ITERATIVE DECODING OF CODES ON GRAPHS

by

Sundararajan Sankaranarayanan

A Dissertation Submitted to the Faculty of the

DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING

In Partial Fulfillment of the Requirements

For the Degree of

DOCTOR OF PHILOSOPHY

In the Graduate College

THE UNIVERSITY OF ARIZONA

2 0 0 6

THE UNIVERSITY OF ARIZONA

GRADUATE COLLEGE

As members of the Dissertation Committee, we certify that we have read the dissertation prepared by *Sundararajan Sankaranarayanan* entitled *Iterative Decoding of Codes on Graphs* and recommend that it be accepted as fulfulling the dissertation requirement for the Degree of *Doctor of Philosophy*

Prof. *Bane Vasić* Date: *05/12/2006*

Prof. *Michael W. Marcellin* Date: *05/12/2006*

Prof. *William Ryan* Date: *05/12/2006*

Final approval and acceptance of this dissertation is contingent upon the candidate's submission of the final copies of the dissertation to the Graduate College.

I hereby certify that I have read this dissertation prepared under my direction and recommend that it be accepted as fulfilling the dissertation requirement.

Prof. *Bane Vasić* Date: *05/12/2006*

STATEMENT BY AUTHOR

This dissertation has been submitted in partial fulfillment of requirements for an advanced degree at The University of Arizona and is deposited in the University Library to be made available to borrowers under rules of the Library.

Brief quotations from this dissertation are allowable without special permission, provided that accurate acknowledgment of source is made. Requests for permission for extended quotation from or reproduction of this manuscript in whole or in part may be granted by the head of the major department or the Dean of the Graduate College when in his or her judgment the proposed use of the material is in the interests of scholarship. In all other instances, however, permission must be obtained from the author.

SIGNED: Sundararajan Sankaranarayanan

ACKNOWLEDGEMENTS

Is it that time-the time to write an acknowledgment-already ? Does this mean the end of heated debates, pointless discussions, comical exchanges, reminiscence of "glory" days, long lunch breaks, and endless trips to coffee shops ? It is surprising that I found time to get some work done, perhaps a debatable conclusion for my advisor.

It was a great pleasure and honor to work with my supervisor, Dr. Bane Vasić. His technical grasp in areas ranging from communications theory, combinatorics, physics to DNA coding helped to create stimulating learning experiences. His approach to guiding guaranteed appreciable freedom in my pursuit of research topics and solutions. This approach had created great many learning experiences, especially those stemming out of my own mistakes. He leads by example in inculcating values of hard work and punctuality.

Prof. Jeffrey J. Rodríguez and Prof. Jo Dale Carothers have been more than just my teachers, they have been excellent guides and guardians. I am grateful for their gracious help during my troubled times. I am thankful for the excellent bunch of teachers including Dr. Marcellin, Dr. Ryan, Dr. Holman, Dr. Lux, Dr. Kennedy, Dr. Vasić and others. Also, thanks to Ivan Djordjević for making a lot of my publications happen. Finally, thanks to Tami Whelan for making life in the department a whole lot easier.

I have been in Tucson and in the department long enough to have witnessed a good number of batches of students come and go. I need to thank quite a lot of them, my friends, who made my life (some would call it as a figment of my imagination) within and outside of the University a great deal more fun. I would cherish all the great times from surfing restaurants, late night bowling, snacks at IHOP to 2AM discussions with Barath (*"voohoo"*) and Jai (*"ennanga"*) (my roommates), Vijay (*"ch..ennada"*), Sudarshan (*"SAM"*), and Yaseen (*"Bhai"*). The lively bunch of lab mates Karun, Varsha, Ali-T, Shashi (*Good to be alive in 1985!*), Hari, Rathna (let's continue bouncing business ideas...millionaires we will be), Andy ("feel pannarein") and other friends Jayanth, Neema, Deepthi, Divya, Shravan (*"Sharvan"*) made life all the more exciting.

Finally but yet most importantly, I attribute all successes and good times of my life to the love and sacrifices of my amma and appa. They have been the pillars of support behind every one of my achievements. Also, a great deal of credit should go to Gansi and my family.

DEDICATION

To amma, appa, Gansi, family, and friends.

TABLE OF CONTENTS

TABLE OF CONTENTS – *Continued*

TABLE OF CONTENTS – *Continued*

TABLE OF CONTENTS – *Continued*

LIST OF FIGURES

LIST OF FIGURES – *Continued*

LIST OF FIGURES – *Continued*

LIST OF TABLES

ABSTRACT

The growing popularity of a class of linear block codes called the low-density parity-check (LDPC) codes can be attributed to the low complexity of the iterative decoders, and their potential to achieve performance very close to the Shannon capacity. This makes them an attractive candidate for ECC applications in communication systems. This report proposes methods to systematically construct regular and irregular LDPC codes.A class of regular LDPC codes are constructed from incidence structures in finite geometries like projective geometry and affine geometry. A class of irregular LDPC codes are constructed by systematically splitting blocks of balanced incomplete block designs to achieve desired weight distributions. These codes are decoded iteratively using message-passing algorithms, and the performance of these codes for various channels are presented in this report.

The application of iterative decoders is generally limited to a class of codes whose graph representations are free of small cycles. Unfortunately, the large class of conventional algebraic codes, like RS codes, has several four cycles in their graph representations. This report proposes an algorithm that aims to alleviate this drawback by constructing an equivalent graph representation that is free of four cycles. It is theoretically shown that the four-cycle free representation is better suited to iterative erasure decoding than the conventional representation. Also, the new representation is exploited to realize, with limited success, iterative decoding of Reed-Solomon codes over the additive white Gaussian noise channel.

Wiberg, Forney, Richardson, Koetter, and Vontobel have made significant contribu-

tions in developing theoretical frameworks that facilitate finite length analysis of codes. With an exception of Richardson's, most of the other frameworks are much suited for the analysis of short codes. In this report, we further the understanding of the failures in iterative decoders for the binary symmetric channel. The failures of the decoder are classified into two categories by defining trapping sets and propagating sets. Such a classification leads to a successful estimation of the performance of codes under the Gallager B decoder. Especially, the estimation techniques show great promise in the high signal-to-noise ratio regime where the simulation techniques are less feasible.

CHAPTER 1

INTRODUCTION

Linear block codes have been a subject of interest since the discovery of Hamming codes [47] in the late 1940s. Hamming's theory encouraged focus on construction of error correcting codes with good properties like high minimum distance and favorable weight-spectrum. One of the most popular class of algebraic codes is the Reed-Solomon codes, and these codes belong to the class of maximum distance separable codes. In 1948, Shannon [86] added an important dimension to the problem of code construction by establishing the capacity limits and the existence of codes that achieve this upper bound. Sudan's tutorial [90] on coding theory is a good read.

Reed-Solomon codes are applied in error correction and control mechanisms of several communication systems. For example, Reed-Solomon codes are used in magnetic disk-drive systems, magnetic tape-drive systems and optical communication systems. The maximum-likelihood decoders of algebraic codes are computationally expensive, and most of the sub-optimal decoders (of algebraic codes) are bounded-distance decoders at best. In magnetic recording applications, Reed-Solomon codes are decoded using sub-optimal hard-decision decoders. In Chapter 3, we present details of the coding subsystem of the magnetic tape systems. The tape systems use Reed-Solomon codes arranged in product form [33] for burst-error correction purposes. Later in this chapter, we develop an analytical method to compute the performance of the Reed-Solomon codes on a simplified tape channel model

As mentioned earlier, only sub-optimal decoders are applied in practice due to the exponential complexity of optimal decoders. This means that the "goodness" of codes

cannot be translated into excellent bit-error rate or block-error rate performances. The work of Gallager [37] on low-density parity-check (LDPC) codes is one of the earliest approaches that sacrificed code optimality for improvement in decoder efficiency. This is a shift in ideology towards constructing codes that facilitate efficient decoding algorithms that achieve the Shannon limit or perform very close to the limit. The probabilistic decoder proposed in [37] performs optimal (in maximum *a posteriori* probability sense) decoding at each one of the *constraint nodes*. In other words, the algorithm is designed to preserve optimality locally. Gallager established that the optimality of the decoder (or the global optimality) is guaranteed if and only if the messages processed at constraint nodes are independent. Also, he demonstrated that very good performance is achieved by iterative application of his decoding algorithm even in a setting that lacks independence of messages. This principle of Gallager's decoder is essentially different from that of other algebraic decoders.

In [92], Tanner generalized the idea in his proposal for a recursive approach to low complexity codes. He introduced the notion of equivalence between the construction of codes and graphs. Tanner defined a code as a bipartite graph, referred to as Tanner graph, with two sets of nodes, namely *digit nodes* and *subcode nodes*, and a set of permutation matrices. The relevance of such an equivalence is established by the three decoding algorithms proposed in the paper. He used girth, length of the shortest cycle of a graph, of Tanner graphs to lower bound the minimum distance property of codes. This initiated further investigations into the graph theoretic aspects of iterative decoding of LDPC codes. Tanner's idea of subcode nodes is a generalization of constraint nodes because these nodes are representative of subcodes that are not necessarily single parity-check codes. The probabilistic decoding algorithm proposed in [92] is an extension of Gallager's decoding algorithm that preserves decoding optimality at each subcode nodes.

MacKay is credited, rightfully so, with the revival of Gallager's LDPC codes and iter-

ative decoders through his seminal paper [58], [59]. He proved that random LDPC codes, when decoded optimally, are good codes for channels with stationary and ergodic noise models. In addition, he showed that LDPC codes decoded with the *sum-product algorithm* (SPA) achieved performance close to that of turbo codes. In fact, the SPA works beyond the minimum distance of the code. It is an exciting result because LDPC code is the only class of good codes that has an efficient polynomial time decoder. Perhaps, the most interesting contribution of his work is that it establishes the universality of the SPA. He established that several flavors of the SPA exist in the areas of expert systems and machine learning, cryptanalysis, and statistical physics. Decades after Gallager's work, *belief propagation algorithms* [67] and other similar algorithms were developed in the field of expert systems to compute marginals of functions using a graphical model. The *free energy minimization* algorithm used in the field of statistical physics belongs to the same category as the SPA. The universality of the SPA is a compelling argument to invest more effort in developing techniques to explain various aspects of the algorithm.

In his paper, MacKay proposed several methods to construct pseudo-random LDPC codes that can be expected to perform well in conjunction with iterative decoders. In addition, he developed simple rules-of-thumb to construct graphs that are better suited for the SPA. These rules include the avoidance of short cycles of length 4 and 6, and other topological structures illustrated in the paper. With further simple analysis, we realize that some of the advanced topological structures are a result of specific interactions of simple six and eight cycles. Although he chooses to avoid these structures from his codes, no further graph theoretic analysis is presented in the paper.

In Chapter 4, we propose methods to systematically construct regular and irregular codes from finite geometries and combinatorial designs. The Tanner graphs of their representations do not have cycles of length 4, and at least, there exist bounds on minimum distances of these regular codes. The parity-check matrices of these codes have

favorable structural properties that can be exploited to design low-complexity encoders and decoders. The applications of these codes in high-speed long-haul optical transmission systems have shown excellent performance as shown in [64], [30], [**?**], [26], [27], [28], [80]. In this chapter, we present the performance of these codes over the AWGN channel and the magnetic recording channel. These codes are decoded iteratively using the sum-product algorithm.

In [92], Tanner founded the general study of codes on graphs, introduced generalized constraints, and proved the optimality of the sum-product algorithm on cycle-free graphs. The next development in the study of codes on graphs was achieved by introducing *state variables* into Tanner graphs (TGs) [107], [106], and this allowed to establish connections to trellises and to turbo codes. In [63] , [13], and [41], the authors have presented important results on minimal trellis-like representations of linear block codes that minimize one or more complexity measures. In [56], Kschischang *et al.* further abstracted the idea of codes on graphs to encompass representation of general functions, and such graphs are referred to as *factor graphs*.

The state variables in TGs, introduced by Wiberg [107], are auxiliary variables that are introduced by the designer to improve code realization in some sense. These variables are not transmitted over the channel, and hence, they are not observed at the receiver-end. In [58], MacKay discussed *generalized parity-check matrix* (GPCM) representations of linear block codes obtained by introducing additional columns, and additional rows. These additional columns corresponding to auxiliary variables can be interpreted as state variables because they are not transmitted over the channel. In [112], Yedidia *et al.* presented an algorithm to generate GPCM representations of codes that have the following characteristics: small number of ones in each row, large number of ones in each column and a small number of four-cycles. With empirical results, they demonstrated an improvement in bit-error rate performance of codes, transmitted over the binary erasure

channel or the additive white Gaussian noise channel, under message-passing algorithms applied on these new representations. Yedidia *et al.* had posed an interesting problem of finding a GPCM of a code that is suitable for message-passing algorithms.

In Chapter 5, we present an algorithm to generate GPCM representations of codes by removing four-cycles, if any, in their conventional (original) representations. Most of the algebraic codes have lots of four cycles in their graph representation. As mentioned earlier, these four cycles are removed by adding auxiliary variables and auxiliary checks to the original representations. For any code, we prove that iterative erasure decoding over the GPCM representation achieves a performance that is at least as good as that achieved with the original representation. We extend this idea to the iterative decoders for the additive white Gaussian noise channel. The performance results of Reed-Solomon codes and Hamming codes obtained from simulation techniques show that the benefits of GPCM representations are limited.

The sum-product algorithm was generalized by Kschischang et al. [56] as a marginalization algorithm for factor graphs. This allowed them to establish connection with algorithms in various other fields like Bayesian networks, Markov random fields, Pearl's belief propagation algorithm, fast Fourier transform etc. Many of these algorithms are merely special instances of the sum-product algorithm. Hence, a deeper understanding of the algorithm will have a widespread impact in areas of expert systems, statistical mechanics, digital signal processing etc. In fact, the study of the algorithm has attracted interest of researchers from the above mentioned fields and others. In [74] and [75], authors show that irregular LDPC codes (in the limit of their length tending to infinity) can achieve capacity when decoded iteratively using the sum-product algorithm. A complete understanding of the iterative erasure decoder is achieved with the help of a combinatorial object called the stopping set. In [24], Di *et al.* characterized the performance of LDPC code ensemble over the binary erasure channel using this object.

But a similar characterization of the SPA still remains an open problem especially in the realm of finite length codes. Regarding a complete understanding of the SPA, in [59], MacKay states that "an explicit positive statement of the sum-product decoder's capabilities remains elusive".

In Chapter 6, we develop an understanding of the causes of decoding failures of the simplest message-passing algorithm called the Gallager B algorithm. Although the goal of this study is modest, especially, compared to the goal of analytical characterization of the performance of the SPA, we believe it is a logical first step. The failures of the Gallager B decoder are classified based on two sets: trapping sets and propagating sets. The idea of trapping and propagating sets are similar to the object introduced by Richardson in [73]. In [73], Richardson aimed to semi-analytically estimate the performance of a code under the SPA-based decoder using failure sets called trapping sets. This chapter studies the dynamics of the Gallager B decoder in detail and characterizes input patterns that drive the decoder to failure. In addition, this knowledge is applied in semi-analytically computing the performance of special class of LDPC codes. Finally, Chapter 7 summarizes the important aspects of the report and suggests avenues for further investigations.

CHAPTER 2

PRELIMINARIES

This chapter presents a brief description of several important components of a communication system. The channel codes of interest are linear block codes like Reed-Solomon codes, product codes, and low-density parity-check codes. Simple channel models like the binary erasure channel, the binary symmetric channel, and the additive white Gaussian noise channel are very useful to achieve analytical/empirical characterization of the performance of channel codes. In addition, we are interested in other channel models that describe magnetic hard-disk and tape mediums. Detectors and decoders are important components of communication systems. For the most part of this dissertation, we are interested in decoders that are based on iterative decoding algorithms. These decoders work by sending messages over edges of a graphical representation of the code.

2.1 Linear Block Codes

Linear block codes belong to a subclass of channel codes that has received the most attention since the invention of Hamming code [47]. As the name suggests, a linear block code $\mathcal{C}$ is a k dimensional subspace of an n dimensional linear vector space over a finite field $\mathbb{F}$, where $k \leq n$. An encoder of $\mathcal{C}$ is a linear map of dimension k from $\mathbb{F}^k$ to $\mathbb{F}^n$.

2.1.1 Reed-Solomon Code

An (n, k, d) Reed-Solomon (RS) [72] code over the Galois field of order q, denoted as GF($q = p^m$), is a nonbinary cyclic code of length $n = q - 1$ and minimum distance d.

The code belongs to the class of *maximum distance separable* code that achieves the Hamming bound and hence, the dimension k of the code is $k = n - d + 1$. If α is a primitive element of GF(q), the generator polynomial of the cyclic code is defined as

$$g(x) = (x - \alpha^b)(x - \alpha^{b+1}) \dots (x - \alpha^{b+d-2}).$$

The *parity-check matrix* $\mathbf{H}$ of the RS code is

$$\mathbf{H} = \begin{pmatrix} 1 & \alpha^b & (\alpha^b)^2 & (\alpha^b)^3 & \dots & (\alpha^b)^n - 1 \\ 1 & \alpha^{b+1} & (\alpha^{b+1})^2 & (\alpha^{b+1})^3 & \dots & (\alpha^{b+1})^n - 1 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & \alpha^{b+d-2} & (\alpha^{b+d-2})^2 & (\alpha^{b+d-2})^3 & \dots & (\alpha^{b+d-2})^n - 1 \end{pmatrix}.$$

In this dissertation, we assume that $b = 1$, and the resultant code is a *narrow-sense* RS code. For more information, see [11], [96], and [57].

2.1.2 Product Codes

A product code $\mathcal{P}$ is a two-dimensional code composed of two linear codes $\mathcal{C}^{(1)}$ and $\mathcal{C}^{(2)}$. The (n_1, k_1, d_1) $\mathcal{C}^{(1)}$ is a k_1 dimensional linear code with n_1-tuple codewords that are at least d_1 Hamming-distance apart from each other. Similarly, $\mathcal{C}^{(2)}$ is defined as a (n_2, k_2, d_2) linear code. The correction capability of a maximum-likelihood decoder of $\mathcal{C}^{(1)}$ ($\mathcal{C}^{(2)}$) is $t_1 = \lfloor \frac{d_1 - 1}{2} \rfloor$ ($t_2 = \lfloor \frac{d_2 - 1}{2} \rfloor$).

The product code is a $(n_1 n_2, k_1 k_2, d_1 d_2)$ linear code whose codewords are represented as an $n_2 \times n_1$ array, shown in Fig. 2.1. Each $k_1 k_2$-tuple information symbol is arranged in a $k_2 \times k_1$ array. A row-column encoding scheme encodes each row of the $k_2 \times k_1$ array with $\mathcal{C}^{(1)}$ resulting in a $k_2 \times n_1$ array. Now, each column of this array is encoded using $\mathcal{C}^{(2)}$. The resultant $n_2 \times n_1$ array is a codeword in the product code, and for obvious reasons, $\mathcal{C}^{(1)}$ and $\mathcal{C}^{(2)}$ are referred to as *row code* and *column code* respectively. A column-row encoding scheme - an alternative encoding scheme–encodes each column of the $k_2 \times k_1$

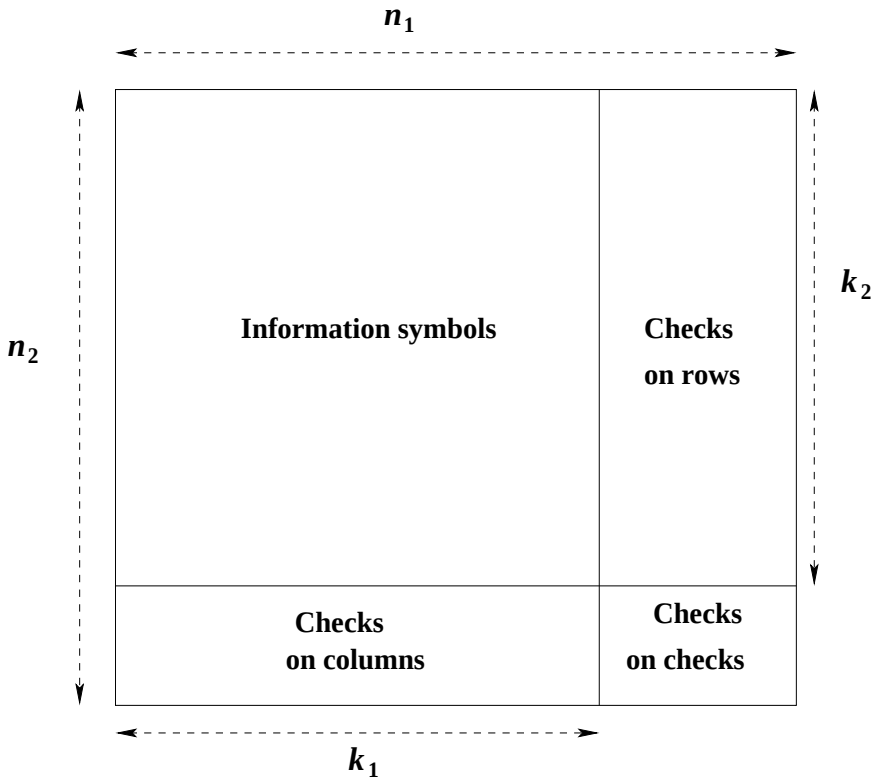


Figure 2.1: Structure of a product code $\mathcal{P}$.

array with $\mathcal{C}^{(2)}$, and then encodes each row of the resultant $n_2 \times k_1$ array with $\mathcal{C}^{(1)}$. In A.1, we show that the row - column encoding scheme and the column-row encoding scheme result in the same codeword for a given information symbol array. Finally, it is easy to show that the minimum distance of the product code is $d_1 d_2$.

2.1.3 Low-Density Parity-Check Codes

A low-density parity-check code (LDPC) $\mathcal{C}$ is a linear block code with codewords of length n and rate $R = \frac{k}{n}$. Unless mentioned otherwise, the codes considered in this work are binary. The code $\mathcal{C}$ is defined using a *sparse parity-check matrix* $\mathbf{H}$ in GF(2) with $m \geq n - k$ rows and n columns, and h_{ij} is the element in position (i, j) of $\mathbf{H}$. A sparse parity-check matrix is defined as a matrix with a small ratio of ones. Typically, the rows of $\mathbf{H}$ are linearly dependent, and the matrix has a rank $n - k$. The code $\mathcal{C}$ is *regular* if

every column and row of $\mathbf{H}$ have *weights* λ and ρ respectively. For examples, see [94], [101], and [98]. The weight of a column (or row) is defined as the number of ones in it. An *irregular* code is one that is not regular, and its column and row weight distributions are $\lambda(x) = \sum_{i=1}^{\lambda_{\max}} \lambda_i x^{i-1}$ and $\rho(x) = \sum_{j=1}^{\rho_{\max}} \rho_j x^{j-1}$ respectively, where λ_i (ρ_j) is the fraction of columns (rows) with weight $i(j)$, $\lambda_{\max}$ is the maximum of column weights and $\rho_{\max}$ is the maximum of row weights. For examples, see [100]. The *generator matrix* $\mathbf{G}$ is a $k \times n$ binary matrix obtained from $\mathbf{H}$ by using the Gaussian elimination process. Let $d_{\min}$ be the minimum distance of the code and $t = \lfloor \frac{d_{\min}-1}{2} \rfloor$ be the correction radius of the conventional *distance-bounded decoder*.

The most popular graphical representation of $\mathcal{C}$ is called the Tanner graph, introduced by Tanner [92] and [93]. The Tanner graph $\mathcal{G}$ of $\mathcal{C}$, defined by $\mathbf{H}$, is a *simple bipartite graph* with two sets of nodes, namely *variable nodes* and *check nodes*. A variable node corresponds uniquely to a column of $\mathbf{H}$, and a check node corresponds uniquely to a row of the matrix. It is common to refer to variable nodes as *bit nodes*, and the term is used interchangeably (without confusion) with bit variables of $\mathcal{C}$. Similarly, it is common to refer to check nodes as *constraint nodes*, and the term is used interchangeably with check or constraint equations of the code. On the same note, it is common to refer to properties of $\mathcal{G}$ as properties of the code. In $\mathcal{G}$, a variable node is adjacent to check node if the bit variable, corresponding to the variable node, is "involved" in the constraint equation, corresponding to the check node. The *degree* of a node is defined as the number of edges incident on the node, and obviously, degree of a variable (check) node is equal to the weight of the corresponding column (row) of $\mathbf{H}$. The length of the shortest cycle of $\mathcal{G}$ is called the *girth* g of the graph, and g is even and at least 4 because the graph is simple and bipartite. Although it will become clear in latter chapters, it is important to note that a graph $\mathcal{G}$ uniquely defines $\mathcal{C}$ up to permutation but the (nontrivial) code can have several Tanner graph realizations. The definition of $\mathcal{G}$ guarantees equivalence with only

$\mathbf{H}$ and not with $\mathcal{C}$.

Example 1 (Graph Representation)**.** *Let* $\mathbf{H}$ *be the parity-check matrix of a* $(7, 3)$ *regular LDPC code with* $\lambda = \rho = 3$.

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

The Tanner graph $\mathcal{G}$ *of* $\mathbf{H}$ *is shown in Fig. 2.2. The circles are the variable nodes and the rectangles are the check nodes.*

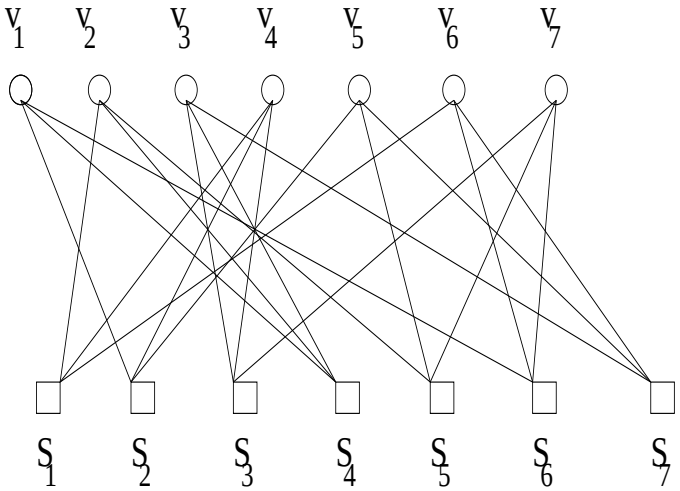


Figure 2.2: Tanner graph of a $(7, 3)$ LDPC code

For more information on LDPC codes, see [78].

2.2 Channel Models

The first step to designing a communication system for an application is to develop a channel model that reliably captures the effects of the transmission process. The channels in relaity are complex, and reliable models are developed to capture significant aspects of these channels. The following is a list of channel models that are simple, and are commonly used in gaining insight into the performance of a coding system.

2.2.1 Binary Erasure Channel

The *binary erasure channel* (BEC) is one of the simplest channel models. The input alphabet is binary, i.e., $\mathbb{F}_2$, and the output alphabet is ternary, namely $\{0, 1, \epsilon\}$. The symbol ϵ will be referred to as an *erasure*. An input bit 0 (1) transmitted over the channel can be received as a 0 (1) or an ϵ. Note that the input to the channel is received error free or as an erasure at the output. The probability of a 0 or 1 received in erasure is p_e and it is referred to as the *erasure probability*. See Fig. 2.3 for an illustration of the channel.

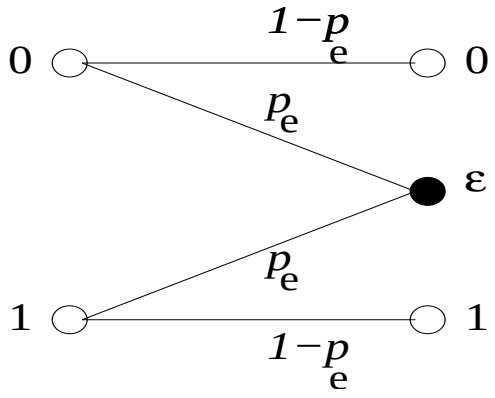


Figure 2.3: Binary erasure channel model

2.2.2 Binary Symmetric Channel

The input and output alphabets of the channel model are binary. An input bit transmitted over the channel is received in error with probability α and error free with probability $1-\alpha$. The symmetry of the channel allows to define transition probabilities independent of the input bit. Unlike the BEC, positions of errors at the output of the *binary symmetric channel* (BSC) are not identifiable. See Fig. 2.4 for an illustration of the channel.

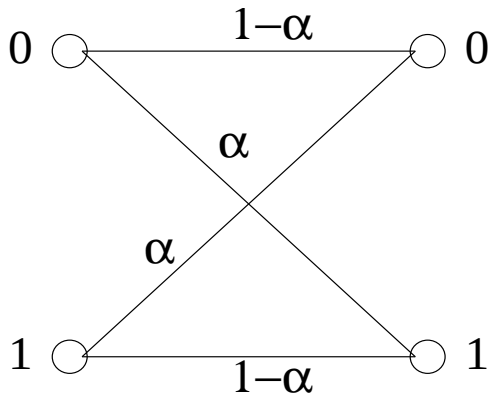


Figure 2.4: Binary symmetric channel model

2.2.3 Hidden Markov Channel Model

In the BSC, channel error sequences are uncorrelated. A *hidden Markov model* (HMM) of a channel is a discrete-time finite state machine where error sequences are correlated. This model can be described as a generalization of the BSC. The state of the HMM at discrete time t, where $t \in \mathbb{Z}^+$, is defined as the realization of the random variable $\mathbf{X}_t$. The state space of $\mathbf{X}_t$ (independent of t) is $\{s_1, s_2, \cdots, s_N\}$. The finite state machine satisfies the Markov property given in (2.2), and the time homogeneity given in (2.3).

$$\Pr(\mathbf{X}_{t+1} = x_{t+1} | \mathbf{X}_t = x_t, \cdots, \mathbf{X}_0 = x_0) = \Pr(\mathbf{X}_{t+1} = x_{t+1} | \mathbf{X}_n = x_t) \quad (2.2)$$

$$\Pr(\mathbf{X}_{i+1} = x_{i+1} | \mathbf{X}_i = x_i) = \Pr(\mathbf{X}_{j+1} = x_{j+1} | \mathbf{X}_j = x_j) \quad (2.3)$$

Any state s_i of the HMM makes an error with a probability P_i. This means that each state of the HMM can be described as a BSC channel with $\alpha_i = P_i$ (assuming equiprobable input).

At any time t, the only observable quantity of the HMM is the presence or absence of error in the channel output. Since the state transitions are not observable, such a Markov model is referred to as the hidden Markov model. The HMM is completely defined by specifying a *state transition matrix* $\mathbf{P}^{(\mathrm{T})}$, and probability P_i,$1 \leq i \leq N$, of making errors in the state s_i. The state transition matrix $\mathbf{P}^{(\mathrm{T})}$ is a $N \times N$ whose (i,j)-th element, denoted as $\mathbf{P}^{(\mathrm{T})}_{i,j}$, is the probability of transition from s_i to s_j. The steady-state probability π_i of the HMM is the average probability of being in the state s_i, and it is computed by solving the system of equations in (2.4).

$$\begin{aligned} \langle \pi_1, \pi_2, \cdots, \pi_N \rangle &= \langle \pi_1, \pi_2, \cdots, \pi_N \rangle \, \mathbf{P}^{(\mathrm{T})} \\ \pi_1 + \pi_2 + \cdots + \pi_N &= 1 \end{aligned} \tag{2.4}$$

The average channel symbol error rate is $p_{\mathrm{s,ch}} = \sum_{i=1}^{N} \pi_i P_i$.

The correlated-error sequences of a HMM are governed by $\mathbf{P}^{(\mathrm{T})}$. An *error burst* of length l is defined as an event of l consecutive error preceded and followed by error-free symbols. Similarly, an *error gap* of length g is defined as an event of g consecutive error-free symbols preceded and followed by erroneous symbols. In this paper, we have considered a two-state HMM which is commonly referred to as the Gilbert-Elliot channel (GEC) model [38], [34]. The two states of the GEC model are referred to as the *Good state* and the *Bad state*. The probability of making an error in the Good state is P_{G} (usually zero), and the probability of making an error in the Bad state is P_{B} (usually non-zero). The state transition matrix is

$$\mathbf{P}^{(\mathrm{T})} = \begin{pmatrix} b & 1-b \\ 1-a & a \end{pmatrix}. \tag{2.5}$$

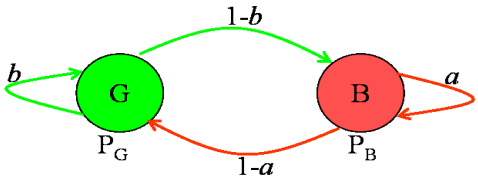


Figure 2.5: Illustration of the GEC model.

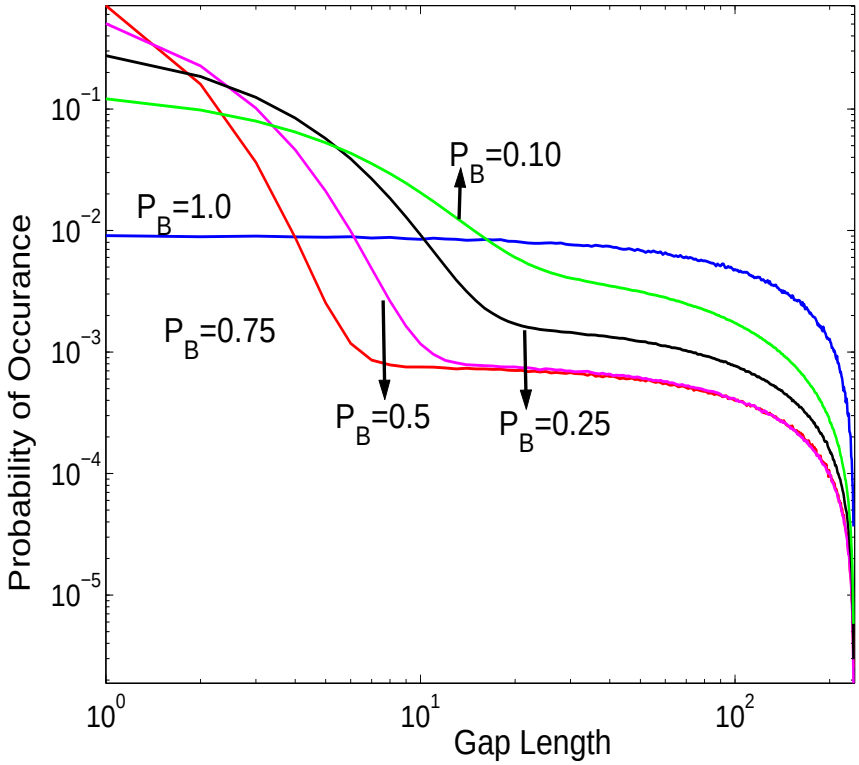


Figure 2.6: Gap length density of GEC channels with $P_{\mathrm{G}} = 0$, $a = 0.90$ and $p_{\mathrm{s,ch}} = 0.01$.

See Fig. 2.5 for an illustration of the GEC model.

The steady-state probabilities of the GEC are $\pi_{\mathrm{G}} = \frac{1-a}{2-a-b}$ and $\pi_{\mathrm{B}} = \frac{1-b}{2-a-b}$, and the average channel symbol error rate is $p_{\mathrm{s,ch}} = \pi_{\mathrm{G}} P_{\mathrm{G}} + \pi_{\mathrm{B}} P_{\mathrm{B}}$. The probability density of error gaps and error bursts in a GEC for different values of g and l are shown in Fig. 2.6 and Fig. 2.7 respectively. Each curve correspond to different values of P_{B} while $P_{\mathrm{G}} = 0$, $a = 0.90$ and $p_{\mathrm{s,ch}} = 0.01$. From Fig. 2.7, the average burst lengths decrease with the decrease in a.

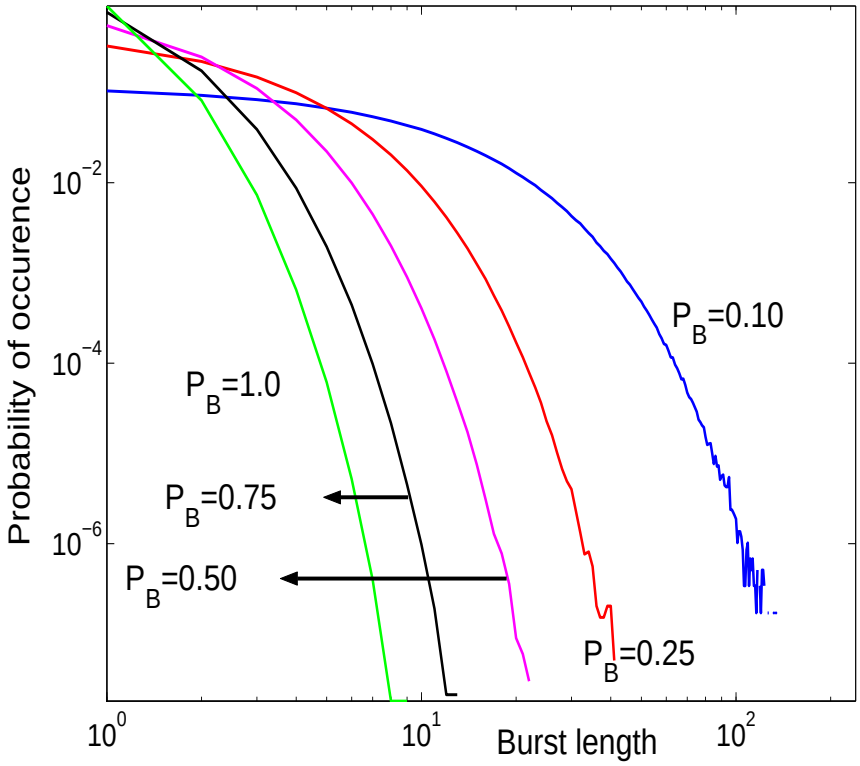


Figure 2.7: Burst length density of GEC channels with $P_{\mathrm{G}} = 0$, $a = 0.90$ and $p_{\mathrm{s,ch}} = 0.01$.

2.2.4 Additive White Gaussian Noise Channel

The *additive white Gaussian noise* (AWGN) channel is one of the simplest channel models of a discrete-input continuous-output channel. In this dissertation, we assume that the input alphabet of the channel is bipolar, i.e., $\{-1, +1\}$, and the output alphabet is the real line $\mathbb{R}$. An input $x \in \mathbb{F}_2$ transmitted over the channel is received at the output as $x+n \in \mathbb{R}$, where n is an instance of a white zero-mean Gaussian process. The variance of the process is σ^2. See Fig. 2.8 for an illustration of the channel.

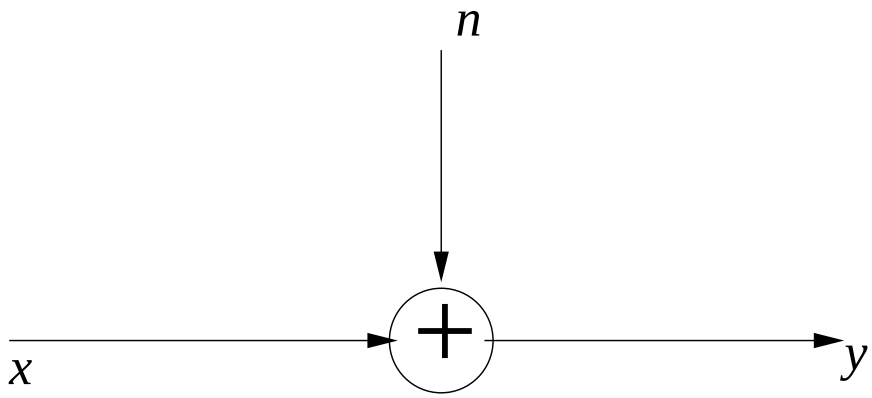


Figure 2.8: Additive white Gaussian noise channel model

2.3 Iterative Decoding Algorithms

The codeword in $\mathcal{C}$ transmitted over the channel is $\mathbf{x} = \langle x_1, x_2, x_3, \ldots, x_n \rangle$, and the word received at the output of the channel is $\mathbf{y} = \langle y_1, y_2, y_3, \ldots, y_n \rangle$. The Tanner graph $\mathcal{G}$ of $\mathbf{H}$ has n variable nodes labeled v_i, $1 \leq i \leq n$, and m check nodes labeled S_j, $1 \leq j \leq m$. Let $\mathcal{N}(v_i) := \{S_j | h_{ji} = 1\}$ be the neighbors of v_i. Similar definition applies for $\mathcal{N}(S_j)$.

Example 2 (Neighbor Sets)**.** *For $\mathcal{G}$ in Example 1, the neighbor sets of v_1 and S_2 are*

$$\mathcal{N}(v_1) = \{S_2, S_4, S_6\}$$

$$\mathcal{N}(S_2) = \{v_1, v_4, v_5\}.$$

The message from v_i to S_j in the l^{th} iteration is labeled $\mu^{(l)}{}_{i\to j}$ and the message from S_j to v_i in the l^{th} iteration is labeled $\mu^{(l)}_{j\to i}$. **Note that message $\mu_{i\to j}$ ($\mu_{j\to i}$) is valid *iff* v_i and S_j are adjacent.** The decoder estimate of $\mathbf{x}$ is labeled $\hat{\mathbf{x}}$.

2.3.1 Iterative Erasure Decoder for the BEC

A decoder for BEC attempts to solve a system of linear equations, defined by a linear block code, for variables in erasure. The Gaussian elimination is an optimal method to solve the system but its complexity is $\mathcal{O}(n^3)$. The iterative erasure decoder is a sub-optimal approach with complexity linear in the length of the code. This decoder works on the Tanner graph of the code and the message alphabet is the same as the output alphabet of the BEC. The decoding algorithm is described as follows.

1. **Initialization**

   - Set $v_i = y_i$, $\forall\ 1 \le i \le n$.

2. **Iteration step**

   - If no check node is connected to exactly one erasure, then go to Step 3.
   - Set $\mu_{i\to j} = v_i$, $\forall i, j$ such that $h_{ji} = 1$.
   - $\forall i, j$ such that $h_{ji} = 1$, set

$$\mu_{j\to i} = \begin{cases} \mu_{i\to j} & \text{if } \mu_{i\to j} = 0 \text{ or } 1 \\ \bigoplus_{\substack{i'=1 \\ i'\neq i}}^{n} \mu_{i'\to j} & \text{if } \mu_{i'\to j} = \epsilon \text{ for only } i' = i \\ \epsilon & \text{otherwise} \end{cases}$$

   - For every i, if there exist at least one $\mu_{j\to i} \neq \epsilon$, then set $v_i = \mu_{j\to i}$.
   - Go to Step 2.

3. **Decoding step**

- Set $\hat{x}_i = v_i$, $\forall\ 1 \leq i \leq n$.

A decoding failure is declared if $\hat{x}_i = \epsilon$ for some values of i. A decoding failure occurs if the set of variable nodes in erasure contains a *stopping set*. A stopping set $\mathcal{S}$ of a given $\mathbf{H}$ is a subset of variable nodes such that all neighbors of $\mathcal{S}$ are adjacent to at least two elements in $\mathcal{S}$. The set of all stopping sets of $\mathbf{H}$ is sufficient to analytically compute the performance of the code under the iterative erasure decoder.

2.3.2 Iterative Hard-Decision Decoder for the BSC

The iterative hard-decision decoder for the BSC is a sub-optimal decoder, and it is one of the simplest decoders that belong to the category of *message-passing decoders*. This decoder works on the Tanner graph of $\mathbf{H}$ by sending binary messages over the edges. Since the input and output alphabets are binary, all operations of the decoder are defined over $\mathbb{F}_2$. The decoding algorithm is described as follows.

1. **Initialization**

   - Set $v_i = y_i$, $\forall\ 1 \leq i \leq n$.
   - Set $\mu_{j \to i}^{(l)} = v_i$, $\forall h_{ji} = 1$.
   - Set iteration round $l \leftarrow 1$

2. **Iteration step**

   - Compute syndrome, i.e., $S_j = \bigoplus_{i=1}^{n} \mu_{j \to i}^{(l)}$ for $1 \leq j \leq m$.
   - If $S_j = 0$ for all $1 \leq j \leq m$, then go to Step 3
   - Set $\mu_{j \to i}^{(l)} = \bigoplus_{\substack{i'=1 \\ i' \neq i}}^{n} \mu_{i' \to j}^{(l)}$, $\forall i, j$ such that $h_{ji} = 1$.

- $\forall i, j$ such that $h_{ji} = 1$, set $\mu_{j\to i}^{(l)}$ based on majority logic as follows.

$$\mu_{j\to i}^{(l)} = \begin{cases} 1 & \text{if } \sum_{\substack{j'=1 \\ j'\neq i}}^{m} (2\mu_{i\to j'}^{(l)} - 1) > 0 \\ 0 & \text{if } \sum_{\substack{j'=1 \\ j'\neq i}}^{m} (2\mu_{i\to j'}^{(l)} - 1) < 0 \\ y_i & \text{otherwise} \end{cases}$$

.

- $l \leftarrow l + 1$
- If $l < l_{\max}$, then go to Step 2.

3. **Decoding step**

- Apply majority logic over all messages sent to v_i, $\forall\ i$.

$$\hat{x}_i = \begin{cases} 1 & \text{if } \sum_{j'=1}^{m} (2\mu_{i\to j'}^{(l)} - 1) > 0 \\ 0 & \text{if } \sum_{j'=1}^{m} (2\mu_{i\to j'}^{(l)} - 1) < 0 \\ y_i & \text{otherwise} \end{cases}$$

.

This algorithm is known as the *Gallager B algorithm* [87]. In determining ${\mu^{(l)}}_{j\to i}$, different thresholds besides majority logic can be used. The iteration process is halted if either all-zeros syndrome is achieved or $l_{\max}$ iterations are complete. See [37] and [88] for more details.

2.3.3 Sum-Product Decoder for the AWGN Channel

The sum-product decoder for the AWGN channel [37] is a sub-optimal decoder (for most codes), and it has gained in popularity after its rediscovery by MacKay [59]. This decoder works on the Tanner graph of $\mathbf{H}$ by sending *log-likelihood ratios* (LLR) over the edges. The input codebit 0 is transmitted as -1, and 1 is transmitted as $+1$. The output alphabet of the channel is $\mathbb{R}$, and the LLRs are in $\mathbb{R}$. The log-likelihood ratio of

x_i is defined as $\log(\frac{\Pr(y_i|x_i=0)}{\Pr(y_i|x_i=1)})$. The quantities in the numerator and denominator of the fraction is defined by the variance σ^2 of the white Gaussian noise process.

The sum-product algorithm computes the *posteriori* probabilities of x_i, $1 \leq i \leq n$, if $\mathcal{G}$ is a *tree*. A tree is defined as a connected graph with no cycles. If $\mathcal{G}$ is a tree, then the sum-product decoder arrives at an estimate $\hat{\mathbf{x}}$ of $\mathbf{x}$ that maximizes the *a posteriori* probabilities. Unfortunately, codes with cycle-free graphs have rates less than half or have a minimum distance of 2. Hence, they are not of great practical interest. The algorithm, described as follows, is sub-optimal and is used to decode on graphs with cycles.

1. **Initialization**

   - Set $v_i^{(0)} = \frac{2y_i}{\sigma^2}$, $\forall\ 1 \leq i \leq n$.
   - Set $\mu_{j\to i}^{(l)} = v_i^{(0)}$, $\forall i, j$ such that $h_{ji} = 1$.
   - Set iteration round $l \leftarrow 1$

2. **Iteration step**

   - Compute syndrome S_j for $1 \leq j \leq m$, i.e., $S_j = \bigoplus_{\substack{i=1 \\ h_i j=1}}^{n} (v_i^{(l-1)} > 0)$.
   - If $S_j = 0$ for all $1 \leq j \leq m$, then go to Step 3
   - For every j, set $\mu_{j\to i}^{(l)} = -2\tanh^{-1}(\prod_{\substack{i'=1 \\ i' \neq i}}^{n} \tanh \mu_{i'\to j}^{(l)})$, $\forall i, j$ such that $h_{ji} = 1$.
   - For every i, compute $\mu_{j\to i}^{(l)}$ and ${v^{(l-1)}}_i$ as follows.

$$\mu_{j\to i}^{(l)} = v_i^{(0)} + \sum_{\substack{j'=1 \\ j' \neq i}}^{m} \mu_{i\to j'}^{(l)}, \forall i, j \text{ such that } h_{ji} = 1$$

$$v_i^{(l)} = v_i^{(0)} + \sum_{j=1}^{m} \mu_{j\to i}^{(l)}, \forall i, j \text{ such that } h_{ji} = 1$$

   - $l \leftarrow l + 1$

- If $l < l_{\max}$, then go to Step 2.

3. **Decoding step**

- For every i, compute $\mu_{j\to i}^{(l)}$ and $v^{(l-1)}{}_i$ as follows.

$$\hat{x}_i = v_i^{(0)} + \sum_{j=1}^{m} \mu_{j\to i}^{(l)}, \forall i, j \text{ such that } h_{ji} = 1$$

Further details of the algorithm can be found in [56], [44], [16] and [109].

2.4 Basic Communication System For Magnetic Data Storage

A bulk of data can be stored in a magnetic medium using one of the two recording modes, namely *longitudinal* and *perpendicular*. Magnetic hard-disk and tape systems are examples of storage systems that use magnetic mediums. The acts of writing and reading introduce linear and nonlinear noise components to the user information. The communication system is designed to retrieve user information with high fidelity, and it uses signal processing and channel coding techniques to effectively combat noise issues. The system model used in simulations is shown in Fig. 2.9. The input bits $c_k, c_k \in \{-1, 1\}$, are recorded onto the magnetic disk using longitudinal (or) perpendicular recording technique. The input bits c_k can be obtained by channel-coding the user bits $b_k, b_k \in \{0, 1\}$. As previously mentioned, channel codes add controlled redundancies in user bits to produce codeword bits. The read-back signal of the recorded input bits can be determined by modeling the perpendicular (or) longitudinal recording channel as a *finite impulse response* (FIR) filter. The FIR filter that models recording channel can be characterized by its impulse response as given below.

$$h_{\mathrm{L}}(x) = \frac{-8x}{\mathrm{ND}^2} \frac{K}{(1 + \frac{4x^2}{\mathrm{ND}^2})} \qquad \text{(for longitudinal recording)} \tag{2.6}$$

$$h_{\mathrm{P}}(x) = \frac{2}{\mathrm{ND}} \sqrt{\frac{ln2}{\pi}} \mathrm{e}^{\frac{-4x^2}{\mathrm{ND}^2} ln2} \qquad \text{(for perpendicular recording)} \tag{2.7}$$

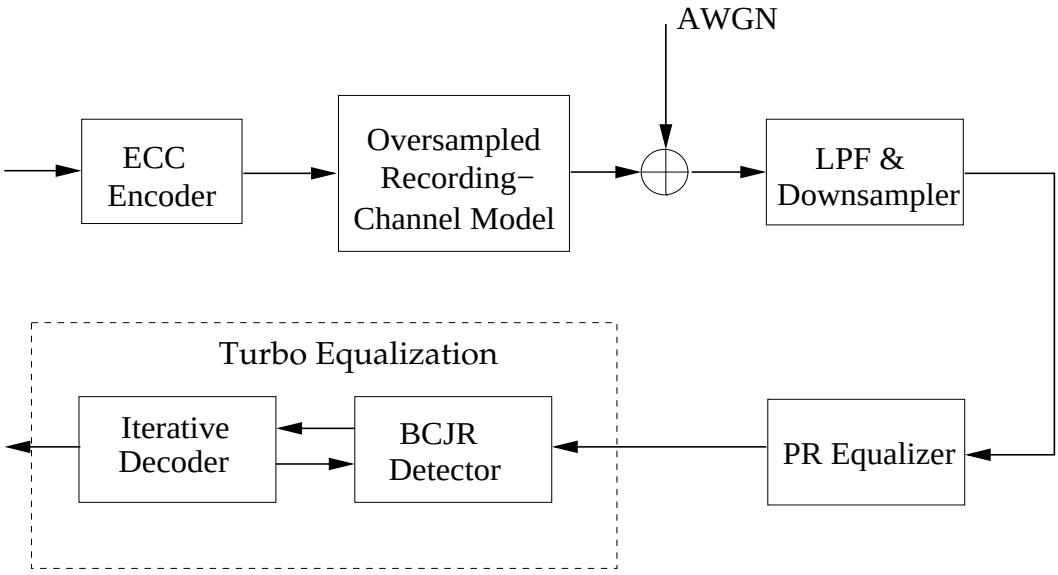


Figure 2.9: Communication System Model of the Magnetic Storage System

In (2.6) and (2.7), x is a time index normalized by the channel clock T, ND is the normalized recording density and K is a scaling constant that can be set to unity. The *normalized density* of perpendicular recording is defined as the width of the impulse response at 50% of its peak value normalized by T. The normalized density of longitudinal recording is the width of the step response at 50% of its peak value normalize by T. For longitudinal recording, the step response $H_{\mathrm{L}}(x)$ of the FIR filter is modeled as a Lorentzian pulse [8].

$$H_{\mathrm{L}}(x) = \frac{K}{1 + (\frac{2x}{\mathrm{ND}})^2} \tag{2.8}$$

The impulse response of perpendicular and longitudinal magnetic recording channels for different values of normalized densities is illustrated in Fig. 2.10 and Fig. 2.11.

The noise $n(t)$ in the recording channel, is modeled as an additive white gaussian noise (AWGN) with variance σ^2. The input signal-to-noise ratio (SNR) is defined as

$$SNR = 10\ \log(\frac{1}{\sigma^2}) \tag{2.9}$$

The read-back signal is passed through a low-pass filter and sampled at T intervals to

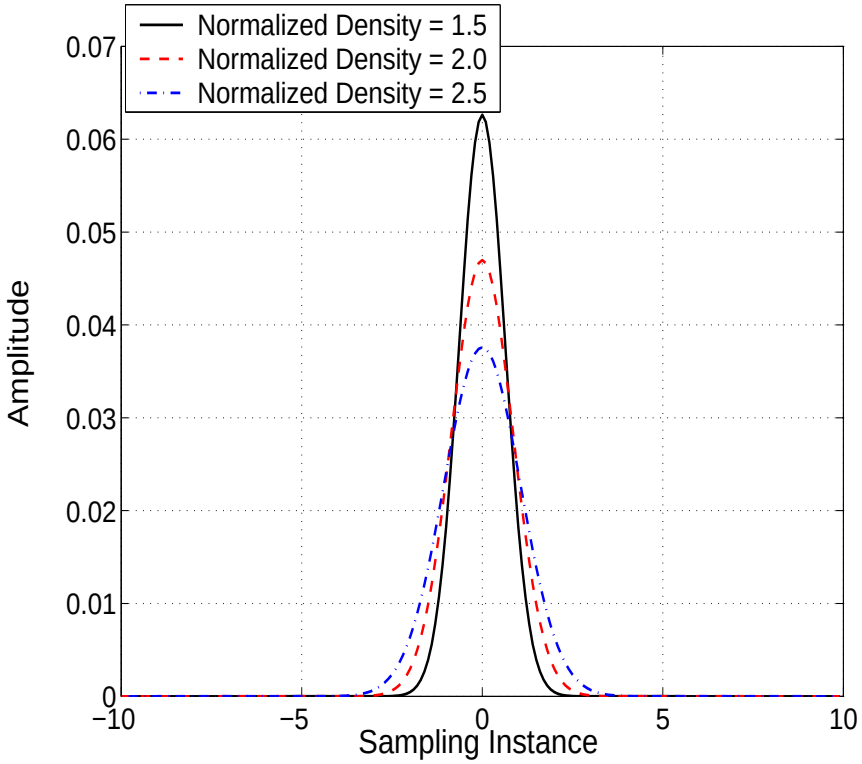


Figure 2.10: Impulse Response of a Perpendicular Magnetic Channel

obtain digital samples r_k. The digital samples r_k are equalized to a given *partial-response* (PR) *target*. The PR-target depends on the recording mode and the normalized density. Detector based on *maximum-likelihood* (ML) criterion or MAP criterion is used to make an estimate of the channel bits. Viterbi algorithm is used to make decisions based on the ML criterion, and the BCJR algorithm is used to make decisions based on the MAP criterion. The details of BCJR algorithm can be found in [7], [104] and [79]. The choice of a decoder depends on the choice of the channel code. For example, if an LDPC code is used, then the sum-product algorithm is used to exploit the redundancies and decode. The SPA produces soft-information about every bit c_k and cooperates with BCJR algorithm to improve the quality of this information. This cooperation technique is often referred to as *turbo equalization*. For further details on the communication system model, refer to [69], [70], [65], [35], [66] and [111]. The communication system

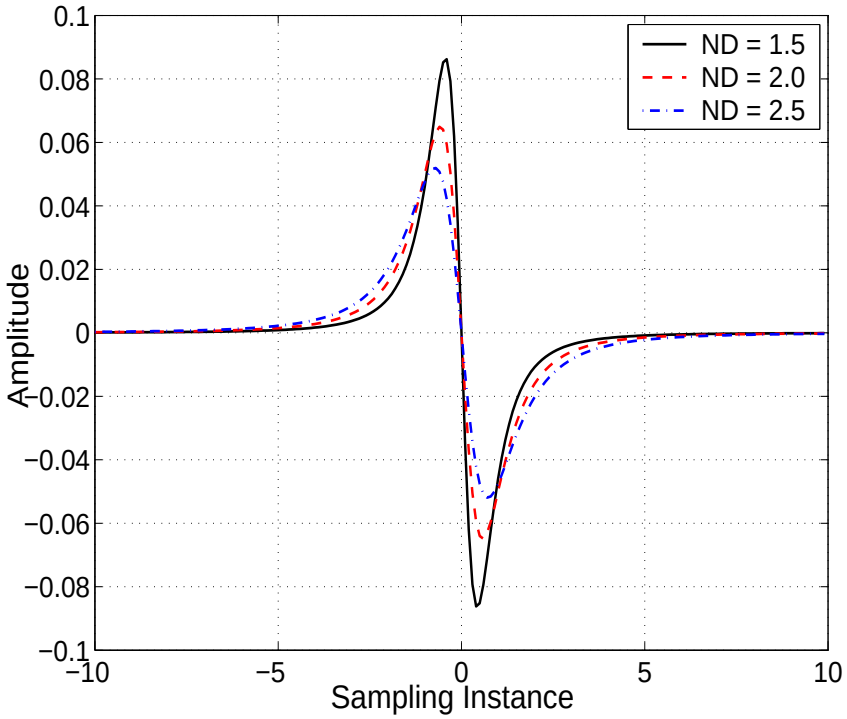
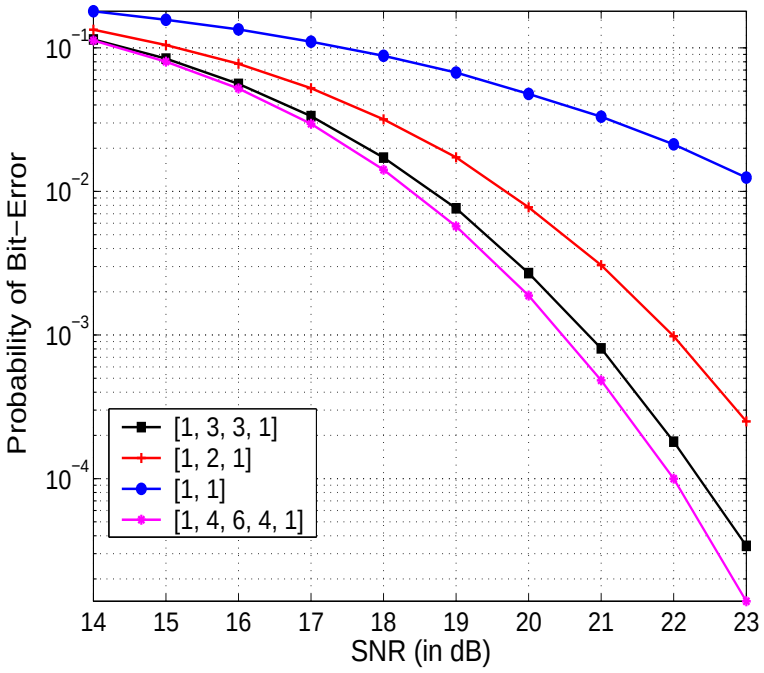
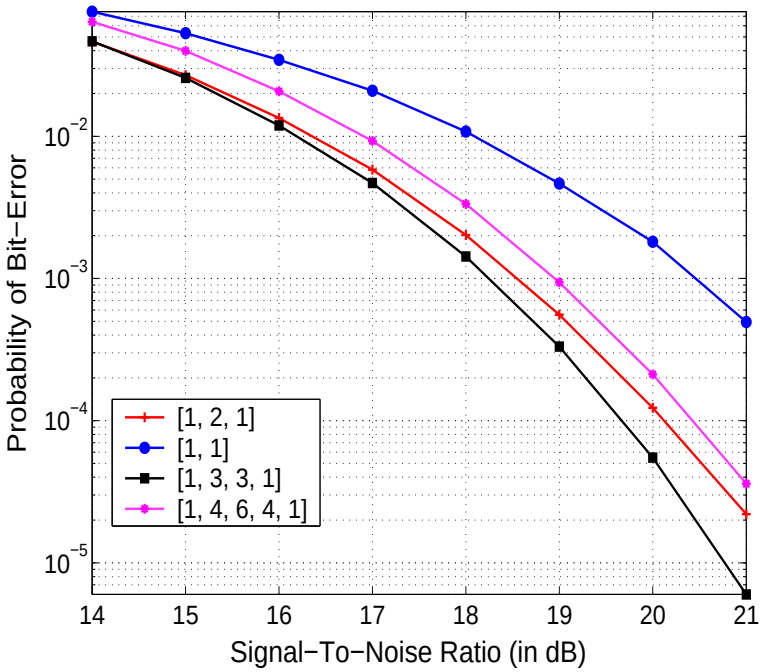


Figure 2.11: Impulse Response of a Longitudinal Magnetic Channel

models described using Matlab and C-language were used to study the performance of various detection and decoding schemes. The bit-error rate performance of the Viterbi detector for different normalized densities and PR-targets on a perpendicular magnetic channel is shown in Fig. 2.12. The bit-error rate performance of the Viterbi detector for different PR-targets on a longitudinal magnetic channel is shown in Fig. 2.13.



(a) ND = 2.5



(b) ND = 2.0

Figure 2.12: Performance of the Viterbi Detector on Perpendicular Magnetic Channel of an Uncoded System

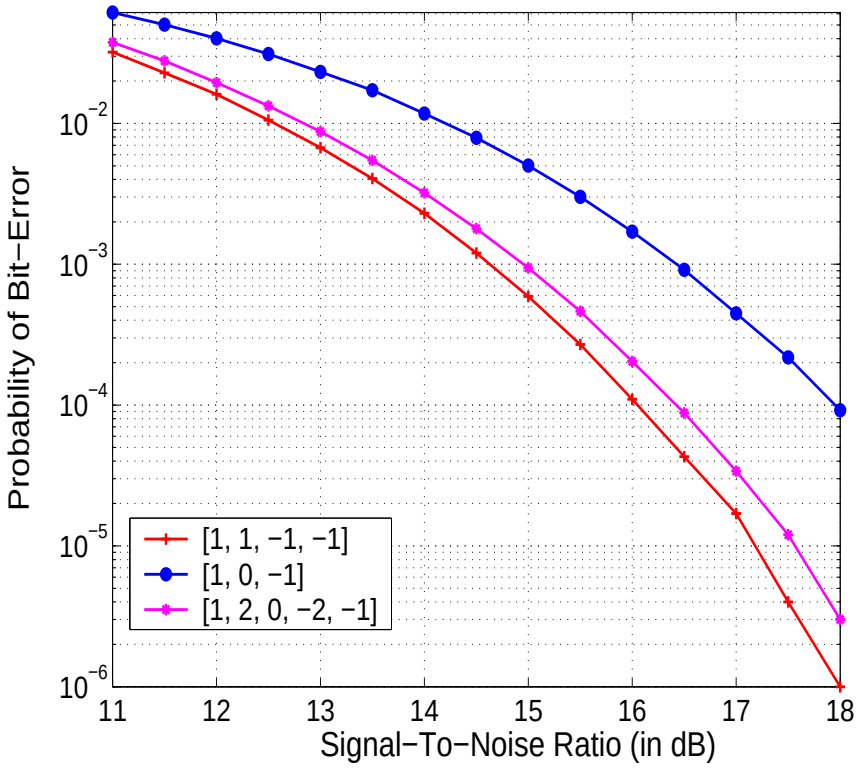


Figure 2.13: Performance of the Viterbi Detector on Longitudinal Magnetic Channel at ND = 2.5

CHAPTER 3

PRODUCT CODES ON CHANNELS WITH MEMORY

The exponential growth of digital information in recent years have immensely increased the demand for efficient and reliable mass storage systems. The magnetic tape drive systems is one of the mass storage devices that are mainly used for archiving and supporting backup and restore procedures. Also, it is the slow and inexpensive counterpart of the fast and expensive hard disk drives. It is not surprising that data reliability and robustness against errors due to the read/write operations or the media is very critical in these systems. One of the methods used to guarantee high reliability is the error-correction coding (ECC) schemes. The ECMA-319 Standard accepted by the industries describes the physical and magnetic characteristics of tape cartridges, quality of the recorded signals, data organization on the tapes, and reading/recording formats. The Standard specifies a Reed-Solomon codes based product code for error correction. In this chapter, we propose a method to evaluate the standard RS-code based product code, and we investigate possible new alternatives for improved performance.

In Section 3.1, we develop a mathematical model of the error-correction coding subsystem of a magnetic tape drive system. This model is developed from the data organization format and error-correction scheme specified in the Linear Tape Open Standard. In Section 3.2, we present a brief description of the decoder for RS-based product code. A simulation model of the ECC subsystem model is developed in Section 3.3 in order to evaluate the performance of a product-code-based error-correction scheme. Although such a simulation model is a convenient way for performance evaluation of complex ECC subsystems, an analytical technique is preferable for computing symbol and codeword

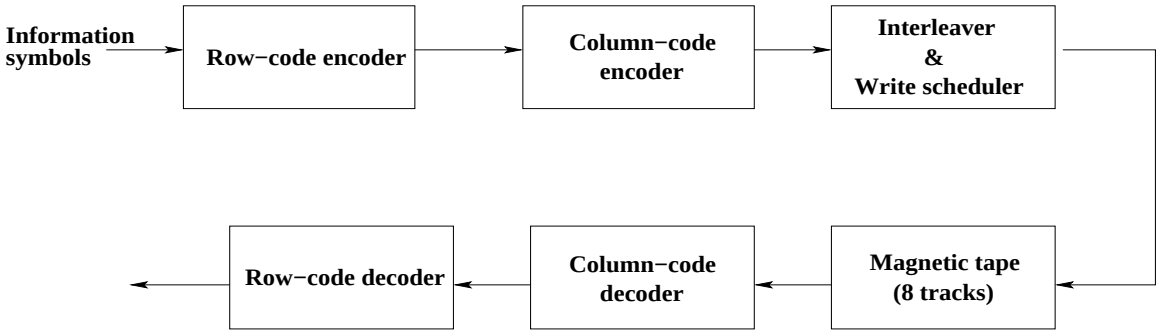


Figure 3.1: Illustration of the ECC subsystem of a tape drive system.

error-rates below 10^{-6} in linear time. In Section 3.4, we describe the analytical computation technique for an ECC subsystem model defined by two-state HMMs. Indeed this technique can be applied for models defined by N-state HMMs. Using the simulation model and the analytical technique, we compute the performance of the error-correcting scheme specified in the Standard and other alternatives. These results are compiled in Section 3.5. The validity of the analytical method is also established by comparing the computed performance with that obtained via simulations. In Section 3.6, we conclude this chapter with a listing of the topics for future research work. This work was published in [84].

3.1 Modeling ECC Subsystem

An application where product codes are used for error correction is the tape drive system. The ECC subsystem of a tape drive system is illustrated in Fig. 3.1. The development of the *ECC subsystem model* for performance analysis is based on the specifications in the ECMA-319 Standard [33]. This ECMA Standard, accepted by the tape-drive manufacturing companies, specifies physical and magnetic characteristics of magnetic tape cartridges in order to facilitate physical interchange of cartridges among drives. This standard, amongst many other things, lays out methods to organize data on tapes, error correction schemes to protect this data, and recording schedules to write this data

Track 0	**0**	**9**	**18**	**27**	**36**	**45**	.....	**484**	**493**	**502**	**511**
Track 1	**1**	**10**	**19**	**28**	**37**	**46**	.....	**485**	**494**	**503**	**504**
Track 2	**2**	**11**	**20**	**29**	**38**	**47**	.....	**486**	**495**	**496**	**505**
Track 3	**3**	**12**	**21**	**30**	**39**	**40**	.....	**487**	**488**	**497**	**506**
Track 4	**4**	**13**	**22**	**31**	**32**	**41**	.....	**480**	**489**	**498**	**507**
Track 5	**5**	**14**	**23**	**24**	**33**	**42**	.....	**481**	**490**	**499**	**508**
Track 6	**6**	**15**	**16**	**25**	**34**	**43**	.....	**482**	**491**	**500**	**509**
Track 7	**7**	**8**	**17**	**26**	**35**	**44**	.....	**483**	**492**	**501**	**510**

Figure 3.2: Recording schedule of the quads of a dataset.

on tapes. For more details on the Standard, refer to [33].

As specified in the Standard, the information symbols (along with a few CRC check bytes) are grouped to form a *dataset* that contains 16 54×468-arrays appended to form a row vector. Each one of the 54×468-arrays are referred to as *sub-datasets*, and are encoded separately. A product code $\mathcal{P}^{(\mathrm{ECMA})}$ with two RS codes $\mathcal{C}^{(1)}$ and $\mathcal{C}^{(2)}$ defined over $\mathrm{GF}(2^8)$ is used to protect each sub-datasets. The code $\mathcal{C}^{(1)}$ is a $(240, 234, 7)$ RS code, and the code $\mathcal{C}^{(2)}$ is a $(64, 54, 11)$ RS code. The encoded sub-dataset is a 64×480-array obtained by encoding even and odd symbols of each row of the sub-dataset with $\mathcal{C}^{(1)}$ to form a *codeword pair*, and the columns with $\mathcal{C}^{(2)}$. Hence, the encoded dataset is a 1×16 vector of 64×480 encoded sub-datasets, and two codeword pairs of the encoded dataset along with header information, referred to as *quads*, are numbered row-wise. The set of 8 quads corresponding to each row of the encoded dataset is written into 8 tracks of the magnetic tape. The recording schedule of the quads of a dataset is shown in Fig. 3.2. The elements encircled in the recording schedule represent the quads containing codewords from the first and second sub-datasets. The tracks of the magnetic tape are modeled

as identical and independent HMMs. The recording schedule of the quads guarantees that each track of the magnetic tape contains exactly 8 row codewords of $\mathcal{P}^{(\mathrm{ECMA})}$, and that they are physically separated from each other by at least $7 \times 960 = 6720$ symbols (or 53760 bits for 8-bit symbols). Since the probability of an error burst covering more than 53760 bits is very small, it is fair to assume as writing these row codewords in 8 identical and independent channels. In other words, the ECC subsystem model consists of 64 identical and independent HMMs, i.e., one for each row codeword of $\mathcal{P}^{(\mathrm{ECMA})}$. Also, this means that an error burst in a row of $\mathcal{P}^{(\mathrm{ECMA})}$ does not spill over to the next row. Hence, the ECC subsystem model is sufficiently defined by the parameters of $\mathcal{C}^{(1)}$, $\mathcal{C}^{(2)}$, and a HMM. Such a model has been used in [14]. Although, with an aim to preserve clarity, we have restricted ourselves to GEC models, the analysis techniques presented in this chapter can easily be extended to N-state HMMs.

Since the quads are written in a track of the magnetic tape using a binary modulation format, it is appropriate to use a *bit-based* definition for the GEC of a ECC subsystem model. In a *bit-based* GEC, each state is equivalent to a binary symmetric channel with the crossover probability equal to P_{G} or P_{B}. The transition probability matrix of the bit-based GEC model is denoted as $\mathbf{P}^{(\mathrm{T,bit})}$. An m-bit symbol is error-free if and only if m transitions in the bit-based GEC are error-free. Hence, the error transition matrix $\mathbf{E}^{(\mathrm{T})}$ of an equivalent *symbol based* GEC can be derived by applying the method shown in A.2 on $\mathbf{P}^{(\mathrm{T})}(D) := \left\{\mathbf{P}^{(\mathrm{T,bit})}(D)\right\}^m$, where

$$\mathbf{P}^{(\mathrm{T,bit})}(D) = \begin{pmatrix} b(1-P_{\mathrm{G}}) + bP_{\mathrm{G}}D & (1-b)(1-P_{\mathrm{B}}) + (1-b)P_{\mathrm{B}}D \\ (1-a)(1-P_{\mathrm{G}}) + (1-a)P_{\mathrm{G}}D & a(1-P_{\mathrm{B}}) + aP_{\mathrm{B}}D \end{pmatrix}. \tag{3.1}$$

Finally, a column-row decoding is employed to correct errors introduced by the read-back process. The ECC subsystem model is illustrated in Fig. 3.3.

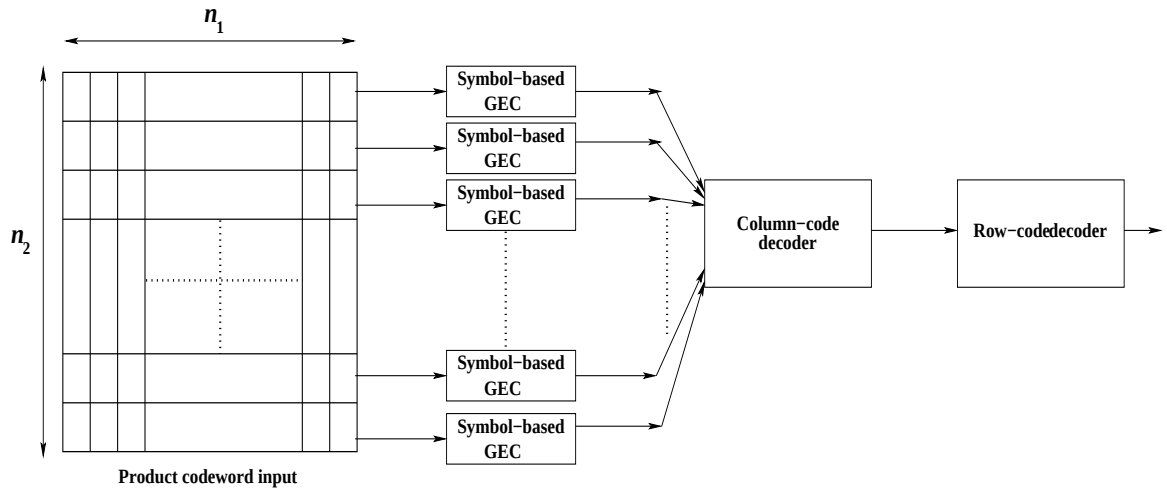


Figure 3.3: Illustration of the ECC subsystem model.

3.2 Decoding Product Codes

Although iterative soft-decision decoding algorithms for product codes, see [71], exist, we consider a sub-optimal hard-decision *bounded-distance decoder*. A *column-row decoder* decodes along the columns first, and then decodes along the rows. If the number of errors in a column (row) codeword is less than or equal to t_2 (t_1), then it is decoded to the correct codeword, if not, the received word is left undisturbed. Unlike the encoding process, the performance of a column-row decoder need not be the same as that of a row-column decoder. The difference in their performance is attributed to the transmit and receive process of the communication system. For the application under consideration in this paper, the column-row decoder is chosen because they perform better than the row-column decoder.

3.3 Simulated Performance of Product Codes

The ECC subsystem model, defined by $\mathcal{C}^{(1)}$, $\mathcal{C}^{(2)}$, and a GEC, is implemented in C to evaluate the performance of the product code $\mathcal{P}$. The performance of $\mathcal{P}$ is analyzed by plotting the row-codeword error rate (and symbol error rate) observed at the output of

the row-decoder for different values of $p_{\mathrm{s,ch}}$.

During each iteration, n_2 mn_1-tuple zero vectors are transmitted over identical n_2 independent channels defined by the GEC model. An element in the output-array of size $n_2 \times n_1$ is in error if there is at least an error in the corresponding block of m channel-output bits. Each column of the output-array is decoded using $\mathcal{C}^{(2)}$, and then each row of the resultant array is decoded using $\mathcal{C}^{(1)}$. The number of symbol errors and row-codeword errors in the column-row decoded array of this iteration are memorized. Several iterations are performed to obtain sufficient error statistics to compute the probability of symbol error, denoted as $p_{\mathrm{s},\mathcal{C}^{(1)}}$, and row-codeword error, denoted as $p_{\mathrm{CW},\mathcal{C}^{(1)}}$. Although interleaving is not explicitly included in the simulator, we can introduce its effects by re-computing the parameters of the GEC models with respect to the given depth of interleaving.

The $p_{\mathrm{CW},\mathcal{C}^{(1)}}$ and $p_{\mathrm{s},\mathcal{C}^{(1)}}$ plots of $\mathcal{P}^{(\mathrm{ECMA})}$ shown in Fig. 3.4 correspond to GEC models with $P_{\mathrm{G}} = 0$, $P_{\mathrm{B}} = 1$, and different values for b. The average burst length of a GEC model increase with an increase in b.

3.4 Analytical Computation of the Performance of Product Codes

Assume that the ECC subsystem model is defined by a product code $\mathcal{P}$ defined over an m-dimension field, and a bit-based GEC with parameters P_{G}, P_{B}, and $\mathbf{P}^{(\mathrm{T,bit})}$. The error transition matrix $\mathbf{E}^{(\mathrm{T})}$ of the equivalent symbol-based GEC is computed. As shown in Fig. 3.3, there are n_2 identically defined GECs that operate independent of each other. The characteristics of these n_2 GECs are sufficiently defined by a matrix $\mathbf{E}^{(n_2,\mathrm{ch})}$, that is the n_2-fold Kronecker product of $\mathbf{E}^{(\mathrm{T})}$, see (3.2). The basic properties of Kronecker

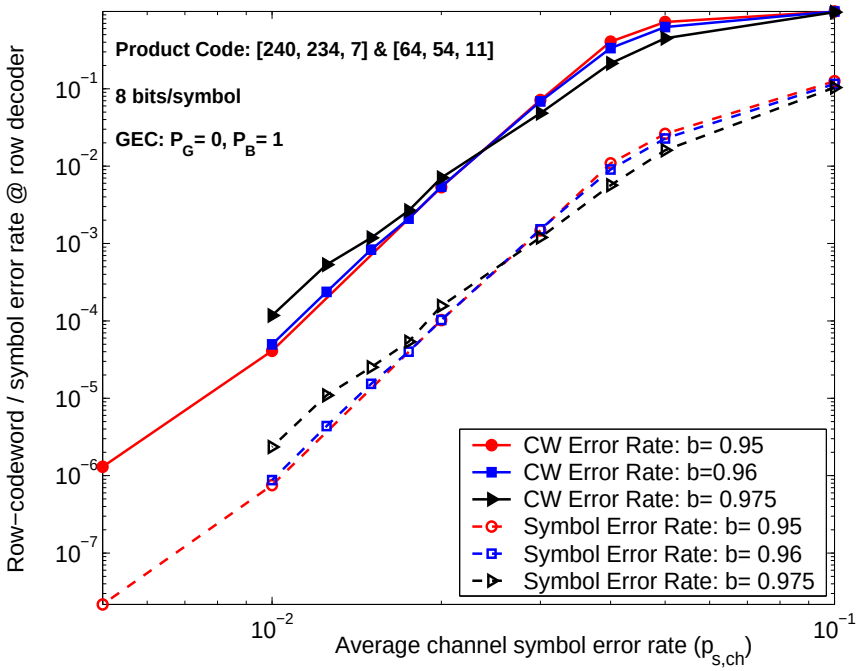


Figure 3.4: $p_{\mathrm{CW},\mathcal{C}^{(1)}}$ and $p_{\mathrm{s},\mathcal{C}^{(1)}}$ of $\mathcal{P}^{(\mathrm{ECMA})}$ over channels with different average burst lengths.

product are listed in A.3.

$$\begin{aligned}\mathbf{E}^{(n_2,\mathrm{ch})} &= \left(\mathbf{E}^{(\mathrm{T})}\right)^{n_2} \\ &:= \underbrace{\mathbf{E}^{(\mathrm{T})} \otimes \mathbf{E}^{(\mathrm{T})} \otimes \cdots \otimes \mathbf{E}^{(\mathrm{T})}}_{n_2 \mathrm{times}} \end{aligned} \tag{3.2}$$

From the convention introduced in 2.2.3, the columns (and rows) of $\mathbf{E}^{(n_2,\mathrm{ch})}$ are labeled using n_2-tuple binary vectors. For example, if $n_2 = 3$, then an element in the $\langle 0,1,0\rangle$-labeled row and $\langle 1,1,0\rangle$-labeled column represents the probability of transition from the error-free state to the error state in the first GEC, from the error state to the error state in the second GEC, and from the error-free state to the error-free state in the third GEC. Unfortunately, the size of $\mathbf{E}^{(n_2,\mathrm{ch})}$ grows exponentially in n_2, and hence it is not feasible to compute this matrix for practical values of n_2.

A reduction in the computation complexity is achieved by realizing that not all information stored in $\mathbf{E}^{(n_2,\mathrm{ch})}$ is necessary for the computation of error-rates. As a first

step, we have grouped the n_2 channels into two groups, namely Group-1 and Group-2. Group-1 contains the first channel, and Group-2 contains all the other channels. A symbol error in Group-1 is corrected by the column-code decoder if and only if there are at most $(t_2 - 1)$ errors in the corresponding column of Group-2. Hence, the Hamming weight of any realization vector of the channels in Group-2 is sufficient for our purposes. In other words, states with *like* Hamming weights in $\left(\mathbf{E}^{(\mathrm{T})}\right)^{n_2-1}$ can be merged to form a single state in $\tilde{\mathbf{E}}^{(n_2-1,\mathrm{ch})}$. Let $\mathcal{F}$ be a function that merges states with *like* Hamming weights such that

$$\tilde{\mathbf{E}}^{(n_2-1,\mathrm{ch})} = \mathcal{F}\left(\left(\mathbf{E}^{(\mathrm{T})}\right)^{n_2-1}\right) \tag{3.3}$$

The computation complexity of this matrix $\tilde{\mathbf{E}}^{(n_2-1,\mathrm{ch})}$, whose dimensions are $n_2 \times n_2$, is dramatically reduced by using the result, see A.4 for proof, given in (3.4).

$$\tilde{\mathbf{E}}^{(n_2-1,\mathrm{ch})} = \mathcal{F}\left(\mathbf{E}^{(\mathrm{T})} \otimes \tilde{\mathbf{E}}^{(n_2-2,\mathrm{ch})}\right),\ i > 1 \tag{3.4}$$

The recursive low-complexity computation technique is as follows:

$$\tilde{\mathbf{E}}^{(n_2,\mathrm{ch})} \ := \ \mathbf{E}^{(\mathrm{T})} \otimes \prod_{i=1}^{n_2-2} \mathcal{F}\left(\mathbf{E}^{(\mathrm{T})} \otimes \tilde{\mathbf{E}}^{(i,\mathrm{ch})}\right), \tag{3.5}$$

where $\tilde{\mathbf{E}}^{(1,\mathrm{ch})} := \mathbf{E}^{(\mathrm{T})}$.

The definition of the channel in Group-1 at the output of the column-code decoder is modified based on the correction capability of $\mathcal{C}^{(2)}$. The new definition $\tilde{\mathbf{E}}^{(n_2,\mathrm{col})}$ of the channel in Group-1 is developed by first observing the column and row arrangements of $\tilde{\mathbf{E}}^{(n_2,\mathrm{ch})} = \mathbf{E}^{(\mathrm{T})} \otimes \tilde{\mathbf{E}}^{(n_2-1,\mathrm{ch})}$ given in (3.6). The first n_2 columns (and rows) of $\tilde{\mathbf{E}}^{(n_2,\mathrm{ch})}$

$$\begin{pmatrix} \tilde{\mathbf{E}}_{1,1}^{(n_2,\mathrm{ch})} & \cdots & \tilde{\mathbf{E}}_{1,n_2+1}^{(n_2,\mathrm{ch})} & \cdots & \tilde{\mathbf{E}}_{1,n_2+t_2}^{(n_2,\mathrm{ch})} & \tilde{\mathbf{E}}_{1,n_2+t_2+1}^{(n_2,\mathrm{ch})} & \cdots & \tilde{\mathbf{E}}_{1,2*n_2}^{(n_2,\mathrm{ch})} \\ \vdots & \ddots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ \tilde{\mathbf{E}}_{n_2+1,1}^{(2*n_2,\mathrm{ch})} & \cdots & \tilde{\mathbf{E}}_{2*n_2,n_2+1}^{(n_2,\mathrm{ch})} & \cdots & \tilde{\mathbf{E}}_{2*n_2,n_2+t_2}^{(n_2,\mathrm{ch})} & \tilde{\mathbf{E}}_{2*n_2,n_2+t_2+1}^{(n_2,\mathrm{ch})} & \cdots & \tilde{\mathbf{E}}_{2*n_2,2*n_2}^{(n_2,\mathrm{ch})} \end{pmatrix} \tag{3.6}$$

correspond to a 0-state in Group-1 channel. The next t_2 columns (and rows) correspond

to a 1-state in Group-1 such that Hamming weights of the Group-2 channels are less than t_2. The rest of the columns (and rows) correspond to a 1-state in Group-1 such that Hamming weights of the Group-2 channels are at least t_2. As mentioned earlier, a symbol in error at the output of the Group-1 channel is corrected by the decoder if and only if there are at most $(t_2 - 1)$ errors in the other tuples of the received word. Also, an error-free symbol at the output of this channel stays error-free at the output of the column-code decoder. Hence, the new definition of the channel in Group-1 is obtained by merging all states corresponding to the first $n_2 + t_2$ columns (and rows) to form an error-free state, and the rest of the states corresponding to the remaining columns (and rows) to form an error-state. Let $\mathbf{E}^{(\mathrm{T,col})}$ be the error-transition matrix of the channel in Group-1 at the output of the column-code decoder.

In [108], Wolf introduced an elegant method to compute the symbol and codeword error-rate performance of ECC over a channel modeled as an HMM. It is straightforward to apply this method on $\mathbf{E}^{(\mathrm{T,col})}$ in order to compute $p_{\mathrm{CW},\mathcal{C}^{(1)}}$ and $p_{\mathrm{s},\mathcal{C}^{(1)}}$ of $\mathcal{C}^{(1)}$. The performance of $\mathcal{P}^{(\mathrm{ECMA})}$, for $b = 0.95$ and other conditions as specified in Sect. 3.3, is computed using the analytical method. In Fig. 3.5, we can observe that the performance computed analytically agree with that computed by running simulations.

3.5 Alternate Error-Correcting Schemes

The analytical computation technique allows to investigate the performance of alternate RS-based product codes for tape drive systems. To design suitable product codes, it is essential to study the effectiveness of increasing the minimum distance of $\mathcal{C}^{(1)}$ to combat very long bursts. To this effect, we considered five different RS codes over GF(2^8) for $\mathcal{C}^{(1)}$, namely $(240, 234, 7)$, $(242, 234, 9)$, $(244, 234, 11)$, $(246, 234, 13)$, and $(248, 234, 15)$, but fixed $\mathcal{C}^{(2)}$ to the one defined in ECMA Standard. The plots in Fig. 3.6 and Fig. 3.7 are analytical result for transmission over a GEC with $P_{\mathrm{G}} = 0$, $P_{\mathrm{B}} = 1$, and values

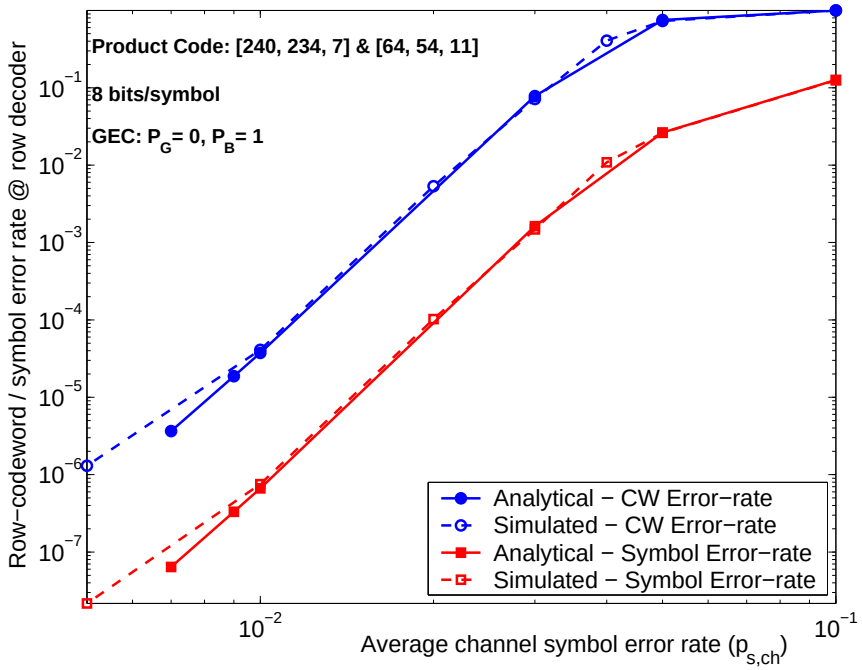


Figure 3.5: Comparison of $p_{\mathrm{CW},\mathcal{C}^{(1)}}$ of $\mathcal{P}^{(\mathrm{ECMA})}$ computed with analytical and simulation methods.

of $b \in \{0.95, 0.97, 0.98, 0.99\}$. The plots are not adjusted for code rates. These plots indicate that when the average burst length increases, the difference in performance between codes with the largest and smallest minimum distances decreases. Another way to improve the performance of the product codes is to use 10-bits-per-symbol RS codes as component codes of the product code. The reason behind this approach is the fact that 10-bits-per-symbol RS codes are more resistant to bursts than their 8-bits-per-symbol counterparts. The first ECC scheme uses a $(386, 374, 13)$ RS code over $\mathrm{GF}(2^{10})$ as $\mathcal{C}^{(1)}$, and a $(51, 43, 9)$ RS code over $\mathrm{GF}(2^{10})$ as $\mathcal{C}^{(2)}$. The second ECC scheme $(480, 468, 13)$ RS code over $\mathrm{GF}(2^{10})$ as $\mathcal{C}^{(1)}$, and a $(64, 54, 11)$ RS code over $\mathrm{GF}(2^{10})$ as $\mathcal{C}^{(2)}$. The third ECC scheme $(480, 468, 13)$ RS code over $\mathrm{GF}(2^{10})$ as $\mathcal{C}^{(1)}$, and a $(128, 108, 21)$ RS code over $\mathrm{GF}(2^{10})$ as $\mathcal{C}^{(2)}$.

Note that the code rates in the second and third schemes are equal to those used specified in ECMA Standard. One of the major criteria in designing an ECC scheme for a

tape drive system is that an erased track should be completely recoverable. This criterion is satisfied by the second and third schemes. In order to evaluate the performance of all three schemes, we assume an ECC subsystem model similar to the one described in Sect. 3.1. Also, the parameters of the GEC models are $P_{\mathrm{G}} = 0$, $P_{\mathrm{B}} = 1$, and values of $b \in \{0.95, 0.96, 0.97, 0.98, 0.99\}$. The performance plots of these schemes are shown in Fig. 3.8, Fig. 3.9, and Fig. 3.10. Once again, these plots are not adjusted for code rates.

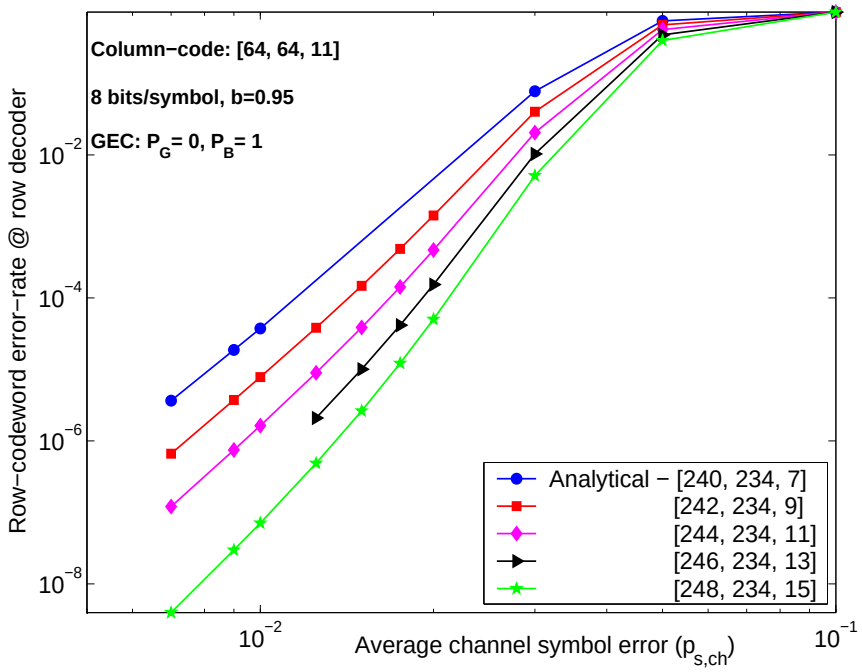
From the plots, it is obvious that these schemes perform better than the one specified in ECMA Standard. But, to make a definitive statement one has to investigate the processing overhead imposed by these new schemes, and also devise a procedure to adjust code rates where necessary to make fair comparisons.

3.6 Discussion

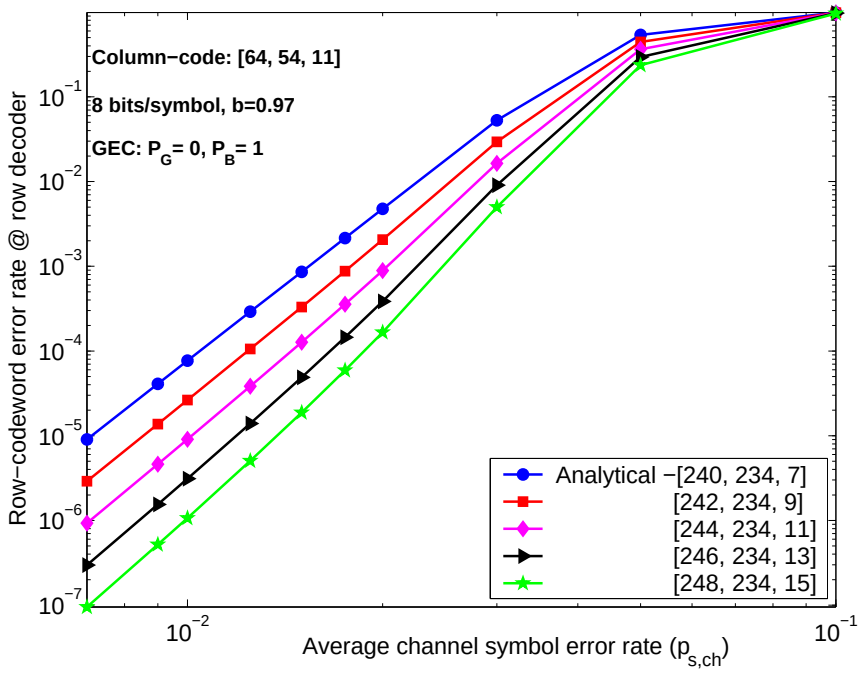
In this chapter, we developed a model of the ECC subsystem of a magnetic tape drive system in order to investiage its performance. Two computation techniques were proposed to compute the codeword and symbol error-rates of product codes over channels with memory. The first technique involved describing the ECC subsystem model in C, and running simulations to compute the error-rate performances. The time-complexity of this technique at high SNR is prohibitive. The second technique is an analytical method to compute the performance, and this alleviates the time-complexity issue. This method makes it practical to compute codeword error rates as low as 10^{-12}. The decoding algorithm considered in this chapter can be iteratively applied to improve the performance of product codes. The analytical method can be easily extended to deal with such an iterative hard-decision decoding scheme.

A shortcoming of the proposed techniques is that they do not consider the possibility of a decoder error which might result in a burst of errors. This issue has to be addressed

in future research. Finally, it is well know that the accuracy of the error-rates is critically dependent on the accuracy of the ECC subsystem model. Hence, experimental results are necessary to design a HMM that serves as a better approximation of the magnetic tape channel.

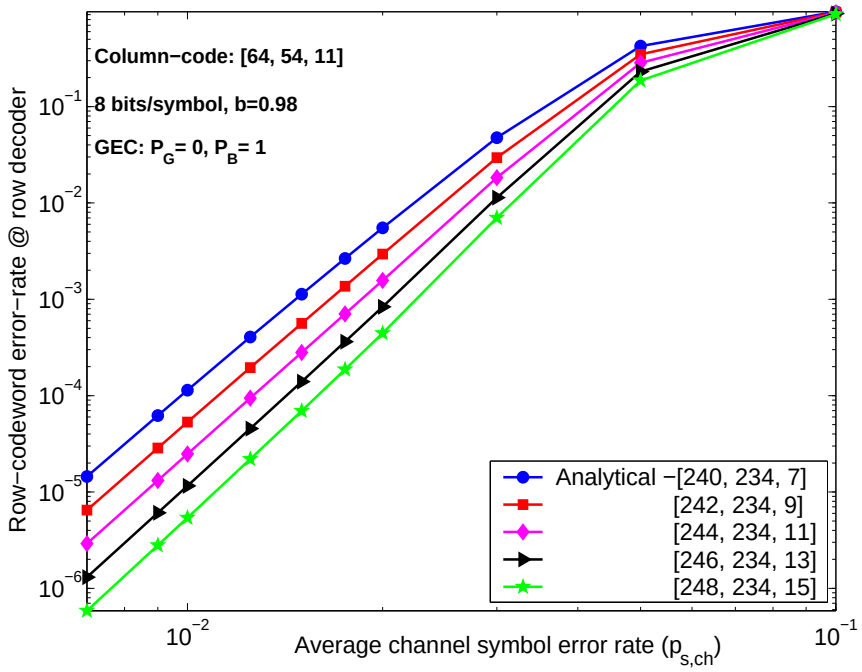


(a) $b = 0.95$

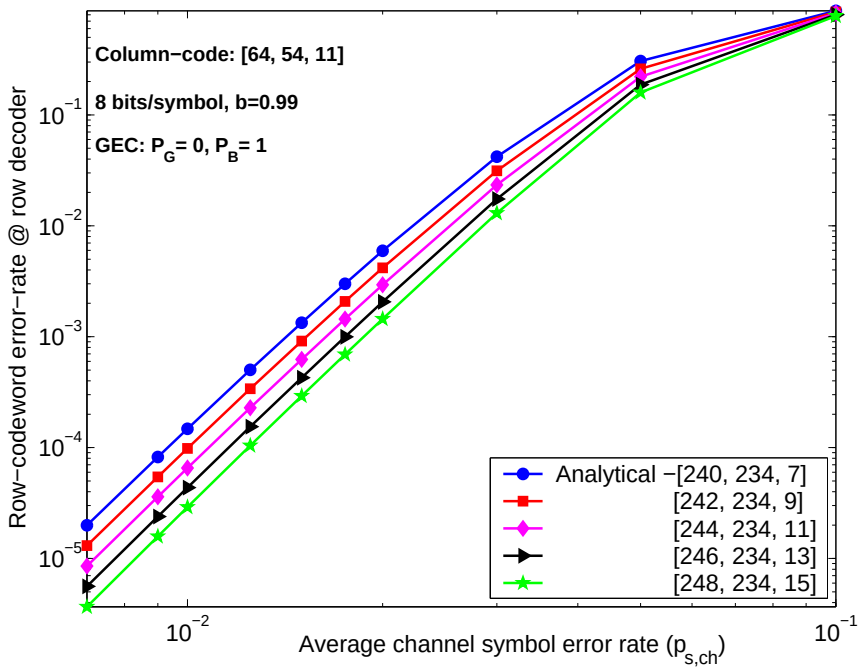


(b) $b = 0.97$

Figure 3.6: $p_{\mathrm{CW},\mathcal{C}^{(1)}}$ of product codes with different row codes for various burst conditions.



(a) $b = 0.98$



(b) $b = 0.99$

Figure 3.7: $p_{\mathrm{CW},\mathcal{C}^{(1)}}$ of product codes with different row codes for various burst conditions.

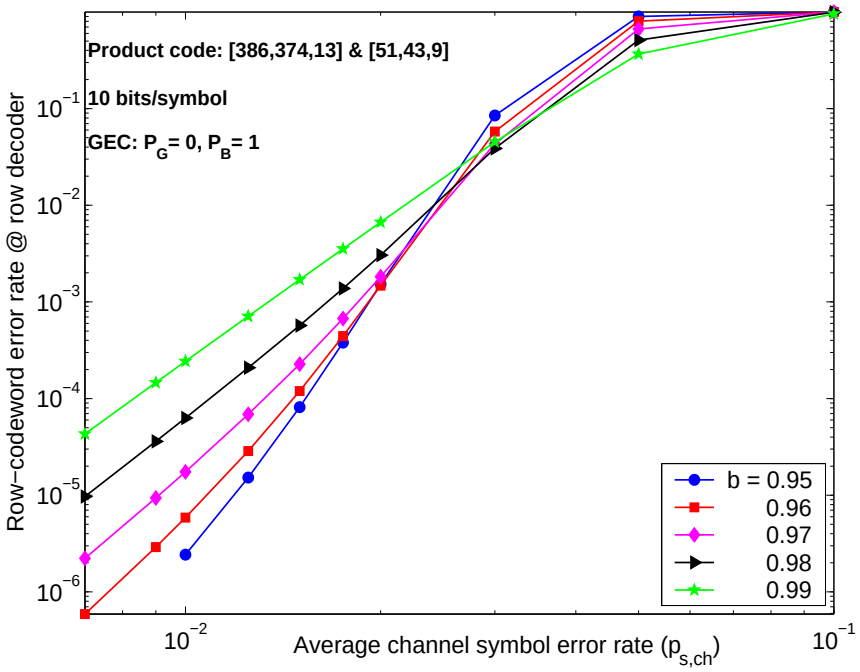


Figure 3.8: $p_{\mathrm{CW},\mathcal{C}^{(1)}}$ of the first ECC scheme for various burst conditions.

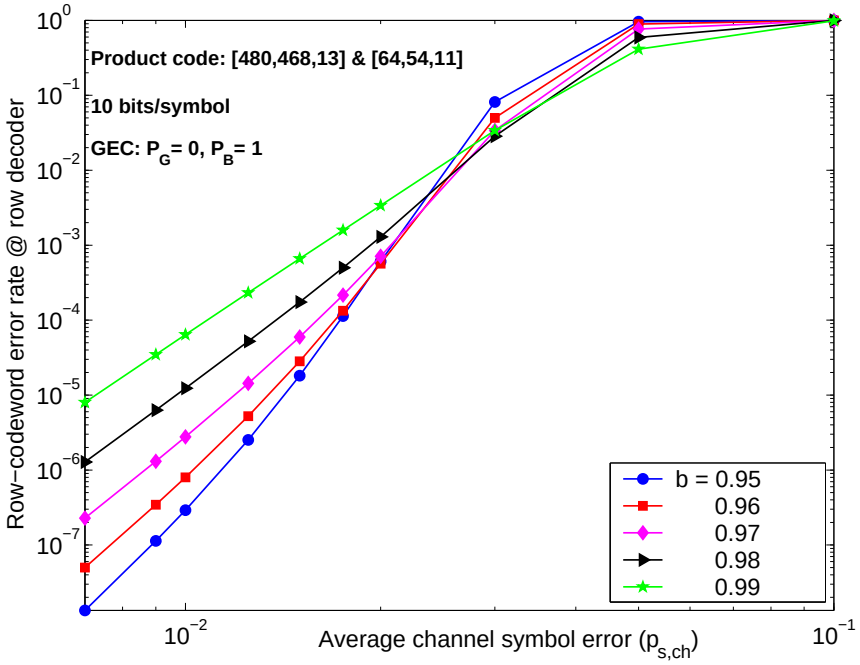


Figure 3.9: $p_{\mathrm{CW},\mathcal{C}^{(1)}}$ of the second ECC scheme for various burst conditions.

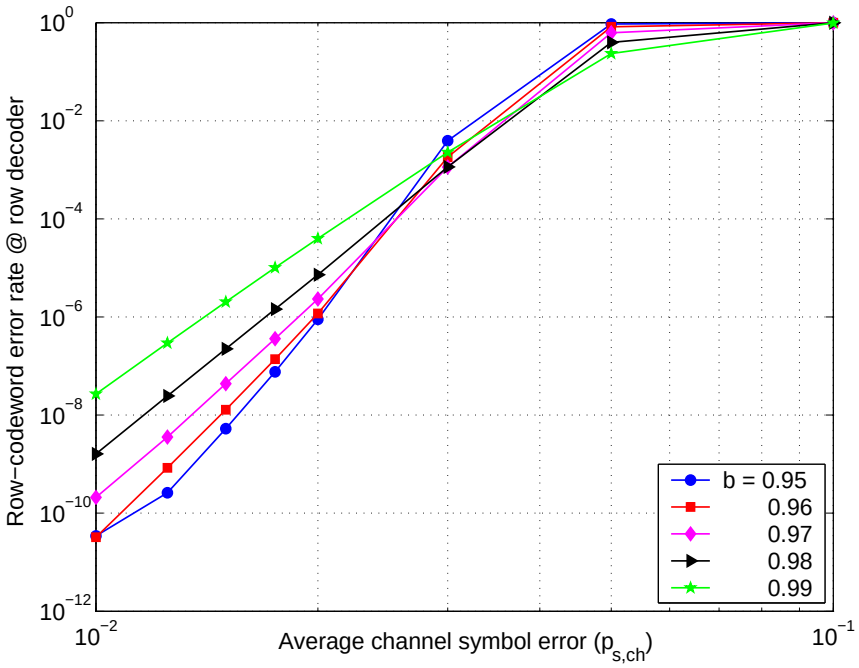


Figure 3.10: $p_{\mathrm{CW},\mathcal{C}^{(1)}}$ of the third ECC scheme for various burst conditions.

CHAPTER 4

Construction of Regular and Irregular LDPC Codes

The increasing popularity of LDPC codes is justified by their excellent error rate performance when iteratively decoded using one of the message-passing algorithms. In general, the complexity of such message-passing algorithms are significantly smaller than that of most conventional distance-bounded decoders.These characteristics of the class of codes make it attractive for error control applications in communication systems. Researchers are investigating the possibilities of using LDPC codes for error control in magnetic storage systems, wireless communication systems, optical communication systems, optical storage systems etc.

It is not surprising that construction of practical LDPC codes involves considerations that go beyond mere error rate performance. Simplicity in LDPC code representation, complexity of the encoding process, complexity of the decoding process, performance gain over the current ECC system are some of the considerations in construcing a practical LDPC code. The simplicity in LDPC code representation, generally, translates to cyclic or quasi-cyclic properties of the parity-chcek matrix. Regular and irregular LDPC codes from combinatorial designs exhibit structures that aid in hardware implementations. Also, these codes have good minimum distance and girth properties that contribute in improving its suitability for iterative decoders. The regular codes of our interest are constructed from incidence structures in finite geometries, and these structures are members of a special class of combinatorial designs called the balanced incomplete block designs. The chapter begins with a brief introduction to these combinatorial objects.

The works of Luby, Mitzenmacher, Richardson, Shokrollahi, and Urbanke [50], [74], [75] show that node-degree distributions have an impact on the performance of the iterative decoder. Graphs with certain degree distributions have been proven to achieve capacity in the limit of increasing codelength and number of iterations. In this chapter, we propose a construction method for irregular LDPC codes that retain some of the structural properties of the systematic regular codes. This is achieved by carefully splitting rows and columns of a suitable regular LDPC code.

Finally, we present simulation results on the performance of the regular and irregular LDPC codes decoded using the sum-product algorithm. The performance of these codes on the AWGN channel and PR equalized channel is presented.

4.1 Balanced Incomplete Block Design

A *balance incomplete block design* (BIBD) is defined as a collection of k-subsets of a v-set P, $k < v$, such that pair of elements of P occur together in exactly l of the blocks [3]. Each k-subset is called a *block* and each element of P is called a *point*. The design is said to be *balanced* because each pair of elements of P occur together in exactly l of the blocks and it is said to be *incomplete* because not all possible k-subsets of P are elements of the set of blocks. BIBD is referred to as a design with parameters 2-(v, k, l). A BIBD is *resolvable* if the blocks can be partitioned into $r = \frac{bk}{v}$ groups such that each group contains blocks whose pairwise intersection is a disjoint set and their union is the set of points P. The groups are called the *resolution classes* or the *parallel classes*. Geometrically, a resolution class can be viewed as a collection of parallel lines. Further details about designs in general and BIBD in specific can be found in [4], [3] and [25].

The *incidence matrix* of a 2-(v, k, l) design with b blocks, is a $b \times v$ matrix $\mathbf{A} = (a_{ij})$

such that

$$a_{ij} = \begin{cases} 1, & \text{if } i^{th} \text{ block contains the } j^{th} \text{ point;} \\ 0, & \text{otherwise.} \end{cases}$$

The parity-check matrix $\mathbf{H}$ of a code from BIBD is the transpose of the incidence matrix.

$$\mathbf{H} = \mathbf{A}^{\mathrm{T}}$$

The parity-check matrix obtained from the design has uniform column and row weights. If l of BIBD is unity, then the resultant code has no four cycles. The generator matrix $\mathbf{G}$ of the code can be obtained by performing Gaussian elimination on $\mathbf{H}$.

4.2 Codes from Projective Geometry

The family of regular LDPC codes from finite projective geometry is constructed from incidence structures in the geometry. For example, point-line incidence, point-plane incidence, and point-hyperplane incidence of the geometry results in regular LDPC codes that belong to the family. In addition, regular LDPC codes can be constructed from special arc-line incidence of the goemetry. We make a distinction between codes from projective planes and higher dimension geometries for purposes of clarity. The codes from projective planes include codes based on ovals and unitals. The construction of LDPC codes from projective geometries was proposed by [55] and [91]. We acknowledge the results of Weller and Johnson [105] on oval and unital designs that was published around the same time as ours.

4.2.1 Codes on Projective Planes

A finite projective plane of order q, say PG$(2, q)$, is a $2 - (q^2 + q + 1, q + 1, 1)$ design, $q \geq 2$ and $q = p^m$ for some prime p. The point set of the design of all *points* in PG$(2, q)$

and the block set of the design consists of all *lines* in PG$(2, q)$. The points and lines of a finite projective plane satisfy the following set of axioms:

1. Every line consists of the same number of points.

2. Any two points in the plane are connected by a unique line.

3. Any two lines in the plane intersect at a unique points.

4. A fixed number of lines pass through any point in the plane.

As mentioned above, since any two lines in a projective plane intersect at a unique point, there are no parallel lines in the plane. The incidence matrix of such a design, for $q = 2^m$, is cyclic and hence any row of the matrix can be obtained by shifting (right or left) another row of the matrix.

The points in PG$(2, q)$ are 3-tuples $\mathbf{x} = \langle x_0, x_1, x_2 \rangle$, where $x_i \in \mathrm{GF}(q)$ not all simultaneously equal to zero. Any two 3-tuples $\mathbf{x}$ and $\mathbf{y}$ represent the same point if $\mathbf{y} = \lambda \mathbf{x}$ for some $\lambda \in \mathrm{PG}(q)$. So, each point can be represented in $q - 1$ ways, and a set of these representations is called the *equivalence class*. It is obvious that the number of points in PG$(2, q)$ is $v = \frac{q^3-1}{q-1}$. For every point $\langle \alpha_0, \alpha_1, \alpha_2 \rangle$ in PG$(2, q)$, there is a homogeneous equation $\alpha_0 x_0 + \alpha_1 x_1 + \alpha_2 x_2 = 0$. A line, represented as $[\alpha_0, \alpha_1, \alpha_2]$, in the plane is a set of points $\langle x_0, x_1, x_2 \rangle$ that satisfy the homogeneous equation. In the projective plane, for every point $\langle \alpha_0, \alpha_1, \alpha_2 \rangle$ there exists a unique line $[\alpha_0, \alpha_1, \alpha_2]$ and for every line $[\alpha_0, \alpha_1, \alpha_2]$ there exists a unique point $\langle \alpha_0, \alpha_1, \alpha_2 \rangle$. Hence, the number of lines $b = \frac{q^3-1}{q-1}$ in the plane is equal to the number of points. Also, each line has $k = q + 1$ points and each point has $q + 1$ lines passing through it.

The parity-check matrix $\mathbf{H}$ of an LDPC code is obtained from the incidence matrix of points and lines. The matrix $\mathbf{H}$ has a column-weight and row-weight of $q + 1$ and the minimum distance is at least $q + 2$. The girth of the code is 6 because two lines intersect

at exactly one point. For example, the parity-check matrix of a code from PG$(2,3)$ is

$$\mathbf{H} = \begin{pmatrix}
1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \\
0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 \\
0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 \\
0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 1 \\
0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\
0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 0
\end{pmatrix}.$$

The code has $\gamma = \rho = 3 + 1 = 4$.

4.2.2 Codes on Oval Designs

Ovals [4], [5], [6], [95] are special geometrical structures namely *arcs* in a projective plane. In PG$(2,q)$, any non-empty set of r points is a $\{r,s\}$-arc, $s \neq 0$, if s is the maximum number of collinear points in the set. A *hyperoval* in PG$(2,q)$ for q even is a $\{q+2,2\}$-arc. The lines in the plane can be divided into the following three classes with respect to the hyperoval:

1. A *tangent* is a line that meets the arc in exactly one point.

2. A *secant* is a line that meets the arc in exactly two points.

3. An *exterior line* is a line that does not meet the arc.

The *nucleus* of a hyperoval is a unique point through which all tangents pass through.

An *algebraic curve* is a collection of points that satisfy a fixed homogeneous equation $f(x_0, x_1, x_2) = 0$. An algebraic curve is *irreducible* if $f(x_0, x_1, x_2)$ is irreducible over GF(q). An irreducible algebraic curve of degree 2 is an *irreducible conic*. A hyperoval can be characterized algebraically as a set of points that satisfies an irreducible conic along with the nucleus of the arc. An *irreducible* conic is a homogenous algebraic function of degree 2 that is irreducible in the Galois field of order n. In general, a conic in GF(q) is of the form

$$f(x_0 = x, x_1 = y, x_2 = z) = ax^2 + by^2 + cz^2 + fxy + gxz + hyx = 0,$$

for coefficients in GF(q).

In PG$(2, q)$, for $q = 2^m$, an *oval design* is the incidence having for points the lines exterior to the hyperoval and for blocks the points not on the hyperoval. The oval design is a $2-(\frac{q(q-1)}{2}, \frac{q}{2}, 1)$ BIBD whose incidence matrix has a rank of $3^m - 2^m$. In addition, the oval design is a resolvable design. The parity-check matrix of an LDPC code is defined as the transpose of the incidence matrix. For example, the parity-check matrix of a code from a hyperoval in PG$(2, 2^2)$ is

$$\mathbf{H} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \end{pmatrix}.$$

Since any two points are connected by exactly one line, no two rows or columns have more than one 1 in common. Therefore, the parity-check matrix and the corresponding Tanner graph is free of cycles of length 4.

4.2.3 Codes on Unitals

A *Hermitian arc* or *unital* in PG$(2, q)$ for q square is a $\{q\sqrt{q}+1, \sqrt{q}+1\}$-arc such that any line intersects the arc in either 1 or $\sqrt{q}+1$ points. A line meeting the unital in exactly one point is called a tangent. A unital can be characterized algebraically as a set of points that satisfies an irreducible homogeneous function of order $\sqrt{q}+1$ given by

$$x_0^{\sqrt{q}+1} + x_1^{\sqrt{q}+1} + x_1^{\sqrt{q}+1} = 0.$$

At each point on the arc there is a unique tangent line. The Hermitian arc together with the non-tangent lines form a $2-(q\sqrt{q}+1, \sqrt{q}+1, 1)$ BIBD. The rank of an incidence matrix of the design is $q^{\frac{3}{2}}$, if q is a power of 2, and $\sqrt{q}(q^{\frac{3}{2}} - \sqrt{q} + 1)$, if q is a power of odd prime [4], [5], [6], [95]. The parity-check matrix of an LDPC code constructed from the unital design is defined as the transpose of the incidence matrix of the design. For example, the parity-check matrix from a unital in PG$(2, 2^2)$ is

$$\mathbf{H} = \begin{pmatrix} 0 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \end{pmatrix}.$$

The girth of the code is 6.

4.2.4 Codes on Projective Geometry

The projective geometry [10] of dimension n over GF(q), represented as PG(n, q), is a set of all proper subspaces of a vector space of dimension $n+1$ over GF(q). In projective geometric terminology, an $(m+1)$-dimensional element of the set is referred to as an m-flat. For example, points are 0 flats, lines are 1 flats, planes are 2 flats, and m-dimensional spaces are $(m-1)$ flats. An l-dimension flat is said to be contained in an m-dimension flat if it satisfies a set-theoretic containment. Since any m flat is a finite vector subspace, it is justified to talk about a basis set composed of basis elements independent in GF(q). The basis element of a 0 flat is referred to as a point and, generally, this should cause no confusion because this basis element is representative of the 0 flat.

Using the above definition and linear algebraic concepts, it is straightforward to count the number of m flats, defined by a basis with $m+1$ points, in PG(n, q). The number of such flats, say $N_{\mathrm{PG}}(m, n, q)$, is the number of ways of choosing $m+1$ independent points in PG(n, q) divided by the number of ways of choosing $m+1$ independent points in any m flat.

$$\begin{aligned} N_{\mathrm{PG}}(m,n,q) &= \frac{(q^{n+1}-1)(q^{n+1}-q)(q^{n+1}-q^2)\dots(q^{n+1}-q^m)}{(q^{m+1}-1)(q^{m+1}-q)(q^{m+1}-q^2)\dots(q^{m+1}-q^m)} \\ &= \prod_{i=0}^{m} \frac{q^{n+1-i}-1}{q^{m+1-i}-1} \end{aligned}$$

Hence, the number of 0 flats in PG(n, q) is $N_{\mathrm{PG}}(0, n, q) = \frac{q^{n+1}}{q-1}$, and the number of 1 flats is $N_{\mathrm{PG}}(1, n, q) = \frac{(q^{n+1}-1)(q^n-1)}{(q^2-1)(q-1)}$. When $n = 2$, the number of points is equal to the number of lines, and this agrees with dimensions of point-line incidence matrices of projective plane codes, introduced in Section 4.2.1. In Section 4.2.1, we considered point-line incidence matrices to construct cyclic LDPC code, while in this section, we consider the incidence of m flats and l flats in PG(n, q).

An algorithm to construct and m flat in PG(n, q) uses the fact that the elements of GF(q^{n+1}) can be used to represent points in the plane. If α is the primitive element

of GF(q^{n+1}), the α^v, where $v = \frac{q^{n+1}-1}{q-1}$, is a primitive element of GF(q). Each one of the first v powers of α can be taken as the basis of one of 0 flats in PG(n, q). In other words, if α^i is the basis of a 0 flat in PG(n, q), then every α^k, such that $i \equiv k \bmod v$, is contained in the subspace, In a similar fashion, a set of $m + 1$ powers of α that are independent in GF(q) forms a basis of an m flats. If $\alpha^{s_1}, \alpha^{s_2}, \ldots, \alpha^{s_{m+1}}$ is a set of $m+1$ basis elements of an m flat, then any point in the flat can be written as

$$\zeta_i = \sum_{j=1}^{m+1} \epsilon_{ij} \alpha^{s_j},$$

where the ϵ_{ij} are chosen such that no two vectors $\langle \epsilon_{i1}, \epsilon_{i2}, \ldots, \epsilon_{i(m+1)} \rangle$ are linear multiples over GF(q). Now, every ζ_i can be equivalently written as a power of the primitive element α.

To construct an LDPC code, we generate all m flats and l flats in PG(n, q) using the method described above. An incidence matrix of m flats and l flats in PG(n, q) is a binary matrix with $N_{\mathrm{PG}}(m, n, q)$ rows and $N_{\mathrm{PG}}(l, n, q)$ columns. The (i, j)-th element of the incidence matrix $\mathbf{A}_{\mathrm{PG}}(m, l, n, q)$ is a one if and only if the j^{th} l flat is contained in the i^{th} m flat of the geometry. An LDPC code is constructed by considering the incidence matrix or its transpose as the parity-check matrix of the code. It is widely accepted that the performance of an LDPC code under an iterative decoder is deteriorated by the presence of four cycles in the Tanner graph [59] of the code. In order to avoid four cycles, we impose an additional constraint that l should be one less than m [91]. If l is one less than m, then no two distinct m flats have more than one l-dimensional subpsace in common. This guarantees that the girth of the graph is 6.

Example 3 (Incidence matrix of 2 flats and 1 flats)**.** *Let us construct the incidence matrix of* 2 *flats and* 1 *flats in PG*$(3, 2)$*. For some primitive element* α *of GF*(2^4)*, each element of the set* $\{\alpha^0, \alpha^1, \alpha^2, \ldots, \alpha^{14}\}$ *is a generator of a unique* 0 *flat in the geometry, and this agrees with the fact that there are* $N_{PG}(0, 3, 2) = 15$ *points in the geometry.*

$$\mathbf{A}_{\mathrm{PG}}(2,1,3,2) = \begin{pmatrix}
1&1&0&0&1&0&0&0&0&0&0&0&0&0&0\\
0&1&1&1&0&0&0&0&0&0&0&0&0&0&0\\
0&0&1&0&1&0&0&0&0&0&0&0&1&0&0\\
0&1&0&0&0&1&0&0&0&0&0&0&1&0&0\\
1&0&1&0&0&1&0&0&0&0&0&0&0&0&0\\
1&0&0&1&0&0&0&0&0&0&0&0&1&0&0\\
0&0&0&1&1&1&0&0&0&0&0&0&0&0&0\\
0&1&0&0&0&0&0&1&1&0&0&0&0&0&0\\
1&0&0&0&0&0&1&1&0&0&0&0&0&0&0\\
0&0&0&0&1&0&0&1&0&0&0&0&0&0&1\\
0&1&0&0&0&0&1&0&0&0&0&0&0&0&1\\
1&0&0&0&0&0&0&0&1&0&0&0&0&0&1\\
0&0&0&0&1&0&1&0&1&0&0&0&0&0&0\\
1&0&0&0&0&0&0&0&0&1&1&0&0&0&0\\
0&0&1&0&0&1&0&1&0&1&0&0&0&0&0\\
0&0&0&0&0&1&1&0&0&1&0&0&0&0&0\\
0&0&0&0&0&0&0&1&0&0&1&0&0&0&0\\
0&0&0&1&0&0&0&1&0&0&0&0&0&1&0\\
0&1&0&0&0&0&0&0&0&1&0&0&0&1&0\\
0&0&1&0&0&0&0&0&1&0&0&0&0&1&0\\
0&0&0&1&0&0&0&0&1&1&0&0&0&0&0\\
0&0&0&0&1&0&0&0&0&1&0&1&0&0&0\\
1&0&0&0&0&0&0&0&0&0&0&1&0&1&0\\
0&1&0&0&0&0&0&0&0&0&1&1&0&0&0\\
0&0&0&0&1&0&0&0&0&0&1&0&0&1&0\\
0&0&0&0&0&0&1&0&0&0&0&0&1&1&0\\
0&0&0&0&0&0&0&1&0&0&0&1&1&0&0\\
0&0&0&0&0&0&1&0&0&1&0&1&0&0&0\\
0&0&0&1&0&0&0&0&0&0&0&0&1&0&1\\
0&0&1&0&0&0&0&0&0&0&0&1&0&0&1\\
0&0&0&0&0&1&0&0&0&0&0&0&0&1&1\\
0&0&0&0&0&1&0&0&1&0&0&1&0&0&0\\
0&0&0&0&0&0&0&0&1&0&1&0&1&0&0\\
0&0&0&1&0&0&0&0&0&0&1&0&0&0&1\\
0&0&1&0&0&0&1&0&0&0&1&0&0&0&0
\end{pmatrix}. \tag{4.1}$$

There are $N_{PG}(0,2,2) = 7$ points in each 2 flat of the geometry, and all points in one of those flats can be generated as linear combinations of $\{\alpha^0, \alpha^1, \alpha^3\}$ in GF(2). By working out all possible linear combinations, one arrives at $\{\alpha^0, \alpha^1, \alpha^3, \alpha^4, \alpha^7, \alpha^9, \alpha^{14}\}$ as the set of all points in the 2 flat. Using the above procedure, all 2 flats and 1 flats in the geometry can be generated. The incidence matrix of the 2 flats and 1 flats in PG(3, 2) is given in Eq. (4.1). There are no four cycles in the Tanner graph of the incidence matrix. This matrix or its transpose will be used as the parity-check matrix of a projective geometry code.

4.3 Codes from Affine Geometry

Similar to the projective geometry case, a family of regular LDPC codes based on incidence structures can be constructed from affine geometry. Once again, we start with a discussion on codes from affine planes and then continue with the generalization of the concept.

4.3.1 Codes on Affine Planes

A finite affine plane of order q, say $\mathrm{AG}(2, q)$, is a $2-(q^2, q, 1)$ design, $q \geq 2$ and $q = p^m$ for some prime p. A finite affine plane of order q is a special case of $\mathrm{PG}(2, q)$. In $\mathrm{AG}(2, q)$ any two lines in the plane need not intersect and hence, leading to the concept of *parallelism* in the affine plane. The $\mathrm{AG}(2, q)$ can be constructed by "discarding" all points in a line of $\mathrm{PG}(2, q)$. The chosen line is "discarded" as well, and it is referred to as the *line at infinity*. Hence, the number of lines in the affine plane is one less than that in $\mathrm{PG}(2, q)$, and each line has one less point than that in the projective plane.

The parity-check matrix $\mathbf{H}$ of an LDPC code is obtained from the incidence matrix of points and lines. The dimensions of $\mathbf{H}$ are $q^2 \times q(q+1)$. The code has $\gamma = q$ and $\rho = q$ and the minimum distance is at least $q+1$. The girth of the code is 6. For example, the

parity-check matrix of a code from AG$(2, 3)$ is

$$\mathbf{H} = \begin{pmatrix} 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 0 \end{pmatrix}.$$

The code has $\gamma = 3$ and $\rho = 4$.

A compilation of parameters of code families from finite planes is presented in Table 4.1. In Table 4.2, we list the properties of the codes from finite planes of order $q = 2^m$. The table contains lower bounds on the minimum distance and the ranks of codes from finite planes.

Table 4.1: FINITE GEOMETRY $2 - (v, k, 1)$ FAMILIES

k	v	Parameter	Designs
$q+1$	q^2+q+1	q-a prime power	Projective planes
q	q^2	q-a prime power	Affine planes
$\frac{q}{2}$	$\frac{q(q-1)}{2}$	q- a prime power and even	Oval designs
$\sqrt{q}+1$	$q\sqrt{q}+1$	q- a prime and square	Unitals

Table 4.2: PROPERTIES OF FINITE GEOMETRY CODES WITH $q = 2^m$

Designs	Rank of $\mathbf{H}$	Lower bound on $d_{\min}$
Projective planes	$3^m + 1$	$2^m + 2$
Affine planes	3^m	$2^m + 1$
Oval designs	$3^m - 2^m$	$2^{m-1} + 1$
Unitals	$2^{\frac{3m}{2}}$	$2^{\frac{m}{2}} + 2$

4.3.2 Codes on Affine Geometry

The affine geometry [10] of dimension n over GF(q), represented as AG(n, q) is a set of all cosets of the proper subspaces of a vector space of dimension n over GF(q). A coset of a subspace U is defined as the set $\mathbf{x} + U$, where $\mathbf{x}$ is any vector in the space of dimension n over GF(q), and its dimension is equal to the dimension of U. Unlike the projective geometry, an m-dimensional element of the set is referred to as an m flat. For example, points are 0 flats, lines are 1 flats, and planes are 2 flats. Note that points are, in fact, cosets of the unique zero-dimension subspace. Similar to the projective geometry, an l-dimension flat is said to be contained in an m-dimension flat if it satisfies a set-theoretic containment.

One of the interesting propoerties of this geometry is the concept of parallelism. It can be shown that cosets of the same dimension are parallel if and only if they are cosets of the same subspace [10]. The construction of AG(n, q) is simplified by the *fundamental embedding theorem*. The affine geometry of dimension n is obtained from a projective geometry of dimension n by removing a hyperplane and all subspaces contained in it. In other words, to create an m flat in AG(n, q) follow the procedure given below.

1. Construct a hyperplane in PG(n, q).

2. Construct an m flat in PG(n, q).

3. Remove all points in the hyperplane contained in the m flat. In order to construct all flats in AG(n, q), fix the hyperplane in PG(n, q) and remove all points in the hyperplane from all flats in PG(n, q). An incidence matrix of m flats and l flats in AG(n, q) is defined similar to the projective-geometry case. Also, the incidence matrix or its transpose can be used as a parity-check matrix that defines an LDPC code. Also, m is constrained to be on more than l in order to avoid four cycles in the resultant LDPC code.

4.4 Construction of Capacity-Achieving Irregular LDPC Codes

In [50], Luby, Mitzenmacher et. al designed LDPC codes from irregular graphs for erasure channels and have shown that using irregular graphs yields codes with much better performance than regular graphs. In [74] and [75], Richardson, Shokrollahi and Urbanke extended the above results to encompass binary AWGN channel. In [75], Richardson et. al have listed good degree distribution pairs(λ, ρ), of rate one-half code with various upper-bounds on the maximum variable degree d_v such that $\lambda_i \in \lambda$ represents the percentage of variable nodes with degree i and $\rho_j \in \rho$ represents the percentage of check nodes with degree j. LDPC codes constructed with large codeword length that satisfy the given degree distribution pairs are observed to perform very close to the Shannon capacity limit.

In this section, we propose an algorithm that enumerates a systematic method to achieve a given good degree distribution pair from a regular LDPC code using designs. This algorithm involves splitting columns and rows of $\mathbf{H}$ of the regular code in a systematic manner to achieve a desired degree distribution pair $(\lambda^{\text{req}}, \rho^{\text{req}})$. The following subsection is intended to serve as an introduction to the theory of linear diophantine systems which forms an important component of the algorithm.

4.4.1 Linear Diophantine System

A *linear diophantine system* is an equation $\mathbf{Ax} = \mathbf{c}, \mathbf{x} \in \mathbb{N}^n$ where $\mathbf{A}$ is an $m \times n$ integer matrix and $\mathbf{c} \in \mathbb{Z}^m$. If $\mathbf{c} = \bar{\mathbf{0}}$, the diophantine system is *homogeneous*, otherwise it is *inhomogeneous* [32]. Given $\mathbf{A}$ and $\mathbf{c}$, the existence of $\mathbf{x}$ is not assured. In [32], Domenjoud has given necessary conditions which are not sufficient and sufficient conditions which are not necessary for the existence of such a solution. In [32], Domenjoud has not given necessary and sufficient conditions to test the existence of solutions of an inhomogeneous

diophantine system.

Theorem 4 (Domenjoud's Theorem-1)**.** *The inhomogeneous diophantine system* $\mathbf{Ax} = \mathbf{c}, \mathbf{x} \in \mathbb{N}^n$*, has a solution only if* $\mathbf{Ax} = \mathbf{c}, \mathbf{x} \in \mathbb{Z}^n$*, has a solution and* $\exists\ \ell \in \mathbb{N}$ *such that* $\frac{1}{\ell} \in$ *Conv(*$\mathbf{A}^1, \mathbf{A}^2, \ldots, \mathbf{A}^n$*), where* $A^j, \forall j = 1, \ldots, n$ *denote the columns of* $\mathbf{A}$*.*

Theorem 5 (Domenjoud's Theorem-2)**.** *If* $\mathbf{Ax} = \mathbf{c}, \mathbf{x} \in \mathbb{Z}^n$ *has a solution and* $\forall i$*,* $\mathbf{Ax} = \mathbf{c}, \mathbf{x} \in \mathbb{N}^n$ *has a solution such that* $x_i \neq 0$*, then* $\mathbf{Ax} = \mathbf{c}, \mathbf{x} \in \mathbb{N}^n$ *has a solution.*

Assuming A has been preprocessed to have full row rank, a solution for $\mathbf{Ax} = \mathbf{c}, \mathbf{x} \in \mathbb{Z}^n$, exists *iff* $\mathbf{L}^{-1}\mathbf{b}$ is integral, where $\mathbf{L}$ is nonsingular and lower triangular with nonnegative entries such that [15]

$$\text{Hermite Normal Form}(\mathbf{A}) = \mathbf{AK} = [\mathbf{L}|\mathbf{0}].$$

In the following discussion, the system of linear diophantine equations is assumed to have at least one solution. The inhomogeneous system can be transferred to a homogeneous system of equations by adding a variable x_{n+1} such that,

$$\begin{pmatrix} \mathbf{A} & -\mathbf{c} \end{pmatrix} \begin{pmatrix} \mathbf{x} \\ x_{n+1} \end{pmatrix} = \bar{\mathbf{0}} \tag{4.2}$$

It is easy to observe that the homogeneous system is equivalent to the inhomogeneous system when, $x_{n+1} = 1$. The homogeneous system above can be solved using the idea of Contejean and Devie [23] [22]. The set of solutions (generally infinite in number) of the homogeneous system can always be represented by a finite set $\mathcal{B}$ called a *basis*. By definition, every solution of the system can be extracted by linear combination of the solutions in $\mathcal{B}$ by using natural numbers for coefficients. The incremental algorithm for solving linear diophantine equations, given by Contejean and Devie, referred to as *ISLDE-algorithm*, can find the basis in finite time.

The homogeneous system with m equations and $n+1$ variables may be written as

$$a(\langle \mathbf{x}, x_{n+1}\rangle) = x_1 a(\mathbf{e_1}) + \ldots + x_n a(\mathbf{e_n}) + x_{n+1} a(-\mathbf{c}) \text{ such that} \tag{4.3}$$
$$a(\langle \mathbf{x}, x_{n+1}\rangle) = \bar{\mathbf{0}} \text{ if } \langle \mathbf{x}, x_{n+1}\rangle \text{ is a solution,}$$

where $\mathbf{e_j}, \forall j = 1, 2, \ldots, n$ is the canonical basis of $\mathbb{N}^n$ and $a(\mathbf{e_j})$ is the j^{th} column of $\mathbf{A}$. The vector $a(-\mathbf{c})$ is same as the vector $-\mathbf{c}$. The vectors $a(\mathbf{e_j}), \forall j = 1, 2, \ldots, n$ and $a(-\mathbf{c})$ are referred to as the *default vectors*. Any solution of the above system can be seen as a collection of default vectors whose sum is $\mathbf{0}$. The idea underlying the ISLDE-algorithm is to choose an arbitrary order of default vectors to construct a sequence of default vectors starting from the origin and returning to it. By restricting the use of the default vector $a(-\mathbf{c})$ only once in the sequence, we can force a solution of the homogeneous system to be a solution of the corresponding inhomogeneous system.

The following ISLDE-algorithm is presented in the form of pseudo-code.

1. **Initialize**
   - Let $\langle \mathbf{x}, x_{n+1}\rangle = \bar{\mathbf{0}}$
2. **First Step**
   - Let $\langle \mathbf{x}, x_{n+1}\rangle = \langle \overbrace{0, 0, \ldots, 0}^{n}, 1\rangle$
   - Calculate $a(\langle \mathbf{x}, x_{n+1}\rangle)$
3. **Iterations**
   - Choose any one default vector from $\mathbf{e_j}, \forall j = 1, 2, \ldots, n$, such that $a(\mathbf{x}).a(\mathbf{e_j}) < 0$
   - Let $\langle \mathbf{x}, x_{n+1}\rangle = \langle \mathbf{x}, x_{n+1}\rangle + \langle \mathbf{e_j}, 0\rangle$
   - Calculate $a(\langle \mathbf{x}, x_{n+1}\rangle)$

- Continue to iterate until $a(\langle \mathbf{x}, x_{n+1} \rangle) = \bar{\mathbf{0}}$

4. **Final Step**

- $a(\langle \mathbf{x}, x_{n+1} \rangle)$ is the solution of the homogeneous system.

The solution obtained from the above algorithm is also a solution of the inhomogeneous system because $x_{n+1} = 1$. The ISLDE-algorithm may result in another solution if a different sequence of default vectors are chosen. The redundancy in the search for $\mathcal{B}$ can be reduced by following some simple techniques. The details of these techniques can be found in [23].

4.4.2 Problem Formalization and Solution

The underlying problem is to design a systematic method to split rows and columns of a regular LDPC code from BIBD that achieves an irregular LDPC code with desired degree distribution pair. Assume that we are given a 2-(v, k, l) design with b blocks and a desired degree distribution pair $(\lambda^{\text{req}}, \rho^{\text{req}})$. Let the above design result in a (d_v, d_c)-regular LDPC code whose bipartite graph consists of $d_v b$ number of edges.

The important constraint imposed on the algorithm is to preserve the number of edges during the transformation from a regular graph to an irregular one. The process of splitting columns or rows of the parity-check matrix translates to creating new variable nodes or check nodes in a graph respectively which in turn is equivalent to creating new blocks or points in a design respectively. The design formed as a result of creating new blocks or points is called a *pairwise balanced block design* [3]. It is evident from the above constraint that splitting columns of a parity-check matrix doesn't affect its row-weight distribution and vice-versa. Hence, the problem of achieving column-weight distribution can be viewed independently from that of achieving row-weight distribution. As a direct result of the above statement, it is sufficient to formalize the problem of achieving a

desired λ^{req}.

A block with k elements can be partitioned in p_k different ways as given by

$$p_k = \sum_{j=1}^{j(j-1)<k} (-1)^{j+1} p_{k-\frac{j(3j-1)}{2}} + \sum_{j=1}^{j(j-1)<k} (-1)^{j+1} p_{k-\frac{j(3j+1)}{2}} \qquad \text{(Euler's formula)}.$$

Let us define a *partition distribution matrix* $\mathbf{W}$, such that its elements w_{ij}, for $i = 1, 2, \ldots, p_k$ and $j = 1, 2, \ldots, k$ denote the number of blocks of size j in the partition of the i^{th}-type.

If $\mathbf{u}^{\text{int}}$ is a non-negative integer vector, whose element u_j^{int} is the number of blocks of size j in the resultant pairwise balanced block design , then the problem can be formulated as the one of trying to find solution(s) to the system of the linear diophantine system of equations in (4.4).

$$\mathbf{hW} = \mathbf{u}^{\text{int}} \tag{4.4}$$

The vector $\mathbf{u}^{\text{int}}$ is obtained by solving the following integer optimization problem.

$$\mathbf{u}^{\text{int}} = \arg_{\mathbf{u}} \min \sum_{j=1}^{k} \left| \frac{u_j}{\sum_{m=1}^{k} u_m} - \lambda_j^{\text{req}} \right| \tag{4.5}$$

$$\textbf{Constraint: } \sum_{j=1}^{k} j u_j = bk \tag{4.6}$$

The vector $\mathbf{h}$ in (4.4) is called the *design partition profile* and its elements $h_i, h_i \in \mathbb{N}$ and $\forall i = 1, 2, \ldots, p_k$ denote the number of blocks that require partition of the i^{th} type. It is evident that Eqn.4.4 is a system of linear diophantine equations and (4.6) formalizes the constraint to preserve the number of edges of the graph.

As mentioned above, (4.5) and (4.6) define an integer optimization problem. To solve the integer optimization problem, determine the solution $\mathbf{u}^{\text{real}}$ for (4.5) and (4.6) in $\mathbb{R}^k$ space and then search for $\mathbf{u}^{\text{int}}$ in the neighborhood of $\mathbf{u}^{\text{real}}$. It is straighforward to find

$\mathbf{u}^{\text{real}}$ because

$$\left(\sum_{j=1}^{k} u_j^{\text{real}}\right)\left(\sum_{j=1}^{k} j\lambda_j^{\text{real}}\right) = bk, \text{ since} \tag{4.7}$$

$$\mathbf{u}^{\text{real}} = \lambda_j^{\text{real}}\left(\sum_{j=1}^{k} u_j^{\text{real}}\right) \tag{4.8}$$

A *neighborhood* $\mathcal{N}$ of $\mathbf{u}^{\text{real}}$ rounded to the nearest integer can be defined as a set of all integer-vectors that are at most unit away at corresponding tuples . The error (e) in rounding all components of $\mathbf{u}^{\text{real}}$ to the nearest integer as shown in (4.9) can be corrected for by solving the linear diophantine equation given in (4.10). The classical technique to solve the equation involves eliminating the variables in (4.10) with Euclid's algorithm and reducing it to a univariate equation [15]. A solution of (4.10) is considered *best* if it results in $\mathbf{u}^{\text{int}}$ that satisfies (4.5).

$$e = bk - \sum_{j=1}^{k} j[u_j^{real}] \tag{4.9}$$

$$\sum_{\forall j, u_j^{real}\neq 0} jx_j = e, \qquad j = 1, 2, ..., k \tag{4.10}$$

$$u_j^{int} = [u_j^{real}] + x_j \qquad u_j^{real} \neq 0 \tag{4.11}$$

An ordered set $\mathcal{N}_{\mathcal{U}} \subseteq \mathcal{N}$ is defined as

$$\mathcal{N}_{\mathcal{U}} = \{\, \mathbf{u}^{\text{int}(j)} \in \mathcal{N} \mid \mathbf{u}^{\text{int}(j)} \text{ satisfy (4.6) and } f(\mathbf{u}^{\text{int}(j)}) < f(\mathbf{u}^{\text{int}(i)}) \forall j < i \,\}, \tag{4.12}$$

$$\text{where } f(\mathbf{u}) = \left| \frac{u_j}{\sum_{m=1}^{k} u_m} - \lambda_j^{\text{req}} \right|. \tag{4.13}$$

Although each $\mathbf{u}^{\text{int}} \in \mathcal{N}_{\mathcal{U}}$ is a possible solution, $\mathbf{u}^{\text{int}} = \mathbf{u}^{\text{int}(1)}$ is the best in $\mathcal{N}_{\mathcal{U}}$. Having obtained $\mathbf{u}^{\text{int}}$, (4.4) can be solved using the ISLDE-algorithm given in Sec. 4.4.1 because

$$\mathbf{hW} = \mathbf{u}^{\text{int}} \tag{4.14}$$

$$\Rightarrow \mathbf{W}^{\text{T}}\mathbf{h}^{\text{T}} = \left(\mathbf{u}^{\text{int}}\right)^{\text{T}} \tag{4.15}$$

If $\nexists \mathbf{h}$ for a particular choice of $\mathbf{u}^{\text{int}} = \mathbf{u}^{\text{int}(j)}$, then the above process can be tried for $\mathbf{u}^{\text{int}} = \mathbf{u}^{\text{int}(j+1)}$.

4.4.3 Design of Irregular LDPC Code

The design of irregular LDPC code is facilitated by the tools described in Sec. 4.4.1 and Sec. 4.4.2. To achieve the desired column-weight distribution λ^{req}, obtain a design partition profile for rows namely $\mathbf{h}^{\text{column}}$, using the algorithms described above. To achieve a desired row-weight distribution ρ^{req}, obtain a design partition profile for rows ($\mathbf{h}^{\text{row}}$) using the method described above. The columns and rows of $\mathbf{H}^{\text{req}}$ are split using $\mathbf{h}^{\text{column}}$ and $\mathbf{h}^{\text{row}}$ to obtain the parity-check matrix of the irregular code. This irregular code ($\mathbf{H}^{\text{irreg}}$) will have a degree distribution pair $(\lambda^{\text{achieved}}, \rho^{\text{achieved}})$ that is close to the desired $(\lambda^{\text{req}}, \rho^{\text{req}})$. This irregular code can be reconstructed by storing the regular code along with partition distribution matrix and designing partition profiles for rows and columns.

Note that splitting of rows may increase the girth and minimum distance of the resultant irregular code. On the other hand, column-splitting may increase the girth and decrease the minimum distance of the resultant irregular code.

4.5 Simulation Results

The simulation results of regular and irregular LDPC codes are obtained using the communication systems described in C and/or Matlab. An efficient implementation of the SPA proposed by Hu *et al.* [109] is used to decode LDPC codes. The performance of the codes are investigated for two channel models: the AWGN channel and the PR equalized magnetic recording channel. Generally, the decoder iterates the SPA until a zero syndrome or a maximum number (usually at least 25) of iteration is achieved.

4.5.1 Regular Codes on the AWGN Channel

The performance of regular codes from projective and affine geometries are compared with that of other regular and irregular codes. The details of the parameters of these codes is given in Table 4.3. The codes labeled MacKay and Irregular are from MacKay's resources on the internet [60]. These codes are constructed using pseudo-random methods. The code labeled Lattice Code is based on the incidence of lines and points of a rectangular lattice [102], and this belongs to the family of regular codes. The code labeled Bose CDF is based on Bose's construction of cyclic difference families. The bit-error rate (BER) performance of these codes is shown in Fig. 4.1. The BER curves are adjusted to code rates. Note that [102] has performance results of an elaborate list of codes from combinatorial designs.

In Fig. 4.1, the regular codes from projective and affine geometries outperform codes labeled MacKay and Irregular. The Lattice Code has girth 8 and it outperforms all other codes whose girth is 6.

Table 4.3: DETAILS OF LDPC CODES

Code Name	Description	Length	Dimension	Rate	Col. Wt.
MacKay	Random	4376	4095	0.94	4
Irregular	Random	4376	4096	0.94	$\{3, 4, 5\}$
PG$(2, 2^5)$	PG plane	1057	813	0.76	33
PG$(2, 2^6)$	PG plane	4161	3431	0.82	65
Oval$(2, 2^6)$	Oval design	4095	3430	0.84	32
Unital$(2, 5^2)$	Unital design	525	420	0.80	6
Unital$(2, 2^6)$	Unital design	3648	3135	0.86	9
Unital$(2, 3^4)$	Unital design	5913	5256	0.89	10
Bose CDF	Difference family	4187	4028	0.96	3
Lattice code	Rectangular lattices	4811	3682	0.77	4

4.5.2 Irregular Codes on the AWGN Channel

The bit-error rate (BER) curves shown in this section correspond to the performance of irregular/regular LDPC codes on memoryless AWGN channel. The LDPC codes were

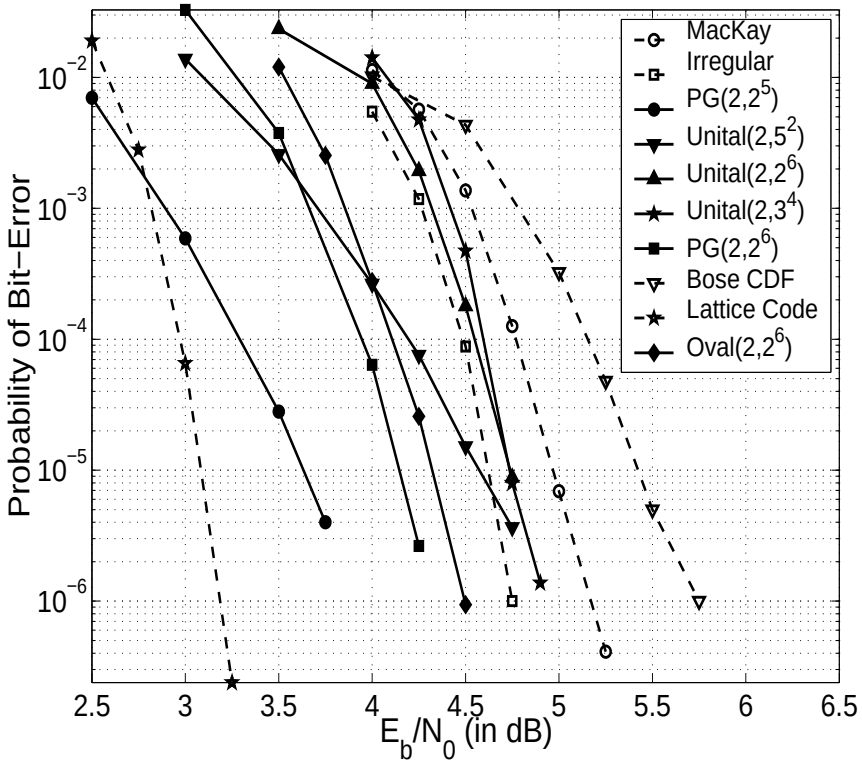


Figure 4.1: Performance of regular code on the memoryless channel

iteratively decoded using the message passing algorithm and hard decisions were made after 15 message passing iterations. The $(81, 81)$-regular LDPC code ($\mathbf{H}^{\text{reg}}$) considered for row and column splitting is a difference-set code [57] of length 273 with 82 parity-bits and has a minimum distance of 18. The BER curve labeled **CS- 1** in Fig. 4.2 corresponds to the performance of a $(3367, 3094)$-irregular code with the following degree distribution pair

$$(\lambda_3 = 0.4054, \lambda_6 = 0.0541, \lambda_7 = 0.0270, \lambda_9 = 0.2973, \lambda_{10} = 0.2162)$$

$$(\omega_{81} = 1)$$

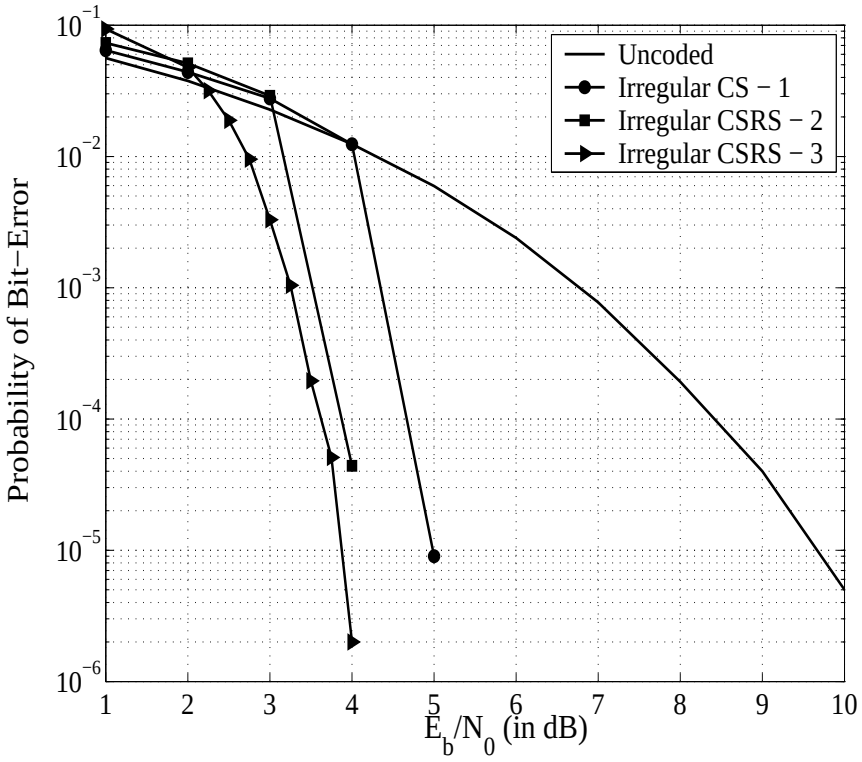


Figure 4.2: Performance of irregular codes on the memoryless channel

The BER curve labeled **CSRS-2** in Fig. 4.2 corresponds to the performance of a $(3367, 2821)$-irregular code with the following degree distribution pair

$$(\lambda_3 = 0.4054, \lambda_6 = 0.0541, \lambda_7 = 0.0270, \lambda_9 = 0.2973, \lambda_{10} = 0.2162)$$

$$(\omega_{40} = 0.5, \omega_{41} = 0.5)$$

The BER curve labeled **CSRS-3** in Fig. 4.2 corresponds to the performance of a $(3367, 2275)$-irregular code with the following degree distribution pair

$$(\lambda_3 = 0.4054, \lambda_6 = 0.0541, \lambda_7 = 0.0270, \lambda_9 = 0.2973, \lambda_{10} = 0.2162)$$

$$(\omega_{15} = 0.25, \omega_{16} = 0.25, \omega_{20} = 0.25, \omega_{30} = 0.25)$$

It can be observed that the performance of irregular codes improves as their degree distribution pair approaches the good degree distribution pair presented in [75]. The performance of irregular LDPC code with columns of weight 2 is observed to have a high

error-floor, possibly due to a very poor minimum distance. Hence, we avoided columns of weight 2 in the irregular codes constructed from the regular LDPC code.

In Fig. 4.3, the performance of regular codes from projective geometry is compared with that of irregular codes. Except for the code labeled CSRS − 1, all other codes have comparable rates. The performance of the code labeled PG is comparable to that of the code labeled CSRS − 2.

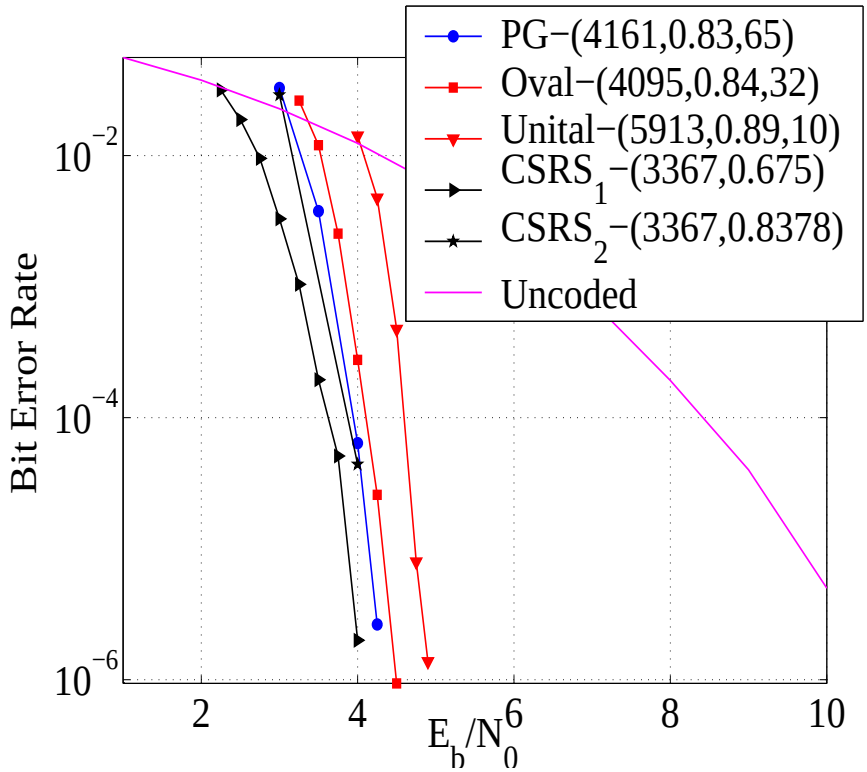


Figure 4.3: Comparison between performance of regular and irregular codes

4.5.3 Irregular Codes on PR Equalized Channel

The BER plots shown in this section correspond to the performance of irregular/regular LDPC codes on perpendicular (magnetic) recording channels. The channel linear density is adjusted according to the code rate. The output of the channel is equalized to one of the generalized partial-response targets by an adaptive least mean-squared equalizer.

The noise in the channel is modeled as AWGN with variance σ^2 and the signal-to-noise ratio (SNR) is defined as SNR$= 10 \log \frac{{V_p}^2}{\sigma^2}$, where $V_p = 1$ is the peak amplitude of an isolated transition response for perpendicular recording.

The soft-information is extracted from a partial-response channel using the BCJR algorithm operating on a channel trellis, and then used in a sum-product algorithm (SPA) [56, 19] for LDPC decoding. The soft-information on variable nodes obtained after five iterations of SPA is used as *a priori* likelihoods for the next round of the BCJR detection. The final hard decisions on the variables are made after three iterations of the combined system of the BCJR algorithm and SPA.

In Fig. 4.4, we compare the BER performance of regular and irregular LDPC codes on a perpendicular (magnetic) recording channel with ND= 2.0. The output of the channel is equalized to a partial response target of PR2= [1 2 1]. The irregular LDPC codes are constructed by splitting columns and rows of randomly generated regular LDPC codes. The weight distribution for columns and rows of these codes are decided by trial-and-error with a computer program.The column and row weight distributions of the codes are given in Table 4.4. As we can see from this figure, irregular LDPC codes, namely CS-1 and CS-5, perform just as well as the regular code labeled CS-3. Also, these irregular LDPC codes outperform the regular code labeled CS-4. But it should be noted that we have constructed a regular $(4352, 4096)$-LDPC code (labeled as CS-0 in the figure) with three ones in a column, that outperforms the irregular codes (refer Table 4.4) by at least 0.25dB .

In Fig. 4.5, we compare the BER performance of some of the above mentioned regular and irregular LDPC codes on a perpendicular (magnetic) recording channel with ND= 2.0 and equalized to a partial response target of EPR2= [1 3 3 1]. The column and row weight distributions of the codes are given in Table 4.4. As we can see from this figure, irregular LDPC codes, namely CS-1, CS-5 and CSRS-6, outperform the regular code

labeled CS-4 by at least 0.20dB.

Table 4.4: COLUMN AND ROW WEIGHT DISTRIBUTIONS

Col.-Wt. Dist.	CS-1 (4120, 3864)	CS-2 (4088, 3832)	CS-3 (4096, 3840)	CS-4 (4096, 3841)	CS-5 (4060, 3804)	CSRS-6 (4068, 3812)
λ_1	0	0	0	0	0	0
λ_2	0.041747	0.058939	0	0	0.002956	0.002956
λ_3	0.93398	0.98232	1	0	0.98522	0.98522
λ_4	0.024272	0.011788	0	1	0.002955	0.002956
λ_5	0	0	0	0	0.005911	0.005911
λ_6	0	0	0	0	0	0.0029557
λ_7	0	0	0	0	0	0
λ_8	0	0	0	0	0.002956	0
Row-Wt. Dist.						
ω_{20}	0	0	0	0	0	0.375
ω_{48}	1	1	1	0	0	0.25
ω_{64}	0	0	0	1	1	0
ω_{76}	0	0	0	0	0	0.375

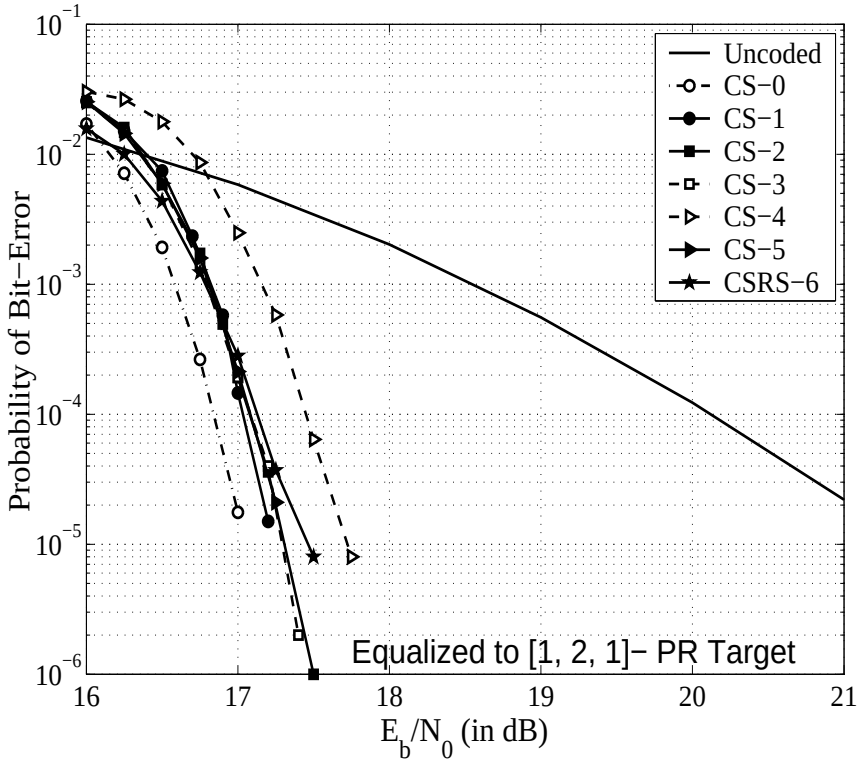


Figure 4.4: Performance of irregular LDPC codes on [1, 2, 1]-equalized perpendicular magnetic recording channel

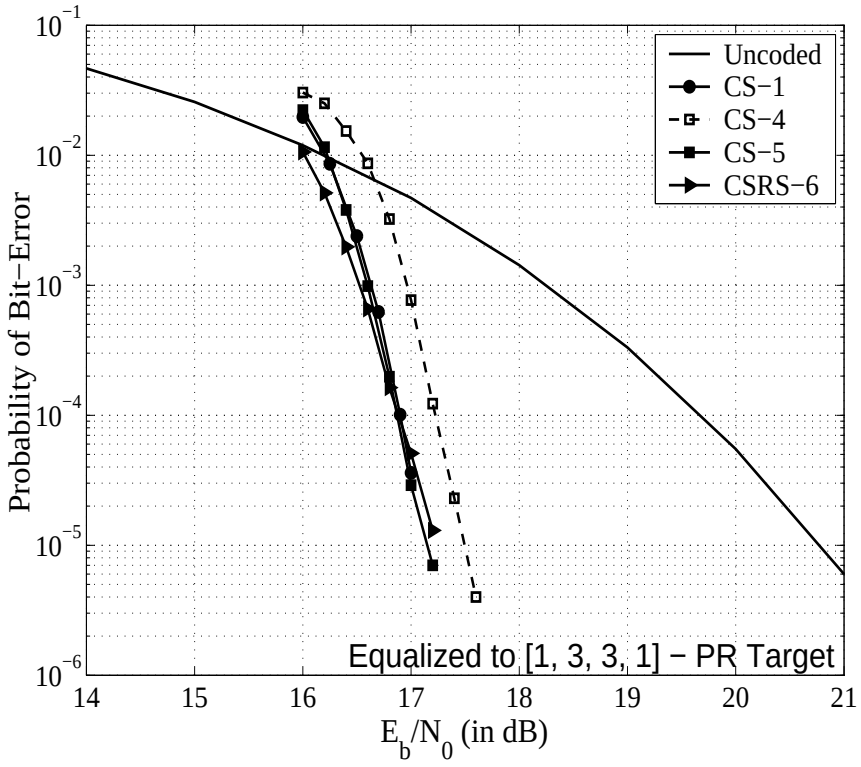


Figure 4.5: Performance of irregular LDPC codes on [1, 3, 3, 1]-equalized perpendicular magnetic recording channel

CHAPTER 5

ITERATIVE DECODING OF LINEAR BLOCK CODES

In the previous chapter, we proposed several methods to systematically construct regular and irregular LDPC codes. Similarly, other researchers have proposed interesting techniques to (systematically and/or pseudo-randomly) construct regular and irregular LDPC codes that exhibit good girth properties. With exceptions of codes from combinatorial designs, it is not easy to investigate the properties of these regular and irregular LDPC codes. Obviously, simulation techniques are used to gain some insight into the properties of these codes. On the other hand, algebraic codes have been a subject of interest since late 1940s. The algebra underlying their construction methods has facilitated in creating a wealth of knowledge about these codes. In fact, Reed-Solomon codes are maximum distance separable codes, and they are some of the best codes that are available. Given the fact that iterative decoders can achieve Shannon capacity, it would be ideal if the algebraic codes are suitable for this class of decoders. Obviously they are not suitable for iterative decoders because their parity-check matrices are high density matrices. In this chapter, we investigate the possibilities of generating alternative (parity-check matrix) representations for algebraic codes that are better suited for iterative decoders.

In this chapter, we present an algorithm that is specific to removing four-cycles in a TG of a linearly block code. Such an effort is necessary to facilitate message-passing decoders (MPD) of well-known algebraic codes. In [56], Kschischang dealt with techniques of clustering (or merging) and stretching nodes in order to remove short cycles in a TG of the code. Unfortunately, these techniques require passing message-vectors

along edges of the modified TG and hence, increasing the complexity of MPDs. The algorithm presented in this chapter preserves binary message-passing and so, the increase in decoding complexity is not significant. Although the idea of introducing *auxiliary variables* and *auxiliary checks* to the parity-check matrix of the code is similar to the one presented in [112], the approach taken in achieving a four-cycle free representation is distinct. In Section 5.2, we prove that this technique is guaranteed to remove all four-cycles in a TG of the code. Now, instead of the conventional TG of the code, the four-cycle free graph (or the modified graph) can be used to iteratively decode on BEC and AWGN channel. In Section 5.3, we prove that the error rate performance of the iterative erasure decoder working on the modified graph is at least as good as that obtained by decoding on the conventional graph of the code. Also, we present simulation results of the bit-error rate (BER) and block-error rate (BLER) performances of an algebraic linear code over the BEC. Theoretical analysis of the effectiveness of GPCM representations in improving the performance of an MPD for AWGN channels is rather difficult. Hence, we resort to empirical results to demonstrate the effectiveness of four-cycle free representations. The results presented in Section 5.4 demonstrate improvements in BER and BLER performances of algebraic codes obtained by decoding iteratively on the modified graphs. This work in full or partial was published in [83] and [84].

5.1 Algorithm to Remove Four-Cycles in Tanner Graphs

Consider a linear block code $\mathcal{C}$ of length n defined as the solution-space (over $\mathbb{F}_2$) of the system of linear equations $\mathbf{Hx} = \mathbf{0}$, where $\mathbf{H}$ is an $m \times n$ binary matrix. The bipartite graph representation $\mathcal{G}$ of $\mathcal{C}$ is assumed to have at least one four-cycle. It is necessary to identify four-cycles in $\mathcal{G}$ before they can be removed. The four-cycle structure of $\mathcal{G}$ is obtained from the square (over $\Re$) of the adjacency matrix $\mathbf{A}$ of the graph. The

adjacency matrix and its square can be written as

$$\mathbf{A} = \begin{pmatrix} 0 & \mathbf{H} \\ \mathbf{H}^{\mathrm{T}} & 0 \end{pmatrix} \quad \text{and} \quad \mathbf{A}^2 = \begin{pmatrix} \mathbf{H}\mathbf{H}^{\mathrm{T}} & 0 \\ 0 & \mathbf{H}^{\mathrm{T}}\mathbf{H} \end{pmatrix}.$$

The element in position (i, j) of $\mathbf{A}^2$ is the number of distinct paths of length 2 between the nodes a_i and a_j. In $\mathbf{A}^2$, the component $\mathbf{H}^{\mathrm{T}}\mathbf{H}$ calculates the number of paths of length 2 between variable nodes, and the component $\mathbf{H}\mathbf{H}^{\mathrm{T}}$ calculates the number of paths of length 2 between check nodes. These component matrices are symmetric. In the bipartite graph $\mathcal{G}$, there is no path of length 2 between a variable node and a check node. Also, any two distinct paths of length 2 between two nodes contribute to a four-cycle. The total number of four-cycles in $\mathcal{G}$ is denoted by $\psi(\mathcal{G})$.

Theorem 6. *In a simple bipartite graph $\mathcal{G}$, the total number of four-cycles $\psi(\mathcal{G})$ is given by*

$$\psi(\mathcal{G}) = \sum_{i=1}^{n} \sum_{j=i+1}^{n} \begin{pmatrix} (\mathbf{H}^{T}\mathbf{H})_{ij} \\ 2 \end{pmatrix}.$$

Proof. Assume that there are l, say $l > 1$, distinct paths of length 2 between distinct variable nodes x_i and x_j. This means that the element in position (i, j) of $\mathbf{H}^{\mathrm{T}}\mathbf{H}$ is l. A pair of distinct paths between these nodes contribute to a four-cycle, and there are $\begin{pmatrix} l \\ 2 \end{pmatrix}$ different ways to choose such a pair. Hence, the total number of four-cycles in $\mathcal{G}$ can be obtained by summing the quantities $\begin{pmatrix} (\mathbf{H}^{\mathrm{T}}\mathbf{H})_{ij} \\ 2 \end{pmatrix}$ over every (ordered) pair of variable nodes. □

Corollary 7. *In a complete bipartite graph $\mathcal{K}_{m,n}$, the total number of four-cycles is* $\psi(\mathcal{K}_{m,n}) = \begin{pmatrix} n \\ 2 \end{pmatrix} \begin{pmatrix} m \\ 2 \end{pmatrix}$.

It is easy to show that the element in position (i,i) of $\mathbf{H}^{\mathrm{T}}\mathbf{H}$ is equal to the degree of the i-th variable node. In [112], the first step of the algorithm involves the process of taking intersections of all pairs of the constraint sets to identify four-cycles in the code. In this correspondence, we compute the product $\mathbf{H}^{\mathrm{T}}\mathbf{H}$ to identify the four-cycles in the code. Also, it will be evident from the following discussion that we do not consider all the intersection sets. Now equipped with the tools to identify the four-cycle structure of the graph, it is easy to remove all four-cycles in $\mathcal{G}$ as shown below.

Assume that there is a four-cycle involving variables x_u and x_v. Taking a componentwise product of the u-th and v-th columns of $\mathbf{H}$, we can determine the two check nodes, say S_i and S_j, that are involved in the four-cycle. Let $x_u - S_i - x_v - S_j - x_u$ represent the four-cycle. In order to remove this four-cycle, replace the term $x_u + x_v$ in S_i and S_j with an auxiliary variable $x_{u,v}$. This is equivalent to appending a new column to the original matrix $\mathbf{H}$, and replacing elements in positions (i,u), (j,u), (i,v), and (j,v) of $\mathbf{H}$ with zeros. In addition, an auxiliary check equation of the form $x_u + x_v + x_{u,v} = 0$ is added to the original set of constraints. This is equivalent to appending a new row to the original matrix $\mathbf{H}$ that corresponds to the auxiliary check equation. If these variable nodes are involved in more than one four-cycle, then every check equation containing the term $x_u + x_v$ is replaced with the auxiliary variable $x_{u,v}$. This process removes all four-cycles involving variables x_u and x_v, and results in a modified parity-check matrix with an additional (auxiliary) constraint and variable in comparison to $\mathbf{H}$. By introducing an auxiliary variable, we have not changed the code-rate because this variable is not transmitted over the channel. The introduction of the auxiliary variable results in the extension of the vector subspace $\mathcal{C}$ to a vector subspace $\mathcal{C}_1$. For any vector $\mathbf{x} = \langle x_1, \cdots, x_u, \cdots, x_v, \cdots, x_n \rangle$ in $\mathcal{C}$, there is a unique vector $\mathbf{x}_1 = \langle x_1, \cdots, x_u, \cdots, x_v, \cdots, x_n, x_u + x_v \rangle$ in $\mathcal{C}_1$. On the other hand, the auxiliary check guarantees that the dimension of $\mathcal{C}_1$ is equal to that of $\mathcal{C}$.

By iteratively applying the procedure on the modified parity-check matrix, it is pos-

sible to remove all four-cycles involving actual and/or auxiliary variables. The iterative algorithm is summarized as follows:

Initialize:

- Let $l = 0$.
- Let $\mathbf{H}_l := \mathbf{H}$.
- $\mathbf{H}_l$ is a $m_l \times n_l$-matrix such that $m_0 = m$ and $n_0 = n$.

Step 1:

- Compute ${\mathbf{H}_l}^{\mathrm{T}}\mathbf{H}_l$.
- If there is at least one four-cycle in $\mathbf{H}_l$, then go to Step 2.
- If $\mathbf{H}_l$ is four-cycle free, then go to Step 4.

Step 2:

- Let x_u and x_v be a pair of variables involved in a four-cycle.
- The column-vector $\mathbf{u}.\mathbf{v}$ is the componentwise product of the u-th and v-th columns of $\mathbf{H}_l$.
- Let $\mathbf{B}$ be a $m_l \times n_l$-matrix defined as:

$$\mathbf{B} = \left[\underbrace{\overbrace{\mathbf{0}, \cdots, \mathbf{0}}^{u-1}, \mathbf{u}.\mathbf{v}, \overbrace{\mathbf{0}, \cdots, \mathbf{0}}^{v-u-1}, \mathbf{u}.\mathbf{v}, \cdots, \mathbf{0}}_{n_l}\right],$$

  where $\mathbf{0}$ is a zero column vector, and $\mathbf{u}$ and $\mathbf{v}$ are the u-th and v-th columns of $\mathbf{H}_l$. The addition operation is defined over the binary field.
- Let $S' := \langle\underbrace{\overbrace{0, \cdots, 0}^{u-1}, 1, \overbrace{0, \cdots, 0}^{v-u-1}, 1, 0, \cdots, 0, 1}_{n_l+1}\rangle$, where S' corresponds to the auxiliary check equation.

Step 3:

- The transformed matrix is written as:

$$\mathbf{H}_{l+1} := \left(\begin{array}{c|c} \mathbf{H}_l + \mathbf{B} & \mathbf{u}.\mathbf{v} \\ \hline S' & \end{array} \right).$$

  The addition operation is defined over the binary field.

- The dimension of $\mathbf{H}_{l+1}$ is $(m_l + 1) \times (n_l + 1)$.

- $l \longleftarrow l + 1$

- Go to Step 1.

Step 4:

- $\mathbf{H}^{'} \longleftarrow \mathbf{H}_l$

- The matrix $\mathbf{H}^{'}$ is free of four-cycles with dimension $M \times N$.

In the algorithm, Step 1, Step 2 and Step 3 constitute an iteration. After the l-th iteration, one auxiliary variable node and an auxiliary check node are inserted into the graph of $\mathbf{H}_{l-1}$. The weight of the augmented row is 3, and the weight of the augmented column is one more than the weight of the column-vector $\mathbf{u}.\mathbf{v}$ computed in Step 2. In Section 5.2, we show that in fact this algorithm guarantees to remove all four-cycles in the original graph $\mathcal{G}$ after a finite number of iterations. The matrix $\mathbf{H}^{'}$, that is free of four-cycles, is referred to as the output GPCM of $\mathcal{C}$; $\mathcal{G}^{'}$ is the bipartite graph corresponding to $\mathbf{H}^{'}$. Step 2 and Step 3 of this algorithm are similar to Steps $2, 3$ and 4 of the algorithm presented in [112]. Unlike in [112], we avoid creating a four-cycle by allowing only one way to break down each constraint set. Also, this reduces the complexity of organizing sets into a *partially ordered set* of sets, and decreases the dimension of the output GPCM. The following example is designed to better illustrate the application of the algorithm.

Example 1: Consider a $(7, 4)$-Hamming code, defined by a parity-check matrix $\mathbf{H}$ with $n = 7$ variables and $m = 3$ check equations. The i-th column of $\mathbf{H}$ corresponds to a variable node x_i of the graph $\mathcal{G}$, and the j-th row of the matrix corresponds to a check node S_j of $\mathcal{G}$. From the component matrix $\mathbf{H}^{\mathrm{T}}\mathbf{H}$, we can compute that there are exactly three four-cycles in $\mathcal{G}$. The parity-check matrix and the component matrix are

$$\mathbf{H} = \begin{pmatrix} 1 & 0 & 0 & 1 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 1 \end{pmatrix},$$

$$\mathbf{H}^{\mathrm{T}}\mathbf{H} = \begin{pmatrix} 1 & 0 & 0 & 1 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 1 \\ 1 & 1 & 0 & 2 & 1 & 1 & 2 \\ 1 & 0 & 1 & 1 & 2 & 1 & 2 \\ 0 & 1 & 1 & 1 & 1 & 2 & 2 \\ 1 & 1 & 1 & 2 & 2 & 2 & 3 \end{pmatrix}.$$

The four-cycle $x_4 - S_1 - x_7 - S_2 - x_4$ in $\mathbf{H}$ is removed by introducing an auxiliary variable $x_{4,7}$, and an auxiliary check equation written as: $x_4 + x_7 + x_{4,7} = 0$. The introduction of $x_{4,7}$ is equivalent to appending a new column to $\mathbf{H}$, and replacing elements in positions $(1, 4)$, $(1, 7)$, $(2, 4)$, and $(2, 7)$ of $\mathbf{H}$ with zeros. The matrix $\mathbf{H}_1$ obtained after the first iteration is written as

$$\mathbf{H}_1 = \begin{pmatrix} 1 & 0 & 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 \end{pmatrix}.$$

Since $\mathbf{H}_1$ is free of four-cycles, it is the desired GPCM of the $(7, 4)$-Hamming code. Thus in this case, just one auxiliary variable is required to remove three four-cycles. ■

For a given graph $\mathcal{G}$, an upper bound on the number of auxiliary variables (or auxiliary checks) required to remove all four-cycles in the graph can be computed from the following observations about the algorithm.

- At each iteration of the algorithm, exactly one auxiliary variable is introduced.
- Just one auxiliary variable $x_{u,v}$ is required to remove any number of four-cycles involving a pair of variables x_u and x_v.

In Section 5.2, we will show that the number of four-cycles in $\mathcal{G}_l$ (output of the l-th iteration) is strictly less than that in $\mathcal{G}_{l-1}$. Hence, the total number of auxiliary variables $\Delta(N, n)$ in $\mathcal{G}'$ is upper bounded by the total number of four-cycles in $\mathcal{G}$.

$$\Delta(N, n) = N - n \leq \psi(\mathcal{G}).$$

The summation quantity is an upper bound because there is always the possibility of an auxiliary variable removing more four-cycles than intended.

5.2 Why does the algorithm work?

In this section, we present a combinatorial argument to substantiate the claim that the algorithm removes all four-cycles in the original graph $\mathcal{G}$ after a finite number of iterations. Assume that $\mathcal{G}_l$ is the bipartite graph corresponding to $\mathbf{H}_l$, obtained after l iterations of the algorithm on the $m \times n$-matrix $\mathbf{H}$. The key idea of the proof is to show that the number of four-cycles in $\mathcal{G}_l$ is strictly less than that in $\mathcal{G}_{l-1}$. We prove this key idea in Lemma 8.

Lemma 8. $\psi(\mathcal{G}_l) < \psi(\mathcal{G}_{l-1})$, $l \in \mathbb{N}$.

Proof. For some l, $\mathcal{G}_l$ is obtained from an iteration of the algorithm on $\mathcal{G}_{l-1}$. Without loss of generality, assume that there are $\begin{pmatrix} p \\ 2 \end{pmatrix}$, $p \geq 2$, four-cycles in $\mathcal{G}_{l-1}$ involving variables x_u and x_v. This means that there are exactly p check equations (in $\mathbf{H}_{l-1}$) with ones in the u-th and the v-th columns. Rearrange the rows of $\mathbf{H}_{l-1}$ such that the first p rows have ones in the u-th and the v-th columns. The i-th row of the matrix corresponds to the check equation S_i. The general form of $\mathbf{H}_{l-1}$ is

$$\begin{pmatrix}
h_{1,1} & \dots & 1 & \dots & 1 & \dots & h_{1,n_{l-1}} \\
h_{2,1} & \dots & 1 & \dots & 1 & \dots & h_{2,n_{l-1}} \\
\vdots & \ddots & \vdots & \ddots & \vdots & \ddots & \vdots \\
h_{p,1} & \dots & 1 & \dots & 1 & \dots & h_{p,n_{l-1}} \\
h_{p+1,1} & \dots & h_{p+1,u} & \dots & h_{p+1,v} & \dots & h_{p+1,n_{l-1}} \\
\vdots & \ddots & \vdots & \ddots & \vdots & \ddots & \vdots \\
h_{m_{l-1},1} & \dots & h_{m_{l-1},u} & \dots & h_{m_{l-1},v} & \dots & h_{m_{l-1},n_{l-1}}
\end{pmatrix}.$$

The four-cycles involving x_u and x_v are removed by augmenting a new variable node x_{n_l}, and a new check node S_{m_l}. The general form of the resultant matrix $\mathbf{H}_l$ is given in (5.1) (overleaf).

$$\left(\begin{array}{ccccccc|c}
h_{1,1} & \dots & 0 & \dots & 0 & \dots & h_{1,n_{l-1}} & 1 \\
h_{2,1} & \dots & 0 & \dots & 0 & \dots & h_{2,n_{l-1}} & 1 \\
\vdots & \ddots & \vdots & \ddots & \vdots & \ddots & \vdots & \vdots \\
h_{p,1} & \dots & 0 & \dots & 0 & \dots & h_{p,n_{l-1}} & 1 \\
h_{p+1,1} & \dots & h_{p+1,u} & \dots & h_{p+1,v} & \dots & h_{p+1,n_{l-1}} & 0 \\
\vdots & \ddots & \vdots & \ddots & \vdots & \ddots & \vdots & \vdots \\
h_{m_{l-1},1} & \dots & h_{m_{l-1},u} & \dots & h_{m_{l-1},v} & \dots & h_{m_{l-1},n_{l-1}} & 0 \\
\hline
0 & \dots & 1 & \dots & 1 & \dots & 0 & 1
\end{array}\right) \tag{5.1}$$

Now, let us examine the four-cycle structure of $\mathbf{H}_l$, and compare it with that of $\mathbf{H}_{l-1}$. If $\mathcal{G}_l$ is free of four-cycles, then the lemma is trivially true. In order to prove the

non-trivial case, we have assumed in the following list of arguments that there is at least one four-cycle in $\mathcal{G}_l$.

- The construction guarantees that there is no four-cycle involving x_u and x_v in $\mathcal{G}_l$. On the other hand, there are $\begin{pmatrix} p \\ 2 \end{pmatrix}$ four-cycles involving these variable nodes in $\mathcal{G}_{l-1}$.

- If there is a four-cycle involving x_u and x_{n_l} in $\mathcal{G}_l$, then this pair of variables should participate in two distinct check equations of $\mathcal{H}_l$. But the algorithm guarantees that there is only one check equation, namely S_{m_l}, that contains this pair of variables. Hence by contradiction, we can conclude that there is no four-cycle involving this pair of variables. Such an argument can be used to prove the absence of four-cycles involving x_v and x_{n_l}.

- Let $x_{n_l} - S_q - x_s - S_r - x_{n_l}$, where $s \neq u, v$, be a four-cycle in $\mathcal{G}_l$. The general form of $\mathbf{H}_l$ reveals that q and r have to take values from $\{1, 2, \ldots, p\}$. For every such four-cycle in $\mathcal{G}_l$, there exist two four-cycles, namely $x_u - S_q - x_s - S_r - x_u$ and $x_v - S_q - x_s - S_r - x_v$, in $\mathcal{G}_{l-1}$.

From the above arguments and the fact that $p \geq 2$, we can conclude that the number of four-cycles in $\mathcal{G}_l$ is strictly less than that in $\mathcal{G}_{l-1}$. □

Now we restate the claim of the algorithm as a theorem, and prove it using Lemma 8.

Theorem 9. *Any linear binary block code $\mathcal{C}$ has an equivalent bipartite graph representation $\mathcal{G}'$ that is free of four-cycles.*

Proof. Let $\mathbf{H}$ be the parity-check matrix of $\mathcal{C}$. If the bipartite graph $\mathcal{G}$ corresponding to $\mathbf{H}$ is free of four-cycles, then the theorem is trivially true. Otherwise, let $\psi(\mathcal{G})$ be the

number of four-cycles in $\mathcal{G}$. After l iterations of the algorithm with $\mathcal{G}$ as the input, we know that the following statement is true from Lemma 8

$$\psi(\mathcal{G}) > \psi(\mathcal{G}_1) > \psi(\mathcal{G}_2) > \ldots > \psi(\mathcal{G}_l).$$

The number of four-cycles in $\mathcal{G}$ with m check nodes and n variable nodes is upper bounded by $\begin{pmatrix} n \\ 2 \end{pmatrix} \begin{pmatrix} m \\ 2 \end{pmatrix}$, see Corollary 7. Thus for finite m and n, there exists a finite l in $\mathbb{N}$ such that l iterations of the algorithm with $\mathcal{G}$ as the input result in a four-cycle free graph $\mathcal{G}'$. Since we started with $\mathcal{G}$, the algorithm guarantees that $\mathcal{G}'$ is an equivalent representation of $\mathcal{C}$. □

5.3 Iterative Decoding of Linear Block Codes over BEC

In BEC, the input alphabet is binary, and the output alphabet, besides symbols 0 and 1, includes an erasure symbol ε. The probability of receiving an ε is called the *erasure probability*. An optimal decoding of a linear block code $\mathcal{C}$ over the channel is achieved by a direct application of Gaussian elimination. The complexity of such an optimal decoder is $\mathcal{O}(n^3)$. In this section, we are interested in a suboptimal erasure decoder, referred to as the MPD, that decodes iteratively by passing messages over the edges of the graph $\mathcal{G}$ of $\mathcal{C}$. The complexity of such a decoder is linear in the length of the code. A *decoding failure* in an MPD is caused exactly when every check equation has more than one variable in erasure. For a fixed $\mathbf{H}$ of $\mathcal{C}$, a subset $\mathcal{S}$ of variable nodes such that all neighbors of $\mathcal{S}$ are connected to $\mathcal{S}$ at least twice is called a *stopping set* [24]. A decoding failure occurs precisely when the set of variables in erasure contains a stopping set. The knowledge of all stopping sets of $\mathbf{H}$ is sufficient to theoretically compute the error performance of the MPD for a given BEC.

The simplicity of the iterative decoder provides a convenient framework to analyze the effects of removing four-cycles on its performance. Let $\mathcal{G}'$ be a four-cycle free repre-

sentation of $\mathcal{C}$. Now $\mathcal{G}^{'}$ can be used to iteratively decode the linear code. In general, the performance of the decoder working on $\mathcal{G}^{'}$ is better than that of the decoder working on $\mathcal{G}$. This will be shown by comparing the stopping set structures of $\mathcal{G}$ and $\mathcal{G}^{'}$. Finally, we compare simulated BER and BLER performances of the MPD working on several representations of a $(23, 12)$-Golay code.

5.3.1 Analysis

Firstly, we present an example to illustrate that eliminating four-cycles helps to remove certain stopping sets and in turn improves the performance of the code under MPD. Also, this example illustrates the iterative decoding process on a GPCM representation of the code.

Example 2: Let S_1,S_2, and S_3 be the constraints of a binary code of length 7. Each codeword, represented as $\mathbf{x} = \langle x_1, x_2, x_3, x_4, x_5, x_6, x_7 \rangle$, satisfies the following set of constraints:

$$\begin{aligned} S_1\text{:} \quad & x_1 + x_2 + x_3 + x_4 = 0 \\ S_2\text{:} \quad & x_1 + x_2 + x_5 + x_6 = 0 \\ S_3\text{:} \quad & x_1 + x_3 + x_7 = 0. \end{aligned}$$

Let $\mathbf{y} = \langle \varepsilon, \varepsilon, \varepsilon, 1, 1, 1, 1 \rangle$ be the channel observation of a codeword $\mathbf{x}$. From the received $\mathbf{y}$, and the set of constraints, it is easy to check that $\mathcal{S} = \{y_1, y_2, y_3\}$ is a stopping set. In other words, each check has at least two neighbors which take value ε. Hence from the above discussion, we can say that the iterative decoder will fail.

We can remove the four-cycle x_1-S_1-x_2-S_2-x_1 in the set of constraints by introducing

an auxiliary variable $x_{1,2}$. Let $S_1^{'}$, $S_2^{'}$, S_3, and $S_4^{'}$ be the set of modified constraints:

$$\begin{aligned} S_1^{'}: \quad & x_{1,2} + x_3 + x_4 &&= 0 \\ S_2^{'}: \quad & x_{1,2} + x_5 + x_6 &&= 0 \\ S_3: \quad & x_1 + x_3 + x_7 &&= 0 \\ S_4^{'}: \quad & x_{1,2} + x_1 + x_2 &&= 0. \end{aligned}$$

Let us try to decode $\mathbf{y}$ with the modified set of constraints. Since y_1 and y_2 are in erasure, the auxiliary variable takes on the value ε. It is clear that the subset of variables $\{y_1, y_2, y_3\}$ does not contain a stopping set and hence, we are able to decode $\mathbf{y}$. Let $\hat{\mathbf{x}}$ represent the decoded word. Then,

$$\begin{aligned} \hat{x}_{1,2} + 1 + 1 = 0 \quad &\Rightarrow \hat{x}_{1,2} = 0 \text{ (from } S_2^{'}) \\ 0 + \hat{x}_3 + 1 = 0 \quad &\Rightarrow \hat{x}_3 = 1 \text{ (from } S_1^{'}) \\ \hat{x}_1 + 1 + 1 = 0 \quad &\Rightarrow \hat{x}_1 = 0 \text{ (from } S_3) \\ 0 + 0 + \hat{x}_2 = 0 \quad &\Rightarrow \hat{x}_2 = 0 \text{ (from } S_4^{'}). \end{aligned}$$

The decoded word is $\hat{\mathbf{x}} = \langle 0, 0, 1, 1, 1, 1, 1 \rangle$. ■

The example also illustrates that auxiliary variables are initialized with values based on channel observations. It will be shown that each auxiliary variable can be expressed as a sum of standard variables by manipulating the auxiliary check equations. Thus, an auxiliary variable takes the value ε if at least one of its component standard variables is in erasure. Otherwise, it takes a value 0 or 1. Also. it is clear from the example that a stopping set is a characteristic of the representation (matrix or graphical), and not necessarily that of the code. In other words, a decoding failure that occurred using a particular graphical representation of a code could have been averted with a different graphical representation of the code.

Let $\mathcal{S}^{'}$ be a stopping set in the four-cycle free GPCM $\mathbf{H}^{'}$, obtained by applying the

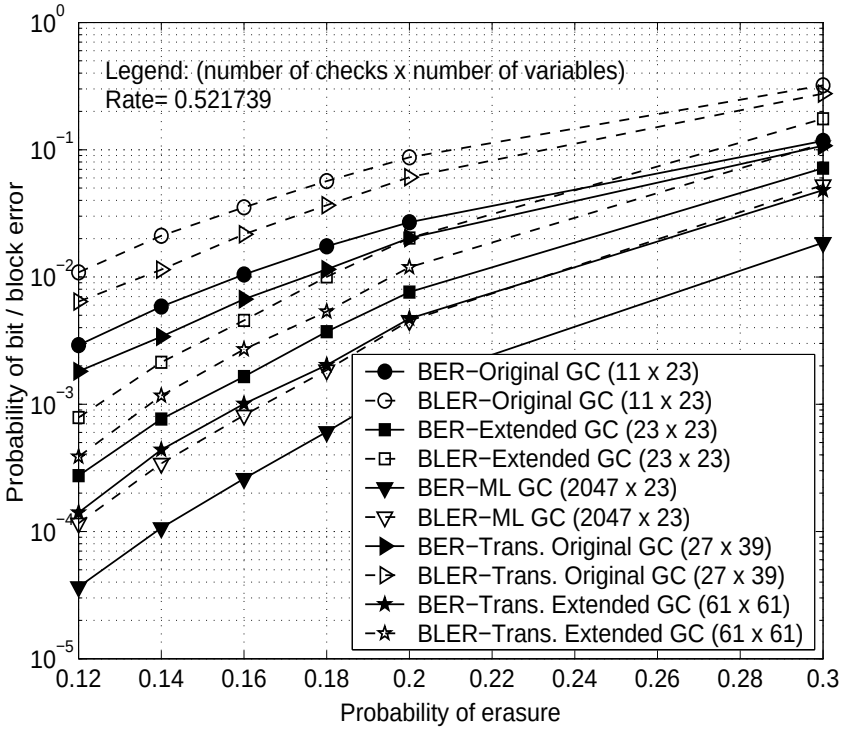


Figure 5.1: Performance of different representations of $(23, 12)$-Golay code under belief propagation for BEC

iterative algorithm, of the code. The variables (checks) corresponding to columns (rows) of $\mathbf{H}'$ are classified into two classes, namely *standard variables* (*checks*) and *auxiliary variables* (*checks*). Any variable x_i corresponding to the i-th column of $\mathbf{H}'$ is a standard variable if $i \leq n$. Otherwise, x_i is an auxiliary variable. Similarly, any check equation S_i corresponding to the i-th row of $\mathbf{H}'$ is a standard check if $i \leq m$. Otherwise, S_i is an auxiliary check. The degree of any auxiliary check node is exactly three.

Any auxiliary variable can be written as a sum (over $\mathbb{F}_2$) of two or more standard variables. We prove this statement by describing a process to achieve such a summation. For a given auxiliary variable x_u, choose the corresponding auxiliary check equation S_{m+u-n}. Such a check exists because for every auxiliary variable introduced to remove four-cycles, a new check equation is also introduced. In general, this check equation can be written as

$$S_{m+u-n}: \quad x_i + x_j \quad = x_u,$$

where $i < u$ and $j < u$. It is important to notice that the i-th and the j-th columns are strictly to the left of the u-th column. If the components of the equation, x_i and x_j, are standard variables, then we are done. Otherwise, for every component that is an auxiliary variable, repeat the above process. This recursive process is guaranteed to achieve the goal in finite number of steps because

- For any auxiliary variable, every component variable corresponds to a column to the left of that of the auxiliary variable.
- The dimension of $\mathbf{H}'$ is finite.

Let us examine the stopping set structure of $\mathcal{G}'$ and compare it to that of $\mathcal{G}$.

Theorem 10. *No subset of auxiliary variables is a stopping set.*

Proof. Assume that $\mathcal{S}^{'}$ is a nonempty subset of auxiliary variables which is a stopping set. This means that there will be a decoding failure if every element in $\mathcal{S}^{'}$ takes the value ε, and every other variable takes a value 0 or 1. Let $\mathcal{N}(\mathcal{S}^{'})$ be the set of all neighbors of $\mathcal{S}^{'}$. For an auxiliary variable $x_u \in \mathcal{S}^{'}$, there exists an auxiliary check S_{m+u-n} that belongs to $\mathcal{N}(\mathcal{S}^{'})$. Since we are interested in the auxiliary variables, let us examine the structure in the last $\Delta(N, n)$ columns of $\mathbf{H}^{'}$. The general form of these columns can be written as

$$\left(\begin{array}{ccccc} h_{1,n+1} & h_{1,n+2} & h_{1,n+3} & \dots & h_{1,N} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ h_{m,n+1} & h_{m,n+2} & h_{m,n+3} & \dots & h_{m,N} \\ \hline 1 & 0 & 0 & \dots & 0 \\ h_{m+2,n+1} & 1 & 0 & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ h_{M,n+1} & h_{M,n+2} & h_{M,n+3} & \dots & 1 \end{array}\right).$$

It is easy to see from the above submatrix that the last $M - m$ rows are in a form equivalent to the echelon form. This observation implies that any number of auxiliary variables in erasures can be decoded using these auxiliary check equations. In other words, every auxiliary variable $x_u \in \mathcal{S}^{'}$ can be decoded by using the check equations S_{m+1} to S_M (in that order). But this contradicts the assumption that $\mathcal{S}^{'}$ is a stopping set. Hence, the theorem is proved. □

Hence, any stopping set in $\mathbf{H}^{'}$ contains at least one standard variable.

The following theorem shows that for any stopping set $\mathcal{S}^{'}$ in $\mathbf{H}^{'}$, there is a corresponding stopping set $\mathcal{S}$ in $\mathbf{H}$. This stopping set in $\mathbf{H}$ is the set of all standard variables in $\mathcal{S}^{'}$. This result is important to show that removing four-cycles cannot hurt the performance of the iterative decoder.

Theorem 11. *For any stopping set $\mathcal{S}'$ in $\mathbf{H}'$, define a set $\mathcal{V}$ such that*

$$\mathcal{V} := \{x_v \in \mathcal{S}' \text{ and } v \leq n\}.$$

Now, $\mathcal{V}$ is a stopping set in $\mathbf{H}$.

Proof. Let $\mathcal{N}(\mathcal{S}')$ be the set of all neighbors of $\mathcal{S}'$ in $\mathcal{G}'$. Similarly, $\mathcal{N}(\mathcal{V})$ is the set of all neighbors of $\mathcal{V}$ in $\mathcal{G}$. For the purpose of clarity, we use $S_j{}'$ to represent the j-th row of $\mathbf{H}'$, and S_i to represent the i-th row of $\mathbf{H}$.

The set $\mathcal{V}$ is nonempty because $\mathcal{S}'$ should contain at least one standard variable, see Theorem 10. If $\mathcal{S}'$ does not contain any auxiliary variable, then the theorem is trivially true. To prove the non-trivial case, assume that $\mathcal{V}$ is not a stopping set. This means that there exists at least one check, say S_i, in $\mathcal{N}(\mathcal{V})$ that is connected to exactly one variable in $\mathcal{V}$. Let $x_u \in \mathcal{V}$ be the neighbor of S_i. Then, let us consider the following two possible cases:

- If x_u in S_i is not involved in any four-cycle with any other variable in S_i, then the element in position (i, u) of $\mathbf{H}'$ is a 1. This means that $S_i{}'$ is in $\mathcal{N}(\mathcal{S}')$ because $x_u \in \mathcal{V}$ implies $x_u \in \mathcal{S}'$ (from the definition of $\mathcal{V}$). And x_u is the only neighbor of $S_i{}'$ in $\mathcal{N}(\mathcal{S}')$. So, $\mathcal{S}'$ cannot be a stopping set. But this contradicts the assumption and hence, $\mathcal{V}$ is a stopping set.

- If x_u in S_i is involved in at least one four-cycle, then $S_i{}'$ in $\mathbf{H}'$ involves an auxiliary variable whose components include x_u. This implies that $S_i{}'$ in $\mathcal{N}(\mathcal{S}')$ has exactly one neighbor in $\mathcal{S}'$. But this contradicts the assumption that $\mathcal{S}'$ is a stopping set. Hence, $\mathcal{V}$ is a stopping set.

Hence, for any stopping set $\mathcal{S}'$ in $\mathbf{H}'$, there is an equivalent stopping set $\mathcal{V}$ in $\mathbf{H}$. □

Since $\mathcal{V}$ has been shown to be a stopping set, we will refer to it as a stopping set $\mathcal{S}$ equivalent to that of $\mathcal{S}'$ in $\mathbf{H}'$. It is important to notice that the number of standard

variables in $\mathcal{S}$ is equal to that in $\mathcal{S}^{'}$. This means that a decoding failure due to $\mathcal{S}^{'}$ will have the same impact on BER and BLER as that of $\mathcal{S}$. On the other hand, it is easy to show that a stopping set in $\mathbf{H}$ does not necessarily have an equivalent stopping set in $\mathbf{H}^{'}$. In fact, the example presented in this section demonstrates this fact. Hence, it can be concluded that the performance of the iterative decoder working on $\mathcal{G}^{'}$ is at least as good as that working on $\mathcal{G}$. In fact for most cases, simulation results indicate that an improvement in performance is obtained by iteratively decoding on $\mathcal{G}^{'}$.

5.3.2 Results

The $(23, 12)$-Golay code is an algebraic code that is considered not suitable for iterative decoders. We obtained four-cycle free GPCMs of the code by iteratively applying the algorithm on the conventional matrix representations of the code. The performance curves of $(23, 12)$-Golay code, shown in Fig. 5.1, indicate a decrease in BERs and BLERs when decoded using four-cycle free GPCMs of the code. The Golay code (GC) can be represented by a parity-check matrix with 11 check equations, referred to as *original GC*, and by a parity-check matrix with 23 check equations, referred to as *extended GC*. The original GC has 88 ones that contribute to 190 four-cycles. The extended GC has 184 ones that contribute to 598 four-cycles. By applying the algorithm, we removed all four-cycles in the original GC and the extended GC and hence, obtained two GPCMs, referred to as *transformed original GC* and *transformed extended GC* respectively. The transformed original GC has 96 ones that is just 8 more than that in the original GC. The transformed extended GC has 209 ones that is just 25 more than in the extended GC. The performance of the code obtained by decoding with each one of these representations is compared with that of the maximum-likelihood decoding. The maximum-likelihood decoding on BEC is performed by iteratively decoding on a bipartite graph obtained from a set of all possible constraints defining the GC. Note that the size of the transformed

matrices are small compared to those presented in [112].

5.4 Iterative Decoding of Linear Block Codes over AWGN Channel

An MPD based on the sum-product algorithm is a sub-optimal (in general) and low-complexity decoder that works by passing soft-messages (log-likelihood ratio in particular) over the edges of a graph representing the code $\mathcal{C}$. This decoding technique is optimal in a maximum *a posteriori* sense if and only if the graph of $\mathcal{C}$ is a tree (graph with no cycles) [56]. In general, good codes have several cycles in their graph representations [36]. Although the MPD working on such graphs is a sub-optimal decoding technique, its performance is very good when girths of these graphs are large.

The algorithm presented in Section 5.1 helps to increase the girth of a graph from 4 to 6. Let $\mathcal{G}'$ be a four-cycle free representation of $\mathcal{C}$. The following example illustrates that $\mathcal{G}'$ can be a more suitable representation for the iterative decoder than $\mathcal{G}$.

Example 3: Let S_1 and S_2 be the constraints of a binary code of length 6. Each codeword, represented as $\mathbf{x} = \langle x_1, x_2, x_3, x_4, x_5, x_6 \rangle$, satisfies the following constraints S_1 and S_2:

$$\begin{aligned} S_1\text{: } x_1 + x_2 + x_3 + x_4 &= 0 \\ S_2\text{: } x_1 + x_2 + x_5 + x_6 &= 0. \end{aligned}$$

Note that the graph of the code has 6 variable nodes and 2 check nodes. It is obvious that variable nodes x_1 and x_2 participate in a four-cycle. Let $\mathbf{y}$ be the received vector when $\mathbf{x}$ is sent over an AWGN channel. Based on $\mathbf{y}$, and the constraints, S_1 and S_2, the i-th bit of the decoded vector $\hat{\mathbf{x}}$ is a 1 if it maximizes the *a posteriori* probability that is written as

$$\Pr(x_i = 1|\mathbf{y}, S_1, S_2) = \frac{\Pr(S_1, S_2|x_i = 1, \mathbf{y})\Pr(x_i = 1|\mathbf{y})\Pr(\mathbf{y})}{\Pr(\mathbf{y}, S_1, S_2)}.$$

For such a simple case, the probability of S_1 and S_2 being satisfied given $x_1 = 1$ and $\mathbf{y}$ can be calculated by brute force, and it is given by

$$\begin{aligned}\Pr(S_1, S_2|x_1 = 1, \mathbf{y}) = &\Big\{q_2(q_5p_6 + p_5q_6)(q_3p_4 + p_3q_4)\Big\} \\ &+ \Big\{p_2(p_3p_4 + q_3q_4)(p_5p_6 + q_5q_6)\Big\},\end{aligned}$$

where $p_i = \Pr(x_i = 1|\mathbf{y})$ and $q_i = 1 - p_i$.

The conventional sum-product algorithm assumes that the constraints are mutually independent, and hence, the probability of S_1 and S_2 being satisfied given $x_1 = 1$ is $\Pr(S_1|x_1 = 1, \mathbf{y})\Pr(S_2|x_1 = 1, \mathbf{y})$. The above assumption is not true because x_1 and x_2 are involved in both S_1 and S_2. This means that the *a posteriori* probability of x_1 calculated by iteratively applying (a finite number of iterations) the sum-product algorithm is not accurate.

In order to remove the four-cycle in the set of constraints, we replace $x_1 + x_2$ in S_1 and S_2 with an auxiliary variable $x_{1,2}$. The modified set of constraints are

$$\begin{aligned} S_1{}' &: \; x_{1,2} + x_3 + x_4 &&= 0 \\ S_2{}' &: \; x_{1,2} + x_5 + x_6 &&= 0 \\ S_3{}' &: \; x_{1,2} + x_1 + x_2 &&= 0. \end{aligned}$$

Note that the corresponding bipartite graph is a tree. The probability of S_1 and S_2 being satisfied given $x_1 = 1$ and $\mathbf{y}$ can be written as

$$\begin{aligned}\Pr(S_1, S_2|x_1 = 1, \mathbf{y}) = &(q_2\Pr(S_1^{'}, S_2^{'}|x_{1,2} = 1, \mathbf{y})) \\ &+ (p_2\Pr(S_1^{'}, S_2^{'}|x_{1,2} = 0, \mathbf{y})),\end{aligned} \tag{5.2}$$

where

$$\begin{aligned}\Pr(S_1{}', S_2{}'|x_{1,2} = 1, \mathbf{y}) &= (q_3p_4 + p_3q_4)(q_5p_6 + p_5q_6) \\ \Pr(S_1{}', S_2{}'|x_{1,2} = 0, \mathbf{y}) &= (p_3p_4 + q_3q_4)(p_5p_6 + q_5q_6).\end{aligned}$$

From (5.2), we see that the introduction of the auxiliary variable has facilitated the calculation of the exact *a posteriori* probability of x_1. ■

The sum-product algorithm is applied on $\mathcal{G}'$ after initializing the log-likelihood ratios of the auxiliary variables to zero. Since it is difficult to theoretically analyze the effects of removing four-cycles, we present Monte-Carlo simulation results that demonstrate the improvement in BER and BLER performance achieved by decoding iteratively (at most 100 iterations) on four-cycle free representations of algebraic codes.

5.4.1 Results for Algebraic Codes

The performance curves of $(31, 26)$-Hamming code, shown in Fig. 5.2, were obtained by running the message-passing algorithm on two different representations of the code: conventional 5×31-matrix representation, and transformed 25×51-GPCM representation. The conventional representation has 80 ones that contribute to 280 four-cycles. The transformed representation is a four-cycle free GPCM obtained at the cost of adding 18 more ones. The performance gain obtained from using the transformed-GPCM increases with an increase in signal-to-noise ratio (SNR). For instance, the gain at a BER of 10^{-3} is approximately 0.25 dB and it increases to approximately 0.5 dB at a BER of 10^{-5}. The BER performance of these two representation under the MPD is compared with that of a symbol-by-symbol maximum *a posteriori* decoder on the trellis of the code. At a BER of 10^{-5}, the BER performance of the transformed-GPCM is 0.25 dB away from that of the optimal decoder.

The performance curves of a $(930, 472)$-LDPC code, obtained from superposition construction based on Euclidean geometry (EG) [110], were obtained by running the message-passing algorithm on two different representations of the code: conventional 465×930-matrix representation, and transformed 930×1395-matrix representation. Each variable in the conventional matrix is involved in exactly one four-cycle, and there are 465

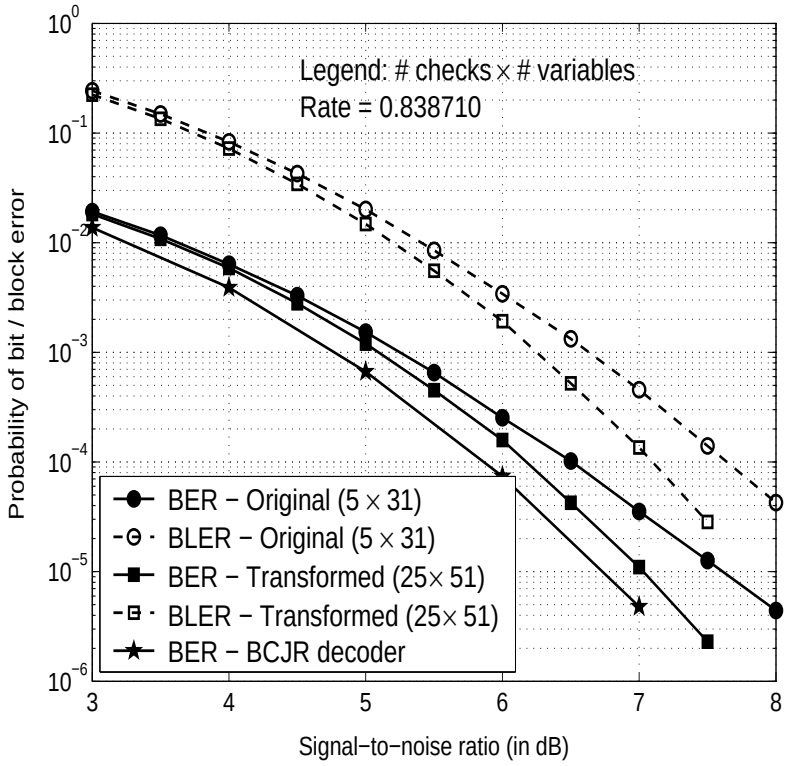


Figure 5.2: Comparison of performance of $(31, 26)$-Hamming code on AWGN channel under MPD and BCJR decoder

four-cycles altogether. Note that error-floor in the performance of GPCM representation is an order of magnitude lower than that of the original representation.

The results thus far agree with the intuition that increasing the girth of a TG helps to improve the performance of the corresponding code. Contrary to this belief, Yedidia *et al.* had presented a special case of the $(23, 12)$-Golay code whose performance deteriorates significantly when iteratively decoded with transformed GPCMs [112]. The performance results that we obtained for the Golay code by running the message-passing algorithm on a four-cycle free graph of the code agree with their observations. But, we would like to note that the size of our transformed matrix is at least 25 times smaller than that of their GPCMs.

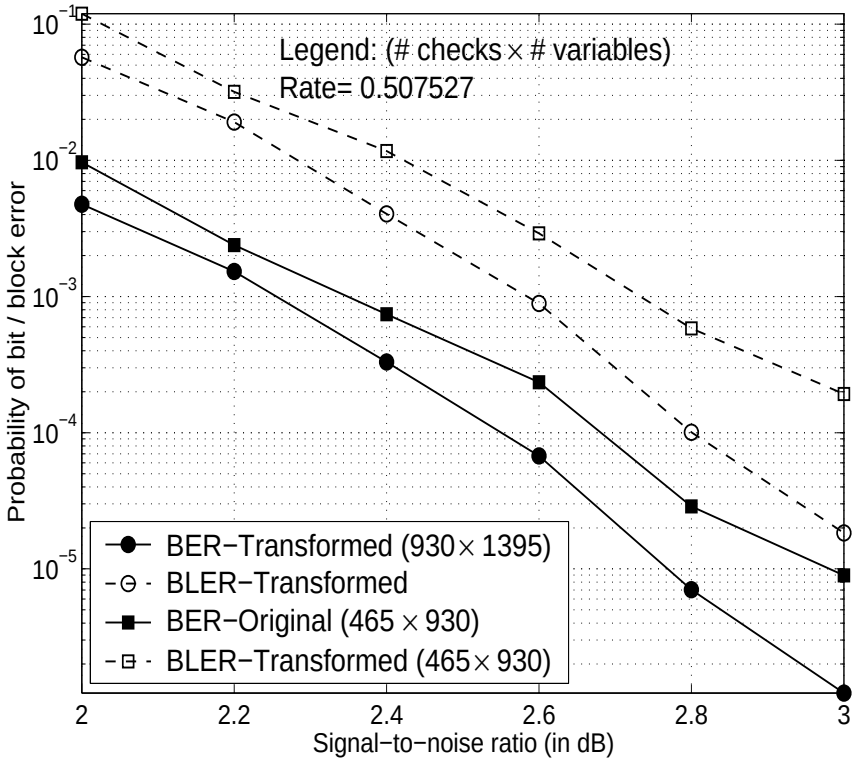


Figure 5.3: Performance of different representations of $(930, 472)$-EG superposition code on AWGN channel

5.4.2 Results for Reed-Solomon Codes

The binary representation of the parity-check matrix of Reed-Solomon codes is derived from equivalence established by Vardy and Be'ery in [97]. In [97], binary equivalent of a (n, k, d) RS code over GF(2^m) is established as a set of all cosets of direct sums of m identical binary Bose-Chaudhuri-Hocquenghem (BCH) codes [12], [48] of length n. The binary generator matrix $\mathbf{G}$ of the RS code is a $mk \times mn$ matrix that exhibits the

following structure:

$$\mathbf{G} = \left(\begin{array}{ccccc} \mathbf{B} & \mathbf{0} & \mathbf{0} & \dots & \mathbf{0} \\ \mathbf{0} & \mathbf{B} & \mathbf{0} & \dots & \mathbf{0} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \dots & \mathbf{B} \\ \hline \dots & \dots & \text{Glue vectors} & \dots & \dots \end{array}\right). \tag{5.3}$$

The matrix $\mathbf{B}$ is the generator matrix of a (n, k) BCH code. The glue vectors are $m(n-k)$ linearly independent vectors in the RS code but not in the BCH code. The rank of $\mathbf{G}$ is km. Since the dual of an RS code is also an RS code, the generator matrix of the (n, k, d) RS code is the binary parity-check matrix of its dual. The four-cycle free GPCM (or transformed representation) is constructed from the binary parity-check matrix and is used to iteratively decode the RS code.

In Fig. 5.4, the performance of the iterative decoder using the transformed representation of a $(7, 2)$ RS code is compared with those of the hard-decision decoder and the ML decoder. The performance of the decoder is just 1dB away from the ML performance. In Fig. 5.5, the performance of the iterative decoder using the transformed representation of a $(7, 3)$ RS code is compared with those of the hard-decision decoder and the ML decoder. The performance of the decoder is about 0.5dB away from the ML performance. The gain over the conventional representation (labeled RS in the figure) is just about 0.1dB. In Fig. 5.6, the performance of the iterative decoder using the transformed representation is worse than that using the conventional representation. This example illustrates that the four-cycle free representation is not beneficial all the time for the AWGN channel. Also, we have observed that decoding iteratively over four-cycle free graphs of binary representations of Reed-Solomon codes is beneficial only if they are short and have low code rates.

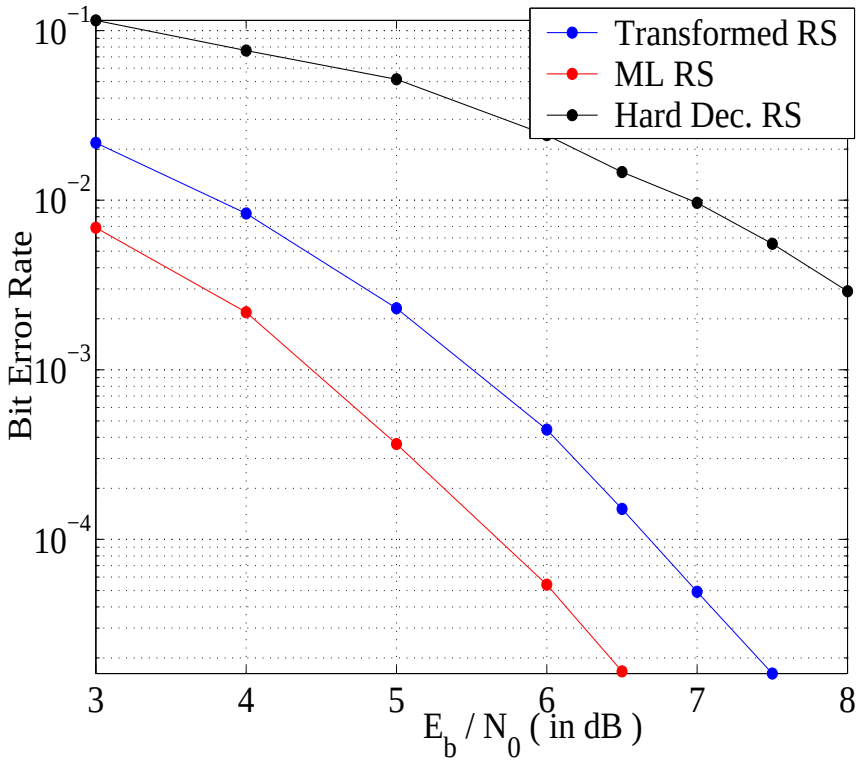


Figure 5.4: Performance of $(7, 2)$-RS code on the AWGN channel

5.5 Discussion

The results indicate that this method of removing four-cycles improves the suitability of iterative decoders for conventional algebraic codes. The limited success of the GPCM representation with the SPA opens up avenues for further investigation. Another logical extension of this work is to develop a general algorithm to remove cycles of any specified length. In fact, we have been able to use a technique, similar to the one presented here, to remove all six-cycles from small graphs. Specifically, we have observed a fractional improvement in performance by iteratively decoding on the TG obtained by removing all six-cycles and four-cycles from a $(7, 3)$-LDPC code, constructed from the point-line incidence matrix of the smallest projective plane.

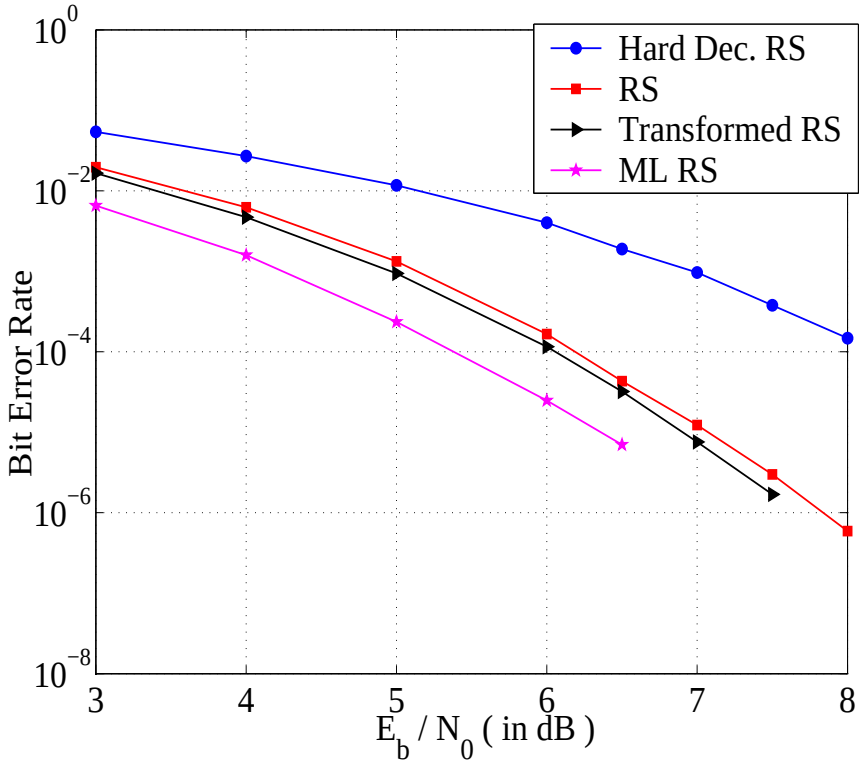


Figure 5.5: Performance of $(7, 3)$-RS code on the AWGN channel

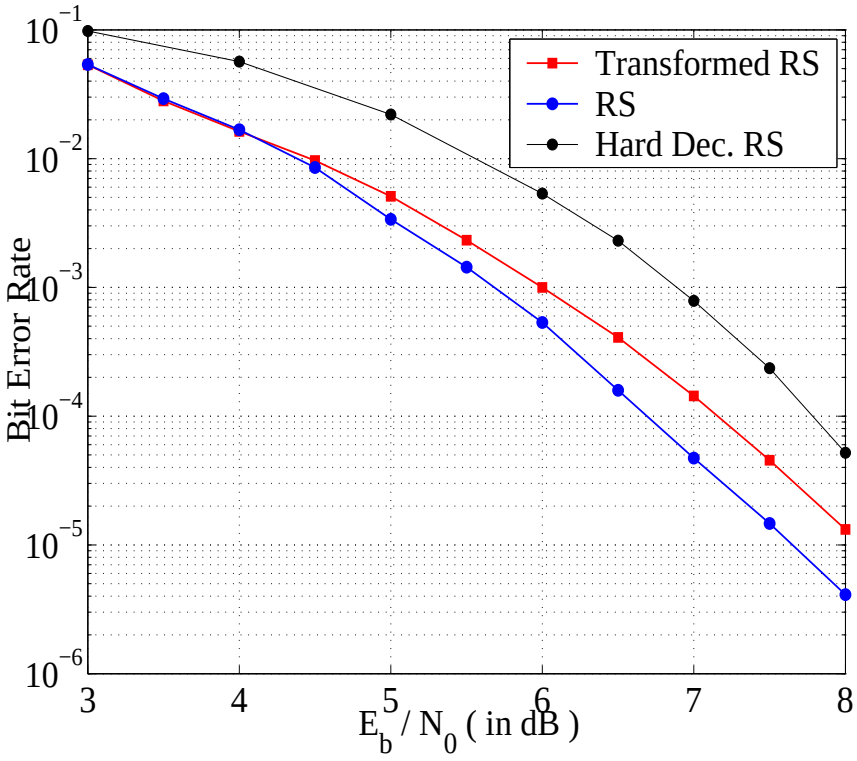


Figure 5.6: Performance of $(15, 6)$-RS code on the AWGN channel

CHAPTER 6

FAILURES OF THE GALLAGER B DECODER

Since Gallager's thesis [37] on low-density parity-check (LDPC) codes and iterative decoding algorithms, researchers have developed a whole array of techniques to construct "good" LDPC codes and analyze the performance thresholds [74] of LDPC ensembles under the assumption that the girth goes to infinity. Unfortunately, the same level of maturity has not been attained in the characterization of iterative decoding of finite-length LDPC codes. Wiberg [107] developed techniques by defining computation trees that achieved some success in the regime of finite-length analysis. Frey *et al.* used computation trees [40] to characterize decoding boundaries and pseudocodewords of iterative decoders. The pseudocodewords [42], [53] include words that are not codewords and they are local minima of the decoding algorithm. Vontobel and Koetter [103] have analyzed all finite covers of Tanner graphs to explain decision boundaries of linear programming decoders and to characterize local minima of message-passing decoders. Chernyak *et al.* [17] used an instanton approach to analyzing finite-length LDPC codes.

Richardson [73] with an aim to estimate the performance of LDPC codes developed a semi-analytical technique that relies on combinatorial objects called trapping sets. The idea of a trapping set is similar to that of near codewords defined in [68]. In this paper, we focus on understanding and characterizing the dynamics of an iterative decoder, namely the Gallager B decoder, for the binary symmetric channel. The state transitions of the decoder are classified into two groups and such a classification is used to characterize failures of the decoder. Finally, we use this knowledge to semi-analytically estimate the performance of LDPC codes from a class of designs.

6.1 One-Step Majority-Logic Decoder

It is easy to show that one iteration of the Gallager B algorithm is similar to a majority-logic decoder and the number of estimates used in the decoding stage is clarified in the following facts.

Fact 12. *For a code with column weight γ, the decoding decision for a variable after one iteration is based on $\gamma + 1$ estimates, if γ even, and γ estimates, if γ odd.* ■

Using the derivation in [77], it is possible to achieve the error correction capability of one iteration of the Gallager B algorithm working on a code from $2-(v, k, 1)$ BIBD.

Theorem 13 (Extension of Rudolph's Result)**.** *For a code from $2-(v, k, 1)$ BIBD with column weight γ, a single iteration of the Gallager B decoder can correct up to $\frac{\gamma-1}{2}$, if γ odd, and $\frac{\gamma}{2}$, if γ even.* ■

Therefore, for codes from $\mathrm{PG}(2, q = 2^m)$, the decoder (in one step) can correct all error patterns up to weight $t = \frac{q}{2} = \lfloor \frac{d_{\min}-1}{2} \rfloor$. For codes from $\mathrm{AG}(2, q = 2^m)$, the decoder can correct all error patterns up to weight $t = \frac{q}{2} = \lfloor \frac{d_{\min}-1}{2} \rfloor$.

6.1.1 Performance of Codes from $\mathrm{PG}(2, q = 2^m)$

The bit error rate (BER) performance of codes from $\mathrm{PG}(2, q = 2^m)$ decoding using the 1-step majority-logic decoder can be analytically computed. Let $\mathcal{C}$ be a regular LDPC code of length $n = q^2 + q + 1$ from $\mathrm{PG}(2, q = 2^m)$ with column weight $\gamma = q + 1$ and $\rho = \frac{kn}{v} = q + 1$. For convenience, let γ be odd. Let $\mathbf{x} = \mathbf{0} \in \mathcal{C}$ is transmitted over the channel and the received word is $\mathbf{y}$. Without loss of generality, let us decode y_i, i^{th} bit of the vector $\mathbf{y}$, using the majority logic.

Lemma 14. *Each x_i, the i^{th} variable node, is involved in γ check equations such that any other variable x_j, $j \neq i$, is involved in exactly one of these check equations.*

Proof. Since any given line intersects every other line of the plane exactly once, the statement of the lemma is true. □

Therefore, each estimate of x_i is a sum of $\gamma - 1$ unique variables and there are γ such estimates.

The probability of $\hat{x}_i$ being in error after decoding can be written as

$$p_{\mathrm{b}} = \sum_{N_e} \Pr(\hat{x}_i = 1 | N_e \text{ errors}) \Pr(N_e \text{ errors}). \tag{6.1}$$

Further, the conditional probability can be written as

$$\begin{aligned} &\Pr(\hat{x}_i = 1 | N_e \text{ errors}) = \\ &\{\Pr(\hat{x}_i = 1 | N_e - 1 \text{ other errors}, y_i = 1) \cdot \Pr(y_i = 1 | N_e \text{ errors})\} + \\ &\{\Pr(\hat{x}_i = 1 | N_e \text{ other errors}, y_i = 0) \cdot \Pr(y_i = 0 | N_e \text{ errors})\}. \end{aligned} \tag{6.2}$$

The conditional probability $\Pr(\hat{x}_i = 1 | N_e - 1 \text{ other errors}, y_i = 1)$ is the probability that the estimate of x_i is wrong given that y_i is in error and $N_e - 1$ other errors and this probability can be combinatorially computed. The total number of ways to distribute $N_e - 1$ errors among $n - 1$ variables is simply $\binom{n-1}{N_e - 1}$. A fraction of the total number of error configurations that result in $\hat{x}_i = 1$ is computed by running through all partitions of $N_e - 1$ with $\gamma + 1$ or less components such that each component is less than or equal to γ. The conditional probability $\Pr(\hat{x}_i = 1 | N_e \text{ other errors}, y_i = 0)$ can be computed in a similar fashion.

Due to Fact 12, it is sufficient to consider values for N_e that are greater than t. For practical purposes, it is generally sufficient to consider values for N_e that are less than $t + 15$ because $\Pr(N_e \text{ errors})$ becomes negligible for $N_e > t + 15$. This is especially true for smaller values of α. The procedure described here can be used for cases where γ is even. The BER of codes from $\mathrm{PG}(2, 2^3)$, $\mathrm{PG}(2, 2^4)$, $\mathrm{PG}(2, 2^5)$ and $\mathrm{PG}(2, 2^6)$ are shown in Fig. 6.1.

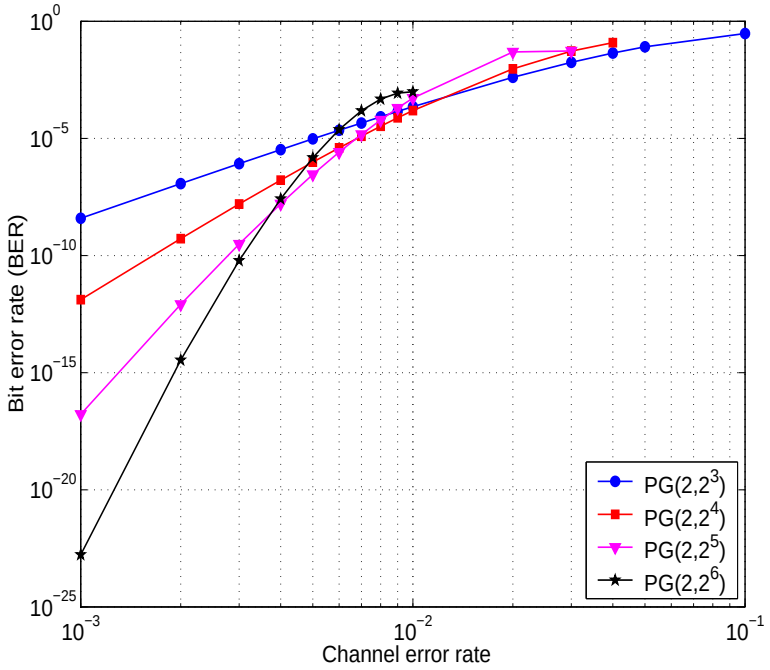


Figure 6.1: Analytically calculated BER of PG(2,2^4) and PG(2,2^5) decoded using one-step majority logic decoder

The method described here can be extended to compute the BER of codes from $2-(v,k,1)$ BIBDs. Unfortunately, this analytical technique cannot be easily extended to characterize decoder behavior for 2 or more iterations of the Gallager B algorithm, since the errors patterns are highly correlated after the 1^{st} iteration.

6.2 State Transitions of the Gallager B Decoder

Let the i^{th} state of the Gallager B decoder is the decoded word obtained after the i^{th} round or iteration. The transition from an i^{th} state to an $(i+1)^{\text{th}}$, given an initial state, depends on the Tanner graph representation of the code. The state transitions of the decoder can be *periodic* or *aperiodic*.

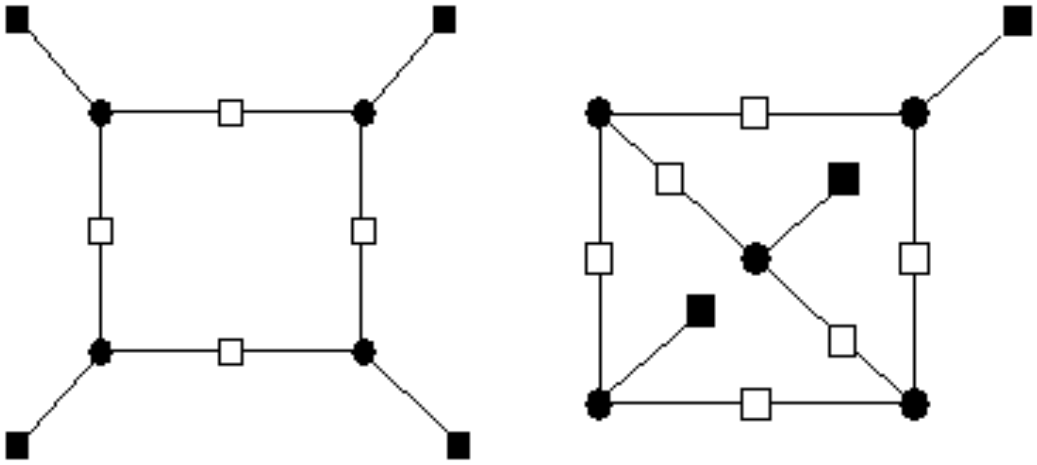
6.2.1 Periodic Transitions

A periodic transition of states with period $T \neq 0$ is said to occur if the $(i+kT)^{\text{th}}$ states, for some $i \geq i_0$ and $\forall\ k \geq 0$, are equal. Let us first focus on the periodic transition with $T = 1$.

The initial state of the decoder that causes periodic transition with $T = 1$ and $i_0 = 0$ is referred to as the *fixed point*. A fixed point of the decoder is characterized by defining a set called the *trapping set*. The trapping set of a decoder with a given graph representation is the support of an initial state that is a fixed point of the decoder. For example, supports of all codewords are trapping sets of the decoder. But there are also other trapping sets as shown in the example below. A (v, c) trapping set $\mathcal{T}$ is a set of v variable nodes whose induced subgraph has c unsatisfied checks.

Example 1: Consider the $(2640, 1320)$ Margulis code [62], [76] which has girth 8 and column weight 3. From the structure of the code we can calculate the number of eight-cycles in the graph to be 1320. Fig. 6.2(a) illustrates the structure of an eight-cycle. Let the support of the initial state of the decoder be the variable nodes of the eight cycle. Each variable node receives a message of 1 from two check nodes and a message of 0 from one check node. If majority logic is taken for decoding, these variable nodes will be decoded as 1. The same messages are passed at every iteration and the decoder fails to correct these errors. Hence, an eight cycle is a $(4, 4)$ trapping set for the Margulis code. Similarly, the Fig. 6.2(b) illustrates another trapping set of the code. The $(12, 4)$ trapping set for the AWGN channel from [73] is also a trapping set for the Gallager B algorithm.

A minimal trapping set $\mathcal{T}_{\text{M}}$ of a code is a trapping set with the smallest possible cardinality. A bound on the cardinality of minimal trapping set of a regular code with girth g and column weight γ is given in (6.3), where $x = \gamma$ if γ odd and $x = \gamma + 1$ if



(a) A $(4,4)$ trapping set (b) A $(5,3)$ trapping set

Figure 6.2: Illustration of Trapping sets

γ even. In Table 6.1, we show the size and number of small trapping sets of certain $(3,6)$ regular codes.

$$|\mathcal{T}_{\mathrm{M}}| \geq \begin{cases} 2 & g=4 \\ \lceil \frac{x}{2} \rceil + 1 & g=6 \\ 2\lceil \frac{x}{2} \rceil & g=8 \\ 1+\sum_{i=0}^{\frac{g-6}{4}} \lceil \frac{x}{2} \rceil (\lceil \frac{x}{2} \rceil - 1)^i & g \geq 10, \frac{g}{2} \text{ odd} \\ 1+\sum_{i=0}^{\frac{g-8}{4}} \lceil \frac{x}{2} \rceil (\lceil \frac{x}{2} \rceil - 1)^i + (\lceil \frac{x}{2} \rceil - 1)^{\frac{g-4}{4}} & g \geq 10, \frac{g}{2} \text{even}, \end{cases} \tag{6.3}$$

Note that subgraphs induced by trapping sets are either simple cycles or unions of simple cycles. For example, a $(12,4)$ trapping set of the $(2640,1320)$ Margulis code [73] is a set of 12 variable nodes and the induced subgraph has 4 odd-degree check nodes. The illustration of the subgraph induced by the trapping set, see Fig. 6.2.1, shows that the subgraph can be factored into a set of four simple 12 cycles, namely $\{v_1 - v_2 - v_3 - v_4 - v_5 - v_6, v_1 - v_2 - v_3 - v_7 - v_8 - v_9, v_7, v_8 - v_9 - v_{10} - v_{11} - v_{12}, v_4 - v_5 - v_6 - v_{10} - v_{11} - v_{12}\}$. Also, it can be visualized as the interaction of two simple eight cycles, namely $\{v_1 - v_9 - v_{12} - v_6\}$ and $\{v_3 - v_7 - v_{10} - v_4\}$.

The periodic transitions with $T > 1$ can be explained based on trapping sets. An

Table 6.1: SMALL TRAPPING SETS OF REGULAR CODES

Code	**Variables**	**Checks**	**Girth**	**Dominant Trapping Sets**	**No. of Trapping Sets**	M
MacKay's code one	1008	504	6	(3,3);(4,4);(5,3)	165; 1215; 14	≈ 30
MacKay's code two	816	408	6	(3,3);(4,4);(5,3)	132; 1372; 41	≈ 25
Margulis code	2640	1320	8	(4,4);(5,5)	1320; 11088	≈ 95
Tanner code	155	93	8	(5,3)	155	≈ 10
QC code one	900	450	8	(5,3);(4,4)	50;675	≈ 25
QC code two	900	450	6	(4,2);(4,4)	150;1125	≈ 25
QC code three	900	450	6	(6,2);(4,4);(3,3);	150;1025;200	≈ 25

initial state that induces such a periodic transition has the support which is a subset of some trapping set. Also, the union of supports of decoder states over an entire period of an oscillation (or transition) is a trapping set.

6.2.2 Aperiodic Transitions

Any sequence of state transitions that do not exhibit periodicity is classified as *aperiodic*. Note that if the support of the initial state is a codeword, then aperiodic state transitions are not possible. Thus if a channel error vector, also the initial state, is nonzero, then aperiodic transitions lead to the propagation of errors across iterations. The support of an initial state that propagates errors is called a *propagating set*. Quite contrary to popular belief, subgraphs induced by minimal propagating sets have been observed to be cycle-free. For example, at high SNR most decoder failures of a $(3195, 2844)$ array code with $\gamma = 5$ are due to cycle-free subgraphs isomorphic to those illustrated in Fig. 6.4. The variable nodes of these cycle-free subgraphs form propagating sets.

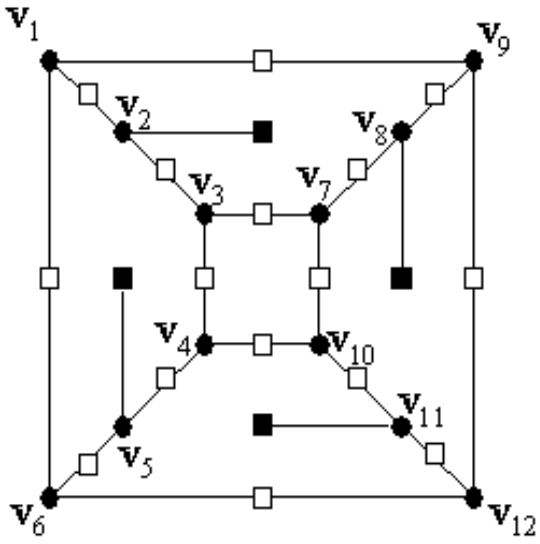


Figure 6.3: A $(12, 4)$ trapping set of the $(2640, 1320)$ Margulis code

6.3 Failures of the Gallager B Decoder

A *decoding failure* is said to have occurred when the decoded word is different from the transmitted codeword. The characterization of failure events is essential to quantify the performance of a decoder. For example, a maximum likelihood decoder fails if and only if the received word is closer to a codeword that is not the same as the transmitted one. The analysis of failures in Gallager B decoder is dependent on the properties of the code and its graph representation. The failures of the decoder can be due to minimum distance error patterns or low (relative to minimum distance) weight error patterns. These failures can be characterized using trapping sets and propagating sets.

Assume that the all-zero codeword is transmitted over the channel. An error vector (introduced by the channel) can result in one of the following possibilities at the decoder.

1. The decoder, after a finite number of iterations l, corrects for the error and arrives at the all-zero state. This is a favorable situation where the decoder exhibits a periodic state transition with $T = 1$ from $i = l$.

2. The decoder, after a finite number of iterations l, exhibits a periodic state transition with $T \geq 1$ from $i = l$ and it never reaches the all-zero state. This is not a favorable situation because this constitutes a decoder failure. Such failures are characterized, as discussed in Section 6.2.1, by trapping sets.

3. The decoder, after a finite number of iterations l, exhibits an aperiodic state transition from $i = l$ and it does not reach the all-zero state. Such a failure is characterized, as discussed in Section 6.2.1, by a propagating set.

The situation described in Item 2 occurs when errors are located (as a result of the channel, i.e. $l = 0$) or transferred (as a result of l iteration) to a set of variable nodes that is contained in a trapping set. For example, it is sufficient for a select three variable nodes of a $(5, 3)$ trapping set in error to result in periodic state transition with $T = 2$. Although minimum distance errors are accounted for in this category, "good" codes exhibit trapping set failures that are not minimum distance failures. Thus the probability of such failures depends on the size of trapping sets and on the population of trapping sets.

The situation described in Item 3 occurs when errors are located in a set of variable nodes that contains a propagating set. It is common for propagating set failures to corrupt approximately half of all the variable nodes. The probability of such failures depends on the size of propagating sets and on the population of propagating sets.

6.4 Application of Failure Analysis

The application of our interest is the estimation of the frame error rate (FER) performance of the Gallager B algorithm. We have shown that such an estimation is dependent on size and population of trapping and propagating sets. Although identifying all trapping sets and propagating sets of a code is a daunting task, especially if the code

is moderately long and pseudo-randomly constructed, we will show that it is possible to considerably reduce the complexity of estimating the performance. The reduction in complexity is achieved by identifying that in the regime of high signal-to-noise ratio (SNR) performance of the decoder is dominated by trapping and propagating sets of smaller sizes. Similar approaches have been used in estimating the performance of conventional decoders with parameters relating to minimum distance codewords in the regime of high SNR.

6.4.1 Performance of $(3, 6)$ LDPC Codes

In [18], we presented a semi-analytical method to estimate the performance of a class of $(3, 6)$ LDPC codes whose decoding failures are dominated by small trapping sets. For example, the performance of the Margulis code, listed in Table 6.1, is dominated by $(4, 4)$ and $(5, 5)$ trapping sets. Similarly, the performance of one of the quasi cyclic codes is dominated by $(5, 3)$ and $(4, 4)$ trapping sets. It is not difficult to identify the dominant trapping sets by running short simulations that track decoder state transitions. The Fig. 6.5 is an illustration of subgraphs induced by other trapping sets identified in LDPC codes (of girth $g = 6, 8$) with column weight $\gamma = 3$. This illustration intends to emphasize the interaction of constituent simple cycles. These induced subgraphs are possible candidates for trapping set failures in any code with $\gamma = 3$.

The total number of the dominant trapping sets in the code can be easily computed, owing to the small sizes of these sets, by running a search algorithm. S. K. Chilappagari and B. Vasić developed an analytical method (AM) that uses information on dominant trapping sets to estimate the performance of $(3, 6)$ LDPC codes. In this report, we present a brief explanation of the method and illustrate its success. For more details, see [18].

Let $\mathcal{X}_{(a,b)}$ of a code of length n be a set of all isomorphic (a, b) trapping set in its

Tanner graph, and $|\mathcal{X}_{(a,b)}|$ is the cardinality of the set. Also, let $m_{(a,b)}$ be the minimum number of variables in a (a,b) trapping set that needs to be in error for the decoder to fail. Note that this quantity is uniquely defined for $\mathcal{X}_{(a,b)}$. In the analytical method, used to estimates in Fig. 6.6, Fig. 6.7, Fig. 6.8 and Fig. 6.9, the FER, say p_{CW}, of a code is approximated as

$$p_{\text{CW}} \approx \sum_{(a,b)} \left\{ \sum_{N_e=m_{(a,b)}}^{n} \binom{N_e}{m_{(a,b)}} \frac{|\mathcal{X}_{(a,b)}|}{\binom{n}{m_{(a,b)}}} \Pr(N_e|\alpha) \right\} \tag{6.4}$$

In (6.4), the outer summation accounts for contributions of $\mathcal{X}_{(a,b)}$ for different values of (a,b) to p_{CW}. Generally at high SNR, it is most likely for small trapping sets to make significant contributions to the frame error rates. As a rule-of-thumb, smaller the values of a and $m_{(a,b)}$ of an (a,b) trapping set, greater is its relevance to FER computation. Refer to Table 6.1 for the list of relevant trapping sets for different codes considered in the example. The fraction $\frac{|\mathcal{X}_{(a,b)}|}{\binom{n}{m_{(a,b)}}}$ in (6.4) is the probability that $m_{(a,b)}$ errors result in a decoding failure. The cardinality of $\mathcal{X}_{(a,b)}$ is determined by running a graph search algorithm. The complexity of the algorithm is dependent on various factors like n, randomness of the code construction process, topological structures that we are searching etc. In general, simple cycles of length 6, 8, and 10 are easier to count [1], [2], [45], especially in structured codes. When the channel with transition probability α introduces N_e errors, $\binom{N_e}{m_{(a,b)}} \frac{|\mathcal{X}_{(a,b)}|}{\binom{n}{m_{(a,b)}}}$ is an approximation of the probability of a decoding failure. Although the inner summation runs up to n, in most cases it is sufficient to stop it at some value $m_{(a,b)} < M < n$ where the term $\Pr(N_e|\alpha)$ becomes negligible. Refer to Table 6.1 for the M values used in the example.

In Fig. 6.6, Fig. 6.7, Fig. 6.8 and Fig. 6.9, FER estimates agree with those computed using simulation technique.

6.4.2 Performance of Codes from BIBDs

In Section 6.1, the analytical expression for calculating BER for codes derived from affine and projective planes when decoded using a one-step majority logic decoder (or a Gallager B algorithm with 1 iteration) was shown. It was pointed out that the methodology could not be extended to Gallager B decoder with 2 or more iterations

In this report, we focus on the problem of estimating the FER of codes from projective planes $\mathrm{PG}(2, 2^m)$ and affine planes $\mathrm{AG}(2, 2^m)$. The codes are constructed from the incidence matrix of $2-(q^2+q+1, q+1, 1)$ BIBDs and $2-(q^2, q, 1)$ BIBDs, where q is a power of prime, respectively. The column weight of a $\mathrm{PG}(2, q)$ code is $q+1$ and its minimum distance is $d_{\min} = q+2$. The column weight of an $\mathrm{AG}(2, q)$ is q and its minimum distance is $d_{\min} = q+1$.

The FER estimate p_{CW} of $\mathrm{PG}(2, q=2^m)$ and $\mathrm{AG}(2, q=2^m)$ codes after l iterations of the Gallager B algorithm can be written as

$$p_{\mathrm{CW}} = \sum_{N_{\mathrm{e}}=t+1}^{n} \Pr(\text{decoding failure}|N_{\mathrm{e}})\Pr(N_{\mathrm{e}}|\alpha), \tag{6.5}$$

where α is the parameter of the binary symmetric channel and N_{e} is the number of errors introduced by the channel. The size of minimal trapping sets of these codes is $t+1$. For example, the subgraph induced by one such minimal trapping set is illustrated in Fig. 6.10. In Fig. 6.10, each line corresponds to a variable node of the code from $\mathrm{PG}(2, 2^4)$ and each point corresponds to a check node of the code. From simulations we observe that the dominant error events are due to propagating sets and not trapping sets. Due to the large sizes of trapping and propagating sets, it is not possible to count them and hence, we resort to a semi-analytical method (SAM) to estimate the FER of these codes.

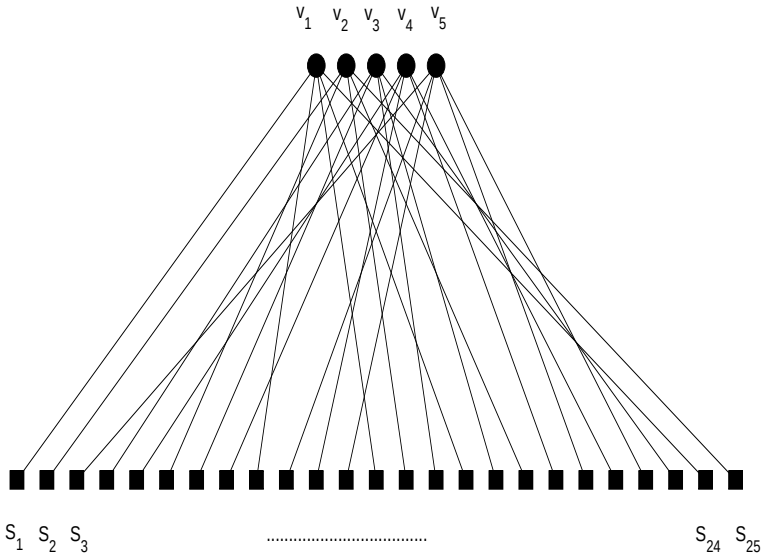
The conditional probability of decoding failures is obtained from simulations. A fixed number of errors, say N_{e}, is introduced at random bit positions of a codeword, and the

resultant word is decoded using the Gallager B algorithm. Note that these simulations are independent of α and hence, these probabilities can be applied to estimate FERs at very high SNRs. This experiment is repeated to compute the probability of decoder failure for different values of N_{e}. In Fig. 6.11, observe that conditional probabilities for $\mathrm{PG}(2, 2^5)$ and $\mathrm{AG}(2, 2^5)$ codes approach unity for values of $N_e \approx t + 4$. This behavior holds for all codes from projective and affine planes. Therefore, the FER in (6.5) is modified as

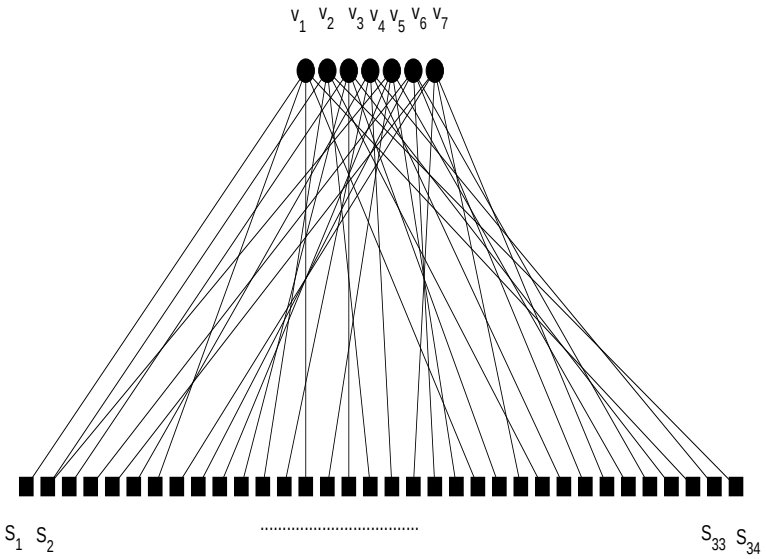
$$\begin{aligned} p_{\mathrm{CW}} \approx \; & \sum_{N_{\mathrm{e}}=t+1}^{M} \Pr(\text{decoding failure}|N_{\mathrm{e}}) \Pr(N_{\mathrm{e}}|\alpha) + \\ & \sum_{N_{\mathrm{e}}=M+1}^{n} 1 \; \Pr(N_{\mathrm{e}}|\alpha), \end{aligned} \tag{6.6}$$

for some value of M in the neighborhood of t.

In Fig. 6.12, we illustrate the success of this approach in long codes such as $(1057, 813)$ projective plane code and $(1056, 813)$ affine plane code. The estimates from the semi-analytical method agree with those obtained from simulations.

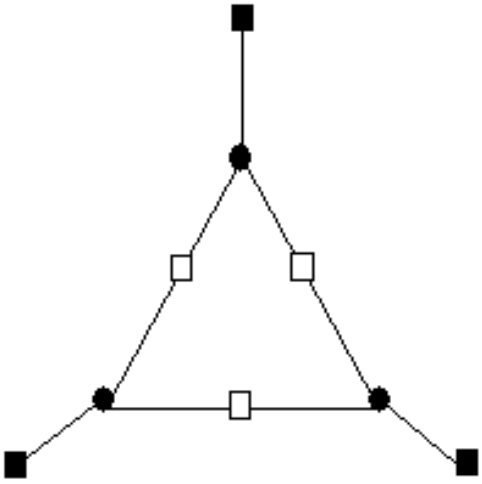


(a) A $(5, 25)$ propagating set of the $(3195, 2844)$ LDPC code with $\gamma = 5$

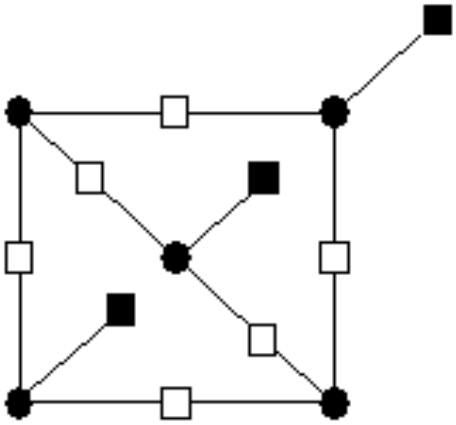


(b) A $(7, 34)$ propagating set of the $(3195, 2844)$ LDPC code with $\gamma = 5$

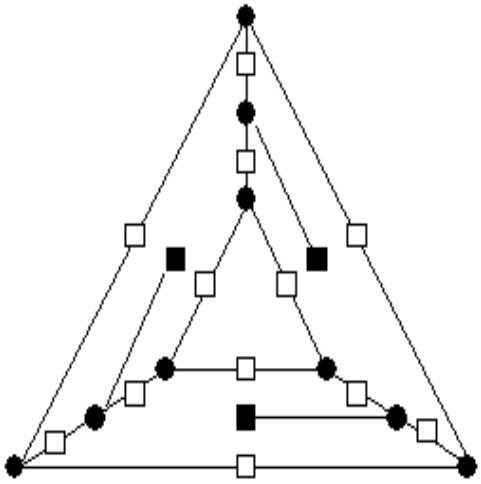
Figure 6.4: Examples of propagating sets of LDPC codes



(a) A (3,3) trapping set



(b) A (5,3) trapping set



(c) A (9,3) trapping set

Figure 6.5: Examples of trapping sets of LDPC codes with $\gamma = 3$

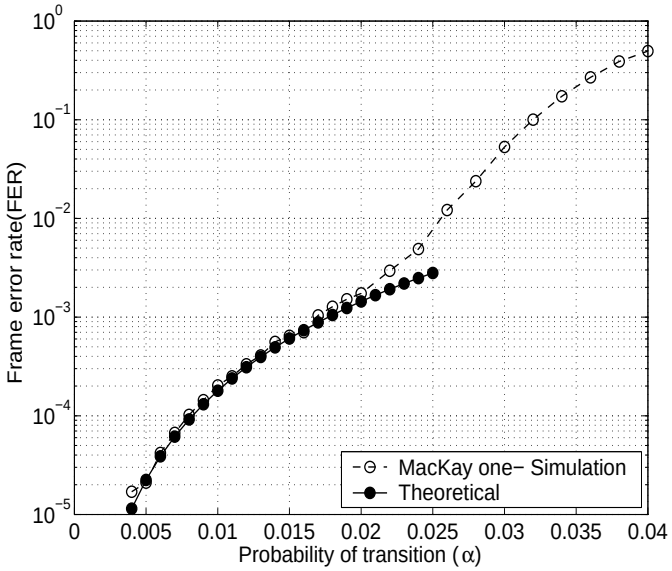


Figure 6.6: MacKay's random $(1008, 504)$ code with $\gamma = 3$ and $g = 6$ (MRC-1)

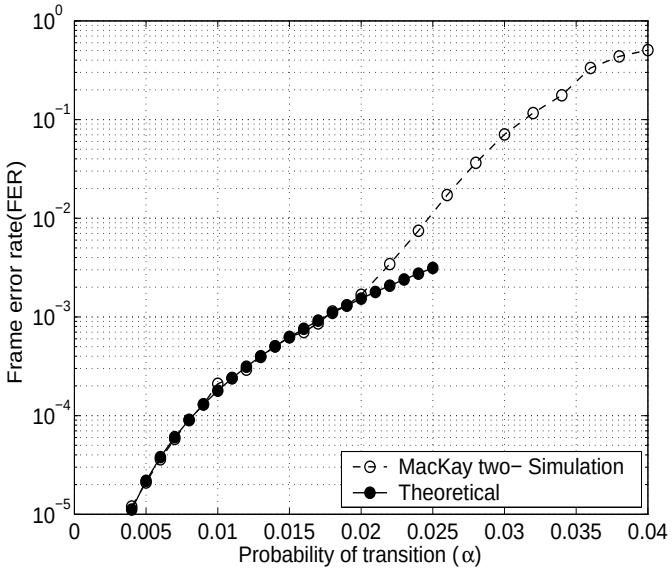


Figure 6.7: MacKay's random $(816, 408)$ code with $\gamma = 3$ and $g = 6$ (MRC-2)

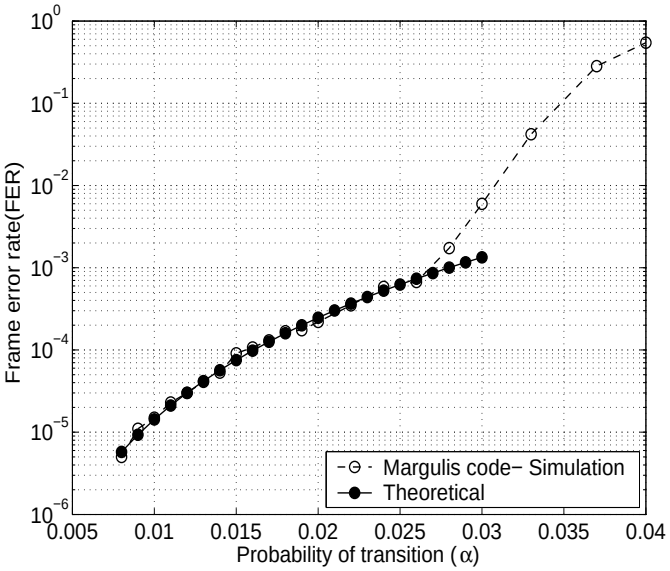


Figure 6.8: $(2640, 1320)$ Margulis code with $\gamma = 3$ and $g = 8$

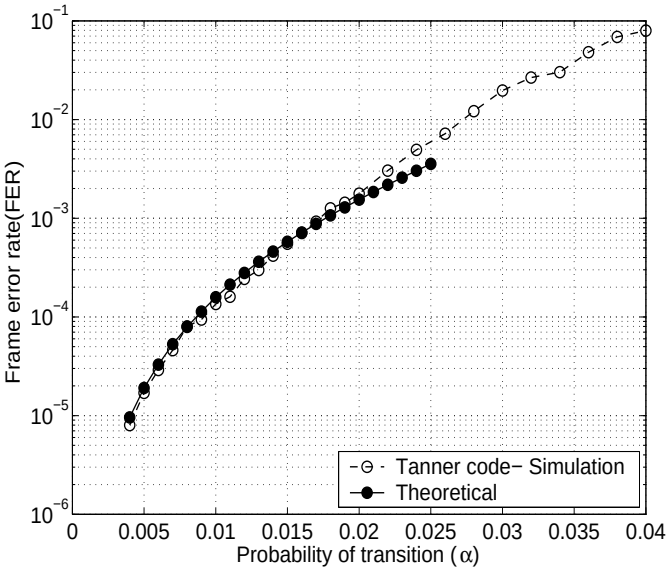


Figure 6.9: $(155, 93)$ Tanner code with $\gamma = 3$ and $g = 8$

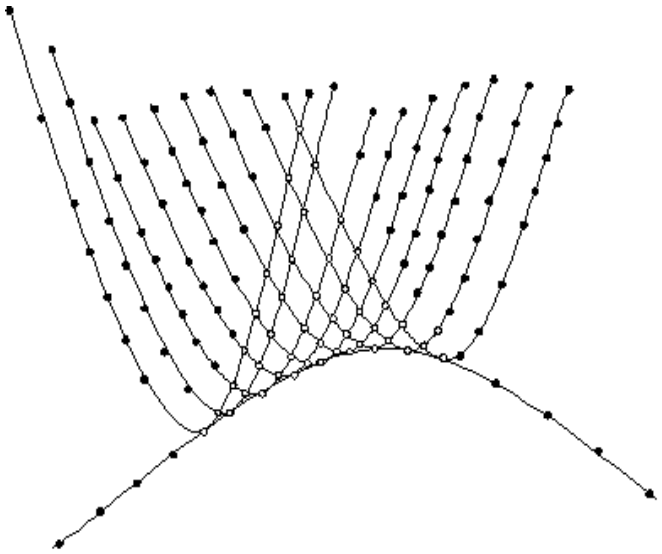


Figure 6.10: A minimal trapping set of PG$(2, 2^4)$

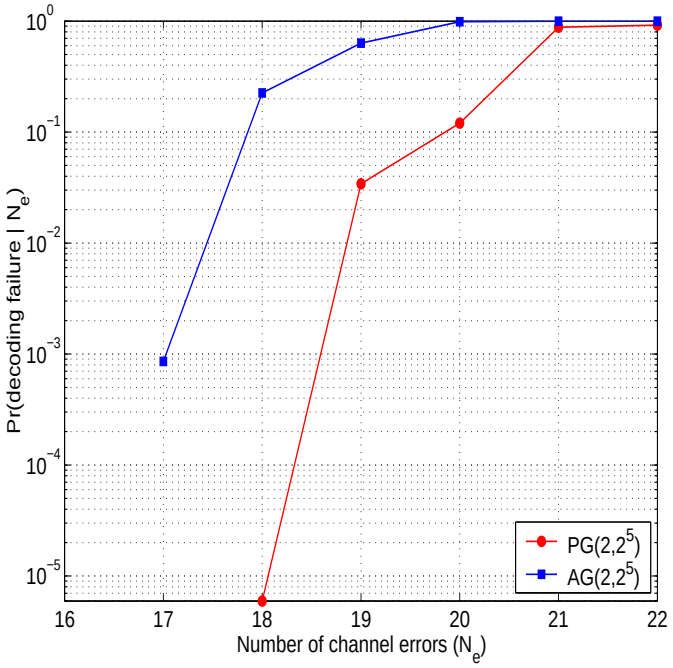
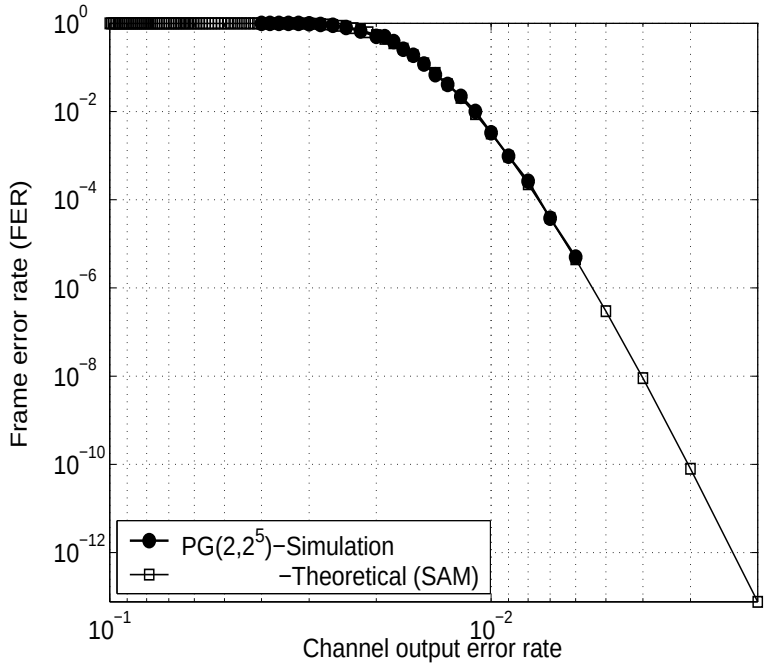
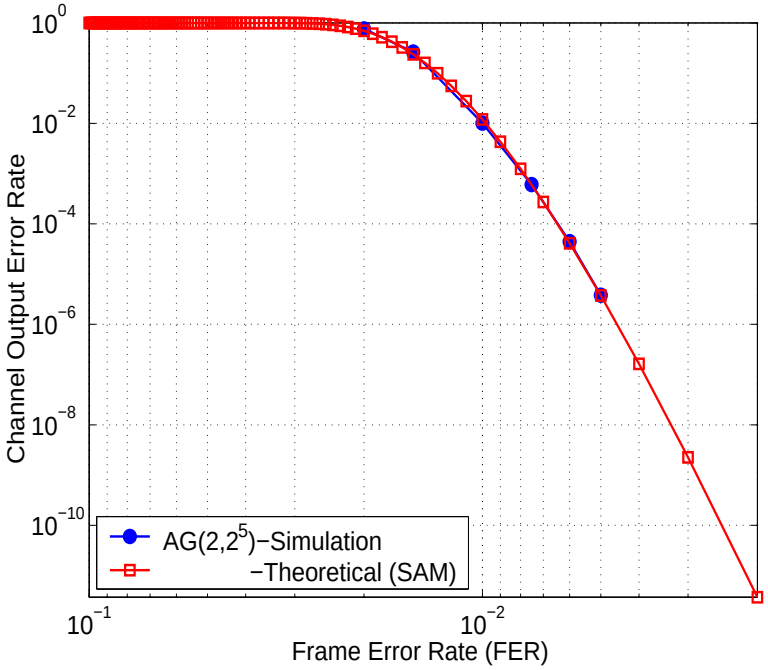


Figure 6.11: Probabilities of decoding failures given N_e channel errors



(a) Comparison between FER estimates and simulated FER for $\mathrm{PG}(2, 2^5)$



(b) Comparison between FER estimates and simulated FER for $\mathrm{AG}(2, 2^5)$

Figure 6.12: FER estimates of codes computed using SAM

CHAPTER 7

CONCLUSION

In Chapter 3, we proposed a technique to analytically compute the performance of product codes on bursty channels. Although the iterative decoding concept did not form a part of the communication system, a reference to a possible extension of the analytical method to apply for iterative hard-decision decoders is made towards the end of the chapter. But the concept of iterative decoding forms central to the investigative exercises and the ensuing results presented in Chapters 4-6. The performance results of systematically constructed regular and irregular LDPC codes are sufficiently encouraging to explore possibilities of applying these codes in future generations of communication systems. Especially, systematic regular low column-weight, like $3, 4$ or 5, LDPC codes hold great promise in achieving significant SNR gains in high-density magnetic storage systems [51]. The performance results of irregular and regular LDPC codes for perpendicular recording channels shown in Chapter 4 stand to support this notion. Similarly, papers [29] on coding for optical communication systems establish the significant gains achieved by regular LDPC codes. Lynch *et al.* [51] claim that constraints imposed by the system design in magnetic storage systems that impedes the application of LDPC codes can be overcome by the improvement in CMOS technology.

This does not mean the transfer of responsibilities to the hardware designers because there are several other questions, critical to the development of LDPC-based ECC systems, that remain unanswered.

1. In papers on LDPC codes for magnetic storage systems [49],[81], [99], [89] , it is common to observe that only codes with column weights less than 6 are studied.

Our studies reveal that the waterfall region of the performance curves is shifted to the right considerably for LDPC codes with high column weights. This shift was quite significant for LDPC codes from projective planes when decoded using the turbo equalization technique. Such a shift prohibits the use of high column-weight LDPC codes that, in general, have good minimum distance properties. The reasons for such significant shifts are not very well understood, and developing an understanding of this phenomenon can prove to be very useful.

2. It is widely agreed upon that decoding errors of practically "good" LDPC codes are rarely minimum distance errors. This means that it is common for the decoded words to have a small (relative to t) number of errors. In [51], the authors suggest the design of a post-processor to rid of these small number of errors. Such a post-processor may be achieved by exchanging bit likelihoods between two independent iterative decoders working on two equivalent parity-check matrices or TGs. The time overhead of such post-processor is expected to be small because it is activated only a very small fraction, especially at high SNR region, of the total number of decoding attempts.

3. The phenomenon of error floor in performance curves of LDPC codes is a cause of concern. Although the phenomenon itself is a subject of academic curiosity, the identification of the SNR region where LDPC codes exhibit the change in slope is critical to the development of LDPC-based ECC systems. It is critical because this knowledge is required to fixing the operating SNR of the communication system. It is judicious to operate in the error floor region because this guarantees a small change in sector error rate performance for a small change in SNR.

The Reed-Solomon codes are one of the most popular class of codes, and they are used for error correcting purposes in several applications including magnetic tape/disk

systems. The hard-decision decoders of RS codes lend to efficient hardware implementations but the same is not true for soft-decision RS decoders. The complexity of soft-decision list decoders [43], [54] is prohibitive. Hence, it would be ideal to develop less complex iterative soft decoding techniques suitable for RS codes. The results in Chapter 5 indicate limited success of the four-cycle free representations with regards to iterative decoding of RS codes. Further progress in this line of research depend on in-depth knowledge of RS codes and the iterative decoders. Since RS codes are probably one of the most very well studied class of codes, our focus should be on developing an understanding of iterative decoding of finite-length codes.

The attempt to achieve the dual goals of understanding iterative decoders and error floor phenomenon had resulted in Chapter 6. The classification of decoding failures using trapping sets and propagating sets has provided great insight into the patterns of failures. The influence of girth of a TG and small cycles in the TG is highlighted by the list of failures due to trapping sets. We realize that the success of estimating error floor of codes for the BSC cannot be extended easily for the AWGN channel. We believe that successful estimation of error floor by extending the proposed methods can be achieved by allowing incremental increase in the quantization levels of the LLRs.

APPENDIX A

Product Codes and GEC Models

A.1 Order of Row and Column Encoding is Irrelevant

Using the row-column encoding scheme, the (p, q)-th element, where $k_2 < p \leq n_2$ and $k_1 < q \leq n_1$, of $\mathbf{c}$ for a given $k_2 \times k_1$ array of information symbols can be written as

$$\mathbf{c}_{pq} = \langle \mathbf{m}_1^{(1)} \mathbf{G}^{(1)} \delta_q^{(1)}, \mathbf{m}_1^{(2)} \mathbf{G}^{(1)} \delta_q^{(1)}, \cdots, \mathbf{m}_1^{(k_2)} \mathbf{G}^{(1)} \delta_q^{(1)} \rangle \;\; \mathbf{G}^{(2)} \delta_p^{(2)}, \tag{A.1}$$

where $\mathbf{m}_i^{(1)}$ is the i-th information symbol vector that is encoded with $\mathbf{G}^{(1)}$, $\delta_q^{(1)}$ is a vector of length n_1 such that the q-th element is 1 and every other element is 0, and $\delta_p^{(2)}$ is a vector of length n_2 such that the p-th element is 1 and every other element is 0.

Let $\mathbf{m}_j^{(2)}$ be the j-th information symbol vector that is encoded with $\mathbf{G}^{(2)}$, and $m_{j,r}^{(2)}$ be the r-th element of this vector. Using this convention, $\mathbf{m}_i^{(1)}$ can be written in terms of column vectors as shown in (A.2).

$$\mathbf{m}_i^{(1)} = \langle m_{1,i}^{(2)}, m_{2,i}^{(2)}, \cdots, m_{k_1,i}^{(2)} \rangle \tag{A.2}$$

By substituting this expression in (A.1), we obtain the expression shown in (A.3) where the (i, j)-th element of a matrix is denoted by $\mathbf{G}_{i,j}$.

$$\begin{aligned}
\mathbf{c}_{pq} &= \left[\left(m_{1,1}^{(2)} \mathbf{G}_{1,p}^{(2)} + m_{1,2}^{(2)} \mathbf{G}_{2,p}^{(2)} + \cdots + m_{1,k_2}^{(2)} \mathbf{G}_{k_2,p}^{(2)} \right) \mathbf{G}_{1,q}^{(1)} \right] + \\
&\qquad \left[\left(m_{2,1}^{(2)} \mathbf{G}_{1,p}^{(2)} + m_{2,2}^{(2)} \mathbf{G}_{2,p}^{(2)} + \cdots + m_{2,k_2}^{(2)} \mathbf{G}_{k_2,p}^{(2)} \right) \mathbf{G}_{2,q}^{(1)} \right] + \\
&\qquad \cdots + \left[\left(m_{k_1,1}^{(2)} \mathbf{G}_{1,p}^{(2)} + m_{k_1,2}^{(2)} \mathbf{G}_{2,p}^{(2)} + \cdots + m_{k_1,k_2}^{(2)} \mathbf{G}_{k_2,p}^{(2)} \right) \mathbf{G}_{k_1,q}^{(k_1)} \right] \\
&= \left[\left(\mathbf{m}_1^{(2)} \mathbf{G}^{(2)} \delta_p^{(2)} \right) \mathbf{G}_{1,q}^{(1)} \right] + \left[\left(\mathbf{m}_2^{(2)} \mathbf{G}^{(2)} \delta_p^{(2)} \right) \mathbf{G}_{2,q}^{(1)} \right] + \cdots + \left[\left(\mathbf{m}_{k_1}^{(2)} \mathbf{G}^{(2)} \delta_p^{(2)} \right) \mathbf{G}_{k_1,q}^{(1)} \right] \\
&= \langle \mathbf{m}_1^{(2)} \mathbf{G}^{(2)} \delta_p^{(2)}, \mathbf{m}_2^{(2)} \mathbf{G}^{(2)} \delta_p^{(2)}, \cdots, \mathbf{m}_{k_1}^{(2)} \mathbf{G}^{(2)} \delta_p^{(2)} \rangle \;\; \mathbf{G}^{(1)} \delta_q^{(1)}
\end{aligned} \tag{A.3}$$

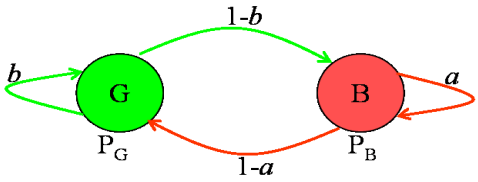


Figure A.1: Illustration of the GEC model.

The expression in (A.3) can be interpreted as encoding the column vectors with $\mathbf{G}^{(2)}$ and then encoding the resultant n_2 rows with $\mathbf{G}^{(1)}$. Hence, the column-row encoding scheme yields the same codeword as the row-column encoding process.

A.2 Computation of Error Transition Matrix

The GEC model with $\mathbf{P}^{(\mathrm{T})}$, $P_{\mathrm{G}} > 0$, and $P_{\mathrm{B}} > 0$ is illustrated in Fig. A.1. A variable D is introduced in $\mathbf{P}^{(\mathrm{T})}$ to track the error occurrences in each state.

$$\begin{aligned}
\mathbf{P}^{(\mathrm{T})}(D) &= \begin{pmatrix} b(1-P_{\mathrm{G}}) + bP_{\mathrm{G}}D & (1-b)(1-P_{\mathrm{B}}) + (1-b)P_{\mathrm{B}}D \\ (1-a)(1-P_{\mathrm{G}}) + (1-a)P_{\mathrm{G}}D & a(1-P_{\mathrm{B}}) + aP_{\mathrm{B}}D \end{pmatrix} \\
&= \begin{pmatrix} b(1-P_{\mathrm{G}}) & (1-b)(1-P_{\mathrm{B}}) \\ (1-a)(1-P_{\mathrm{G}}) & a(1-P_{\mathrm{B}}) \end{pmatrix} + D \begin{pmatrix} bP_{\mathrm{G}} & (1-b)P_{\mathrm{B}} \\ (1-a)P_{\mathrm{G}} & aP_{\mathrm{B}} \end{pmatrix}
\end{aligned}$$

Since there is a non-zero probability of making errors in each state, we obtain a new GEC model by splitting each state into two new states with subscripts 0 and 1, see Fig. A.2. A state with subscript 0 produces no errors, while a state with subscript 1 produces errors with unit probability. The state transition matrix $\tilde{\mathbf{P}}^{(\mathrm{T})}$ of the new GEC is written as:

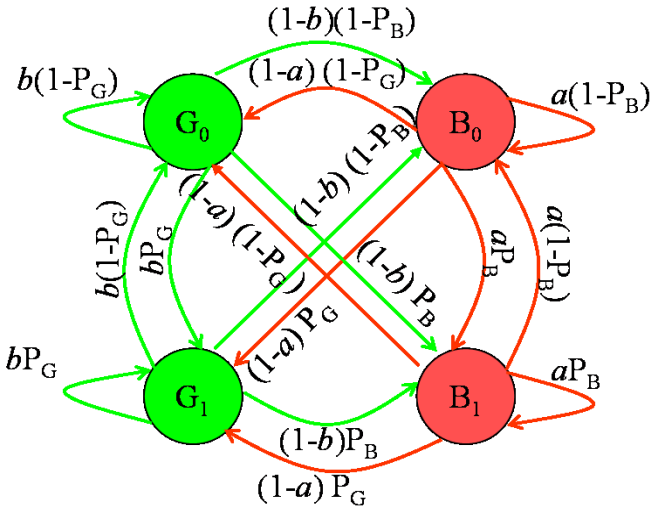


Figure A.2: Illustration of the split GEC model.

$$\tilde{\mathbf{P}}^{(\mathrm{T})} = \begin{pmatrix} \mathbf{P}^{(\mathrm{T})}(0) & \mathbf{P}^{(\mathrm{T})}(1) - \mathbf{P}^{(\mathrm{T})}(0) \\ \mathbf{P}^{(\mathrm{T})}(0) & \mathbf{P}^{(\mathrm{T})}(1) - \mathbf{P}^{(\mathrm{T})}(0) \end{pmatrix}$$

$$= \begin{pmatrix} b(1-P_{\mathrm{G}}) & (1-b)(1-P_{\mathrm{B}}) & bP_{\mathrm{G}} & (1-b)P_{\mathrm{B}} \\ (1-a)(1-P_{\mathrm{G}}) & a(1-P_{\mathrm{B}}) & (1-a)P_{\mathrm{G}} & aP_{\mathrm{B}} \\ b(1-P_{\mathrm{G}}) & (1-b)(1-P_{\mathrm{B}}) & bP_{\mathrm{G}} & (1-b)P_{\mathrm{B}} \\ (1-a)(1-P_{\mathrm{G}}) & a(1-P_{\mathrm{B}}) & (1-a)P_{\mathrm{G}} & aP_{\mathrm{B}} \end{pmatrix} \quad \text{(A.4)}$$

The matrix $\tilde{\mathbf{P}}^{(\mathrm{T})}$ of the new GEC model has two states that produce no errors, and two other states that produce errors with unit probability. If π_{G_0}, π_{B_0}, π_{G_1}, and π_{B_1} are the steady state probabilities, then the required error transition matrix $\mathbf{E}^{(\mathrm{T})}$ is obtained by merging *like*-states as shown in (A.5).

$$\mathbf{E}^{(\mathrm{T})} = \begin{pmatrix} \frac{\pi_{\mathrm{G}_0}}{\pi_{\mathrm{G}_0}+\pi_{\mathrm{B}_0}} & \frac{\pi_{\mathrm{B}_0}}{\pi_{\mathrm{G}_0}+\pi_{\mathrm{B}_0}} & 0 & 0 \\ 0 & 0 & \frac{\pi_{\mathrm{G}_1}}{\pi_{\mathrm{G}_1}+\pi_{\mathrm{B}_1}} & \frac{\pi_{\mathrm{B}_1}}{\pi_{\mathrm{G}_1}+\pi_{\mathrm{B}_1}} \end{pmatrix} \tilde{\mathbf{P}}^{(\mathrm{T})} \begin{pmatrix} 1 & 0 \\ 1 & 0 \\ 0 & 1 \\ 0 & 1 \end{pmatrix} \quad \text{(A.5)}$$

This method can be extended to compute error transition matrix for any N-state HMM.

A.3 Basic Properties of Kronecker Product

The *Kronecker product* of two matrices, $\mathbf{A}$ of dimensions $(\rho_{\mathbf{A}}, \gamma_{\mathbf{A}})$ and $\mathbf{B}$ of dimensions $(\rho_{\mathbf{B}}, \gamma_{\mathbf{B}})$, is a matrix $\mathbf{C}$ of dimensions $(\rho_{\mathbf{A}}\rho_{\mathbf{A}}, \gamma_{\mathbf{A}}\gamma_{\mathbf{B}})$ defined as $\mathbf{C} = \mathbf{A} \otimes \mathbf{B}$, where $\mathbf{C}_{i,j} = \mathbf{A}_{i,j}\mathbf{B}, \forall 1 \leq i \leq \rho_{\mathbf{A}}$ and $1 \leq j \leq \rho_{\mathbf{B}}$. The basic properties of Kronecker product are as follows:

1. Associativity: $\mathbf{A} \otimes (\mathbf{B} \otimes \mathbf{C}) = (\mathbf{A} \otimes \mathbf{B}) \otimes \mathbf{C}$.
2. Distributivity over addition: $(\mathbf{A} + \mathbf{B}) \otimes (\mathbf{C} + \mathbf{D}) = (\mathbf{A} \otimes \mathbf{C}) + (\mathbf{B} \otimes \mathbf{C}) + (\mathbf{A} \otimes \mathbf{D}) + (\mathbf{B} \otimes \mathbf{D})$.
3. Compatibility with multiplication: $(\mathbf{A} \times \mathbf{B}) \otimes (\mathbf{C} \times \mathbf{D}) = (\mathbf{A} \otimes \mathbf{C}) \times (\mathbf{B} \otimes \mathbf{D})$.
4. Compatibility with inversion: $(\mathbf{A} \otimes \mathbf{B})^{-1} = \mathbf{A}^{-1} \otimes \mathbf{B}^{-1}$.
5. Commutativity up to permutation: $\mathbf{A} \otimes \mathbf{B} = \mathbf{\Phi}\,(\mathbf{B} \otimes \mathbf{A})\,\mathbf{\Psi}$, where $\mathbf{\Phi}$ and $\mathbf{\Psi}$ are permutation matrices.

For proof of the properties and other details, refer to [39].

A.4 Proof of the Local State-Merging Technique

Theorem 15. *For any* $N > 1$, $\tilde{\mathbf{E}}^{(N,ch)} = \mathcal{F}\left(\mathbf{E}^{(T)} \otimes \tilde{\mathbf{E}}^{(N-1,ch)}\right)$.

Proof. The columns of $\mathbf{E}^{(N,\mathrm{ch})}$ are grouped based on the Hamming weights of the corresponding states, and in turn each group is divided into two subgroups based on the most significant bit (MSB) of the states. Such a grouping of columns is illustrated in (A.6), where HW(states) refers to the Hamming weight of states in that group. A similar arrangement is enforced on the rows of $\mathbf{E}^{(N,\mathrm{ch})}$. The states with *like* Hamming weights

in $\mathbf{E}^{(N,\mathrm{ch})}$ are merged together to form a single state in order to obtain $\tilde{\mathbf{E}}^{(N,\mathrm{ch})}$. This operation can be written as:

$$\mathbf{E}^{(N,\mathrm{ch})} = \begin{pmatrix} & \overbrace{\text{HW(states)=1 \& MSB is 0}}^{} & & & \overbrace{\text{HW(states)=1 \& MSB is 1}}^{} & & & & \\ \mathbf{E}^{(N,\mathrm{ch})}_{1,1} & \mathbf{E}^{(N,\mathrm{ch})}_{1,2} & \cdots & \mathbf{E}^{(N,\mathrm{ch})}_{1,\binom{N-1}{1}} & \mathbf{E}^{(N,\mathrm{ch})}_{1,\binom{N-1}{1}+1} & \cdots & \mathbf{E}^{(N,\mathrm{ch})}_{1,\binom{N}{1}+1} & \cdots & \mathbf{E}^{(N,\mathrm{ch})}_{1,2^N} \\ \vdots & & \ddots & & & \vdots & & \ddots & \vdots \\ \mathbf{E}^{(N,\mathrm{ch})}_{2^N,1} & & \cdots & & & \cdots & & \cdots & \mathbf{E}^{(N,\mathrm{ch})}_{2^N,2^N} \end{pmatrix} \tag{A.6}$$

$$\begin{aligned} \tilde{\mathbf{E}}^{(N,\mathrm{ch})} &= \mathcal{F}\left(\mathbf{E}^{(N,\mathrm{ch})}\right) \\ &= \begin{pmatrix} \frac{1}{\binom{N}{0}} & 0 & \cdots & 0 & 0 & \cdots & 0 & 0 & \cdots & 0 & 0 \\ 0 & \overbrace{\frac{1}{\binom{N}{1}}}^{\binom{N}{1}} & \cdots & \frac{1}{\binom{N}{1}} & 0 & \cdots & 0 & 0 & \cdots & 0 & 0 \\ 0 & 0 & \cdots & 0 & \overbrace{\frac{1}{\binom{N}{2}}}^{\binom{N}{2}} & \cdots & \frac{1}{\binom{N}{2}} & 0 & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots & \vdots & \cdots & \vdots & \vdots \\ 0 & 0 & \cdots & 0 & 0 & \cdots & 0 & 0 & \cdots & 0 & \frac{1}{\binom{N}{N}} \end{pmatrix} \mathbf{E}^{(N,\mathrm{ch})} \\ &\quad \begin{pmatrix} 1 & 0 & \cdots & 0 & 0 & \cdots & 0 & 0 & \cdots & 0 & 0 \\ 0 & \overbrace{1}^{\binom{N}{1}} & \cdots & 1 & 0 & \cdots & 0 & 0 & \cdots & 0 & 0 \\ 0 & 0 & \cdots & 0 & \overbrace{1}^{\binom{N}{2}} & \cdots & 1 & 0 & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots & \vdots & \cdots & \vdots & \vdots \\ 0 & 0 & \cdots & 0 & 0 & \cdots & 0 & 0 & \cdots & 0 & 1 \end{pmatrix}^{\mathrm{T}} \end{aligned} \tag{A.7}$$

$$= \mathbf{\Pi}_{\mathrm{R}}\, \mathbf{E}^{(N,\mathrm{ch})}\, \mathbf{\Pi}_{\mathrm{C}},$$

where $(.)^{\mathrm{T}}$ is the transpose operator. For example, in (A.8), there are $\binom{N}{1}$ states with a unit Hamming weight, and the columns corresponding to these states in $\mathbf{E}^{(N,\mathrm{ch})}$ are summed up by the second column of $\mathbf{\Pi}_{\mathrm{C}}$.

Both $\mathbf{\Pi}_{\mathrm{R}}$ and $\mathbf{\Pi}_{\mathrm{C}}$ can be written as a product of two matrices as shown in Eqs. (A.8) and (A.9). The scale-factors in rows of $\mathbf{\Pi}_{\mathrm{R}}$ correspond to the steady-state probabilities of states with *like* Hamming weights. In addition, computing these scale factors are greatly simplified by the fact that all channels are identical and independent. By substituting these expressions for $\mathbf{\Pi}_{\mathrm{R}}$ and $\mathbf{\Pi}_{\mathrm{C}}$ in (A.8), we prove the theorem statement as shown in (A.10).

$$\mathbf{\Pi}_{\mathrm{R}} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & \cdots & 0 \\ 0 & \frac{\binom{N-1}{1}}{\binom{N}{1}} & \frac{\binom{N-1}{0}}{\binom{N}{1}} & 0 & 0 & 0 & \cdots & 0 \\ 0 & 0 & 0 & \frac{\binom{N-1}{2}}{\binom{N}{2}} & \frac{\binom{N-1}{1}}{\binom{N}{2}} & 0 & \cdots & 0 \\ \vdots & \ddots & \vdots & \ddots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & 0 & 0 & \cdots & 1 \end{pmatrix}$$

$$\begin{pmatrix} \frac{1}{\binom{N}{0}} & \overbrace{0 \cdots 0}^{\binom{N-1}{1}} & 0 \cdots 0 & 0 \cdots 0 & 0 \cdots 0 \\ 0 & \underbrace{\frac{1}{\binom{N-1}{1}} \cdots \frac{1}{\binom{N-1}{1}}}_{\binom{N-1}{1}} & 0 \cdots 0 & 0 \cdots 0 & 0 \cdots 0 \\ 0 & 0 \cdots 0 & \overbrace{\frac{1}{\binom{N-1}{0}} \cdots \frac{1}{\binom{N-1}{0}}}^{\binom{N-1}{0}} & 0 \cdots 0 & 0 \cdots 0 \\ 0 & 0 \cdots 0 & 0 \cdots 0 & \overbrace{\frac{1}{\binom{N-1}{2}} \cdots \frac{1}{\binom{N-1}{2}}}^{\binom{N-1}{2}} & 0 \cdots 0 \\ \vdots & \vdots \ddots \vdots & \vdots \ddots \vdots & \vdots \ddots \vdots & \vdots \ddots \vdots \\ 0 & 0 \cdots 0 & 0 \cdots 0 & 0 \cdots 0 & 0 \cdots 0 \end{pmatrix} \begin{matrix} 0 \\ 0 \\ 0 \\ 0 \\ \vdots \\ \frac{1}{\binom{N}{N}} \end{matrix} \tag{A.8}$$

$$\mathbf{\Pi}_{\mathrm{C}} = \begin{pmatrix} 1 & 0 & \cdots & 0 & 0 & \cdots & 0 & 0 & \cdots & 0 & 0 & \cdots & 0 & 0 & 0 & 0 \\ & \overbrace{\quad}^{\binom{N-1}{1}} & & & & & & & & & & & & & & \\ 0 & 1 & \cdots & 1 & 0 & \cdots & 0 & 0 & \cdots & 0 & 0 & \cdots & 0 & 0 & 0 & 0 \\ & & & & \overbrace{\quad}^{\binom{N-1}{0}} & & & & & & & & & & & \\ 0 & 0 & \cdots & 0 & 1 & \cdots & 1 & 0 & \cdots & 0 & 0 & \cdots & 0 & 0 & 0 & 0 \\ & & & & & & & \overbrace{\quad}^{\binom{N-1}{2}} & & & & & & & & \\ 0 & 0 & \cdots & 0 & 0 & \cdots & 0 & 1 & \cdots & 1 & 0 & \cdots & 0 & 0 & 0 & 0 \\ & & & & & & & & & & \overbrace{\quad}^{\binom{N-1}{1}} & & & & & \\ 0 & 0 & \cdots & 0 & 0 & \cdots & 0 & 0 & \cdots & 0 & 1 & \cdots & 1 & 0 & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots & \vdots & \cdots & \vdots & \vdots \\ 0 & 0 & \cdots & 0 & 0 & \cdots & 0 & 0 & \cdots & 0 & 0 & \cdots & 0 & 0 & 0 & 1 \end{pmatrix}^{\mathrm{T}} \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & \cdots & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 & \cdots & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & \cdots & 0 \\ \vdots & \ddots & \vdots & \ddots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & 0 & 0 & \cdots & 1 \end{pmatrix}^{\mathrm{T}} \tag{A.9}$$

$$
\begin{aligned}
\tilde{\mathbf{E}}^{(N,\mathrm{ch})} &= \mathbf{\Pi}_{\mathrm{R}}\ \mathbf{E}^{(N,\mathrm{ch})}\ \mathbf{\Pi}_{\mathrm{C}} \\
&= \begin{pmatrix}
1 & 0 & 0 & 0 & 0 & 0 & \cdots & 0 \\
0 & \frac{\binom{N-1}{1}}{\binom{N}{1}} & \frac{\binom{N-1}{0}}{\binom{N}{1}} & 0 & 0 & 0 & \cdots & 0 \\
0 & 0 & 0 & \frac{\binom{N-1}{2}}{\binom{N}{2}} & \frac{\binom{N-1}{1}}{\binom{N}{2}} & 0 & \cdots & 0 \\
\vdots & \ddots & \vdots & \ddots & \vdots & \ddots & \vdots & \vdots \\
0 & 0 & 0 & 0 & 0 & 0 & \cdots & 1
\end{pmatrix} \tilde{\mathbf{E}}^{(N-1,\mathrm{ch})} \\
&\quad \begin{pmatrix}
1 & 0 & 0 & 0 & 0 & 0 & \cdots & 0 \\
0 & 1 & 1 & 0 & 0 & 0 & \cdots & 0 \\
0 & 0 & 0 & 1 & 1 & 0 & \cdots & 0 \\
\vdots & \ddots & \vdots & \ddots & \vdots & \ddots & \vdots & \vdots \\
0 & 0 & 0 & 0 & 0 & 0 & \cdots & 1
\end{pmatrix}^{\mathrm{T}} \\
&= \mathcal{F}\left(\mathbf{E}^{(\mathrm{T})} \otimes \tilde{\mathbf{E}}^{(N-1,\mathrm{ch})}\right). \qquad \text{(A.10)}
\end{aligned}
$$

□

REFERENCES

[1] N. Alon, R. Yuster, and U. Zwick,"Color-coding: a new method for finding simple paths, cycles, and other small subgraph within large graph," in *Proceedings of the* 26^{th} *Annual ACM Symposium on Theory of Computing*, Montréal, Canada, pp. 326-335, 1994.

[2] N. Alon, R. Yuster, and U. Zwick, "Finding and counting given length cycles," in *Proceedings of the 2nd European Symposium on Algorithms*, Utrecht, The Netherlands, pp. 354-364, 1994.

[3] Ian Anderson, "Combinatorial Designs : Construction Methods", *Ellis Horwood Limited*, Chichester, West Sussex, PO19 1EB, England, 1990.

[4] E. F. Assmus Jr., J. D. Key, *Design and Their Codes*. Cambridge, U.K.: Cambridge Univ. Press, 1992.

[5] E. F. Assmus Jr., J. D. Key, "Designs and their codes: An update," *Designs, Codes, Cryptography*, vol. 9, pp. 7-27, 1996.

[6] E. F. Assmus, Jr. and J. D. Key, "Baer subplanes, ovals, and unitals," in *Coding Theory and Design Theory, Part I*, vol. 20, *The IMA Volumes in Mathematics and Its Applications*, D. Ray-Chaudhuri, Ed. New York: Springler-Verlag, 1990, pp. 1-8.

[7] L.R. Bahl, J. Cocke, F. Jelinek, and J. Raviv, "Optimal Decoding of Linear Codes for Minimizing Symbol Error Rate, " *IEEE Trans. on Information Theory*, Vol. 20, March 1974, pp. 284-287.

[8] Jan W. M. Bergmans, "Digital Baseband Transmission and Recording, " *Kluwer Academic Publishers*, 1996.

[9] T. Beth, D. Jungnickel, and H. Lenz, *Design Theory*. Cambridge, U.K.: Cambridge Univ. Press, 1986.

[10] I. F. Blake, and R. C. Mullin, *The Mathematical Theory of Coding*, New York: Academic Press, 1975.

[11] M. Blaum, "An introduction to error-correcting codes," in *CRC Handbook for Coding and Signal Processing for Recording Systems* (B. Vasić, ed.) CRC Press, Sept. 2004.

[12] R. C. Bose, and D. K. Ray-Chaudhuri, "On a class of error correcting binary group codes," *Information and Control*, vol. 3, pp. 68-79, 1960.

[13] A. R. Calderbank, G. D. Forney, and A. Vardy, "Minimal tail-biting trellises: The Golay code and more," *IEEE Trans. Inform. Theory*, vol. 45, pp. 1435-1455, July 1999.

[14] S. Cavin, "Error-correction coding for tape drive systems," M.S. Thesis, Swiss Federal Institute of Technology, EPFL, Lausanne, Switzerland, 2004.

[15] V. Chandru, M. R. Rao, "Integer Programming," *Handbook of Algorithms*, Edited by M. J. Atallah, CRC Press, 1998.

[16] Jung-Fu Cheng, "Iterative Decoding," *Doctoral Dissertation*, California Institute of Technology, Pasadena, California, March 1997.

[17] V. Chernyak, M. Chertkov, M. Stepanov, and B. Vasić, "Error correction on a tree: An Instanton approach," *Physical Review Letters*, vol. 93, pp. 198702-1-198702-4, Nov. 2004.

[18] S. K. Chilappagari, S. Sankaranarayanan, B. Vasic, "Error Floors of LDPC Codes on Binary Symmetric Channel, accepted for presentation at *IEEE International Conference on Communications*, June, 2006.

[19] R. D. Cideciyan, E. Eleftheriou, and T. Mittelholzer, "Perpendicular and longitudinal recording: a signal-processing and coding perspective," *IEEE Trans. Magn.*, vol. 38, pp. 1698-1704, Jul. 2002.

[20] C. J. Colbourn and J. H. Dinitz, Eds., *The Handbook of Combinatorial Designs*. Boca Raton, FL: CRC Press, 1996.

[21] C. J. Colbourn and A. Rosa, *Steiner Systems*. London, U.K.: Oxford Univ. Press, 1999, Oxford Mathematical Monographs.

[22] Hubert Comon, Mehmet Dincbas, Jean-Pierre Jouannaud and Claude Kirchner, " A Methodological View of Constraint Solving", *Constraints*, Vol. 4 Issue. 4, 1997, pp. 337-361, also available at *http:// www.lri.fr/ jouannau/biblio.html*.

[23] Evelyne Contejean and Hervé Devie, "An efficient incremental algorithm for solving systems of linear diophantine equation," *Information and Computation*, vol. 113, issue 1, pp. 143-172, August 1994.

[24] C. Di, D. Proietti, T. Richardson, E. Telatar, and R. Urbanke, "Finite length analysis of low-density parity-check codes," *IEEE Trans. Inform. Theory*, vol. 48, pp. 1570-1579, June 2002.

[25] J. Colbourn, and J. H. Dinitz, "The handbook of combinatorial designs," Boca, Raton, FL: CRC Press, 1996.

[26] I. B. Djordjević, S. Sankaranarayanan, and B. Vasić, "Projective plane iteratively decodable block codes for WDM high-speed long-haul transmission sys-

tems," *IEEE/OSA Journal of Lightwave Technology*, vol. 22, pp. 695-702, March 2004.

[27] I. B. Djordjević, S. Sankaranarayanan, and B. Vasić, "Irregular low-density parity-check codes for long haul optical communications," *IEEE Photonics Technology Letters*, vol. 16, no. 1, pp. 338-340, Jan. 2004.

[28] I. B. Djordjević, B. Vasić, and S. Sankaranarayanan, "Regular and irregular low-density parity-check codes for ultra long-haul high-speed optical communication: Construction and performance analysis," in *Proc. Optical Fiber Communication Conference 2004*, vol. 1, Feb. 2004, pp. 23-27.

[29] I. B. Djordjevic, S. Sankaranarayanan, S. K. Chilappagari, and B. Vasić, "Low-density parity-check codes for 40 Gb/s optical transmission systems," submitted for review to *IEEE Selected Topics on Quantum Electronics*.

[30] I. B. Djordjević and B. Vasić, "High code rate low-density parity-check codes for optical communication systems," *IEEE Photonics Technology Letters*, vol. 16, pp. 1600-1602, June 2004.

[31] I. B. Djordjević and B. Vasić, "Projective geometry low-density parity-check codes for ultra-long haul WDM high-speed transmission," *IEEE Photonics Technology Letters*, vol. 15, no. 5, pp. 784-786, May 2003.

[32] Eric Domenjoud, "Solving Systems of Linear Diophantine Equations: An Algebraic Approach," *Mathematical Foundations of Computer Science*, 1991, pp. 141-150.

[33] "Data interchange on 12.7 mm 384-track magnetic tape cartridges," *ECMA*-319 Standard, 2001.

[34] E. O. Elliot, "Estimates of error-rates for codes on burst-noise channels," *Bell Syst. Tech J.*, vol. 42, pp.1977-1997, 1963.

[35] M. F. Erden, E. M. Kurtas, and M. J. Link, "Evaluation of thermal asperity in magnetic recording," in *CRC Handbook for Coding and Signal Processing for Recording Systems* (B. Vasić, ed.) CRC Press, Sept. 2004.

[36] T. Etzion, A. Trachtenberg, and A. Vardy, "Which codes have cycle-free Tanner graphs?," *IEEE Trans. Inform. Theory*, vol. 45, pp. 2173-2181, Sept. 1999.

[37] R. G. Gallager, Low-Density Parity-Check Codes. Cambridge, MA: MIT Press, 1963.

[38] E. N. Gilbert, "Capacity of burst-noise channel," *Bell Syst. Tech J.*, vol. 39, pp.1253-1265, 1960.

[39] A. Graham, *Kronecker products and matrix calculus with applications*. New York: John Wiley and Sons, 1981.

[40] B. J. Frey, R. Koetter, and A. Vardy, "Signal-space characterization of iterative decoding, *IEEE Trans. on Inform. Theory*, vol. IT47, no. 2, pp. 766781, 2001.

[41] G. D. Forney, "Codes on graphs: Normal realizations," *IEEE Trans. Inform. Theory*, vol. 47, pp. 520-548, Feb. 2001.

[42] G. D. Forney, Jr., R. Koetter, F. Kschischang, and A. Reznik, "On the effective weights of pseudocodewords for codes defined on graphs with cycles," vol. 123 of*Codes, systems, and graphical models, IMA Vol. Math. Appl.,* ch. 5, pp. 101-112, Springer, 2001.

[43] V. Guruswami and M. Sudan, Improved decoding of ReedSolomon and algebraic-geometric codes, *IEEE Trans. Inform. Theory*, vol. 45, pp. 17571767, Sept. 1999.

[44] Joachim Hagenauer, "Iterative Decoding of Binary Block and Convolutional Code, " *IEEE Trans. on Information Theory*, Vol. 42, No. 2, March 1996, pp. 429-445.

[45] Thomas R. Halford and Keith M. Chugg. "An algorithm for counting short cycles in bipartite graphs". submitted to *IEEE Trans. on Inform. Theory*, [Online]. Available: *http://www-scf.usc.edu/ halford*.

[46] N. Hamada, "On the p-rank of the incidence matrix of a balanced or partially balanced incomplete block design and its applications to error-correcting codes," *Hiroshima Math. J.* , vol. 3, pp. 153-226, 1973.

[47] R.W. Hamming, "Error detecting and error correction codes," *Bell Syst. Tech. J.*, vol. 39, pp. 147-160, 1950.

[48] A. Hocquenghem, "Codes correcteurs d'erreurs," *Chiffres (Paris)*, vol. 2, pp. 147-156, 1959.

[49] J. Li, K. R. Narayanan, E.Kurtas, and C. N. Georghiades, On the performance of the high-rate TPC/SPC and LDPC codes over partial response channels, *IEEE Trans. Commun.*, vol. 50, pp. 723735, May 2002.

[50] M. Luby, M. Mitzenmacher, A. Shokrollahi, and D. Spielman, "Analysis of Low-Density Codes and Improved Designs Using Irregular Graphs," *Proc. 30th Annu. ACM Symp. Theory of Computing*, 1998, pp.249-258.

[51] R. Lynch, E. M. Kurtas, A. Kuznetsov, E. Yeo, and B. Nikolić, "The search for a practical iterative detector for magnetic recording," *IEEE Trans. Magn.*, vo. 40, no. 1, pp. 213-218, Jan. 2004.

[52] A. Kavčić, X. Ma, and M. Mitzenmacher, "Binary intersymbol interference channels: Gallager codes, density evolution, and code performance bounds," *IEEE Trans. Inform. Theory,*, submitted for publication.

[53] C. Kelley, D. Sridhara, J. Xu, and J. Rosenthal, "Pseudocodeword weights and stopping sets," in *Proc. IEEE Int. Symp. Information Theory*, 2004, p. 68.

[54] R. Koetter and A. Vardy, Algebraic soft-decision decoding of Reed-Solomon codes, *IEEE Trans. Inform. Theory,* vol. 49, pp. 28092825, Nov. 2003.

[55] Y. Kou, S. Lin, M. P. C. Fossorier, "Low-density parity-check codes based on finite geometries: A rediscovery and new results," *IEEE Trans. Inform. Theory*, vol. 47, no. 7, pp. 2711-2736, Nov. 2001.

[56] F. R. Kschischang, " Factor graphs and the sum-product algorithm," *IEEE Trans. Inform. Theory*, vol. 47, no.2, Feb. 2001.

[57] Shu Lin, and Daniel J. Costello, JR.,"Error Control Coding: Fundamentals and Applications," *Prentice-Hall, Inc.*, Englewood Cliffs, NJ, 1983.

[58] D. J. C. MacKay, "Relationships between sparse graph codes," *Proc. of Information-Based Induction Sciences*, Japan, 2000. Available online at *http:// www.inference.phy.cam.ac.uk/mackay/abstracts/ibis.html*.

[59] D. J. C. MacKay, "Good error-correcting codes based on very sparse matrices," *IEEE Trans. Inform. Theory*, vol. 45, pp. 399-431, March 1999.

[60] D. J. C. MacKay, "Encyclopedia of sparse graph code." [Online]. Available: *http://www.interference.phy.cam.ac.uk/mackay/codes/data.html*.

[61] D. MacKay and M. Davey, "Evaluation of Gallager codes for short block length and high rate applications," [Online]. Available: *http://www.cs.toronto.edu/ mackay/CodesRegular.html*.

[62] M. A. Margulis, "Explicit group-theoretic constructions for combinatorial designs with applications to expanders and concentrators," *Probl. Pered. Inform.*, vol. 24, no. 1, pp. 51-60, 1988.

[63] R. J. McEliece, "On the BCJR trellis for linear block codes," *IEEE Trans. Inform. Theory*, vol. 42, pp. 1072-1092, July 1996.

[64] O. Milenkovic, I. B. Djordjević, and B. Vasić, "Block-circulant low-density parity-check codes for optical communication systems," *IEEE/LEOS Journal of Selected Topics in Quantum Electronics*, vol. 10, no. 2, pp. 294-299, March/April 2004.

[65] J. Moon, "Modeling the recording channel," in *CRC Handbook for Coding and Signal Processing for Recording Systems* (B. Vasić, ed.) CRC Press, Sept. 2004.

[66] T. Oenning, and J. Moon, "Modeling the Lorentzian magnetic recording channels with transition noise," *IEEE Trans. Mag.*, vo. 37, pp.583-591, 2001.

[67] J. Pearl, *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference.* San Mateo, CA: Morgan Kaufmann, 1988.

[68] D. MacKay and M. Postol, "Weaknesses of margulis and ramanujan-margulis low-density parity-check codes, *Electronic Notes in Theoretical Computer Science*, vol. 74, 2003.

[69] J. G. Proakis, *Digital Communications*, McGraw-Hill, New York, 1995.

[70] J. G. Proakis, "Partial response equalization with application to high density," in *CRC Handbook for Coding and Signal Processing for Recording Systems* (B. Vasić, ed.) CRC Press, Sept. 2004.

[71] R. Pyndiah, "Near-optimum decoding of product codes: Block turbo codes," *IEEE Tran. Commun.*, vol. 46, pp. 1003-1010, Aug. 1998.

[72] I. S. Reed, and G. Solomon, "Polynomial codes over certain finite fields," *J. SIAM*, vol. 8, pp. 300-304, 1960.

[73] T. J. Richardson, "Error floors of LDPC codes," in emphProc. 41st *Annual Allerton Conf. on Communications, Control and Computing*, 2003.

[74] T. Richardson and R. Urbanke, "The Capacity of Low-Density Parity-Check Codes Under Message-Passing Decoding," *IEEE Trans. on Information Theory*, Vol. 47, pp. 599-618, Feb. 2001.

[75] T. Richardson, A. Shokrollahi, and R. Urbanke, "Design of capacity-approaching irregular low-density parity-check codes," *IEEE Trans. Inform. Theory*, vol. 47, pp. 619-637, Feb. 2001.

[76] J. Rosenthal, and P. O. Vontobel, "Construction of LDPC codes using Ramanujam graphs and ideas from Margulis," in *Proc.* 2001 *IEEE Int. Symp. Information Theory*, Washington, June 2001, p. 4.

[77] L. D. Rudolph, "A class of majority logic decodable codes," *IEEE Trans. Inform. Theory*, vol. IT-13, pp. 305-307., Apr. 1967.

[78] W. E. Ryan, "An introduction to LDPC codes," in *CRC Handbook for Coding and Signal Processing for Recording Systems* (B. Vasić, ed.) CRC Press, Sept. 2004.

[79] William E. Ryan, " A Turbo Code Tutorial," [Online]. Available: *http://www.ece.arizona.edu/ ˜ryan*

[80] S. Sankaranarayanan, I. B. Djordjević, and B. Vasić, "Iteratively decodable codes on m-flats for WDM high-speed long-haul transmission," *J. Lightw. Technol.*, vol. 23, no. 11, pp. 3696-3701, Nov. 2005.

[81] S. Sankaranarayanan, B. Vasić, and E. M. Kurtas, "Irregular low-density parity-check codes: construction and performance on perpendicular magnetic recording channels," *IEEE Trans. Magn.*, vol. 39, no. 5, pp. 2567-2569, Sept. 2003.

[82] S. Sankaranarayanan, B. Vasic, and E. M. Kurtas, "A systematic construction of irregular low-density parity-check codes from combinatorial designs," in *IEEE Intern. Symp. Inform. Theory*, June 29- July 4, 2003. p. 1.

[83] S. Sankaranarayanan, and B. Vasić, "Analysis of iterative erasure-decoding of linear block codes: A parity-check orthogonalization approach," in *Allerton Conference on Communication, Control, and Computing*, October 2004.

[84] S. Sankaranarayanan, and E. Eleftheriou,"Performance of product codes on channels with memory," in *IEEE Intern. Inform. Theory*, 4-9 Sept., 2005.

[85] S. Sankaranarayanan, and B. Vasić, "Analysis of iterative erasure-decoding of linear block codes: A parity-check orthogonalization approach," in *IEEE Trans. Inform Theory*, Sept. 2005.

[86] C. E. Shannon, "A mathematical theory of communication," *Bell Syst. Tech. J.*, pp. 372-423, 1948.

[87] A. Shokrollahi, "LDPC Codes: An introduction." [Online]. Available: *http://www.ipm.ac.ir/IPM/homepage/Amin2.pdf.*

[88] M. Sipser, and D. Spielman, "Expander codes," *IEEE Trans. Inform. Theory*, vol. 42, pp. 1710-1722, 1996.

[89] H. Song, B.V.K. Vijaya Kumar, E. Kurtas, Y. Yuan, L.L. McPheters, and S.W. McLaughlin, Iterative decoding for partial response (PR) equalized magneto-optical (MO) data storage channels, *IEEE J Select. Areas. Commun.*, vol. 19, no. 4, pp. 774883, 2001.

[90] M. Sudan, "Coding theory: Tutorial and survey," [Online]. Available: *http://theory.lcs.mit.edu/ madhu/papers*

[91] H. Tang, J. Xu, S. Lin, and K. Abdel-Ghaffar, "Codes on finite geometries," *IEEE Trans. Inform. Theory*, vol. 51, pp. 572-596, Feb. 2005.

[92] R. M. Tanner, "A recursive approach to low complexity codes," *IEEE Trans. Inform. Theory*, vol. IT-27, pp. 533-547, Sept. 1981.

[93] R. M. Tanner, "Minimum-distance bounds by graph analysis," *IEEE Trans. Inform. Theory*, vol. 47, pp. 808-821, Feb. 2001.

[94] R. M. Tanner, D. Sridhara, and T. Fuja, "A class of group-structured LDPC codes," *ISCTA 2001*, Ambleside, England, 2001.

[95] J. A. Thas, *Handbook of Incidence Geometry: Buildings and Foundations.* F. Buekenhout, Ed. Amsterdam, The Netherlands: Elsevier Science, 1996, ch. 7.

[96] J. H. van Lint, "Introduction to coding theory," Graduate Texts in Mathematics 86, 3rd Edition, Springer-Verlag, Berlin, 1999.

[97] A. Vardy, and Y. Be'ery, "Bit-level soft-decision decoding of Reed-Solomon codes," *IEEE Trans. Commn.*, vol. 39, no. 3, pp. 440-444, March 1991.

[98] B. Vasić, "Structured iteratively decodable codes based on Steiner systems and their application in magnetic recording," in *Proc. GLOBECOM 2001*, vol. 5, San Antonio, TX, Nov. 26-29, 2001, pp. 2954-2960.

[99] B. Vasic, E. Kurtas, and A. Kuznetsov, LDPC codes based on mutually orthogonal Latin rectangles and their application in perpendicular magnetic recording, *IEEE Trans. Magn.*, pt. 1, vol. 38, pp. 23462348, Sept. 2002.

[100] B. Vasic, A. Cvetković, S. Sankaranarayanan, and M. Marcellin, "A systematic construction of irregular low-density parity-check codes from combinatorial designs," in *IEEE Intern. Symp. Inform. Theory*, June 29- July 4, 2003, p. 267.

[101] B. Vasić, E. Kurtas, and A. Kuznetsov, "Lattice low-density parity-check codes and their application in partial response channels," in *Proc. 2002 IEEE Int. Symp. Information Theory*, Lausanne, Switzerland, June/July 2002, p. 453.

[102] B. Vasić and O. Milenković, "Combinatorial constructions of low-density parity-check codes for iterative decoding," *IEEE Trans. Inform. Theory*, vol. 50, no. 6, pp. 1156-1176, June 2004.

[103] P. Vontobel, and R. Koetter, "Graph-cover decoding and finite-length analysis of message-passing iterative decoding of LDPC codes," submitted to *IEEE Trans. Inform. Theory*, 2005.

[104] Andrew J. Viterbi, " An Intuitive Justification and a Simplified Implementation of the MAP Decoder for Convolutional Codes, " *IEEE Journal on Selected Areas in Communications*, Vol. 16, No. 2, February 1998.

[105] S. R. Weller, S. J. Johnson, "Iterative decoding of codes from oval designs," *European Trans. on Telecommunications*, vol. 14, no. 5, pp. 399-409, October 2003.

[106] N. Wiberg, H.-A. Loeliger, and R. Kötter, "Codes and iterative decoding on general graphs," *Euro. Trans. Telecomm.*, vol. 6, pp. 513-525, Sept./Oct. 1995.

[107] N. Wiberg, "Codes and decoding on general graphs," Ph.D. dissertation, Univ. Linköping, Linköping, Sweden, 1996.

[108] J. K. Wolf, "ECC performance of interleaved RS codes with burst errors," *IEEE Trans. Magn.*, vol. 34, pp. 75-79, Jan. 1998.

[109] Xiao-Yu Hu, Evangelos Eleftheriou, Dieter-Michael Arnold, and Ajay Dholakia, "Efficient Implementation of the Sum-Product Algorithm for Decoding LDPC Codes, " *IEEE Trans. on Information Theory*, Vol. 20, March 1974, pp. 284-287.

[110] J. Xu, and S. Lin, "A combinatoric superposition method for constructing low-density parity-check codes," *Proc. 2003 IEEE Int. Symp. Inform. Theory*, Yokohama, Japan, June/July 2003, p. 30.

[111] X. Yang, and E. M. Kurtas, "Signal and noise generation for magnetic recording channel simulations," in *CRC Handbook for Coding and Signal Processing for Recording Systems* (B. Vasić, ed.) CRC Press, Sept. 2004.

[112] J. S. Yedidia, J. Chen, and M. Fossorier,"Generating code representations suitable for belief propagation decoding," *Proc. 40th Annual Allerton Conf. on Communications, Control, and Computing*, Oct. 2002.