



**EFFECTS OF DEGREE DISTRIBUTION IN RATELESS CODING**

**OMAR RAAD KADHIM KADHIM**

**SEPTEMBER 2014**

**EFFECTS OF DEGREE DISTRIBUTION IN RATELESS CODING**

**A THESIS SUBMITTED TO  
THE GRADUATE SCHOOL OF NATURAL AND APPLIED  
SCIENCES OF  
ÇANKAYA UNIVERSITY**

**BY  
OMAR RAAD KADHIM KADHIM**

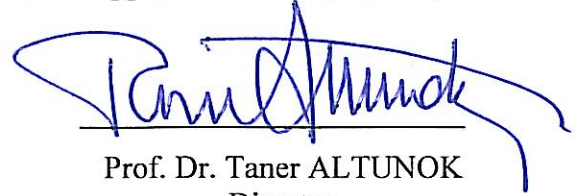
**IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE  
DEGREE OF  
MASTER OF SCIENCE  
IN  
THE DEPARTMENT OF  
ELECTRONIC AND COMMUNICATION ENGINEERING**

**SEPTEMBER 2014**

Title of Thesis: **Effects of Degree Distribution in Rateless Coding**

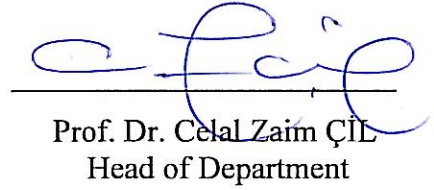
Submitted by **Omar Raad Kadhim KADHIM**

Approval of the Graduate School of Natural and Applied Sciences, Çankaya University.



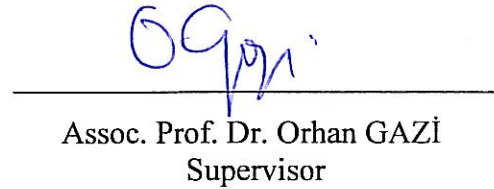
Prof. Dr. Taner ALTUNOK  
Director

I certify that this thesis satisfies all the requirements as a thesis for the degree of Master of Science.



Prof. Dr. Celal Zaim ÇİL  
Head of Department




This is to certify that we have read this thesis and that in our opinion it is fully adequate, in scope and quality, as a thesis for the degree of Master of Science.



Assoc. Prof. Dr. Orhan GAZI  
Supervisor

**Examination Date: 22.09.2014**

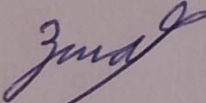
**Examining Committee Members**

Assoc. Prof. Dr. Orhan GAZI	(Çankaya Univ.)	
Assoc. Prof. Dr. Fahd JARAD	(UTAA Univ.)	
Assist. Prof. Dr. Göker ŞENER	(Çankaya Univ.)	

## STATEMENT OF NON-PLAGIARISM PAGE

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all materials and results that are not original to this work.

Name, Last Name: Omar Raad KADHIM

Signature : 

Date : 22.09.2014

## **ABSTRACT**

### **EFFECTS OF DEGREE DISTRIBUTION IN RATELESS CODING**

KADHIM, Omar Raad Kadhim

M.Sc., Department of Electronic and Communication Engineering

Supervisor: Assoc. Prof. Dr. Orhan GAZI

September 2014, 49 pages

In this thesis rateless codes which are adopted by a variety of applications such as, wireless transmission, 3GPP, data storage, multicasting, video streaming are inspected in details. The performance of two important types of rateless codes which are Luby Transform and Raptor codes are measured via computer simulations. Both hard decision and soft decision methods are used while measuring the performance of these codes. For the soft decision decoding Belief Propagation algorithm was used in an iterative manner. Degree distribution is an important criteria for the performance of Luby Transform codes. A new degree distribution called random degree (or exponential random) distribution is proposed for Luby Transform codes. And simulation results support that the proposed distribution shows better performance than the classical degree distributions such as all-at-once, ideal soliton, robust soliton, and sparse.

**Keywords:** Rateless Coding, Luby Transform, Belief Propagation, Degree Distribution

## ÖZ

### DERECE DAĞILIMININ ORANSIZ KODLAR ÜZERİNDEKİ ETKİLERİ

KADHİM, Ömer Raad Kadhim

Yüksek Lisans, Elektronik ve Haberleşme Mühendisliği Anabilim Dalı

Tez Yöneticisi: Doç. Dr. Orhan GAZİ

Eylül 2014, 49 sayfa

Bu tezde kablosuz iletişim, 3GPP, veri depolama, video iletişim gibi günümüz iletişim teknolojilerinin birçoğunda kullanılan oransız kodlar detaylı olarak incelenmiştir. İki önemli oransız kod türü, Luby dönüşüm ve Raptor kodlarının performansları bilgisayar benzetimi yoluyla ölçülmüştür. Benzetim esnasında hem katı hem de yumuşak çözümleme algoritmaları kullanılmıştır. Yumuşak çözümleme için karar yayılımı algoritması kullanılmıştır. Derece dağılımı Luby dönüşüm kodlarının performansları önemli ölçüde etkilemektedir. Rasgele derece dağılımı adında yeni bir derece dağılımı önerilmiş ve performansı benzetim yoluyla ölçülmüştür. Elde edilen sonuçlar önerdiğimiz derece dağılımının hali hazırda literatürde var olanlara göre (ideal soliton, robust soliton, ve sparse) daha iyi performans gösterdiğini desteklemektedir.

**Anahtar Kelimeler:** Oransız Kodlar, Luby Dönüşümü, Karar Yayılımı, Derece Dağılımı

## **ACKNOWLEDGEMENTS**

I am sincerely indebted and obliged to Assoc. Prof. Dr. Orhan GAZI for his support, special attention, surveillance and suggestions, as well as the encouragement he showed me during the writing of this thesis. I am certain it would have not been possible without his help.

I would also like to express my gratitude to my father and mother and wife for their valuable and special support.

## TABLE OF CONTENTS

STATEMENT OF NON PLAGIARISM.....	iii
ABSTRACT.....	iv
ÖZ.....	v
ACKNOWLEDGEMENTS.....	vi
TABLE OF CONTENTS.....	vii
LIST OF FIGURES.....	x
LIST OF ABBREVIATIONS.....	xi

### CHAPTERS:

<b>1. INTRODUCTION.....</b>	<b>1</b>
<b>1.1. Coding Theory: Classical vs. Modern Approach.....</b>	<b>1</b>
<b>1.2. Fundamental of Channel Coding.....</b>	<b>3</b>
<b>1.3. Linear Codes.....</b>	<b>4</b>
<b>1.4. Belief Propagation Algorithm (BP).....</b>	<b>5</b>
<b>1.5. Decoding Graph.....</b>	<b>6</b>
<b>1.6. Thesis Outline.....</b>	<b>6</b>
<b>2. RATELESS CODES.....</b>	<b>8</b>
<b>2.1. Rateless Encoding Decoding.....</b>	<b>8</b>
<b>2.2. Tanner Graph.....</b>	<b>9</b>
<b>2.3. Low Density Equality-Check Code (LDPC).....</b>	<b>11</b>
<b>2.3.1. LDPC encoding.....</b>	<b>12</b>
<b>2.3.1.1. Lower triangular modification approach.....</b>	<b>12</b>
<b>2.3.2. LDPC soft decoding using belief propagation.....</b>	<b>14</b>
<b>2.4. Luby Transformer Code (LT).....</b>	<b>18</b>
<b>2.4.1. LT encoder.....</b>	<b>19</b>
<b>2.4.2. Degree distributions for LT code.....</b>	<b>22</b>

vii



2.4.2.1.	All-at-once distribution.....	22
2.4.2.2.	Ideal soliton distribution.....	23
2.4.2.3.	Robust soliton distribution.....	23
2.4.2.4.	Sparse distribution.....	24
2.4.3.	Decoding of Luby transform (LT) code.....	24
2.4.3.1.	Hard decoding algorithm.....	24
2.4.3.1.1.	Serial decoding.....	25
2.4.3.1.2.	Parallel decoding.....	25
2.4.3.2.	Soft decoding algorithm.....	26
2.5.	Raptor Codes.....	28
2.5.1.	Raptor encoding.....	29
2.5.2.	Raptor decoding using belief propagation.....	30
2.5.2.1.	Local-iteration decoder.....	30
2.5.2.2.	Global-iteration decoder.....	33
3.	NEW DEGREE DISTRIBUTION AND MODELING RESULTS.....	36
3.1.	New Degree Distribution Implementation.....	36
3.1.1.	Exponential distribution.....	36
3.1.2.	Random degree distribution.....	37
3.1.3.	Gaussian distribution.....	39
3.2.	Modeling Results.....	40
3.2.1.	Low density equality check code (LDPC).....	40
3.2.2.	Raptor codes.....	41
3.2.2.1.	Local-iteration decoder.....	41
3.2.2.2.	Global-iteration decoder.....	42
3.3.	LT Code Modeling Result.....	42
3.3.1.	LT code with robust soliton distribution.....	43
3.3.2.	LT code with sparse degree distribution.....	43
3.3.3.	LT code with exponential degree.....	43
3.3.4.	LT code with Gaussian degree distribution.....	44

<b>3.3.5.</b> Comparative between normal and new degree .....	45
<b>4.</b> CONCLUSION AND FUTURE WORKS.....	48
<b>4.1.</b> Conclusion.....	48
<b>4.2.</b> Future Works.....	49
REFERENCES.....	R1
APPENDICES.....	A1
A. CURRICULUM VITAE.....	A1

## LIST OF FIGURES

### FIGURES

<b>Figure 1</b>	Equality check knot 1 linked up with adjustable knots.....	10
<b>Figure 2</b>	Tanner graph of the H matrix of the Hamming code.....	10
<b>Figure 3</b>	Equality-check matrix in estimated lower triangular shape.....	13
<b>Figure 4</b>	The LT encoding/decoding process.....	20
<b>Figure 5</b>	Parallel decoding of LT decoder.....	26
<b>Figure 6</b>	Raptor encoding diagram.....	29
<b>Figure 7</b>	Raptor local-iteration decoder configuration.....	31
<b>Figure 8</b>	Raptor global-iteration decoder configuration.....	34
<b>Figure 9</b>	Exponential degree distribution.....	37
<b>Figure 10</b>	Random degree distribution.....	38
<b>Figure 11</b>	Gaussian distribution.....	39
<b>Figure 12</b>	LDPC code performance.....	40
<b>Figure 13</b>	Raptor code performance using local-iteration decoder.....	41
<b>Figure 14</b>	Global-iteration decoder.....	42
<b>Figure 15</b>	LT with exponential degree.....	44
<b>Figure 16</b>	LT code with Gaussian degree distribution.....	45
<b>Figure 17</b>	LT code with random degree and robust soliton distribution.....	46
<b>Figure 18</b>	LT code with random degree and spars degree distribution.....	46

## LIST OF ABBREVIATIONS

Cap	Channel Capacity
3G	3rd Generation
IEEE	Institute of Electrical and Electronics Engineers
LDPC	Low-Density Equality-Check
i.i.d	Independent Identical Distribution
BIMS	Binary Input Memoryless Symmetrical
BP	Belief Propagation
NP	Nondeterministic Polynomial
LT	Luby Transform
Wi-Fi	Wireless Fidelity
Wi-MAX	Worldwide Interoperability for Microwave Access
DVB	Digital Video Broadcasting
DVBS2	Digital Video Broadcasting Second Generation
3GPP	3rd Generation Partnership Project
LLR	Log-Likelihood Ratio
BPSK	Binary Phase Shift Keying
AWGNC	Additive White Gaussian Noise Channel
BER	Bit Error Rate
SNR	Signal to Noise Ratio

## CHAPTER 1

### INTRODUCTION

#### 1.1. Coding Theory: Classical vs. Modern Approach

In 1948, an article about the mathematical theory of communication [1] was published by Shannon that presented the essentials of communication boundaries. The article explained the basic issues of transmission that regenerate at a point where either exact or estimated information was chosen at another point. A fundamental solution of this issue is somewhat instinctive and the chosen information can be encoded with redundant information. In such a case, if delivered information becomes corrupted due to noise, the redundant information will be sufficient to recover the genuine information. Here, two serious questions regarding designer's code rose: the first is what type of redundancy will be feasible, and the second is in order to identify how much redundancy will be used. Both questions are essential in terms of theory as well as practice. With the assessment of the extra quantity required with regard to constantly regenerating the actual information on the receiver side, it is very necessary to know what the optimal use of these communication resources is. Therefore, all coding schemes are allocated defined numbers (this process is called information degree) which work in a genuine way to determine which parts of the delivered data are beneficial. In contrast, the descriptive question actually defines the coding scheme that is not only utilized for communication resources, but also sufficiently capable to encode and decode efficiently. Therefore, the code designer's main function is to classify the coding structures with the highest possible data percentage that must contain: (a) a reduced likelihood of interpreting error, and (b) effectual encoding and decoding procedures.

He responded to the above questions by defining a certain parameter for information rate to ensure dependable communication even with the existence of noise in the channel. His results explained that there is a definite value  $\text{Cap} (C) \in [0; 1]$  called the channel capability such as if only  $R < \text{Cap} (C)$  then reliable communication is possible. With the existence of these values, a trustworthy coding structure of data rate ensures  $R$ , such as the coding structure with very little likelihood of arbitrary error. However, Shannon's response was non-productive and based on likelihood. In response, another question was asked as to which coding schemes will be efficient enough to bring us near to that channel. For many years, coding theory has seen some unexpected improvements that include: deriving observations from several areas of arithmetic, engineering, and solving both theoretical and practical approaches. However, attaining an effective coding scheme is still not easy. According to Shannon, an arbitrary lined coding scheme is a methodology for channel capacity; however, this does not contain a sufficient number of characteristics to form an effective decoding path. This distinction among coding theories is best described by Reiffen and Wozencraft [2]:

*Any code that we can't think of is better.*

This pointed towards the leading approach that has proved this claim in just a few attempts [3]. Before the early 90s, the response to the descriptive question with the introduction of Turbo codes has been revealed [4, 5]. Their originality was aligned within the utilization of the virtual arbitrary interleavers in encoding process and within a sensibly planned repetitive decoding process. Turbo codes found sufficient "arbitrariness" to the closest capacity method very similarly to arbitrary interleavers; however, it was capable of reserving adequate structure to permit effective encoding and decoding procedures. In reality, turbo codes that were introduced in the preliminary conference publication were banned by the arbitrators [6]. Nevertheless, today Turbo codes have become an essential part of day-to-day technology and make our lives easier as they can be found on 3G mobiles, and various communications standards such as IEEE 802.16 metropolitan wireless network standards [7]. After the turbo codes that delivered the preliminary motion to the model of data coding correction, the LDPC (low density equality check) coding approach was introduced by Gallager in 1963 [8], but soon forgotten afterwards and later rediscovered by

MacKay, Neal [9 10, 11], Wiberg [12, 13], Sipser and Spielman [14, 15]. LDPC codes were presented to have outstanding performance in comparison to large Turbo codes. Later on, LDPC codes and their repetitive decoding approach were extensively accepted and analyzed. The official knowledge of this new method resulted in the entire arena of modern coding theory [16] in contrast to the traditional coding theory that specifically handles the arithmetical production of codes. As an outcome, decoding methods and applied codes are very well known nowadays because they are amazingly close to the channel capacity [17], having outstanding low mathematical issues and actively responding to severe calculated assessment [18].

## 1.2. Fundamental of Channel Coding

The main purpose in channel coding is to deliver reliable data. For example, the arrangement of  $k$  symbols  $x = (x_1; x_2; \dots; x_k) \in X^k$ , that are components of a prearranged character  $X$ , diagonally is a channel with noise. In this regard, an encoding leads to the  $x$  arrangements to the code word  $y = (y_1; y_2; \dots; y_n) \in Y^n$  that later delivers and reduces by a noisy channel. The decoding process detects a sequence of disorganized values, such as a conventional character  $z = (z_1; z_2; \dots; z_n) \in Z^n$  assessed  $y$  based on  $z$ . Vectors  $x$ ,  $y$  and  $z$  acknowledge the arbitrary variables,  $X$  on  $X^k$ ,  $Y$  on  $Y^n$ ,  $Z$  on  $Z^n$ , correspondingly. Subsequently, every  $x_i$ ,  $y_i$  and  $z_i$  is an awareness of scalar arbitrary variables  $X_i$ ,  $Y_i$  and  $Z_i$ , one-to-one. Moreover, we usually consider that each  $X_i$ ,  $Y_i$  and  $Z_i$  is self-governing and equally dispersed (i.i.d.) in a likelihood concept consistent operation  $P_X(x)$ ,  $P_Y(y)$  and  $P_Z(z)$ , correspondingly. The association between  $Y$  and  $Z$  characters is demonstrated by a likelihood operation  $P_{Z|Y}(z|y)$ , which is conditional.

In this study, the greater concern was with binary-input, low memory and symmetry (BIMS channels). A binary codeword  $Y$  is the basic symbol of these channels that showed either  $F2 = \{0; 1\}$  or set  $\{-1; +1\}$ . Each time the codeword value character  $\{-1; +1\}$  is utilized, plotting  $0 \rightarrow +1$ ,  $1 \rightarrow -1$  is implied in this study. Furthermore, BIMS channels have no memory: the outcome of such channels is immediate and

only depends on its input data. Additionally, the symmetry situation infers that the outcome of the channel is symmetric to inserting data. This situation is too complex to describe when  $Y = F_2$ . However, if model  $Y = \{-1; +1\}$  and  $Z \subseteq \mathbb{R}$ , then the regularity situation is simplified:

$$P_{Z|Y}(z|1) = P_{Z|Y}(-z| -1), \forall z \in Z. \quad (1.1)$$

In the scenario of a low capacity channel  $C$ , the highest value of each data symbol that delivers codeword  $Y$  from codeword  $Z$  is known as the channel capacity:

$$\text{Cap}(C) = I(Y, Z) \quad (1.2)$$

Here,  $I(Y;Z)$  signifies the joint data between the  $Y$  and  $Z$  variables, and conveyed data in bits. Shannon demonstrated that at all code rates  $R < \text{Cap}(C)$ , consistent broadcast is possible.

### 1.3. Linear Codes

Binary linear codes are the most demanded channel codes having data values as the alphabet and the code word values, in which the alphabet is limited to  $F_2$ . A dual lined coding structure can be seen as a lined plotting from the group of information  $F_2^k$  to the group of code words  $C \subseteq F_2^n$ ; here  $C$  formulates a  $k$ -dimensional route of  $F_2^n$ . Characteristically, this dimensional route is called code, because it fetches the related structure of the coding scheme. This is denoted as  $(n; k)$  binary linear code, where  $n$  is the size and  $k$  is the dimension of the code, and its code rate  $R$  is distinct as  $k/n$ , unless stated, all vectors are activated.

Linear codes are defined with a foundation set  $\{g_1; g_2; \dots; g_k\}$ , where  $g_i \in F_2^n$ , that indicate the producer matrix illustration of a lined code. Specifically, an  $n \times k$  matrix  $G$  is known as the producer matrix of code  $C$  if

$$c \in C \Leftrightarrow \exists x \in F_2^k : Gx = c \quad (1.5)$$

It is to be noticed that every matrix that formulates a foundation of  $C$  with columns is actually a producing matrix of  $C$  that offers a simple measurement of plotting data over the code words. Eventually,  $C$  can be specified ultimately with subspace  $C^\perp$  within  $F_2^n$  and its base  $\{h_1; h_2; \dots; h_{n-k}\}$ . Given below, the  $C$  subspace is shown as



$$C^\perp = \{c' \in F_2^n : c'c = 0 \forall c \in C\} \quad (1.6)$$

Though it is possible to formulate the equality check matrix demonstration of a lined code, the  $A(n-k) \times n$  matrix  $H$  would be the equality check matrix of  $C$  if

$$c \in C \Leftrightarrow Hc = 0$$

Evidently, a matrix formed by rows based on  $C^\perp$  is called the equality check matrix of  $C$ . The dual subspace  $C^\perp$  of a linear  $(n, k)$  code  $C$  is an additional lined code with distance  $n$  and measurement  $n - k$ , of  $C$ . The transferred equivalence check matrix of  $C$  is the producer matrix of  $C^\perp$  and the converse also holds true

In fountain codes, transactions being performed by the coding system without fix percentage a priori. Every row of the producer matrix produces rapidly and considered as an arbitrary variable  $F_2^k$ , whereas  $k$  is the measurement symbol. Thus, instantly  $j \in N$ , the origin encoder produces a solo encrypted symbol  $y_j = v_j$  and  $x \in F_2^k$ , where  $v_j$  is an arbitrarily selected row vector from  $F_2^k$ . In such an arrangement, the receiver detects the quantity of conventional word symbols  $z_{i1}; z_{i2}; \dots; z_{in}$  consistent with the transferred symbols  $y_{i1}; y_{i2}; \dots; y_{in}$ . This shows that the derived code  $(n; k)$  is binary lined code that is defined by a producer matrix with  $V_{i1}; v_{i2}; \dots; V_{iN}$  vectors. If decoding of such code fails, the additional encrypted symbols can be gathered by a receiver that provides a better distance.

#### 1.4. Belief Propagation Algorithm (BP)

The entire problem of linear encoding for BIMS channels is known as NP accomplishment [19]. Having an effective and practical method for the encoding of BIMS channels is not easy. Therefore, concretely forming a suboptimal repetitive approach for a very small subclass of decrypting issues is problematic. In a similar way, other approaches to decode lined codes are more complex because they are held with customization. Due to the large numbers of code, decoding is no longer easy. However, if we categorize this methodology into local functions, for example functions described by a group of small divisions all variables, we will obtain a starting point to develop new approaches. This complete factorization is typically

envisioned with a graph known as a factor graph, which actually illustrates an association between variables and functions and indicates which variable influences which function. These graphs are simple and are general forms of Tanner graphs [20] that defines LDPC codes as well. It could be said that a factor graph is a kind of graphical model and Bayesian inference [21] can be performed over it easily. However, the concept behind this is predictable. An ethical broadcast approach merely abuses the factorization of the universal function to competently calculate. This theoretical level is similar to the distributive law calculations [19]. The common toy example is about the use of the distributive law in the efficient calculation of the function variables  $f(a, b, c) = AB + AC$ , whereas it is visibly capable of calculating the factored version of the function  $f(a; b; c) = a(b + c)$  (one addition and one multiplication) than its un-factored version (two multiplications and one addition).

### **1.5. Decoding Graph**

The most traditional and hands-on performance of linear codes could be the Matrices, while Graphs have been demonstrated effective to prove and explore the presentation of broadcast decoding with linear codes. Graphic codes include fountain codes, turbo codes and LDPC codes. Their decoding matrices are typically signified by Tanner graphs suitable for the low density of the matrices. Graphic codes are mostly low density, which is why they are called sparse-graph codes. The bareness of the decoding graph confirms the low calculation difficulty. Tanner [20] revived LDPC codes by recommending Tanner graphs which are capable of recursive.

### **1.6. Thesis Outline**

This thesis contains four chapters and these chapters are explained as follows:

Chapter-1 begins with a general introduction about coding theory. Furthermore, this chapter presents an introduction to linear code and a brief description of belief propagation. A decoding graph is also presented in this chapter.

Chapter-2 contains an introduction to rateless code, and Tanner graphs are explained in more detail. Furthermore, Low Density Equality Check codes are mentioned with specific information. Luby Transform encoding and decoding and normal degree distribution are also demonstrated in this Chapter. Finally, Raptor code with two types of decoder is explained.

Chapter-3 is divided into two sections: the first section discusses new degree distribution design and the second section deals with simulation results for all codes and degree distribution and comparisons made between these degrees.

Chapter-4 consists of a conclusion and a list of future works.

## CHAPTER 2

### RATELESS CODES

#### 2.1. Rateless Encoding Decoding

In 1998, Byers et al. [22] revealed the “digital fountain method” inspired by authentic data delivery to various self-directed customers. Given below is an innovative project of broadcast and multicast protocols. A receiver can re-establish the source information with encoded symbols arbitrarily composed of a loss channel. The channel’s lossy design can even be unidentified and data access starts on arbitrary time. While a Tornado code is preferable in order to meet this fundamental concept, it should already assess the channel loss rate and essentially allocate a fixed code rate  $R$ , which is near to the loss rate, and in return it bounds the quantity of user knots that is unevenly equal to  $1/R$  [23] [24]. Consequently, the features of digital fountains must be: to reliably deliver data in a single transmission, have no response, have effective encoding and decoding, have data that should be accessible when necessary, and a loss rate that is sufficiently acceptable. At this time, the immense image of a theoretical digital fountain over a memoryless removal channel. A demanded file is split into  $k$  blocks of similar size and each block is encapsulated into a packet. Given these  $k$  packets, it is possible for the fountain encoder to create a boundless stream of self-governing and similarly-scattered encoding packets. These encoded packets are conveyed over the deletion channel, but only a fraction of the packets is received error free with the remainder either lost or discarded. The fountain decoder then, with high likelihood, improves the source packets from any subset of established encoded packets of a number equal to, or higher than,  $k$ . This procedure is usually used as a cup arbitrarily to gather some drops from a fountain

and once having accumulated a sufficient number of drops in the cup, thirst will be satisfied.

Fountain codes, also known as rateless code, even though at both the sender and receiver knots the code rate is unfixed and can possibly reach zero. Conversely, in practice, only one should be asked to reflect any shortened fountain codes. As per the requirements of the application, such as the decipher completion rate and calculate price, there would be a linked up least number of arrived packets to comply with conditions. For example, this least number is  $n$ , and the decoder improves the data at a rate of  $k/n$  shortened fountain code. For a perfect or optimum fountain code, the code rate needs to be 1. When  $k$  is somewhat smaller than  $n$ , the fountain code is sub-optimum or nearby optimum. Now we have the two major classes of applied fountain codes: Luby Transform (LT) codes and Raptor codes. Both are closer to optimum for limited data size.

## 2.2. Tanner Graph

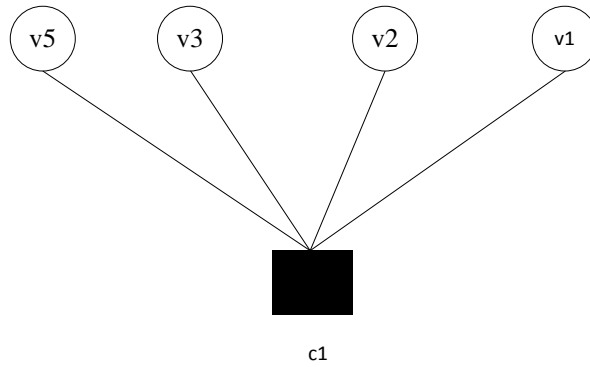
A Tanner graph or dynamic graph [25] is a divided graph which is related to the equality check matrix  $H$  ( $N-K \times N$ ) or designer matrix  $G$  ( $K \times N$ ) where  $N$  and  $K$  correspondingly accomplish the code word size and information length. The divided graph contains two kinds of knots, the first being equal to  $N$  and achieving the data bits. Additionally, it is also called a variable knot. The other knot is equal to  $N-K$  and shows the equality check bits [26]. Moreover, it is acquainted with check knots. Adjustable knots can link up only to see whether that variable is a particular check knot. Similar knots cannot associate with each other easily, thereby showing that there are no limits among identical knots.

### Tanner Graph (Factor) example

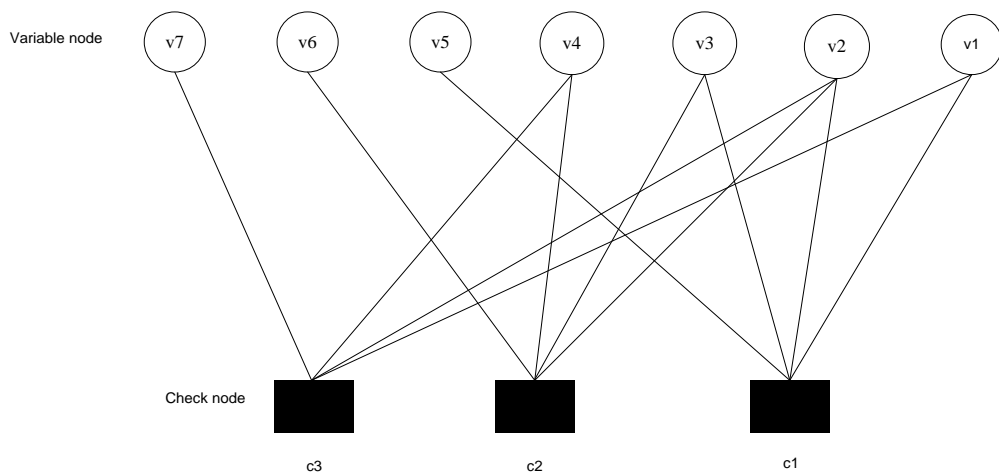
Here, the Hamming (7, 4, 3) code with equality check matrix  $H$  ( $3 \times 7$ ) and producer matrix  $G$  ( $4 \times 7$ ) is mentioned to prove [27] how a Tanner graph can be constructed. Hamming (7,4,3) equality check matrix  $H$  code can be define as:

$$H = \begin{pmatrix} 1 & 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 1 \end{pmatrix} \quad (2.1)$$

By detecting equality check matrix above, H has found three check knots,  $c_m$   $m=1, \dots, 3$  and seven adjustable knots  $v_l$ ,  $l=1,2,3,\dots,7$ . This means each row acts as a check knot and each column shows an adjustable knot. The first row of H satisfies the equality check equation  $g_1 \oplus g_2 \oplus g_3 \oplus g_5 = c_1$  meaning that the variable knots  $\{1, 2, 3, 5\}$  attach straight to a check knot via the boundaries as shown in Fig. 1. This process recurs every time for each row. Finally, the Tanner graph is visible in Fig. 2. Similarly, the columns of H act as incidence vectors in which equality check equations contribute to the variable knot. On the other hand, the column of H specifies that an adjustable knot is linked to the check knot  $\{1,2,3\}$



**Figure 1:** Equality check knot 1 linked up with adjustable knots



**Figure 2:** Tanner graph of the H matrix of the Hamming code

The G (4\*7) of pretense (7, 4 ,3) equal to:

$$G = \begin{pmatrix} 1 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 1 \end{pmatrix} \quad (2.2)$$

### 2.3. Low Density Equality-Check Code (LDPC)

Low density equality-check code (LDPC) is basically error linear code, mostly utilizing a noisy transmission network to decrease the likelihood of waste of data. This likelihood can be decreased with LDPC maximall , hence the data transferred percentage can be as close as desired to Shannon’s perimeter. LDPC was originally altered by Robert Gallager in his PhD thesis at MIT in 1960 [28] by MIT Press 3 years after LDPC was published. Because of the restrictions on the mathematical potential in examining the encoder and decoder for the given codes and the outline of Reed-Solomon codes, LDPC was neglected for almost 30 years. R. Michael Tanner defined LDPC code in 1981, as the Tanner graph [25]. Since turbo codes had become inventive in 1993, the main emphasis was transformed to find low multifaceted code which is closer to the Shannon channel size. LDPC was rediscovered by Mackay [29], [11] and Luby [30]. Meanwhile, LDPC has also established its identity in a few advanced applications such as 10GBase-T Ethernet, Wi-Fi, WI-MAX, and Digital Video Broadcasting (DVB, DVBS2).

All lined code has both an equality check matrix and a mutual graph, but not all lined code is capable of highlighting a clear image. If the 1 values in any row in an (n× m) matrix where weight denoted as wr, and when the 1 values are in any column, the column weight is denoted as wc, which is far lower than the measurement (wr << m, wc << n). Code characterized by a sparse equality-check matrix is known as low density equality check code (LDPC). The sparse characteristics of LDPC encourage its mathematical benefits.

### 2.3.1. LDPC encoding

If the originator matrix  $G$  of error altering code is recognized, then the encoding can be completed by reproducing the originator matrix  $G$  with a group of data. As we can say that the encoded vector ( $c$ ) is equal to the following:

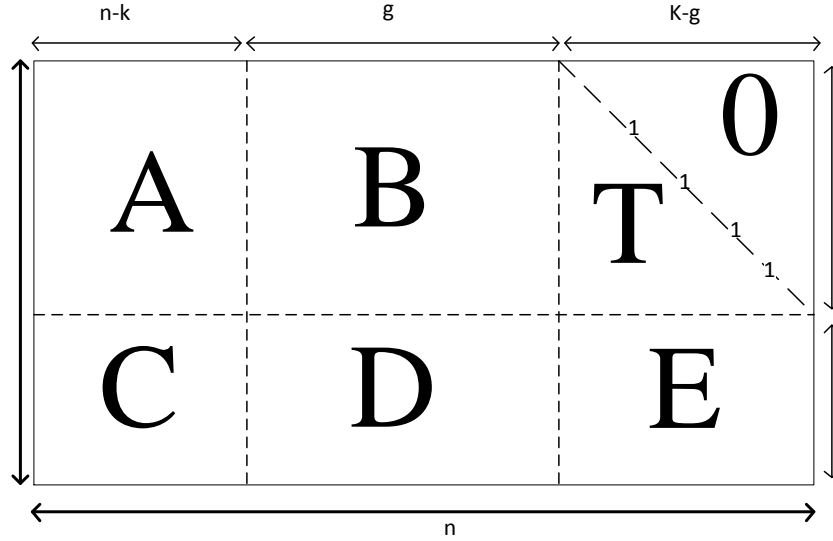
$$c = G * m \quad (2.3)$$

$m$  here is the data vector, the number of processes of this technique relies on the number of 1's (Hamming weights) of the base vectors of  $G$ . If the vectors are solid, the number of processes (cost) of encoding using this process is comparable to  $n^2$ . This cost becomes linear with  $n$  if  $G$  is sparse. Conversely, by the null space of a sparse equality-check matrix  $H$ , the LDPC is specified. It is doubtful that the originator matrix  $G$  will be sparse. For that reason, the unambiguous tactic of encoding an LDPC needs number of processes, comparative (cost) to  $n^2$ . This is too sluggish for most practical applications. After that, it is suitable to have encoding approaches that run on lined time.

#### 2.3.1.1. Lower triangular modification approach

Urbanke & Richardson presented an encoding process that possessed proficiently lined performance time for some codes with a sparse equality-check matrix. This method consists of stages: pre-processing and encoding. In the pre-processing stage,  $H$  is deployed into the shape as given in Fig. 3 by column and row variations.





**Figure 3:** Equality-check matrix in estimated lower triangular shape

Here T has an inferior three-sided shape with all transverse entrances is equal to 1. Subsequently the process is complete by column and row variables and if H is sparse, then A, B, C, D, E, T are also sparse.

Let the codeword  $s = (m, p1, p2)$  whereas m is the data bits, p1 and p2 are the parity bits, p1 has length g, p2 has length  $k - g$ .

Calculating p1 and p2 as fallow in:

1. Compute top syndrome of m by using Eq. (2.4).

$$v_r = Am^T \quad (2.4)$$

2. Calculate the series of parity bits,  $k_2^A$ , which assume the top syndrome to 0.

$$k_2^A = T^{-1}v_r$$

3. Determine the bottom syndrome of the vector  $[m \ 0 \ k_2^A]$

$$v_b = Cm^T + Ek_2^A \quad (2.5)$$

4. Find the parameter Z from Eq. (2.6).

$$Z = -ET^{-1}B + D \quad (2.6)$$

After that, the first equality fraction is calculated as in Eq.(2.7).

$$p_1 = Z^{-1}v_b \quad (2.7)$$

5. Compute the new top syndrome in Eq. (2.8)

$$v_x = v_r + Bp_1 \quad (2.8)$$

6. In the final, the else sets of equality bits  $k_2^A$ , can be gained in Eq. (2.9).

$$p_2 = v_x T^{-1} \quad (2.9)$$

By using back- subtraction,  $k_2^A$  and  $p_2$  in step 2 & 6 can be calculated in linear time. Selecting the gap (g) as tiny as possible because the complexity of Eq. (2.9) in step six is  $O(k^2)$ . Furthermore, the complexity of computing D at Eq. (2.6) is  $O(k^3)$  and, this is computed for one time just previous any message block encoding.

### 2.3.2. LDPC soft decoding using belief propagation

The flexible chosen decoder performed to prepare the data that have a provisional probability that the incoming bit is either a 1 or a 0 by given vector y. By letting  $P_i = P_r [c_i = 1 | y]$  be the provisional likelihood that  $c_i$  is a 1 given the value of y, we have

$$P_r [c_i = 0 | y] = 1 - P_i \quad (2.10)$$

By letting  $q_{ij}^l$  be the data sent by data knot  $c_i$  to check knot  $f_j$  at round l, then every datum contains a pair  $q_{ij}^l(0)$  and  $q_{ij}^l(1)$  which stands for the “amount of belief” that  $y_i$  is 0 or 1.

$$q_{ij}^l(0) + q_{ij}^l(1) = 1 \quad (2.11)$$

Specifically,  $q_{ij}^l(1) = P_i$  and  $q_{ij}^l(0) = 1 - P_i$

Likewise, we let  $r_{ij}^l$  be the data transfer by check knot  $f_j$  to data knot  $c_i$  in round l. Each datum comprises a pair  $r_{ij}^l(0)$  and  $r_{ij}^l(1)$  which stands for the “quantity of confidence” that  $y_i$  is 0 or 1. Here, we obtain

$$r_{ij}^l(0) + r_{ij}^l(1) = 1 \quad (2.12)$$

$r_{ij}^l(0)$  is a likelihood that there is an even number of 1's of on all other data knots instead of  $c_i$ . Initially, let us suppose the likelihood of having an even number of 1's on 2 data knots. Let  $q_1$  be the likelihood that there is a 1 at data knot  $c_1$  and let  $q_2$  be the likelihood that there is a 1 at data knot  $c_2$ . We have

$$P_r[c_1 \oplus c_2 = 0] = q_1 q_2 + (1 - q_1)(1 - q_2) \quad (2.13)$$

$$= 1 - q_1 - q_2 + 2q_1 q_2 \quad (2.14)$$

$$= \frac{1}{2}(2 - 2q_1 - 2q_2 + 4q_1 q_2) \quad (2.15)$$

$$= \frac{1}{2}[1 + (1 - 2q_1)(1 - 2q_2)] = q \quad (2.16)$$

Then we assume the likelihood to have an even number of 1's on 3 data knots,  $c_1$ ,  $c_2$  and  $c_3$ . Here, notice that  $1 - q$  is the likelihood of having an odd number of 1's on  $c_1$  and  $c_2$ .

$$= \frac{1}{2}[1 + (1 - 2q_1)(1 - 2q_2)(1 - 2q_3)] \quad (2.17)$$

Generally

$$P_r[(c_1 \oplus \dots \oplus c_n = 0)] = \frac{1}{2} + \frac{1}{2} \prod_{i=1}^n (1 - 2q_i) \quad (2.18)$$

Consequently, the data that  $f_j$  transmits to  $c_i$  at round  $l$  is

$$r_{ji}^{(l)}(0) = \frac{1}{2} + \frac{1}{2} \prod_{i' \in V_j, i' \neq i} (1 - 2q_{i'}(l)) \quad (2.19)$$

$$r_{ji}^{(l)}(1) = 1 - r_{ji}^{(l)}(0) \quad (2.20)$$

whereas  $f_j$  is the group of all data knots linked with the check knot. The data that  $c_i$  transmits to  $f_j$  at round  $l$  is

$$q_{ji}^{(l)}(0) = k_{ij}(1 - p_i) \prod_{j' \in C_i, j' \neq j} r_{j'i}^{(l-1)}(0) \quad (2.21)$$

$$q_{ji}^{(l)}(1) = k_{ij} p_i \prod_{j' \in C_i, j' \neq j} r_{j'i}^{(l-1)}(1) \quad (2.22)$$

while  $c_i$  is the combination of all check knots associated with the data knot  $c_i$ . The constant  $k_{ij}$  is selected to perform

$$q_{ji}^{(l)}(0) + q_{ji}^{(l)}(1) = 1 \quad (2.23)$$

At every data knot, the computations given below are completed

$$Q_i^{(l)}(0) = k_i(1 - p_i) \prod_{j \in C_i} r_{ji}^{(l-1)}(0) \quad (2.24)$$

$$Q_i^{(l)}(1) = k_i p_i \prod_{j \in C_i}^n r_{ji}^{(l-1)}(1) \quad (2.25)$$

$Q_i^{(l)}$  is the efficient likelihood of 0 and 1 at data knot  $c_i$  at round  $l$ .

If

$$Q_i^{(l)}(1) > Q_i^{(l)}(0) \quad (2.26)$$

the assessment at the point where  $c_i = 1$ , and  $c_i = 0$ . This analysis fulfills the equality-check equations and then dismisses them. Otherwise, the technique is executed via an early decided number of repetitions.

As mentioned earlier, this technique uses many multiplications, which is not feasible to deploy. One more method is the logarithmic likelihood percentage as

$$L_i = \left( \frac{P_r[c_i = 0 | y]}{P_r[c_i = 1 | y]} \right) = \frac{1 - p_i}{p_i} \quad (2.27)$$

$$l_i = \ln L_i = \ln \left( \frac{P_r[c_i = 0 | y]}{P_r[c_i = 1 | y]} \right) \quad (2.28)$$

Here,  $L_i$  is the likelihood percentage and  $l_i$  is the log likelihood percentage where the data knot is  $c_i$ . By applying the log percentage, it converts multiplications into additions, which uses fewer hardware resources. With a log likelihood percentage, we obtain

$$P_i = \frac{1}{1 + L_i} \quad (2.29)$$

From Eq. (2.21) and Eq. (2.22), the data that  $c_i$  transmits to  $f_j$  at round  $l$  is

$$= l_i + \sum_{j \in C_i \neq i} m_j^{(l-1)} \quad (2.30)$$

$$m_{ji}^{(l)} = \ln \frac{r_{ji}^{(l)}(0)}{r_{ji}^{(l)}(1)} = \ln \frac{\frac{1}{2} + \frac{1}{2} \prod_{i \in M_j \neq i} (1 - 2q_{ij}^{(l-1)}(1))}{\frac{1}{2} - \frac{1}{2} \prod_{i \in M_j \neq i} (1 - 2q_{ij}^{(l-1)}(1))} \quad (2.31)$$

$$= \ln \frac{1 + \prod_{i \in V_j \neq i} \tanh\left(\frac{m_{ij}^{(l-1)}}{2}\right)}{1 - \prod_{i \in V_j \neq i} \tanh\left(\frac{m_{ij}^{(l-1)}}{2}\right)} \quad (2.32)$$

We have (2.30) due to (2.32),

$$e^{m_{ij}} = \frac{1 - q_{ij}(1)}{q_{ij}(1)} \quad (2.33)$$

Thus

$$q_{ij}(1) = \frac{1}{1 + e^{m_{ij}}} \quad (2.34)$$

And

$$1 - 2q_{ij}(1) = \frac{e^{m_{ij}} - 1}{e^{m_{ij}} + 1} = \tanh\left(\frac{m_{ij}}{2}\right) \quad (2.35)$$

Eq. (2.24) and Eq. (2.26) fit into

$$l_i^{(l)} = \ln \frac{Q_i^{(l)}(0)}{Q_i^{(l)}(1)} = l_i^{(0)} + \sum_{j \in C_i} m_{ji}^{(l)} \quad (2.36)$$

If  $l_i^{(l)} > 0$ , then  $c_i=0$  else  $c_i=1$

In training, a certain broadcast is performed for the first of either a large number of repetitions or until the transmitted probabilities of approaching the belief. As assured chance based  $l_i = \pm\infty$ , where  $P_i = 0$  for  $l_i = \infty$  and  $P_i = 1$  for  $l_i = -\infty$ .

One essential feature of belief propagation is its performance timing, which is linear to the code size. The process negotiates among check knots and data knots thereby reducing the number of traversals. Furthermore, if the technique performs a steady repetition, then every boundary will traverse a fixed number of times, thereby showing that a number of processes are stable and only rely upon the number of limitations. If we let the number of check knots and data knots grow linearly with the code length, the number of processes carried out by belief propagation also rises linearly with the code size.

## 2.4. Luby Transformer Code (LT)

The comprehension of LT codes for a period of removal codes are called common removal codes. The symbol size for the codes may be arbitrary, from one-bit binary value to general 1 bit value. It is possible to analyze that value route time in the aspect encoding and decoding of symbolic processes, where a  $v$  process is either an exclusive-or or duplicate of one symbol to another. If the real data is comprised of  $k$  input symbols, then every encoded symbol can be produced self-reliantly of all other encoded symbols on average by  $O(\ln(k/\delta))$  symbol processes, and the  $k$  real input symbols can be improved from any  $k+O(\sqrt{k} \ln 2(k/\delta))$  of the encoded symbols with likelihood  $1 - \delta$  at an average of  $O(k \ln(k/\delta))$  symbol processes. LT codes are rateless. For example, the number of encoded symbols produced from the data would be infinite. Likewise, encoded values can be produced rapidly and might be less or more as required. Similarly, decoding can improve a quiet same copy of the data from any group of the produced encoded symbols that collectively are only somewhat lengthier than the data. Therefore, without any doubt the loss model is on the erasure channel and encoded symbols can be produced as needed and transmitted over the erasure channel until a sufficient number has been reached at the decoder in order to repair the data. Meanwhile, the decoder can repair the data from the least number possible of encoded symbols, thereby implying that LT codes are near optimum with deference to any erasure channel. Additionally, the encoded and decoding periods are asymptotically very well-organized as a function of the data size. Therefore, LT codes are world demanded in the aspect that they are concurrently close optimal for every erasure channel and they are very effectual as the data size develops. The analyses of LT codes are more diverse than examinations of Tornado codes [31] [32] [33]. In actuality, the study of Tornado codes is only appropriate to graphs with constant all-out degree. LT codes use graphs of logarithmic solidity, hence the Tornado code analysis does not do so. In addition, the investigation of Tornado codes depends on the techniques that respond overhead, that is, integrally at least to a constant fraction of the data size, while the evaluation of LT codes shows us that the response overhead is an asymptotically disappearing fraction of the data size, in spite of being at the price of less advanced asymptotic

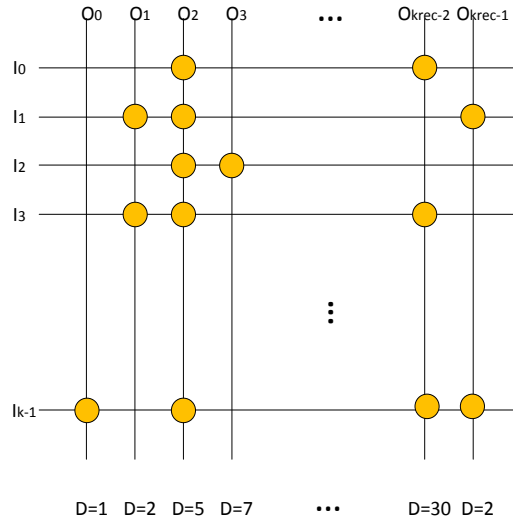
encoding and decoding times. The key to the design and examination of LT codes is to introduce and analyze the LT procedure. This procedure and its analysis are innovative simplifications of the traditional procedure and examination of throwing balls arbitrarily into baskets and therefore may well be of self-governing attention. Provided here is the analysis of the performance of the LT code using initial values of likelihood theory, which accurately grab the performance of the data retrieval method. The “digital fountain method” idea presented in [34] [35] is similar to universal erasure code, with LT codes leading full recognition of this study.

### 2.4.1. LT encoder

The procedure of creating an encoded symbol is theoretically easy to define:

- Arbitrarily select the degree  $d$  of the code symbol from a degree distribution.
- Select consistently at chance based  $d$  separate input symbols as the closest of the code symbols.
- The value of the code symbol is the exclusive-or of the  $d$  fellows.

These input symbols are known as a neighbor for the particular output code symbol and the directories of these fellows is given by the set  $\gamma$ . Stating to Fig.4, we analyze that for  $O_0$ ,  $D = 1$  is yielded and the parallel to arbitrarily selected input index (or the fellows)  $\gamma$  is  $I_{k-1}$ . Hence,  $O_0$  comprises only  $I_{k-1}$ . For  $O_1$ , we gain  $D = 2$  and  $\gamma = \{1; 3\}$ , and henceforth we have  $O_1 = I_1 \text{ XOR } I_3$  etc.



**Figure 4:** The LT encoding/decoding process

The inputs  $I$  are to the left and the output encoded symbols  $O$  at the top. The input/output association in reality shows that the circles link to the given input and output.

At the end of each column in Fig.4, the total degree  $D$  of the output symbol can be seen. Similarly, note that the procedure mentioned above is not bound to bits only. Bit vectors can be utilized as input symbols and an additional process, then substitutes the XOR process for encrypting symbol generation. As analysed, the association among the inputs and outputs can be demonstrated again by a connectivity matrix (denoted by  $H_c$ ). Outcomes are characterized again by columns and the inputs are shown as rows of  $H_c$ . Each time when an input establishes a portion of an outcome, the consistent entry in the matrix for the particular row and column is marked 1. For example, for Fig.4, the connectivity matrix can be shown as in Eq. (2.37):



$$H_c = \begin{bmatrix} 0 & 0 & 1 & 0 & \cdots & \cdots & 1 & 0 \\ 0 & 1 & 1 & 0 & \cdots & \cdots & 0 & 1 \\ 0 & 0 & 1 & 1 & \cdots & \cdots & 0 & 0 \\ 0 & 1 & 1 & 0 & \cdots & \cdots & 1 & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \cdots & \vdots & \vdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & 0 & 1 & 0 & 0 & 0 & 1 & 1 \end{bmatrix} \quad (2.37)$$

This matrix needs to be aware of the receiver for suitable deciphering of the inputs.

And later, the encoding procedure can be mentioned as:

$$O = I \times H_c \quad (2.38)$$

Where O is of size  $1 \times k_{rec}$ , I is size  $1 \times k$  and Hc is of size  $k \times k_{rec}$ . While utilizing the encoding symbols to retrieve the original input symbols of the data, the decipherer requires being familiar with the degree and group of fellows of each code symbol. There are various methods of collaborating this data with the decipherer, based on the application, including the degree and a list of companion directories which might be given openly to the interpreter for each code symbol. In one more example, the degree and companion indices of each code symbol can be calculated by the decoder indirectly. This is all dependent upon the timing of receiving of the code symbol or location of the code symbol as compared to the situations of other code symbols. Here is another example where an encoding key may be connected with each code symbol and then both the encoder and decoder apply the same purpose of the key in order to yield the degree and a group of companions of the code symbol. In this case, the encoder may arbitrarily select each key it uses to generate an code symbol and keys may be passed to the decoder along with the code symbols. Every key might rather be generated such as by a deterministic procedure, for example, where each key may be bigger than the previous key. The encoder and decoder may be permitted access to the few sets of arbitrary bits, and the key may be utilized as the kernel to a pseudo-arbitrary producer that uses these arbitrary bits in order to produce the degree and companion of the code symbol.

## 2.4.2. Degree distributions for LT code

If we repeat each code symbol that has an independently selected degree from a degree distribution, the degree dispersal of all  $d$ ,  $\rho(d)$  is the likelihood that an code symbol has degree  $d$ . Here, we are supposed to create the arbitrary performance of the LT procedure which is totally defined by the degree distribution  $\rho(\cdot)$ , the number of code symbols  $K$ , and the number of input symbols  $k$ . Our objective is to design degree distributions that meet the following two design goals.

As few code symbols as possible in the medium are needed to ensure the completion of the LT procedure. Repeating the number of code symbols, which ensures success of the LT procedure resembles the number of code symbols that assure complete recovery of the data. The medium degree of the code symbols is as low as possible. The normal degree is the number of a symbol's processes on average it takes to produce an code symbol. The average degree times  $K$  is the number of symbol processes on average it takes to recuperate the whole data.

Here, we will see how to design  $\rho(D)$  to achieve the given conditions.

### 2.4.2.1. All-at-once distribution

For  $\rho(1) = 1$ , the degree distribution is named as All-At-Once and every code symbol has a degree of accurate 1 and thus  $k_{\text{rec}} = K$ . Henceforth, all the input symbols are added to the wave at once. Also, we observe that each symbol has a minimal degree of 1, thus the amount of processing of each distinct symbol is negligible. Nevertheless, the total number of code symbols required is unnecessarily excessive, i.e.  $k \ln(k/\delta)$ . Consequently, this kind of delivery is never used in practice.

### 2.4.2.2. Ideal soliton distribution

The major aim of this delivery is to keep  $k_{rec}$  as low as possible by following to the stated total degree of  $K$ . One outcome from being the total of degrees that is at least  $k \cdot \ln(k/\delta)$  to conceal all input symbols. This is the traditional approach of ideal soliton circulation. It allows to use just above  $k$  code symbols with a total degree of  $k \cdot \ln(k/\delta)$  in order aptly to cover  $k$  input symbols. This will make sure that the whole number of code symbols transferred is lowest and also the total processes to decipher the incoming vector is also less, for the provided number of input symbols and failure likelihood. The Ideal Soliton circulation upholds the wave size through the decoding procedures, equal to 1. For example, with each repetitive step one symbol is done and precisely one symbol is attached to the wave. This behaviour can be expressed by the following equation:

$$\rho(d) = \begin{cases} 1/K & \text{for } d = 1 \\ 1/d(d-1) & \text{for } d = 2, \dots, K \end{cases} \quad (2.39)$$

### 2.4.2.3. Robust soliton distribution

Robust soliton distribution tries to enhance best soliton distribution in this method describe a new division which is as follows:

$$R = C \cdot \sqrt{K} \cdot \ln(K/\delta) \quad (2.40)$$

Here  $C$  and  $\delta$  shows two limitations. The mean of degree distribution relies on  $C$  parameter. That shows the likelihood of low degrees can be upsurge, when we choose  $C$  as a small value.  $\delta$  this parameter signifies the failure likelihood of the decoder to rebuild the real data. This distribution as follows:

$$\rho(d) = \begin{cases} 1/K & \text{for } d = 1 \\ 1/d(d-1) & \text{for } d = 2, \dots, K \end{cases} \quad (2.41)$$

$$\varphi(d) = \begin{cases} R/(dK) & d = 1, 2, 3, \dots, (K/R) - 1 \\ R \ln(R/\delta)/K & d = (K/R) \\ 0 & \text{else} \end{cases} \quad (2.42)$$

The standardization factor is:

$$\tau = \sum_{d=1}^K (\rho(d) + \varphi(d)) \quad (2.43)$$

Then the Robust Soliton Distribution (RSD) is:

$$\lambda_{RSD}(d) = \Sigma(\rho(d) + \varphi(d)) / \tau \quad (2.44)$$

#### 2.4.2.4. Sparse distribution

This study presents the possibility to optimize degree distributions by evolutionary process. In the optimization framework, the context of enormous search space is a key issue. The issue dimensionality is compared to the source data size  $k$  since an individual should represent the likelihood on each degree from 1 to  $k$ . It is not easy to manage too many variables even though  $k$  reaches into hundreds. Adopting a sparse degree delivery is a substitute solution which often used in LT codes optimization. In a sparse degree distribution, non-zero likelihoods distribute on only partial degrees, which are predefined by a set of tags. The set of tags is a subset of all degrees and used to bound the search in a sub-space whose size is much less than full degrees.

$$\Psi(x) = \frac{0.008x + 0.494x^2 + 0.166x^3 + 0.073x^4}{0.083x^5 + 0.056x^8 + 0.037x^9 + 0.056x^{19} + 0.025x^{65}} \quad (2.45)$$

The index of  $x$  is the degree and the value which is in front of  $x$  is the likelihood of the degree.

#### 2.4.3. Decoding of Luby transform (LT) code

This segment presents two techniques which are used in LT decoding to rebuild the data bits. The first is a hard decoding approach and used the data transient technique and the second method known as soft decoding process based on likelihood changing among variable knot and check knot.

##### 2.4.3.1. Hard decoding Algorithm

While introducing the invention of LT codes, Luby presented a data passing decrypting process to these erasure codes. The data passing approach is toughest

decoding scheme, which can be very fast and effective. We call LT codes with Robust soliton distribution under data passing decoding, conventional LT codes or Luby's LT codes. There are two types of data passing decoding with the LT code: serial decoding, and parallel decoding.

#### **2.4.3.1.1. Serial decoding**

The serial decoding procedure of LT codes is as follows:

Step 1: Find one  $v$ -knot of degree 1 and allocate its symbol value to its neighboring  $s$ -knot. The  $v$ -knot is now said to have been released, no longer useful and the neighboring  $s$ -knot is improved. At the same time, the edge incident on this  $v$ -knot is removed, which is equal to delete a corresponding row from the producer matrix reputable by the decoder.

Step 2: The value of the newly reinstated  $s$ -knot is XORed to the rest of its companions and its edges are then detached from the graph, i.e., the column corresponding to the improved source symbols is deleted from the generator matrix.

Step 3: If all the  $s$ -knots are recovered, the decoding is done and fruitful. Otherwise, repeat from Step 1. If there is no degree-1  $v$ -knot and the decoding is not done,  $c$  flag failure.

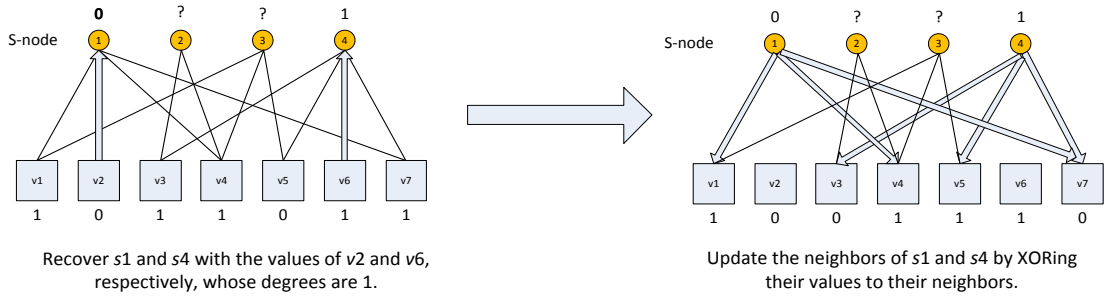
#### **2.4.3.1.2. Parallel decoding**

Parallel decoding can be employed as easily as a serial technique but with faster performance. An example of parallel data passing decoding of an LT code is given in Fig.5. The code dimension is  $k = 4$  and the number of received packets is  $n = 7$ , so an  $(n, k) = (7,4)$  truncated LT code is measured at the receiver. Suppose the producer matrix of the truncated LT code is  $G$  given by Eq. (2.46). Because a packet is equivalent to a symbol which can be presented with a bit order whose size ranges from one bit to multiple bits, for simplicity we set the symbol size to be one bit. Let assume the received encrypted symbols are [1 0 1 1 0 1 1]. It takes four iterations to

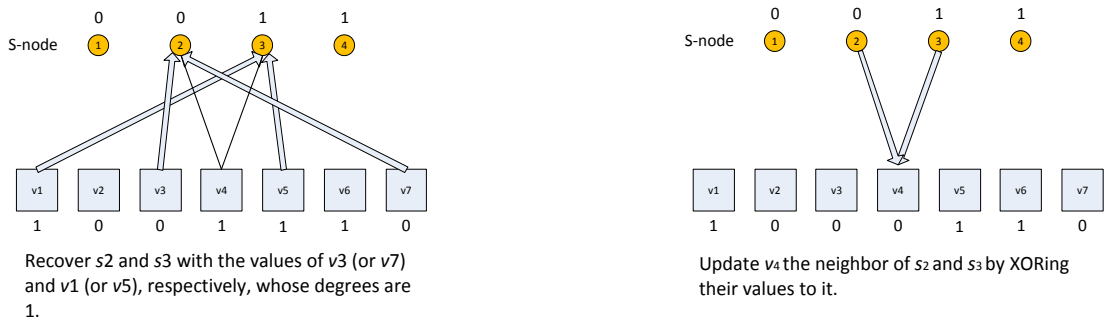
restore the four source symbols [0 0 1 1] in the serial manner, but it needs only two iterations to finish the parallel decoding.

$$G = \begin{pmatrix} 1 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 & 1 \end{pmatrix} \quad (2.46)$$

### First iteration



### The second iteration – end of decoding



**Figure 5:** Parallel decoding of LT decoder

### 2.4.3.2. Soft decoding algorithm

Though various distinctive techniques used for the deciphering of block codes can be employed for the decoding of LT codes, the well-known belief propagation (BP) procedure is most favoured by many researchers [28-33]. The BP approach uses data transferring logic between check knots and variable knots in the Tanner graph. Data exchanged between check and variable knots in a repetitive process until complete deciphering occurs. LDPC codes are signified by Tanner graphs with one class of variable knots which represent the data symbols. LT codes, on the other hand, have

two classes of variable knots. One of which presents the data symbols and the other showing the position of the data symbols, including symbol indices which are used in the encoding operation liable on the degree of these symbols. The first one presents the data knots and these knots are not communicated, there are no data flows in the channel. The second one is delivered over the channel which presents the code variable knots.

Let  $g_{i,j}$  be the value of  $i$  row and  $j$  column in the generator matrix  $G$ . Some parameters of the BP algorithm are as follows:

$L(v) = \{ l: g_{v,l} = 1 \}$  : the set of code positions that participate in the  $v$ -th check equation.

$V(l) = \{ v: g_{v,l} = 1 \}$  : the set of check positions in which code position  $l$  participates.

$q_{v,l}^y$  : The likelihood that the value of variable knot  $l$  is  $y$ , given the information found via the check knots other than check knot  $v$ .

$r_{v,l}^x$  : The likelihood that a check knot  $v$  is pleased when variable  $l$  is fixed to a value  $y$  and given the information obtained via the variable knots other than variable knot  $l$ . In the following, a binary broadcast over an AWGN channel is expected. Moderated symbols  $m(i)$  are transmitted over an AWGN channel and  $r(i) = m(i) + n(i)$  denote received symbols where  $n(i)$  is Gaussian white noise, which is self-governing and obeys normal distribution  $\eta(0, \sigma^2)$ .

#### Initialization

Since the receiver obtains only the encoded variable knots which are communicated over the channel, the initial knowledge of source symbols is zero. For  $l = 1, 2, \dots, L, K$ , initialize the priori probabilities of the knots of source symbols:

$$p_l^1 = (1/1 + \exp(-2r/\sigma^2)) \& p_l^0 = (1 - p_l^1) \quad (2.47)$$

For every  $(l,v)$  , such that  $g_{v,l}=1$

$$q_{v,l}^1 = p_l^1 \& q_{v,l}^0 = p_l^0 \quad (2.48)$$

Data crossing via the source symbols and encrypting symbols of LT code are considered variable knots with which to deal with.

Step 1: Bottom-up (horizontal): For each  $(l,v)$ , compute  $\delta r_{v,l}$  &  $r_{v,l}^1$  and  $r_{v,l}^0$ .

$$r_{v,l}^1 = (1 - \delta r_{v,l}) / 2 \& r_{v,l}^0 = (1 + \delta r_{v,l}) / 2 \quad (2.48)$$

Step 2: Top-down (vertical): For each  $(l,v)$ , update the values of  $q_{v,l}^1$  &  $q_{v,l}^0$  and normalize:

$$q_{v,l}^1 = p_l^1 \prod_{v' \in \mathcal{V}(l) \setminus v} r_{v,l}^1 \& q_{v,l}^0 = p_l^0 \prod_{v' \in \mathcal{V}(l) \setminus v} r_{v,l}^0 \quad (2.49)$$

$$\alpha = 1 / (q_{v,l}^1 + q_{v,l}^0) \quad (2.50)$$

$$q_{v,l}^1 = q_{v,l}^1 * \alpha \quad (2.51)$$

$$q_{v,l}^0 = q_{v,l}^0 * \alpha \quad (2.52)$$

For each  $l$ , calculate the a posteriori probabilities:

$$q_l^1 = p_l^1 \prod_{v \in \mathcal{V}(l)} r_{v,l}^1 \quad q_l^0 = p_l^0 \prod_{v \in \mathcal{V}(l)} r_{v,l}^0 \quad (2.53)$$

Finally, the hard decision approach is used to rebuild the data as follows:

If  $q_l^1 > q_l^0$ , the bit is 1 otherwise the bit is zero.

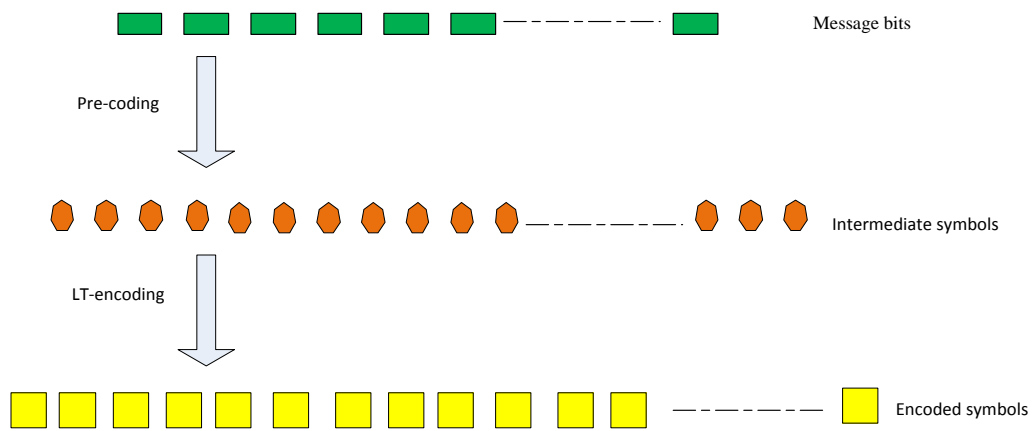
## 2.5. Raptor Codes

A Raptor code [36] is an extension lead of an LT code, but it can accomplish linear difficulty with data passing deciphering. Raptor codes have been assumed by The 3<sup>rd</sup> Generation Partnership Project (3GPP), a teamwork contract that was carried out in December 1998. Within the framework of 3GPP, Raptors codes are used for dependable data delivery in mobile wireless broadcasting and multicasting. Moreover, Raptor codes are concatenated codes.



### 2.5.1. Raptor encoding

Raptor codes are concatenated codes. A diagram of overall Raptor encoding is shown in Fig.6. The source data symbols are first pre-coded by an outer code C. The outcome symbols of the pre-code are known as midway symbols which are the input symbols of an inner LT code. The pre-code is a stable rate erasure code usually with a fairly high rate. The pre-code can be multistage, i.e., the pre-code can be a concatenation of multiple fixed rate erasure codes. The inner LT code is sometimes called a weakened LT code [37].



**Figure 6:** Raptor encoding diagram.

A Raptor code can be signified by  $(k, C, \rho(x))$ , where  $k$  is the number of source symbols and  $\Omega(x)$  is the polynomial of the encoded symbol degree distribution that the inner LT code takes on the intermediate symbols.  $\rho(x)$  is specified as

$$\rho(d) = \sum_d \rho_d x^d \quad (2.54)$$

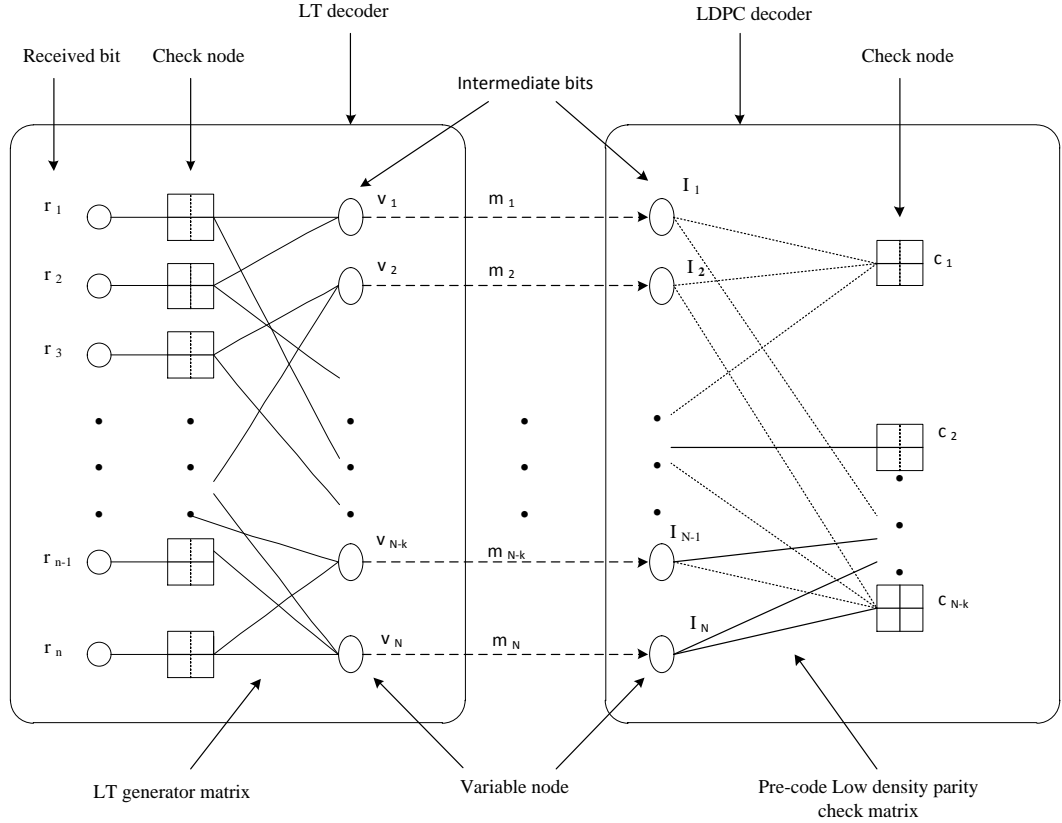
where,  $d$  is the selected degree of the LT encrypted symbol and  $\rho_d$  is the likelihood that the value  $d$  is chosen. The average degree of the encoded symbols is equal to  $\rho(1)$ , where  $\rho'(x)$  is the derivative of  $\rho(x)$  with respect to  $x$ .

## 2.5.2. Raptor decoding using belief propagation

The iterative data passing procedure is executed on two matrices: the producer matrix of the inner LT code, and the equality-check matrix of the outer erasure code. The intermediate symbols are not only the  $v$ -knots of the Tanner graph of the equality-check matrix of the pre-code's, but also the  $s$ -knots of the generator matrix LT code. The LT decoder passes updated decoding data to the pre-code decoder and hard decisions are made on the output of the pre-code decoder. If the highest allotted number of iterations ends and if all the interstitial symbols have obtained non-zero values, the decoding is successful. There are two types of data-passing approaches for Raptor codes [38]. In the first method, the LT decoder applies a data-passing algorithm to recover as many intermediate symbols as possible until the LT decoding is halted by a stopping set. The values of the recovered intermediate symbols are then passed to the  $v$ -knots of the Tanner graph of the equality-check matrix of the pre-code and the outer decoder restores the remaining intermediate symbols. This method is a local-iteration scheme. The second method is a global-iteration scheme. Each decrypting step consists of two phases: first on the LT code and the other on the pre-code. At the end of either stage, the updated values of the intermediate symbols are passed to the other decoder. The inventor of Raptor codes, Shokrollahi, stated in [36] that the pre-codes for good raptor codes are generally of high rates. The higher the pre-code rate, the fewer intermediate symbols are created. Therefore, with an identical Raptor code rate, a greater proportion of intermediate symbols can be restored by an identical inner LT code and then the error performance is accordingly improved.

### 2.5.2.1. Local-iteration decoder

A local-iteration decoder for binary Raptor codes is shown in Fig 7. At the receiver side, the fountain code is a shortened version, i.e., it is a fixed rate code at a time. Suppose right now  $n$  Raptor encoded bits have been composed at a user terminal to improve  $k$  source bits.



**Figure 7:** Raptor local-iteration decoder configuration

The corresponding decoder configuration is represented by Fig 7. The pre-code is already known by the decoder, so the Tanner graph of the equality check matrix  $\mathbf{H}$  of the pre-code is usually set up in advance. The Tanner graph of the inner weakened LT code's generator matrix  $\mathbf{G}$  can be built on the fly. The incoming output bits  $r_1, \dots, r_n$  are associated one-to-one to the c-knots of the LT Tanner graph. The rate  $k/N$  pre-code  $\mathbf{C}$  is an LDPC code. The two columns of v-knots  $v_1, \dots, v_N$  and  $I_1, \dots, I_N$  are the same set of intermediate bits distinctly for the two Tanner graphs. With packet erasure fountain code,  $v_i$  hands in its final decoding data (some LLR)  $m_i$  to  $I_i$  once the LT decoding is halted by a stopping set. However, for a Raptor code with soft decoding over an AWGN channel,  $v_i$  does not pass  $m_i$  to  $I_i$  until the maximum allocated number of LT decoding iterations is finished. The LDPC decoder takes the decoding data from the LT decoder as observed LLRs.  $c_1, \dots, c_{N-k}$  are the c-knots of the Tanner graph of the equality check matrix of the pre-code. The LDPC decoding is terminated when a valid code word output from hard decision on  $I_1, \dots, I_N$  or the maximum assigned number of LDPC decoding iterations are exhausted. Siva

Subramanian and Leib [38] introduced a concise and complete algorithm of local-iteration Raptor decoding. In the Raptor decoder, we denote decoding data by  $m$  and the decoding repetition by the superscript. In the inner LT code, we denote an accepted encoded bit by  $r$  and its LLR value by  $L$ . The LLR data sent from a v-knot  $v$  (an intermediate bit) to an encoded bit  $r$  (a c-knot) is presented by  $m_{v,r}$ , and  $m_r$ , is the deciphering data sent from a c-knot to a v-knot  $v$ . The data passed from the LT decoder to the LDPC decoder is denoted by  $m_i$  with  $1 \leq i \leq N$ . In the pre-code decoder,  $m_{i,c}$ , is the data sent from the v-knot  $I$  (the intermediate bit) to the c-knot  $c$ , and  $m_c$ , is the data sent from the c-knot  $c$  to the v-knot  $I$ . The local-iteration decoding starts with the LT decoder, transmitting the data from the v-knots to the c-knots and then transferring the data back to the v-knots. At the start of iteration 0, all data are originated to zeros, except for the experiential log likelihood ratios on the accepted encoded bits. At each iteration step, v-knots send data to c-knots and then c-knots feedback updated data to v-knots. The on-going LLR information is updated as follows [40]:

$$m_{v,r}^{(l)} = \begin{cases} 0 & l = 0 \\ \sum_{r' \neq r} m_{r',v}^{(l-1)} & l > 0 \end{cases} \quad (2.55)$$

And

$$m_{r,v}^{(l)} = 2 \tanh^{-1} \left[ \tanh(L/2) \prod_{v' \neq v} \tanh(m_{v',r}^{(l)}/2) \right] \quad (2.56)$$

where  $\tanh(\cdot)$  and  $\tanh^{-1}(\cdot)$  are the hyperbolic tangent function and the inverse hyperbolic tangent function,  $m_{r',v}^{(l-1)}$  is the LLR data sent to an intermediate bit at the already existing repetition from all its surrounding received bits except for  $r$ , and  $m_{v',r}^{(l)}$  is the LLR data sent to a received bit from all its surrounding intermediate bits except for  $v$ . After the LT decoding is terminated by a preventing set, or the allocated number  $p$  of LT reiterations are ended, the final LLR data of a transitional bit is specified as

$$m_i = \sum_r m_{r,v}^{(p)} \quad (2.57)$$

Here  $r$  is the group of incoming bits surrounding the  $i$ -th transitional bit. Taking  $m_i$  as observed LLR information from the channel, the LDPC decoder works with iteration 0. Similarly to the LT decoder, all data are originated to zeros with

exception of the received  $mi$  from the LT decoder. Each time in the repetitive process, the v-knots send data first and then receive rationalized info from c-knots. The LDPC decoder acts with given update instructions below [39]:

$$m_{I,c}^{(l)} = \begin{cases} m_i & l = 0 \\ \sum_{c' \neq c} m_{c',I}^{(l-1)} + m_i & l > 0 \end{cases} \quad (2.58)$$

$$m_{c,I}^{(l)} = 2 \tanh^{-1} \left[ \prod_{I' \neq I} \tanh(m_{I',c}^{(l)} / 2) \right] \quad (2.59)$$

Where the  $m_{c,I}^{(l-1)}$  LLR information is transmitted to a transitional bit at the earlier repetition from all its surrounding c-knots excluding  $c$ ,  $m_{I',c}^{(l)}$  is the LLR info sent to a c knot from all its surrounding interstitial bits with the exception of  $I$ . Finally, each pre-code repeats the decoding that can make hard choices on the transitional bits. With the sign function  $\text{sgn}(\cdot)$ , the hard decision on the these v knots at repetition  $l$  is provided by the column vector:

$$w_i = \left[ -\text{sgn}(\sum_c m_{c,I}^{(l)} + m_i) + 1 \right] / 2 \quad (2.60)$$

It is to be noticed that small bits of  $\mathbf{w}$  may be 0.5's because the current LLR values of these transitional bits are still 0's. Henceforth, the decrypting still desires to remain. While all the entries of  $\mathbf{w}$  are 0's or 1's, the result can be confirmed as done.

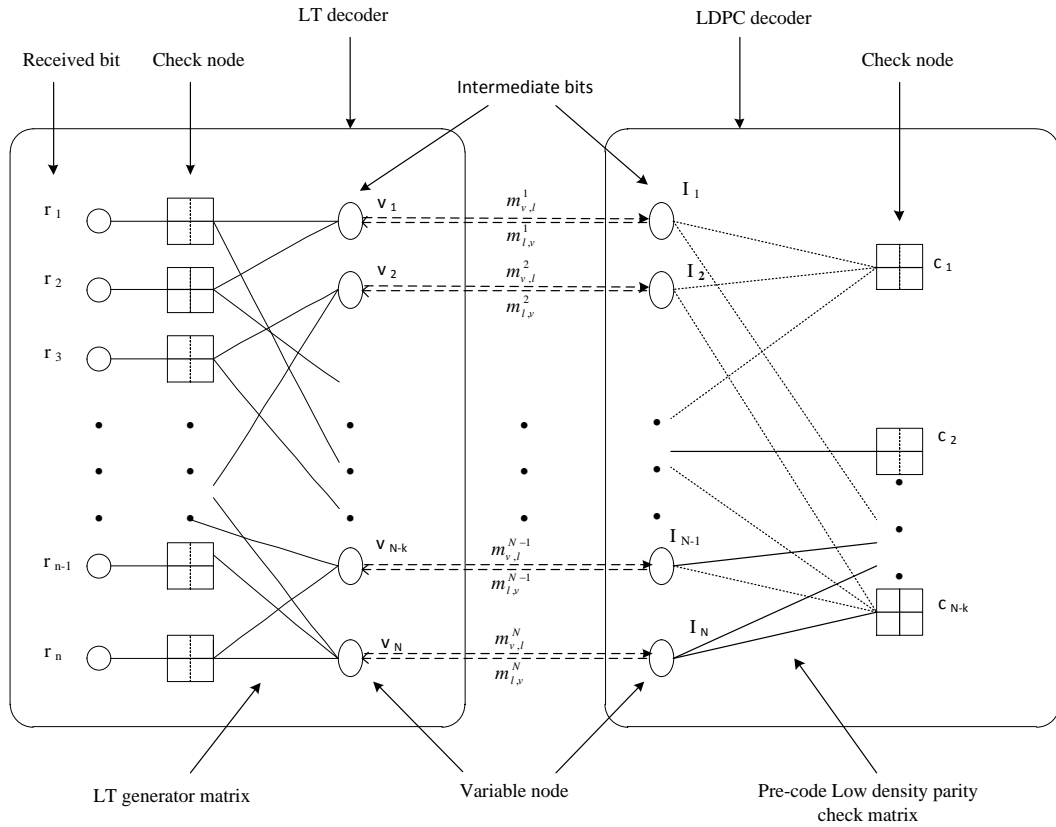
$$H\mathbf{w} = \mathbf{b} \quad (2.61)$$

With the packet adjusting fountain code, subsequently the log-likelihood ratio data are  $\pm\infty$  or 0,  $\mathbf{w}$  is surely correct if  $w_i \in \{0,1\}$ . With a lenient decoding fountain code, if  $\mathbf{b} = \mathbf{0}$ ,  $\mathbf{w}$  is a code word and the decoding of the pre-code endures and ends; if  $\mathbf{b} \neq \mathbf{0}$ , the pre-code decoding desires to endure until the repetition threshold  $q$  is grasped. If the decoder cannot output a codeword at the termination reiteration  $q$ , it may announce that the decoding is unsuccessful and gather extra encoded bits to endure decoding.

### 2.5.2.2. Global-iteration decoder

A global-iteration decoder for binary Raptor codes is shown in Fig. 8. This decoder configuration works for both packet erasure correcting and bit error correcting fountain codes. The global-iteration decoder is almost identical to the local-iteration

decoder. The only difference is that, with the alternating global-iteration scheme, the LT decoder and the pre-code decoder hand over their updated LLR data of the intermediate bit to each other at the end of every one of their own iterations. These exchanged data are also viewed as observed LLR's for the decoders.



**Figure 8:** Raptor global-iteration decoder configuration

Siva Subramanian and Leib [38] [41] and Huang et al. [42] offered the particulars of global-iteration decoding of Raptor codes. Yet again, a rate of  $k/n$  Raptor code at an arbitrary station at a rate of  $k/N$  LDPC pre-code is reflected, as presented in Figure 9. The measured significations are similar to those used with the local-iteration, excluding the data directed from  $v_i$  to  $I_i$ , which is signified by  $m_{v,i}^i$  and the data fed back from  $I_i$  to  $v_i$  by  $m_{i,v}^i$ . In the beginning, all data are originated to zeros, with exception of the experiential channel data on the composed encoded bits. The data update instructions are assumed as follows:

$$m_{v,r}^{(l)} = \begin{cases} 0 & l = 0 \\ \sum_{r' \neq r} m_{r',v}^{(l-1)} + m_{I,v}^{(l-1)} & l > 0 \end{cases} \quad (2.62)$$

$$m_{v,r}^{(l)} = 2 \tanh^{-1} \left[ \tanh(L/2) \prod_{v' \neq v} \tanh(m_{v',r}^{(l)} / 2) \right] \quad (2.63)$$

$$m_{v,I}^{(l)} = \sum_r m_{r,v}^{(l)} \quad (2.64)$$

$$m_{I,c}^{(l)} = \begin{cases} m_{v,I}^{(l)} & l = 0 \\ \sum_{c' \neq c} m_{c',I}^{(l-1)} + m_{v,I}^{(l-1)} & l > 0 \end{cases} \quad (2.65)$$

$$m_{c,I}^{(l)} = 2 \tanh^{-1} \left[ \prod_{I' \neq I} \tanh(m_{I',c}^{(l)} / 2) \right] \quad (2.66)$$

And

$$m_{I,v}^{(l)} = \sum_c m_{c,I}^{(l)} \quad (2.67)$$

In this circling procedure, the decoder can make tough decisions on the interstitial bits at the end of the pre-code decoder:

$$w_i = \left[ -\text{sgn} \left( \sum_c m_{c,I}^{(l)} + m_i \right) + 1 \right] / 2 \quad (2.68)$$

Once more, certain entries of  $\mathbf{w}$  may be 0.5's, since the consistent transitional bits have not acquired non-zero LLR values. Thus, the decoding requests to endure until the maximum number of repetitions are drained. The result can still be confirmed over Eq. (2.61).

## CHAPTER 3

### NEW DEGREE DISTRIBUTION AND MODELING RESULTS

#### 3.1. New Degree Distribution Implementation

In this chapter, a new degree distribution is discussed, and the Luby transform (LT) codes are implemented for various degree distributions over Preservative White Gaussian Noise Channels AWGN. The MATLAB program is used to deploy proposed codes and the new degree distribution. A Bit Error Rate (BER) against the signal-to-noise ratio (SNR) is accepted to amount the presentation of the Luby transform (LT) code with the new degree distribution.

##### 3.1.1. Exponential distribution

Here, two independent operations in Fig. 9 are used to perceive the degree of the likelihood values. This independent operation presents two levels of likelihood values: the first presents a high likelihood level when we compare it with the second. To begin, allocating a likelihood of degree one less than the likelihood of degree two to confirm low redundant degree packets are sent. This leads to more effective communication. After that, we select the likelihood of degree two from the first exponential and the likelihood of degree three from the second dependent values respectively and repeat this process for all degrees until obtaining all degree likelihoods. We can write the dependable degree as in Eq. (3.1) thus

$$\beta(d) = \begin{cases} \exp 1(d) & \text{for } d(\text{even} - \text{number}) = 2, 4, 6, \dots, K \\ \exp 2(d) & \text{for } d(\text{odd} - \text{number}) = 3, 5, \dots, K - 1 \end{cases} \quad (3.1)$$

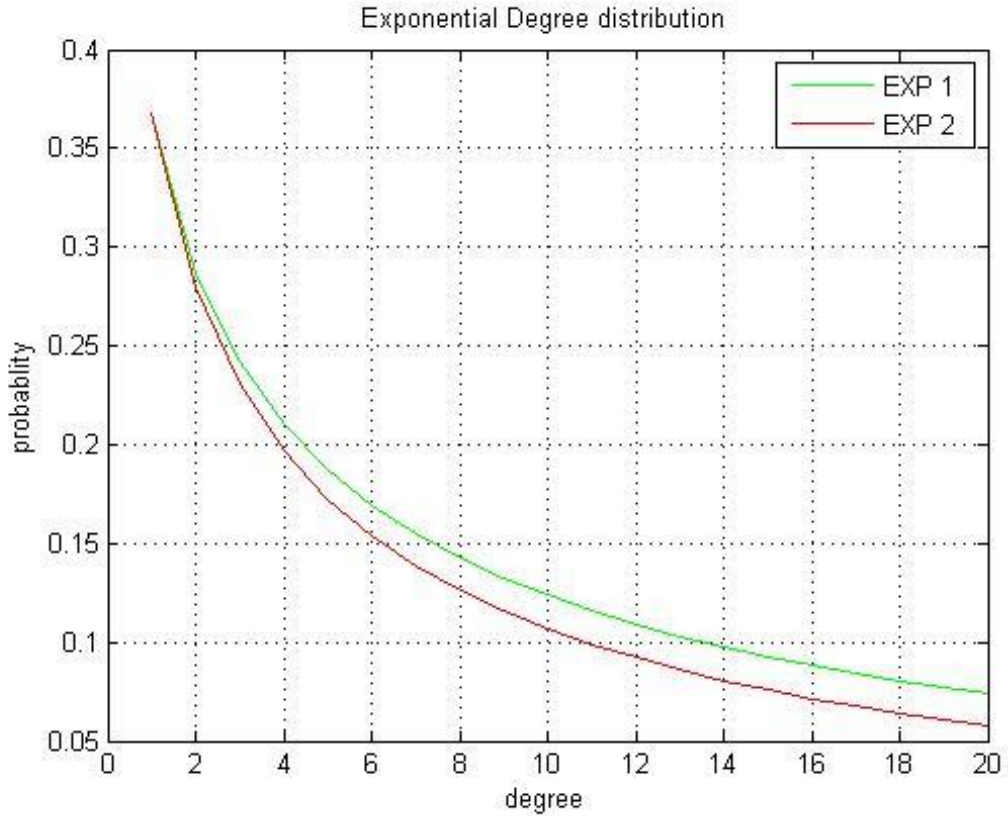
Then, standardization is used to this degree as in Eq. (3.2) as follows:



$$X = \sum_{d=1}^K \beta(d) \quad (3.2)$$

Finally, the exponential degree distribution is as in Eq. (3.3) follows:

$$\eta_{RD} = \beta(d)/X \quad (3.3)$$



**Figure 9:** Exponential degree distribution

### 3.1.2. Random degree distribution

This degree shows the customization of spars degree distribution. Random degree distribution can generate as follows:

1. In the first, random numbers  $x$  are generated with range from 1 to  $K-1$ .

$$x = 1, 2, 3, 4, 5, 6, \dots, K-1 \quad (3.4)$$

When  $K$  represents the length of message. In ours case  $K=1000$  and this means the range of generated numbers is equal to (1-999)

- After that, tags numbers  $R$  are selected from  $x$  values and these tags are chosen as multiples of the number five depends on the next Eq. (3.5).

$$R(i) = \begin{cases} x(i) & i = (5:5:K-5) \\ 0 & else \end{cases} \quad (3.5)$$

- The probabilities of these tags  $P_r$  are computed as Eq. (3.6) follows :

$$P_r(i) = R(i) / K \quad (3.6)$$

- The probability values of degree one and degree  $K$  are calculated in same manner as in ideal soliton distribution algorithm and those degree as follows:

$$P_r(1) = 1 / K \quad (3.7)$$

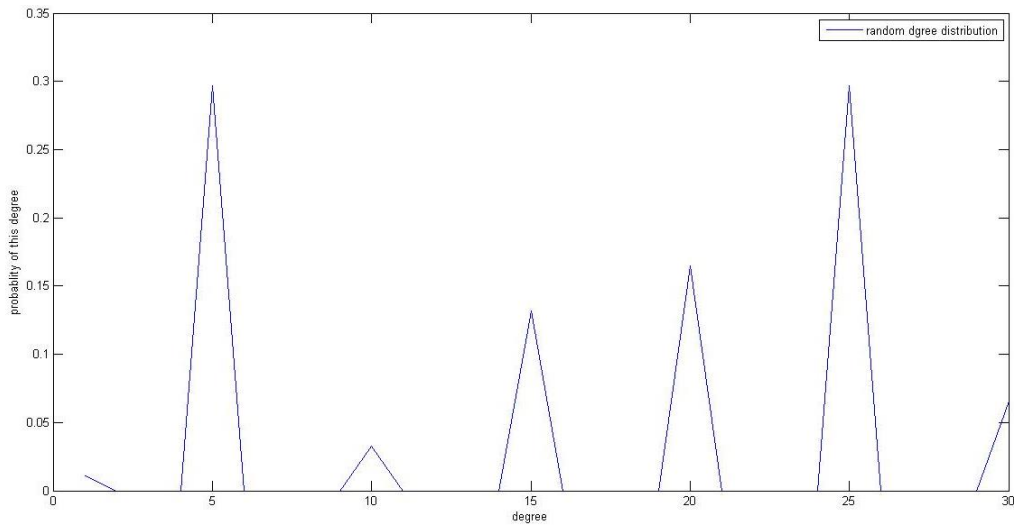
$$P_r(K) = 1 / K (K - 1) \quad (3.8)$$

- Normalization factor is used as follows:

$$Z = \sum_{i=1}^K p_r(i) \quad (3.9)$$

- Finally, calculate the random distribution  $R_{ds}$  by using Eq. (3.10) follows:

$$R_{ds} = P_r / Z \quad (3.10)$$



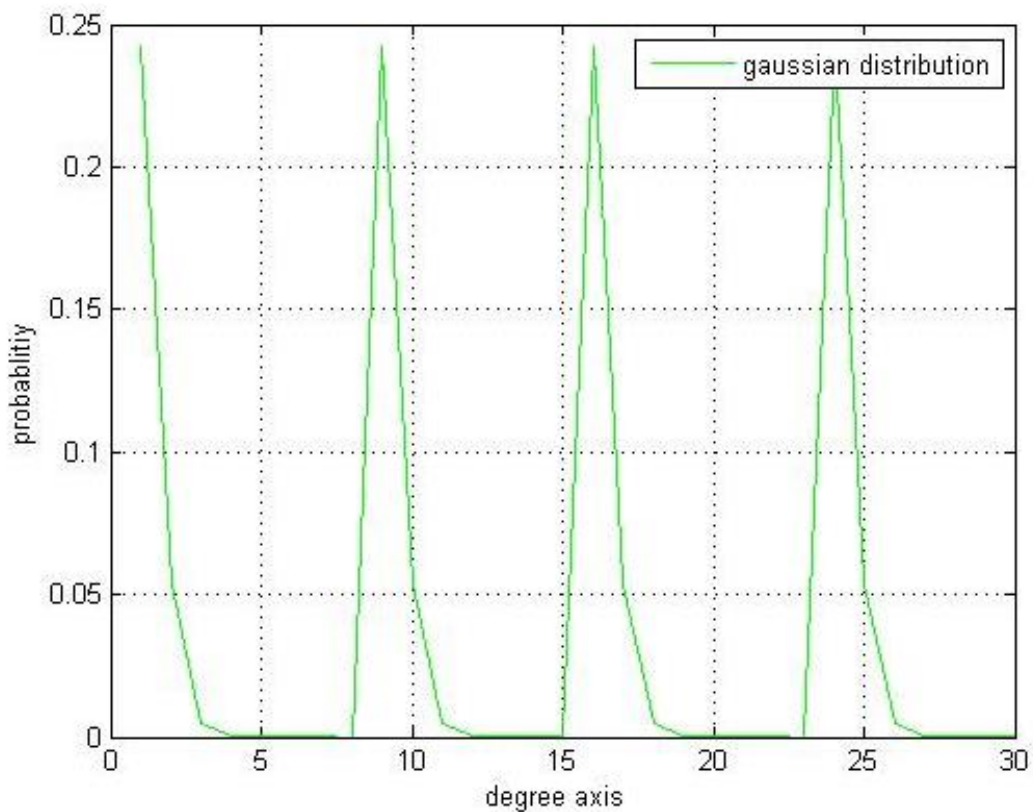
**Figure 10:** Random degree distribution

### 3.1.3. Gaussian distribution

A Gaussian degree distribution is presented in Fig. 11. The Gaussian distribution curve is attained by using Eq. (3.11). The Gaussian distribution curve is recurrent many times with the size of the degree axis. From the modeling outcomes in Fig. 16, we observe that when we increase the number of curve repetitions along the degree axis, better performance is gained. In contrast, increasing the number of curves reiterating means that growth in the mean degree distribution and many bits will take high likelihoods.

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-(x-\mu)^2/(2\sigma^2)} \quad (3.11)$$

Where  $\mu$  is the mean value and  $\sigma^2$  presents the variance. In this simulation, we set the mean to zero and the variance to 1.



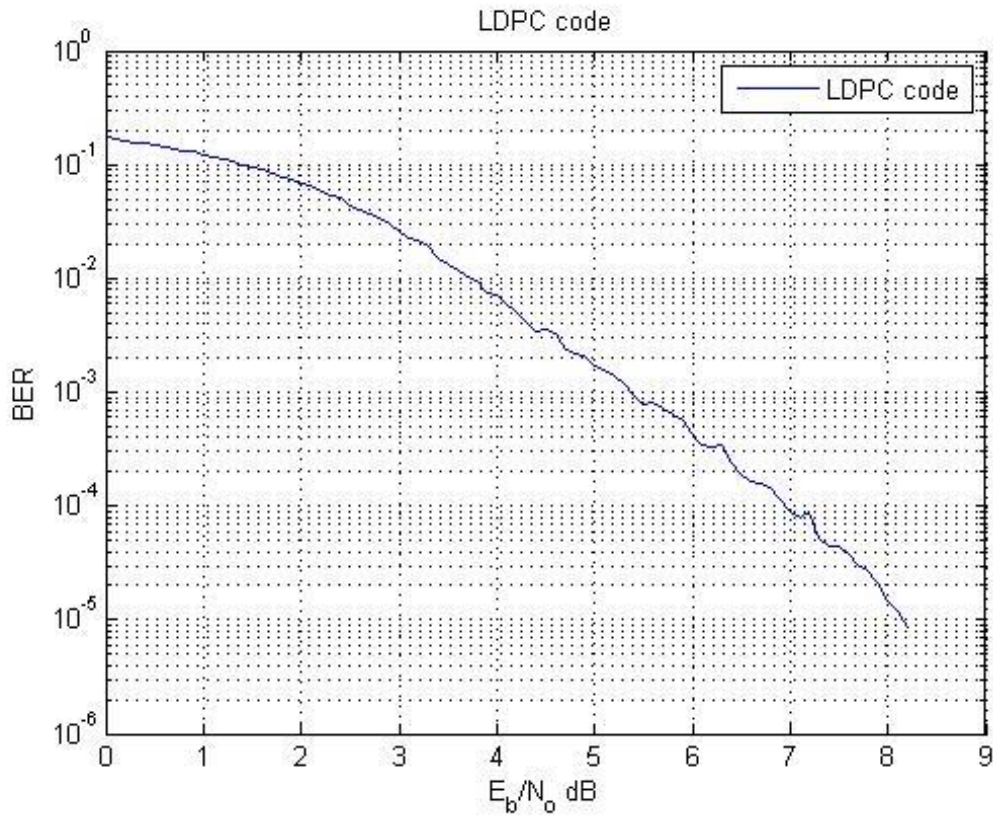
**Figure 11:** Gaussian distribution

### 3.2. Modeling Results

The employment of a variety of rate-less codes over Additive White Gaussian Noise Channels (AWGN) is achieved in this section. Luby transform code (LT) is examined with all the new degree distributions which are obtainable in this chapter. In this method, we associate the impact of two types of degrees on the performance of the LT code.

#### 3.2.1. Low density equality check code (LDPC)

Firstly, we present the simulation result for the LDPC code in Fig. 12 which is used as a pre-code in a raptor code.



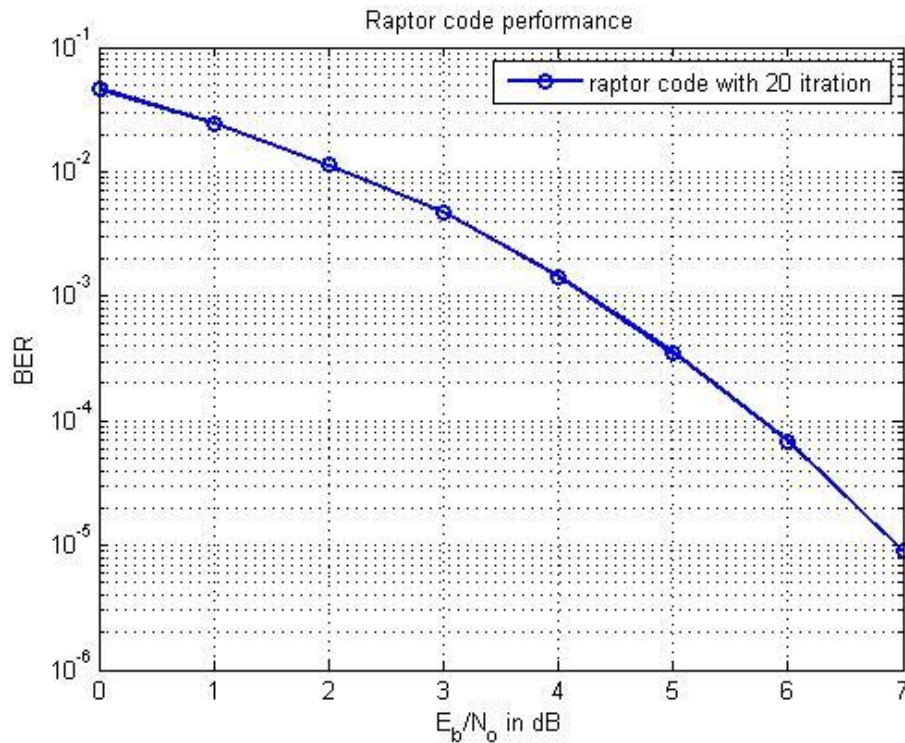
**Figure 12:** LDPC code performance

Fig. 12 shows an LDPC code presentation with a rate of  $1/2$ . The size of the code word is  $N=1032$  and data bits is  $K=516$  and by using a BPSK modification over the AWGN channel, with 20 repetitions and a signal to noise ratio change from 0 to 9.

### 3.2.2. Raptor codes

In this type of code, two codes are concatenated. The first one involves one stage of pre-code (LDPC) and the second one is LT code with two parameters  $c=0.1$ , and  $\delta=0.05$  with robust soliton distribution as already discussed in Chapter 2. The two decoding methods are achieved in a raptor code decoder (local-iteration decoder and global iteration decoder). The presentation of two decoding schemes is the main reason for presenting this chapter with a comparison between the results of these methods.

#### 3.2.2.1. Local-iteration decoder



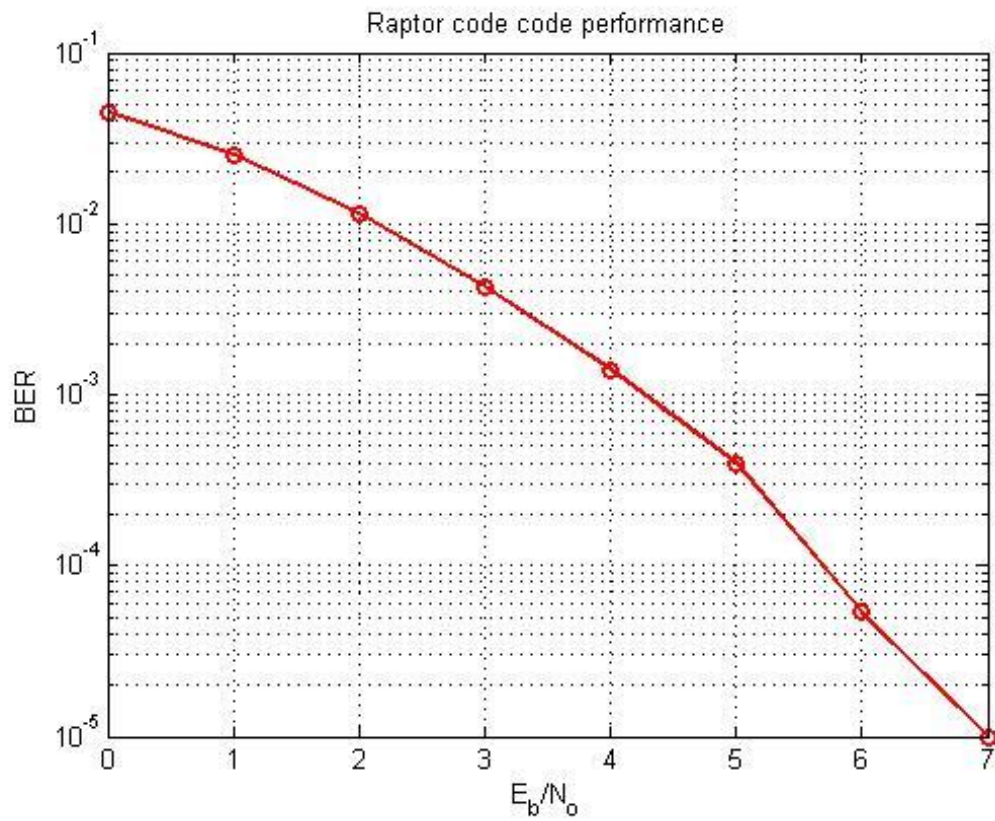
**Figure 13:** Raptor code performance using local-iteration decoder

Fig. 13 introduces raptor codes with a local-iteration Decoder with a pre-code LDPC at a rate of  $\frac{1}{2}$  and an inner Luby transform code (LT). The length of the Raptor codeword is  $N=1416$  and message bits is  $K=516$ . BPSK modulation over an AWGN

channel, in 20 iterations and at a signal-to-noise ratio (SNR) in dB varying from 0 to 7 is used.

### 3.2.2.2. Global-iteration decoder

Fig. 14 shows the performance of raptor codes with Global-Iteration Decoder with pre-code LDPC at a rate of  $\frac{1}{2}$  and the internal Luby transform code (LT). The size of the raptor codeword is  $N=1416$  and data bits  $K=516$  using a BPSK modulation over an AWGN channel, with 20 repetitions and a signal to noise ratio SNR in dB varying from 0 to 7.



**Figure 14:** Global-iteration decoder

### **3.3. LT Code Modeling Result**

This section is a review of the Luby transforms code with impacts of various degrees and relates these degree impacts on presentation of this code over an AWGN channel.

#### **3.3.1. LT code with robust soliton distribution**

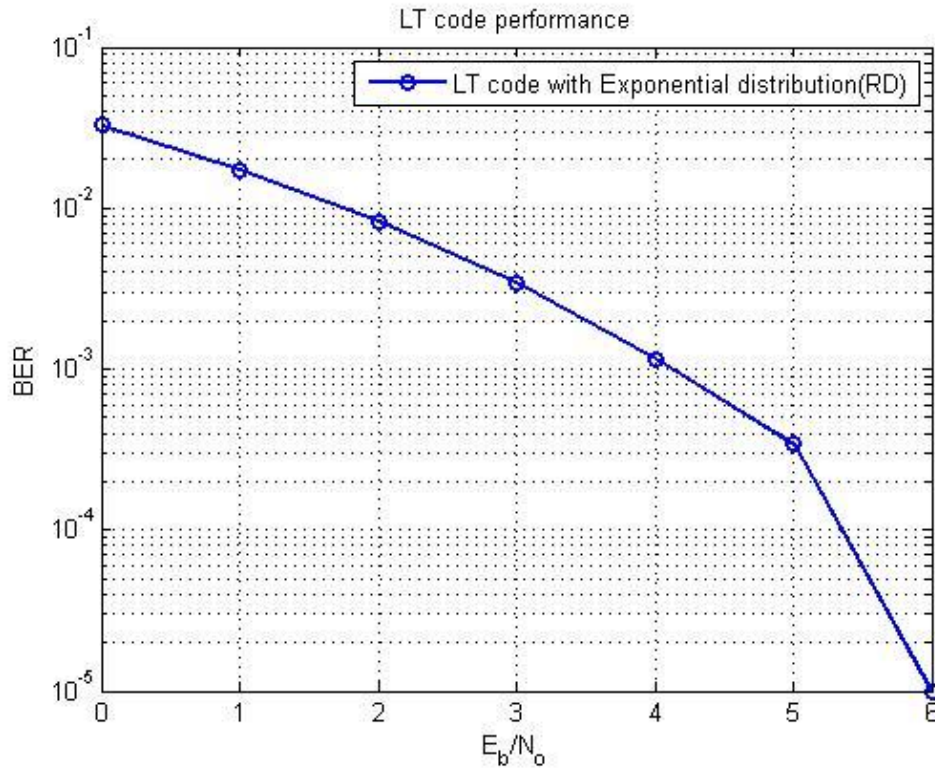
This degree relies on two limitations,  $c$  and  $\delta$ , as explained in Chapter 2. In this modeling, we take  $c=0.1$  and  $\delta=0.05$ . Fig. 17 shows the performance of the LT code adapting with robust soliton distribution and a signal-to-noise ratio (SNR) varying from 0 to 7 dB. We send 100 frames per SNR and each frame contains upwards of 1400 bits accepted with BPSK alteration and is transmitted over an AWGN channel.

#### **3.3.2. LT code with spars degree distribution**

This degree is used to optimize LT code performance. The sparse degree is adapted to the LT code Fig. 18 and we can observe the performance of this degree. It reaches  $10^{-5}$  in signal-to-noise ratio (SNR) equal to 7 dB. This modeling is implemented over an AWGN channel and a BPSK modulation is used in this modeling. This degree is based on the selection tags from all degrees and this degree is based o on Eq. (2.45)

#### **3.3.3. LT code with exponential degree**

A dependable degree is accepted to LT code in Fig. 15 and we can analyse the process of this degree. It reached  $10^{-5}$  in signal-to-noise ratio (SNR) which is equal to 6 dB. This imitation is applied over an AWGN channel and a BPSK modulation is utilized.

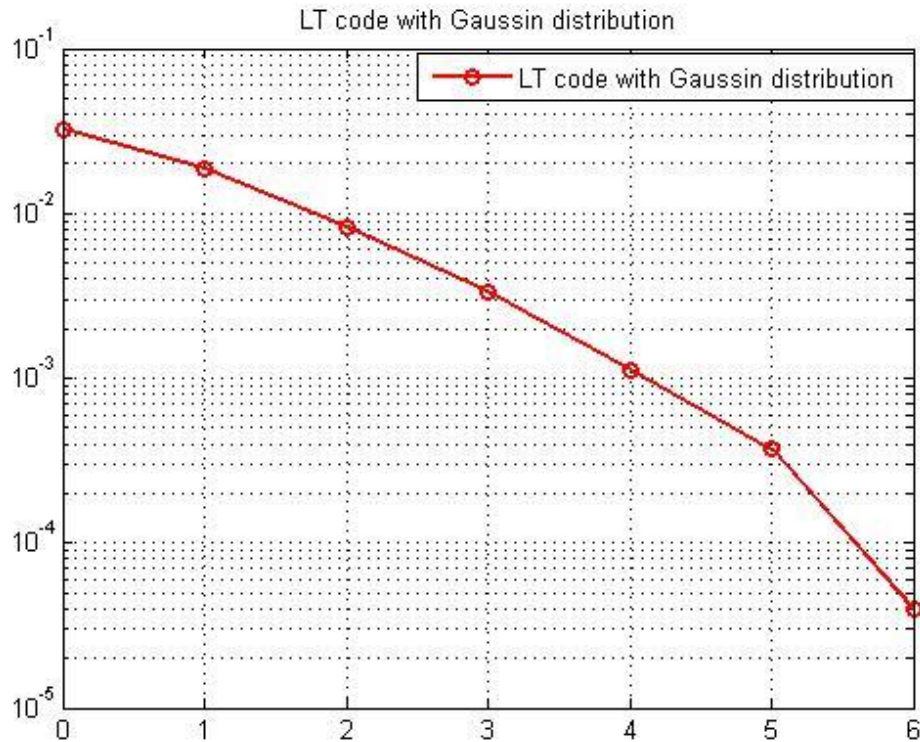


**Figure 15:** LT with exponential degree

### 3.3.4. LT code with gaussian degree distribution

Gaussian degree adapts to LT code in Fig. 16 and we can assume the behavior of this degree. It reaches  $10^{-5}$  in signal-to-noise ratio (SNR) equaling 6 dB. This imitation is used over the AWGN channel and a BPSK inflection is applied.

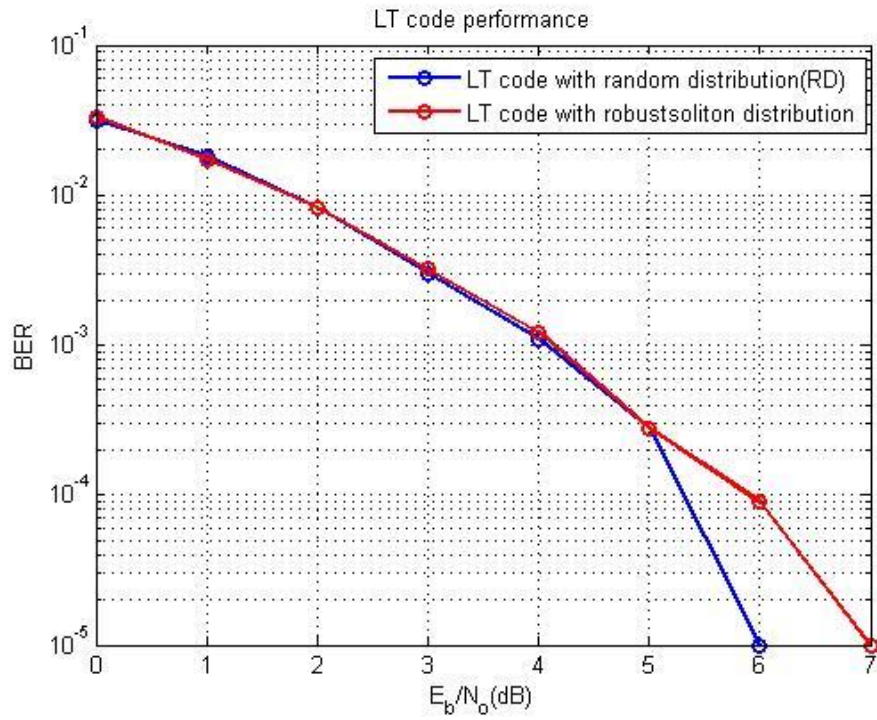




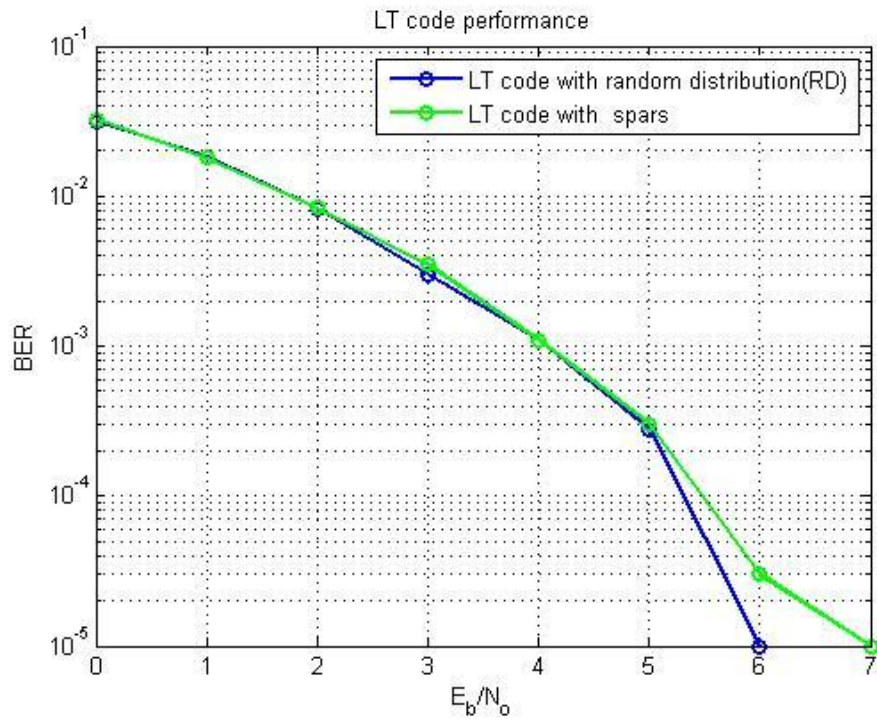
**Figure 16:** LT code with Gaussian degree distribution

### 3.3.5. Comparative between normal and new degree

Random degree is accomplished to LT code Fig. 17 where it is assumed that there is a difference between the arbitrary degree distribution effects and the robust degree distribution effects on LT code performance. The LT code with arbitrary degree distribution reaches the value  $10^{-5}$  dB in signal-to-noise ratio (SNR) equalling 6 dB and the LT code with the robust soliton distribution reaching the same value in SNR which is 7 dB.



**Figure 17:** LT code with random degree and robust soliton distribution



**Figure 18:** LT code with random degree and spars degree distribution

Fig. 18 demonstrates the difference between the random degree distribution impact and sparse degree distribution impact on the LT code presentation. From the result, we can see the LT code with a random degree distribution approaching the value  $10^{-5}$  dB in signal to noise percentage (SNR) equal to 6 dB and LT code with a sparse degree distribution approach a similar value in the SNR equal to 7 dB.

## CHAPTER 4

### CONCLUSION AND FUTURE WORKS

#### 4.1. Conclusion

This thesis focused on noisy decoding of rateless code using hard and soft decision decoding methods. Rateless codes were originally invented for binary erasure channels. Rateless codes are used in many applications nowadays, particularly in broadcast and multicast services or in digital communication. Rateless codes are the appropriate choice because they ensure reliable data transmission. The decoder of rateless code can reconstruct the source packet without depending on a channel coefficient or the manner of this channel. In this thesis, the two rateless code models are illustrated in a general form for Raptor codes, LT codes and LDPC codes. We tested LDPC code with the effective linear time encoding method (Lower Triangular Modification Approach). Two types of decoding algorithm, namely classical belief propagation and logarithmic likelihood ratio, are adapted to these codes. We have an exhibit Raptor code with one stage of pre-code and use LDPC code as a pre-code. On the decoder side, two decoder techniques, namely the local iteration decoder and the global iteration decoder, are proposed for the raptor code. We proposed LT code with a different degree distribution. In this dissertation, new degrees of Luby Transform code distribution are developed after detailed studies with implementation in the AWGN channel. Furthermore, comparisons are made between these degree distribution effects on the performance of LT code over the AWGN channel in Chapter Three. LT code gave better performance when new degree distributions are adapted to it.

## **4.2. Future Works**

In this work, we use belief propagation to decode these codes and this method sometimes suffers from high error floors at high SNR. The MAP decision algorithm is another idea to recover data in the decoder of these codes. Yet another idea would be rateless convolutional code, which is block code and its implementation over AWGN channels.

## REFERENCES

1. **Shannon C. E., (1948)**, “*A Mathematical Theory of Communication*”, Bell System Technical Journal, vol. 27, pp. 623- 637.
2. **Wozencraft J., Reiffen B., (1961)**, “*Sequential Decoding*”, Massachusetts Institute of Technology Press, Cambridge, USA, pp. 130-138.
3. **Coffey J. T., Goodman R. M., (1990)**, “*Any Code of Which We Cannot Think Is Good* ”, IEEE Transactions on Information Theory, vol. 36, no. 6, pp. 1453-1461.
4. **Berrou C., Glavieux A., Thitimajshima P., (1993)**, “*Near Shannon Limit Error Correcting Coding and Decoding: Turbo-Codes*”, in Proceedings IEEE International Conference on Communications ICC, vol. 2, pp. 1064-1070.
5. **Berrou C., Glavieux A., Thitimajshima P., (1996)**, “*Near Optimum Error Correcting Coding and Decoding: Turbo-Codes*”, IEEE Transactions on Communications, vol. 44, no. 10, pp. 1261-1271.
6. **Burr A., (2001)**, “*Turbo-Codes: The Ultimate Error Control Codes*”, Electronics and Communication Engineering Journal, vol. 13, no. 4, pp. 155-165.
7. **Gracie K., Hamon M. H., (2007)**, “*Turbo and Turbo-Like Codes: Principles and Applications in Telecommunications*”, Proceedings of the IEEE, vol. 95, no. 6, pp. 1228-1254.
8. **Gallager R., (1963)**, “*Low-Density Parity-Check Codes*”, Massachusetts Institute of Technology Press, Cambridge, USA, pp. 168-195.

9. **MacKay D. J. C., Neal R. M., (1996)**, “*Near Shannon Limit Performance of Low Density Parity Check Codes*”, Electronics Letters, vol. 32, no. 18, pp. 1645-1646.
10. **MacKay D. J. C., (1997)**, “*Near Shannon Limit Performance of Low Density Parity Check Codes*”, Electronics Letters, vol. 33, no. 6, pp. 457-458.
11. **MacKay D. J. C., (1999)**, “*Good Error-Correcting Codes Based on Very Sparse Matrices*”, IEEE Transactions on Information Theory, vol. 45, no. 2, pp. 399-431.
12. **Wiberg N., Loeliger H. A., Kötter R., (1995)**, “*Codes and Iterative Decoding on General Graphs*”, Euro. Transactions Telecommunication, vol. 6, pp. 513-525.
13. **Wiberg N., (1996)**, “*Codes and Decoding on General Graphs*”, PHILOSOPHIAE DOCTOR, Department of Electrical Engineering, Linköping university, Sweden, pp. 30-50.
14. **Sipser M., Spielman D. A., (1994)**, “*Expander Codes*”, Department of Computer Science, IEEE 54<sup>th</sup> Annual Symposium on Foundations of Computer Science, pp. 566-576.
15. **Sipser M., Spielman D. A., (1996)**, “*Expander Codes*”, IEEE Transactions on Information Theory, vol. 42, no. 6, pp. 1710-1722.
16. **Richardson T., Urbanke R., (2008)**, “*Modern Coding Theory*”, Cambridge University Press, New York, pp. 314- 325.
17. **Chung S. Y., Forney J., Richardson T., Urbanke R., (2001)**, “*On The Design of Low Density Parity-Check Codes Within 0.0045 dB. of The Shannon Limit*”, IEEE Communications Letters, vol. 5, no. 2, pp. 58-60.
18. **Richardson T., Urbanke R., (2001)**, “*The Capacity of Low-Density Parity-Check Codes Under Message-Passing Decoding*”, IEEE Transactions on Information Theory, vol. 47, no. 2, pp. 599-618.

19. **Aji S. M., McEliece R. J., (2000)**, “*The Generalized Distributive Law*”, IEEE Transactions on Information Theory, vol. 46, no. 2, pp. 325-343.
20. **Tanner R., (1981)**, “*A Recursive Approach to Low Complexity Codes*”, IEEE Transactions on Information Theory, vol. 27, no. 5, pp. 533-547.
21. **Bishop C., (2006)**, “*Pattern Recognition and Machine Learning*”, 1<sup>st</sup> Edition, Springer, California, pp. 359-422.
22. **Byers J. W., Luby M., Mitzenmacher M., Rege A., (1998)**, “*A Digital Fountain Approach to Reliable Distribution of Bulk Data*”, Association for Computing Machinery Special Interest Groups Computer Communication, vol.28, no. 4, pp. 56-67.
23. **Mitzenmacher M., (2004)**, “*Digital Fountains: a Survey and Look Forward*”, IEEE Information Theory Workshop, pp. 271-276.
24. **Etesami O., Shokrollahi A., (2006)**, “*Raptor Codes on Binary Memoryless Symmetric Channels*”, IEEE Transactions Information Theory, vol. 52, no. 5, pp. 2033-2051.
25. **Tanner R. M., (1981)**, “*A Recursive Approach to Low Complexity Codes*”, Information Theory, IEEE Transactions on, vol. 27, no. 5, pp. 533-547.
26. **Kschischang, F. R., Frey B. J., Loeliger H. A., (2001)**, “*Factor Graphs and The Sum Product Algorithm Information Theory*”, IEEE Transactions, vol. 47, no. 2, pp. 498-519.
27. **Morelos-Zaragoza R. H., (2002)**, “*The Art of Error Correcting Coding*”, 1<sup>st</sup> Edition, Wiley, England, pp. 172-192.
28. **Gallager R., (1962)**, “*Low Density Parity-Check Codes*”, Institute of Radio Engineers Transactions Information Theory, pp. 21-28.
29. **MacKay D. J., Neal R. M., (1995)**, “*Good Codes Based on Very Sparse Matrices*”, In Cryptography and Coding, Springer Berlin Heidelberg, pp. 100-111.



30. **Alon N., Luby M., (2001)**, “*A Linear Time Erasure Resilient Code with Nearly Optimal Recovery*”, IEEE Transactions Information Theory, vol. 47, no. 2, pp. 638-656.
31. **Luby M., Mitzenmacher M., Shokrollahi A., (1998)**, “*Analysis of Random Processes Via and-or Tree Evaluation*”, Proceedings of 9th Annual Association for Computing Machinery-Society for Industrial and Applied Mathematics Symposium on Discrete Algorithms, vol. 98, pp. 364-373.
32. **Luby M., Mitzenmacher M., Shokrollahi A., Spielman D., Stemann V., (1997)**, “*Practical Loss Resilient Codes*”, Proceedings of 9th Annual Association for Computing Machinery Symposium on Theory of Computing, vol. 96, pp. 150-159.
33. **Luby M., Mitzenmacher M., Shokrollahi A., Spielman D., (2001)**, “*Efficient Erasure Correcting Codes*”, IEEE Trans. on Information Theory, Special Issue on Codes and Graphs and Iterative Algorithms, vol. 47, no. 2, pp. 569-584.
34. **Xiaojing F., Zhou W., (2009)**, “*Study on Belief Propagation Decoding Algorithm of LDPC Codes*”, Electronic Test, no.7, pp. 42-43.
35. **Robert H., Zaragoza M., (2006)**, “*The Art of Error Correcting Coding*”, Second Edition, John Wiley&Sons, pp. 204-206.
36. **Shokrollahi A., (2006)**, “*Raptor Codes*”, IEEE Transactions Information Theory, vol. 52, no. 6, pp. 2551–2567.
37. **MacKay D. J. C., (2005)**, “*Fountain Codes*”, in Proceedings of IEEE Communications, vol. 152, no. 6, pp. 1062-1068.
38. **Sivasubramanian B. Leib H., (2008)**, “*Fixed-Rate Raptor Codes Over Rician Fading Channels*”, IEEE Transaction Vehicular Technology, vol. 57, no. 6, pp. 3905-3911.
39. **McEliece R. J., MacKay D. J. C., Cheng J. F., (1998)**, “*Turbo Decoding as an Instance of Pearl's Belief Propagation Algorithm*”, IEEE Journal on Selected Areas in Communications, vol. 16, no. 2, pp. 140-152.

40. **Hu X. Y., Eleftheriou E., Arnold D. M., Dholakia A., (2001)**, “*Efficient Implementations of The Sum-Product Algorithm for Decoding LDPC Codes*”, Proceedings IEEE Global Telecommunications Conference, vol. 2001, pp. 1036–1036E.
  
41. **Sivasubramanian B., Leib H., (2007)**, “*Fixed-Rate Raptor Code Performance Over Correlated Rayleigh Fading Channels*”, Canadian Conference on Electrical and Computer Engineering, pp. 912-915.
  
42. **Huang W., Li H., Dill J., (2010)**, “*Digital Fountain Codes System Model and Performance Over AWGN and Rayleigh Fading Channels*”, In Proceedings International Conference on Computing, Communications and Control Technologies, pp. 1-2.

## APPENDICES A

### CURRICULUM VITAE

#### PERSONAL INFORMATION

**Surname, Name:** KADHIM, Omar Raad Kadhim

**Date and Place of Birth:** 30 March 1987, Baghdad

**Phone:** +90 5312 460 774

**Email:** [c1282557@student.cankaya.edu.tr](mailto:c1282557@student.cankaya.edu.tr)



#### EDUCATION

Degree	Institution	Year of Graduation
M.Sc.	Çankaya Univ., Electronics and Communication Engineering	2014
B.Sc.	Baghdad Univ., Electronics and Communication Engineering	2008
High School	Al Markzia School	2004

#### WORK EXPERIENCE

Year	Place	Enrollment
2009- Present	Ministry of Higher Education. Al- Iraqi Univ.	Specialist

#### FOREIN LANGUAGES

Advanced English, Beginner French, Beginner Turkish

## **PUBLICATIONS**

1. Kadhim O., "*Degree Distribution Effects on Rateless Codes and Performance of These Codes in AWGN Channel*", Progress in URSI Electronic and Communication Research, Not Printed, Firat University, Turkey, (2014).

## **HOBBIES**

Football, Travel, Video Games