DECODING LDPC CODES WITH PROBABILISTIC LOCAL MAXIMUM LIKELIHOOD BIT FLIPPING

by

Rejoy Roy Mathews

A thesis submitted in partial fulfillment
of the requirements for the degree

of

MASTER OF SCIENCE

in

Computer Engineering

UTAH STATE UNIVERSITY
Logan, Utah

2020

ABSTRACT

Decoding LDPC Codes with Probabilistic Local Maximum Likelihood Bit Flipping

by

Rejoy Roy Mathews, Master of Science

Utah State University, 2020

Major Professor: Chris Winstead, Ph.D.
Department: Electrical and Computer Engineering

Low-density parity-check (LDPC) codes are high-performance linear error correcting codes with application to communication channels and digital storage media. LDPC codes are decoded using graph algorithms wherein a channel sample is decoded with the aid of information from its adjacent graph neighborhood, called the syndrome. This work studies the conditional probability of a channel error given syndrome information at a particular iteration to formulate a new algorithm called Probabilistic Local Maximum Likelihood Bit Flipping (PLMLBF). The PLMLBF algorithm uses a three dimensional Multi-iteration Probability Flip Matrix (MIPFM) to quantify the frequency of errors in a noise corrupted message frame being decoded using a specific LDPC code. The matrix is used to probabilistically decode noise corrupted message frames. The motivation for this work is to provide a theoretical framework for constructing probabilistic and noisy bit-flipping algorithms, such as the Noisy Gradient Descent Bit Flipping (NGDBF) algorithm, which up to now have been mainly heuristic in nature. For specific SNR values, the PLMLBF algorithm outperforms the deterministic multi-bit GDBF algorithm and the multi-bit NGDBF algorithm. PLMLBF does not outperform various heuristic improvements that have been developed for NGDBF decoders, but PLMLBF is the first probabilistic bit-flipping decoder with an explicit construction.

(67 pages)

PUBLIC ABSTRACT

Decoding LDPC Codes with Probabilistic Local Maximum Likelihood Bit Flipping

Rejoy Roy Mathews

Communication channels are inherently noisy making error correction coding a major topic of research for modern communication systems. Error correction coding is the addition of redundancy to information transmitted over communication channels to enable detection and recovery of erroneous information. Low-density parity-check (LDPC) codes are a class of error correcting codes that have been effective in maintaining reliability of information transmitted over communication channels. Multiple algorithms have been developed to benefit from the LDPC coding scheme to improve recovery of erroneous information. This work develops a matrix construction that stores the information error probability statistics for a communication channel. This combined with the error correcting capability of LDPC codes enabled the development of the Probabilistic Local Maximum Likelihood Bit Flipping (PLMLBF) algorithm, which is the focus of this research work.

ACKNOWLEDGMENTS

My experience at Utah State University has molded me into a well-rounded person and has left me with fond memories that I will always cherish. One of the best aspects about studying at Utah State University is the financial assistance made available by the School of Graduate Studies. I feel blessed to be able to benefit from the financial aid made possible by USU School of Graduate Studies, USU College of Engineering and the Nagendra Grandhi Fellowship.

My advisor Dr. Chris Winstead is one of the most patient, multi-faceted, innovative, professional and charismatic individuals I have met. One of his best qualities is his ability to understand a question's implied meaning and suggest solutions that one can easily comprehend. As I wrap up my thesis, I am forever grateful for his confidence in me, the opportunities he gave me as a TA and the financial backing he has given me through my time at USU. I am also grateful to Joe Shope, who I had the privilege to work with as an IT Assistant. His hunger to learn new things, his attitude towards tackling problems, his people management and leadership skills are something I really admire and try to adopt. He has always looked out for me and has had faith in my abilities.

I am thankful to my committee members, Dr. Zhen and Dr. Davidson for reviewing my thesis and suggesting improvements. Tasnuva and Rakin have been great lab-mates at the LEFT lab and have always motivated me through the course of my research. My deep conversations with Rakin on varied subjects is something I will always cherish. I would like to thank Tricia, Kathy, Diane and Metta for all their assistance over the last couple of years. They have gone out of their way to make sure that my experience at USU was nothing short of amazing.

My experience would have been incomplete without the moments I have spent with some great friends I have met along the way. I am thankful to Navin for teaching me the importance of having a well-rounded personality and a confident attitude, Sheril for teaching me the importance of discipline to maintain an optimal lifestyle, Alora for being a great friend and being a mentor into understanding American culture, Krishna for his shared enthusiasm in science and engineering, Aditi, Pratyusha and Kristi for the deep and meaningful conversations we have had, Deepesh for the healthy debates we used to have and Melvin for the financial assistance and long lasting friendship.

I am ever grateful to my extended family for their financial and moral support. Lastly, I would like to thank my parents and sister for their everlasting support for all the decisions I make and the path I choose to follow.

Rejoy R. Mathews

CONTENTS

LIST OF TABLES

LIST OF FIGURES

NOTATION

H	$m \times n$ parity check matrix
k	Uncoded message length
$\vec{c} \in F_2^n$	Binary codeword $\{H\vec{c} = 0\}$
$\hat{c} \in \{+1, -1\}^n$	Bipolar codeword $\{\hat{c} = 1 - 2\vec{c}\}$
G	Generator matrix
n	Codeword length
$N_0 \in \mathbb{R}$	Channel noise spectral density
R	Code rate
$\sigma \in \mathbb{R}$	Channel noise standard deviation
$\hat{z} \in \mathbb{R}^n$	Additive white gaussian noise vector
$\hat{y} \in \mathbb{R}^n$	Noisy channel message frame $\{\hat{y} = \hat{c} + \hat{z}\}$
$\hat{x} \in \{+1, -1\}^n$	Bipolar hypothesis $\hat{x} = \text{sign}(\hat{y})$
$\vec{x} \in F_2^n$	Binary hypothesis $\vec{x} = 0.5(1 - \hat{x})$
$\tilde{C}$	Valid decision vector list
$\mathbf{x}_{\text{ml}}$	Maximum likelihood decision vector
$\sigma_n^2 \in \mathbb{R}$	NGDBF perturbation noise variance
$\theta \in \mathbb{R}$	GDBF flipping threshold
$\eta \in \mathbb{R}$	Noise scale in NGDBF
$w \in \mathbb{R}$	Syndrome weight
$a \in \mathbb{R}$	NGDBF perturbation noise
T	Decoding iterations
$Y_{\max} \in \mathbb{R}$	Clipping value for noisy channel samples
$\hat{q} \in \mathbb{R}^n$	Quantized noisy channel samples within $[-Y_{\max}, Y_{\max}]$

F	Frame Count
d_v	Variable node degree
d_c	Check node degree
p	syndrome parameter
$\mathcal{P}_\ell$	PFM for iteration ℓ
$\mathcal{H}_\ell$	Sample frequency matrix for iteration ℓ
$\mathcal{E}_\ell$	Sample error frequency matrix for iteration ℓ
ω	List of possible syndrome values
λ	List of possible values for xq

ACRONYMS

ASIC	Application-specific integrated circuit
AWGN	Additive white Gaussian noise
BAWGNC	Binary Additive white Gaussian noise channel
BSC	Binary Symmetric Channel
BER	Bit Error Rate
BF	Bit flipping
BP	belief-propagation
ECC	Error correcting code
ES-AT-GDBF	early stopping Adaptive Threshold Gradient Descent Bit Flipping
FER	Frame Error Rate
FPGA	Field Programmable Gate Array
GDBF	Gradient Descent Bit Flipping
LDPC	low-density parity-check
MAP	maximum a posteriori
MIPFM	Multi-iteration Probability Flip Matrix
ML	maximum-likelihood
M-NGDBF	multi-bit Noisy Gradient Descent Bit Flipping
MP	Message-Passing
MS	Min-Sum
NGDBF	Noisy Gradient Descent Bit Flipping
OMS	Offset Min-Sum
PEG	Progressive Edge Growth
PFM	Probability Flip Matrix
PGDBF	Probabilistic Gradient Descent Bit-Flipping

PLMLBF	Probabilistic Local Maximum Likelihood Bit Flipping
SM-NGDBF	Noisy Gradient Descent Bit Flipping with Smoothing
SM-PLMLBF	Probabilistic Local Maximum Likelihood Bit Flipping with Smoothing
S-NGDBF	single-bit Noisy Gradient Descent Bit Flipping
SNR	Signal-to-noise ratio
SPA	sum-product algorithm
QC	Quasi-Cyclic
VO-PLMLBF	Probabilistic Local Maximum Likelihood Bit Flipping with 2 out of 3 voting

CHAPTER 1
INTRODUCTION

Transmission of information over any communication channel is affected by the channel noise leading to information error. Error correction coding is the addition of redundant information to a transmitted message in order to correct the errors induced by the communication channel noise. The decoding problem involves recovering near error free information from the received information. The *Shannon Capacity Limit* has been known to impose a bound on the maximum achievable performance of different error decoding systems [1] that try to solve this decoding problem. In his paper published in 1948, Shannon demonstrated that a proper coding scheme can transmit data on a noisy channel with a rate less than the channel capacity with a small frequency of error. Turbo Codes, published in 1993 were first known to achieve these limits [2]. Soon it was discovered that LDPC [3] was another class of codes to have performance close to the achievable limits. Low-density parity-check (LDPC) codes are linear block codes, defined by a binary sparse parity check matrix containing mostly 0's and relatively few 1's [3] was first proposed by Robert G. Gallager in his doctoral dissertation in 1963. Among many other uses, LDPC codes have found applications in IEEE 802.3 standards for 10GBase-T Ethernet and IEEE 802.11 standards for Wi-Fi [4,5].

In addition to the type of code used, decoding is impacted by the quality of the decoding algorithm used. Most decoding algorithms have a decoding mechanism that is informed by the rules defined by maximum a posteriori (MAP) or maximum likelihood (ML) decoding. The MAP decoding problem involves maximizing the decoded code symbol conditioned to a received channel sample. The belief-propagation (BP) decoding algorithm and the approximate min-sum (MS) decoding algorithm are MAP decoding algorithms for LDPC codes and offer the best performance over the binary additive white Gaussian noise channel (BAWGNC) with reasonable implementation cost. However, these algorithms are complex to be realized on hardware because of the large number of arithmetic operations that need to be repeated over multiple iterations [6,7]. These algorithms also need to be implemented with a degree of parallelism to meet the high throughput requirements of modern communication systems [8,9], further increasing hardware complexity.

The ML decoding algorithm involves finding a codeword that has the largest correlation with the received channel message frame. A channel message frame is an n-sample vector that is received over a communication channel. Gallager proposed a ML decoding algorithm called the bit-flipping (BF) algorithm which flips bits in the initial hypothesis of the received channel message frame till all the parity check equations of the LDPC parity check matrix are satisfied. Many variants of the BF algorithm proposed by Gallager [3] are presently used as low complexity decoding algorithms for LDPC codes. Gradient Descent Bit Flipping (GDBF) [10], a BF decoding algorithm proposed by Wadayama et. al. considers the ML decoding problem as an objective function for gradient descent optimization on a BAWGNC. The GDBF algorithm however has a tendency of getting stuck in a local maxima which does not allow convergence on a codeword. Non-convergence results in sub-optimal Bit Error Rate (BER) performance, which is a metric to measure a decoders performance. Sundararajan et. al. proposed the Noisy Gradient Descent Bit-Flip (NGDBF) [11] algorithm which adds a random perturbation to the objective function in GDBF to escape the local maxima and aid convergence. The random perturbation used in NGDBF decoding, in works published till date is a Gaussian perturbation [11].

The objective function in GDBF and NGDBF makes use of the channel sample information, which is the value of each individual sample in the received channel message frame. This is combined with the syndrome parameter, which is the information available from the parity check matrix about the correctness of a channel samples decoded value. These algorithm's keep flipping the channel message frames initial hypothesis based on the objective function, till the decoder converges on a codeword or till the maximum number of decoding iterations has been reached. With the objective function as a starting point, this research work studies the conditional probability of a channel sample error given the syndrome information to formulate the Probabilistic Local Maximum Likelihood Bit Flipping (PLMLBF) algorithm.

Quantifying channel sample values and computed syndrome values during BF decoding of multiple channel message frames can aid estimating error probability in a received channel sample. In this work, multiple n-bit all '0' codeword are transmitted over a communication channel and the probability of error in the received channel sample values hard decision, given its syndrome is estimated. The all '0' codewords are modulated before transmission over a BAWGNC. A binary '0' corresponds to a '+1' after modulation and binary '1' corresponds to a '-1' after modulation. A hard decision is the hypothesis as to whether the received channel sample is equal to a '+1' or a '-1'. For an all '+1' codeword, a value of '-1' in the hard decision indicates a decoding error. In this work 200,000 frames of the all '+1' codeword is transmitted to develop informative error probability statistics. Error probability is the ratio of the erroneous channel samples count to the total channel samples count, for a specific channel sample value with a specific syndrome. The estimated error probability is stored in a matrix, termed as the Probability Flip Matrix (PFM).

The PFM can be indexed with a specific channel sample value $\times$ decoding hypothesis and syndrome value for a specific iteration of decoding. Similar to the GDBF algorithm, the code symbols in the initial hypothesis that was made for all the 200,000 channel message frames are now probabilistically flipped with the value obtained from the matrix for that iteration of decoding. This completes one iteration of PFM computation and PFM application for probabilistic flipping. This cycle of PFM computation and application is repeated for T iterations generating a PFM for each iteration of decoding. Once all the PFM's are generated, it can then be used to decode any channel message frame transmitted over a communication channel with the same Signal-to-noise Ratio (SNR) for T decoding iterations.

1.1 Scope

The scope of this research includes:

- **Algorithm Development:** Develop the Probabilistic Local Maximum Likelihood Bit Flipping(PLMLBF) decoding algorithm which can probabilistically correct erroneous channel message samples. Development of this algorithm is based on empirically generating a Multi-iteration Probability Flip Matrix (MIPFM) that quantifies the channel message errors given a specific syndrome.

- **Heuristic Improvements:** Develop certain heuristic improvements to the algorithms decoding performance. In the scope of this work, some heuristics are specific to the developed algorithm and other heuristics are incorporated from the NGDBF algorithm.

- **Develop a theoretical framework for the GDBF and NGDBF algorithms:** PLMLBF is an algorithm developed based on theoretical exploration of the parameters that affect GFDBF and NGDBF decoding. The findings from this work can form the statistical groundwork for the efficacy of the NGDBF and GDBF algorithms which have relied on heuristics for BER performance improvement.

1.2 Organization

The rest of the thesis is organized as follows:

- **Background:** A literature review of all the research work leading up to the development of the Gradient Descent Bit Flipping class of algorithms is presented. The literature review is a gateway to understanding the motivation behind development of the PLMLBF algorithm which draws from the fundamentals of the GDBF and NGDBF decoding algorithm (Chapter 2).

- **Motivation:** The theoretical motive behind development of the PLMLBF algorithm is presented in this chapter. The PLMLBF algorithm has a strong theoretical background and additional heuristics built on this background has led to BER performance comparable to similar BF algorithms. (Chapter 3).

- **Multi Iteration Probabilistic Flipping Matrix Construction:** Construction of the Multi Iteration Probabilistic Flipping Matrix (MIPFM), a matrix used to decode channel samples in a probabilistic manner is discussed. This matrix is constructed by studying error occurrences in 200,000 channel message frames received over a communication channel with a specific SNR value. 200,000 channel frames ensures that the computed flipping probabilities are independent of small sample size anomalies (Chapter 4).

- **Simulation Results:** The impact of varying different parameters that affect the decoding performance are presented in the form of BER and FER performance curves. This section also discusses about the impact of heuristics specific to the PLMLBF algorithm and the impact of heuristics developed for the NGDBF algorithm on the performance of the PLMLBF algorithm. (Chapter 5).

- **Conclusions and Future Work:** This chapter highlights some of the key research findings and drawbacks of this work. Hardware implementation of the PLMLBF algorithm is proposed as a future scope of work. (Chapter 6).

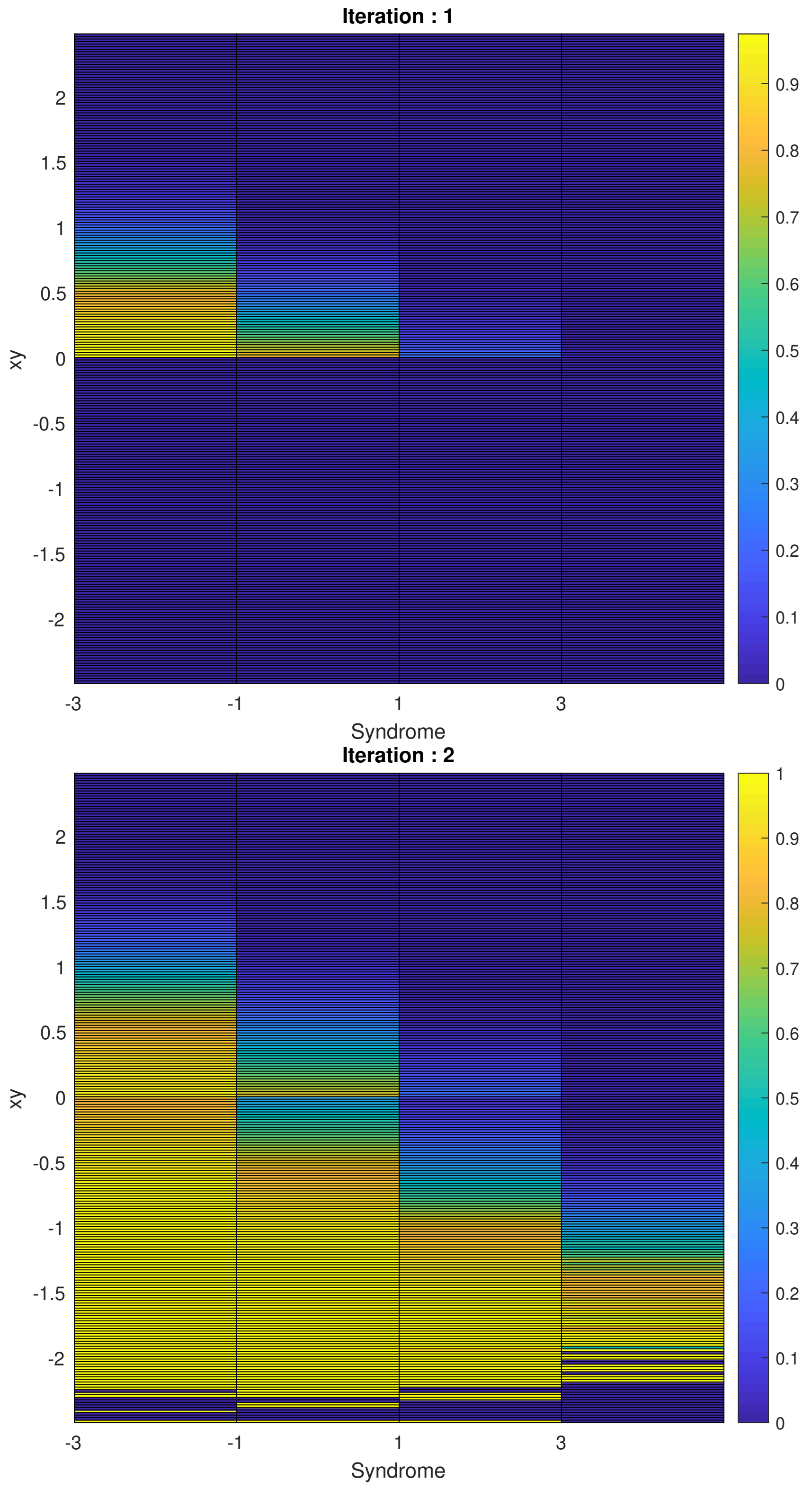


Fig. 1.1: PFM's for 2 consecutive iterations of the MIPFM. The PFM's are constructed by estimating probability of errors in 200,000 channel message frames received over a communication channel with an SNR of 4 dB. The frames were quantized using 8-bit quantization and are saturated at 2.5 before decoding using the 1/2 PEGReg504x1008 LDPC code for MIPFM construction.

CHAPTER 2
BACKGROUND

2.1 Linear Block Codes

A linear block code $\tilde{C}$ can be defined as a set of vectors called codewords which are of fixed length n and constitute a linear space. For the scope of this work we will consider the binary field F_2. For any two codewords $\vec{c}_1, \vec{c}_2 \in \tilde{C}$, the sum $\vec{c}_1 + \vec{c}_2$ is also in $\tilde{C}$. The all '0' codeword also lies in $\tilde{C}$ and is often used as the codeword for evaluating decoder performance. The set of codewords in $\tilde{C}$ are embedded in a much larger linear space F_2. On account, the odds of a bit error causing the codeword $\vec{c}_1$ being interpreted as another codeword is less likely as the neighborhood of $\vec{c}_1$ are not necessarily codewords. Hamming distance is a simple measure that is useful in understand distance between two codewords $\vec{c}_1$ and $\vec{c}_2$. Hamming distance is defined as the minimum number of positions in which $\vec{c}_1$ and $\vec{c}_2$ differ and is denoted as $h(\vec{c}_1, \vec{c}_2)$. Hamming Codes [12] are considered as the first class of linear block codes that was developed by Richard Hamming and has been widely adopted in Error Correction Coding.

In linear block codes, k bits of information are coded for redundancy to obtain an n bit codeword $\vec{c}$, by multiplying the k information bits with the Generator Matrix, often denoted as the G-matrix. The G-matrix has matrix dimensions $k \times n$. In the codeword $\vec{c}$, $(n-k)$ are parity bits that are used to detect and correct information bit errors during transmission. The information digits are the first k bits of the codeword.

2.2 LDPC Codes

Low-density parity-check (LDPC) codes are a class of linear block codes characterized by sparsely populated parity check matrices with mostly 0's and relatively few 1's. The parity check matrix, also called the H-matrix is used for decoding the received noisy channel samples. In the $m \times n$ H-matrix, the m rows represent the parity check equations that need to be satisfied by the hypothesis vector for a received channel message frame to be counted as a codeword and the n columns represents the individual code symbols. The H-matrix is constructed to satisfy $GH^T = 0$. On account, the hypothesis vector for a channel frame when multiplied by the H-matrix will yield a $\vec{0}$ only if it is a codeword generated using the G-matrix.

LDPC codes can be represented as bipartite graphs called Tanner graphs. The two sets of vertices of the Tanner graphs are the variable nodes and the check nodes. An edge in a Tanner graph connects a variable node to a check node if it is a part of the parity check equation for a corresponding check node, and so the number of edges in a Tanner graph equals the number of 1's in an LDPC parity check matrix [13]. An H-matrix is represented in Eqn. 2.1 and its equivalent Tanner graph is represented in Fig. 2.1.

$$H = \begin{bmatrix} 1 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 \end{bmatrix} \tag{2.1}$$

2.3 Quasicyclic LDPC codes

Quasi-Cyclic (QC) LDPC codes [14] are a class of LDPC codes that are widely used in practical applications. QC codes are characterized by a base matrix B that is of size R $\times$ C, where R defines the rows of the base matrix and C defines the columns in the base matrix. Each element b_{ij} of the base matrix is characterized by an integral value such that $b_{ij} \geq -1$ with $i \in 1, 2, ..., R$ and $j \in 1, 2, ..., C$. The H matrix is constructed from the base matrix B by replacing b_{ij} by a circulant matrix, obtained by right-shifting a z$\times$z identity matrix by b_{ij} positions. b_{ij} with values equal to -1 are replaced by an all zero matrix of size z$\times$z. On account the constructed H matrix will have R$\times$z rows and C$\times$z columns. Due to the specific structure of QC LDPC codes, they are well suited for hardware realizations allowing for parallelizable architecture.

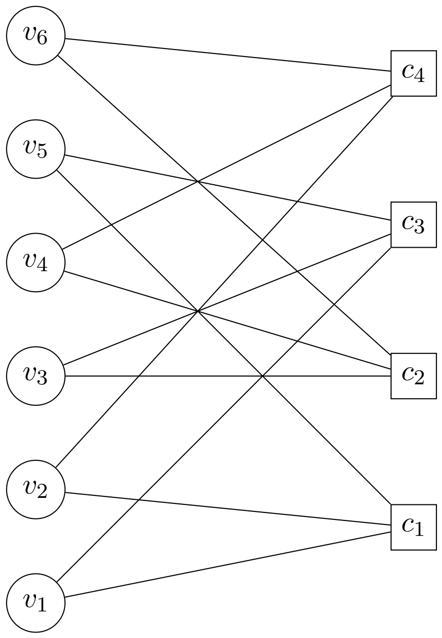


Fig. 2.1: Tanner Graph for H-matrix represented by Eqn. 2.1. The j^{th} variable node is labeled v_j in Tanner Graph and the i^{th} check node is labeled as c_i in the Tanner Graph. A 1 in the H matrix corresponding to the i_{th} row and the j_{th} column indicates an edge between the c_i check node and the v_j variable node.

2.4 Scheduling Strategies

Understanding different scheduling strategies is important for the hardware implementation of LDPC codes. In the implementation of a BP algorithm the check nodes and variable nodes communicate with each other by passing messages. This technique of decoding is also termed as Message-Passing (MP) decoding. The conventional scheduling strategy is the flooded scheduling [15] strategy in which, during each iteration of decoding all the check nodes and subsequently all the variable nodes pass messages to all its neighbors. This type of scheduling is also termed as two phase message-passing decoding with the check nodes being processed in phase one and the variable nodes being processed in phase two. Flooded scheduling runs into latency issues with an increase in the number of check nodes and variable nodes.

Another scheduling strategy is the layered scheduling [16] strategy in which the parity check matrix is divided into multiple layers L, with m/L parity check equations in each layer. Each time a layer is processed, the decoder updates the neighboring variable nodes and then proceeds to the next layer. This scheme enables faster information propagation and enables convergence in half the number of decoding iterations as the flooded scheduling scheme, in turn decreasing decoding latency [17].

2.5 LDPC Decoder Hardware Architecture

A typical hardware architecture of an LDPC decoder consists of a set of check node units, a set of variable node units and an interconnect network between the check nodes units and the variable nodes units to allow for message exchange. Additionally, there can be memory components that allow for storage of information and messages considered vital to the decoding process. LDPC decoder architectures can be broadly classified into three types - fully parallel, serial and partly parallel architectures. Fully parallel architectures are best suited for flooded scheduling as it comes with a large number of check node units and variable node units and a dense network of interconnections between the check node units and variable node units. This dense network leads to place and route issues because of wire congestion's and are seldom used for any practical applications.

Serial implementations are an alternative to the fully parallel architecture with only one check node unit and one variable node unit being reused to process all the check nodes and variable nodes in the LDPC tanner graph. This architecture is simpler to implement and does not have any of the problems related to wire congestion and placement and routing. However, this architecture has the drawback of significantly less throughput which may be too less for modern communication applications.

Partly parallel is a compromise between the fully parallel and serial architectures where the number of check node units and the variable node units is less than the number of check nodes and variable nodes in the tanner graph. The check node units and the variable node units are reused to process all the check nodes and variable nodes. The memory assists in storage of the computed message that can be used by the next set of variable nodes and check nodes that are being processed by the variable node units and check node units. These types of architectures are naturally suited for layered scheduling [18].

2.6 BSC Channel Transmission

As the name implies, both the input and output of a Binary Symmetric Channel (BSC) are binary. A bit in the n bit binary codeword that needs to be transmitted over the BSC is defined as c_n. On account of channel noise, c_n is flipped with a crossover probability α to obtain y_n. The probability that $y_n = c_n$ is α and the probability that $y_n = \text{not}(c_n)$ is $1 - \alpha$. A BSC is a common communication channel model because of its simplicity.

2.7 AWGN Channel Transmission

The n bit binary codeword to be transmitted over the BAWGNC is referred to as $\vec{c} \in \tilde{C}$ and satisfies the condition. $H\vec{c} = 0$. To transmit this binary codeword over the BAWGNC, it is converted into a bipolar codeword $\hat{c}$, defined by $\hat{c} \triangleq (1-2c_1), (1-2c_2), ..., (1-2c_n) : c_i \in \vec{c}$. When transmitting an all '0' n-bit binary codeword $\vec{c}$, the equivalent bipolar codeword is characterized by an all '+1' vector. Depending on the SNR, AWGN gets added to this vector. The AWGN can be denoted as an n bit vector $\hat{z} \in \mathbb{R}^n$. The channel noise power spectral density is denoted by N_0 whose definition is provided in Eqn. 2.2. R denotes the rate of the code and SNR denotes the signal-to-noise ratio of the channel. The standard deviation of the channel is defined as $\sigma = \sqrt{\frac{N_0}{2}}$.

$$N_0 = \frac{1}{R} \cdot 10^{\frac{-SNR}{10}} \tag{2.2}$$

For an all '+1' bipolar codeword ($\hat{c} = \{+1^n\}$), transmitted over a noisy channel each channel sample in the channel message frame $\hat{y} \in \mathbb{R}^n$, defined by $\hat{y} = \hat{c} + \hat{z}$ can be represented as a Gaussian distribution with mean '+1' and standard deviation σ, $\mathcal{N}(1, \sigma^2)$.

2.8 Bit Flipping Decoding

The count of columns in the H-matrix is equal to the number of symbols in the transmitted codeword. On account, each individual channel sample gets mapped to a variable node in the tanner graph for the H-matrix. For each variable node of the LDPC code, a hard decision is made as to whether the associated channel sample is a bipolar +1 or -1.

The vector of hard decisions for a channel frame can be referred to as the channel frame bipolar hypothesis and is defined as $\hat{x}$ where $\hat{x} = \text{sign}(\hat{y}) : \hat{x} \in \{+1, -1\}^n$.The equivalent binary hypothesis is defined as $\vec{x} = 0.5(1 - \hat{x}) : \vec{x} \in F_2^n$. The hard decision for each channel sample can be referred to as the channel sample bipolar hypothesis. The term 'hypothesis' is used to refer to both the channel frame hypothesis and channel sample hypothesis and the meaning depends on the context of usage.

The binary hypothesis at each variable node is broadcasted to all its neighboring check nodes. The check nodes performs a modulo-2 sum on the binary values from its neighborhood to check if its parity check equation is satisfied. It is important to note that the bipolar equivalent of the binary modulo-2 operation is multiplication and is deployed by this work and in the GDBF and NGDBF algorithms. The algorithms perform information exchange between the variable nodes and check nodes in the bipolar domain without conversion to the binary domain. The binary result of the modulo-2 sum is then broadcasted to all the variable nodes a certain check node is connected to. Based on the information from the neighboring check nodes, a variable node either flips the hypothesis initially taken or retains this hypothesis. If most of the incoming message bits into a variable node (from the connected check nodes) are different from the initial hypothesis, the hypothesis is flipped. In the bipolar domain the variable node does a summation of all the incoming values from the check node neighborhood. This cycle of passing information from the variable nodes to the check nodes and from the check nodes back to the variable nodes keeps repeating till all the parity check equations are satisfied or if a certain number of decoding iterations has been reached. This type of decoding is called Bit Flipping decoding and is a hardware efficient way of implementing decoding of LDPC codes.

Gallager introduced the BF algorithm for the BSC. The realm of BF decoding was left unexplored after its initial introduction until Y. Kou et. al. introduced the Weighed Bit-Flipping (WBF) algorithm [7], which assigns specific computed weights to the parity check equation results. These results are then used to compute energy (E_n) for each symbol being decoded which determines bit flipping. The symbol with the maximum (or minimum depending on the convention) E_n is flipped. Zhang et. al. introduced the modified Weighed Bit-Flipping (MWBF) [19] algorithm that adds the channel sample value scaled by an empirically optimized value α to the WBF energy function. Wu et. al. introduced the Parallel Weighed Bit-Flipping (PWBF) [20] algorithm which identifies the symbol with the maximum E_n (or minimum depending on the convention) within a subset of symbols associated with each parity check. The Dynamic Weighted Bit-Flipping (DWBF) [21] proposed by Chang et. al. and the Recursive WBF (RECWBF) [22] continued to modify the computed weight values for decoding performance gains.

2.9 PBF

Probabilistic Bit Flipping (PBF) introduces randomness to the original Bit Flipping decoding mechanism that was proposed. The decoding steps followed are similar to the original Bit Flipping decoding with the exception that the bits identified for flipping are flipped with some probability $p_0 < 1$. This results in only a random number of the original identified bits being flipped. The PBF technique was first proposed by Miladinovic et. al. in 2005 [23]. This technique helped improve decoding performance when compared to the original Bit Flipping decoding.

2.10 GDBF

Wadayama et. al. proposed a different approach to BF decoding. Based on the channel sample vector $\hat{y}$ received over a BAWGNC, the ML decoding problem is to find a codeword in $\tilde{C_b}$ that gives maximum correlation to the channel sample vector $\hat{y}$, where $\tilde{C_b}$ is the set of possible bipolar codewords for a specific LDPC code. This codeword is explicitly defined in Eqn. 2.3.

$$\mathbf{x}_{ml} = \arg\max_{\hat{x}\in\tilde{C_b}} \sum_{k=1}^{n} x_k y_k \tag{2.3}$$

where $\hat{x}$ is the bipolar hypothesis vector in each iteration of decoding and k specifies the index for each individual bit in the vector $\hat{x}$. The GDBF algorithm proposed by Wadayama et. al. adds information from the parity check matrix and converts the ML decoding problem into a gradient descent optimization problem. [10]. Information from the neighboring check nodes of an individual variable node are added as a penalty condition. The final inversion function proposed by Wadayama is defined in Eqn. 2.4. Hypothesis symbols having energy less than θ are considered for flipping. In multi-bit flipping, all the hypothesis vector symbols having E_k less than the flipping threshold are considered for flipping and in single-bit flipping, the hypothesis vector symbol computed based on a global function is considered for flipping.

$$E_k = x_k y_k + \sum_{i \in \mathcal{M}(k)} s_i \tag{2.4}$$

where $s_i = \prod_{j \in \mathcal{N}(i)} x_j$, for all $i \in \{1, 2, ...m\}$ and $\mathcal{M}(k)$ is the check node neighborhood of a symbol node.

2.11 PGDBF

The GDBF algorithm that was developed by Wadayama et. al. for the BAWGNC was modified by Rasheed et. al. to be applied on the Binary Symmetric Channel (BSC) [24]. Since the BSC does not contain any soft channel information, the inversion function proposed by Wadayama is computed as an Exclusive-OR between the channel sample and the binary hypothesis in addition to the syndrome information as indicated in 2.5

$$E_k = \vec{x}_k \oplus \vec{y}_k + \sum_{i \in \mathcal{M}(k)} s_i \tag{2.5}$$

The channel symbol with the highest energy value E_k is flipped. As there is no soft channel information on the BSC there can be multiple channel symbols with $E_k = \max(E_k)$. To avoid this problem Rasheed et. al. proposed the PGDBF which flips only the hypothesis of those channel samples which have a randomly assigned p_k less than p_0 similar the PBF algorithm. PGDBF offered the best decoding performance on the BSC compared to all the BF based decoders.

2.12 NGDBF

Sundararajan et. al. added perturbation noise a_k, which is a Gaussian distributed random variable with zero mean and variance $\sigma_n^2 = \eta^2\sigma^2$ to the GDBF decoding problem and added a weighing parameter w to the information from the parity check matrix to develop an energy function E_k for flipping bits in the hypothesis vector $\hat{x}$ as defined in Eqn. 2.6 [11]. η is the AWGN channel noise scale parameter.

$$E_k = x_k y_k + w \sum_{i \in M(k)} s_i + a_k \tag{2.6}$$

Similar to GDBF, in single-bit NGDBF (S-NGDBF), x_k with the lowest E_k is flipped. In the multi-bit NGDBF (M-NGDBF), a parameter θ is considered as the threshold for flipping bits in the decision vector $\hat{x}$. All the x_k bits with E_k less than θ are flipped. The optimal value of θ is obtained empirically.

NGDBF when combined with certain heuristics like output smoothing and threshold adaptation results in performance close to MS decoding with T = 10.

2.13 Redecoding for NGDBF

Tithi et. al. proposed a redecoding scheme for the original NGDBF algorithm, which targets redecoding the channel message frames that failed to converge on a codeword. The perturbation noise that is added to the failed frames during redecoding is independent of the perturbation noise that is added to the failed frames during initial decoding and therefore increases the likelihood of successful decoding. Redecoding yields performance very close to a benchmark Offset Min-Sum (OMS) decoder for the IEEE 802.3 standard LDPC code. [25] . However, the proposed re-decoding for NGDBF incurs a substantial latency penalty and is effective if the end application can accept a latency penalty at the expense of lower FER.

2.14 Trapping sets and Error Floors for NGDBF

As seen in Fig. 2.3, the BER decreases rapidly (this rapid decrease is also called the waterfall region of the performance curve), after which it reaches a saturation region called

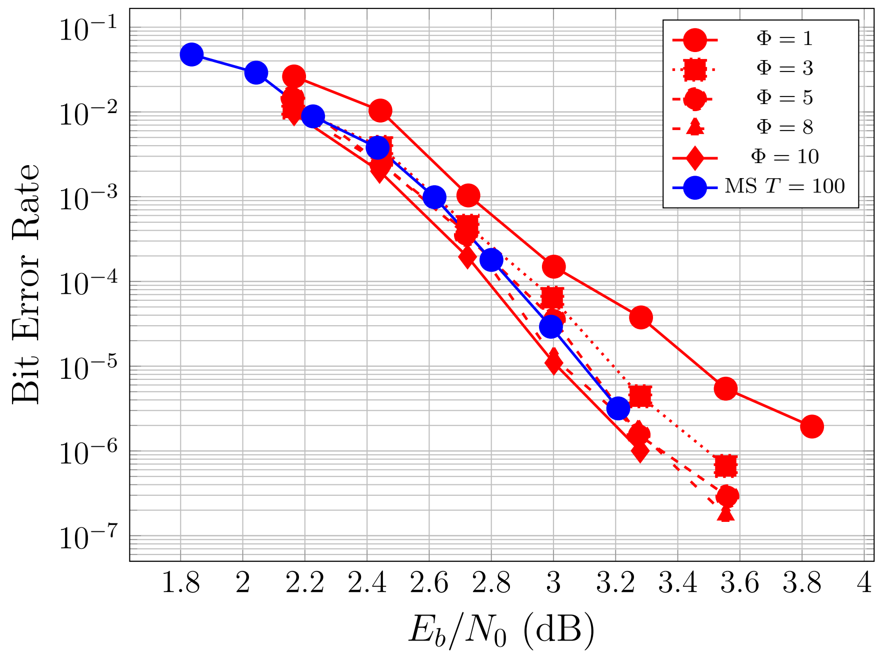


Fig. 2.2: Bit Error Rate for re-decoding with the SM-NGDBF on the 1/2PEGReg504x1008 code with different phases of redecoding Φ_s. Performance curve for the Min-Sum algorithm is provided for reference.

the error-floor region where the BER no longer decreases with an increase in the Signal-to-noise ratio (SNR). This error-floor region is the result of "*trapping sets*" which are sub graphs within the Tanner graph, that do not allow the decoder to converge on the correct codeword. Trapping sets were first explored by Richardson in his work "Error floors for LDPC codes" [26]. A related concept called "*absorbing sets*", which are a sub-type of trapping sets [27, 28], that lead to convergence on a non-codeword and usually prevents a decoding algorithm from any further decoding. In her doctoral dissertation, Tithi studied the effect of the dominant (8,8) absorbing set in the 802.3an 10GBASE-T LDPC code on the performance of NGDBF decoding [29].

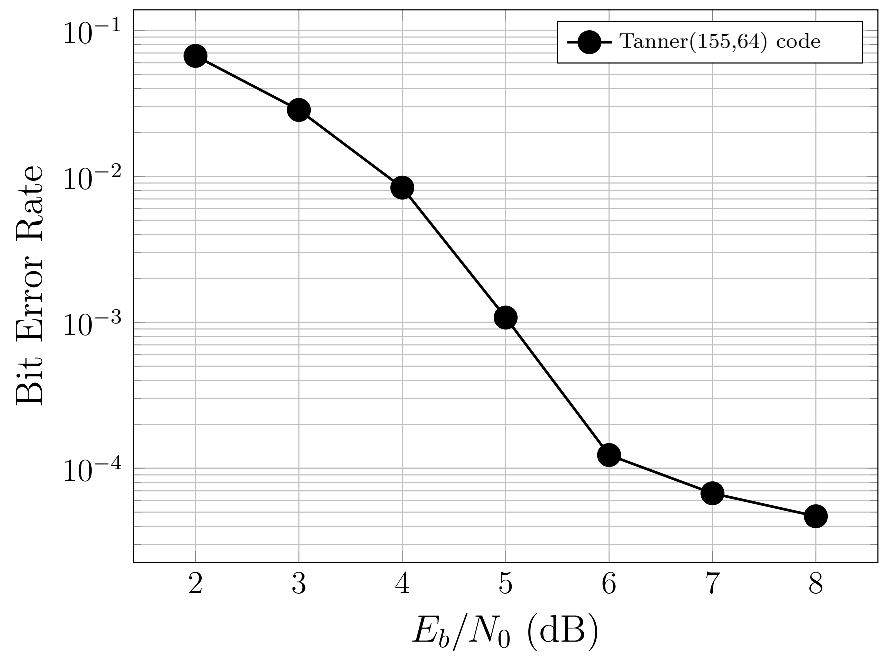


Fig. 2.3: Performance curve indicating error floor while decoding with the Tanner (155,64) code.

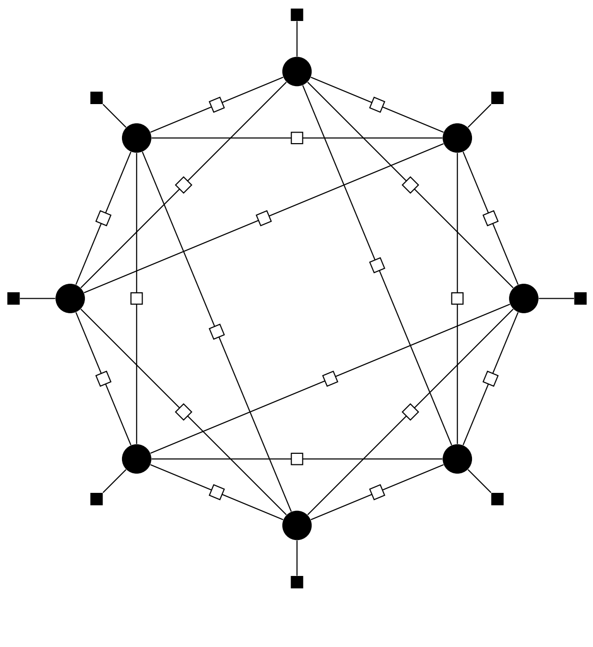


Fig. 2.4: The dominant (8, 8) absorbing set in the 802.3an 10GBASE-T LDPC code. The hollow squares represent the degree-2 check nodes, the filed square represent the degree-1 check nodes and the filled circles represent the variable nodes. Degree indicates the number of neighbors of a node.

CHAPTER 3
MOTIVATION

Significant research has been carried out in the area of BF decoding algorithms with low complexity LDPC hardware architectures. The NGDBF and GDBF decoding algorithms are significantly less complex to implement on an Field Programmable Gate Array (FPGA) or Application-specific integrated circuit (ASIC) when compared with the hardware implementation complexity of some of the other LDPC decoding algorithms including the Belief Propagation (BP) algorithm and the approximate Min-Sum (MS) algorithm.

At the fundamental level, BF in the NGDBF or GDBF decoding algorithm involves information about the received channel sample combined with its computed syndrome to progress through the different decoding iterations to potentially converge on a codeword. For the scope of this work, we can refer to the channel sample information and syndrome as the parameters that affect decoding. Based on our understanding of Eq. 2.6, the channel sample can be termed as y_k, the bipolar decision hypothesis for a specific channel sample is termed as x_k and the syndrome parameter is termed as p_k where $p_k = \sum_{i \in \mathcal{M}(k)} s_i$.

However, a statistically informative approach that quantifies the frequency of error occurrence for specific values of the parameters that affect the decoding performance was not followed while developing the GDBF and the NGDBF decoding algorithms. These algorithms have relied on heuristics for performance improvement and presently lack a theoretical framework. The motivation behind this work is to establish an algorithm that has a solid theoretical framework and is backed by informative statistics whilst using the same decoding mechanism as the GDBF and NGDBF algorithms. The outcome of this research work is the development of the PLMLBF algorithm which probabilistically decodes a sample channel message frame from prior estimated error statistics.

NGDBF uses perturbation noise to escape the local maxima that GDBF has a tendency to get stuck in. In works published till date, the added perturbation noise in the NGDBF decoding algorithm has been normally distributed (Gaussian). The selection of the Gaussian distribution has been based purely on heuristics. NGDBF could potentially benefit from other statistical distributions used as perturbation noise. Identification of these distributions requires informative error statistics that are developed as a part of this research work. While GDBF and NGDBF have developed a more heuristic approach to gradient descent optimization based BF, the PLMLBF is more theoretically motivated to improve BF performance.

CHAPTER 4
MULTI ITERATION PROBABILISTIC FLIPPING MATRIX CONSTRUCTION

4.1 Syndrome

For a specific channel sample being decoded in a specific decoding iteration, the syndrome parameter is equal to the sum of all the incoming messages from the neighboring check nodes. Positive syndrome values p_i indicate agreement with the hard decision x_i and negative p_i indicate that the hard decision x_i may be erroneous. For a specific LDPC code, d_v and d_c determine the number of edges connected to a variable node and check node respectively. The calculated syndrome value p_i in each iteration of decoding lies within $[-d_v, -d_v+2, ..., d_v-2, d_v]$.

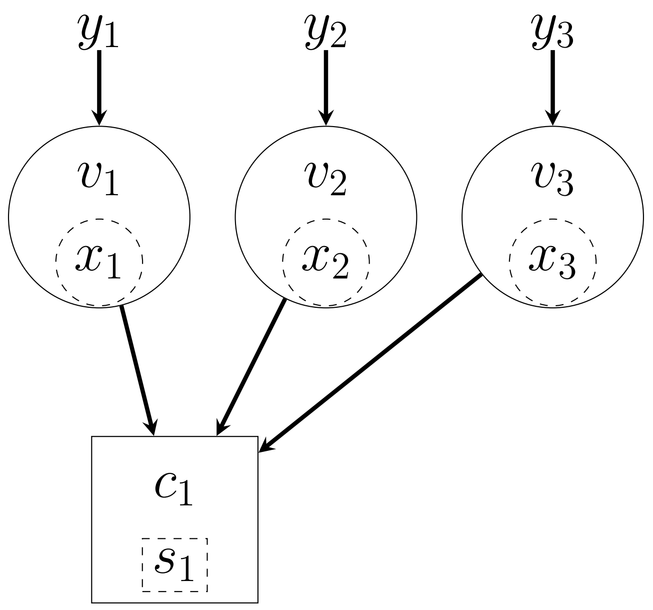


Fig. 4.1: $s_j = \prod_{i\in\mathcal{N}(j)} x_i$ where $\mathcal{N}(j)$ is the neighborhood of a certain check node.

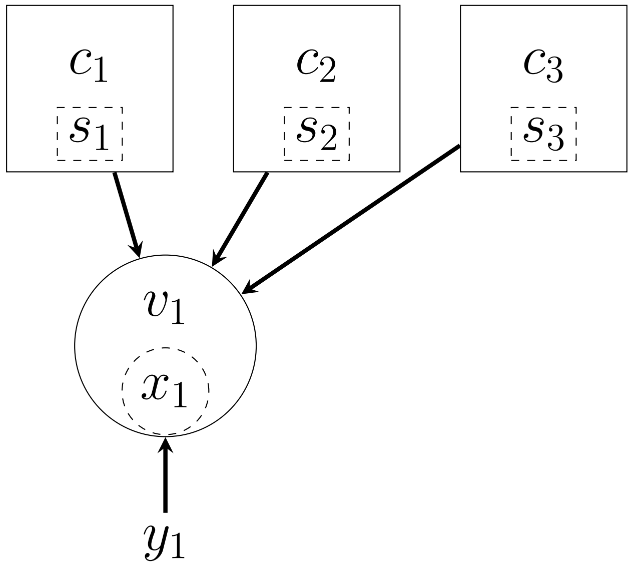


Fig. 4.2: Syndrome component $p_i = \sum_{j \in \mathcal{M}(i)} s_j$ where $\mathcal{M}(i)$ is the neighborhood of a specific variable node. The s_j value is computed as indicated in Fig. 4.1

The 1/2 PEGReg504x1008 LDPC code has been used for decoding in most of the simulations carried out as part of this research work. For the 1/2 PEGReg504x1008 LDPC code the value of d_v is 3. On account the possible values for the syndrome parameter p_i are -3,-1,1 and 3.

- **-3:** If all the incoming messages into a variable node are -1, the sum of the three incoming messages results in the syndrome parameter value being -3.

- **-1:** If two of the three incoming messages into a variable node are -1 and the third incoming message has a value of +1, the sum of the three incoming messages results in the syndrome parameter value being -1.

- **+1:** If two of the three incoming messages into a variable node are +1 and the third incoming message has a value of -1, the sum of the three incoming messages results in the syndrome parameter value being +1.

- **-3:** If all the incoming messages into a variable node are +1, the sum of the three incoming messages results in the syndrome parameter value being +3.

4.2 MIPFM Computation

The Multi-iteration Probability Flip Matrix (MIPFM) is a three dimensional matrix that is used as a flipping probability lookup table for a specific channel sample y_i with computed syndrome p_i that is being decoded in iteration ℓ. A Probability Flip Matrix (PFM) refers to the two dimensional matrix that is used as a flipping probability lookup table in a specific iteration of decoding. The three indices into the MIPFM are $x_i \times q_i$ (xq_i), p_i and ℓ. q_i is the channel sample value y_i after applying Q bits of quantization. The product of x_i and q_i is considered instead of y_i to index into the matrix, to avoid a bias to any specific codeword. This is in line with the inversion function for the GDBF and NGDBF algorithms.

In this research work, the total frame count used to establish informative error statistics free from small sample size anomalies is 200,000 channel message frames. Negligible impact on the PFM probability values is observed beyond 200,000 channel message frames. To generate the MIPFM, the below steps are followed. Alg. 1 provides an algorithmic representation of the MIPFM computation process.

Step 1: Depending on the LDPC code used for BF decoding, 200,000 frames of an all +1 n-symbol bipolar codeword is transmitted over a noisy communication channel. F denotes the transmission frame count. For the 1/2PEGReg504x1008 LDPC code, the size of a bipolar codeword would be 1008 code symbols and each codeword is denoted by $\hat{c}_o$ where $o \in \{1, ..., F\}$ and each code symbol in a specific frame $\hat{c}_o$ can be addressed as $\hat{c}_o(i)$.

Step 2: To mimic the transmission over a noisy channel with a specific SNR value, noise vector $\hat{z}_o$ is superimposed on $\hat{c}_o$ for all $o \in \{1, ..., F\}$. Noise samples $\hat{z}_o(i)$ are samples from a normal distribution with a standard deviation of σ. The value of σ is computed as determined in Sec. 2.7. The noise added codeword frames are the channel message frames and are denoted as $\hat{y}_o$ where $\hat{y}_o = \hat{c}_o + \hat{z}_o$ for all $o \in \{1, ..., F\}$. Each $\hat{y}_o(i)$ is clipped at $Y_{\max}$ such that $-Y_{\max} \leq \hat{y}_o(i) \leq Y_{\max}$.

Step 3: Uniform quantization is applied to each sample in $\hat{y}_o$ for all $o \in \{1, ..., F\}$ depending on the quantization levels expected in the PFM. For Q quantization bits, the total number of quantization levels N_Q is equal to 2^Q. The quantization function $Q(y)$ is denoted as:

$$Q(y) = \text{sign}(y) \left(\left\lfloor \frac{|y| N_Q}{2Y_{\max}} \right\rfloor + \frac{1}{2} \right) \left(\frac{2Y_{\max}}{N_Q} \right) \tag{4.1}$$

The quantized form of $\hat{y}$ is denoted as $\hat{q}$.

Step 4: Define λ as the set of all possible values for $\hat{q}(i)$ such that row r in the PFM has $\lambda(r)$ as its xq indexing value. The possible values of $\hat{q}(i)$ and $\hat{xq}(i)$ are the same because $\hat{x}(i)$ is a hypothesis and has values $\in \{+1, -1\}$.

Step 5: Define ω as the set of possible values for the syndrome such that column s in the PFM has $\omega(s)$ as its syndrome indexing value.

Step 6: An initial bipolar hypothesis vector $\hat{x}_o$ is computed based on the sign of the received channel samples in the vector $\hat{q}_o$ for all $o \in \{1, ..., F\}$.

Step 7: The value of the decoding iteration index ℓ is initialized to 0.

Step 8: The product of $\hat{x}_o(i)$ and $\hat{q}_o(i)$ is computed for all the individual channel samples $i \in \{1, ..., n\}$ for each channel frame $o \in \{1, ..., F\}$.

Step 9: Check node value $\hat{s}_o(j)$ is computed for all the check nodes $j \in \{1, ..., m\}$ for each channel frame $o \in \{1, ..., F\}$ where $\hat{s}_o(j) = \prod_{i \in \mathcal{N}(j)} \hat{x}_o(i)$.

Step 10: Syndrome component $\hat{p}_o(i)$ is computed for all the variable nodes $i \in \{1, ..., n\}$ for each channel frame $o \in \{1, ..., F\}$ where $\hat{p}_o(i) = \sum_{j \in \mathcal{M}(i)} \hat{s}_o(j)$.

Step 11: Create two dimensional matrices $\mathcal{H}_\ell$ and $\mathcal{E}_\ell$ of size $\lambda \times \omega$ and initialize all the elements of the respective matrix to 0.

Step 12: For each $r \in \{1, ...\lambda\}$ and each $s \in \{1, ...\omega\}$, $\mathcal{H}_\ell(r, s)$ is the count of the total channel samples in all the $o \in \{1, ...F\}$ channel message frames having an xq value equal to $\lambda(r)$ and syndrome equal to $\omega(s)$.

Step 13: For each $r \in \{1, ...\lambda\}$ and each $s \in \{1, ...\omega\}$, $\mathcal{E}_\ell(r, s)$ is the count of the total channel samples in all the $o \in \{1, ...F\}$ channel message frames having an xq value equal to $\lambda(r)$, syndrome equal to $\omega(s)$ and $x = -1$.

Step 14: For each $r \in \{1, ...\lambda\}$ and each $s \in \{1, ...\omega\}$, $\mathcal{P}_\ell(r, s)$ is equal to $\mathcal{E}_\ell(r, s)$ divided by $\mathcal{H}_\ell(r, s)$. This matrix is termed as the PFM for iteration ℓ.

Step 15: The hypothesis $\hat{x}_o(i)$ for each channel sample in each individual frame is flipped with a probability equal to $\mathcal{P}_\ell(\hat{xq}_o(i), p_o(i))$.

Step 16: ℓ is incremented by 1 and all the sequence of steps starting from **Step 8** are repeated till $\ell = T$

Once the MIPFM is constructed, it can be used as the probability lookup table for the PLMLBF algorithm. In each iteration of PLMLBF decoding channel samples q_i, with computed syndrome p_i can be probabilistically decoded over multiple iterations by indexing into the MIPFM using (xq, p, ℓ).

Algorithm 1 MIPFM Construction

```
    ▷ Parameter Definitions for MIPFM construction
1:  for $o \in \{1, ..., F\}$ do                                        ▷ F is the total frame count
2:      $\hat{c}_o(i) \leftarrow +1 \quad \forall\, i \in \{1, ..., n\}$
3:      $\hat{y}_o \leftarrow \hat{c}_o + \hat{z}_o$                                         ▷ $\hat{z}$ is the AWGN vector
4:      $\hat{q}_o \leftarrow Q(\hat{y}_o)$                                   ▷ $Q()$ is the Quantization function
5:      $\hat{x}_o \leftarrow \text{sign}(\hat{q}_o)$                                  ▷ $\hat{x}$ is the hypothesis vector
6:  Initialize $\ell \leftarrow 0$
7:  while $\ell < T$ do
    ▷ MIPFM Construction Steps
8:      for $o \in \{1, ..., F\}$ do
9:          $\hat{xq}_o(i) \leftarrow \hat{x}_o(i) \times \hat{q}_o(i) \quad \forall\, i \in \{1, ..., n\}$
10:         $\hat{s}_o(j) \leftarrow \prod_{i \in \mathcal{N}(j)} \hat{x}_o(i) \quad \forall\, j \in \{1, ..., m\}$        ▷ Check node update
11:         $\hat{p}_o(i) \leftarrow \sum_{j \in \mathcal{M}(i)} \hat{s}_o(j) \quad \forall\, i \in \{1, ..., n\}$       ▷ Variable node update
12:     $\mathcal{H}_\ell(r, s) \leftarrow 0 \quad \forall\, r \in \{1, ..., \lambda\} \quad \forall\, s \in \{1, ..., \omega\}$       ▷ Frequency matrix definition
13:     $\mathcal{E}_\ell(r, s) \leftarrow 0 \quad \forall\, r \in \{1, ..., \lambda\} \quad \forall\, s \in \{1, ..., \omega\}$ ▷ Error frequency matrix definition
14:     for $o \in \{1, ..., F\}$ do
15:         Increment $\mathcal{H}_\ell(\hat{xq}_o(i), \hat{p}_o(i))$ by 1 $\quad \forall\, i \in \{1, ..., n\}$
16:         if $\hat{x}_o(i) == -1$ then Increment $\mathcal{E}_\ell(\hat{xq}_o(i), \hat{p}_o(i))$ by 1 $\quad \forall\, i \in \{1, ..., n\}$
17:     $\mathcal{P}_\ell(r, s) \leftarrow \mathcal{E}_\ell(r, s)\ /\ \mathcal{H}_\ell(r, s) \quad \forall\, r \in \{1, ..., \lambda\} \quad \forall\, s \in \{1, ..., \omega\}$   ▷ Compute PFM
    ▷ MIPFM Application (Decoding) Steps
18:     for $o \in \{1, ..., F\}$ do
19:         Flip $\hat{x}_o(i)$ with probability $\mathcal{P}_\ell(\hat{xq}_o(i), \hat{p}_o(i)) \quad \forall\, i \in \{1, ..., n\}$
20:     $\ell \leftarrow \ell + 1$
```

CHAPTER 5
SIMULATION RESULTS

MIPFM's are constructed prior to the start of PLMLBF decoding to obtain simulation results. The simulation results are obtained after decoding 2000 channel message frames. The 200,000 channel message frames that are used to construct the MIPFM are independent of the 2000 channel message frames used to obtain the simulation results. Unless otherwise stated, the MIPFM used for decoding is constructed using the exact parameter values as the 2000 channel message frames that are being decoded to obtain the simulation results. These parameters include $Y_{\max},\ T,\ Q,$ SNR and the LDPC code used for decoding. The channel sample saturation value is denoted as $Y_{\max}$ and Q is the bits of quantization applied on the channel samples. For a BER performance plot indicating PLMLBF decoding performance over 4 different SNR values would have 4 respective MIPFM's constructed for each of those SNR values.

Performance curves for the PLMLBF algorithm are obtained, primarily using the rate 1/2 PEGReg504x1008 code on the BAWGNC using binary antipodal modulation. The 1/2 PEGReg504x1008 code represents the Regular Progressive Edge Growth code. (d_v, d_c) for this code is (3,6) and regular indicates that the (d_v, d_c) for each check node and variable node respectively is the same throughout the code construction. Although the actual d_c has some irregularities in the code construction in MacKay's encyclopedia, the code is still considered as a regular code. Additional simulations were performed using the 1/2 4000.2000.4.244 code and the 0.9356 4376.282.4.9598 code to demonstrate the effectiveness of the PLMLBF algorithm. All the three LDPC code constructions are from MacKay's encyclopedia [30]. Comparison results are provided for the GDBF, NGDBF, BP algorithm with 250 iterations and for the MS algorithm with 5, 10 and 100 iterations respectively. Unless otherwise stated, the following values are considered for each of the parameters that are crucial to the decoding process. The channel sample values are saturated at 2.5, quantized using 8-bit quantization and decoded over 100 iterations ($Y_{\max} = 2.5$, $Q = 8$ and $T = 100$). The total number of frames decoded simultaneously are 2000, and the decoding is stopped when a total of 100 frame errors are detected.

5.1 BER Performance

BER performance curves, usually have an initial waterfall region which exhibits an improvement in BER performance with an increase in SNR followed by an error floor region, where there is very little improvement in BER performance with an increase in SNR. Instead of having the standard waterfall region, from Fig. 5.1 it can be observed that the PLMBF algorithm performance curve is almost straight up to an SNR of 3.25 dB, after which it exhibits the standard waterfall region.

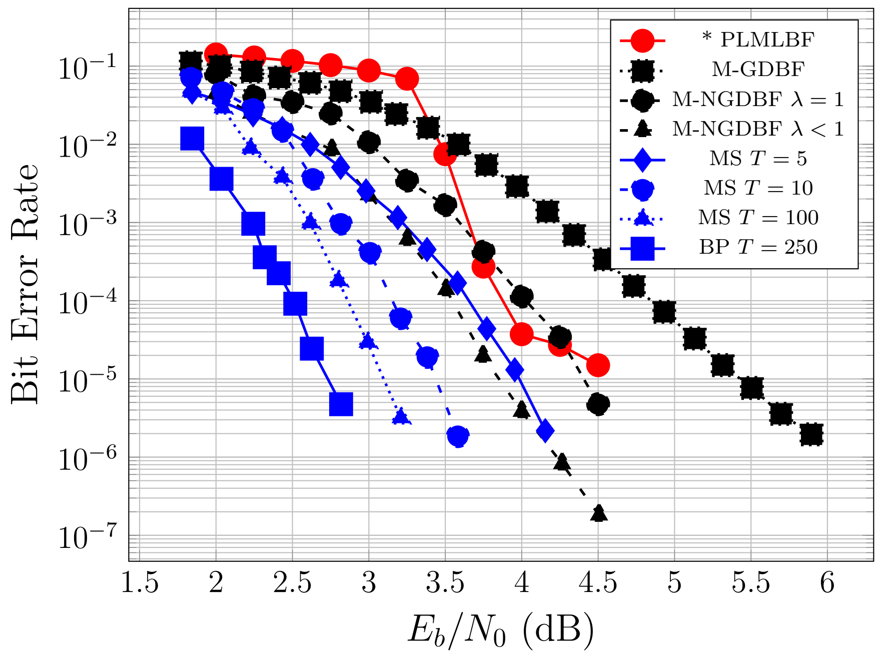


Fig. 5.1: Bit Error Rate versus E_b/N_0 performance curve for the PLMLBF algorithm with T = 100, Q = 8 and separate PFM construction for each iteration l $(1 \leq l \leq T)$ in the MIPFM using the rate 1/2 PEGReg504x1008 LDPC code simulated over a BAWGNC with binary antipodal modulation. Performance curves for several other algorithms are provided for comparison.

Fig. 5.2 is useful to understand why PLMLBF decoding for SNR values below 3.25 dB is sub-optimal. In the initial iteration of the MIPFM construction, for a certain xq_i value for a channel sample q_i, the probability matrix is effective only if the count of the erroneous xq_i samples in the message frames being decoded are not a significant portion of the total xq_i used to construct the MIPFM. At lower SNR's, this condition is however not met as observed in Fig. 5.2. The error at lower SNR's are too high for the error statistics to be informative. On account when the initial PFM in the MIPFM is used for the decoding stage of MIPFM construction, a lot of non-erroneous channel samples are flipped erroneously leading to high error flipping probabilities. As the iterations proceed, the matrix construction does not recover from these initial erroneous flips and continues erroneous flipping in the later iterations of MIPFM construction. This on account leads to poor BER performance when these MIPFM's are used for simulations. In Fig. 5.2, it can be observed that for an SNR of 1 dB, for a specific value of q_i, the probability density of $\mathrm{P}(-q_i)$ (indicating error probability) is significant compared to $(\mathrm{P}(-q_i)+\mathrm{P}(q_i))$. For an SNR of 4 dB, $\mathrm{P}(-q_i)$ is a very small portion of $(\mathrm{P}(-q_i)+\mathrm{P}(q_i))$ leading to very little erroneous flipping.

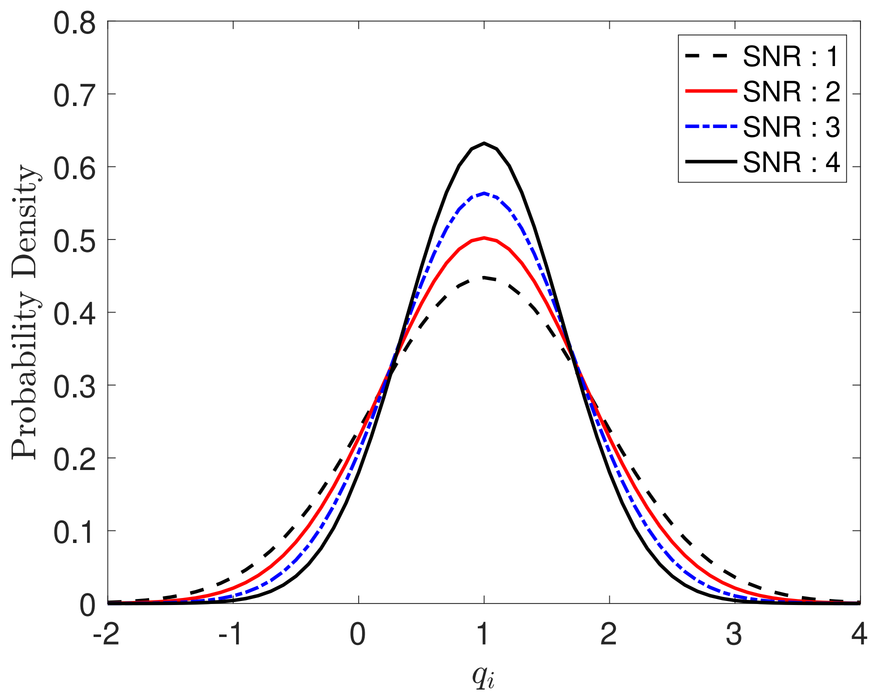


Fig. 5.2: Comparison of the probability density for a bipolar +1 being transmitted over BAWGNC of different SNR values. The plot is meant to be a representative for quantized channel sample values for the purpose of explanation. The hypothesis $x_i = -1$ for channel sample values of $q_i < 0$. This hypothesis is considered erroneous and is a potential candidate for flipping.

From an SNR of 3.25 dB to an SNR of 4 dB, the performance curve for the PLMLBF algorithm exhibits the waterfall curve. The PLMLBF algorithm provides performance gains over the M-NGDBF algorithm when implemented without heuristics at SNR values of 3.75 dB and SNR values of 4 dB and is found to be most effective when working with communication channels exhibiting an SNR close to 4 dB.

After an SNR of 4 dB, trapping sets have a significant impact on the decoding performance. Trapping sets do not allow convergence to a code word which affects the decoding performance. The constructed MIPFM is developed empirically and on account does not have any additional heuristic to deal with the trapping sets. The performance curve enters into the error floor region after an SNR of 4 dB on account of trapping sets. NGDBF adds the perturbation noise to escape trapping sets and on account exhibits better decoding performance for SNR's over 4 dB.

To escape trapping set behavior, a lower SNR MIPFM is used to decode channel message frames transmitted over a communication channel with a higher SNR. As represented in Fig. 5.3, decoding channel message frames transmitted over a BAWGNC with an SNR of 4.25 dB and an SNR of 4.5 dB with an MIPFM constructed for an SNR of 4 dB lowers the error floor for the 1/2PEGReg504x1008 code. The PLMLBF outperforms the M-NGDBF algorithm (implemented without heuristics) at an SNR of 4.25 dB and matches the M-NGDBF algorithm (implemented without heuristics) at an SNR of 4.5 dB. Decoding with a lower SNR MIPFM provides additional probability of flipping which helps escape some of the trapping sets and on account, provides improved BER performance.

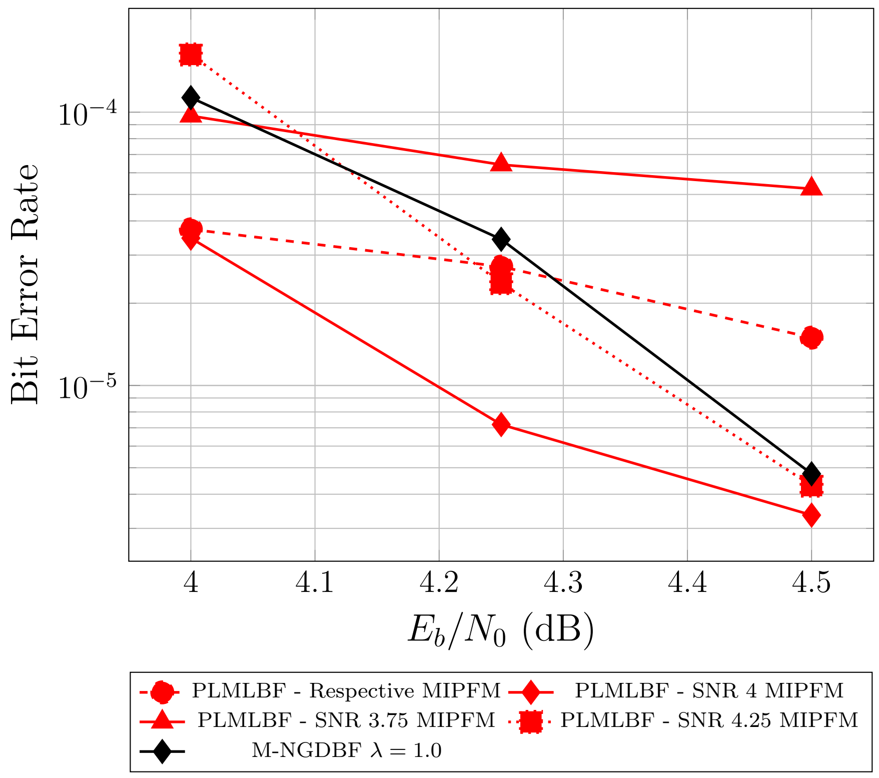


Fig. 5.3: PLMLBF BER performance curve indicating decoding specific SNR channel samples with an MIPFM constructed using a different SNR for the 1/2PEGReg504x1008 LDPC code with T = 100, Q = 8.

The PLMLBF algorithm outperforms the M-GDBF in Fig. 5.4 and Fig. 5.5 indicating comparable performance on codes with higher variable-node degree (in the case of Fig. 5.4) and for codes with rate above 0.9 (in the case of Fig. 5.5)

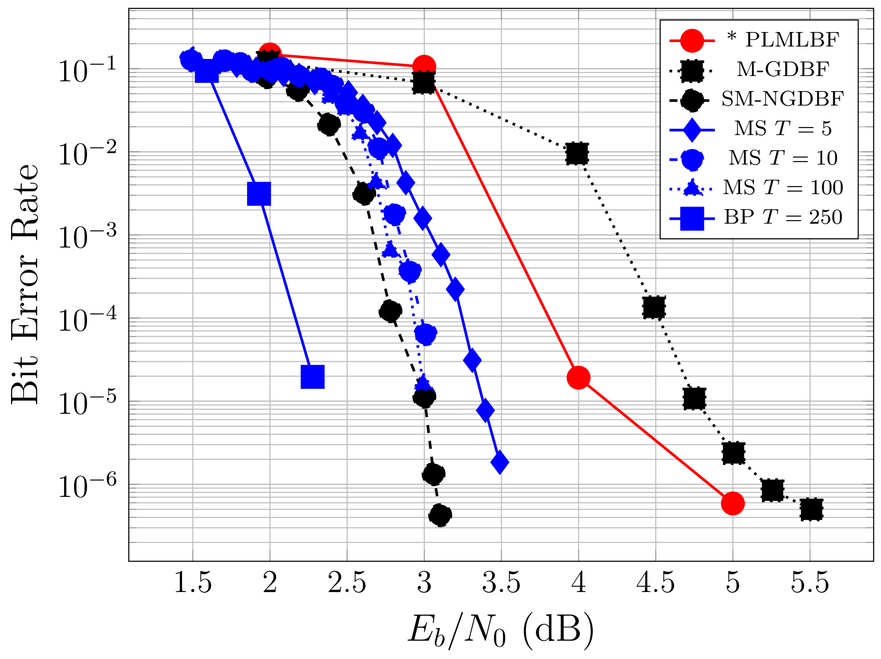


Fig. 5.4: Bit Error versus E_b/N_0 performance curve for the PLMLBF algorithm with T = 100, Q = 8 and separate PFM construction for each l ($1 \leq l \leq T$) in the MIPFM using the rate 1/2 4000.2000.4.244 LDPC code simulated over a BAWGNC with binary antipodal modulation. Performance curves for several other algorithms are provided for comparison.

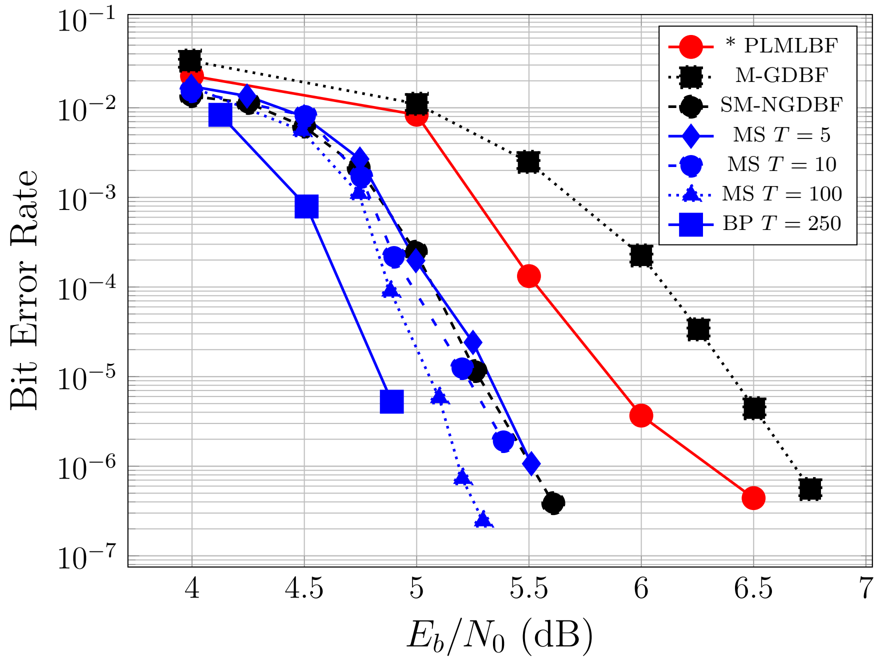


Fig. 5.5: Bit Error versus E_b/N_0 performance curve for the PLMLBF algorithm with T = 100, Q = 8 and separate PFM construction for each l $(1 \leq l \leq T)$ in the MIPFM using the rate 0.9356 4376.282.4.9598 LDPC code simulated over a BAWGNC with binary antipodal modulation. Performance curves for several other algorithms are provided for comparison.

5.2 Impact of variation of Quantization Bits (Q)

Based on the observations in Fig. 5.6 the PLMLBF algorithm is not effective for SNR's lower than 3.5 dB, even if the quantization bits are altered. A lower number of quantization bits indicates a higher loss of information, while lowering the hardware complexity. For an SNR of 3.5 dB, it is observed that having additional quantization information beyond 6 bits of quantization does not provide any significant BER performance. Likewise, for SNR of 4 dB and SNR of 4.5 dB increasing the quantization bits beyond 4 does not provide any gain in BER performance. For SNR of 4.5 dB, it is observed that increasing the quantization bits beyond 4 bits actually worsens the BER performance. A possible explanation of the same would be that at higher SNR's, the channel sample errors are being spread across too many bins in the PFM with a higher error probability accuracy in some of the bins and a lower error accuracy in the others leading to some erroneous flipping.

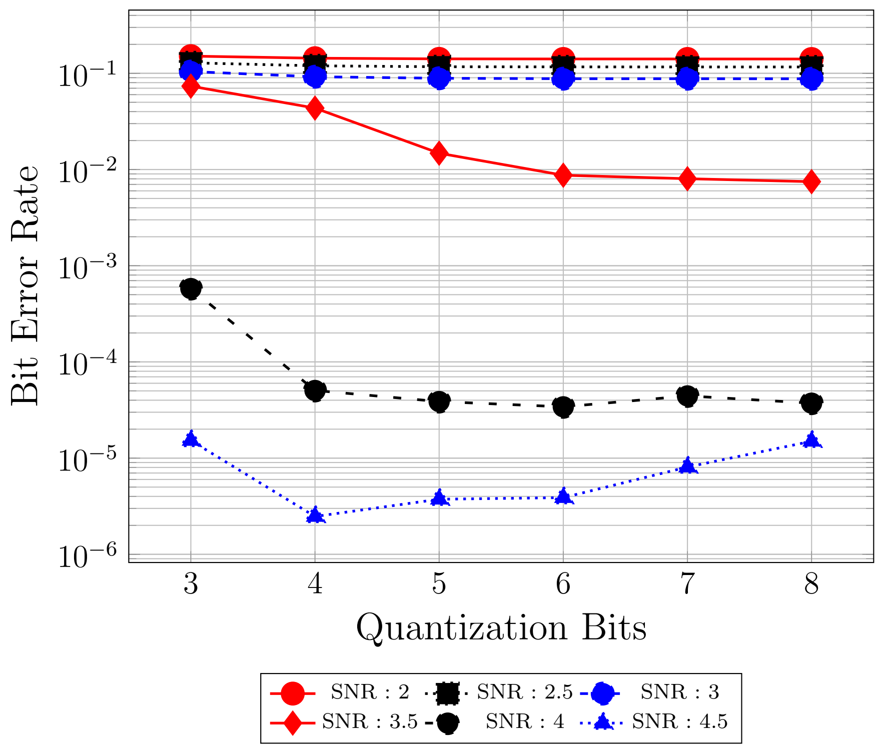


Fig. 5.6: BER performance curves indicating the impact of quantization bits (Q) on PLMLBF decoding with T = 100 and separate PFM for each l ($1 \leq l \leq T$) in the MIPFM simulated over the BAWGNC with binary antipodal modulation using the 1/2PE-GReg504x1008 LDPC code.

At an SNR of 4 dB, as the quantization bits are increased, there is an improvement in the PLMLBF FER performance. Beyond an SNR of 4 dB, consistent with the BER performance curves for quantization bits variation in PLMLBF, an increase in quantization bits beyond 6 bits actually worsens the performance. 6 bit quantization provides the best FER performance at an SNR of 4.25 dB.

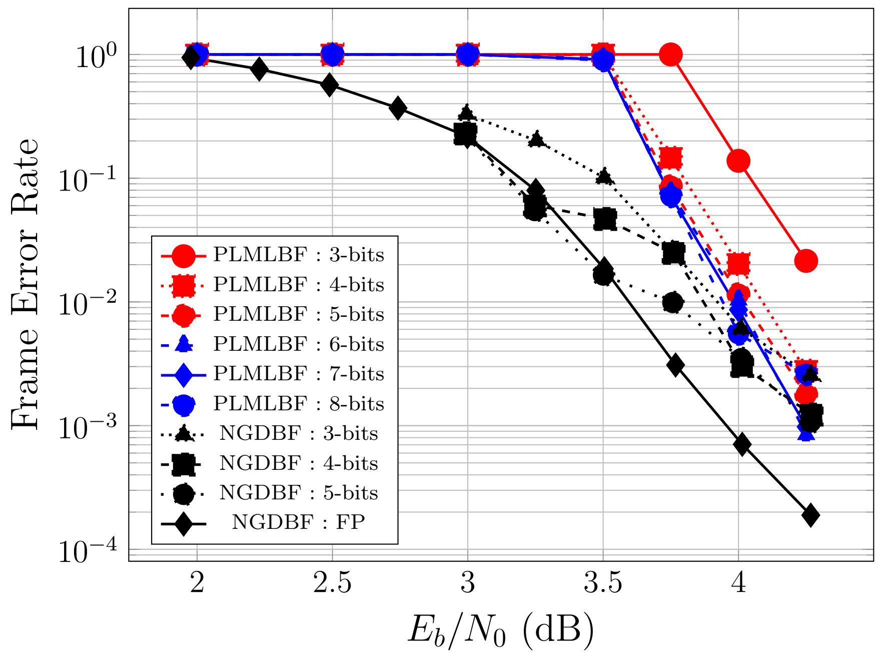


Fig. 5.7: FER performance curves indicating the impact of quantization bits (Q) on PLMLBF decoding simulated over the BAWGNC with binary antipodal modulation using the 1/2PEGReg504x1008 LDPC code. The MIPFM is constructed for T = 100 and separate PFM are constructed for each l ($1 \leq l \leq T$) in the MIPFM. Impact of quantization on the FER performance of NGDBF is provided for reference.

5.3 Impact of variation of Decoding Iterations (T)

Similar to the findings of the previous section, for SNR values up to 3 dB, the algorithm performs in a sub-optimal manner and changing the number of decoding iterations does not impact BER. From Fig. 5.8, it can be observed that for an SNR of 3.5 dB, an increase in the number of decoding iteration leads to an improvement in the decoding performance. For an SNR of 4 dB, an increase in the number of decoding iterations beyond 125 iterations does not yield any additional performance gains and for an SNR of 4.5 dB, an increase in decoding iterations beyond 50 iterations does not give any significant improvement in decoding performance.

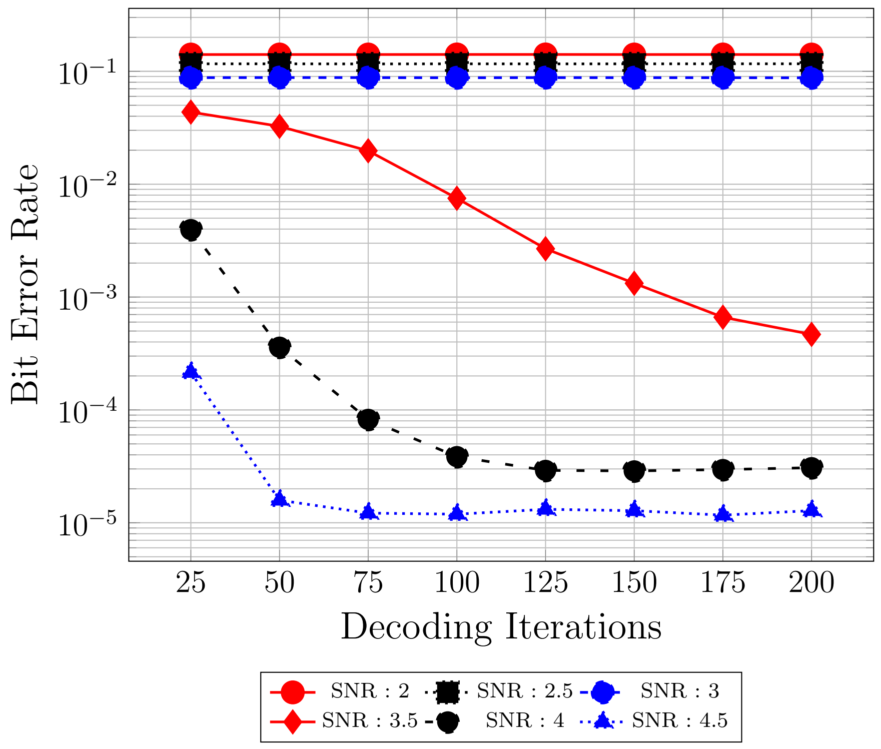


Fig. 5.8: PLMLBF BER performance curve simulated over a BAWGNC with antipodal modulation using the 1/2PEGReg504x1008 LDPC code. The respective MIPFM used for decoding is constructed using Q = 8 for T decoding iterations with separate PFM construction for each l $(1 \leq l \leq T)$.

Early stopping, introduced in the early stopping Adaptive Threshold GDBF (ES-AT-GDBF) algorithm is a technique that significantly reduces the number of bit flipping decoding iterations [31]. Decoding a channel message frame is stopped before it completes the pre-determined number of decoding iterations if all the parity check equations are satisfied. For SNR values greater then 3.5 dB it can be observed in Fig. 5.9 that the average number of decoding iterations required for the PLMLBF algorithm to converge onto a codeword decreases significantly. For SNR values greater than 4 dB, the PLMLBF curve almost aligns with the curves for the M-GDBF and M-NGDBF algorithms.

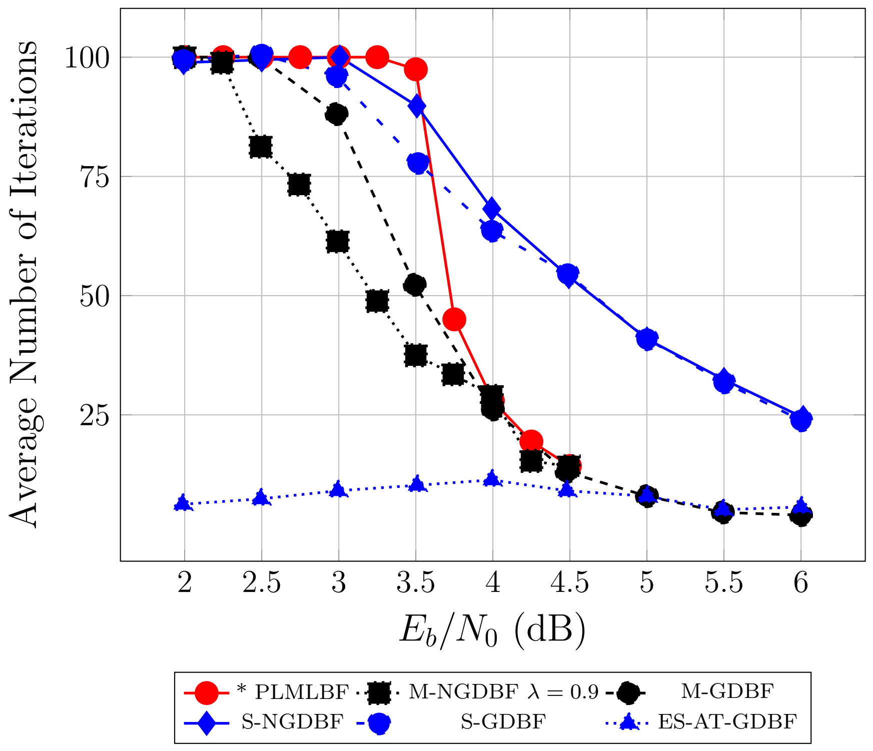


Fig. 5.9: Average number of convergence iterations versus Eb/N0 curve for the PLMLBF algorithm simulated over a BAWGNC with antipodal modulation using the 1/2PE-GReg504x1008 code and Q = 8. Average convergence iteration curves for several other algorithms are provided for comparison.

5.4 PLMLBF decoding with redecoding

The redecoding technique proposed by Tithi et. al. for NGDBF was applied to the PLMLBF decoding algorithm and the performance plot for the same is represented by Fig. 5.10. When probabilistically flipping the hypothesis for a channel sample a random value from the uniformly distribution [0 to 1] is generated and compared with the probability value from the PFM. If the probability from the PFM is greater than the random probability value, the hypothesis is flipped. Redecoding is effective for the PLMLBF algorithm, because the uniformly distributed random values in the range of [0 to 1], that are used for comparison with the actual flipping probabilities obtained from a PFM are independent in each phase of decoding. Beyond redecoding phase 5, there could be a possible increase in the BER performance curve with additional phases of redecoding. However, software simulations might take months to reach the target frame errors to stop decoding and requires that the implementation of the PLMLBF algorithm be moved to an FPGA/ASIC based implementation. FPGA/ASIC implementation (alternatively hardware implementation) however, is not a part of this research work. From Fig. 5.10, it can be observed that redecoding when implemented with T = 100 provides improved performance over lower iterations of decoding. Increasing T beyond 100, does not yield any significant performance improvement for all the redecoding phases.

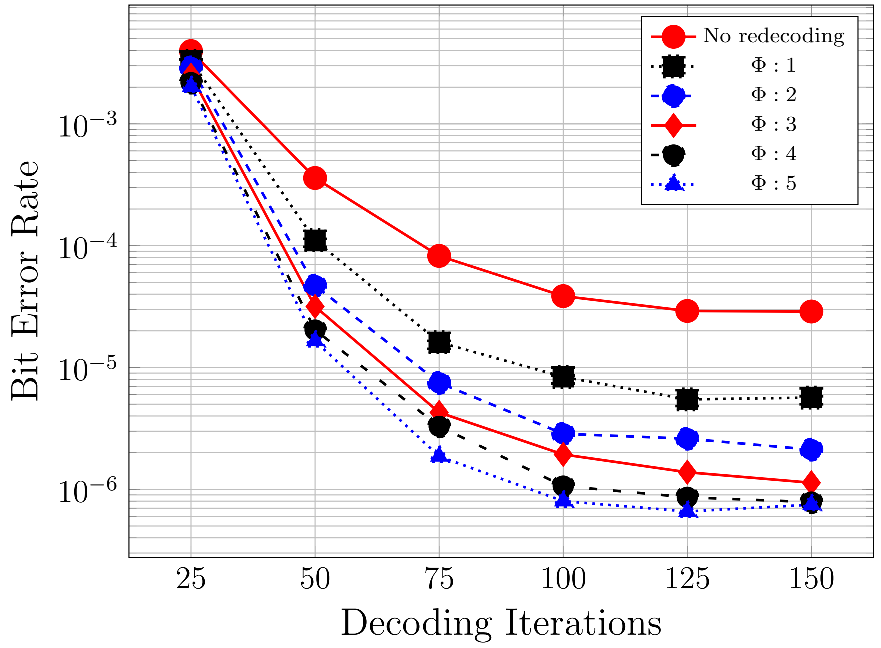


Fig. 5.10: Impact of redecoding phases Φ on the PLMLBF decoding performance for the 1/2 PEGReg504x1008 LDPC code simulated over a BAWGNC with an SNR of 4 dB and antipodal modulation.

5.5 Impact of repeated PFM in the MIPFM on PLMLBF performance

Instead of constructing a separate PFM for each iteration of decoding when constructing the MIPFM, this subsection presents the findings of the impact of repeating a PFM over multiple decoding iterations and then constructing a new PFM. The resultant MIPFM now has repeating PFM's for consecutive iterations before a new PFM is generated empirically for consecutive iterations. Based on Fig. 5.11, it can be observed that using just 13 PFM's, gives the best performance for all SNR values. The notation [1,2,3,10:10:100] implies that PFM 1,2,3 are used, PFM 3 is repeated for iterations 3 through 10, PFM 10 is repeated for iterations 10 through 19, PFM 20 is repeated through iterations 20 through 29 and so forth.

Using a lesser number of PFM's for PLMLBF decoding provides a hardware benefit to implement the algorithm when compared to the standard PLMLBF algorithm implementation without PFM repetition. Lesser registers would be required to store the probability values, when PLMLBF is implemented with PFM repetition on a hardware platform.

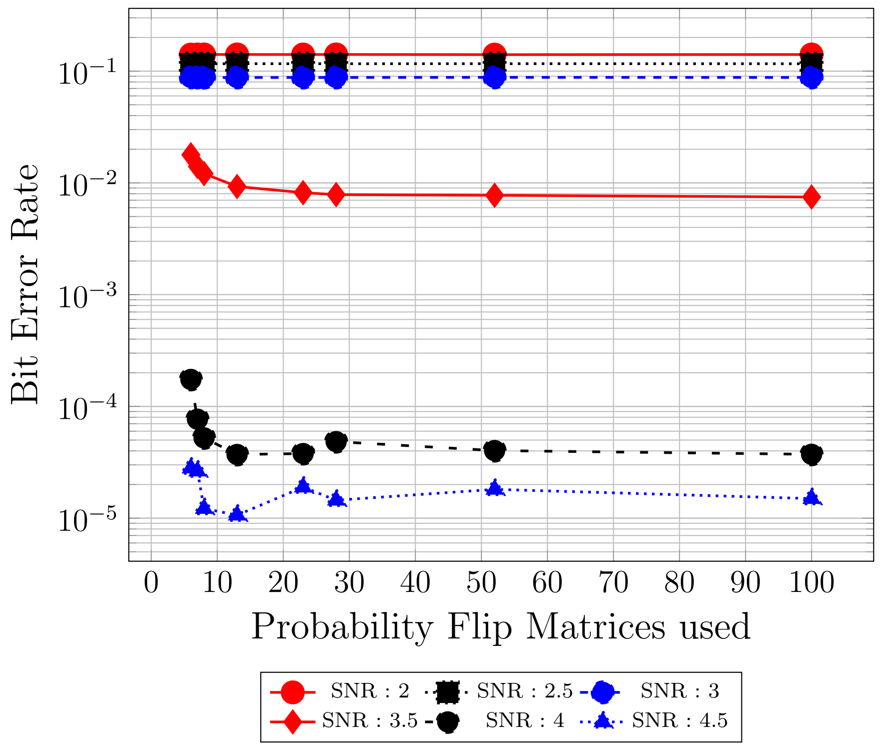
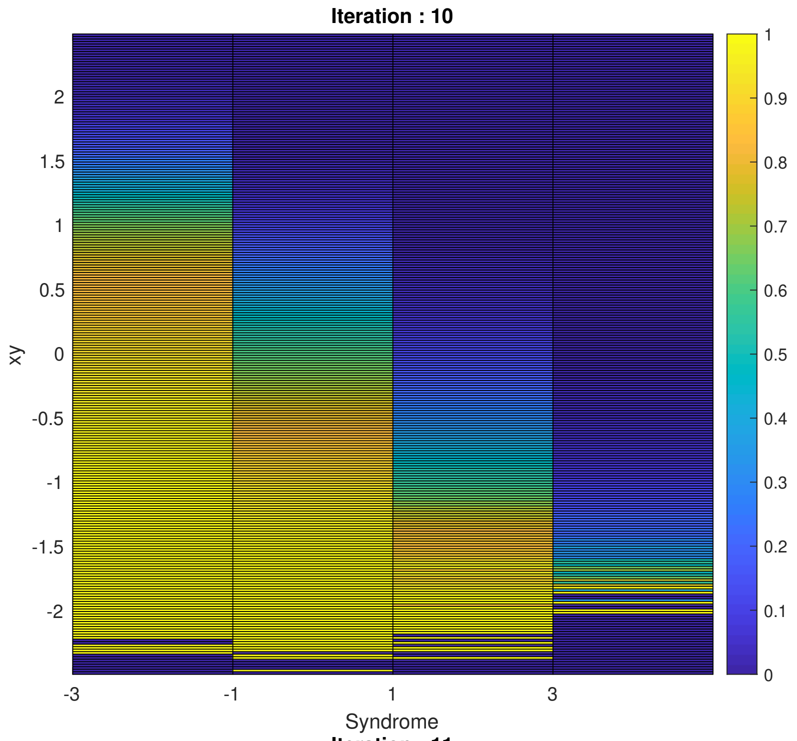


Fig. 5.11: BER performance curve indicating the impact of repeated PFM on the performance of the PLMLBF decoding algorithm for the 1/2PEGReg504x1008 code simulated over the BAWGNC with antipodal modulation. The data points are representative of repetition for the following PFM.

6 PFM : [1,2,3,33,66,100] 7 PFM : [1,2,3,25:25:100] 8 PFM : [1,2,3,20:20:100]

13 PFM : [1,2,3,10:10:100] 23 PFM : [1,2,3,5:5:100] 28 PFM : [1,2,3,4:4:100]

52 PFM : [1,2,3,2:2:100] 100 PFM : [1:1:100]



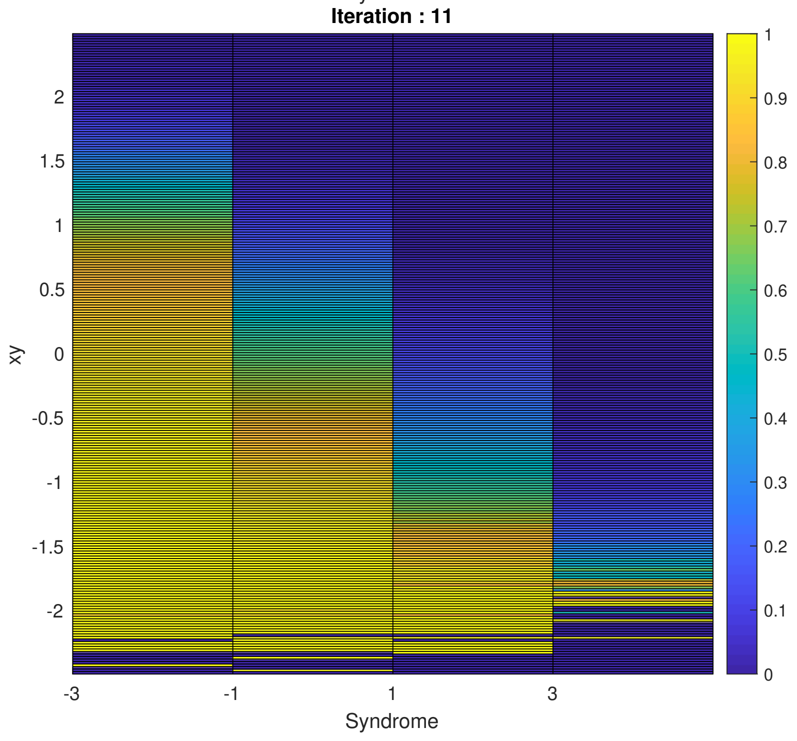


Fig. 5.12: PFM for consecutive iterations 10 and 11 of the MIPFM construction for the 1/2 PEGReg504x1008 LDPC code simulated over a BAWGNC with an SNR of 4 dB and binary antipodal modulation. PFM 11 can be replaced by PFM 10 in the repeated PFM scheme for PLMLBF decoding because of the similar flipping probability values.

5.6 Output decision smoothing and Output decision 2 out of 3 voting

The probabilistic nature of PLMLBF may lead to erroneous flipping in a certain code symbol. Output smoothing is a heuristic that is used to smooth out erroneous flips and enable convergence on the correct word. For every code symbol that is being decoded in a channel message frame, a running counter is maintained. In each decoding iteration the counter increments for the decoded value being +1 and decrements for the decoded value being -1. At the end of T iterations of decoding, the sign of the counters value determines the value for a specific code symbol. In effect, the decoding decision is delayed till the last iteration of decoding. The counter is initialized to 0 and the smoothing technique is started only after the average number of decoding iterations required for codeword convergence at a specific SNR have been completed. In any iteration of decoding if the early stopping condition has been satisfied, the decoding is stopped. Smoothing when applied to PLMLBF is denoted as SM-PLMLBF within the scope of this work. SM-PLMLBF is implemented over 300 iterations of decoding. A 300 iteration MIPFM is created by repeating each PFM in a 100 iteration MIPFM 3 times consecutively for decoding.

Similar to the smoothing heuristic, output 2 out of 3 voting is a heuristic that is used to smooth out erroneous flips and enable convergence on the correct word. A channel message sample is decoded thrice using the same PFM before voting on the decoded value for a certain iteration. On account for a 100 PFM MIPFM, we require 300 iterations of decoding. This heuristic is termed as VO-PLMLBF within the scope of this work.

Both SM-PLMLBF and VO-PLMLBF come within 0.25 dB of the SM-NGDBF algorithm and outperforms the H-GDBF and the MS algorithm with T=5 for SNR values above 3.5 dB.

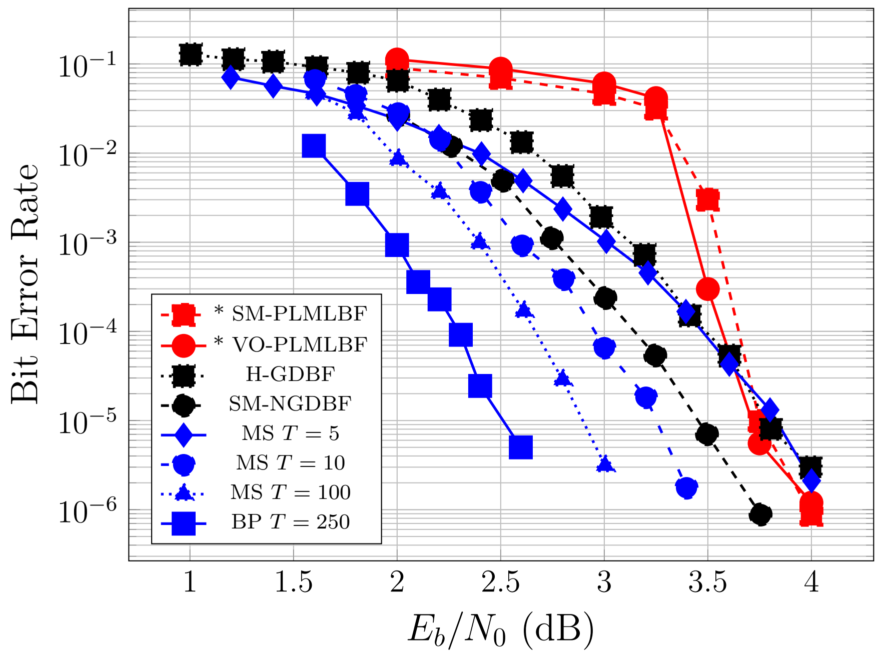


Fig. 5.13: Bit Error Rate versus E_b/N_0 performance curve for the SM-PLMLBF and VO-PLMLBF algorithms with T = 300, Q = 8 and early stopping for the rate 1/2 PEGReg504x1008 LDPC code simulated over a BAWGNC with binary antipodal modulation. Performance curves for several other algorithms are provided for comparison.

5.7 PLMLBF with NGDBF inversion function

This research work defines an explicit MIPFM construction. This section explores the impact of populating the probabilities in the MIPFM based on the inversion function defined in the NGDBF algorithm. For a certain quantized channel sample q_k with a specific syndrome, NGDBF defines the probability of flipping the hypothesis x_k as $P_F(x_k) = Pr(\bar{E}_k + a_k < \theta)$. The value of $\bar{E}_k$ is fixed for a certain quantized channel sample with a specific syndrome and is defined by $x_k q_k + w \sum_{i \in \mathcal{M}(k)} s_i$. The Gaussian perturbation noise a_k results in the probabilistic flipping value $P_F(x_k)$.

In the first iteration (initial PFM) of MIPFM construction, the probabilities are populated based on the NGDBF flipping probabilities. In each subsequent iteration of the MIPFM construction, the probabilities within each bin of a PFM is tuned empirically based on a biasing function. For a specific probability $P_F(x)$ within a certain bin, the biasing function is defined in eq. 5.1.

$$P_F(x_{h+1}) = \frac{N_h}{N_h+1} P_F(x_h) + \frac{e_{h+1}}{N_h+1} \tag{5.1}$$

In each iteration of MIPFM construction as h new samples are obtained for a specific bin within the PFM, the probability in the bin is tuned every time a new sample is obtained for that bin. The biased sample count N_h is the sum of the sample count h and a biasing value B. The value of B was computed empirically and found to be 10. For a specific bin, the hypothesis x_{h+1} of the $(h+1)^{th}$ sample being erroneous is defined as e_{h+1} with $e \in \{0,1\}$ and 1 indicating an erroneous sample.

For each SNR, the below table specifies the value of θ and η in the NGDBF inversion function used to construct the initial iteration of the MIPFM. These values were calculated through an empirical search. The value of w was set as 0.75.

SNR	θ	η
2	-0.3	0.8
2.25	-0.3	0.85
2.5	-0.4	0.95
2.75	-0.4	1
3	-0.35	1
3.25	-0.3	0.9
3.5	-0.2	0.9
3.75	-0.3	0.9
4	-0.3	0.9
4.25	-0.45	1
4.5	-0.2	0.75

Table 5.1: NGDBF θ and η values for MIPFM construction at each SNR

The MIPFM constructed based on the NGDBF inversion function is used in PLMLBF decoding. From Fig. 5.14 it is evident that beyond an SNR of 4 dB, the PLMLBF clearly outperforms the M-NGDBF algorithm when used without local heuristics and comes in close proximity to the M-NGDBF algorithm when used with the threshold adaptation local heuristic. Up to an SNR of 3.75 dB, the PLMLBF algorithm when used with a biased NGDBF based MIPFM construction performs sub-optimally. Avoiding the biasing function and using the first iteration of the MIPFM for all iterations of PLMLBF decoding will effectively solve the sub-optimality at lower SNR's. At lower SNR's, using this technique will enable PLMLBF to match the NGDBF performance curve when implemented without any local threshold adaptation heuristic.

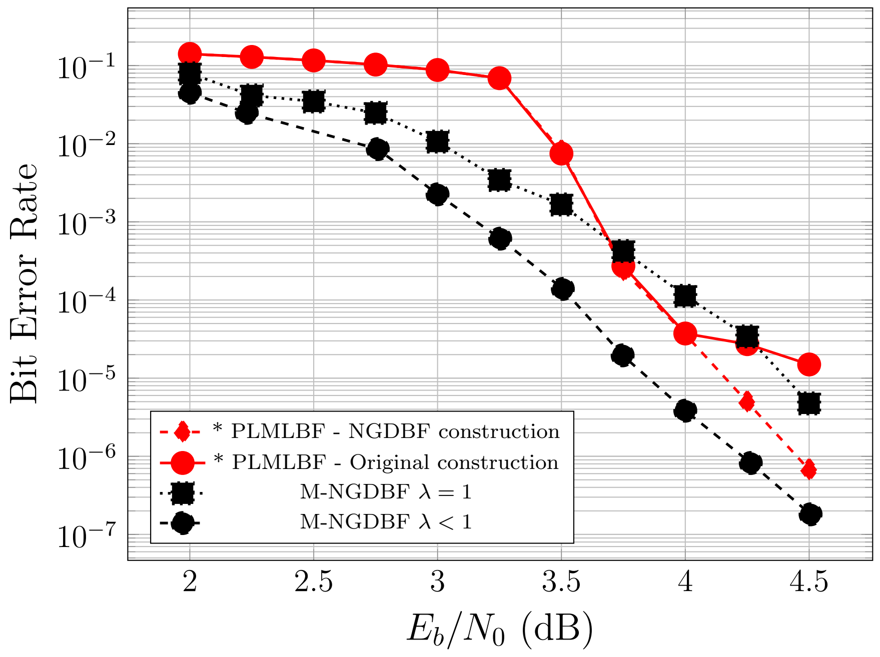


Fig. 5.14: Bit Error Rate versus E_b/N_0 performance curve for the PLMLBF algorithm with T = 100, Q = 8 and the NGDBF inversion function used as the starting point for MIPFM construction using the rate 1/2 PEGReg504x1008 LDPC code simulated over a BAWGNC with binary antipodal modulation. Performance curves for M-NGDBF is provided for reference.

CHAPTER 6
CONCLUSIONS AND FUTURE WORK

This work proposes a novel BF algorithm with an explicit construction that builds on an error probability matrix based approach to BF decoding. The developed PLMLBF algorithm out-performs the M-GDBF algorithm at higher SNR's when simulated with the 1/2PEGReg504x1008 LDPC code and comes within 0.25 dB of the M-NGDBF algorithm (with local threshold adaptation). The algorithm was tested on a variety of codes with varying code-length and code-degree to demonstrate efficacy. Unlike GDBF and NGDBF algorithms which were heuristically motivated for performance gains, the proposed PLMLBF algorithm is primarily theoretically motivated.

The smoothing with early stopping and redecoding heuristics offer significant performance gains over the basic PLMLBF algorithm. SM-PLMLBF and VO-PLMLBF comes within 0.25 dB of the SM-NGDBF algorithm when implemented for the 1/2PEGReg504x1008 LDPC code. Redecoding when implemented with PLMLBF gives successive BER performance gains as the number of redecoding phases are increased. Beyond 5 redecoding phases, the simulation might take weeks to complete and on account must be implemented on an FPGA/ASIC.

From the simulations performed, it is observed that beyond an SNR of 4.25 dB the basic PLMLBF algorithm has sub-optimal performance. Beyond 4.25 dB, trapping sets have a significant impact on the decoding performance. The PLMLBF decoding algorithm is not very effective in offsetting the impact of trapping sets. The errors based on trapping sets form a very small proportion of the total samples and on account do not impact the decoding probabilities significantly to enhance bit flipping. The NGDBF algorithm adds the perturbation noise which induces the uncertainty required to reach the flipping threshold and escape trapping sets. The PLMLBF when implemented with an NGDBF based MIPFM construction was an attempt at resolving the PLMLBF trapping set problem by populating the MIPFM with NGDBF error probabilities. This technique resulted in improved BER performance at higher SNR's with error correction capability close to M-NGDBF implemented with local threshold adaptation.

PLMLBF uses a global lookup table for decoding. This approach can significantly reduce hardware implementation complexity when compared to the NGDBF decoding algorithm which uses hardware for each codeword symbol being decoding. Some of the proposed heuristics including lowering quantization bits, repeating MIPFM's for higher SNR values and repeating PFM's in a MIPFM are targeted at reducing the hardware implementation complexity even further.

The proposed MIPFM construction mechanism which is fundamental to the PLMLBF algorithm could be the potential gateway to develop probability matrix constructions with more robust error probabilities. Hardware implementation of the proposed PLMLBF algorithm on an FPGA/ASIC needs to be explored to enable a conclusive performance and hardware complexity comparison with the existing state of the art decoders. A Stochastic computing based hardware implementation of the proposed PLMLBF algorithm is a foreseen research objective.

REFERENCES

[1] C. E. Shannon, "A mathematical theory of communication," *Bell system technical journal*, vol. 27, no. 3, pp. 379–423, 1948.

[2] C. Berrou, A. Glavieux, and P. Thitimajshima, "Near shannon limit error-correcting coding and decoding: Turbo-codes. 1," in *Proceedings of ICC'93-IEEE International Conference on Communications*, vol. 2. IEEE, 1993, pp. 1064–1070.

[3] R. G. Gallager, "Low density parity-check codes," Ph.D. dissertation, Massachusetts Institute of Technology, Cambridge, MA, 1963.

[4] B. Bellalta, L. Bononi, R. Bruno, and A. Kassler, "Next generation ieee 802.11 wireless local area networks: Current status, future directions and open challenges," *Computer Communications*, vol. 75, pp. 1–25, 2016.

[5] E. Perahia and R. Stacey, *Next generation wireless LANs: 802.11 n and 802.11 ac.* Cambridge University Press, 2013.

[6] D. J. MacKay and R. M. Neal, "Near shannon limit performance of low density parity check codes," *Electronics letters*, vol. 33, no. 6, pp. 457–458, 1997.

[7] Y. Kou, S. Lin, 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. 2711–2736, 2001.

[8] M. M. Mansour and N. R. Shanbhag, "High-throughput ldpc decoders," *IEEE transactions on very large scale integration (VLSI) Systems*, vol. 11, no. 6, pp. 976–996, 2003.

[9] F. Guilloud, E. Boutillon, J. Tousch, and J.-L. Danger, “Generic description and synthesis of ldpc decoders,” *IEEE transactions on Communications*, vol. 55, no. 11, pp. 2084–2091, 2007.

[10] T. Wadayama, K. Nakamura, M. Yagita, Y. Funahashi, S. Usami, and I. Takumi, “Gradient descent bit flipping algorithms for decoding ldpc codes,” *IEEE Transactions on Communications*, vol. 58, 2010.

[11] G. Sundararajan, C. Winstead, and E. Boutillon, “Noisy gradient descent bit-flip decoding for ldpc codes,” *IEEE Transactions on Communications*, vol. 62, pp. 58–60, 2014.

[12] R. W. Hamming, “Error detecting and error correcting codes,” *The Bell system technical journal*, vol. 29, no. 2, pp. 147–160, 1950.

[13] S. J. Johnson, “Introducing low-density parity-check codes,” *University of Newcastle, Australia*, p. V1, 2006.

[14] M. P. Fossorier, “Quasicyclic low-density parity-check codes from circulant permutation matrices,” *IEEE Transactions on Information Theory*, vol. 50, no. 8, pp. 1788–1793, 2004.

[15] F. R. Kschischang and B. J. Frey, “Iterative decoding of compound codes by probability propagation in graphical models,” *IEEE Journal on Selected Areas in Communications*, vol. 16, no. 2, pp. 219–230, 1998.

[16] D. E. Hocevar, “A reduced complexity decoder architecture via layered decoding of ldpc codes,” in *IEEE Workshop onSignal Processing Systems, 2004. SIPS 2004.* IEEE, 2004, pp. 107–112.

[17] J. Zhang, Y. Wang, M. P. Fossorier, and J. S. Yedidia, “Iterative decoding with replicas,” *IEEE Transactions on Information Theory*, vol. 53, no. 5, pp. 1644–1663, 2007.

[18] T. T. N. Ly., "Efficient hardware implementations of ldpc decoders, through exploiting impreciseness in message-passing decoding algorithms." Ph.D. dissertation, Université de Cergy Pontoise, France, 2017.

[19] J. Zhang and M. P. Fossorier, "A modified weighted bit-flipping decoding of low-density parity-check codes," *IEEE Communications Letters*, vol. 8, no. 3, pp. 165–167, 2004.

[20] X. Wu, C. Zhao, and X. You, "Parallel weighted bit-flipping decoding," *IEEE Communications letters*, vol. 11, no. 8, pp. 671–673, 2007.

[21] T. C.-Y. Chang and Y. T. Su, "Dynamic weighted bit-flipping decoding algorithms for ldpc codes," *IEEE Transactions on Communications*, vol. 63, no. 11, pp. 3950–3963, 2015.

[22] S. Imani, R. Shahbazian, and S. A. Ghorashi, "An iterative bit flipping based decoding algorithm for ldpc codes," in *2015 Iran Workshop on Communication and Information Theory (IWCIT).* IEEE, 2015, pp. 1–3.

[23] N. Miladinovic and M. P. Fossorier, "Improved bit-flipping decoding of low-density parity-check codes," *IEEE Transactions on Information Theory*, vol. 51, no. 4, pp. 1594–1606, 2005.

[24] O. Al Rasheed, P. Ivaniš, and B. Vasić, "Fault-tolerant probabilistic gradient-descent bit flipping decoder," *IEEE Communications Letters*, vol. 18, no. 9, pp. 1487–1490, 2014.

[25] T. Tithi, C. Winstead, and G. Sundararajan, "Decoding ldpc codes via noisy gradient descent bit-flipping with re-decoding," *arXiv preprint arXiv:1503.08913*, 2015.

[26] T. Richardson, "Error floors of ldpc codes," in *Proceedings of the annual Allerton conference on communication control and computing*, vol. 41, no. 3. The University; 1998, 2003, pp. 1426–1435.

[27] Z. Zhang, L. Dolecek, B. Nikolic, V. Anantharam, and M. Wainwright, "Gen03-6: Investigation of error floors of structured low-density parity-check codes by hardware emulation," in *IEEE Globecom 2006*. IEEE, 2006, pp. 1–6.

[28] L. Dolecek, Z. Zhang, V. Anantharam, M. J. Wainwright, and B. Nikolic, "Analysis of absorbing sets and fully absorbing sets of array-based ldpc codes," *IEEE Transactions on Information Theory*, vol. 56, no. 1, pp. 181–201, 2010.

[29] T. T. Tithi, "Error-floors of the 802.3an ldpc code for noise assisted decoding," Ph.D. dissertation, Utah State University, Logan, UT, 2019.

[30] D. MacKay, "Encyclopedia of sparse graph codes, accessed june. 2019. available: http://www.inference.phy.cam.ac.uk/mackay/codes/data.html."

[31] M. Ismail, I. Ahmed, and J. Coon, "Low power decoding of ldpc codes," *ISRN Sensor Networks*, vol. 2013, 2013.