Equation 5.1, the threshold E_{th} is varied from E_{min} and E_{max} taking note of which bits if strengthened contribute to errors in the codeword. The term $\boldsymbol{z}_{n(E_{th})}$ in the following equations refers to positions in the vector $\boldsymbol{z}$ that meet the condition $\boldsymbol{z}_{n(E_{th})} = \{z_n = z_n, E_n \leq E_{th} | z_n = 0, E_n > E_{th}, n, 1 \leq n \leq N\}$.

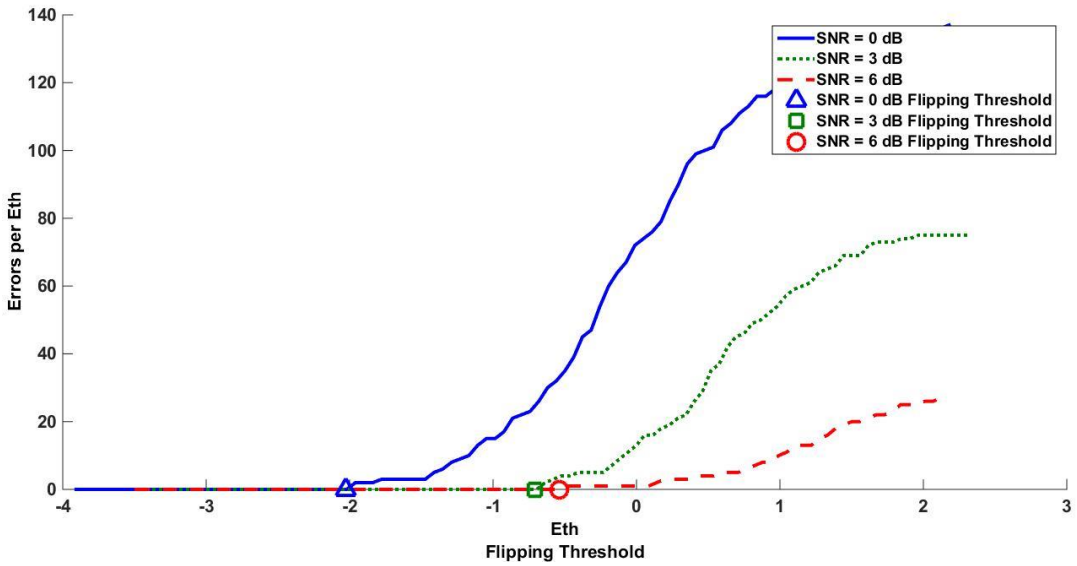


Figure 5. 4: Determining the strengthening threshold for PEGReg (1008, 504)

The Figures 5.4 and Figure 5.5 show the strengthening threshold curves for PEGReg (1008, 504) and PEGReg (504, 252) respectively. It can be seen that there are zero errors as you increase E_{th} from the minimum inversion function value E_{max}, there are zero errors up to a certain value $E_{th} = E_{sth}$ where errors start to appear. This value E_{sth} also known as the strengthening threshold is the largest number of non-erroneous bits the scheme can identify

using the E_{th}. These are the bits the algorithm has the highest confidence in their correctness. These are the bits whose soft information the algorithm can enhance to influence the inversion function calculations in subsequent iterations. In this case, the modification involves the addition of a strengthening offset value hence the name strengthening threshold.

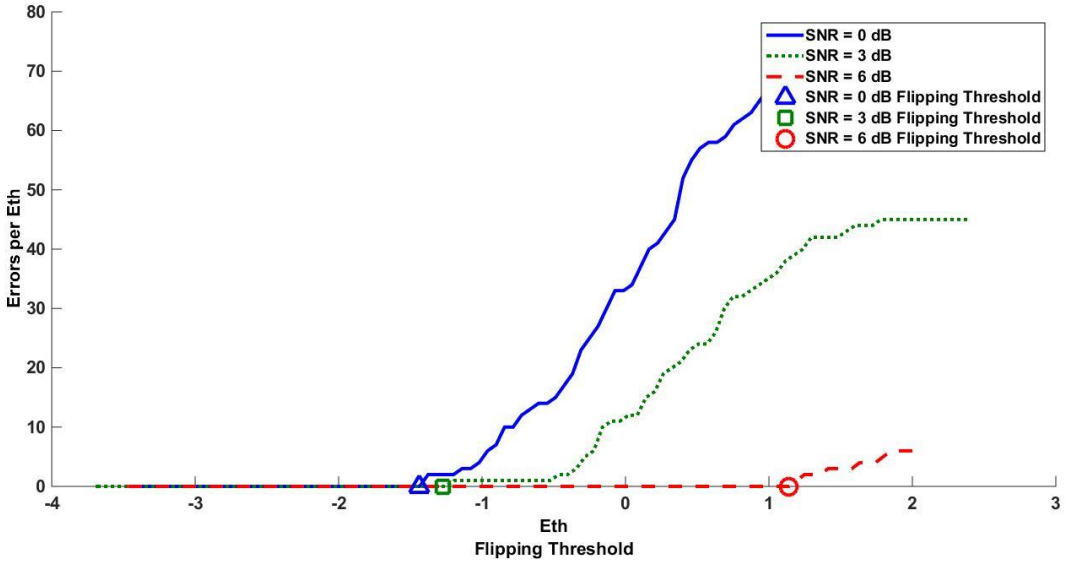


Figure 5. 5: Determining the strengthening threshold for PEGReg (504, 252)

The Figure 5.6 shows the strengthening thresholds for PEGReg (1008, 504) and PEGReg (504, 252) from 0 dBs to 6 dBs. It can be seen that the strengthening threshold has a positive gradient linear relationship with the SNR.

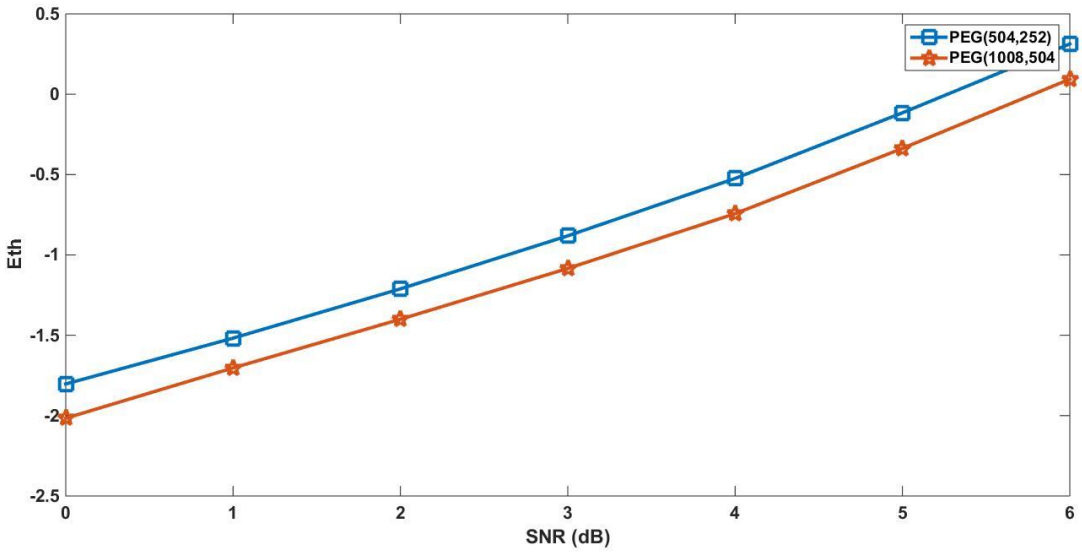


Figure 5. 6: Strengthening thresholds for varying LDPC codes

$$\boldsymbol{e}(E_i) = \left(\boldsymbol{c} + \boldsymbol{z}_{n(E_i)}\right) mod 2 \tag{5.5}$$

To obtain the secondary threshold, consider the valid codeword $\boldsymbol{c}$ and the noise corrupted received codeword $\boldsymbol{z}$. Obtain the tuple $\boldsymbol{E}, E_n \in \boldsymbol{E}$, E_{min} and E_{max} for the received codeword. Obtain the mod 2 addition of bits whose positions are identical to those in tuple $\boldsymbol{E}$ which meet the condition $E_n \leq E_i$ as shown in Equation 5.5. Equation 5.6 is used to obtain the total number of errors. The Figure 5.4 and Figure 5.5 shows a plot of $\varepsilon(E_i)$ against E_i for random codewords at varying SNR.

$$e(E_i) = \sum_{n=1}^{N} \left(c_n + z_{n(E_i)}\right) mod 2 \tag{5.6}$$

To obtain the optimal secondary threshold per codeword, consider the subset of all E_i for which $\varepsilon(E_i) = 0$ as $E_{i|\varepsilon(E_i)=0} = \{E_i \in \boldsymbol{E} | \varepsilon(E_i) = 0\}$. The optimal threshold is the largest value in the subset $E_{i|\varepsilon(E_i)=0}$ as shown in equation.

$$E_{sth} = max\left(E_{i|\varepsilon(E_i)=0}\right) \tag{5.7}$$

To obtain the optimal strengthening threshold, E_{STH} for a particular signal to noise ratio (SNR), a sufficiently large number of codeword curves are obtained and averaged such that if R codewords are simulated, the optimal threshold is obtained as follows:

$$E_{STH} = \frac{1}{R} \sum_{r=1}^{R} E_{sth_r} \tag{5.8}$$

The Figure 5.3 and Figure 5.6 shows the flipping and strengthening threshold plots respectively. This data has been obtained for rate 0.5 MacKay PEG codes (1008, 504).

$$E_{FTH}(Y_{SNR}) = -0.1156 Y_{SNR} + 0.8105 \tag{5.9}$$

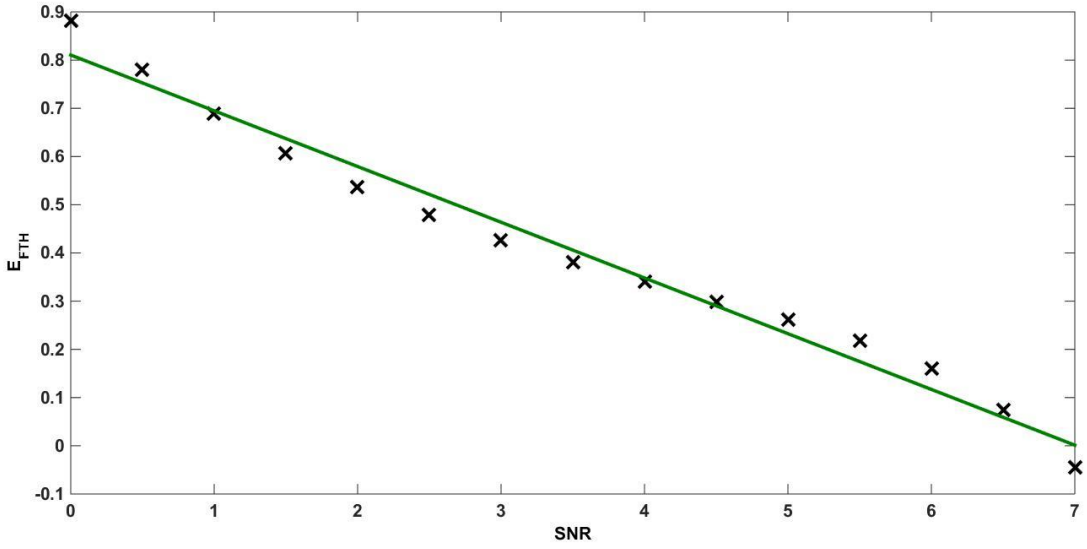


Figure 5. 7: Approximate flipping thresholds

$$E_{FTH}(Y_{SNR}) = 0.362Y_{SNR} - 2.111 \tag{5.10}$$

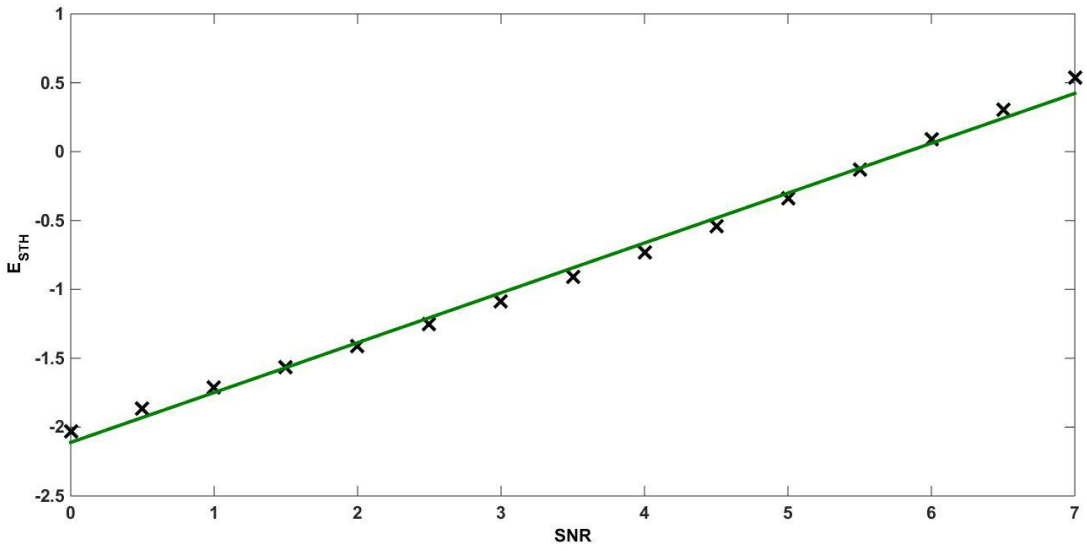


Figure 5. 8: Approximate strengthening thresholds

The flipping and the strengthening thresholds for the PEGReg (1008, 504) have been obtained via simulation and are shown in Figure 5.3 and Figure 5.6. The results obtained seem to suggest a linear relationship between the Thresholds and the signal to noise ratio. Curve fitting analysis

has been carried out to produce Equation 5.9 and 5.10 for the flipping and strengthening thresholds respectively. The plots are shown in Figures 5.7 and 5.8. The hypothesis is that anatomy of the Low Density Parity Check Matrix has a bearing on these thresholds in such a way that they can be derived from calculation as opposed to simulation.

5.2.3 Distribution of the inversion function values in a Random codeword

As described in Section 4, the inversion functions calculated for each bit in the codeword give a measurement of how reliable the hard decision is for a particular bit. Taking a random codeword transmitted via an AWGN channel with a known signal to noise ratio, the inversion function values where calculated and the original codeword was used as a control to obtain correct and incorrect hard decisions.

It has been observed that, for low SNR values, for the bits above the Flipping threshold, E_{th1} 30 % of the bits flipped are correct and therefore the algorithm introduces errors into the codeword while it corrects some of them.

Between, the flipping threshold E_{FTH} and the strengthening threshold E_{STH} lies the majority of the bits. At low SNR, the incorrect and correct bits' inversion function values are intermingled as to be indistinguishable using the thresholds.

However, the bits below the strengthening threshold E_{STH} which are considered by the algorithm to be correct are actually over 95 % correct and therefore, the algorithm does not enhance errors by strengthening this set of bits.

The distribution of inversion function values for high SNR values has been observed. In Figure 5.9 and Figure 5.10, over 75 % of the erroneous bits to be flipped have been correctly identified by E_{FTH} while over 90 % of correct bits to be strengthened have been identified by E_{STH}. It is evident that the higher the signal to noise ratio, the easier it is to distinguish the correct bits from the erroneous bits via the inversion function value which makes the flipping and strengthening threshold more effective in correcting errors

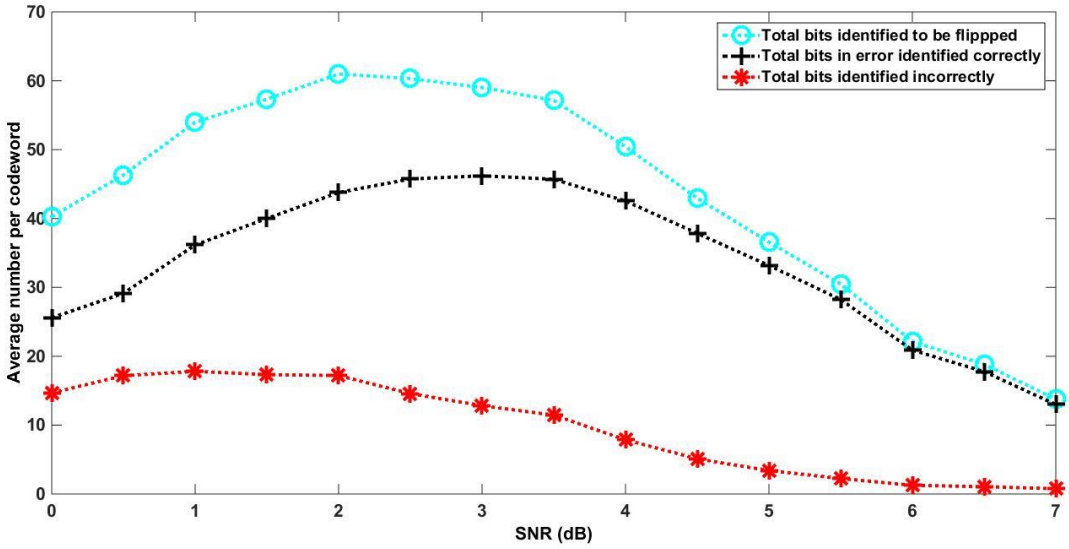


Figure 5. 9: Flipping threshold analysis

The plot in Figure 5.9 and Figure 5.10 shows the effect of the flipping and strengthening thresholds with respect to the SNR respectively. From 0 dBs to about 3 dBs it is observed that more than 60 % of the bits identified by the algorithm as bits to be flipped are correct and this increases to over 90 % as the SNR increases.

Figure 5.10 shows that the algorithm correctly picks the bits to strengthen almost 100 % of the time for all SNR simulated.

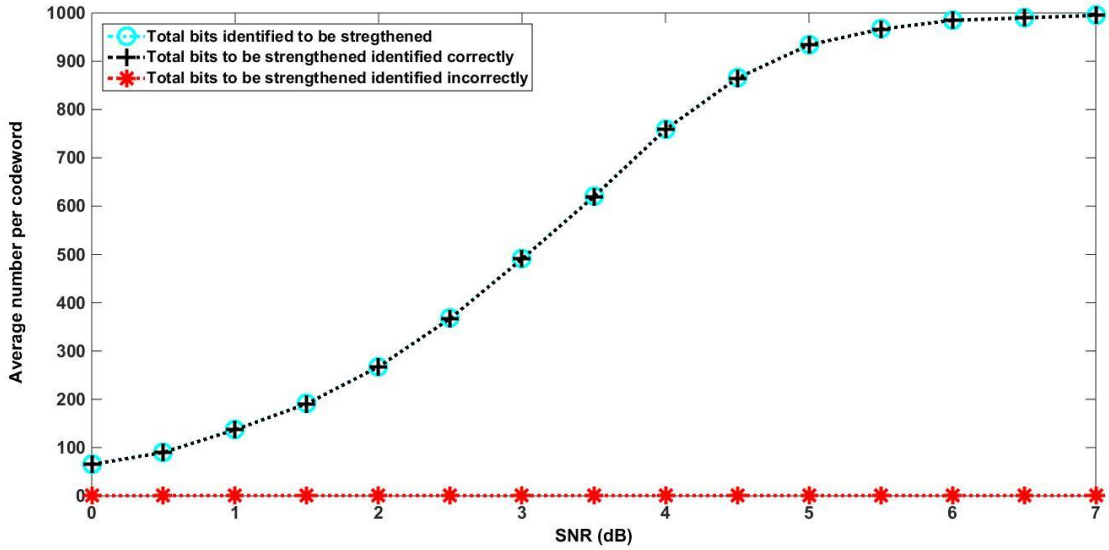


Figure 5. 10: Strengthening threshold analysis

5.3 Channel Information Modification Scheme

The channel soft information enhancement scheme has also been implemented in this algorithm to enhance its decoding capabilities. In the previous chapter, the soft information modification scheme has been implemented in the AMTMBF successfully. The purpose of multiple thresholds is to enable decoding of nodes with varying reliability values. Enhancing the codeword soft information provides the opportunity for the decoder to make incremental changes to the codeword based on the reliability information provided.

5.3.1 The proposed scheme

The proposed scheme is identical to the scheme described in Section 4.3.1.

i) Flip –

$$y_n^{(k+1)} = (-1).sgn\left(y_n^{(k)}\right).\left|y_n^{(k)}\right|$$

ii) Maintain –

$$y_n^{(k+1)} = sgn\left(y_n^{(k)}\right).\left|y_n^{(k)}\right|$$

iii) Strengthen –

$$y_n^{(k+1)} = sgn\left(y_n^{(k)}\right).\left|y_n^{(k)}\right| + |\delta|$$

5.3.2 Determining the enhancement value $|\delta|$

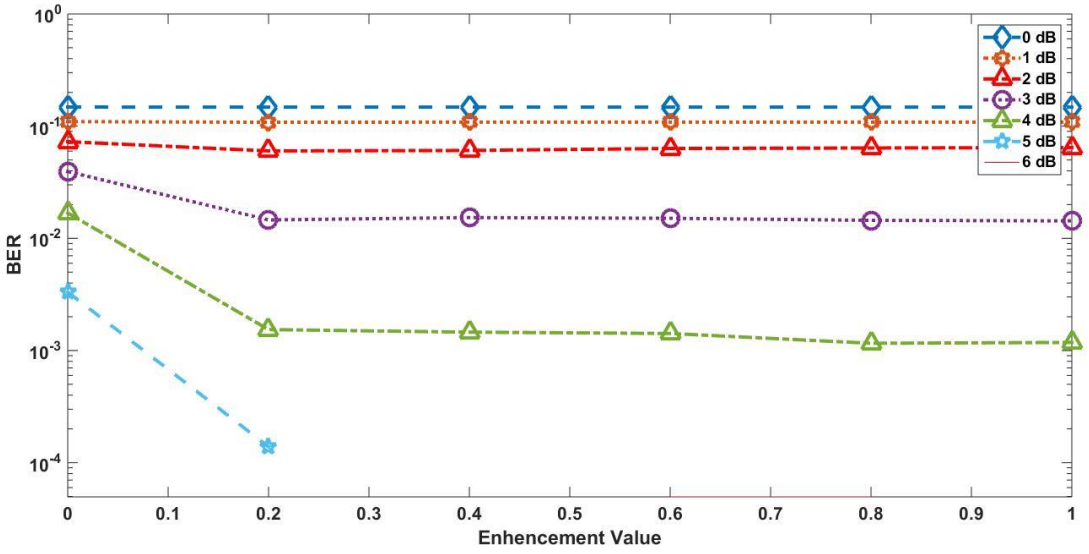


Figure 5. 11: Determining the optimal enhancement value |δ|

The Figure 5.11 shows the variation of the BER with the change in enhancement value for PEGReg (1008, 504) regular codes. It can be seen that the enhancement factor does not influence the BER for SNR vales below 2 dBs. An enhancement factor of over 0.2 has a significant effect on higher SNR values. Using larger enhancement values has no benefit and in some cases results in worse performance as shown in Figure 5.12.

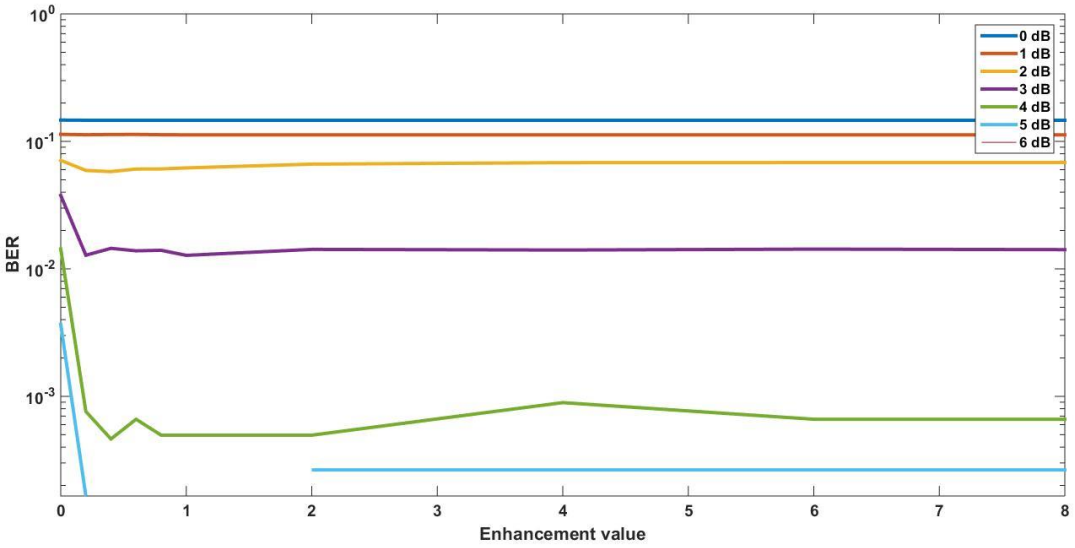


Figure 5. 12: Enhancement values investigation

5.4 Proposed Decoding Algorithm

This section is the conclusion of the design discussed in this chapter so far. The proposed algorithm is known as the Near Optimal SNR Dependent Threshold Multi-bit Flipping Algorithm (NOSMBF). The algorithm uses SNR dependant fixed multi-thresholds as well as soft information modification.

The full algorithm is described below:

Initialize: set $k = 0$, as well as $\boldsymbol{z^{(0)}} = \boldsymbol{z}$. For $1 \leq m \leq M$ and compute $min_{i \in \boldsymbol{N}(m) \backslash n} |y_i|$ and store.

1) While $k \leq k_{max}$, compute the syndrome $\boldsymbol{s^{(k)}} = \boldsymbol{z^{(k)}} \boldsymbol{H^{(T)}}$. If $\boldsymbol{s^{(k)}} = \boldsymbol{0}$, terminate algorithm and output decoded codeword $\boldsymbol{z^{(k)}}$.

2) Compute the vector $\boldsymbol{E^{(k)}} = \left(E_1^{(k)}, E_2^{(k)}, \dots, E_N^{(k)} \right)$ based on:

$$E_n \triangleq \sum_{m \in \boldsymbol{M}(n)} (2s_m - 1)\, min_{i \in \boldsymbol{N}(m) \backslash n} |y_i| - \alpha . |y_n|$$

3) Compute the upper flipping and lower strengthening threshold E_{th1} and E_{th2} respectively based on:

$$E_{th1} \triangleq E_{FTH}(Y_{SNR})$$

And

$$E_{th2} \triangleq E_{STH}(Y_{SNR})$$

4) Obtain $\boldsymbol{y}^{(\boldsymbol{k+1})} = \left(y_1^{(k)}, y_2^{(k)}, \ldots, y_N^{(k)}\right)$ on the basis of the following:

(a) Flip:

If $E_{FTH} \leq E_n^{(k)}$,

$$y_n^{(k+1)} = (-1).sgn\left(y_n^{(k)}\right).\left|y_n^{(k)}\right|$$

(b) Maintain:

If $E_{STH} \leq E_n^{(k)} < E_{FTH}$,

$$y_n^{(k+1)} = sgn\left(y_n^{(k)}\right).\left|y_n^{(k)}\right|$$

(c) Strengthen:

If $E_{STH} \leq E_n^{(k)}$,

$$y_n^{(k+1)} = sgn\left(y_n^{(k)}\right).\left|y_n^{(k)}\right| + |\delta|$$

5) $k \leftarrow k + 1$, if $k > k_{max}$ terminate algorithm otherwise go to step 1.

5.5 Simulation Results

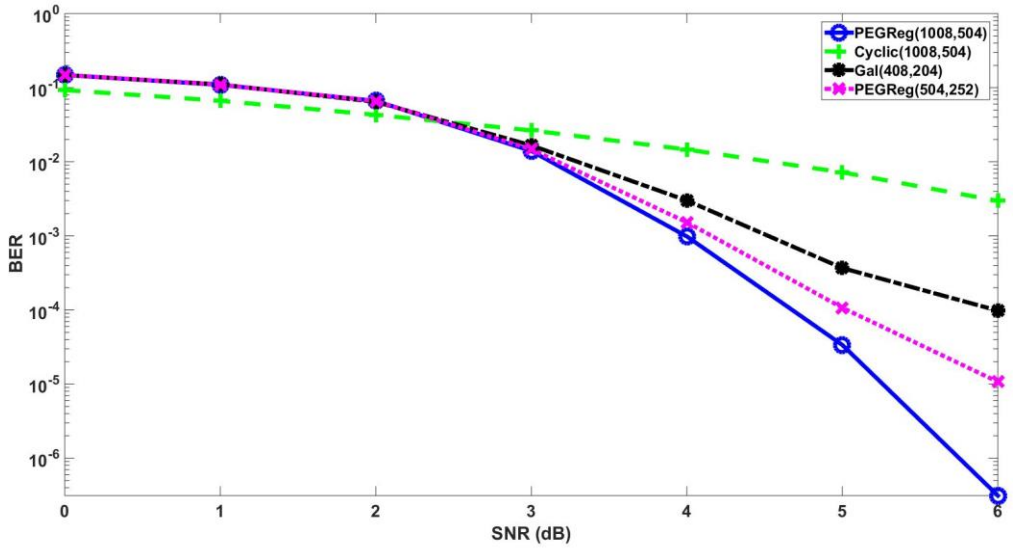


Figure 5. 13: NOSMBF BER analysis for different LDPC codes

The near optimal SNR dependent threshold multi-bit flipping (NOSMBF) algorithm is simulated via the BI-AWGN channel with the signal to noise ratio variation from 0 dBs to 6 dBs. The code symbols have been modulated using the BPSK modulation scheme. Matlab has been used to carry out all simulations.

As with the AMTMBF in the previous chapter, it is very important to determine the most effective codes with which the NOSMBF performs the best. A set of existing codes have been obtained and used to carry out the simulation.

The following subset of the most effective regular codes have been used to simulate the NOSMBF algorithm.

i) Geometric Cyclic codes – Cyclic (1008, 504)
ii) Progressive Edge Growth Regular codes – PEGReg(1008, 504)
iii) Gallager random codes – Gal (408, 204)
iv) Progressive Edge Growth Regular codes – PEGReg(504, 252)

These codes where acquired from [17].

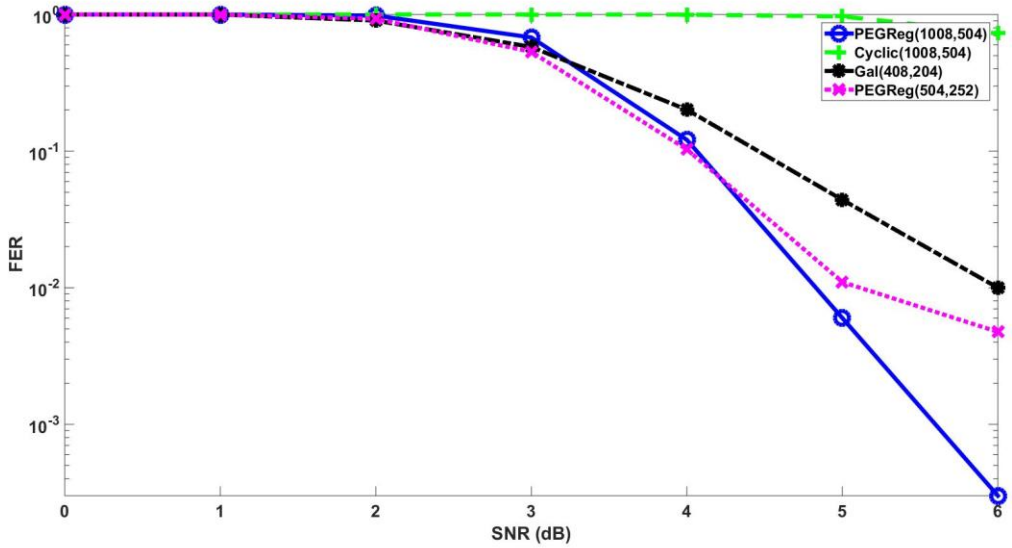


Figure 5. 14: NOSMBF FER analysis for different LDPC codes

The BER and FER of the NOSMBF when decoding using different LDPC codes are shown in Figure 5.13 and Figure 5.14 respectively. The best performance of the proposed algorithm is obtained when decoding with PEGReg (1008, 504) codes especially from the SNR region of 2.5 dBs and beyond. From 0 dBs to about 2 dBs, Cyclic (1008, 504) codes provide the best error correction performance while all the other codes have relatively similar performance. The PEGReg (504, 252) codes provide the second best performance in the high SNR region. However error floors start to appear on the FER plot from 5 dBs onward which seems to suggest that the longer the PEGReg codes are, the more suitable they are. The second from last performing LDPC codes are the random Gallager Gal (408, 204) codes which are amongst the first LDPC codes ever devised. The worst performing codes, in overall for the NOSMBF are the geometric Cyclic (1008, 504) which are the same length as the PEGReg (1008, 504). In most cases, high weight cyclic codes are considered to be the best codes for bit-flipping algorithms but in this case they are not because of the manner in which the high weight exaggerates the soft weight incorporated into the inversion calculation. This is as a result of the way in which the NOSMBF modifies the soft information.

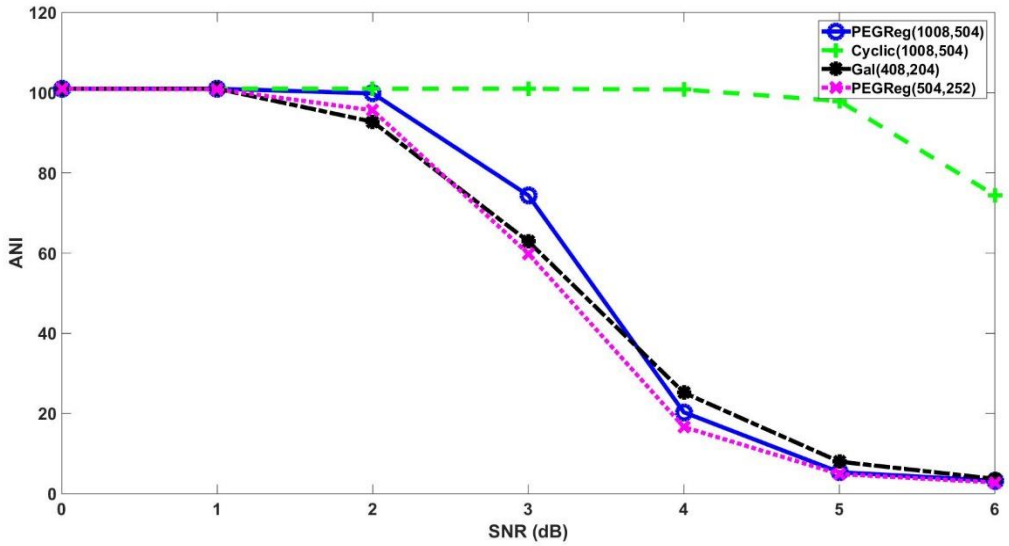


Figure 5. 15: NOSMBF ANI analysis for different LDPC codes

The best BER and FER performing codes, the PEGReg (1008, 504) results in the second worst ANI performance for a better part of the signal to noise ratio simulation range when compared to the PEGReg (504, 252) and Gal (408, 204) codes. This could be due to the oscillations that might occur before the algorithm converges to a correct codeword. The soft information modification assists as an oscillation escape mechanism in some cases because reliability values change with every iteration which may influence flipping which would be impossible to accomplish with static reliability values. The PEGReg (504, 252) codes provide the best ANI performance for a better part of the SNR range. The Cyclic (1008, 504) codes have the worst ANI performance for all SNR values and are therefore unsuitable for use with the NOSMBF.

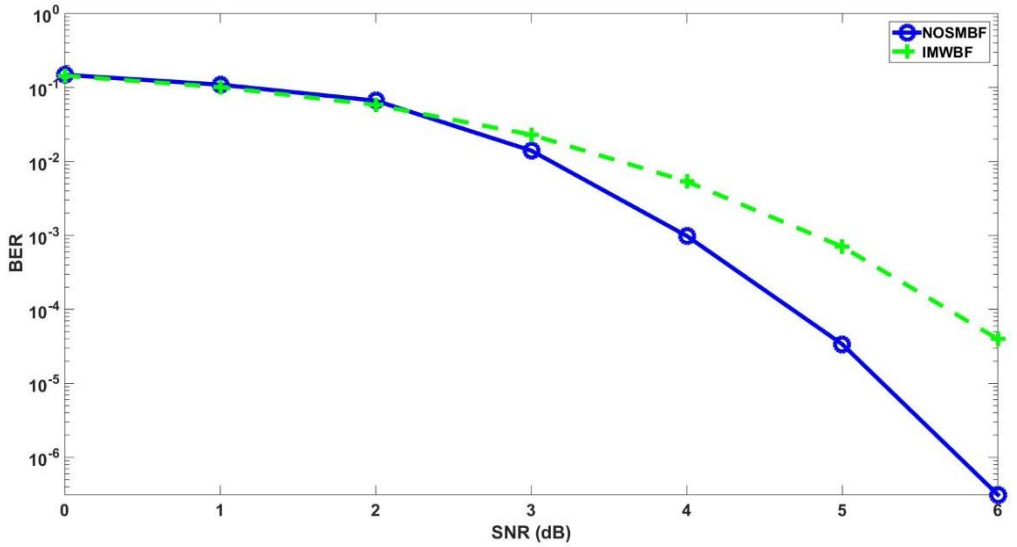


Figure 5. 16: NOSMBF vs IMWBF BER performance analysis

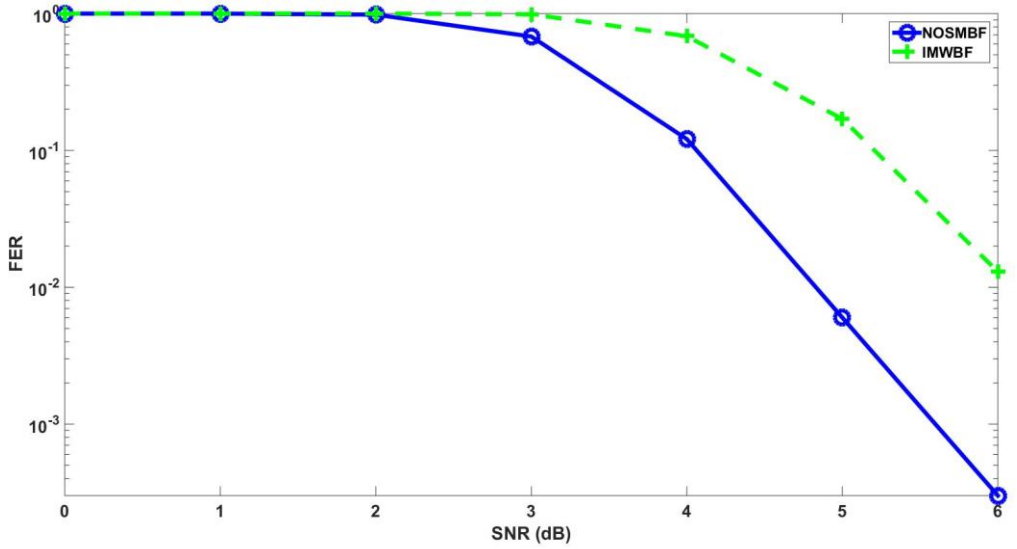


Figure 5. 17: NOSMBF vs IMWBF FER performance analysis

The NOSMBF is simulated with the IMWBF and the results are shown in Figure 5.16 and Figure 5.17. The BER performance of the two algorithms is almost comparable for low SNR. However, from 2 dBs onwards, the performance gap starts to widen with the NOSMBF significantly outperforming the single-bit flipping IMWBF with a performance disparity of over 1 dBs for a BER of 10^{-5}.

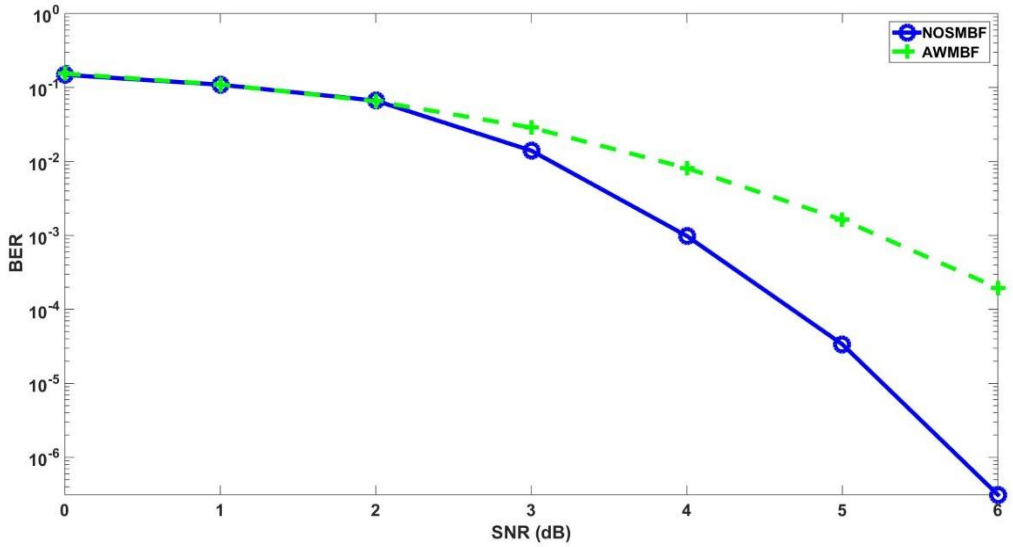


Figure 5. 18: NOSMBF vs AWMBF BER performance analysis

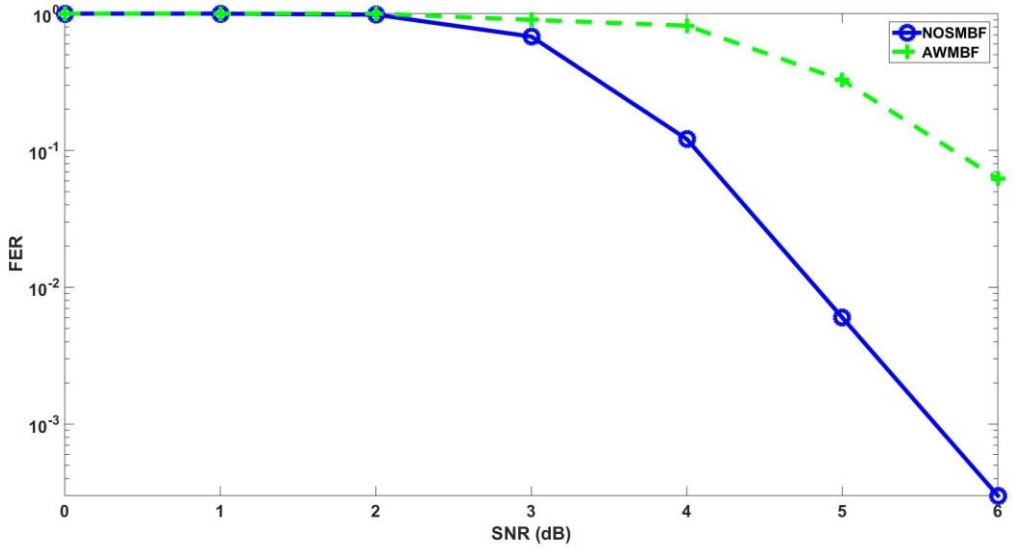


Figure 5. 19: NOSMBF vs AWMBF FER performance analysis

The NOSMBF is compared with the AWMBF in Figure 5.18 and Figure 5.19. Much like the results obtained in Section 4.5, the NOSMBF presents a considerable performance advantage from about 2 dBs onwards. The performance gap continues to grow with performance gaps almost as large as 1.5 dBs at 10^{-4} BER. The AWMBF uses a single adaptive threshold to identify bits to flip while the NOSMBF uses near optimal SNR dependent thresholds which are more effective especially for high SNR regions.

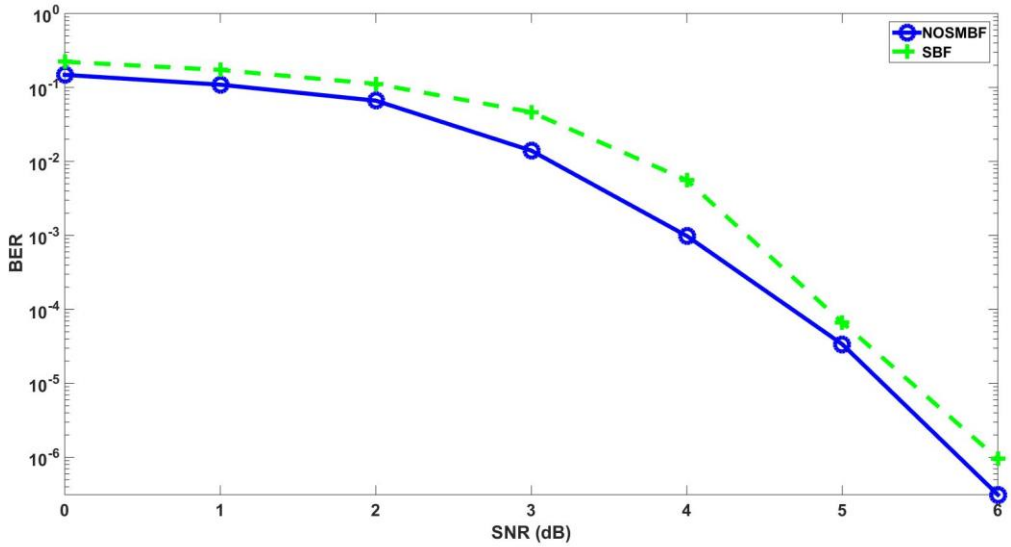


Figure 5. 20: NOSMBF vs SBF BER performance analysis

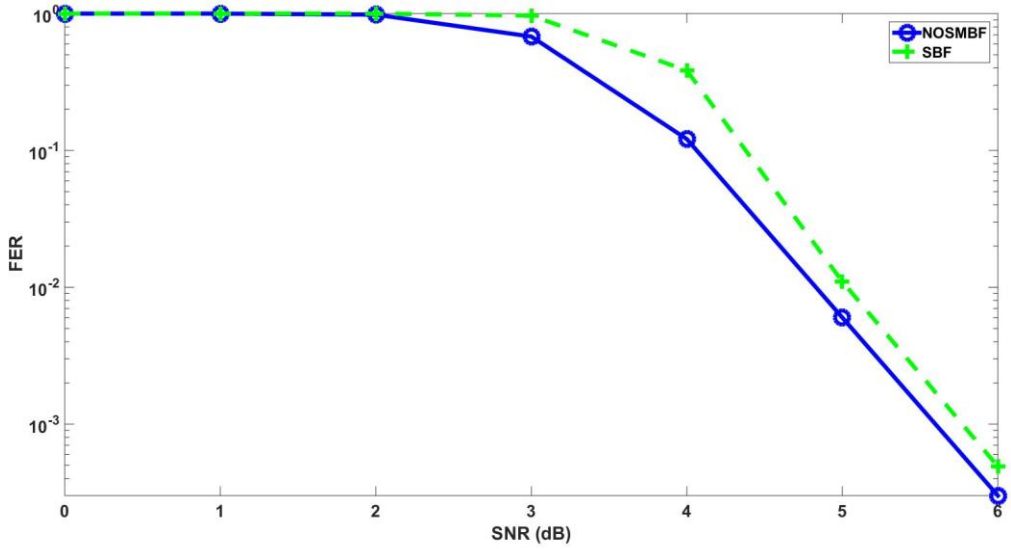


Figure 5. 21: NOSMBF vs SBF FER performance analysis

Figure 5.20 and 5.21 shows the comparison between the SBF as well the NOSMBF. Both algorithms make use of fixed thresholds. However, these thresholds are obtained differently. The SBF uses the probability density function to obtain optimal thresholds while the NOSMBF uses error pattern analysis. The SBF also uses three thresholds which is one more than those used in the NOSMBF. Both algorithms implement soft information modification but in different ways. The NOSMBF has slighty better performance compared to the SBF from the low SNR region to the high SNR region. It can be seen that the performance gap is constantly maintained throughout the simulated noise region.

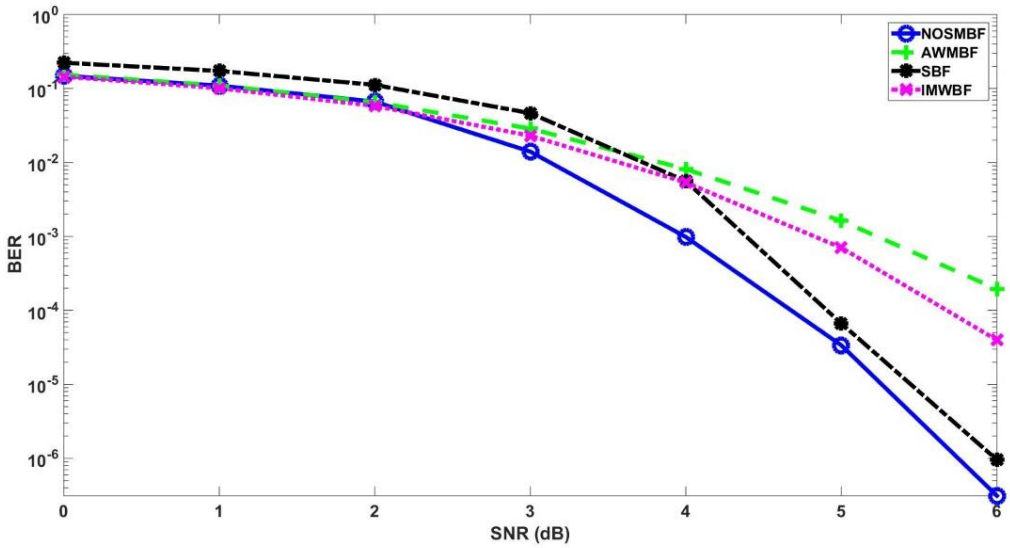


Figure 5. 22: NOSMBF vs IMWBF vs AWMBF vs SBF ANI performance analysis

The NOSMBF compared to all the significant algorithm, the IMWBF, AWMBF and SBF is shown in Figure 5.19. Performance comparisons described earlier are mirrored in the frame error rate (FER) results presented in Figure 5.20. The NOSMBF performs better than the IMWBF, AWMBF and SBF in terms of FER.

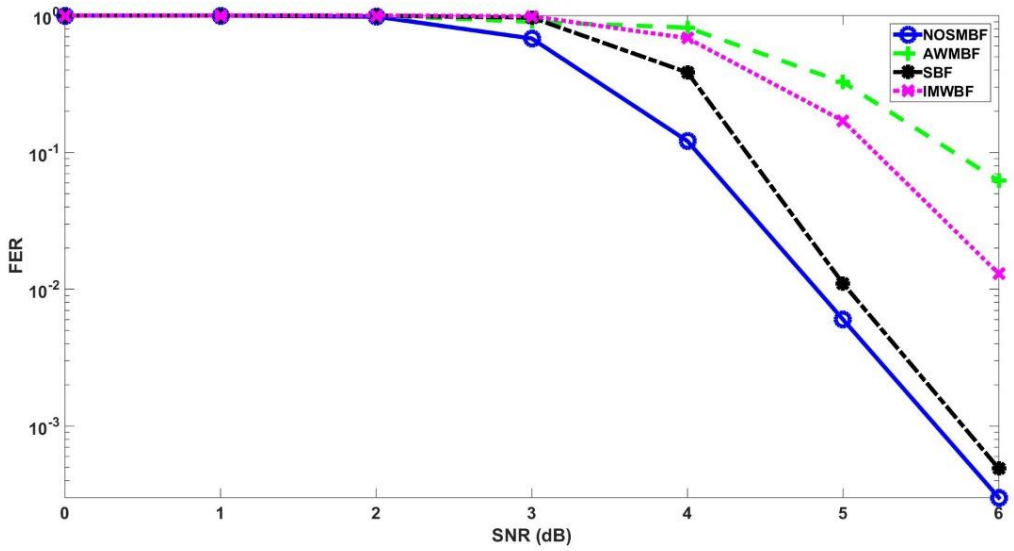


Figure 5. 23: NOSMBF vs IMWBF vs AWMBF vs SBF FER performance analysis

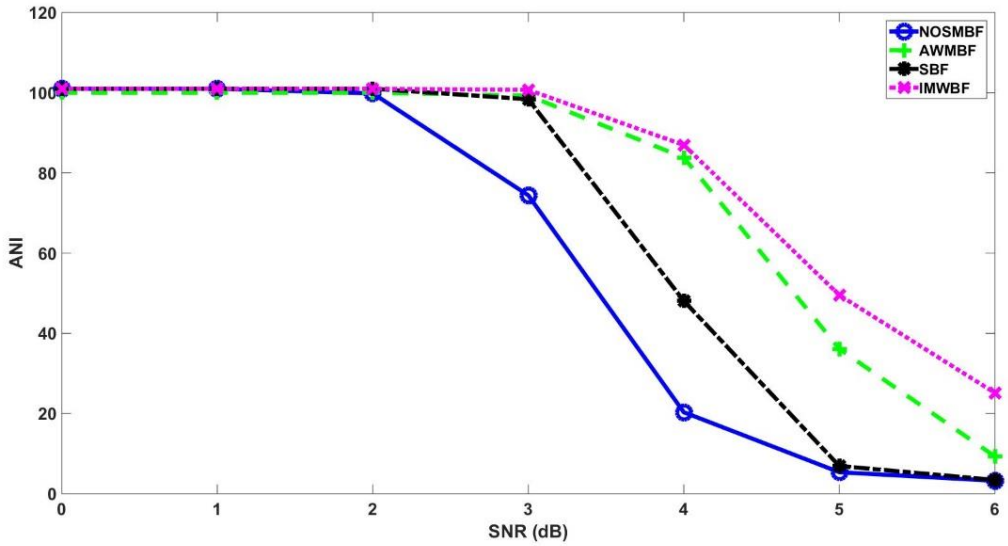


Figure 5. 24: NOSMBF vs IMWBF vs AWMBF vs SBF ANI performance analysis

As depicted in Figure 5.21, all the algorithms fail to converge at low SNR but the NOSMBF reaches the waterfall region faster than all the other algorithms and maintains the lead throughout the whole SNR range. A notable milestone is at the SNR of 4 dBs where the NOSMBF requires 50 % of the number of iterations required by the SBF and 25 % of those required by both the AWMBF and the IMWBF.

5.6 Discussion

The proposed algorithm, the near optimal SNR-based threshold multi-bit flipping (NOSMBF) algorithm has been simulated against several pre-existing algorithms namely the IMWBF, AWMBF and the SBF and the results have been presented. It has also been noted that amongst some of the more popular codes available, the best codes to use are known as the regular progressive edge growth (PEGReg) codes.

As expected, the multi-bit flipping NOSMBF outperforms the single-bit flipping IWMBF. This performance advantage can be attributed to the precision with which the proposed algorithm identifies multiple incorrect bits and corrects them while enhancing the correct bits soft information so as to influence the reliability information calculation in the next iteration. While the IWMBF might accurately identify incorrect bits at high SNR, the obvious handicap is its inability to correct all the errors before reaching the maximum number of iterations.

Simulations in Section 4.5 suggested that the fixed threshold scheme may have superior decoding performance when compared to single adaptive threshold scheme at high SNR. It actually turns out that the NOSMBF has a significant performance advantage when compared to the AWMBF for the SNR range investigated. The near optimisation of the fixed thresholds has a greater probability of identifying incorrect bits as opposed to “hunting” for them as is the case with the AWMBF. This is because the NOSMBF identifies the threshold at which most errors will occur which results in better performance for the NOSMBF when compared to the AWMBF for all SNR and not just high SNR.

The NOSMBF has a notable performance advantage over the soft bit flipping (SBF) algorithm in the simulation region from 0 dBs to 6 dBs. There is a clear performance gap between the NOSMBF and the SBF which is maintained throughout the simulated noise region. The curve profiles are similar and this can be attributed to both algorithms making use of fixed thresholds and soft information modification. Even though the two concepts are incorporated into both algorithms, they are done differently which results in the NOSMBF performing better than the SBF.

In conclusion, simulation results have been presented and analysed to show that the near optimisation of fixed thresholds does indeed lead to LDPC decoding algorithms with improved decoding capabilities. The results also show that the near optimal threshold scheme is more efficient when compared to the probability density function threshold scheme.

6. Quantisation of Bit Flipping algorithms

6.1 Introduction

A very important aspect in the decoding of LDPC codes is the process of quantizing channel output to accurately reflect the information that is being transmitted by the sender. In most cases, decoder designs do not take into account the effects of quantization. Most simulations are carried out using infinite accuracy which is not practical in real life communication devices. In practice, information will have to be quantized by a finite number of bits.

The quantization of channel information simplifies decoder implementation but presents a new problem because it is a lossy process so the decoder makes use of approximate information which degrades the performance of the decoder by enhancing the effects of error prone structures such as trapping sets, short cycles and stopping sets which raises error floors, produces less gentle waterfall regions and also increases the number of iterations required for convergence [49] [50]particularly for the SPA. The problem listed in the last statement presents a fundamental question which is how to optimally quantize information such that the performance degradation due to quantization is mitigated.

In the paper [51], there is a comparative discussion of three methods of channel value quantization. The main argument of this paper is that capacity maximization is the most important criterion for quantizer design in the case of capacity approaching codes. A quantizer for each of the methods is designed and compared in terms of mean-square error (MSE), capacity maximization and cut-off rate. These quantizers are compared by examining the convergence of LDPC codes over the equivalent quantized channels along the code-specific convergence-optimizing quantizers.

The research paper [52] pays attention to code dependent error prone structures known as absorbing sets as well as the effects of quantizing the SPA algorithm and its approximations which are the normalized and offset conditional offset min-sum algorithms.

The notation for the quantization scheme is $Qm.f$ in which there are m bits and $f$bits to the left and right of the radix. Such a scheme produces an asymmetric quantizer in the range $[-2^{m-1}, 2^{m-1} - 2^{-f}]$.

The codes are simulated by a sophisticated parallel processor to exhaustively simulate codes to a BER of 10^{-10} for the SPA, $Q3.2, Q3.3, Q4.2\ and\ Q5.2$ [52].

The paper also gives definitions for error prone structures known as absorbing sets and trapping sets. These structures cause three kinds of errors to occur. These are fixed patterns, oscillatory patterns and random-like patterns [52].

It is observed that low bit e.g. 5-bit Q3.2 quantizer results in clipping and low resolution which leads to errors, mostly oscillatory behaviour.

The log-tahn function in the SPA algorithm has been observed to majorly operate in two different sections which are the upper left corner and the lower right corner of the curve. To accurately capture this profile, the algorithm makes use of a dual quantizer. The $Qm.f$ quantizer is configured as follows: For the upper left corner where range is important because of high LLR values $m > f$ in terms of number of bits, whereas $m < f$ for the lower right corner where there are much smaller LLR values thus resolution is much more important. Simulations carried out with array-based LDPC codes show that the dual quantized SPA is more robust against oscillation errors caused by absorbing sets and therefore has an improved waterfall region and lower error floors [52].

The approximate sum-product algorithm has also been simulated with a fixed quantization scheme and it is shown that it has a 0.2 dB performance degradation in the waterfall region but is more robust in the error floor region [52].

In conclusion it has been established that sub-optimal resolution and range choices in quantization schemes exacerbates error floor performance and oscillatory behaviour. It has also been observed that absorbing sets dominate error floors in a high-range quantizer are as a result of code construction [52].

Irregular LDPC codes are used in the Wireless MAN (IEEE 802.16e) and the paper [53] describes a quantized normalized min-sum decoder for LDPC codes. The normalized algorithm reduces the magnitude of extrinsic information to curb premature saturation states at the bit nodes. Some forms of the decoder use down-scaled intrinsic information iteratively to cater for quantization errors introduced by quantization at the bit-nodes. It is an established fact that the approximate SPA algorithms' performance is not affected by the scaling of a-posteriori information therefore, the channel information is input to the algorithm without any scaling [53].

The (q, f) quantization scheme, which is uniform, uses q bits in total and f bits to represent fractional values to a precision of 2^{-f} while the maximum value is $2^{q-f-1} - 2^{-f}$ and a

minimum value of -2^{q-f-1}. Simulations using a 6 bit quantizer shows that the (6, 1) quantizer has a performance gain larger than 0.4 dBs at a BER in the order of 10^{-7} [53].

A further modification implemented on the algorithm which makes it adaptive stems from the fact that the approximate SPA algorithms are insensitive to channel estimation information. The LLR input is modified as follows:

$$y_v = \begin{cases} \frac{2r_v}{\delta^2}, & SNR < C \\ r_v, & SNR \geq C \end{cases}$$

The value C is obtained via simulations. At SNRs lower than C, the (6, 2) quantization scheme is used while (6,3) is used for SNRs higher than C. Simulations in [53] show that this adaptive quantization has a 0.2 dBs performance gain when compared to the (6, 2) normalized min-sum with channel estimation [53].

The paper [54] describes and analyses several ways to obtain optimal quantizers for AWGN channel output for BP decoding of LDPC codes. The paper briefly describes the process of obtaining a 2-bit optimal quantizer as shown in

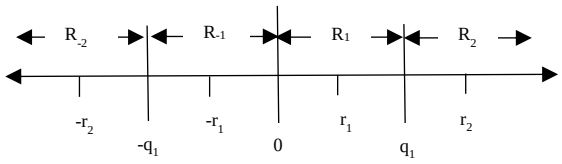


Figure 6. 1: Quantization boundaries (q_i), levels (r_i), and regions (R_i) for a symmetric 2-bit quantizer

The Lloyd algorithm which is a Minimum Mean Square Error (MMSE) quantizer, finds both quantization boundaries q_i as well as levels r_i for the region of quantization R_i as shown in Figure 6.1. A 2-bit quantizer for BP decoding for the BI-AWGN channel is as follows

$$r_1 = \mathcal{L}^{-1}\left(\frac{\int_0^{q_1} \mathcal{L}(y)p(y)dy}{\int_0^{q_1} \mathcal{L}(y)\,dy}\right) = \frac{\int_0^{q_1} yp(y)dy}{\int_0^{q_1} p(y)\,dy} \qquad (6.1)$$

And

$$r_2 = \mathcal{L}^{-1}\left(\frac{\int_{q_1}^{\infty}\mathcal{L}(y)p(y)dy}{\int_{q_1}^{\infty}\mathcal{L}(y)\,dy}\right) = \frac{\int_{q_1}^{\infty} yp(y)dy}{\int_{q_1}^{\infty} p(y)\,dy} \tag{6.2}$$

Where $\mathcal{L}(y) = log\frac{p(y|x=1)}{p(y|x=-1)} = 2\frac{y}{\delta^2}$

A new boundary is obtained using the nearest neighbour rule as follows

$$q_1(r_1, r_2) = \mathcal{L}^{-1}\left(\frac{\mathcal{L}(r_1)+\mathcal{L}(r_2)}{2}\right) = \frac{r_1+r_2}{2} \tag{6.3}$$

Iteratively, $q_1^{(k)}$at iteration k can be used to obtain $q_1^{(k+1)}$ as follows

$$q_1^{(k+1)} = \left(\frac{\int_0^{q_1^{(k)}} yp(y)dy}{\int_0^{q_1^{(k)}} p(y)\,dy} + \frac{\int_{q_1^{(k)}}^{\infty} yp(y)dy}{\int_{q_1^{(k)}}^{\infty} p(y)\,dy}\right)\Big/2 := f\left(q_1^{(k)}\right) \tag{6.4}$$

For the simulations in [54], $q_1^{(0)} = 3$ and $q_1^{(1)} = 2.154$ was obtained followed by $q_1^{(2)} = 1.728$ until convergence $q_1^{*Lloyd} = 1.207$ of which $q_1^{*Lloyd} = f\left(q_1^{*Lloyd}\right)$. In this section, discrete DE is used to evaluate quantizers with the minimum error. This process results in uniform quantizers [54].

For Uni-parametric search for optimal quantizers, the initial search is for a 1-bit quantizer and it does not require an optimization process because $q_0 = 0$. The pdf of the channel output y is $y = \frac{1}{\sqrt{2\pi\delta^2}}e^{-\frac{(y-1)^2}{2\delta^2}}$. The crossover probability is $\epsilon = \int_{-\infty}^{0} p(y)dy$, the two quantization levels are calculated as follows $r_{-1} = log\frac{\epsilon}{1-\epsilon}$ and $r_1 = log\frac{1-\epsilon}{\epsilon} = -r_{-1}$. The LLR $\mu_0$of y we have the initial pdf

$$p_{1-bit}(\mu_0) = \epsilon\delta(\mu_0 + r_1) + (1-\epsilon)\delta(\mu_0 - r_1) \tag{6.5}$$

The Equation 6.5 is fed into the DE and the resulting noise threshold values are obtained.

A cumulative distribution (cdf's) $\varphi_i = \int_{R_i} p(y)dy$ is introduced which is integrated over the quantization regions R_i.

In addition, a quantizer with an erasure region is considered. The quantization levels $r_{-1}, r_0\ and\ r_1$ are separated by $-q_e$ by symmetry such that $r_{-1} = -r_1$. The initial pdf of the channel output LLR is shown in Equation 6.6.

$$p_{erasure}(\mu_0) = \varphi_{-1}\delta(\mu_0 - r_{-1}) + \varphi_0\delta(\mu_0) + \varphi_1\delta(\mu_0 - r_1) \tag{6.6}$$

In which $r_{-1} = log\frac{\varphi_{-1}}{\varphi_1}$ and $r_1 = log\frac{\varphi_1}{\varphi_{-1}} = -r_{-1}$.

A 2-bit symmetric quantizer can also be found using (34) in which $r_i = log\frac{\varphi_i}{\varphi_{-i}}$ for $i = -2, -1, 1, 2$ and $q_1 > 0$ and the other boundary is at 0.

$$\begin{aligned} p_{2-bit}(\mu_0) = {} & \varphi_{-2}\delta(\mu_0 - r_{-2}) + \varphi_{-1}\delta(\mu_0 - r_{-1}) + \varphi_1\delta(\mu_0 - r_1) \\ & + \varphi_2\delta(\mu_0 - r_2) \end{aligned} \tag{6.7}$$

It is demonstrated in [54] that the 2-bit symmetric quantization found by DE outperforms the MMSE Lloyd quantizer by 0.42 dB.

A general $n\ (\geq 2)$ bit quantizer can be obtained using uni-parametric exhaustive search method [54]. A single quantization step $q\Delta$ defines the quantizer and the boundaries are represented by $q_i = i \times q\Delta$ for $-2^{n-1} + 1 \leq i \leq 2^{n-1} - 1$. Optimizing the quantizer is done by tweaking $q\Delta$ and performing DE. The pdf of μ_0 is given by

$$\begin{aligned} p_{n-bit}(\mu_0) = {} & \varphi_{-2^{n-1}}\delta(\mu_0 - r_{-2^{n-1}}) + \cdots + \varphi_{-1}\delta(\mu_0 - r_{-1}) \\ & + \varphi_1\delta(\mu_0 - r_1) + \cdots + \varphi_{2^{n-1}}\delta(\mu_0 - r_{2^{n-1}}) \end{aligned} \tag{6.8}$$

Where $r_i = log\frac{\varphi_i}{\varphi_{-i}}$ for $i = -2^{n-1} + 1, \dots, -1, 1, \dots, 2^{n-1} - 1$. The integration regions for partial cdf's φ_i are $R_{-2^{n-1}} = (-\infty, q_{-2^{n-1}-1}), R_{2^{n-1}} = [q_{2^{n-1}-1}, \infty)$ and $R_i = [q_{i-1}, q_i)$ and $R_{-i} = [q_{-i}, q_{-i+1})$ for $i = 1, \dots, 2^{n-1} - 1$ and $q = 0$ [54].

The simulation includes code rates varying from 0.25 to 0.95. For all code rates the difference between 3-bit uniform and non-uniform quantizer is less than 0.01 dBs. The 1-bit erasure quantizer has a 1 dBs gain over the simple 1-bit quantizer. The results also show that the 3-bit

quantizers only have about a 0.1 dBs performance degradation as compared to floating point precision for all code rates. A higher column weight and code rate results in finer quantization about the origin [54].

The paper [55] describes how error floors are not just a result of a small girth or Error Prone Structures (EPS) such as near codewords, trapping or absorbing sets. It is also known that error floors are caused by the quantization of channel information [55]. The paper investigates the causal effects of error floors in the binary LDPC codes from the perspective of the MP decoder implementation, giving particular attention to limitations that reduce the accuracy of messages during decoding [55].

The quantizer is motivated by the analysis of the decoding process occurring in the region of an EPS such that a $(q+1)$-*bit quasi-uniform quantizer* with an extremely high saturation level is proposed. The decoding algorithm itself is unchanged [55].

The paper goes on to define an *absolute trapping set*, a *k-iteration computation tree*, the *separation assumption* and an *unsaturated decoder* [55].

The q-bit uniform quantizer has a quantization step Δ with one sign bit. Quantized values are Δl for $-N \leq l \leq N$ in which $N = 2^{q-1} - 1$.

A $(q+1)$-*bit quasi-uniform quantizer* is introduced to attempt to capture exponentially growing trapping set messages while still capturing small messages precisely. The quasi-uniform quantization rule is given by

$$Q(L) = \begin{cases} (0,l), & if\ l\Delta - \frac{\Delta}{2} < L < l\Delta + \frac{\Delta}{2} \\ (0,N), & if\ N\Delta - \frac{\Delta}{2} < L < dN\Delta \\ (0,-N), & if\ -dN\Delta < L \leq -N\Delta + \frac{\Delta}{2} \\ (1,r), & if\ d^r N\Delta \leq L < d^{r+1}N\Delta \\ (1,-r), & if\ -d^{r+1}N\Delta \leq L < -d^r N\Delta \\ (1,N+1), & if\ L \geq d^{N+1}N\Delta \\ (1,-N-1), & if\ L \leq d^{N+1}N\Delta \end{cases} \tag{6.9}$$

Where $N = 2^{q-1} - 1, -N - 1 \leq l \leq N + 1, 1 \leq r \leq N$ and d is a in the range $(1, d_v - 1])$ [55].

Simulations were done using quasi-cyclic (QC) LDPC and Margulis codes on both the BSC and AWGNC codes [55]. The quasi-uniform quantizer significantly reduces error floors and the performance is much like that of an unsaturated decoder. It is also noted that when the BSC crossover probability is low or when the AWGNC SNR is high, the error patterns do not correspond to the support of small trapping sets but the decoding failures in the uniform quantizers could be attributed to small trapping sets [50].

Significant research has gone into quantization of the SPA algorithm and its variants. Apart from [12], there is very little research that has gone into understanding the quantization effects on BF decoding algorithms.

6.2 Simulation Results

6.2.1 Quantisation Schemes

The quantizer of choice is the mid-tread algorithm [56] for uniform quantization and the minimum mean square error (MMSE) Lloyd-Max algorithm [57] for non-uniform quantization. Both schemes of quantization are important in order to ascertain the quantization scheme that provides the best performance. The bit length that was used was 6-bits and the 8-bit for both quantization schemes.

6.2.2 Uniform Quantization

The proposed AMTMBF and NOSMBF algorithms where simulated against their quantized versions and the results obtained. The quantized versions processed quantized channel information while reliability values themselves remain unquantized. The quantized versions used 6 bit and 8 bit quantization schemes.

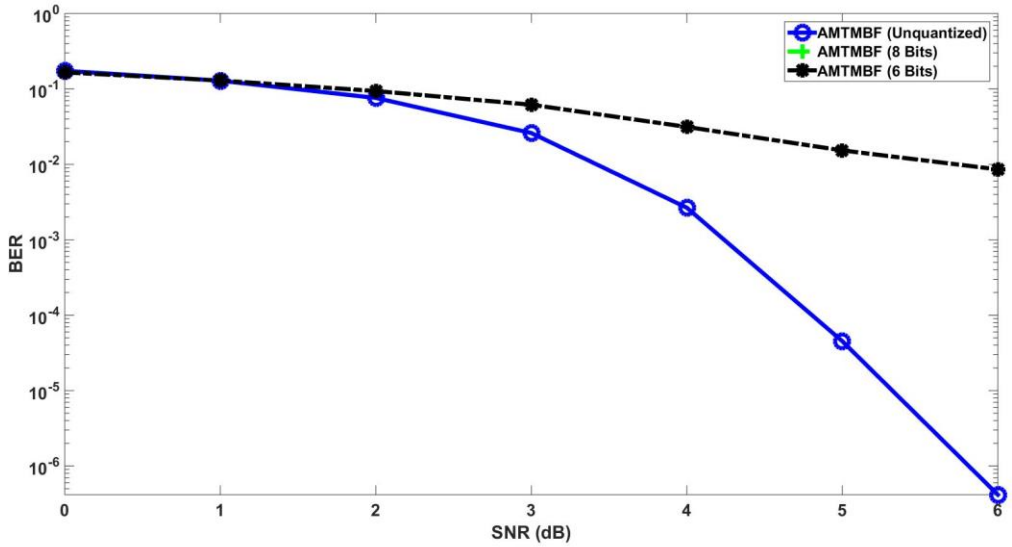


Figure 6. 2: AMTMBF BER uniform quantization

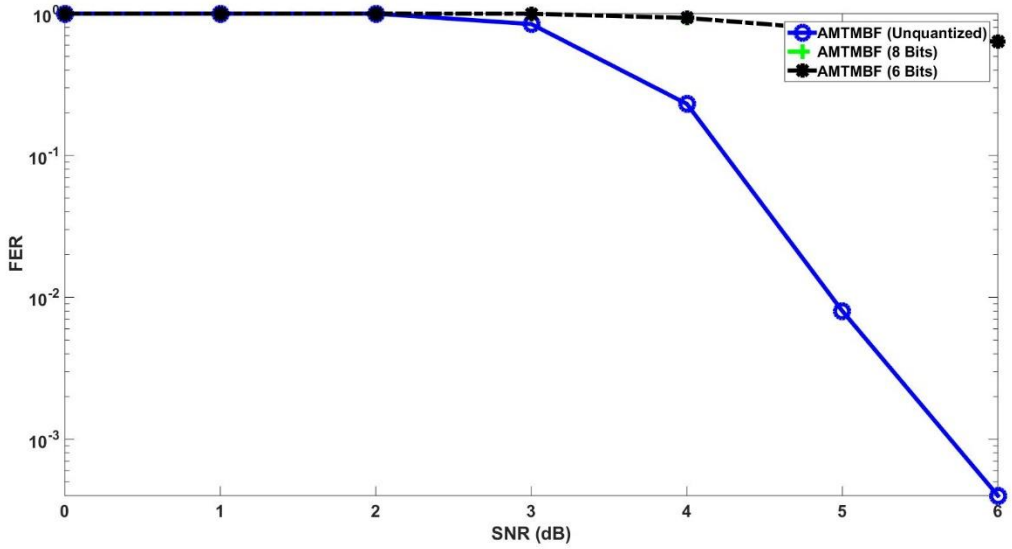


Figure 6. 3: AMTMBF FER uniform quantization

Figure 6.2 and Figure 6.3 shows the BER and FER performance of the 6 bit and 8 bit quantized AMTMBF. The results show that quantization brings about a significant degradation in performance when compared to the unquantized version. The results also show that there is virtually no performance gained from increasing the uniform quantization scheme from 6 bits to 8 bits.

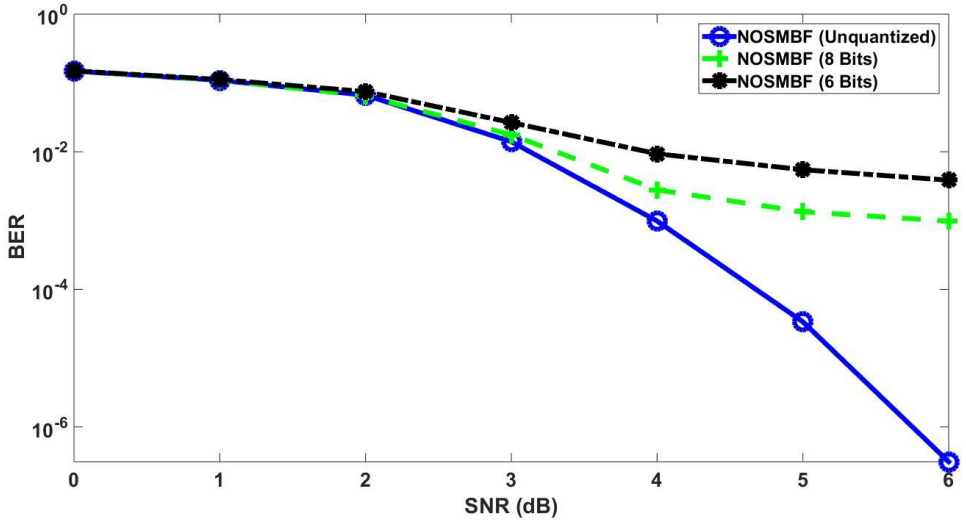


Figure 6. 4: NOSMBF BER uniform quantization

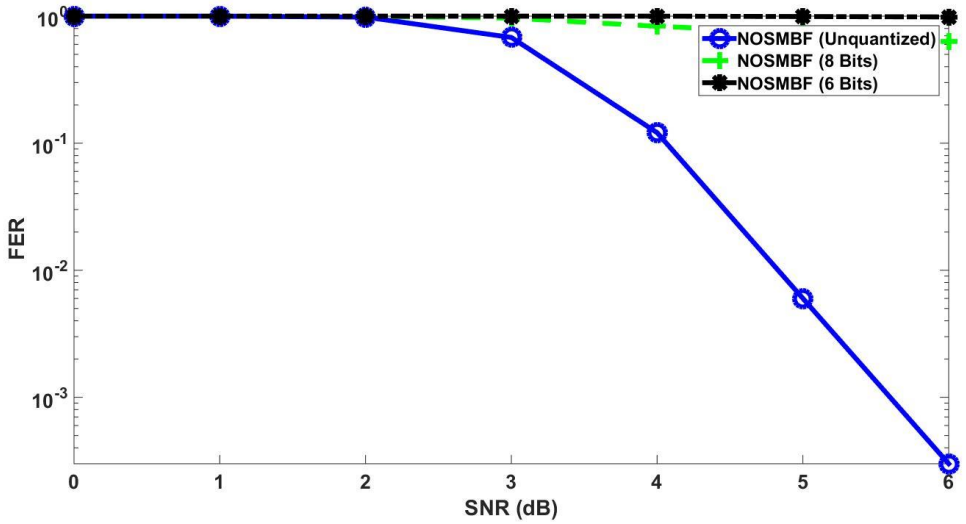


Figure 6. 5: NOSMBF FER uniform quantization

The results in Figure 6.4 and Figure 6.5 show the effect of uniform quantization on the NOSMBF. The performance degradation is quite substantial but not as severe as in the case of the AMTMBF. It can be observed that the increase in bit word length from 6 bits to 8 bits introduces a notable performance gain. There are also discernible error floors due to quantization.

6.2.3 Non-Uniform Quantization

The proposed algorithms where also simulated using non-uniform quantization with word lengths of 6 bits and 8 bits. The Lloyd-Max Minimum Mean Square Error (MMSE) quantizer [57] to determine the partitions and codebook.

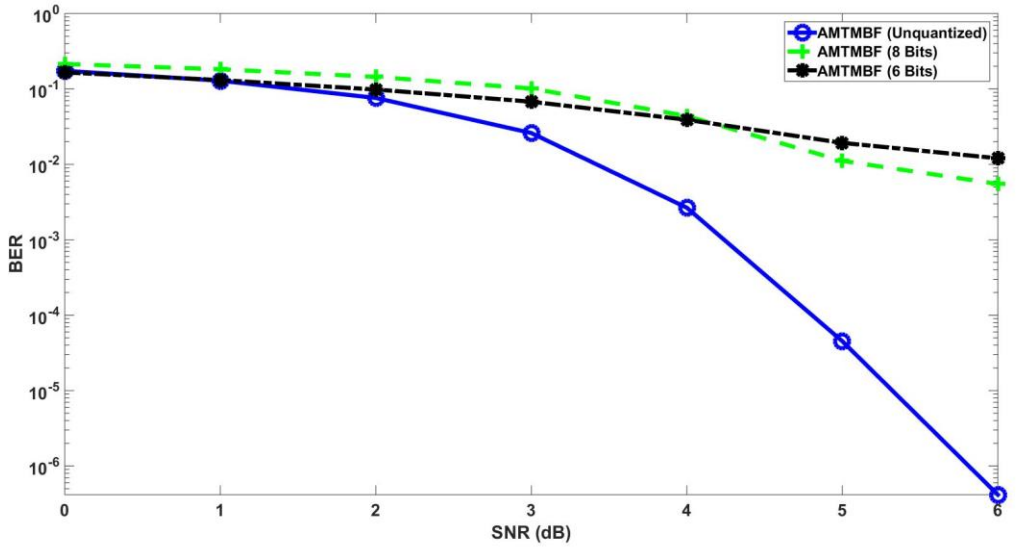


Figure 6. 6: AMTMBF BER non-uniform quantization

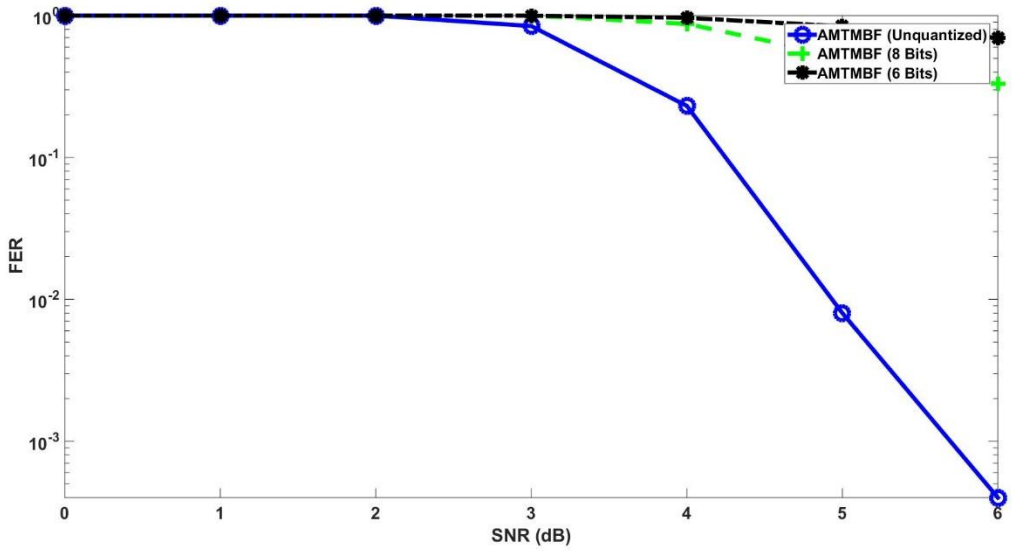


Figure 6. 7: AMTMBF FER uniform quantization

The 6 bit non-uniform quantized AMTMBF algorithm has BER performance that is comparable to the non-quantized AMTBF while the 8 bit version exhibits worse performance at low SNR. The 8 bit algorithm eventually improves at high SNR even though error floors start to appear. The FER performance of the quantised AMTMBF also shows that increase in word length from 6 bits to 8 bits improves the decoding performance.

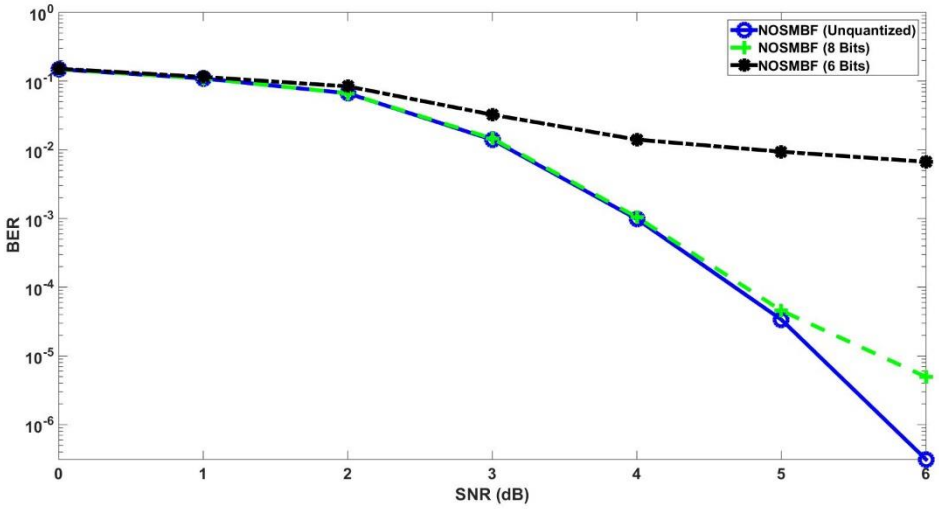


Figure 6. 8: NOSMBF BER non-uniform quantization

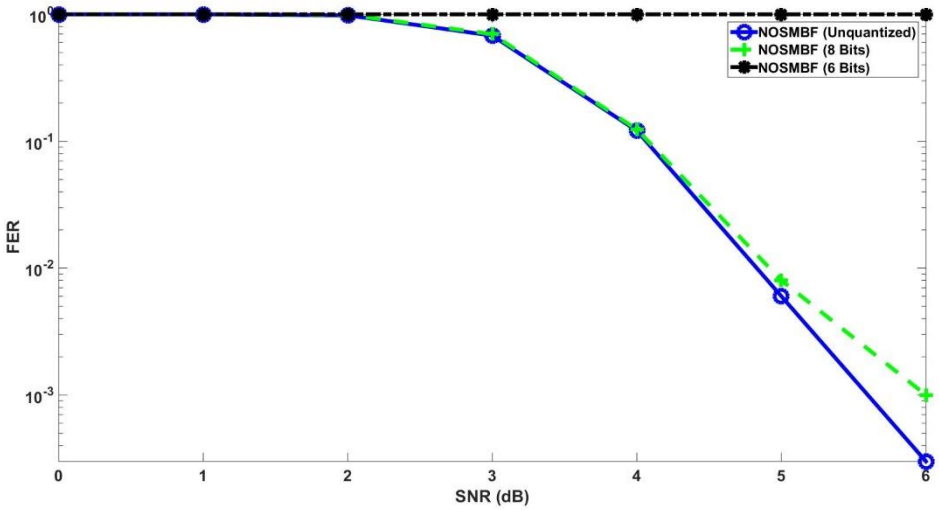


Figure 6. 9: NOSMBF FER non-uniform quantization

The simulation of the non-uniform quantized NOSMBF shows more dramatic results. While the 6 bit quantized algorithm is significantly compromised by quantization, the BER and FER performance of the 8-bit quantized version of the NOSMBF is marginally close to that of the non-quantized version.

6.2.4 Comparative analysis of the AMTMBF with existing algorithms

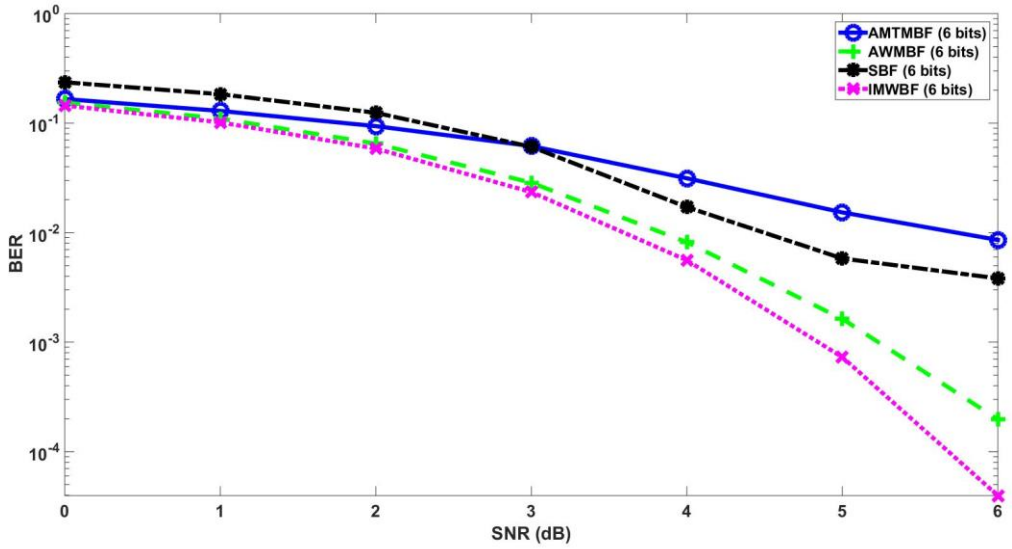


Figure 6. 10: 6 bit uniform quantization BER performance analysis – AMTMBF vs IMWBF vs SBF vs AWMBF

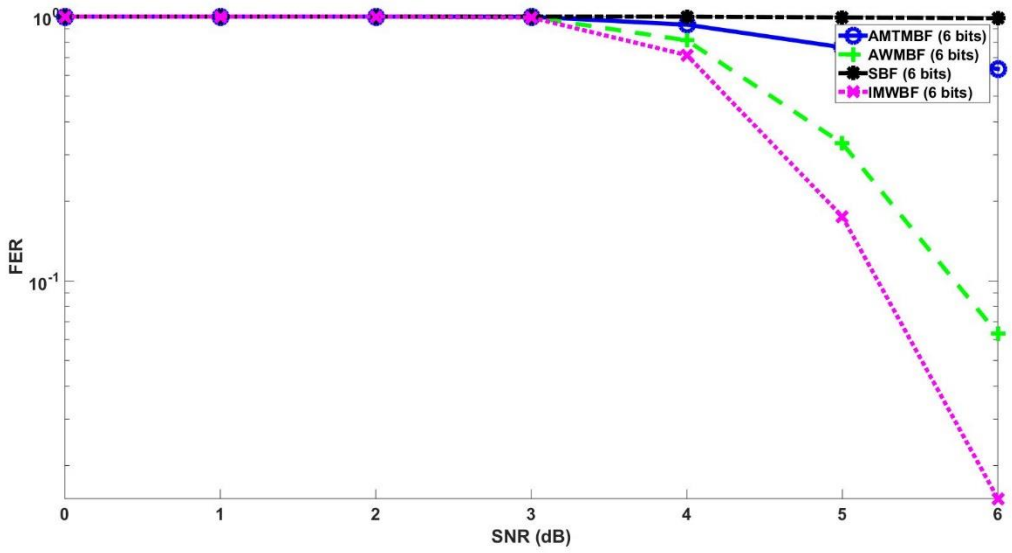


Figure 6. 11: 6 bit uniform quantization FER performance analysis – AMTMBF vs IMWBF vs SBF vs AWMBF

Figure 6.10 and Figure 6.11 shows the comparative performance of the 6 bit quantized versions of the AMTMBF and the existing algorithms. The best performing algorithm according to the results is the IMWBF followed by the AWMBF then the SBF and lastly the AMTMBF. The same trend can also be observed in the 8 bit quantized versions of the algorithm shown in Figure 6.12 and Figure 6.13. The IMWBF and the AWMBF algorithms do not incorporate the soft information modification or soft bit flipping scheme and therefore the loss of information

due to quantization does not degrade the performance of these two algorithms as drastically as it does the soft bit flipping counterparts.

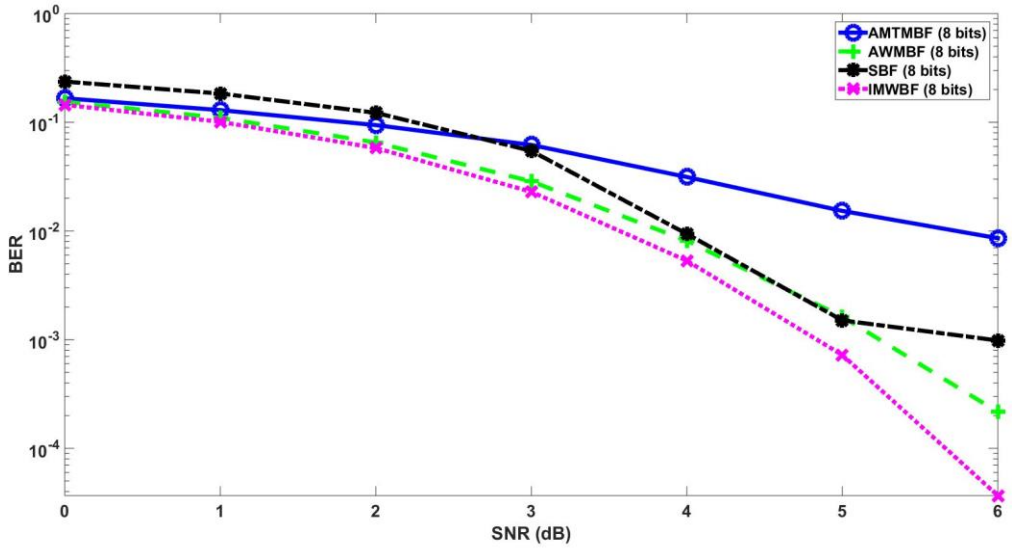


Figure 6. 12: 8 bit uniform quantization BER performance analysis – AMTMBF vs IMWBF vs SBF vs AWMBF

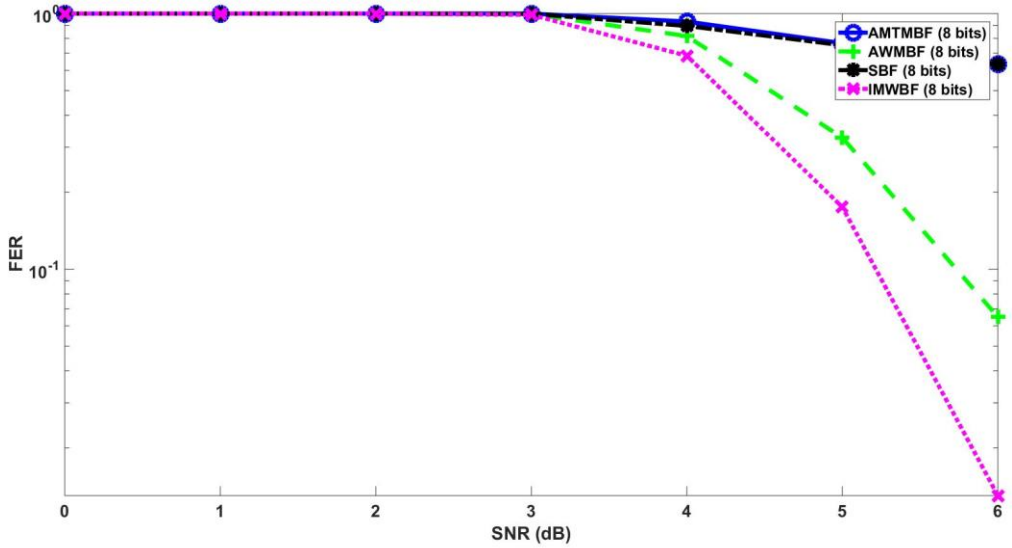


Figure 6. 13: 8 bit uniform quantization FER performance analysis – AMTMBF vs IMWBF vs SBF vs AWMBF

The SBF has more sophisticated soft bit flipping functions as well as three thresholds as opposed to the simpler soft bit flipping functions and two adaptive thresholds of the AMTMBF. These features of the SBF offer better insulation to the effects of quantization for SBF when compared to the AMTMBF.

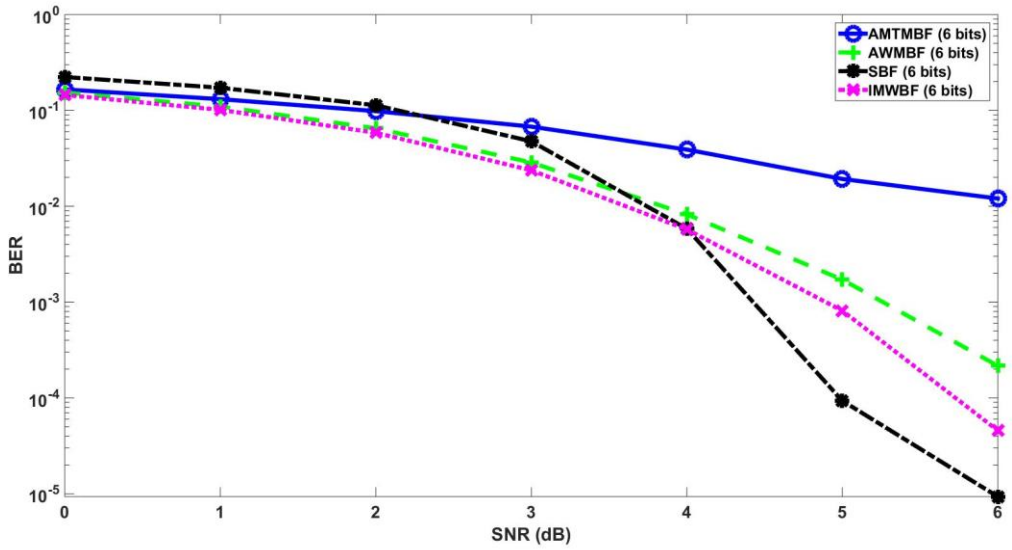


Figure 6. 14: 6 bit non-uniform quantization BER performance analysis – AMTMBF vs IMWBF vs SBF vs AWMBF

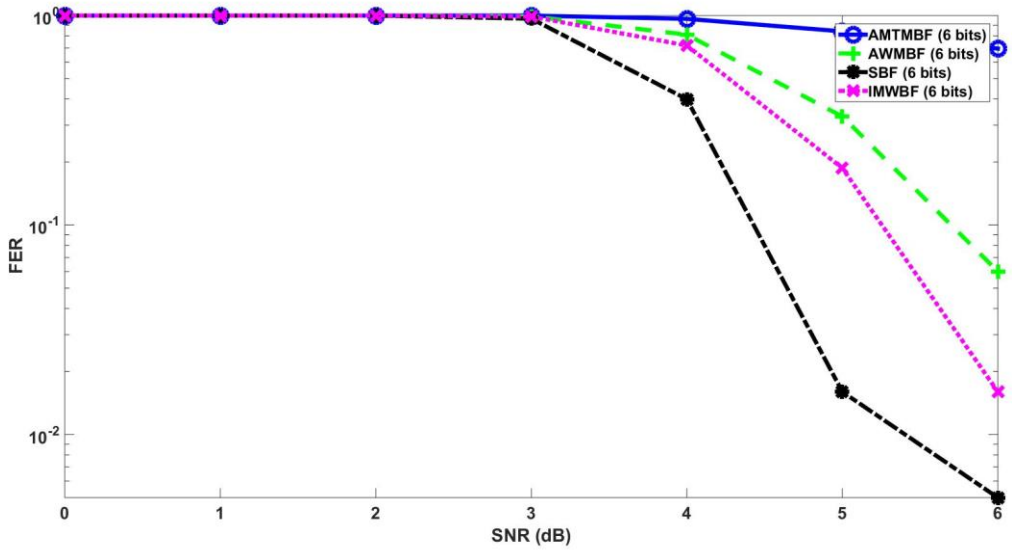


Figure 6. 15: 6 bit non-uniform quantization FER performance analysis – AMTMBF vs IMWBF vs SBF vs AWMBF

The Lloyd-Max quantizer gives a more accurate quantization scheme compared to the uniform quantization scheme which results in reduced loss of information as is the case with uniform quantization. The SBF is the best performing algorithm in these simulations while the AMTMBF is the worst performing algorithm. This performance discrepancy can be attributed to the differences in soft bit flipping mechanism in which soft information values are

manipulated. As shown in [12], the SBF is tailored for quantized values and can therefore take full advantage of the non-uniform quantization. The results also show that the increase from 6 bits to 8 bits results in very little increase in performance for the AMTMBF.

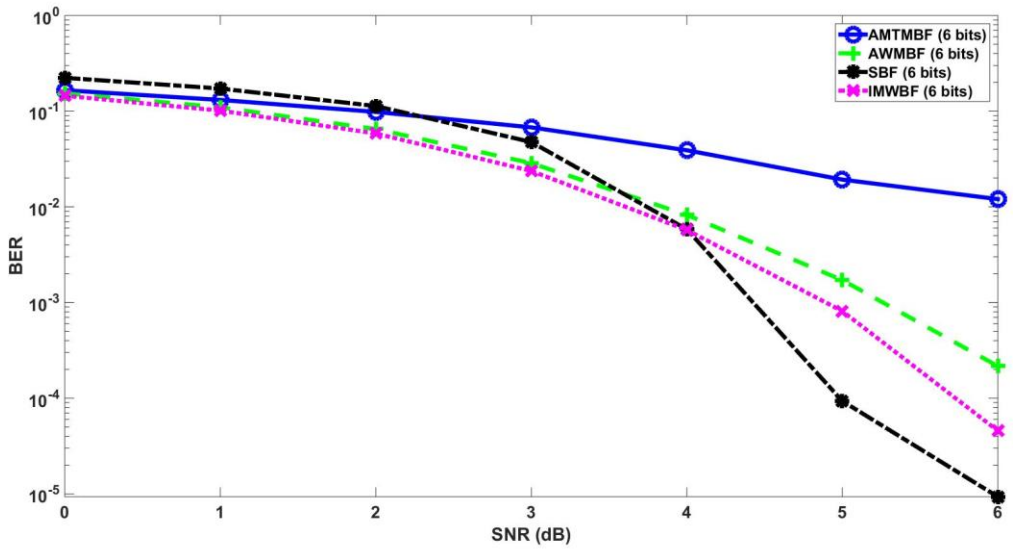


Figure 6. 16: 8 bit non-uniform quantization BER performance analysis – AMTMBF vs IMWBF vs SBF vs AWMBF

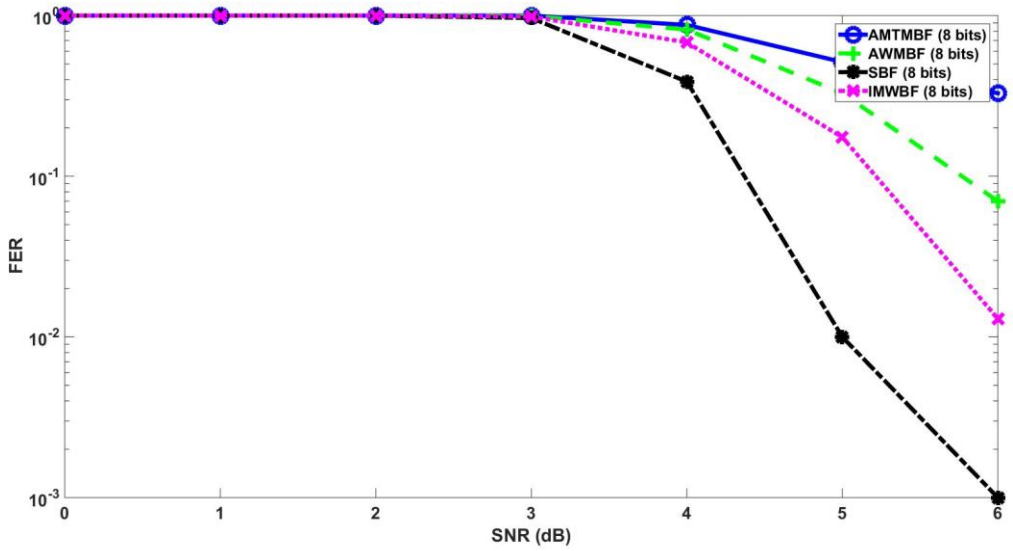


Figure 6. 17: 8 bit non-uniform quantization BER performance analysis – AMTMBF vs IMWBF vs SBF vs AWMBF

6.2.5 Comparative analysis of the NOSMBF with existing algorithms

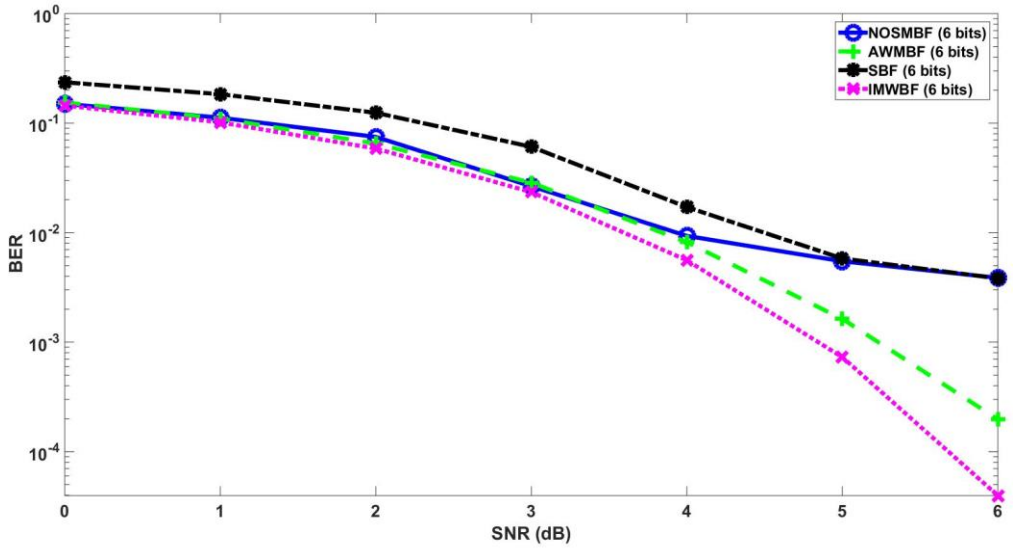


Figure 6. 18: 6 bit uniform quantization BER performance analysis – NOSMBF vs IMWBF vs SBF vs AWMBF

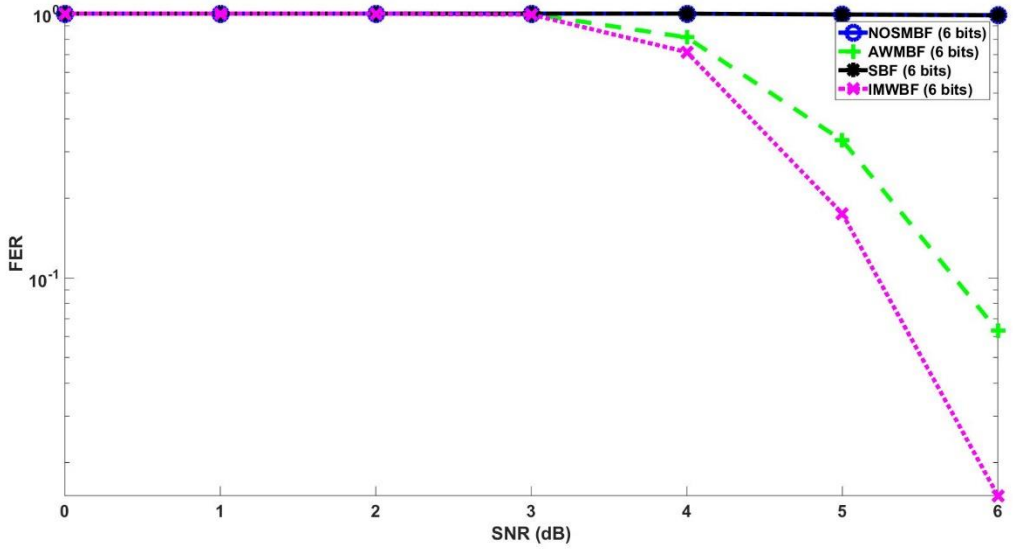


Figure 6. 19: 6 bit uniform quantization BER performance analysis – NOSMBF vs IMWBF vs SBF vs AWMBF

As shown in Figures 6.18 and Figure 6.19, the performance of the 6 bit quantized versions of the IMWBF and that of the AWMBF is better than that of the NOSMBF and the SBF as is expected. This is because both the IWBF and the AWMBF are not heavily reliant on soft information as is the case with the SBF and the NOSMBF. The NOSMBF performs better than

the SBF for a better part of the simulated SNR range. However, both algorithms have error floors at high SNR due to both algorithms' inability to distinguish between correct and incorrect bits due the inaccuracies introduced by coarse quantisation. Increase of quantisation bits from 6 bits to 8 bits has a significant performance improvement as shown in Figure 6.20 and 6.21. However, error floors are still evident.

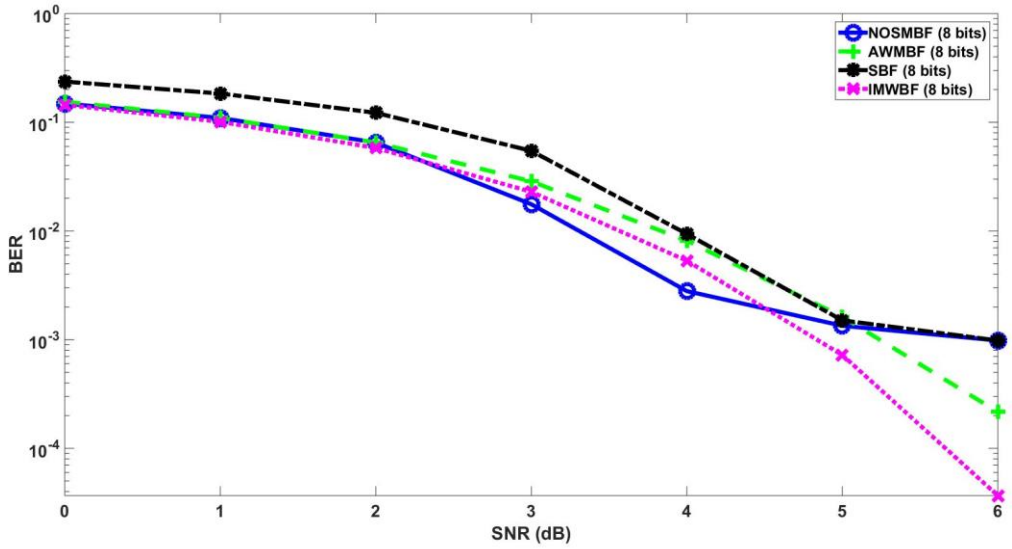


Figure 6. 20: 8 bit uniform quantization BER performance analysis – NOSMBF vs IMWBF vs SBF vs AWMBF

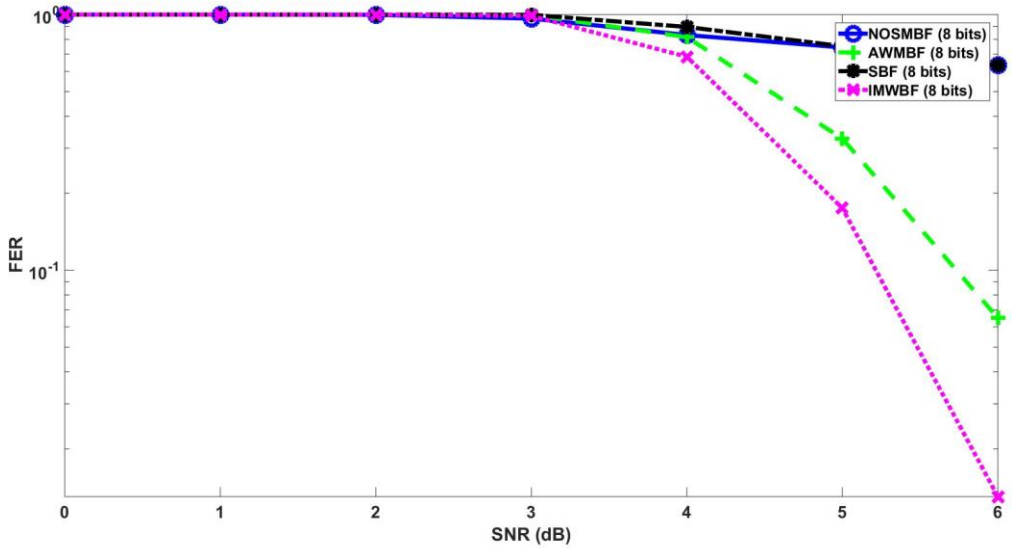


Figure 6. 21: 8 bit uniform quantization FER performance analysis – NOSMBF vs IMWBF vs SBF vs AWMBF

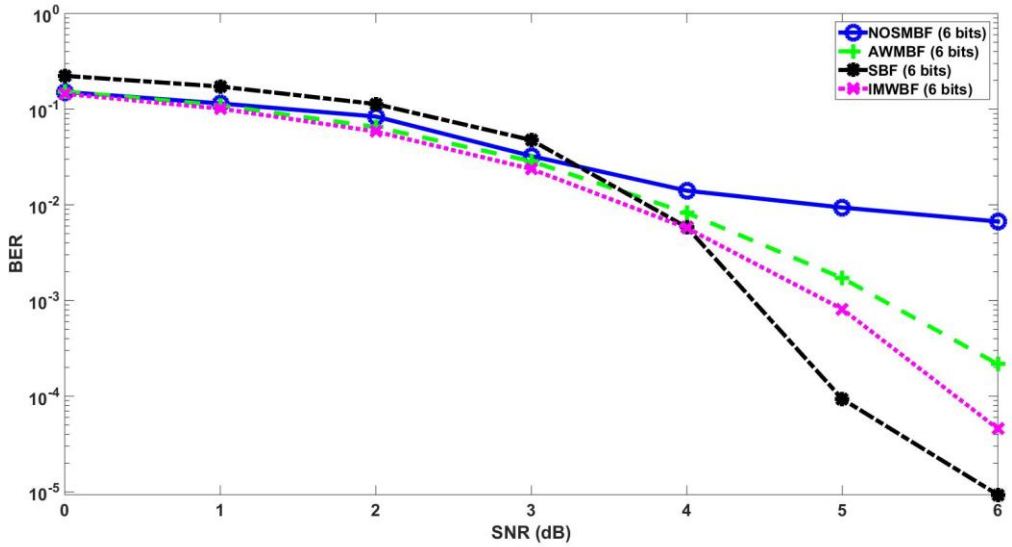


Figure 6. 22: 6 bit non-uniform quantization BER performance analysis – NOSMBF vs IMWBF vs SBF vs AWMBF

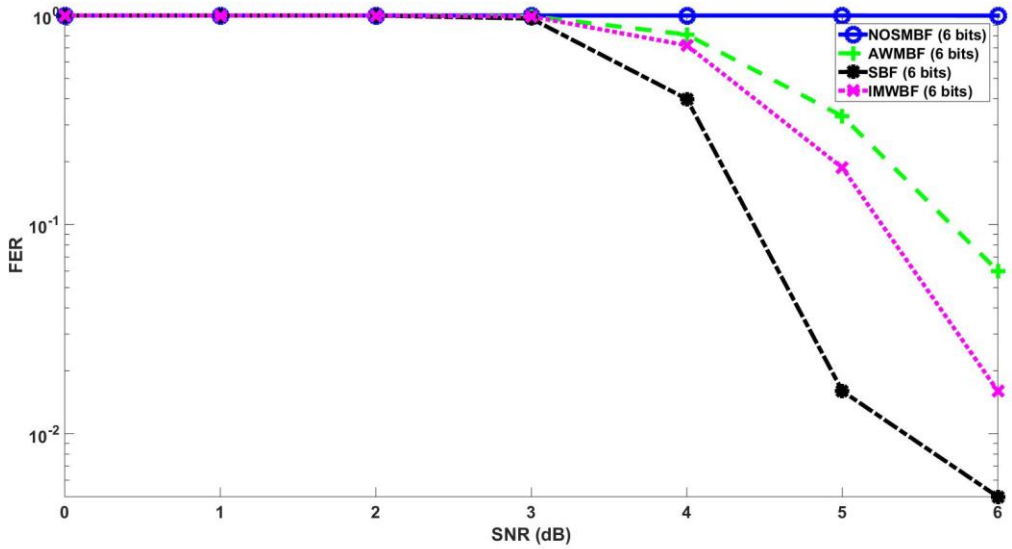


Figure 6. 23: 6 bit uniform quantization FER performance analysis – NOSMBF vs IMWBF vs SBF vs AWMBF

The NOSMBF's performance as shown in Figure 6.22 and Figure 6.23, is not improved by the non-uniform 6-bit quantization scheme. However, the SBF's performance is improved dramatically especially in the high SNR range. The reason for this is the SBF's more sophisticated soft bit flipping mechanism designed to combat the effects of coarse quantization. The IMWBF and the AWMBF show significant resilience against coarse quantization. However, Figure 6.24 and Figure 6.25 shows a more dramatic performance improvement for the NOSMBF when quantized with 8 bits. The performance for both the SBF and the NOSMBF soft bit flipping algorithms is close to that of the non-quantized version.

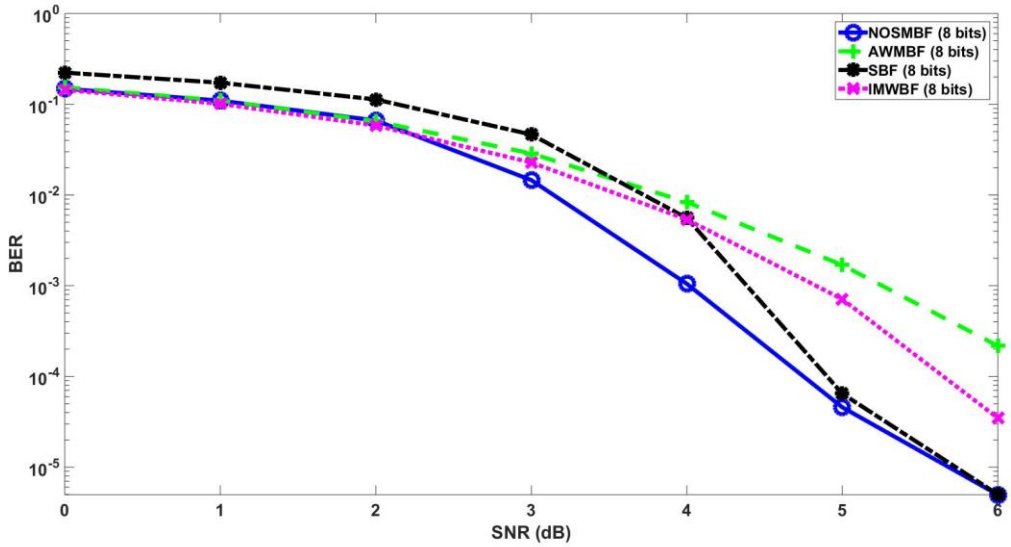


Figure 6. 24: 8 bit uniform quantization BER performance analysis – NOSMBF vs IMWBF vs SBF vs AWMBF

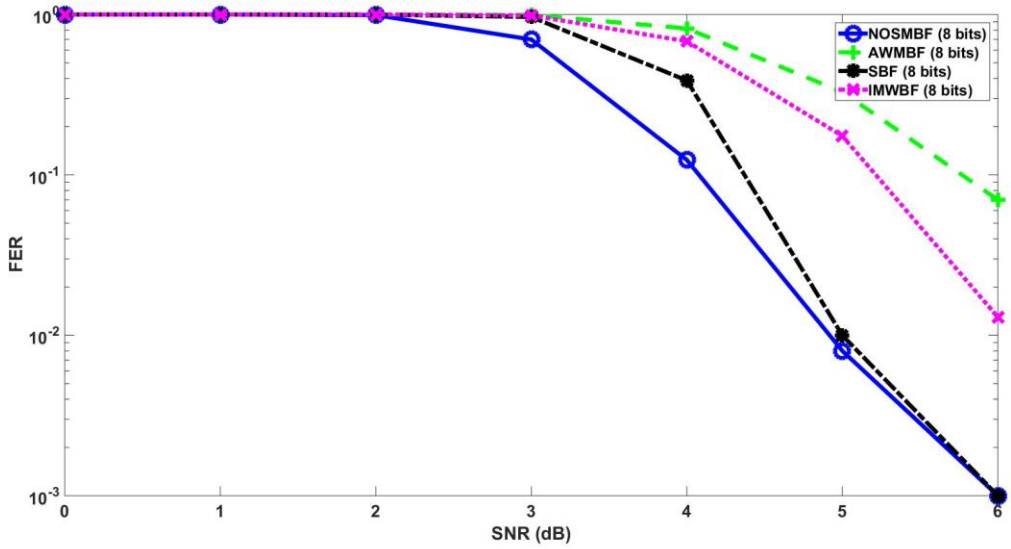


Figure 6. 25: 8 bit uniform quantization BER performance analysis – NOSMBF vs IMWBF vs SBF vs AWMBF

6.3 Discussion

Investigating the effect of quantization is very important so as to determine the performance of the algorithm when it is implemented in a practical decoder.

In the case of uniform quantization, the mean square error is not minimized therefore the partition and codebook are a poor approximation of the distribution of the channel soft information therefore the loss of information is considerable. Short quantizer bit length severely reduces the performance of the AMTMBF due to the fact that the algorithm makes

use of soft information modification with a substantial quantization error. The effect of this quantization error is amplified by the adaptive nature in which the threshold is determined. This means that the thresholds obtained are suboptimal hence the poor performance. Similar observations are made for the NOSMBF. However, the effect of the quantization error is not as severe because the fixed thresholds are impervious to soft information quantization error since they are obtained from more accurate values.

One solution that can improve the performance would be to increase the quantization word length as is evident in the simulation results of the NOSMBF.

Non-uniform quantization minimizes the mean square error between the actual values and the codebook. This results in improved performance for both the AMTMBF and the NOSMBF. The increase in quantization word length causes a notable difference in performance for the AMTMBF as opposed to the case in uniform quantization. The 8 bit NOSMBF performs almost as well as the unquantized version which demonstrates the robustness of the near-optimal threshold scheme for multi-bit flipping algorithms.

It has also been observed that the IMWBF and the AWMBF which do not make extensive use of soft channel information in determining flipping thresholds, are marginally affected by coarse quantization. The proposed algorithms make use of soft bit flipping which, as expected worsens the performance of these algorithms when the soft information is quantized.

7. Comparison of the AMTMBF to the NOSMBF

7.1 Introduction

The previous chapters have discussed two threshold based multi-bit flipping algorithms and simulations have shown that both adaptive and near-optimal fixed threshold schemes are viable methods for improving the decoding performance of LDPC bit flipping decoders. In this chapter, both algorithms are analysed and compared against each other. Even though both methods of obtaining thresholds have yielded a notable performance advantage, it is important to determine which of the two methods is the most effective.

7.2 Simulation Results

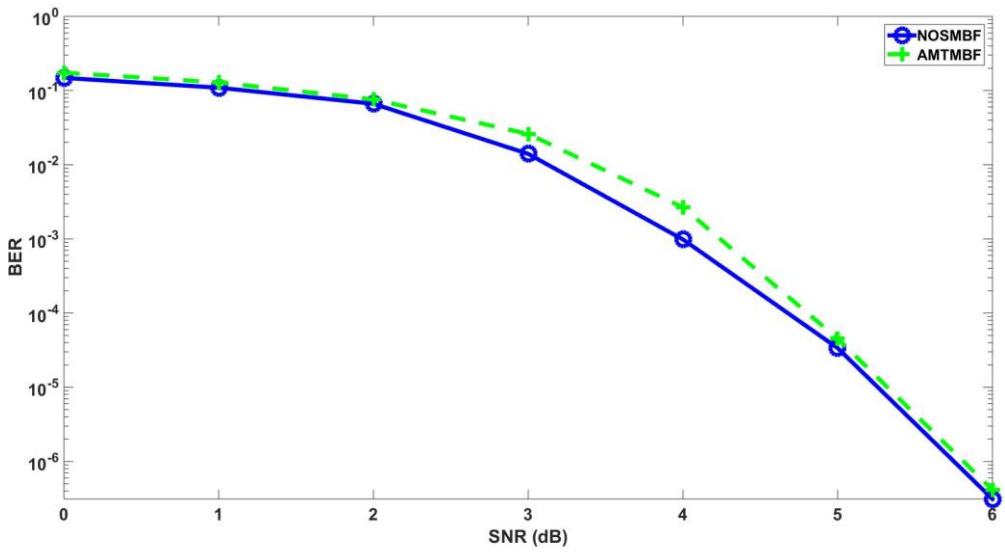


Figure 7. 1: Proposed algorithms BER performance analysis

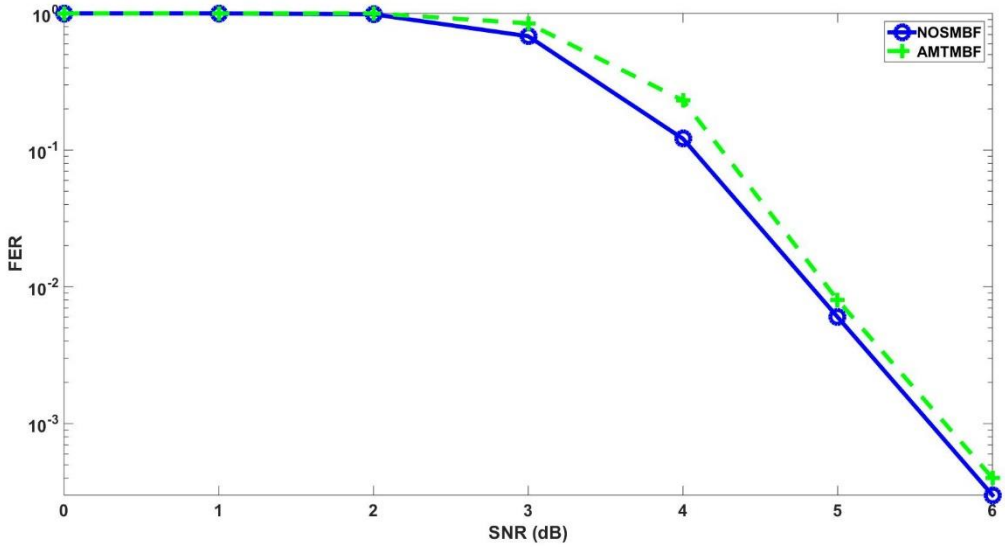


Figure 7. 2: Proposed algorithms FER performance analysis

Figure 7.1 and Figure 7.2 is a plot of the BER and FER performance of both algorithms respectively. It is observable that the NOSMBF has better performance for the simulated SNR range. In both performance comparisons, both the AMTMBF and the NOSMBF have a similar slope or curve profile and this can be attributed to the identical way in which both algorithms modify the codewords' soft information. The performance gain by the NOSMBF is a result of the more efficient way in which it identifies and corrects erroneous bits thus the near-optimal thresholds are more ideal in this case when compared to adaptive thresholds.

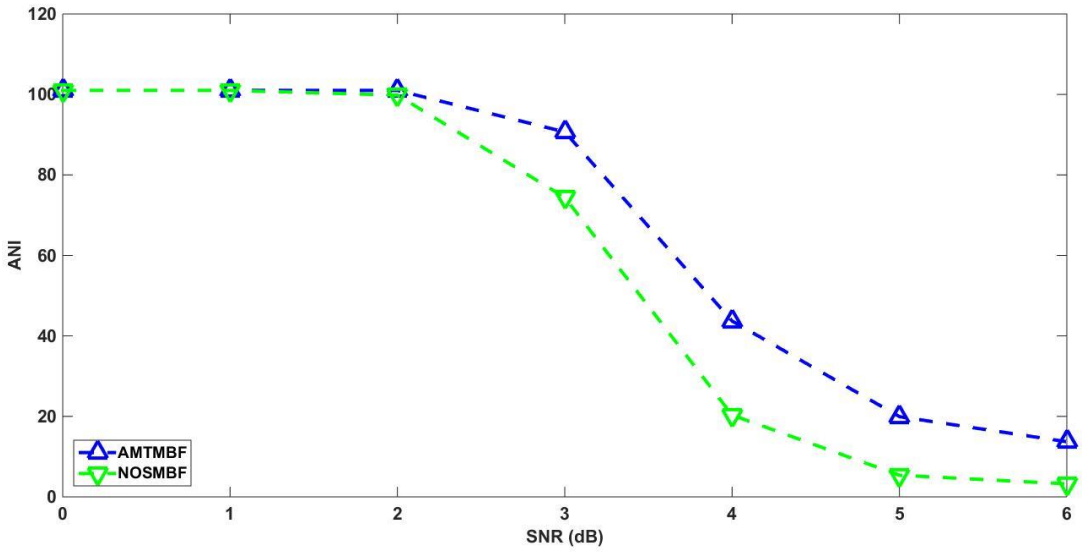


Figure 7. 3: Proposed algorithms ANI performance analysis

Figure 7.3 shows the comparative results of the two proposed algorithms in terms of the average number of iterations (ANI). Both algorithm ANI curves are characterised by similar slope profiles which is influenced to a large extend by the manner in which both algorithms modify the codeword channel soft information as observed in Figure 7.3. At low SNR, both algorithms reach the maximum number of iterations without converging. At intermediate signal to noise ratio the NOSMBF picks up ahead of the AMTMBF in terms of ANI performance and stays ahead all the way to 6 dBs. At 4 dBs the NOSMBF requires roughly around 50 % the number of iterations required by the AMTMBF which reduces to about 20% at 5 dBs and 6dBs.

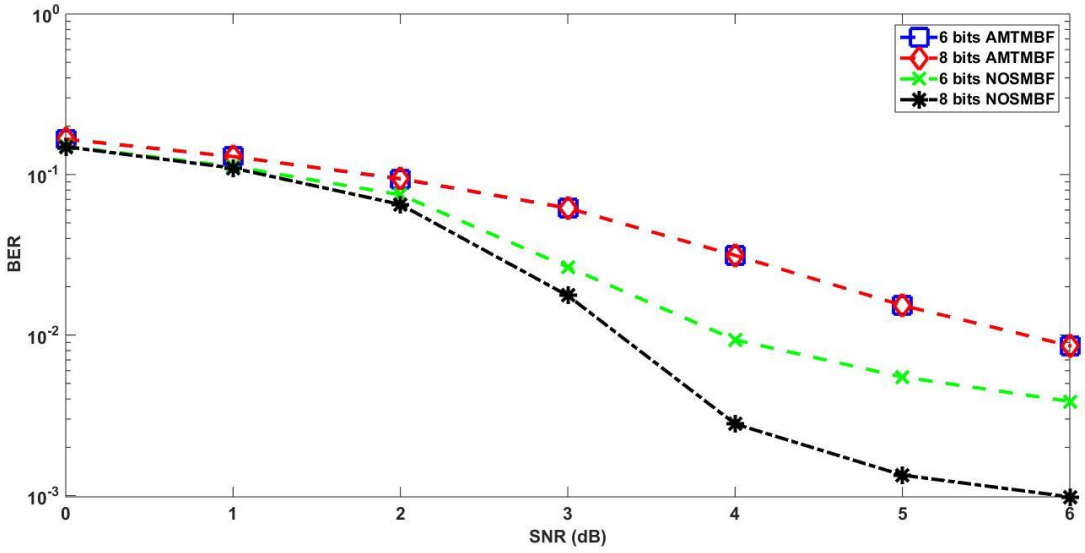


Figure 7. 4: Proposed algorithms' uniform quantization BER performance analysis

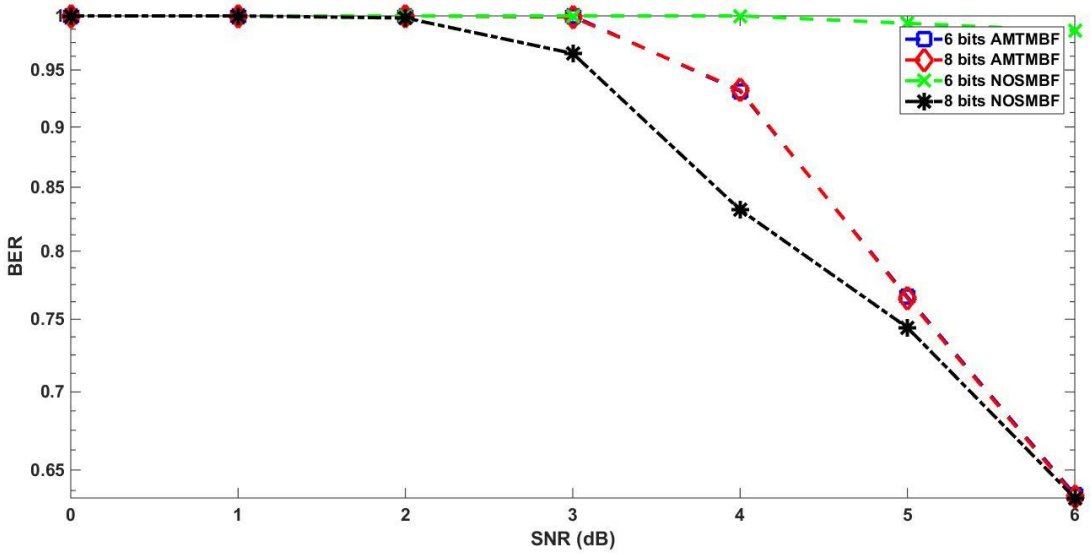


Figure 7. 5: Proposed algorithms uniform quantization FER performance analysis

Figure 7.4 and Figure 7.5 depict the performance of the quantized versions of the proposed algorithms. As discussed earlier, the 6 bit and 8 bit uniform quantization scheme makes no notable difference in the performance of the AMTMBF which suggests that a much larger bit size is required to improve performance. The uniformly quantized NOSMBF has a slight performance advantage which improves with increase in the number of quantization bits even though there is a noticeable manifestation of error floors which can be remedied by increasing the number of quantization bits.

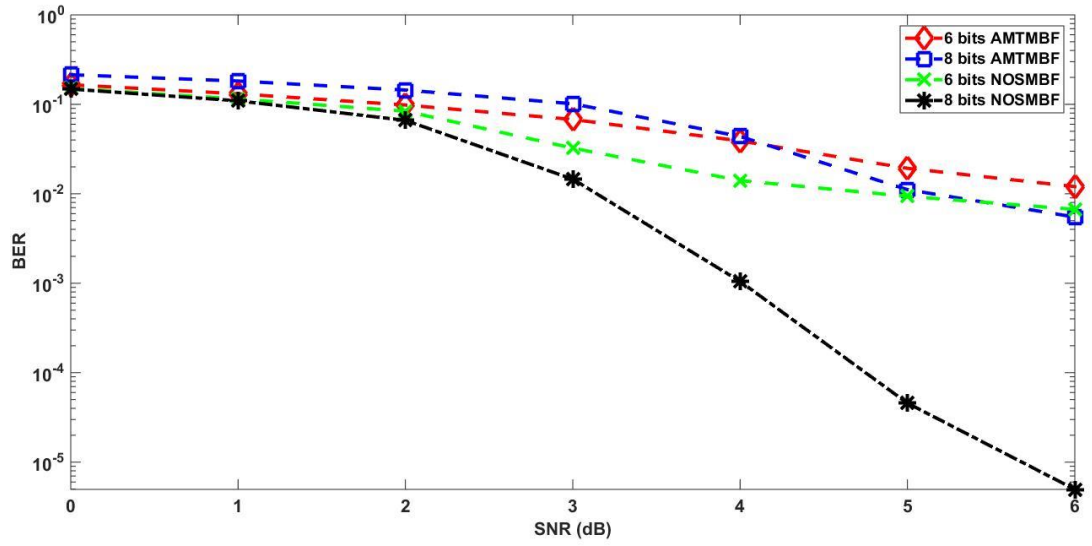


Figure 7. 6: Proposed algorithms non-uniform quantization BER performance analysis

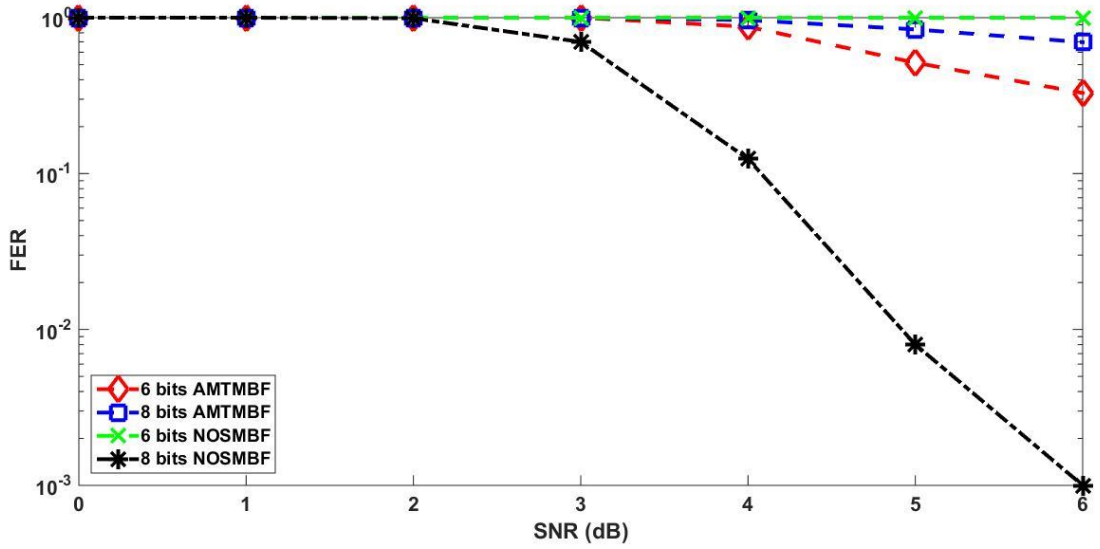


Figure 7. 7: Proposed algorithms non-uniform quantization FER performance analysis

The Figures 7.6 and 7.7 show the performance of the non-uniformly quantized versions of the AMTMBF and the NOSMBF algorithms. The 6 bit AMTMBF has a better FER performance when compared to both the 8 bit AMTMBF and 6 bit NOSMBF. The 6 bit AMTMBF also has better BER performance when compared to the 8 bit AMTMBF which eventually improves slightly at high SNR. The most notable result is that of the 8 bit non-uniformly quantized NOSMBF which is significantly well ahead of the all the other non-uniformly quantized algorithms.

7.3 Discussion

Both the adaptive and near optimal threshold schemes have proved to be effective techniques when it comes to improving the performance of LDPC decoding algorithms. This chapter has focussed on the comparison of the two techniques to determine the most effective of the two threshold determination methods. The NOSMBF algorithm emerged as the best algorithm over the AMTMBF for both the quantized and unquantized versions.

Similar curve profiles seem to suggest that the soft information modification scheme plays a major role in the BER, FER and ANI performance of both algorithms. A different soft information scheme is expected to yield a different curve profile as well.

The NOSMBF will be more suitable for application in a channel whose noise behaviour can easily be characterised and estimated in which case near optimal thresholds can easily obtained

via simulation for any given signal to noise ratio. The AMTMBF is more suitable for application in a situation where the channel parameters are not known and cannot be characterised, therefore, the flipping threshold is obtained on-the-go, depending on the state of the channel and algorithm at that particular time.

8. Conclusion

8.1 Validity of results

The proposed algorithms, the AMTMBF and the NOSMBF were simulated with various LDPC codes namely, Geometric Cyclic codes – Cyclic (1008, 504); Progressive Edge Growth Regular codes – PEGReg (1008, 504); Gallager random codes – Gal (408, 204) and Progressive Edge Growth Regular codes – PEGReg (504, 252) as show in Section 4.5 and Section 5.5. It then emerged that, the most suitable codes for the proposed algorithm are the PEGReg (1008, 504) codes. Amongst the algorithms used in the comparative performance analysis, none of them have been simulated using the PEGReg (1008, 504) codes. In literature, the GDBF described in Section 3.2.1 uses PEGReg (1008, 504) codes in its simulation. The single bit flipping GDBF has been simulated using the same codes and the results have been compared with those documented in [39] to ascertain the correctness of the simulation environment as well as the integrity of the results obtained. The test environment used is:

- Software – Matlab 2014
- Hardware – Intel Quad Core i7, 8 Gig RAM

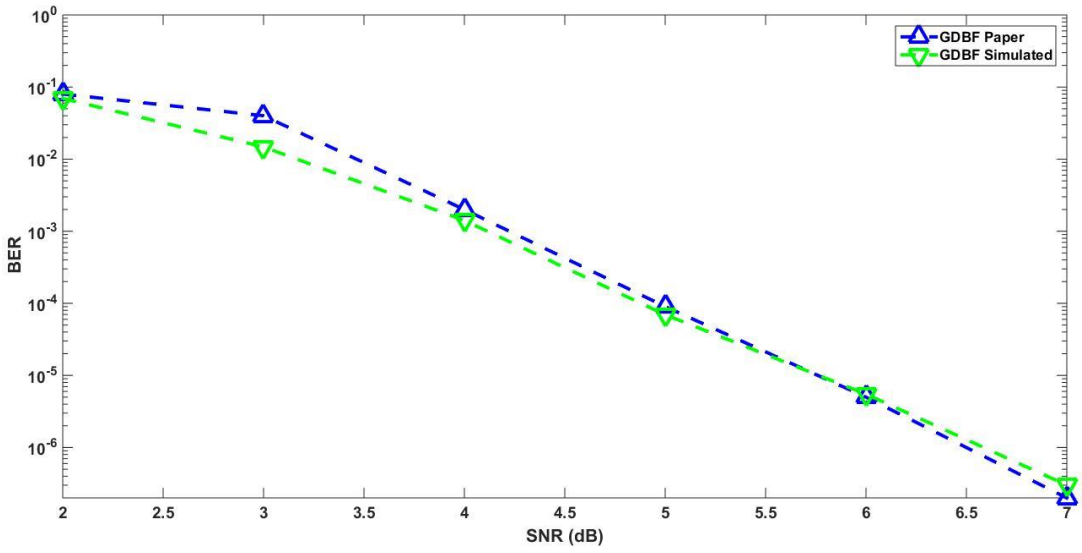


Figure 8. 1: Verification of Simulated results using the GDBF and PEGReg (1008, 504)

The results obtained using the identical single bit flipping GDBF and PEGReg (1008, 504) codes are shown in Figure 8.1. The *GDBF Paper* curve was obtained from [39] while the *GDBF Simulated* curve was produced by the simulation environment. It can be seen from Figure 8.1 that the *GDBF Paper* curve and the *GDBF Simulated* curve are almost identical with exception

to the SNR value of 3 dBs where there is a small discrepancy. However, the results obtained show that the testing environment is capable of providing valid results particularly with the chosen PEGReg (1008, 504) codes.

8.2 Future Work

While it has been established that multi-adaptive thresholds and near optimal SNR based thresholds improve the performance of multi-bit flipping algorithms, there is more work that is still outstanding to improve the overall understanding of this work. The following suggestions have been made for future work:

- The computational complexity of the proposed algorithms has not yet been determined. The only work done with any relation to complexity is with regard to the average number of iterations (ANI).
- The soft information modification scheme is a simple offset based scheme. In other papers, there are various soft information modification functions. The next step is to determine the optimal soft information modification scheme for both the AMTMBF and the NOSMBF.

The subset of linear block codes known as low density parity check (LDPC) codes have steadily risen in popularity due to their Shannon limit approaching performance. Initially, when they were invented in 1963 by Robert Gallager, they did not garner a lot of attention because of the sheer complexity that was associated with decoding them using the sum-product algorithm (SPA). In that era, the technology to analyse, develop and improve upon the SPA. As technology advanced, LDPC codes where rediscovered by MacKay and Neal in 1993. The purpose of this research was to improve upon a subsection of sub-optimal bit-flipping decoding algorithms whose most attractive quality is a level of complexity that is far below that of the SPA. However, this low complexity is attained at the cost of error performance. Therefore, this research focussed on multi-bit flipping LDPC algorithms which are threshold based. Two different threshold schemes, the adaptive multi-threshold scheme and the near-optimal fixed multi-threshold were investigated and two new LDPC multi-bit flipping algorithms where proposed. Both algorithms also make use of soft information modification to improve decoding performance.

The first algorithm proposed, the adaptive multi-threshold multi-bit flipping (AMTMBF) algorithm used the adaptive multi-threshold scheme to identify erroneous bits. The algorithm uses the IMWBF inversion function to determine the reliability of each codeword bit. The adaptive thresholds determine the three different sections of reliability in which all the codewords fall into. The algorithm proceeds to modify the associated soft information based on which section each bit has fallen into. Simulation results show that the new algorithm exhibits improved performance compare to some existing algorithms in terms of bit error rate (BER), frame error rate (FER) and average number of iterations (ANI).

The proposed second algorithm, the near optimal SNR dependent multi-bit flipping algorithm (NOSMBF) uses the near optimal SNR dependent multi-threshold scheme to identify erroneous bits that need to be corrected. Similar to the AMTMBF, the NOSMBF uses the IMWBF inversion function to calculate the reliability measurement of all the bits in the codeword. The multiple thresholds used in the algorithm demarcate the sections bits with different reliability measurements fall into. Each section results in a unique soft information modification to improve the decoding performance of the algorithm. Comparison of the proposed algorithm with existing algorithms has shown that the proposed algorithm has improved performance when compared to the existing algorithms in terms of bit error rate (BER), frame error rate (FER) and average number of iterations (ANI).

Quantized versions of both algorithms where algorithms where also simulated. Both the uniform and the non-uniform quantization scheme where used. It was observed that quantization does indeed significantly reduce the performance of both algorithms. This is due to the fact that soft information modification is integral to the design of both algorithms, therefore, loss of information due to quantization results in degraded performance. The 8- bit Lloyd-Max MMSE non-uniformly quantized NOSMBF is almost impervious to the effects of quantization as its performance is almost as good as the unquantized version.

The two algorithms have been compared to some of the existing LDPC bit flipping algorithms namely the AWMBF, SBF and the IMWBF. It has been observed that the new algorithms perform better than the pre-existing algorithms in terms of BER and FER. At high SNR, the SBF performs better than the AMTMBF in terms of ANI. The 6 bit and 8 bit uniformly quantized versions of the new algorithms do not perform better than the quantized pre-existing algorithms. The 8-bit non-uniformly quantized NOSMBF algorithm performs very close to the non-quantized NOSMBF.

The two algorithms where compared against each and it emerged that even though the adaptive threshold scheme AMTMBF had a notable improvement over the existing algorithms, the near-optimal SNR dependent threshold scheme NOSMBF emerged as the best threshold best algorithm of the two in all aspects. The NOSMBF is a suitable algorithm for channels that can be characterised or estimated but for channels where this is impossible, the AMTMBF which generates thresholds on the go is more suitable.

It has therefore been established that adaptive and near optimal SNR dependent multi-threshold schemes improve the decoding performance of multi-bit flipping LDPC algorithms in response to the main research question posed in the research proposal. It is also possible to determine the near optimal thresholds heuristically given a parity check matrix and some channel output to use as simulation training data. There are still opportunities for further work to be carried out. Due to time constraints it was not possible to determine the effect of increasing the number of thresholds for both algorithms. The analytical determination of the NOSMBF thresholds is also still outstanding which may also lead to the discovery of optimal thresholds for high SNR regions. There is also still opportunity to experiment with various soft information modification schemes.

References

[1] R. W. Hamming, “Error Detecting and Error Correcting Codes,” *The Bell System Technical Journal*, vol. 29, no. 2, pp. 147-160, 1950.

[2] C. E. Shannon, “A Mathematical Theory of Communication,” *The Bell System Technical Journal*, vol. 27, pp. 379-423, 623-656, 1948.

[3] R. G. Gallager, “Low Density Parity Check Codes,” *IRE Trans. Inform. Theory vol. IT-8*, pp. 21-28, 1962.

[4] D. Seo, The 1G (First Generation) Mobile Communications Technology Standards, 2013.

[5] J. Scourias, “Overview of the Global System for Mobile Communications,” University of Waterloo, Ontario, Canada, 1996.

[6] P. Kumar, “LDPC Coding for Performance Enhancement of MC-CDMA System,” Department of Information technology,Sri Venkateswara College of Engineering, Sriperumbudur – 602105, Chennai, INDIA, 2013.

[7] E. Abboud , D. Gurram and J. Pearce , “4G - 4th Generation Mobile Communication Networks,” University of Technology, Sydney.

[8] F. Mattoussi, B. Sayadi and V. Roca, “Good coupling between LDPC-Staircase and Reed-Solomon for the design of GLDPC codes for the Erasure Channels,” Inria, Alcatel-Lucent Bell Lab, France, 2013.

[9] S. H. Gupta and B. Virmani, “LDPC for Wi-Fi and WiMAX Technologies,” Deptt. of Electronics and Communication, ASET, Amity University, Noida, India, 2009.

[10] W. S. Shaker, “DVB-S2 LDPC Finite-Precision Decoder,” DVB-S2 LDPC Finite-Precision Decoder, Cairo, 2011.

[11] T. C. Chen and K. L. Lin, “An Adaptive Weighted Multibit-Flipping Algorithm for Decoding LDPC Codes,” Hsinchu, Taiwan, 2013.

[12] J. Cho, J. Kim and W. Sung, “VLSI Implementation of a High-Throughput Soft-Bit-Flipping Decoder for Geometric LDPC Codes,” *IEEE Transactions on Circuits and Systems—I: Regular Papers*, vol. 57, no. 5, 2010.

[13] D. J. MacKay, “Good error-correcting codes based on very sparse matrices,” *IEEE Transactions, Information Theory*, vol. 45, no. 3, pp. 399-431, 1999.

[14] R. M. Tanner, “A recurcive approach to low complexity codes,” *Information Theory*, vol. 27, pp. 533-547, 1981.

[15] T. J. Richardson and R. L. Urbanke, “Efficient encoding of low-density parity check codes,” *IEEE Transactions on Information Theory*, vol. 47, no. 2, pp. 1570-1579, 2001.

[16] S. Johnson, “Introducing Low-Density Parity-Check Codes,” School of Electrical Engineering and Computer Science, The University of Newcastle, Australia.

[17] D. MacKay, “Encyclopedia of Sparse Graph Codes - Inference Group,” [Online]. Available: www.inference.phy.cam.ac.uk/mackay/codes/data.html. [Accessed 26 February 2017].

[18] E. Eleftheriou and S. Oelcer, “IEEE international conference on communications.,” *Low-Density Parity-Check Codes for Digital Subscriber Lines,* vol. 3, pp. 1752-1757, 2002.

[19] M. G. Luby, M. Mitzenmacher,, A. Shokrollahi, and D. A. Spielman, “Improved Low-Density Parity-Check Codes Using Irregular Graphs,” *IEEE Transaction on Information Theory,* vol. 47, no. 2, pp. 585-598, 2001.

[20] S. J. Johnson and S. R. Weller, “Construction of low-density parity-check codes from Kirkman triple systems,” in *IEEE Globecom 2001,* San Antonio, TX, 2001.

[21] M. P. Fossorier, “Quasi-Cyclic Low-Density Parity-Check Codes From Circulant Permutation Matrices,” *IEEE Transactions on Information Theory,* vol. 50, no. 8, pp. 1788-1793, 2004.

[22] D. Divsalar, H. Jin and R. J. McEliece, “Coding theorems for ‘turbo-like’ codes,” in *36th Allerton Conf. on Communication, Control, and Computing,* Allerton, Illinois, 1998.

[23] H. Jin, R. McEliece and A. Khandekar, “Irregular Repeat–Accumulate Codes,” in *Second International,* Brest, France, 2000.

[24] M. Yang, M. E. Ryan and Y. Li, “Design of efficiently encodable moderate-length high-rate irregular LDPC codes,” *IEEE Trans. on Communications,* vol. 52, pp. 564-571, 2004.

[25] F. Zarkeshvar and A. H. Banihashemi, “On Implementation of Min-Sum Algorithm for Decoding Low-Density Parity-Check (LDPC) Codes,” *Global Telecommunications Conference, 2002. GLOBECOM '02. IEEE (Volume:2),* 2002.

[26] J. Chen and M. P. Fossorier, “Density Evolution for Two Improved BP-Based Decoding Algorithms of LDPC Codes,” *IEEE Communication Letters,* vol. 6, no. 5, pp. 208-210, 2002.

[27] Y. Kou and M. P. Fossorier, “Low-Density Parity-Check Codes Based on Finite Geometries: A Rediscovery and New Results,” *IEEE Transactions on Information Theory, Vol. 47, No 7,* pp. 8-9, 2001.

[28] A. Nouh and A. H. Banihashemi, “Bootstrap Decoding of Low-Density Parity-Check Codes,” *IEEE Communication Letters, Vol. 6, No.9,* vol. 6, no. 9, pp. 391-393, 2002.

[29] Y. Kou and M. P. Fossorier, “Low-Density Parity-Check Codes Based on Finite Geometries: A Rediscovery and New Results,” *IEEE Transactions on Information Theory,* vol. 47, no. 7, pp. 8-9, 2001.

[30] J. Zhang and M. P. C. Fossorier, “A Modified Weighted Bit-Flipping Decoding of Low-Density Parity-Check codes,” *IEEE Communications Letters Vol. 8, No. 3,* vol. 8, no. 3, pp. 165-167, 2004.

[31] M. Jiang, C. Zhao, Z. Shi and Y. Chen, “An Improvement on the Modified Weighted Bit Flipping Decoding Algorithm for LDPC Codes,” *IEEE Communications Letters,* vol. 9, no. 9, pp. 814-816, 2005.

[32] F. Guo and L. Hanzo, “Reliability Ratio Based Weighted Bit-Flipping Decoding for LDPC codes,” pp. 1-3, 2005.

[33] T. C. Chen, “Channel-Independent weighted bit-flipping decoding algorithm for low-density parity-check codes,” *IET Communications,* vol. 6, no. 17, pp. 1-5, 2012.

[34] G. Zhang, L. Zhou and H. Wen, “Modified Channel-Independent Weighted Bit-Flipping decoding algorithm for low-density-parity-check codes,” *IET Communications,* vol. 8, no. 6, pp. 1-4, 2013.

[35] S. Imani, R. Shahbazian and S. A. Ghorashi, “An Iterative Bit Flipping based Decoding Algorithm for LDPC Codes,” in *IEEE,* Shahid Beheshti University, G. C, Tehran, Iran, 2015.

[36] C. Wang, X. Wu and W.-P. Zhu, “A Self-Normalized Weighted Bit-Flipping Decoding,” in *IEEE,* Nanjing 210003, CHINA, 2016.

[37] H. Li, H. Ding and L. Zheng, “Hybrid Iterative Decoding for LDPC Codes Based on Gradient Descent Bit-Flipping Algorithm,” School of Electronic Science and Engineering National University of Defense Technology, Changsha, China, 2016.

[38] S. Khittiwitchayakul, W. Phakphisut and P. Supnithi, “Weighted Bit-Flipping Decoding for Product LDPC Codes,” Faculty of Engineering, King Mongkut's Institute of Technology Ladkrabang, Bangkok 10520, Thailand, 2016.

[39] T. Wadayama, K. Nakamura, M. Yagita, Y. Funahashi, S. Usami and I. Takumi, “Gradient Descent Bit Flipping Algorithms for Decoding LDPC Codes,” 2008.

[40] G. Dong, Y. Li, N. Xie, T. Zhang and H. Liu, “Candidate Bit Based Bit-Flipping Decoding Algorithm for LDPC Codes,” ECSE Department, Rensselaer Polytechnic Institute, NY, USA, 2009.

[41] T. M. N. Ngatched, F. Takawira and M. Bossert, “An Improved Decoding Algorithm for Finite-Geometry LDPC Codes,” *IEEE Transactions Communications,* vol. 57, no. 2, pp. 1-5, 2009.

[42] H. Huang, Y. Wang and G. Wei, “Mixed modified weighted bit-flipping decoding of low-density parity-check codes,” *IET Communications,* vol. 9, no. 2, pp. 283-290, 2015.

[43] Z. Liu and A. Pados, “Low Complexity Decoding of Finite Geometry LDPC Codes,” *IEEE Trans Communication,* vol. 53, pp. 415-421, 2005.

[44] K. Ma, “Two-Staged Weighted Bit Flipping (WBF) Decoding Algorithm for LDPC Codes,” Beijing Institute of Astronautical Systems Engineering, Xián, China, 2015.

[45] M. Jiang, C. Zhao and Y. Chen, “An Improvement on the Weighted Bit Flipping decoding algorithm for LDPC codes,” *IEEE Communication Letters,* vol. 9, pp. 814-816, 2005.

[46] D. Declercq and C. Winstead, “Noise-Aided Gradient Descent Bit-Flipping Decoders approaching Maximum Likelihood Decoding,” in *9th International Symposium on Turbo Codes & Iterative Information Processing*, 2016.

[47] M. Fossorier, M. Mihaljevic and H. Imai, “Reduced Complexity iterative decoding of low-density parity check codes based on belief propagation,” *IEEE Trans. Comm,* vol. 47, no. 5, pp. 673-680, 1999.

[48] R. Palinki, M. P. Fossorier and J. S. Yedidia, “Iterative decoding of multiple-step majority logic decodable codes,” *IEEE Trans Communication,* vol. 55, no. 6, pp. 1099-1102, 2007.

[49] L. Ping and W. K. Leung, “Decoding Low Density Parity Check Codes with Finite Quantization Bits,” *IEEE Communications Letters,* vol. 4, no. 2, 2000.

[50] T. Zhang, Z. Wang and K. K. Parhi, *On Finite Precision Implementation of Low Density Parity Check Codes Decoder,* Minnesota, Minneapolis, USA: Dept of Electrical and Computer Engineering, University of Minnesota, 2001.

[51] A. D. Liveris and C. N. Georghiades, “On Quantization of Low-Density Parity-Check Coded Channel Measurements,” *GLOBECOM 2003,* 2003.

[52] Z. Zhang , L. Dolecek, B. Nikolic, V. Anantharam and M. J. Wainwright, “Design of LDPC Decoders for Improved Low Error Rate Performance: Quantization and Algorithm Choices,” *IEEE Transactions on wireless communications,* vol. 8, no. 11, 2009.

[53] S. Kim, G. Sobelman and H. Lee, “Adaptive Quantization in Min-Sum based Irregular LDPC Decoder,” 2008.

[54] J. Cho, J. Kim and W. Sung, “Optimal Output Quantization of Binary Input AWGN Channel for Belief-Propagation Decoding of LDPC Codes,” *IEEE Workshop on Signal Processing,* 2012.

[55] X. Zhang and S. H. Paul, “Quantized Min-Sum Decoders with Low Error Floor for LDPC Codes,” in *IEEE International Symposiumon Information Theory Proceedings,* Carlifornia, 2012.

[56] “Characteristic of a Linear Uniform Quantizer,” DSP, [Online]. Available: http://dsp-nbsphinx.readthedocs.io/en/nbsphinx-experiment/quantization/linear_uniform_characteristic.html. [Accessed 27 02 2017].

[57] S. P. Lloyd, “Least squares quantization in PCM,” *IEEE Transactions on Information Theory,* vol. 28, no. 2, pp. 129-137, 1982.

[58] J. Ming , C. Zhao, Z. Shi and C. Yu, “An Improvement on the Modified Weighted Bit Flipping Decoding Algorithm for LDPC Codes,” *IEEE Communications letters, Vol 9, No 9,* pp. 1-3, 2005.

[59] P. Elias, “Coding for two noisy channels,” Department of Electrical Engineering and Research Laboratory of Electronics, Massachusetts Insitute of Technology, Cambridge, Massachusetts, 1955.

[60] C. E. Shannon, “The zero error capacity of a noisy channel,” *IRE Transactions on Information Theory,* vol. 2, no. 2, pp. 112-124, 1956.

[61] C. Wang, X. Wu and W.-P. Zhu, “A Self-Normalized Weighted Bit-Flipping Decoding Algorithm for LDPC Codes,” Nanjing University of Posts and Telecommunications, Nanjing 210003, CHINA, 2016.

[62] H. Huang, Y. Wang and G. Wei, “Combined Modified Weighted Bit-Flipping Decoding of Low-Density Parity-Check Codes,” in *IEEE,* Sydney, NSW, Australia, 2012.