ERASURE BASED SOFT DECISION DECODING FOR OPTICAL TRANSPORT NETWORKS

A Thesis

by

DILEEP KUMAR SOMA

Submitted to the Office of Graduate and Professional Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

Chair of Committee,	Krishna Narayanan
Committee Members,	Seong Choi
	Alexander Sprinston
	Anxiao Jiang
Head of Department,	Miroslav M. Begovic

December 2018

Major Subject: Electrical Engineering

ABSTRACT

A concatenated soft decision forward error correcting (FEC) coding scheme with an inner low density generator matrix (LDGM) code and an outer product code (PC) is considered for applications in high-speed optical communications. First, we evaluate the performance of various choices of inner and outer codes when the inner decoder uses a soft-decision decoder and the outer decoder is a hard-decision error-only decoder. Then, we evaluate the performance of the concatenated coding scheme when erasures are introduced at the output of the inner decoder and the outer decoder is an errors-and-erasures decoder. An exact expression for the number of errors and erasures that are guaranteed to be decoded by an iterative decoder is derived. Then, an approach for deriving an approximation of the error probability for the erasures decoder is proposed. Using this approach, optimal thresholds can be chosen for declaring erasures at the output of the inner decoder. It is shown that the codeword error rate for the error and erasures iterative decoder can be better than that of errors only iterative decoder in the error floor region. It is also shown that simulation results are in close agreement with the mathematical approximations developed at error floors.

ACKNOWLEDGMENTS

I would like to thank the Electrical and Engineering Department of Texas A&M University for providing necessary resources for the completion of my thesis. They are very helpful in every regard and provided resources in a timely manner. Thanks to HRPC for providing necessary computation power for the simulations provided in this work. Thanks to my committee members for their time, effort and inputs for the successful completion of my thesis.

I would like to thank my colleague, and friend, Vamsi Krishna Amalladinne, for his constant support, inputs and motivation throughout writing this material. Discussions with him are always fruitful and enabled me to think in a different perspective.

I would like to thank my family members for their constant support and love irrespective of circumstances. I would like to thank my friends Deepika Ravipati, Jyothsna Kurra, Deepak Singh Mahendersingh and Amritha Bal for coming into my life and being there for me during all the situations.

I specially like to thank my advisor Dr. Krishna Narayanan. His amazing perspective and skills helped to complete my thesis in a timely manner. Every discussion with him is new learning and motivating. I really want to thank him for his patience with me and the opportunity to work with him.

CONTRIBUTORS AND FUNDING SOURCES

Contributors

This work was supported by a dissertation committee consisting of Professor Krishna Narayanan, professor Seong Choi and professor Alexander Sprinston of the Department of Electrical and Computer Engineering and Professor Anxiao Jiang of the Department of Computer Science Engineering.

All the work conducted for the thesis (or) dissertation was completed by the student with the guidance of committee.

Funding Sources

Graduate study was supported by an assistantship from Texas A&M University and research grant from Dr. Kishna Narayanan.

NOMENCLATURE

LDPC	Low Density Parity Check
LDGM	Low Density Generator Matrix
TAMU	Texas A&M University
FEC	Forward Error Correcting
BCH	Bose-Chauduri Hocquengem
RS	Reed-Solomon
PEG	Progressive Edge Growth
BSC	Binary Symmetric Channel
BEC	Binary Erasure Channel

TABLE OF CONTENTS

Page

ABSTRACT ii

ACKNOWLEDGMENTS iii

CONTRIBUTORS AND FUNDING SOURCES iv

NOMENCLATURE v

TABLE OF CONTENTS vi

LIST OF FIGURES viii

LIST OF TABLES x

1. INTRODUCTION 1

    1.1 Main Contributions 2
    1.2 Organization 3

2. LDPC CODES 4

    2.1 Construction of Tanner graph 5
    2.2 Belief propagation Decoding Algorithm 6
    2.3 Low density generator matrix (LDGM) codes 8

3. PRODUCT CODES AND HALF PRODUCT CODES 11

    3.1 Product Codes 11
        3.1.1 Preliminaries 11
        3.1.2 Encoding Process 11
        3.1.3 Decoding Process 11
        3.1.4 Error floor analysis 13
    3.2 Half Product codes 14
        3.2.1 Encoding of HPC 15
        3.2.2 Decoding and Error Floor Analysis 16
    3.3 Comparison 17

4. CONCATENATED CODES 20

    4.1 Encoding and Decoding Operations 20
    4.2 LDGM code Design 21

4.3 Product code Design 21

5. INTRODUCTION OF ERASURES 23

5.1 Enumeration of stopping patterns 25
5.2 Correcting capability of product code decoder 26
5.3 Analytical Model for Error Floors 30
5.4 Optimal threshold 32

6. RESULTS AND DISCUSSION 33

6.1 Comparison between Product and Half Product codes 33
6.1.1 Observations 34
6.2 Results based on erasures 34
6.2.1 Simulations of Product code with (15,11) BCH codes as component codes .. 34
6.2.2 Simulations of Product code with (1023,992,8) BCH code as component codes 35

7. CONCLUSIONS 38

REFERENCES 39

LIST OF FIGURES

FIGURE Page

2.1 LDPC code example 4

2.2 Demonstration of 2 and 4 cycles 5

2.3 LDGM ensemble (reprinted form [1]) 8

3.1 Structure of product code (reprinted form [2]) 12

3.2 Decoding process of a product code (reprinted form [2]) 12

3.3 A Stopping pattern with 4 errors 13

3.4 Half and full forms of half product code 15

3.5 Half and full forms of half product codes illustrating a stopping pattern 16

3.6 Illustrating the minimum stopping pattern for half product code in its half and full form [2]. 17

4.1 Overall Channel Model 20

5.1 Proposed Channel Model 23

5.2 Figure for illustrating the channel model 24

5.3 Stopping pattern for PC formed by (1023,992) component code 26

5.4 Stopping pattern for PC formed by (15,11) component code 27

5.5 Stopping patterns for a code Q with minimum distances of column and row codes are $d_1 = 8$ and $d_2 = 6$ respectively. In both the patterns, X represents error, E represents erasure and D represents don't care (a) Illustrating a stopping pattern with $x = 7$, $q = 1$ and $k = 1$ (b) Illustrating a stopping pattern with $x = 9$, $q = 1$ and $k = 1$ 28

6.1 Comparison plot between HPC and PC 33

6.2 Performance under erasures with BCH decoder of(1023,993) 35

6.3 Performance of PC formed by (15,11)with erasures 36

6.4 Mathematical approximation of PC formed by (1023,993) with erasures..... 37

6.5 Actual plot of PC formed by (1023,993) with error and erasures case 37

LIST OF TABLES

TABLE Page

5.1 Table for the mathematical values of error probability for $(1023^2, 992^2, 8^2)$. 31

1. INTRODUCTION

The amount of data carried through the internet backbone is increasing due to services such as social networking, video streaming, coordinated high graphic gaming, cloud computing etc. Indeed, the estimated global IP traffic has increased five folds from 2011 to 2016. To cope with increasingly higher traffic, optical transport networks (OTN) need to carry substantially high data rates. While current OTNs carry 100 to 400 Gbps, the increasing trend in traffic suggests that OTNs will roughly need data rates of the order of 10 Tbps by 2025. High data rates introduce impairments such as non-linearities, chromatic dispersion, noise due to amplified spontaneous emission and polarization mode dispersion. Providing reliable communication at these high data rates is a challenging task due to the above mentioned distortions. This requires encoding and decoding schemes which can provide high coding gains.

Several powerful classes of codes have been developed in recent times which can provide near-Shannon-limit performance. Quasi cyclic-low density parity check (LDPC) codes and polar codes are examples of such coding schemes. While these codes provide large coding gains, the computational complexity required for implementing the encoder/decoder at very high data speeds is extremely high. It is natural to wonder if advances in hardware technology predicted by Moore's law will be sufficient to make standard implementations of message-passing LDPC decoders possible at the required data rates. Unfortunately, advances in hardware technology seem to be offset by the increasing data rates predicted by Butter's law[3]. This leaves a gap between the hardware resources available and the hardware resources needed to implement computationally complex advanced coding schemes at very high data rates. Thus, current approaches are not scalable even with the improved hardware technologies. This makes innovation in the design of encoders and decoders essential for implementing decoders for OTNs.

To support the data rate requirements of the future communication systems OTNs need to obtain error floors as low as 10^{-15} at code rates exceeding 0.9. Most of the classical research on forward error correction (FEC) is focused on hard decision decoders. However, with the increase

demand for low error rates and recent advances in hardware technology led to a interest on decoders which exploit soft outputs of the channel for OTNs. To get an idea of the possible gains from soft-decision decoding, notice that for rate 0.8, the Shannon limit of a quantized binary-input Gaussian channel is 3.37 dB with two levels and 2.37 dB with four levels. Since a gain of 1 dB is possible, it makes sense to focus on the design of low-complexity soft-decision decoders.

The design of soft decision decoders that provide high coding gains at such high data rates is very challenging. One promising candidate for such an FEC is concatenated codes, using a combination of algebraic codes and codes on graphs. Near-optimal coding gains can be achieved by using concatenated codes with reasonable complexity. Algebraic codes such as Bose- Chaudhuri and Hocquenghem (BCH) and Reed-Solomon (RS) codes have very efficient hard decision and erasure decoders which can be implemented with minimal hardware even at high code rates, and soft decision decoders for small length LDPC or low density generator matrix (LDGM) can be implemented with reasonable hardware. These class of codes are studied in[1]. This work utilizes soft decision outputs from the channel in a concatenated coding scheme. This scheme contains an inner soft decision decoder and outer hard decision decoder. The inner decoder accepts the soft values from the channel and provides soft output. Hard decisions are made by the outer decoder on the soft outputs provided by the inner decoder.

1.1 Main Contributions

The decoding scheme presented in this thesis improves on [1] with the introduction of erasures into the outer decoder. Erasures can exploit soft values even further for improved coding gain. For an iterative row-column decoder in product code (PC),an exact expression for the number of errors and erasures that are guaranteed to be decoded is derived. An approximate expression for the error rate in the error floor region for this decoding scheme is derived. Using this approach, optimal thresholds can be chosen for declaring erasures at the output of the inner decoder. It is concluded that introduction of erasures performs better than the existing error only scheme.

1.2 Organization

Chapter 2 gives an introduction to LDPC and LDGM codes. Chapter 3 discusses product and half product codes. It is shown in this chapter that product codes perform better than half product codes for our application. Chapter 5 discusses our work on concatenation of a soft decision inner code with a hard decision outer code. It details encoding and decoding for the proposed scheme. It discusses error floor analysis of the proposed scheme and bounded distance formulation for product codes using iterative row-column decoder. Chapter 6 provides various simulations that are performed in this work, discussion of results and conclusions are provided.

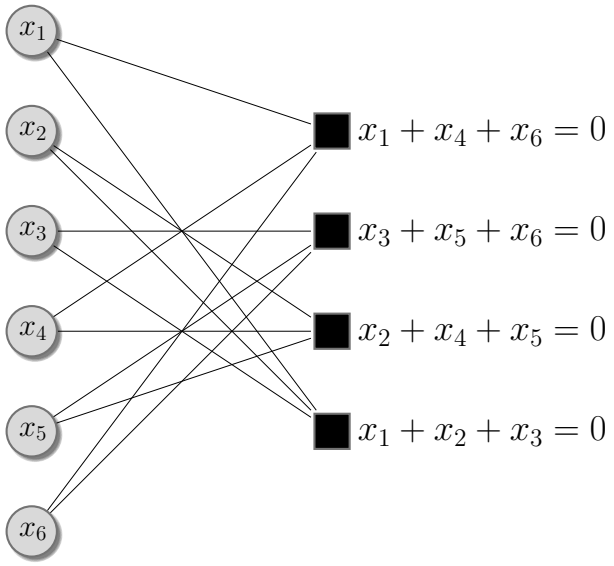
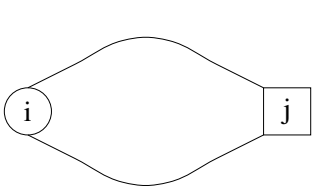


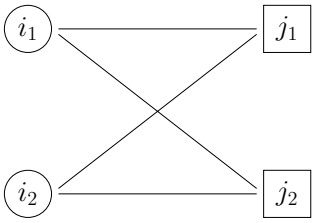
Figure 2.1: LDPC code example

Introduced by Gallager in 1960, LDPC codes are linear block codes defined by a sparse parity-check matrix[4]. An (n, k) LDPC code can be represented by a Tanner graph which is a bipartite graph with n circle nodes, representing the bit nodes, and $n - k$ square nodes, representing the check nodes as shown in fig: 2.1. The above bipartite graph can be represented in a matrix form. Let $H = [h_{ij}]$ be a binary $(n - k) \times n$ matrix where (h_{ij}) is 1 when there is an edge between i^{th} check node and j^{th} variable in the graph. The degree d_v of a variable node v is defined as the number of check nodes connected to it. Similarly,the degree d_c of a check node c is defined as the number of variable nodes connected to it. The adjacency matrix of the Tanner graph acts as the parity check matrix for the LDPC code. The sparsity of H is a key property that enables efficient implementation of decoding algorithms for LDPC codes [1].

[1] see the paper on "Low density parity check codes" by gallager, 1962 for more detailed information



(a) Two cycle Example



(b) Four Cycle Example

Figure 2.2: Demonstration of 2 and 4 cycles

2.1 Construction of Tanner graph

There are many ways of constructing a parity check matrix for an LDPC code. Apart from sparsity of the matrix, there are other factors which can effect the performance of decoding algorithms. One of the key factors that drastically effects the performance is cycles in the graph.A Tanner graph is said to have an n cycle if it has a cycle of length n. Cycles, especially short cycles degrade the performance of LDPC decoders, because they affect the independence of the extrinsic information exchanged in the iterative decoding. It is known that parity check matrix with minimal cycles gives the best performance. Fig:2.2 shows an example for four and two cycles in a Tanner graph .

Parity check matrix is generated generally by a randomized permutation. consider each edge is connected to a variable node through a socket and check node through another socket. Total number of sockets on the left will be equal to the total number of sockets to the right. Random connection between left and right sockets results in a parity check matrix. This process of random generation is repeated until resultant parity check matrix is 2-cycle free. Another mode of construction is progressive edge growth (PEG). In PEG based construction, each edge is to be constructed progressively such that there are no n-cycles for some fixed integer n. Though this kind of construction is slow, PEG is more reliable as it gives constructions which are n-cycle free. This construction is discussed in detail later during LDGM construction in section 4.2.

2.2 Belief propagation Decoding Algorithm

LDPC codes are decoded using iterative message passing algorithms . At every iteration messages are passed between variable node to check node and check node to variable node. Message sent by a variable node to a check node is computed based on the information it receives from the channel and messages received from other check nodes attached to it. The check nodes connected to a variable node are called as neighboring check nodes. Similarly the variable nodes connected to a a check node are called as neighboring variable nodes. The messages that are sent from either check node or variable node should contain only extrinsic information i.e, The information passed from any check node i to variable node j should not contain the information sent by j to i previously. The same applies for the messages passed by variable nodes to check nodes.

Belief propagation algorithm is an important subclass of message passing algorithms. In belief propagation algorithms, the messages that are passed between variable and check nodes are probabilities or beliefs.

For any binary random variable x, its likelihood is defined as $l(x) = \Pr[x = 0]/\Pr[x = 1]$. The conditional likelihood between a binary random variable x and a real random variable y is given by $l(x|y) = \Pr[x = 0|y]/\Pr[x = 1|y]$. The log-likelihoods $L(x)$ and $L(x|y)$ are defined as $L(x) = \ln\ \ l(x)$ and $L(x|y) = \ln\ \ l(x|y)$.

Let $\underline{v} = (v_1, v_2, ..., v_n)$ be a codeword of an (n, k) LDPC code. Let $\underline{x} = (x_1, x_2, ..., x_n)$ denote the BPSK modulated vector denoting the corresponding codeword vector $\underline{v}$. Hence, $x_j = (-1)^{v_j}$ denotes the j^{th} transmitted binary symbol over the channel.($v_j \in \{0, 1\}$and hence$x_j \in \{1, -1\}$). Let $\underline{y} = (y_1, y_2, ..., y_n)$ denote the corresponding received vector at the decoder. Then $\underline{y}$ can be modeled as $\underline{y} = \underline{x} + \underline{n}$, where $\underline{n}$ denotes additive white Gaussian noise(AWGN) of mean 0 and covariance matrix $\sigma^2 I$(I denotes identity matrix).

The conditional probability of x_j given y_j can be written as

$$\Pr(x_j = b|y_j) = \left[1 + \exp \frac{-2y_j b}{\sigma^2}\right]^{-1}, b \in \pm 1 \tag{2.1}$$

Hence, the conditional log-likelihood of v_j and y_j can be computed as

$$L(v_j|y_j) = \frac{2y_j}{\sigma^2}. \tag{2.2}$$

The algorithm can be summarized in five steps as given below[5].

Step 1: For all variable nodes $j \in \{1, 2, ...n\}$ of the graph, initialize L_j according to the equation (2.2). Then initialize $L_{j\to i}$, the beliefs that are scheduled to be sent from variable node$j$ to check node i as $L_{j\to i} = L_j \; \forall \; j \in \{1, 2, ..., n\}, i \in \{1, 2, .., n-k\}$.

Step 2: This step marks the start of iterative procedure of the algorithm. Once the beliefs are received from variable nodes to check nodes, the check node update and propagate beliefs to its neighbor variable node. At iteration l of the algorithm, check node i sends belief $L_{i\to j}$ to the variable node j according the update equation.

$$L_{i\to j} = 2\tanh^{-1}\left(\prod_{j'\in N(i)\setminus\{j\}} \tanh\left(\frac{L_{j'\to i}}{2}\right)\right), \tag{2.3}$$

where $N(i)$ denotes all the variable nodes connected to the check node i.

Step 3: Variable node computes the outgoing beliefs as follows.

$$L_{j\to i} = L_j + \sum_{i'\in N(j)\setminus\{i\}} L_{i'\to j}. \tag{2.4}$$

Step 4: In this step, total Log likelihood ratio at each variable node $j \in \{1, 2, ..., n\}$ is computed as

$$L_j^{\text{total}} = L_j + \sum_{i\in N(j)} L_{i\to j} \tag{2.5}$$

Step 5: This step checks for the stopping condition of the algorithm for each variable node $j \in \{1, 2, ..., n\}$.

$$\hat{v}_j = \begin{cases} 1 & \text{if } L_j^{\text{total}} < 0 \\ 0 & \text{else,} \end{cases} \tag{2.6}$$

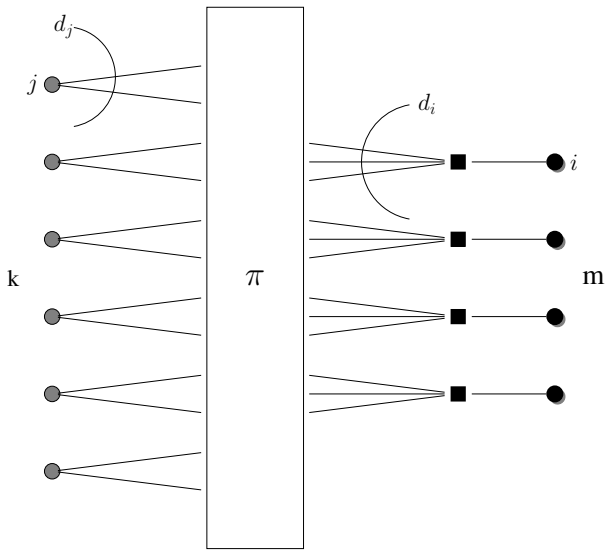


Figure 2.3: LDGM ensemble (reprinted form [1])

to obtain an estimated vector $\underline{v}$. If $H\underline{\hat{v}}^T = 0$ or if the number of iterations l reaches the maximum permitted number, then stop, otherwise start over again from step 2.

2.3 Low density generator matrix (LDGM) codes

LDGM codes are an extension to LDPC codes. LDPC codes have a parity check matrix which is sparse while the LDGM codes have a sparse generator matrix.

LDGM codes have a sparse generator matrix in contrast to LDPC codes which have a sparse parity check matrix.

Similar to LDPC codes, LDGM codes can also be represented by Tanner graphs, circles represent variable nodes while squares represent check nodes(see figure 2.3). The variable nodes in LDGM can be partitioned into 2 different types, message nodes (lightly shaded circle nodes) and parity nodes (dark shaded circle nodes). The number of check nodes in LDGM code is equal to the number of parity nodes. Each parity node has a degree 1 and is connected to a unique check node. Hence, the degree d_{i} of a check node $i \in \{1, 2, .., m\}$ is the number of connections on that check node minus 1. This implies that the connection between parity node and check node is not counted towards d_{i}. The degree d_{j} of variable node $j \in \{1, 2, .., k\}$ is equal to the number of edges

incident on that node.

The connections between check nodes and variable nodes are determined as follows. A message node j with a degree d_j will be assigned j sockets. Similarly, a check node i with a degree d_i will be assigned i sockets. Every check node is connected to a unique parity node. Thus, the number of sockets on the left and right are equal. These sockets can be connected using any algorithm. In this work, progressive edge growth (PEG) algorithm is used to establish connections between these sockets. The motivation behind choosing this algorithm and implementation details are discussed later in section 4.2.

The parity check matrix H corresponding to a systematic LDGM code takes the form $H = [P|I_m]$, where P is an $m \times k$ sparse matrix and I_m denotes the $m \times m$ identity matrix. The $(i, j)^{th}$ entry of matrix P is 1 if there exists an edge between check node i and message node j, else it is 0. The systematic generator matrix G corresponding to this LDGM code is given by $G = [I_k|P^T]$.

Various forms of LDGM codes are studied in literature with different names. The Luby-transform (LT) [6], which are a form of rateless codes have a variable number of parity check bits, whose size can be changed upon the request of the decoder. Another important class of LDGM codes are the Raptor codes[7, 8, 9]., which are formed by concatenating an LT code with an outer block code to lower the error floors. These two models uses LDGM with puncturing to increase code rate.

The LDGM implementation provided in this work does not consider puncturing. The parameters k, m and the graph connections are not changed after the design. Hence, the rate of such a LDGM code is $\frac{k}{k+m}$. Thus, the LDGM codes used in this design are similar in spirit to the class of Error Reduction codes (ERC) [10] [[11],chap 9]. These class of codes do not guarantee the complete correction of the codeword, but corrects a fraction of errors.

The decoding procedure of LDGM codes is same as that of LDPC codes. LDPC codes are known to provide lower error floors than the LDGM codes. However, for this concatenated scheme to work, the inner code only needs to achieve a threshold error probability at a reasonably low SNR. It is the job of the outer decoder to further reduce this probability to attain error floors. If attaining

error floors is not the motive of the inner code, LDGM codes are a better choice than LDPC codes as they achieve this threshold probability at a lower SNR than LDPC codes. Also, the complexity of encoding an LDGM code is less than that of the LDPC code. The outer code is designed in such a way that, error floor is reduced to the target error probability 10^{-15}, when inner code decodes the received codeword to p_{th}. Product codes(PC) and half product codes(HPC) are known to provide very low error floors and hence are a good choice for the outer code. The design and decoding procedure of these codes are discussed in next chapter.

3. PRODUCT CODES AND HALF PRODUCT CODES

3.1 Product Codes

3.1.1 Preliminaries

Elias introduced product codes in 1954 and Tanner generalized it in 1981. In a codeword of $n_1 \times n_2$ matrix of symbols, each column is a codeword corresponds to a code of length n_1 and each row codeword corresponds to a code of length n_2. If the column and row codes are $C_1(n_1, k_1, d_1)$ code and $C_2(n_2, k_2, d_2)$ code respectively, then the resultant product code is a $C(N, K, D)$ linear code with length $N = n_1 n_2$, dimension $K = k_1 k_2$ and minimum distance $D = d_1 d_2$ [12]. Throughout this chapter, the notation C is used to define a product code formed by component codes $C_1(n_1, k_1, d_1)$ and $C_2(n_2, k_2, d_2)$ as column and row codes respectively. The structure of this product code C can be seen in Figure 3.1.

3.1.2 Encoding Process

The message bits are arranged in the form of $k_1 \times k_2$ matrix. Each column of this matrix is encoded using the code C_1 resulting in an $n_1 \times k_1$ matrix. Each row of this resulting matrix is then encoded using $C_2$to a get a codeword of size $n_1 \times n_2$ of the product code. The generator matrix of a product codes is a Kronecker product of generator matrices of C_1 and C_2. Thus for any codeword $c \in C_1 \times C_2$ can be written as

$$c = \mathbf{G_1}^T m \mathbf{G_2} \tag{3.1}$$

where $\mathbf{G_1}$ and $\mathbf{G_2}$ are generator matrices of C_1 and C_2 and m denotes the message matrix of size $k_1 \times k_2$.

3.1.3 Decoding Process

An iterative hard decision decoding algorithms for product codes was described by Abrahamson in 1968. The structure of the product code facilitates independent decoding of row and column codewords. A cascade of row-column decoders are employed at each iteration. All rows are de-

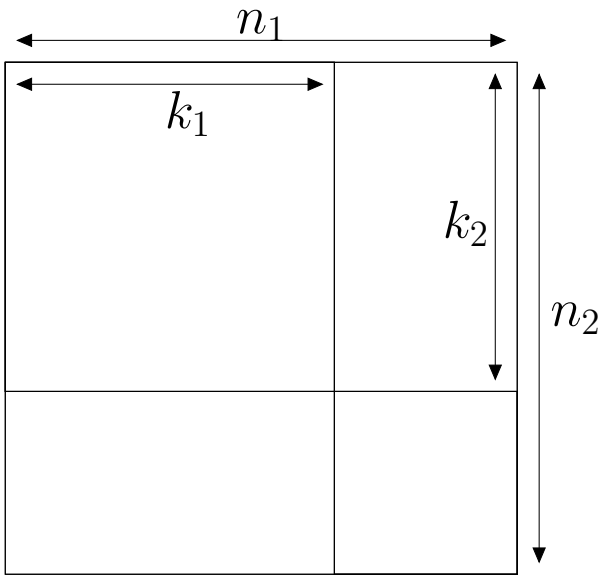


Figure 3.1: Structure of product code (reprinted form [2])

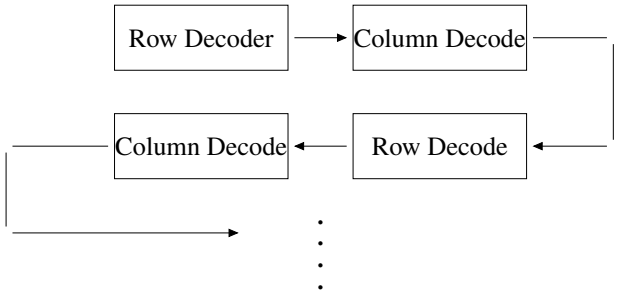


Figure 3.2: Decoding process of a product code (reprinted form [2])

coded at once using the row decoder of C_2 and all columns are decoded with the column decoder of C_1. As with any iterative decoder, iterative process is continued until all the errors are corrected or the codeword becomes uncorrectable. The decoding process is explained in figure 3.2.

Since the decoder is a cascade of row-column decoding, the entire codeword matrix is not considered in totality in any decoding step. This results in a decoder which is incapable of correcting some error patters, which are otherwise correctable by the optimal maximum likelihood(ML) decoder. As the product code is a linear code, the ML decoder should be able to correct upto $\lfloor \frac{d_{min}-1}{2} \rfloor$($\lfloor \rfloor$ denotes the least integer function) errors, where $d_{min} = d_1 d_2$. However, the iterative

$$\begin{array}{ccccccc} 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & X & 0 & 0 & 0 & X & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & X & 0 & 0 & 0 & X & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{array}$$

Figure 3.3: A Stopping pattern with 4 errors

decoder is guaranteed to correct upto $\lfloor \frac{d_1-1}{2} \rfloor \lfloor \frac{d_2-1}{2} \rfloor$ errors which is less than $\lfloor \frac{d_{min}-1}{2} \rfloor$. This occurs due to certain uncorrectable pattern of errors, also known as stopping patterns. This is explained in more detail in the following example. Consider a (49,16,9) product code formed by (7,4,3) 1-error correcting Hamming codes as row and column codes. The ML decoder can correct maximum of 4 errors. However, the iterative decoder explained above cannot correct the error pattern given in figure 3.3, even though the number of errors is 4 (errors are represented by X). This happens because the row decoder(column decoders) cannot correct rows(columns) 2 and 6, since there are two errors in these rows(columns) These stopping error patterns give rise to a error floors.

3.1.4 Error floor analysis

To analyze the error floors, the probability of occurrence of all the stopping patterns needs to be taken into account. However, only the dominant stopping patterns are considered in this work which provides an approximation for the error floors. Consider the product code $C(n_1n_2, k_1k_2, d_1d_2)$ defined in section 3.1.1. The error correcting capabilities of the column code C_1 and row code C_2 are given by $t_1 = \lfloor \frac{d_1-1}{2} \rfloor$ and $t_2 = \lfloor \frac{d_2-1}{2} \rfloor$ respectively. Thus, a column cannot be corrected by C_1 if it contains more than t_1 errors and a row cannot be corrected by C_2 if it contains more than t_2 errors.This happens when there are atleast $t_1 + 1$ errors among n_1 column positions and atleast $t_2 + 1$ errors among n_2 row positions. Hence, the minimum number of errors N_{min} required for a pattern to become a stopping pattern of this iterative decoder is given by,

$$N_{min} = (t_1 + 1)(t_2 + 1). \tag{3.2}$$

However, the optimal ML decoder can correct a maximum of t errors, where,

$$\begin{aligned} t &= \lfloor \frac{d_{min} - 1}{2} \rfloor \\ &= \lfloor \frac{(2t_1 + 1)(2t_2 + 1) - 1}{2} \rfloor \\ &= 2t_1 t_2 + t_1 + t_2. \end{aligned} \tag{3.3}$$

Notice that $t > N_{min}$ and the difference $t - N_{min}$ being positive is a reason for error floors. The number of stopping patterns with $t_1 + 1$ errors in each row and $t_2 + 1$ errors in each column that can occur in the product code C is given by,

$$N_{DEP} = \binom{n_1}{t_1 + 1} \binom{n_2}{t_2 + 1}. \tag{3.4}$$

Even though there are many stopping patterns which can contribute to error floors, in low error rate regime, the dominant stopping patterns are the ones which have $t_1 + 1$ errors in each row and $t_2 + 1$ errors in each column. The probability of occurrence of these dominant stopping patterns serves as the lower bound on error floor. This quantity is given by,

$$\begin{aligned} P_{\mathrm{e}} &= N_{\mathrm{DEP}} \times p^w \\ &= \binom{n_1}{t_1 + 1} \binom{n_2}{t_2 + 1} \times p^{(t_1+1)(t_2+1)}, \end{aligned} \tag{3.5}$$

where p denotes the input bit error rate of the channel, w denotes weight of the dominant stopping pattern.

3.2 Half Product codes

Symmetric product codes, also known as half product codes were first discussed by Justesen [13]. For any component code $C(n, k, d)$, an $n \times n$ symmetric product code $C_{\mathrm{HPC}}(N, K, D)$ can be

$$\begin{matrix} 0 & 0 & 1 & 1 & 1 & 0 & 1 \\ & 0 & 0 & 0 & 1 & 0 & 0 \\ & & 0 & 1 & 0 & 1 & 0 \\ & & & 0 & 1 & 0 & 0 \\ & & & & 0 & 1 & 0 \\ & & & & & 0 & 0 \\ & & & & & & 0 \end{matrix}$$

(a) Half form of the half product code used for transmission

$$\begin{matrix} 0 & 0 & 1 & 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 \end{matrix}$$

(b) Full code of half product code used for decoding

Figure 3.4: Half and full forms of half product code

formed[12]. In a HPC, rows and columns are formed by the same component code. The diagonal elements are made to be zeros (Figure 3.4). The HPC $C_{HPC}(N, K, D)$ formed by component code $C(n, k, d)$ will have $N = \frac{n(n-1)}{2}$ and dimension $K = \frac{k(k-1)}{2}$ and minimum distance D that is described by (3.6) [12].

$$D_{\min} \geq \begin{cases} \frac{3d^2}{4} & \text{if } d \text{ is odd,} \\ \frac{d(3d+1)}{4} & \text{if } d \mod 4 = 1, \\ \frac{d(3d+1)+2}{4} & \text{if } d \mod 4 = 3. \end{cases} \tag{3.6}$$

Because of the symmetric nature of HPC, only the upper triangular matrix of the codeword matrix is transmitted. The following example of a half product code formed with (7,4,3) Hamming code as component code illustrates the structure of both half code (used for transmission) and full code (used for decoding).

3.2.1 Encoding of HPC

A symmetric $k \times k$ message matrix is formed by filling the lower triangular matrix with $\frac{k(k-1)}{2}$ message bits, the diagonal with all zeros and an upper triangle which is the transpose of the lower triangle [1]. This $k \times k$ symmetric message matrix is then encoded into an $n \times n$ symmetric PC using the (n, k, d) row and column component codes similar to the encoding of a PC [2]. Let m denote

[1] check the paper on "symmetric product codes" by H.D. Pfister, S.K. Emmadi and K. Narayanan for the detailed analysis on half product codes

$$\begin{array}{ccccccc} 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ & 0 & 0 & 0 & 0 & 0 & 0 \\ & & 0 & 0 & 0 & 0 & 0 \\ & & & 0 & X & 0 & X \\ & & & & 0 & X & 0 \\ & & & & & 0 & X \\ & & & & & & 0 \end{array}$$

(a) Half form of the half product code

$$\begin{array}{ccccccc} 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & X & 0 & X \\ 0 & 0 & 0 & X & 0 & X & 0 \\ 0 & 0 & 0 & 0 & X & 0 & X \\ 0 & 0 & 0 & X & 0 & X & 0 \end{array}$$

(b) Full code of half product code

Figure 3.5: Half and full forms of half product codes illustrating a stopping pattern

a $k \times k$ symmetric message matrix, then the codeword $c \in C_{HPC}$ corresponding to m is given by,

$$c = \mathbf{G}^T m \mathbf{G}, \tag{3.7}$$

where $\mathbf{G}$ is the generator matrix of component code C.

3.2.2 Decoding and Error Floor Analysis

The decoding of half product codes is performed with a iterative cascade of row and column decoding similar to product code decoding. Similar to PC, stopping patterns cause error floors in HPC. An example for such a stopping pattern of a half product code formed by (7,4,3) Hamming code as component code is given in figure 3.5 (X denotes error and 0 denotes the correct bit). It is straight forward to see that the error pattern shown in Figure 3.5b cannot be corrected when this iterative decoder is used to decode the HPC.

Consider HPC $C_{HPC}\left(\frac{n(n-1)}{2}, \frac{k(k-1)}{2}, D\right)$ formed using $C(n,k,d)$ as component code which can correct $t = \lfloor \frac{d-1}{2} \rfloor$ errors. For an error pattern to be uncorrectable, all the erroneous rows and columns should contain atleast $t+1$ errors. Error patterns can be enumerated as shown in figure 3.6 (figure shows only erroneous rows and columns).

It is seen that, in the above pattern there are $t+2$ rows and columns with weight $t+1$. The number of such stopping patterns in C_{HPC} is given by $N_{\text{DEP}} = \binom{n}{t+2}$. Thus, the probability of occurrence of these stopping patterns is given by [2],

$$\begin{array}{cccccccc}
0 & X_1 & X_2 & X_3 & ... & X_t & X_{t+1} \\
 & 0 & X_2 & X_3 & ... & Xt & X_{t+1} \\
 & & 0 & X_3 & ... & Xt & X_{t+1} \\
 & & & . & & . & . \\
 & & & & . & . & . \\
 & & & & & 0 & X_{t+1} \\
 & & & & & & 0
\end{array}$$

(a) Half form

$$\begin{array}{cccccccc}
0 & X_1 & X_2 & X_3 & ... & X_t & X_{t+1} \\
X_1 & 0 & X_2 & X_3 & ... & X_t & X_{t+1} \\
X_2 & X_2 & 0 & X_3 & ... & X_t & X_{t+1} \\
 & . & & . & & . & . \\
 & . & & . & . & . & . \\
X_t & X_t & X_t & & & 0 & X_{t+1} \\
X_{t+1} & X_{t+1} & X_{t+1} & & & X_{t+1} & 0
\end{array}$$

(b) Full code

Figure 3.6: Illustrating the minimum stopping pattern for half product code in its half and full form [2].

$$\begin{aligned}
P_{\mathrm{e}} &= N_{\mathrm{DEP}} \times p^{w} \\
&= \binom{n}{t+2} \times p^{\frac{(t+1)(t+2)}{2}},
\end{aligned} \tag{3.8}$$

where p denotes the input bit error rate of the channel, w denotes weight of the dominant stopping pattern in its half form.

3.3 Comparison

As HPC uses its half form for transmission while PC uses it full form, for a fair comparison, the codeword length (block length) should be the same. Thus, the relationship between the parameters of component codes $C_1(n_1, k_1, d_2)$ and $C_2(n_2, k_2, d_2)$ corresponding to PC and HPC respectively should be as given below

$$\begin{aligned}
n_1^2 &= \frac{n_2(n_2-1)}{2} \\
&\approx \frac{n_2^2}{2} \\
\implies n_2 &\approx n_1\sqrt{2}.
\end{aligned}$$

Similarly, $k_2 \approx k_1\sqrt{2}$ and $d_2 \approx d_1\sqrt{2}$. Hence, for the same code rate, HPC has a higher minimum distance than PC. The minimum number of errors for a pattern to become a stopping pattern in HPC is given by,

$$W_{min}^{HPC} = \frac{(t_2+1)(t_2+2)}{2},$$

where t_2 denotes the error correcting capability of the component code C_2.

Contrastingly, the minimum number of errors for a pattern to become a stopping pattern in PC is given by,

$$W_{min}^{PC} = (t_1+1)^2,$$

where t_1 denotes the error correcting capability of the component code C_1.

Since $d_2 \approx d_1\sqrt{2}$, we have $t_2 \approx t_1\sqrt{2}$. Thus, the quantity $W_{min}^{HPC} - W_{min}^{PC}$ can be computed as

$$\begin{aligned} W_{min}^{HPC} - W_{min}^{PC} &= \frac{(t_2+1)(t_2+2)}{2} - (\frac{t_2}{\sqrt{2}}+1)^2 \\ &= \frac{(3-2\sqrt{2})t_2}{2} \\ &\approx 0.08t_2 > 0. \end{aligned}$$

It can be seen from the above equation, that for a given code rate, the size of the minimum stopping pattern for an HPC is larger than that of a PC. This implies that HPC has a lower error floor than PC, and this effect is pronounced when block length is large. However, the error floors resulting from these codes are of magnitudes less than 10^{-25}, a regime not of our current interest. Hence, it is more meaningful to compare the performance of these two codes in the waterfall region, where error floors are not the significant cause for error probability. Section 6.1 shows the

performance comparison between a PC formed by $(1023, 993, 7)$ BCH code and a HPC formed by $(1446, 1402, 9)$ punctured BCH code as component codes. It is observed that PC outperforms the HPC in the regime of interest for the codes mentioned above. Also, the computational complexity of implementing this HPC is higher than that of the PC because of increased codeword length. This makes the PC formed by $(1023, 993, 7)$ BCH code a more attractive choice than HPC for the outer code. It is also shown in section 6.1, that threshold probability p_{th} required for the inner code with PC used as an outer code is 4×10^{-3}, when the target error rate is fixed at 10^{-15}.

The Concatenated code employed in this work contains a soft decision inner LDGM code and an outer product code (Figure 4.1). This construction was proposed in [1] with the stair case code as outer code. The job of the inner LDGM decoder is to reduce the bit error rate to a target threshold and the product code further brings this down to the error floor.

4.1 Encoding and Decoding Operations

The information bits are arranged into a $M \times M$ matrix which is encoded into a $N \times N$ matrix with a product code encoder. A random permutation π which is known at the decoder is then applied on the encoded $N \times N$ matrix. The inner code consists of S parallel LDGM codes, each with dimension k. Hence, N^2 bits are divided into S blocks, Each containing k bits, which implies $S = \frac{N^2}{k}$ (it is assumed without loss of generality that k divides N^2). The LDGM encoders add m parity check bits to each block of size k. Thus, Sk bits are encoded into $S(k+m)$ bits by the inner LDGM encoders. The rate $R_{\rm pc}$ of the outer product code is defined as $R_{\rm pc} = M^2/N^2$. Similarly, the rate R of the LDGM code is $R = \frac{k}{k+m}$. Hence, the overall rate of the concatenated code is given by $R_{cat} = R_{pc}R$.

At the output of the channel, S blocks are received which are passed through a soft decision LDGM decoder which decodes to decode k bits out of $k + m$ received soft values for each of

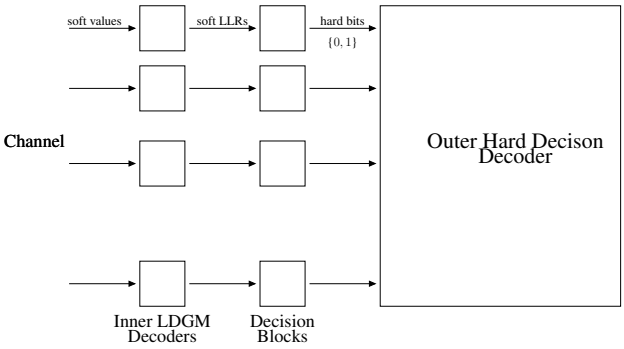


Figure 4.1: Overall Channel Model

blocks. These soft values are then classified as 0,1 or erasure. The inverse permutation π^{-1} is then applied on the $Sk = N^2$ decisions of the LDGM decoders. The resulting bits are now rearranged into a block of N^2 bits and passed through the product code decoder.

4.2 LDGM code Design

Given a product code, the inner LDGM code must decode a received codeword, to an error rate specified by the threshold probability p_{th}. p_{th} is determined by the choice of outer code. Also, the rate of inner LDGM code should satisfy the constraint $R_{\mathrm{cat}} = RR_{\mathrm{pc}}$ [1].

Progressive edge growth (PEG) algorithm is used to design the LDGM code. PEG has its advantages and disadvantages. For a given variable and check node degree distribution, PEG establishes a bipartite graph in such a way it completely avoids l-cycles (l is a parameter of choice). However, PEG takes a lot of time to establish a graph. By the name PEG, the graph is constructed edge by edge. For every edge it establishes, the graph is checked for a depth of l to avoid l cycles. Since, this steps account to preprocessing, they cannot be considered as decoder or encoder complexity.

The check node to variable LLR messages at the end of each iteration i can be modeled as a random variable Y^i which follows a symmetric Gaussian distribution $N(\mu^i, 2\mu^i)$. This can be seen as a common assumption in LDPC and LDGM codes in [8], [9], [14]. The decoding of LDGM is similar to that of the decoding of an LDPC code. From Figure 2.3, consider all circle nodes as variable nodes and square nodes as check nodes. Every step shown in the decoding of LDPC code can be followed between these variable nodes and check nodes. The exchange of messages from parity nodes to check nodes do not change over the the iterations due to a single link connected between them.

4.3 Product code Design

The outer code used for this analysis to meet the requirements of code rate and error floor are product codes. A PC is formed by the 3-error correcting (1023,993,7) BCH component code as

[1] Detailed information on the design for concatenated code is from the work on "Low Complexity soft-decision concatenated LDGM-staircase FEC for high-bit-rate fibre-optic communication" by L.M. Zhang and F.R. Kschischang

row and column codes. The code rate for this PC is $R_{\mathrm{pc}} = \frac{993^2}{1023^2} \approx 0.94$. The decoding will be carried out as explained in section 3.1.3.

Due to iterative nature of the outer decoder, introducing erasures at the input of outer decoder can be useful because of the increased size of the dominant stopping patterns it generates. This phenomenon is discussed in detail in subsequent chapter.

5. INTRODUCTION OF ERASURES

The concatenated coding scheme proposed in [1] makes a hard decision on LLRs obtained from the inner LDGM decoders and given to the outer hard decision decoder as shown in Figure 4.1. In this thesis, an enhancement to the decoder shown in [1] is proposed by introducing erasures. The erasures are introduced on the LLRs at the output of LDGM decoders. In contrast to outer decoder in [1], the outer decoder in this proposed scheme can decode combinations of erasures and hard bits (Figure 5.1). Erasures can be introduced in different ways. One approach is to introduce erasures with threshold γ on the LLRs. As discussed in the previous section, it is assumed that output LLRs at variable nodes follow symmetric Gaussian distribution. To analyze the behavior of decoder with erasures, the overall channel observed by the product code decoder is modeled as shown in figure 5.2b. Each bit in the PC is considered as a BPSK signal sent over an AWGN channel. The soft values $\underline{y}$ received from the channel at the decision block (from Figure 5.1) are

$$y_i = x_i + n_i \ \ \forall \ \ i \in \{1, 2, 3, ..., N\} \tag{5.1}$$

where $x_i \in \{\pm 1\}$ denotes the transmitted BPSK symbols and $n_i \in \mathcal{N}(0, \sigma^2)$ denotes AWGN noise

The received value y_i is mapped to either 0 or 1 or erasure producing a z_i by the decision

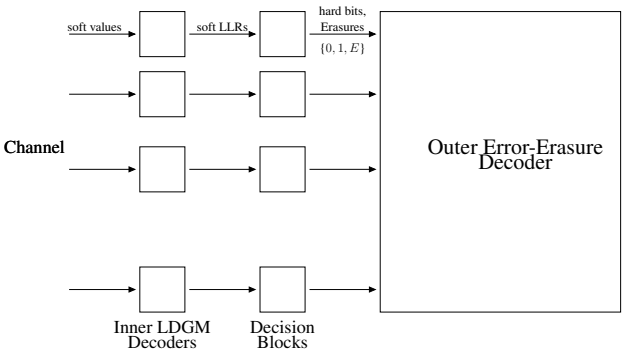
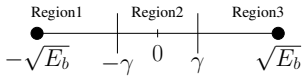
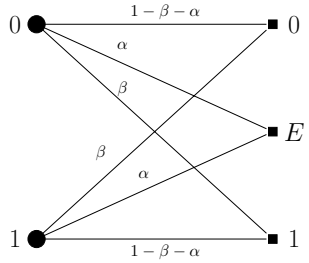


Figure 5.1: Proposed Channel Model



(a) BPSK divided into regions



(b) Channel observed by PC decoder

Figure 5.2: Figure for illustrating the channel model

block. Decision block introduces an erasure with probability α and 0 or 1 with probability $1-\alpha$. Erasures are declared when the soft value lies between $-\gamma$ and γ. as shown in figure 5.2a. The error is defined either when $+1$ lands in region 1 or -1 lands in region 3.

The analytical expression for output of the channel z_i is given by,

$$z_{\mathrm{i}} = \begin{cases} +1 & \text{if } y_i > \gamma, \\ -1 & \text{if } y_i < -\gamma, \\ \mathrm{E} & \text{else}, \end{cases} \tag{5.2}$$

where E denotes erasure.

For this model, error is declared if z_i equals ± 1 when x_i is ∓ 1. Hence, probability of error β is computed as

$$\begin{aligned} \beta &= P(z_i = 1, x_i = -1) + P(z_i = -1, x_i = 1) \\ &= P(z_i = 1 | x_i = -1) P(x_i = -1) + P(z_i = -1 | x_i = 1) P(x_i = 1) \\ &= \frac{1}{2} \left[P(z_i = 1 | x_i = -1) + P(z_i = -1 | x_i = 1) \right]. \end{aligned} \tag{5.3}$$

Similarly, erasure probability α can be computed as

$$\begin{aligned}\alpha &= P(z_i = E, x_i = -1) + P(z_i = E, x_i = 1) \\ &= P(z_i = E|x_i = -1)P(x_i = -1) + P(z_i = E|x_i = 1)P(x_i = 1) \\ &= \frac{1}{2}\left[P(z_i = E|x_i = -1) + P(z_i = E|x_i = 1)\right] \end{aligned} \tag{5.4}$$

Both equations (5.3) and (5.4) contain two terms. Since the channel shown in figure 5.1 is symmetric, both terms result in same magnitude. Hence, it is sufficient to calculate only one term. The erasure probability is calculated as below

$$\begin{aligned}\alpha &= \int_{-\gamma}^{\gamma} \mathcal{N}(-1, \sigma^2)dx \\ &= \int_{-\gamma}^{\infty} \mathcal{N}(-1, \sigma^2)dx - \int_{\gamma}^{\infty} \mathcal{N}(-1, \sigma^2)dx \\ &= Q\left(\frac{-\gamma + 1}{\sigma}\right) - Q\Big(\frac{\gamma + 1}{\sigma}\Big). \end{aligned} \tag{5.5}$$

Similarly, the error probability is calculated as

$$\begin{aligned}\beta &= \int_{\gamma}^{\infty} \mathcal{N}(-1, \sigma^2)dx \\ &= Q\left(\frac{\gamma + 1}{\sigma}\right). \end{aligned} \tag{5.6}$$

5.1 Enumeration of stopping patterns

With the error and erasure probabilities for each bit in PC are defined, it is important to define the error event in product codes. An iterative decoder cannot correct a codeword, if it encounters some pattern of errors and erasures. These patterns are termed as stopping patterns, as iterative decoder stops when these patterns are encountered. Stopping patterns are responsible for the error in a product code and key for finding the probability of codeword error. Probability of codeword

$$\begin{matrix} X & X & X & X \\ X & X & X & X \\ X & X & X & X \\ X & X & X & X \end{matrix}$$

(a) Stopping pattern 1

$$\begin{matrix} X & E & E & X & X \\ E & X & X & X & E \\ X & X & X & E & E \\ E & X & X & E & X \\ X & E & E & X & X \end{matrix}$$

(b) Stopping pattern 2

$$\begin{matrix} X & E & E & E & E & X \\ E & X & E & E & X & E \\ E & E & X & X & E & E \\ E & E & X & X & E & E \\ E & X & E & E & X & E \\ X & E & E & E & E & X \end{matrix}$$

(c) Stopping pattern 3

$$\begin{matrix} X & E & E & E & E & E & E \\ E & X & E & E & E & E & E \\ E & E & X & E & E & E & E \\ E & E & E & X & E & E & E \\ E & E & E & E & X & E & E \\ E & E & E & E & E & X & E \\ E & E & E & E & E & E & X \end{matrix}$$

(d) Stopping pattern 4

$$\begin{matrix} E & E & E & E & E & E & E & E \\ E & E & E & E & E & E & E & E \\ E & E & E & E & E & E & E & E \\ E & E & E & E & E & E & E & E \\ E & E & E & E & E & E & E & E \\ E & E & E & E & E & E & E & E \\ E & E & E & E & E & E & E & E \\ E & E & E & E & E & E & E & E \end{matrix}$$

(e) Stopping pattern 5

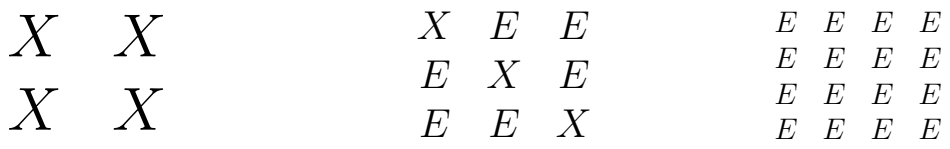
Figure 5.3: Stopping pattern for PC formed by (1023,992) component code

error is defined as the probability that a codeword is uncorrectable by the decoder. Dominant stopping patterns are patterns which considerably contribute to the probability of error. The outer product code formed by the (1023,992,8) 3-error correcting BCH code is considered to attain required code rate and error floor. Some stopping patterns for this PC is shown in Figure 5.3. In addition to patterns shown, there are other stopping patterns which contribute to error floors. Adding a row or column, or both, error-erasure combinations to the patterns shown in 5.3 will also become a stopping pattern. Any E replaced with X in any pattern is also a stopping pattern.

The other code of interest is a PC formed by a $(15, 11, 4)$ 1-error correcting BCH code. This code is considered due to its higher error floors. To show the real time performance of erasures, error floors should be simulated. Simulating error floors for $(1023^2, 992^2, 8^2)$ PC is very time and resource consuming, while error floors for $(15^2, 11^2, 4^2)$ is can be simulated with reasonable hardware. Some of the stopping patterns for this code are shown in Figure 6.3. Any E replaced with X in 5.4 is also a stopping pattern and is demonstrated in the mathematical modeling.

5.2 Correcting capability of product code decoder

With the complete channel modeled, errors and erasures are introduced in a product code with probabilities β and α respectively. The bounding distance for the combination of erasures and



(a) pattern 1 (b) pattern2 (c) pattern3

Figure 5.4: Stopping pattern for PC formed by (15,11) component code

errors is stated in the theorem below. Bounding distance is defined as the combination of number of errors and erasures that are correctable by the decoder.

Theorem 5.2.1 (Bounding Distance of PC)**.** *For a product code $C(N, K, D)$ formed by column component code $C_1(n_1, k_1, d_1)$ and row component code $C_2(n_2, k_2, d_2)$, any pattern of x errors and e erasures are correctable if $4x + e < d_1 d_2$ with iterative cascade row and column decoding.*

Proof. Consider a product code $C(N, K, D)$ formed by column component code $C_1(n_1, k_1, d_1)$ and row component code $C_2(n_2, k_2, d_2)$. The number of errors in a PC provides a lower bound on the number of erasures required to be a stopping pattern. This will provide a lower bound on the dimensions of the stopping pattern i.e. number of erroneous rows and columns. For any x number of errors in the product code, it is sufficient to prove the lower bound on the number of erasures required to be a stopping pattern should satisfy $4x + e \geq d_1 d_2$. Through out this proof, considering an example PC Q formed by row and column codes with minimum distances 6 and 8 respectively. To obtain the lower bound on the number of erasures, the smallest possible stopping pattern should be determined. For any x number of errors in the pattern, there exists a largest possible q and k ($q, k \geq 0$) such that each erroneous row and column contains at least q and k errors respectively (Refer to the example pattern in Figure 5.5a of Q with $q = 1$ and $k = 1$). There exists at least one row with exactly q errors. That row cannot by corrected by C_2 if and only if at least $d_2 - 2q$ erasures exist. At least one row contains q errors and $d_2 - 2q$ erasure, making the length of the stopping pattern to be $d_2 - q$. Similarly, there exists at least one column with exactly k errors. That column cannot be corrected by C_1 if and only if at least $d_1 - 2k$ erasures exist. At least one column contains k errors and $d_1 - 2k$ erasures, making the height of the stopping pattern to be $d_1 - k$. The

$$\begin{matrix} X & E & E & E & E \\ E & X & E & E & E \\ E & E & X & E & E \\ E & E & E & X & E \\ E & E & E & E & X \\ E & X & E & E & E \\ X & E & E & E & E \end{matrix}$$

(a)

$$\begin{matrix} X & E & E & E & E \\ E & X & E & E & E \\ E & E & X & E & E \\ E & E & E & X & E \\ E & E & E & E & X \\ E & X & X & D & E \\ X & E & D & X & E \end{matrix}$$

(b)

Figure 5.5: Stopping patterns for a code Q with minimum distances of column and row codes are $d_1 = 8$ and $d_2 = 6$ respectively. In both the patterns, X represents error, E represents erasure and D represents don't care (a) Illustrating a stopping pattern with $x = 7$, $q = 1$ and $k = 1$ (b) Illustrating a stopping pattern with $x = 9$, $q = 1$ and $k = 1$

size of the resultant stopping pattern will be $(d_1 - k) \times (d_2 - q)$. As, at least q errors are present in each row of length $(d_2 - q)$,

$$\begin{aligned} d_2 - q &\geq q \\ d_2 &\geq 2q \end{aligned} \tag{5.7}$$

As the total number of errors equal x, any row or column may contain more than q and k errors. The total number of such errors equals $x - q(d_1 - k)$ or $x - k(d_2 - q)$. These two quantities need not be equal.

To obtain the minimum number of erasures required, the maximum number of don't cares(Ds) present in the stopping pattern should be determined. Don't cares are positions which can be either an error or an erasure or a correct bit. To prove the theorem, the least number of errors and erasures are desired, consider the don't cares to be positions of correct bits in the stopping pattern without loss of generality.

If observed by row, $q(d_1 - k)$ errors are fixed in position. So, it is necessary to arrange the remaining $x - q(d_1 - q)$ errors in the pattern. Say, any a^{th} ($a \in \{1, .., d_1 - k\}$) row contains $q + i_a (i_a \geq 0)$ errors. The maximum number of don't cares that can be introduced in a row with

$q+i_a$ errors will be i_a without altering the size of the stopping pattern. Thus, a row can have $q+i_a$ errors, i_a don't cares and $d_2 - 2(q+i_a)$ erasures (Refer 6^{th} and 7^{th} row in Figure 5.5b of Q with $d_1 = 8$, $d_2 = 6$, $x = 9$, $q = 1$ and $k = 1$). This can not be corrected by the row component code. $\sum_{a=1}^{d_1-k} i_a$ equals the extra errors which is equal to $x - q(d_1 - k)$. This is also equal maximum number of don't cares possible, if seen by row.

Similarly, if observed column wise, the maximum number of don't cares possible are $x-k(d_2-q)$, if seen column wise.

Thus the maximum number of don't cares possible in a product code is min $\{x-q(d_1-k), x-k(d_2-q\}$ (where min$\{.\}$ indicates minimum function). To proceed with the proof, without loss of generality assume $x-(d_1-k)q$ is minimum (Refer to the example pattern in Figure 5.5b), which implies

$$\begin{aligned} x - q(d_1 - k) &\leq x - k(d_2 - q) \\ \implies q(d_1 - k) &\geq k(d_2 - q) \end{aligned} \tag{5.8}$$

Total number of errors(x) = x

Maximum number of don't cares possible(D) = $x - q(d_1 - k)$

Total number of erasures(e) present in the stopping pattern is given by

$$\begin{aligned} e &\geq (d_1 - k)(d_2 - q) - x - (x - q(d_1 - k)) \\ &= (d_1 - k)d_2 - 2x \end{aligned} \tag{5.9}$$

Consider the equation $4x + e$ with above x and e.

$$
\begin{aligned}
4x + e &= 4x + (d_1 - k)d_2 - 2x(from\ (5.9)) \\
&= 2x + (d_1 - k)d_2
\end{aligned} \tag{5.10}
$$

As $x - q(d_1 - k)$ cannot be negative, it is easy to see

$$
x \geq q(d_1 - k) \tag{5.11}
$$

Substituting (5.11) in (5.10) yields

$$
\begin{aligned}
4x + e &\geq 2q(d_1 - k) + d_2(d_1 - k) \\
&\geq 2k(d_2 - q) + d_2(d_1 - k)\ \ (from\ (5.8)) \\
&= d_1 d_2 + k(d_2 - 2q) \\
&\geq d_1 d_2\ \ (from\ (5.7))
\end{aligned} \tag{5.12}
$$

□

5.3 Analytical Model for Error Floors

The error floor for the $(15^2, 11^2, 4^2)$ PC is calculated with known probabilities. This model assumes a genie decoder which contains component decoders that are capable of not performing miscorrections. Miscorrection occurs when a decoder wrongly corrects a erroneous codeword. Real time decoders can miscorrect a erroneous codeword, while the genie decoder (ideal) provides a decoding failure in such a case.

Each pattern k ($k \in \{1, 2, 3\}$ for this product code from figure 5.4) occurs with the probability p_k. The equations quoted below gives the probability of occurrence of each stopping pattern.

$$P_1 = \binom{15}{2} \times \binom{15}{2} \times \beta^4 \tag{5.13}$$

$$P_2 = \binom{15}{3} \times \binom{15}{3} \times N_{\mathrm{c}} \times \beta^3 \times (\alpha+\beta)^6 \tag{5.14}$$

$$P_3 = \binom{15}{4} \times \binom{15}{4} \times (\alpha+\beta)^{16} \tag{5.15}$$

Similar to the above, the probability for each dominant stopping pattern in product code formed by (1023,992,8) BCH component code shown in figure 5.3 is derived as,

$$P_{\mathrm{k}} = \binom{1023}{b} \times \binom{1023}{b} \times N_{\mathrm{c}} \times \beta^{\mathrm{l}} \times (\alpha+\beta)^{\mathrm{m}} \tag{5.16}$$

where k indicates pattern number, N_{c} indicates the number of combinations possible once positions are fixed, l and m indicates the number of errors and number of erasures respectively in each pattern. The values of these parameters are provided in table 5.1 for each pattern.

k	b	N_{c}	l	m
1	4	1	16	0
2	5	2040	15	10
3	6	67950	12	24
4	7	7!	7	42
5	8	1	0	64

Table 5.1: Table for the mathematical values of error probability for $(1023^2, 992^2, 8^2)$.

The values of N_{c} for k= 2 and 3 (Table 5.1) is determined by exhaustively running all the combinations of b $\times$ b matrix with l errors and m erasures.

All the above equations are the functions of σ and threshold γ. The mathematical approximation for the error is chosen as,

$$P_e \approx \max_k P_k, \tag{5.17}$$

where $k \in \{1, 2, 3, ...\}$ represents the index of a stopping pattern among all possible stopping patterns. Stopping patterns with greater probability of occurrence are considered to be dominant stopping patterns as shown in figures 5.4 and 5.3, but not limited to them. Equation 5.17 considers all possible stopping patterns for a given product code.

5.4 Optimal threshold

With many different choices of thresholds to declare erasure, the question remains to find the optimal threshold. The closed form expression for the optimal threshold cannot be realized with this model in section 5.3, but it can be computed numerically when all the stopping patterns are considered. The expression for optimal threshold is given by,

$$\text{Optimal Threshold } (\gamma) = \min_{\gamma} \max_k P_k(\sigma, \gamma) \tag{5.18}$$

6.1 Comparison between Product and Half Product codes

Emmadi has discussed some results in his thesis [2] about comparison between PC and HPC. It states that PC perform better than HPC if codeword length and error correcting capabilities are large.This was shown using a HPC of length 1023 and error correcting capability 26. But, HPC performs better than PC for relatively smaller lengths and error correcting capabilities. This was illustrated using a HPC of length 255 and error correcting capability 6. As discussed earlier the product code which can meet the requirements of both code rate (≈ 0.9) and error floor ($< 10^{-15}$) is formed by a 3 error correcting $(1023, 993, 7)$ BCH component code. This code doesn't fall into any of the above discussed categories, which makes it essential to compare the performances in this case. The equivalent HPC will be formed by a 4 error correcting $(1446, 1402, 9)$ BCH code.

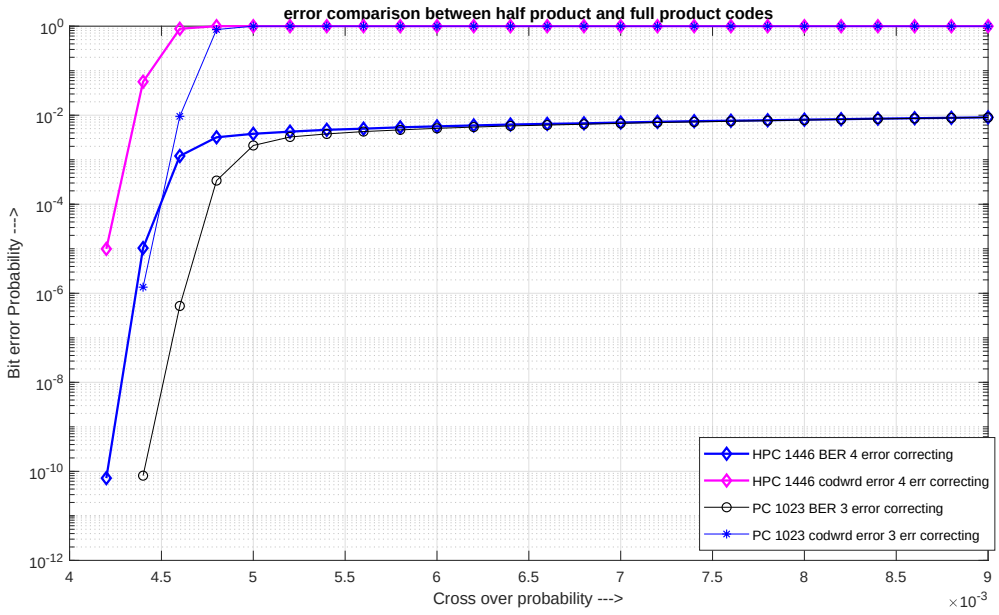


Figure 6.1: Comparison plot between HPC and PC

As limited by hardware, the simulations are presented until the bit error rate of $\approx 10^{-10}$. The codeword error rate is defined as the ratio of uncorrected codewords to the total number of codewords sent, while bit error rate is defined as the ratio of total number bits in error to total number of bits sent over the channel. X-axis denotes the cross over probability in a standard binary symmetric channel(BSC). Y-axis denote error rate probability, we can see both codeword error rate and bit error rate in the same graph. For an AWGN channel $\mathcal{N}(0, \sigma^2)$, the crossover probability p is defined as $Q\left(\frac{1}{sigma}\right)$.

6.1.1 Observations

It is clearly seen that product code performs better than the half product code in the waterfall region. It can be seen that the slope of HPC is decaying faster than that of PC. As we know the error floor for this PC and HPC is lower than 10^{-15}, though HPC may cross PC somewhere at a lower error rate, but not at rates higher than 10^{-15}. This can be concluded with a naive extension using the current trend of the slope. Base on this, we propose PC as the choice of outer code. This graphs also explains the p_{th} required for the outer code to perform in the error rate of below 10^{-15}. With the slope, p_{th} can be approximated as 4×10^{-3} for PC.

6.2 Results based on erasures

6.2.1 Simulations of Product code with (15,11) BCH codes as component codes

Consider the channer model shown in figure 5.2. Erasures are declared if the received value from the channel falls in region 2. Introduction of erasures in a BCH code degrades the performance. This is illustrated in figure 6.2 with the example of a 3-error correcting (1023,993,7) BCH code. Threshold $\gamma = 0$ indicates error only scenario, while other thresholds provides the combination of errors and erasures with probabilities computed as in equations 5.6 and 5.5.

Introduction of erasures in PC and HPC with iterative decoders improves the error rate performance. A product code P formed by a $C(15, 11, 4)$ BCH code is used as an example to illustrate the effect of erasures. The actual code of interest is a PC formed by a (1023,992,8) BCH component code. P is almost similar to this kind of frame work and it is chosen because of its high error

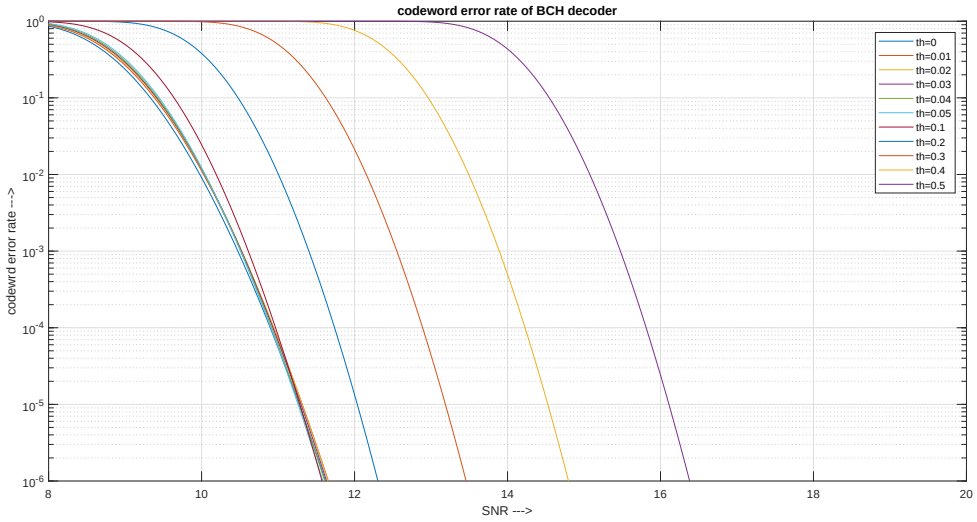


Figure 6.2: Performance under erasures with BCH decoder of(1023,993)

floor and lower computational complexity. The mathematical approximation for error probabilities are calculated according to equation 5.17.

Figure 6.3 shows graphs for the simulated codeword error rate for P along with the approximations derived in section 5.3. In Figure 6.3, dashed lines indicates the mathematical approximation for codeword error rate, while solid lines indicates the corresponding simulated results. From the observations, erasures introduced with various thresholds are performing better than error only ($\gamma = 0$) scenario. The simulated codeword error rates less than 10^{-6} are not accurate due to the computational limit (number of codewords considered is 10^7) in the simulations. The gap between the simulated and mathematical approximation is very minimal. This gap occurs due to miscorrections and also when more than one stopping pattern occurs dominantly. This shows the approximation is reasonable and can be used as a measure for calculating codeword error rate probability.

6.2.2 Simulations of Product code with (1023,992,8) BCH code as component codes

Discussion in the above section concludes that introduction of erasures in PC provides large coding gains at the error floors. Mathematical approximations derived for codeword error prob-

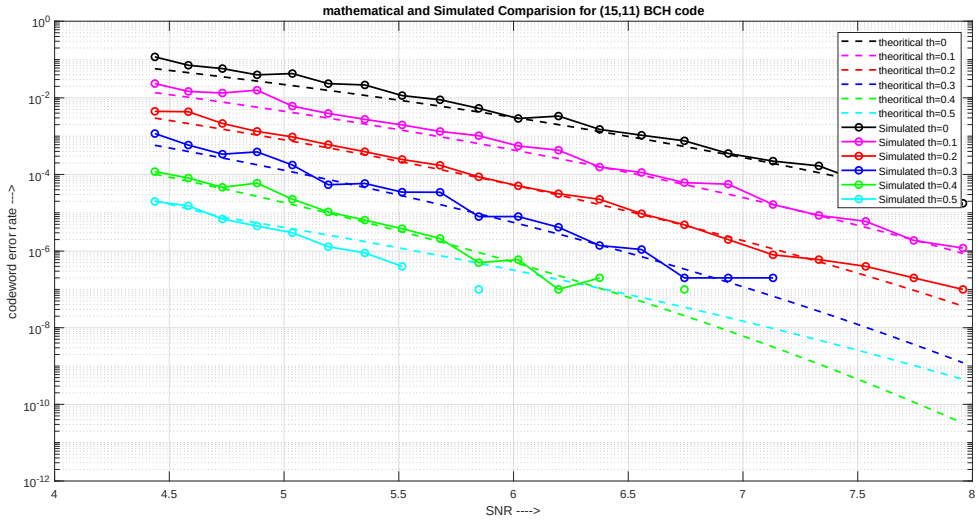


Figure 6.3: Performance of PC formed by (15,11)with erasures

abilities are close approximations only at error floors. Consider a PC formed by a (1023,992,8) BCH code. The mathematical approximation for codeword error rate is shown in Figure 6.4.

The codeword error rate performance improves until the threshold γ is increased to ≈ 0.4, then it gradually starts deteriorating for any increment in threshold. The error floor for this codes with the advent of erasures are way lower than the requirement. In the waterfall region (not error floor), real curves wont follow this trend, as iterative decoder fails in the primary step due to too many erasures and errors at lower SNRs. The error floors cannot be simulated with the available hardware resources. The simulations for the PC with (1023,992,8) are presented in Figure 6.5 only when threshold $\gamma = 0.035$, which seemed optimal upon various choices for the water fall region against error only$\gamma = 0$ scenario. The curves for higher thresholds have steeper slopes as SNR increases.

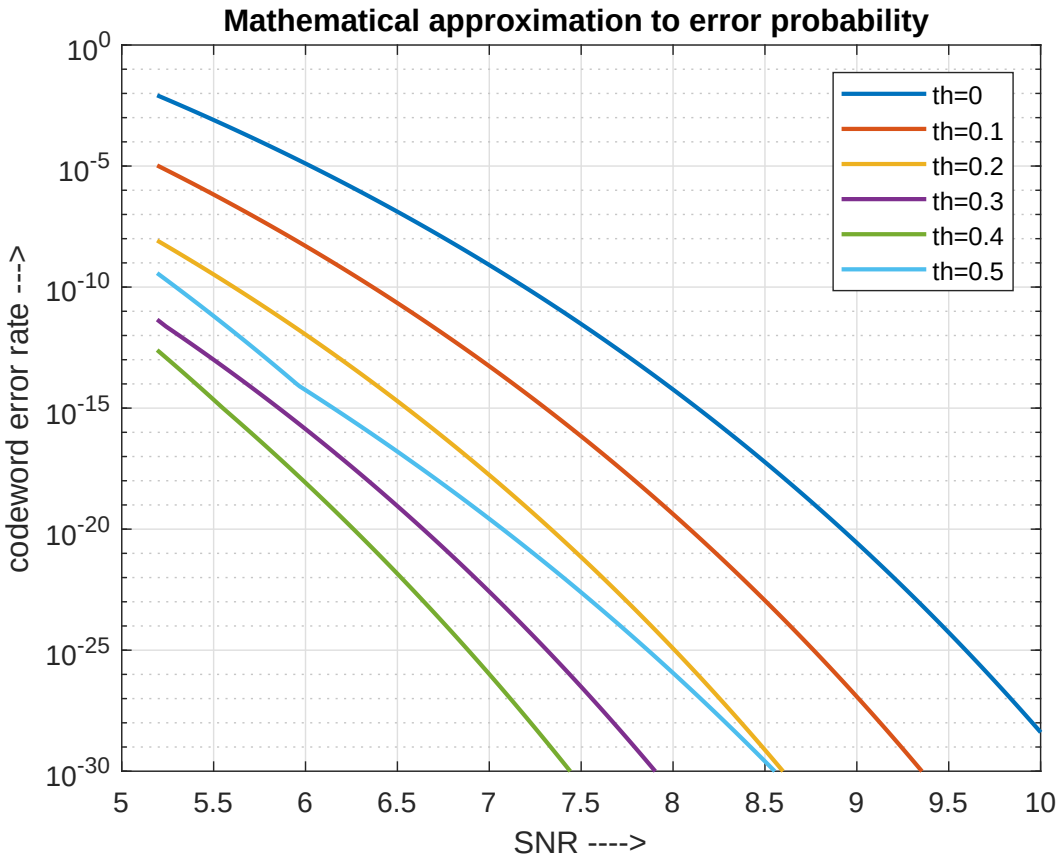


Figure 6.4: Mathematical approximation of PC formed by (1023,993) with erasures

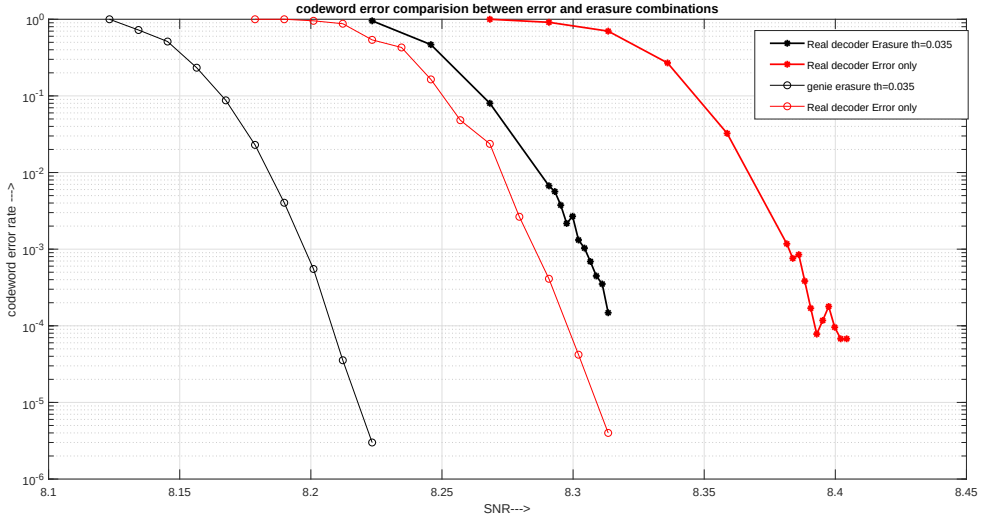


Figure 6.5: Actual plot of PC formed by (1023,993) with error and erasures case

7. CONCLUSIONS

Bounded distance for the product code is proposed i.e, a codeword from the product code formed by $C_1(n_1, k_1, d_1)$ and $C_2(n_2, k_2, d_2)$ as component codes with x errors and e erasures can be corrected if it follows $4x + e < d_1 d_2$. It is concluded that introduction of erasures attains coding gain. Erasures are introduced with threshold γ from channel model shown in figure 5.2. Equation 5.18 provides the optimal threshold to chose at a particular SNR. There are many other ways of choosing the threshold. One way is to check for the LLR value at different iterations on left nodes, if the sign toggles, that node can be declared an erasure. One other way can be based on the messages incident on variable node. if the signs of them vary during iterations, we can declare it as an erasure. These optimizations can be applied and the analysis provided in this work hold true for any method of erasure introduction and define some erasure probability (α) and error probaility (β). Mathematical approximations for codeword error rates at error floors are derived.

REFERENCES

[1] L. M. Zhang and F. R. Kschischang, "Low-complexity soft-decision concatenated ldgm-staircase fec for high-bit-rate fiber-optic communication," *Journal of Lightwave Technology*, vol. 35, no. 18, pp. 3991–3999, 2017.

[2] S. K. Emmadi, "Half-product codes," 2014.

[3] F. Chang, K. Onohara, and T. Mizuochi, "Forward error correction for 100 g transport networks," *IEEE Communications Magazine*, vol. 48, no. 3, 2010.

[4] R. Gallager, "Low-density parity-check codes," *IRE Transactions on information theory*, vol. 8, no. 1, pp. 21–28, 1962.

[5] W. Ryan and S. Lin, *Channel codes: classical and modern*. Cambridge University Press, 2009.

[6] M. Luby, "Blt codes,[in proc. 43rd annu. ieee symp. foundations of computer science (focs)," *Vancouver, Canada*, pp. 271–280, 2002.

[7] R. Palanki and J. S. Yedidia, "Rateless codes on noisy channels," in *Information theory, 2004. ISIT 2004. Proceedings. International symposium on*, p. 37, IEEE, 2004.

[8] O. Etesami and A. Shokrollahi, "Raptor codes on binary memoryless symmetric channels," *IEEE Transactions on Information Theory*, vol. 52, no. 5, pp. 2033–2051, 2006.

[9] Z. Cheng, J. Castura, and Y. Mao, "On the design of raptor codes for binary-input gaussian channels," *IEEE Transactions on Communications*, vol. 57, no. 11, 2009.

[10] D. A. Spielman, "Linear-time encodable and decodable error-correcting codes," *IEEE Transactions on Information Theory*, vol. 42, no. 6, pp. 1723–1731, 1996.

[11] S. B. Wicker and S. Kim, *Fundamentals of codes, graphs, and iterative decoding*, vol. 714. Springer Science & Business Media, 2002.

[12] H. D. Pfister, S. K. Emmadi, and K. Narayanan, "Symmetric product codes," in *Information Theory and Applications Workshop (ITA), 2015*, pp. 282–290, IEEE, 2015.

[13] J. Justesen, "Performance of product codes and related structures with iterated decoding," *IEEE Transactions on Communications*, vol. 59, no. 2, pp. 407–415, 2011.

[14] S. Ten Brink, G. Kramer, and A. Ashikhmin, "Design of low-density parity-check codes for modulation and detection," *IEEE Transactions on Communications*, vol. 52, no. 4, pp. 670–678, 2004.