

This is to certify that the
dissertation entitled


THE MAXIMUM-LIKELIHOOD DECODING ALGORITHMS OF
LOW-DENSITY CODES OVER BINARY ERASURE
CHANNELS

presented by

KI-MOON LEE

has been accepted towards fulfillment
of the requirements for the

PH.D. degree in MATHEMATICS


Major Professor's Signature

8/16/2007

Date

PLACE IN RETURN BOX to remove this checkout from your record.
TO AVOID FINES return on or before date due.
MAY BE RECALLED with earlier due date if requested.

DATE DUE	DATE DUE	DATE DUE

**THE MAXIMUM-LIKELIHOOD DECODING
ALGORITHMS OF LOW-DENSITY CODES OVER
BINARY ERASURE CHANNELS**

By

KI-MOON LEE

A DISSERTATION

Submitted to
MICHIGAN STATE UNIVERSITY
in partial fulfillment of the requirements
for the degree of

DOCTOR OF PHILOSOPHY

DEPARTMENT OF MATHEMATICS

2007

ABSTRACT

THE MAXIMUM-LIKELIHOOD DECODING ALGORITHMS OF LOW-DENSITY CODES OVER BINARY ERASURE CHANNELS

By

KI-MOON LEE

We develop an advanced form of the Maximum Likelihood Decoding Algorithm (MLDA) for Low-Density Parity-Check (LDPC) codes and Luby Transform (LT) codes called the Separated MLDA (S-MLDA). We then present our design of LT degree distributions by supplementing the Robust Soliton Distribution with small fractions of dense rows that is optimized for the S-MLDA based decoding of LT codes. Simulation results which show the viability of the proposed MLDA of LDPC and LT codes are also presented. We also substantiate by extensive experimental results that, under the S-MLDA, LT codes from an arranged encoder matrix can achieve performance in stable overhead γ for the successful S-MLDA close to 0, while the S-MLDA maintains the computational complexity in number of symbol additions less than few tens of block lengths n .

To my parents, Si-Hyuk Lee and Kyung-Im Bong, to my wife Jin-Hee
Gil, and to my daughter Lynn

ACKNOWLEDGMENTS

First of all, I take this opportunity to express my profound gratitude to my parents Lee and Bong, my wife Jin-Hee Gil, and my younger brother Ki-Nam Lee for their moral support and patience during my study in Michigan State University.

I would like to express my deepest sense of gratitude to my adviser Dr. Hayder Radha for his patient guidance, encouragement and excellent advice throughout this study. I have to say that, with the Markov Process I learned from his *Random Process* class, I was able to establish and finalize the main theorems of this thesis, particularly the design of LT degree-distributions by analyzing the MPA as a time- and degree-varying Markov Process on a random matrix H over \mathbb{F}_2 .

I would like to express my gratitude to my co-adviser Dr. Jonathan I. Hall who taught me Algebraic Coding Theory and Low-Density Codes over Binary Erasure Channels. From his coding theory class and the independent study for Low-Density codes, I was able to set up the essential problems in both developing decoding algorithms for the S-MLDA and generalizing degree-distributions of Low-Density codes for the RSD.

My sincere thanks to Dr. Nikolai Ivanov for his excellent teaching in both Combinatorics and Graph Theory classes. I should say that all combinatorial arguments in my thesis, particularly the rank-distribution theorems in chapter 2 related with Kovalenko's Rank Distribution Theorem, come from his Combinatorics and Graph Theory class.

I am thankful to Dr. Ulrich Meierfrankenfeld for his excellent lectures in Algebra I and II. I still remember the moment when he addressed the role of changing bases and its matrix representations over Galois extension fields and modules. I should say that the essential part of the algorithm design for the S-MLDA and the MPA in the thesis was contributed by the basis changes and their matrix representations.

I am thankful to Dr. Gerald D. Ludden for being my committee member and his recommendation for teaching reference.

I also express my special thanks to my friends in Korea, Hyung-Seok No, Young-Sam Lee, and Young-Seok Hong for their friendships and financial support during my Ph.D. in Michigan State University.

Finally, I acknowledge my gratitude to my colleagues Kiran Misra and Shirish Karandes, and Dr. Wook-Joong Kim from ETRI Korea for their fruitful collaboration and valuable advises in the research and software design.

TABLE OF CONTENTS

LIST OF FIGURES	vii
LIST OF ALGORITHMS	viii
LIST OF TABLES	ix
1 Introduction	1
1.1 Overview of the Thesis	2
1.2 Gaussian Elimination on a Linear System over \mathbb{F}_2	11
1.3 Low-Density Parity-Check Codes	20
1.3.1 Encoding and Decoding Algorithm of LDPC Codes	20
1.3.2 Probability Density Evolutions on Degree Ensembles	26
1.4 Luby Transform Codes over BEC	30
1.4.1 Encoding and Decoding Algorithms of LT codes	31
1.4.2 The Robust Soliton Degree Distribution	33
2 The Maximum-Likelihood Decoding Algorithms of LT Codes	39
2.1 Introduction and Backgrounds	39
2.2 The Separated MLDA	45
2.3 Computational Complexities of the MLDA	50
2.4 Degree Distribution Design with Dense Rows	52
2.5 The Rank Distributions of \tilde{H}	58
2.6 Simulation Results	63
2.7 LT Codes of Short Block Lengths From an Arranged Encoder Matrix	70
2.8 Conclusions	78
3 The Maximum-Likelihood Decoding Algorithms of LDPC Codes	79
3.1 Introduction and Backgrounds	79
3.2 The S-MLDA Design with LDPC Codes	83
3.3 The Complexity of the MLDA	90
3.4 Simulations	93
3.5 Conclusions	99
INDEX	100
BIBLIOGRAPHY	102

LIST OF FIGURES

1.1	Date Transmission over Binary Erasure Channels	3
2.1	The MLDA on \tilde{H}	41
2.2	Performances of LT Codes in Decoding Failure Rates (DFR) (under the S-MLDA (black curves 1) and the MPA (gray curves 2). The codes are generated by the $\rho(x)$ in Table 2.1 over the block lengths from $n = 1,000$ to $10,000$	65
2.3	Number of Symbol Additions by the Post-Decoding and the MLDA in [3]	66
2.4	Fraction of References	68
2.5	The Rank-Deficiency $\eta = \dim(\text{Ker}(H))$ (or the Number of Free Variables)	69
2.6	Rank Deficiency (top figure) and DFR (bottom figure) of LT codes of $n = 5,000$, generated by $\rho(x)$ and $\rho_{\mathcal{D}_1 \cup \mathcal{D}_2}$	70
2.7	Performances in Decoding Error Rates under the S-MLDA and the MPA	74
2.8	Number of Symbol Additions made by the post-decoding and the original MLDA based on the encoding scheme E1	75
2.9	Fraction of References based on the Encoding Scheme E1	76
2.10	Number of Free Variables based on the Encoding Scheme E1	77
3.1	Decoding Failure Rate of LDPC codes under the S-MLDA (black curves) and the MPA (gray curves) for block lengths $2,000 \leq n \leq 20,000$	94
3.2	Number of Symbol Additions made by the Post-Decoding and the original MLDA	96
3.3	Number of References	97
3.4	Number of Free Variables by the S-MLDA	98

LIST OF ALGORITHMS

1.1	The LU -factorization on H	16
1.2	Recovery of α by the column-wise FS and BS	18
1.3	Recovery of α by the row-wise FS and BS	19
1.4	The Message Passing Algorithm on M	24
1.5	The recovery of $QX_{\mathcal{R}}$ by the FS in Algorithm 1.3	25
3.1	The ALTA on M	84
3.2	The BSR on $M_{\mathcal{R}}$ by $\prod_{k=l}^1 \bar{S}^{(k)} \cdot M_{\mathcal{R}}$	86
3.3	The GE on $\bar{M}_{\mathcal{R}}$ with \bar{T} and \mathcal{R}	87
3.4	The computation of $\bar{U}^{-1} \bar{L}^{-1} \bar{S}^{(k)} \cdot \beta^T$	89
3.5	The recovery of $X_{\bar{\mathcal{R}}}$ by FFS	90
3.6	The overall S-MLDA	90

LIST OF TABLES

2.1 The row-degree distribution $\rho(x)$ 64

CHAPTER 1

Introduction

In the advent of the Internet, data transmission over the (TCP/IP based) Internet becomes a part of our daily lives. Over the Internet, data transmission between a sender and a receiver is accomplished in a form of packet transmission in the following manner. For a given packet, a sender transmits the packet repeatedly until a receiver acknowledges the arrival of the packet. In this transmission scheme, there is no “decoding success/failure”, because every lost packet is retransmitted by a sender till a receiver acquires the packet. Therefore, this retransmission scheme is 100% *reliable*. This acknowledgement (ACK) based retransmission scheme, however, feasibly causes heavy network congestion, that could lead to explosively many retransmission requests if a large number of dropped (or lost) packets occur. This is particularly true for multicast services where ACK-based communication is virtually impossible to support (see [5,6,10,11,14] and references therein).

Let us now consider the following data transmission scheme. For a given binary information data set I , a sender subdivides it into k packets first. It then transforms the k packets into n packets with $n > k$, and transmits packets till a receiver provides a feedback message that informs the sender to stop the transmission. A receiver then recovers the original k packets with randomly received $(1 + \epsilon)k$ packets (out of n possible packets) for some $\epsilon > 0$. If there exists an efficient way for the recovery of the original k packets from the $(1 + \epsilon)k$ received packets, particularly with ϵ close to

0, then this transmission scheme may reduce the number of retransmission requests significantly.

Then the question is how to recover the n packets. Many advanced solutions have been developed to optimally address this question. This includes the Forward Error Correction (FEC) schemes that is based on Reed-Solomon codes as proposed by Rizzo et al. in [10, 11] and the FEC scheme based on LT and Raptor codes proposed by Digital Fountain co. [2, 5, 6].

1.1 Overview of the Thesis

In this thesis, rather than using polynomial based coding techniques such as the Reed-Solomon codes in [10, 11], we approach the above issue using linear algebra on vector spaces over \mathbb{F}_2 . We will define several terminologies shortly. We then continue the issue of transformations that take place at the sender and the receiver. Let α_I be a representation of a given data set I that consists of k packets of equal size s , i.e., $\alpha_I = (\alpha_1, \dots, \alpha_i, \dots, \alpha_k)$, where $\alpha_i \in \mathbb{F}_2^s$ for $i = 1, \dots, k$. Let us call a packet α_i as a *symbol*. Let $\alpha \in (\mathbb{F}_2^s)^n$ be the transformed vector from α_I . We refer to the transformer and α_I as the *encoder* and the *information symbol vector*, respectively, and we call the α as the *codeword*. Now let Z denote a received symbol vector that consists of m received symbols with $m > k$ at the receiver side. Suppose α can be recovered from Z by a certain (inverse) transformer of the encoder. We call a transformer of the receiver as a *decoder*. In the remainder of the thesis, we assume that a received vector Z is always a sub-vector of a codeword α . If a symbol α_i is not arrived at the receiver, we refer to the symbol as an *erasure*. Considering that a symbol is a binary vector in \mathbb{F}_2^s , we call the overall routes (or paths) between a sender and a receiver a *Binary Erasure Channel* (BEC). This data transmission scheme can be depicted as the diagram in **Figure 1.1**.

Let us now consider two types of encoding methods that use linear transformations

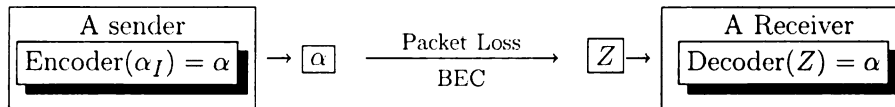


Figure 1.1. Data Transmission over Binary Erasure Channels

over \mathbb{F}_2 . By doing so, we turn the task of a decoder into a problem of solving a consistent linear system over \mathbb{F}_2^s . The first type is as follows. At the encoder side, we first fix k , the number of symbols of α_I , so that any given information data set I is represented as a symbol vector α_I in $(\mathbb{F}_2^s)^k$. We then transform α_I into a longer symbol vector $\alpha = (\alpha_1, \dots, \alpha_n)$ in the kernel space

$$\text{Ker}(H) = \{V \in (\mathbb{F}_2^s)^n | H \cdot V^T = 0\}, \quad (1.1.1)$$

where H is an $m \times n$ matrix over \mathbb{F}_2 with $\text{Rank}(H) = m$, $m = n - k$, and V^T is the symbol-wise transpose of V . Let $G = [S_{m \times k}; I_{m \times m}]$ be row-equivalent to H which is obtainable by Gaussian Elimination (GE) on H . Let us now consider $\text{En}(G) = \begin{bmatrix} I_{k \times k} \\ S_{m \times k} \end{bmatrix}$, the $n \times k$ induced matrix from G . Then for any $\alpha_I \in (\mathbb{F}_2^s)^k$, α_I is transformed to a kernel vector α such that

$$\alpha^T = \text{En}(G)\alpha_I^T = (\alpha_I, \alpha_P)^T, \quad \alpha_P^T = S_{m \times k}\alpha_I^T, \quad (1.1.2)$$

where α^T is the symbol-wise transpose of α . It is not hard to see that $\text{En}(G)$ is a (vector space) isomorphism between $(\mathbb{F}_2^s)^k$ and $\text{Ker}(H)$. Therefore, in this case, an encoder is simply the isomorphism $\text{En}(G)$ and a codeword is a symbol vector α in $\text{Ker}(H)$. Notice that, for any $\alpha_I \in (\mathbb{F}_2^s)^k$, $H \cdot \alpha^T = 0$. For a given codeword α , let us now suppose that a receiver acquires $n_{\bar{e}}$ symbols of α at random with $n_{\bar{e}} \geq (1 + \epsilon)k$, and denote the received symbol vector as $\alpha_{\bar{e}}$. By rearranging symbols of α , we may express α into a form $(\alpha_{\bar{e}}, X)$, where X represents the lost symbol vector. Then by rearranging columns of H associated with the expression $(\alpha_{\bar{e}}, X)$, we may also express H into a form $[N; M]$, where N and M consists of columns of H associated

with symbols of $\alpha_{\bar{e}}$ and X , respectively. With the expressions, the kernel space constraint $H\alpha^T = 0$ in (1.1.1) is expressed as $N\alpha_{\bar{e}}^T + MX^T = 0$, and thus,

$$MX^T = \beta^T, \quad \text{where} \quad \beta^T = N\alpha_{\bar{e}}^T. \quad (1.1.3)$$

Therefore, in this encoding scheme, the task of a decoder is in solving a consistent linear system (1.1.3) for its unique solution, say the unique solution by $X = \alpha_e$. Once α_e is obtained, then an information symbol vector α_I can be retrieved from $(\alpha_{\bar{e}}, \alpha_e)$. It should be emphasized that, for the unique solution of the system, the number of columns of M should be less than or equal to the number of rows m . In other words, the number of received symbols $n_{\bar{e}}$ must be greater than or equal to $n - k$. We refer to $\text{Ker}(H)$ as the *Parity-Check* code over \mathbb{F}_2^s . If H has relatively few 1's, or say H is sparse, then we refer to $\text{Ker}(H)$ as the Low-Density Parity-Check (LDPC) code generated by H .

The second type of encoding is as following. We first consider the following transmission scheme over BEC. For a given information symbol vector α_I in $(\mathbb{F}_2^s)^k$, an encoder directly sets a codeword α as $\alpha := \alpha_I$ so that $n = k$. It then constantly generates row vectors $H_i \in \mathbb{F}_2^n$ in random by following a certain rule, and at the same time, it generates a symbol β_i by $\beta_i := H_i\alpha_I^T$ and transmits it over BEC. The transmission continues in this fashion, till a receiver acquires enough number of such β_i 's. Suppose that a receiver acquires more than $(1 + \gamma)n$ symbols in random, say the acquired symbol vector as $\beta = (\beta_1, \dots, \beta_m)$. Then at a receiver end, with each acquired β_i , a decoder generates the associated check row H_i , $1 \leq i \leq m$, and sets up the linear system

$$HX^T = \beta^T, \quad \beta \in (\mathbb{F}_2^s)^m. \quad (1.1.4)$$

where H is now an $m \times n$ matrix over \mathbb{F}_2 that consists of rows H_i 's such that $H_iX^T = \beta_i$. In this transmission scheme, the task of the decoder is also in solving the consistent linear system (1.1.4) for its unique solution $X = \alpha$. For a given $(0, 1)$ -vector $V \in \mathbb{F}_2^n$,

let $|V|$ denote the number of 1's of V and refer to as the *degree* of V . If an encoder uses a certain probability distribution function, say $\mu(x) = \sum \mu_d x^d$, for the degree of H_i by $\Pr(|H_i| = d) = \mu_d$, then we refer to this transmission scheme as Luby Transform (LT) transmission, and refer to the set of pairs $\{(H, \beta)\}$ as the LT code generated by $\mu(x)$. Note that each pair (H, β) corresponds to a consistent linear system $HX^T = \beta^T$.

We now impose two fundamental questions on the systems (1.1.3) and (1.1.4):

Q1) With how many received symbols of Z can the decoders solve the systems uniquely?

Q2) At the same time, how efficiently can they solve the systems?

Let M in system (1.1.3) be an $m \times n_e$ random matrix that consists of columns of H . Likewise, let H in system (1.1.4) be an $m \times n$ random matrix generated by a probability distribution $\mu(x)$. What we want to do in Q1) is to *maximize* n_e the column dimension of M and is to *minimize* m the row dimension of H , while maintaining $\text{Rank}(M) = n_e$ and $\text{Rank}(H) = n$. With Q2), at the same time, we also want to solve the systems in the fastest way as possible. Straightforwardly, both Q1) and Q2) are the problem of designing the check matrix H in (1.1.1) for LDPC codes and the distribution $\mu(x)$ for LT codes from which

1. a randomly chosen M and H (in (1.1.4)) has its full column rank with the largest number of columns and with the least number of rows as possible, respectively;
2. at the same time, a check matrix H in (1.1.1) and a random H (1.1.4) are as sparse as possible.

In the thesis, we exploit LDPC codes for system (1.1.3) and LT codes for system (1.1.4). With the codes, we develop an efficient decoding algorithm that can solve the systems as long as they have their unique solution. Let us call LDPC and

LT codes together as *Low-Density* codes. Known so far, LDPC and LT codes are generally considered as the best answer to the questions Q1) and Q2). The reason behind this is in the fact that, for a large n the column dimension of H in (1.1.1) and (1.1.4), if a check matrix H and $\mu(x)$ is designed well then a random M and H in (1.1.3) and (1.1.4), respectively, can be lower triangulated by a simple row and column permutation, called the Message Passing Algorithm (MPA) [3–6], and thus, the solution of the systems can be solved by a simple Forward Substitution (FS) [23, 24] over a triangulated matrix very efficiently. It is also possible to design an LDPC check matrix H and an LT degree-distribution $\mu(x)$ with the log-density condition such that an H in both (1.1.1) and (1.1.4) meets the log-density constraint $|H| \leq cn \ln(n)$ for some constant $c > 0$, where $|H|$ indicates the number of 1's of H .

LDPC codes were pioneered by Gallager [1] at 1969, and they were originally designed to protect data transmission against Binary Gaussian Noisy Channels and Binary Symmetric Channels. The codes were widely forgotten for the decades and rediscovered by Mackay and Neal in [8] at 1995. LDPC codes were used for BEC based transmission scheme for the first time in tornado codes by Luby et al in [5, 6] with the MPA. Soon later, Shokrollahi et al developed LDPC codes over BEC as capacity approaching codes optimized with the MPA for large block lengths n [7, 12]. Briefly speaking, a capacity approaching code is a code such that, when a check matrix H in (1.1.1) is generated by a certain row and column degree distribution, say a capacity approaching sequence, then system (1.1.3) can be solved by the MPA with rate $p = \frac{n_e}{n}$, referred to as loss rate or erasure rate, close to $\frac{m}{n}$ (or the block-rate $\frac{n_e}{m}$ of M close to 1). Further analysis for capacity approaching sequences can be found at [4, 9, 13].

LT codes were invented by Luby [2] and were designed for multi-cast of mass data. At a sender side, an LT encoder constantly generates symbols by $\beta_i = H_i \alpha_I^T$, where each check row H_i is randomly generated by a degree distribution $\mu(x)$, such

as the Robust Soliton Distribution (RSD) in [2]. At a receiver end, assuming that H in system (1.1.4) follows the RSD in its row-degree distribution and the number of received symbols m is greater than $(1 + \gamma)n$ for some $\gamma > 0$, the system can be solved uniquely by the MPA with an overhead γ close to 0. With this feature, LT codes were classified as *optimal* codes for multi-cast and broadcast. Shokrollahi generalizes the codes into Raptor codes by employing pre-coding strategy on α with LDPC codes or other known codes in prior to LT encoding. The decoding algorithm of Raptor code is a combination of the MPA and GE [14, 15].

In practice, however, those capacity approaching and optimal features are not guaranteed when n is not large enough, say n within several thousands. Through our extensive simulations with the MPA, we (the author KiMoon Lee and Hayder Radha) observed that a stable erasure rate $\frac{n_e}{n}$ of LDPC codes and a stable overhead γ of LT codes for the successful MPA are far away from their ideal limits $1 - \frac{m}{n}$ and 0, respectively. Instead, we observed that, by the Approximate Lower Triangulation Algorithm (ALTA) in [3, 4], a random M of an LDPC code system (1.1.3) or a random H in LT code system (1.1.4) can be approximate lower triangulated into a form PMQ^T (or PHQ^T) = $\begin{bmatrix} A & B \\ C & D \end{bmatrix}$, where B is an $l \times l$ lower triangular matrix with l close to a column dimension n_e (or n) and (P, Q) is a pair of row and column permutation of M (or H) (see **Figure 2.4** and **Figure 3.3**). Once such a triangulation is obtained, the systems (1.1.3) and (1.1.4) can be permuted to

$$\begin{bmatrix} A & B \\ C & D \end{bmatrix} QX^T = P\beta^T \xleftrightarrow{(P, Q)} \begin{cases} MX^T = \beta^T & (1.1.3) \\ HX^T = \beta^T & (1.1.4) \end{cases}. \quad (1.1.5)$$

Assuming that the permuted system (the left-hand side in 1.1.5) has its unique solution, it can be solved efficiently by the Maximum-Likelihood Decoding Algorithm (MLDA), developed by Burshtein and Miller in [3] for decoding of LDPC codes. We tested the MLDA with $\frac{1}{2}$ -rate PEG-LDPC codes [13], and the result was quite

surprising. A random $m \times n_e$ matrix M in system (1.1.3) feasibly has its full column rank with the loss-rate $p = \frac{n_e}{n}$ very close to $1 - \frac{m}{n}$. For an example, with n a few thousands and n_e close to $m - 20$, a random M has its full rank n_e with high probability (see **Figure 3.4**).

Although the MLDA is much more efficient than a conventional GE in computational complexities, it still has a lot of redundant computations. First, the MLDA in [3] requires an explicit construction of the permuted $PMQ^T = \begin{bmatrix} A & B \\ C & D \end{bmatrix}$ after the ALTA that is not necessary for solving systems (1.1.3) and (1.1.4). To remove the explicit construction, we interpreted systems of the MLDA into a equivalent set of systems that do not require the construction of PMQ^T . Second, it also results in a large number of redundant symbol-additions on β . To remove all possible redundant additions, we further developed the MLDA into the Separated MLDA (S-MLDA) by exploiting the MLDA in two steps: the *pre-decoding* on M in a bit-level and then the *post-decoding* in a symbol-level with β . Shortly speaking, in the *pre-decoding* step, the S-MLDA computes all the row operations with M (or H) alone that needs for recovery of the solution $X = \alpha_e$, it then discards all redundant equations in $MX^T = \beta^T$. In the *post-decoding* step, the algorithm computes the solution by applying the obtained operations on β with an alternative recovery step. To see the improvement in computational efficiency in number of symbol additions, compare the curves in **Figure 3.2** for LDPC codes and **Figure 2.3** for LT codes. The S-MLDA with simulation results tested with $\frac{1}{2}$ -rate PEG-LDPC codes [13] was presented in [18, CISS 2007].

We also applied the S-MLDA to the system (1.1.4) for decoding of LT codes. At the very first simulation, the obtained decoding failure rate of the S-MLDA with a random H , generated by the RSD, was significantly less than that of the MPA. As γ decreases, however, the S-MLDA feasibly fails to recover the unique solution α of system (1.1.4) due to the rank deficiency of H , i.e., $\eta = \dim(\text{Ker}(H)) > 0$. On the other hand, with n several thousands, η was less than 15 for any $\gamma > 0$. To remove

those small deficiencies, we re-designed the RSD $\mu(x)$ and altered into a $\rho(x)$ by supplementing a small fraction of dense rows, so that a random H by a $\rho(x)$ may include a few tens of dense rows. By doing so, we gracefully removed the deficiencies with overheads γ less than 0.008. (To see the simulation results, compare the curves in **Figure 2.6**). Besides, we develop several combinatorial analysis for the design of RSD $\mu(x)$ and the $\rho(x)$ in section 2.4. In section 2.5, we also develop a finite version of Kovalenko's Rank-Distribution Theorem in [21, 22, 29] for the rank distribution of a random H , generated by the $\rho(x)$. The S-MLDA tested LT codes generated by our designed $\rho(x)$ was presented at [19, ISIT 2007].

Compare to LDPC codes, although LT codes naturally inherit rate-less feature from its random transmission scheme, the time-efficiency for both encoding and decoding of LT codes is far inferior to that of LDPC codes. The reason to this is in the facts that, in LDPC codes, a fixed check matrix H is used for every instance of $\alpha = (\alpha_I, \alpha_P)$, furthermore, H has no dense rows in general. In contrast, an LT decoder has to generate a random H by using the same random generator of an encoder for every instance of α . Otherwise, each check row H_i should be directly delivered to a decoder attached on β_i . To resolve this problem, we developed LT codes with an arranged encoder matrix M . Specifically speaking, the check matrix H in system (1.1.4) is a random sub-matrix that consists of rows of M . With this arranged encoding scheme, we tested the S-MLDA with LT codes for short block-lengths n from 10^2 to 10^3 . Our experimental results exhibited that, although a stable overhead γ for the successful MPA is degraded seriously, a stable overhead γ for the successful S-MLDA with codes from M is slightly better than the one with codes, generated by the original LT encoding scheme. The time-efficiency of both encoding and decoding of LT codes in this arranged encoding scheme is about to be same with that of LDPC codes. The experimental result together with our combinatorial analysis for the $\mu(x)$ and $\rho(x)$ has been submitted to Allerton Conference 2007 [20].

The rest of the thesis is as follows. Chapter 1 is dedicated to the introductory backgrounds for the later chapters 1 and 2. In section 1.2, we describe GE as an LU -factorization algorithm [23]. In section 1.3, we introduce LDPC codes and the MPA, as a decoding algorithm of both LDPC and LT codes. The MPA is described as a lower triangulation algorithm by using row and column permutations on a check matrix. Then in section 1.4, we introduce row-degree distributions of LT codes.

Chapter 2 is as follows. In section 2.2, based on the factorization, we explain both the MLDA and S-MLDA as an advanced form of GE by exploiting partial pivoting process over the triangular block $\begin{bmatrix} B \\ D \end{bmatrix}$, as shown in (1.1.5). We first describe the MLDA as an natural extension of the MPA. We then develop the MLDA into the S-MLDA. In section 2.3, we present the computational complexity of the S-MLDA with LT codes in terms of the number of {sign, bit}-flips and the number of symbol additions made by the *pre-* and *post-decoding* stage of the S-MLDA, respectively. In section 2.4, we derive the RSD $\mu(x)$ by using the Mackay's recursive formula [27, ch. 50]. We then alter $\mu(x)$ into a $\rho(x)$ by supplementing a small fraction of dense rows, so that a randomly generated H by our designed $\rho(x)$ may fit for the S-MLDA. In section 2.5, we derive a finite version of Kovalenko's Rank-Distribution formula [21,22,29]. With the rank-distribution, we demonstrate the rank-distribution of a random H generated by our designed $\rho(x)$. Simulation results under the S-MLDA with LT codes, generated by a $\rho(x)$, are presented in section 2.6. In section 2.7, we further develop LT codes of short block lengths n , generated by an arranged encoder matrix, and present the simulation results tested with the LT codes under the S-MLDA. Finally, we conclude the chapter in section 2.8.

In chapter 3, we apply the S-MLDA demonstrated in section 2.2 for the decoding of LDPC codes. In section 3.2, we modify the S-MLDA in section 2.2 for decoding of LDPC codes. In section 3.3, we present the computational complexity of the S-MLDA with LDPC codes in terms of the number of {sign,bit}-flips and the number of

symbol additions made through the *pre*- and the *post-decoding* stage of the S-MLDA, respectively. In section 3.4, we use PEG software [13] to construct H of block lengths n from $n = 2,000$ to $20,000$. Simulation results tested with $\frac{1}{2}$ -rate PEG-LDPC codes under the S-MLDA are presented in this section. Finally, we conclude the chapter in section 3.5.

1.2 Gaussian Elimination on a Linear System over \mathbb{F}_2

In this section, we describe GE on H as an LU -factorization algorithm that returns $LU = PHQ^T$, where P and Q is a row and column permutation matrix of H and L and U is a lower and upper triangular matrix, respectively. GE is generally considered to be the most efficient computational method for solving a consistent linear system $HX^T = \beta^T$, since it involves the least amount of arithmetic operations. In particular, when GE is aimed to compute the unique solution of the system only, obtaining an LU factorization of H in the fastest way is the primary aim of GE algorithms. Further analysis and issues for GE can be found in linear algebra textbooks [23–26].

Let us first define a linear system over \mathbb{F}_2^s .

Definition 1.2.1 (A Linear System over \mathbb{F}_2^s). Let $H = (h_{ij})$ be an $m \times n$ matrix over \mathbb{F}_2 , and let $\beta = (\beta_1, \dots, \beta_m) \in (\mathbb{F}_2^s)^m$. A linear system $HX^T = \beta^T$ over \mathbb{F}_2^s is a system that consists of m linear equations over \mathbb{F}_2^s such that

$$\begin{pmatrix} \sum_{j=1}^n h_{1j}x_j & = & \beta_1, \\ \vdots & \vdots & \vdots \\ \sum_{j=1}^n h_{mj}x_j & = & \beta_m \end{pmatrix} \equiv HX^T = \beta^T, \quad (1.2.1)$$

where the sums are taken on \mathbb{F}_2^s and β^T is the symbol-wise transpose of β . Let us refer to β_i and β as a *syndrome symbol* and a *syndrome symbol vector* of system (1.2.1), respectively.

Throughout the thesis, unless specified, we assume that a given linear system

$HX^T = \beta^T$ has at least one solutions, and we refer to the system as a *consistent* linear system over \mathbb{F}_2^s . Let us denote $\text{Img}(H)$ as the image space of H in $(\mathbb{F}_2^s)^m$, i.e.,

$$\text{Img}(H) = \{HV^T \in (\mathbb{F}_2^s)^m \mid V \in (\mathbb{F}_2^s)^n\}. \quad (1.2.2)$$

Thus, a given system (1.2.1) is consistent iff $\beta \in \text{Img}(H)$. Considering that X^T and β^T can be expressed as an $n \times s$ matrix (x_{ij}) and an $m \times s$ matrix (β_{ij}) over \mathbb{F}_2 , where $x_i = (x_{i1}, \dots, x_{is})$ and $\beta_i = (\beta_{i1}, \dots, \beta_{is})$, respectively, system (1.2.1) can be arranged in a parallel form of s number of systems over \mathbb{F}_2 such that

$$HX^T = \beta^T \Leftrightarrow [Hx^1 = \beta^1, \dots, Hx^j = \beta^j, \dots, Hx^s = \beta^s], \quad (1.2.3)$$

where x^j and β^j now represents the j^{th} column of X^T and β^T , respectively. Obviously, solving system (1.2.1) is equivalent to solving one single system $Hx^j = \beta^j$ over \mathbb{F}_2 , and $Hx^j = \beta^j$ has its unique solution iff there exists n independent rows of H that form an $n \times n$ nonsingular sub-matrix, say H' . Once such H' and its inverse $(H')^{-1}$ are obtained, the unique solution of $HX^T = \beta^T$ can be obtained by applying the same $(H')^{-1}$ to the system, i.e., $X^T = (H')^{-1}\beta^T$. So, identifying such H' and then $(H')^{-1}$ (independently from β) is the primary task for solving system (1.2.1).

Let us clarify several terminologies for the description of GE.

Definition 1.2.2 (An Approximate Lower Triangular Matrix). Let $H = (h_{ij})$ be an $m \times n$ matrix over \mathbb{F}_2 with $m \geq n$.

1. *A Lower Triangular Matrix:* H is said to be in a *lower triangular* form, if

$$h_{ij} = \begin{cases} 1, & \text{for } i = j \\ 0, & \text{for } i < j \end{cases}. \quad (1.2.4)$$

2. *An Approximate Lower Triangular Matrix:* Let H be in a block matrix form $[A; B]$ such that B is an $l \times m$ lower triangular matrix with l close to n . We call H an *approximate lower triangular* matrix.

Definition 1.2.3 (Elementary Row Operation). We call a row operation on H over \mathbb{F}_2 by adding one row to other rows of H as an *elementary row operation*.

Let σ and τ be a permutation on the row index set $[m] = \{1, 2, \dots, m\}$ and a column index set $[n] = \{1, 2, \dots, n\}$ of H , respectively.

Definition 1.2.4 (A Row and Column Permutation of H by σ and τ). A row permutation of H by σ is the rearrangement of rows of H in the order of (i_1, \dots, i_m) , i.e., $H_\sigma = [H_{i_1}, \dots, H_{i_m}]^T$ (a row-wise transpose) where $\sigma(i_k) = k$. Similarly, a column permutation of H by τ is the rearrangement of columns of H in the order of (j_1, \dots, j_n) , i.e., $H^\tau = [H^{j_1}, \dots, H^{j_n}]$ where $\tau(j_k) = k$.

An elementary row operation and a permutation on H can be explained as matrix multiplication to H that has the same effect on H as does the operation and permutation. Let us first consider row and column permutations of H . Let $PH = [H_{i_1}, \dots, H_{i_m}]^T$ and $HQ^T = [H^{j_1}, \dots, H^{j_n}]$. Let P and Q be the matrix representation of σ and τ , respectively. Foremost, P and P^T is formed by permuting rows and columns of $I_{m \times m}$ in the order of σ , respectively, i.e., $P = [e_{i_1}, \dots, e_{i_m}]^T$ and $P^T = [e_{i_1}^T, \dots, e_{i_m}^T]$. By direct calculation, $P^{-1} = P^T$. Equivalently, they can be expressed as $P = (p_{st})$ and $P^T = (p'_{st})$, such that

$$p_{st} = \begin{cases} 1, & \text{if } t = i_s \\ 0 & \text{otherwise} \end{cases} \quad \text{and} \quad p'_{st} = \begin{cases} 1, & \text{if } s = i_t \\ 0 & \text{otherwise} \end{cases}. \quad (1.2.5)$$

In the same way, $Q = [e_{j_1}, \dots, e_{j_n}]^T$ and $Q^T = [e_{j_1}^T, \dots, e_{j_n}^T]$. Thus by a row and column permutation pair (P, Q) of H , system (1.2.1) can be permuted as

$$HX^T = \beta^T \Leftrightarrow (PHQ^T)(QX^T) = P\beta^T. \quad (1.2.6)$$

An elementary row operation on H can be also explained as matrix multiplication to H on the right. For a given row index k of H , let S_k be a subset of $[m]$ that

includes k , and let $\chi_{S_k} \in \mathbb{F}_2^m$ be the support vector of S_k such that

$$\chi_{S_k} = (\epsilon_1, \dots, \epsilon_{k-1}, 1, \epsilon_{k+1}, \dots, \epsilon_m) \quad \text{where } \epsilon_i = \begin{cases} 1, & \text{if } i \in S_k \\ 0 & \text{otherwise} \end{cases}. \quad (1.2.7)$$

When the k^{th} equation $H_k X^T = \beta_k$ is added to other equations $H_i X^T = \beta_i$, whose row index i is in $S_k \setminus k$, system (1.2.1) is transformed to

$$HX^T = \beta^T \Rightarrow \begin{cases} (H_1 + \epsilon_1 H_k) X^T & = & \beta_1 + \epsilon_1 \beta_k \\ \vdots & \vdots & \vdots \\ (H_{k-1} + \epsilon_{k-1} H_k) X^T & = & \beta_{k-1} + \epsilon_{k-1} \beta_k \\ H_k X^T & = & \beta_k \\ (H_{k+1} + \epsilon_{k+1} H_k) X^T & = & \beta_{k+1} + \epsilon_{k+1} \beta_k \\ \vdots & \vdots & \vdots \\ (H_m + \epsilon_m H_k) X^T & = & \beta_m + \epsilon_m \beta_k \end{cases}. \quad (1.2.8)$$

Let $E^{(k)}$ be the $m \times m$ matrix formed by replacing the k^{th} column of the identity matrix $I_{m \times m}$ with $\chi_{S_k}^T$, i.e.,

$$E^{(k)} = [e_1^T, \dots, e_{k-1}^T, \chi_{S_k}^T, e_{k+1}^T, \dots, e_m^T]. \quad (1.2.9)$$

Then system (1.2.8) is exactly same with the system

$$(E^{(k)} \cdot H) X^T = E^{(k)} \cdot \beta^T. \quad (1.2.10)$$

Noting that $E^{(k)} E^{(k)} H$ is simply the addition of H_k to the rows H_i twice for $i \in S_k \setminus k$, $E^{(k)} E^{(k)} H = H$. This is true for any H , therefore, $(E^{(k)})^{-1} = E^{(k)}$. Let us refer to $E^{(k)}$ as an *elementary matrix*. $E^{(k)}$ can be explained the representation of the identity map $I : \mathbb{F}_2^m \rightarrow \mathbb{F}_2^m$ with respect to, say, the standard basis $\mathcal{B} = \{e_1, \dots, e_m\}$ on the domain \mathbb{F}_2^m and the basis \mathcal{B}_k on the range \mathbb{F}_2^m of the identity map, where

$$\mathcal{B}_k = \{w_1 = (e_1 + \epsilon_1 e_k), \dots, w_k = e_k, \dots, w_m = (e_m + \epsilon_m e_k)\}. \quad (1.2.11)$$

Thus, the transformed system (1.2.10) is actually a re-expression of $HX^T = \beta^T$ by

changing the coordinate axis \mathcal{B} with \mathcal{B}_k of the range \mathbb{F}_2^m . In this interpretation, what GE does on H is a sort of changing bases on the range, till H is transformed into a lower or upper triangular matrix. Notice that, no operations were made on the domain of H , except column permutations. This implies that solutions of the system, up to rearrangements by column permutations, are not changed by any elementary row operations on H .

Let us now describe the LU -factorization algorithm on $HX^T = \beta^T$. We describe the algorithm based on the exemplary pseudo-codes in **Algorithm 1.1**. At each pivoting round k of the algorithm (see **while** loop in the algorithm), starting from $H(0) = H$, let $H(k) = (h_{ij}^{(k)})$ denote the transformed matrix by adding the k^{th} pivot row $(H(k-1))_{i_k}$ to the rows $(H(k-1))_i$, such that $i \in \bar{\mathcal{T}}$ and $h_{i,j_k}^{(k-1)} = 1$. Let $S_{i_k} = \{i \in \bar{\mathcal{T}} \mid h_{i,j_k}^{(k-1)} = 1\}$ (see the line 10 in the algorithm), and let $\bar{E}^{(k)}$ denote the elementary matrix obtained by replacing the i_k^{th} column of $I_{m \times m}$ with the support vector $\chi_{S_{i_k}}^T$ (see (1.2.7)), i.e.,

$$\bar{E}^{(k)} = [e_1^T, e_2^T, \dots, e_{i_k-1}^T, \boxed{\chi_{S_{i_k}}^T}, e_{i_k+1}^T, \dots, e_m^T]. \quad (1.2.12)$$

Thus, the lines 10-12 (see **foreach** loop) in the algorithm correspond to $H(k) = \bar{E}^{(k)}H(k-1)$. At the last round l , the returned $H(l)$ can be expressed as

$$H(l) = \bar{E}^{(l)} \dots \bar{E}^{(2)} \cdot \bar{E}^{(1)} H = \left(\prod_{k=l}^1 \bar{E}^{(k)} \right) H. \quad (1.2.13)$$

With the returned $\sigma_l = (i_1, \dots, i_l)$ and $\tau_l = (j_1, \dots, j_l)$ at the end (see line 14 of the algorithm), let

$$\{i_{l+1}, \dots, i_m\} = [m] \setminus \sigma_l, \quad \{j_{l+1}, \dots, j_n\} = [n] \setminus \tau_l. \quad (1.2.14)$$

Algorithm 1.1: The LU -factorization on H

```

1 Input:  $H$ ,  $[m]$    Output:  $H(l)$  and  $(\sigma_l, \tau_l)$ .
   /%<-- Initialization: -->%/
2 set  $\tilde{T} := [m]$ ;
3 foreach  $i \in [m]$  do
4   if  $H_i = 0$  then
5     discard  $\tilde{T} := \tilde{T} \setminus i$ ;

   /%<-- Pivoting Round  $k$ : -->%/
6 while  $\tilde{T} \neq \emptyset$  do
   /%<-- Pivot Selection: see Remark 1.2.2 -->%/
7   choose  $\exists i_k \in \tilde{T}$  such that  $h_{i_k, j_k}^{(k-1)} = 1$  in one's own way;
8   set  $\sigma_l(i_k) = k$  and  $\tau_l(j_k) = k$ ;
9   discard  $\tilde{T} := \tilde{T} \setminus i_k$ ;

   /%<-- Pivoting:  $H(k) := \bar{E}^{(k)} H(k-1)$  -->%/
10  foreach  $i \in \tilde{T}$  with which  $h_{i, j_k}^{(k-1)} = 1$  do
11    update  $H(k)_i := H(k-1)_i + H(k-1)_{i_k}$ ;
12    if  $H(k)_i = 0$  then
13      discard  $\tilde{T} := \tilde{T} \setminus i$ ; //<-- To discard null rows

   /%<-- Say the 'while' loop stops at  $k = l$  -->%/
14 return  $H(l)$  and  $(\sigma_l, \tau_l)$ ;

```

Then rearranging $[m]$ and $[n]$ into

$$\sigma = (\underbrace{i_1, \dots, i_l}_{\sigma_l}; \underbrace{i_{l+1}, \dots, i_m}_{[m] \setminus \sigma_l}) \quad \text{and} \quad \tau = (\underbrace{j_1, \dots, j_l}_{\tau_l}; \underbrace{j_{l+1}, \dots, j_n}_{[n] \setminus \tau_l}), \quad (1.2.15)$$

respectively, where $\sigma(i_k) = k$ and $\tau(j_k) = k$, let (P, Q) be the pair of permutation matrix of (σ, τ) . Then by $P^{-1} = P^T$, $PH(l)Q^T$ can be expressed as

$$PH(l)Q^T = \left(\prod_{k=l}^1 P \bar{E}^{(k)} P^T \right) (PHQ^T) = \begin{bmatrix} U_{l \times l} & A_{n-l \times n-l} \\ 0 & 0 \end{bmatrix}. \quad (1.2.16)$$

Now let $L^{(k)} = P \bar{E}^{(k)} P^T$, which is the elementary matrix formed by replacing the

k^{th} column of $I_{m \times m}$ with the $P\chi_{S_{i_k}}$, i.e.,

$$L^{(k)} = [e_1^T, \dots, e_{k-1}^T, \boxed{P\chi_{S_{i_k}}^T}, e_{k+1}^T, \dots, e_m^T]. \quad (1.2.17)$$

Noting that $(L^{(k)})^{-1} = L^{(k)}$, the factorization (1.2.16) becomes the factorization of PHQ as below

$$PHQ^T = \left(\prod_{k=1}^l L^{(k)} \right) \begin{bmatrix} U_{l \times l} & A_{n-l \times n-l} \\ 0 & 0 \end{bmatrix}. \quad (1.2.18)$$

In the algorithm, for each index pair $(i_k, j_k) \in (\sigma_l, \tau_l)$, the $(i_k, j_k)^{th}$ 1 is selected as the k^{th} diagonal entry of both L and U . Once the pair is chosen, then i_k is discarded from $\bar{\mathcal{T}}$ for the remainder of pivoting rounds (see the second box at line 9 in **Algorithm 1.1**). Therefore, each $L^{(k)}$ is an $m \times m$ lower triangular matrix, and thus, L is an $m \times m$ lower triangular matrix, and $U_{l \times l}$ is an upper triangular matrix. In fact,

$$L = \prod_{k=1}^l L^{(k)} = [P\chi_{S_{i_1}}^T, P\chi_{S_{i_2}}^T, \dots, P\chi_{S_{i_l}}^T, e_{l+1}^T, \dots, e_m^T], \quad (1.2.19)$$

where $S_{i_k} = \{i \in \bar{\mathcal{T}} \mid h_{i,j_k}^{(k-1)} = 1\}$ for $k = 1, 2, \dots, l$ (see line 10 in **Algorithm 1.1**).

In particular, if $\text{Rank}(H) = n$ then $A_{n-l \times n-l} = \emptyset$, and therefore,

$$PHQ^T = \left(\prod_{k=1}^n L^{(k)} \right) \begin{bmatrix} U \\ 0 \end{bmatrix}. \quad (1.2.20)$$

It can be also seen that by pivoting columns of L from the first to the last column of it, any $m \times m$ lower triangular matrix L over \mathbb{F}_2 can be factorized into $L = \prod_{k=1}^m L^{(k)}$, where $L^{(k)}$ is formed by replacing the k^{th} column of $I_{m \times m}$ with the k^{th} column L^k . Similarly, an $n \times n$ upper triangular matrix U can be factorized as $U = \prod_{k=n}^1 U^{(k)}$ by pivoting its columns from the last to the first column, where $U^{(k)}$ is now formed by replacing the k^{th} column of $I_{n \times n}$ with the k^{th} column U^k . Without loss of generality,

Algorithm 1.2: Recovery of α by the column-wise FS and BS	
1 Input: β	Output: α
$\% \leftarrow \text{FS: } \left(\prod_{k=n}^1 L^{(k)} \right) \cdot P\beta^T \rightarrow \%$	
2 for $k = 1$ to n by $k := k + 1$ do	
3 $\% \leftarrow P\beta^T := L^{(k)}P\beta^T$, where $L^{(k)} \equiv L^k$ by $S_k = \{i l_{ik} = 1\}$. $\rightarrow \%$	
4 $\left[\text{add } \beta_{i_k} \text{ to all other } \beta_i \text{ where } i \in S_k; \right.$	
$\% \leftarrow \text{BS: } \left(\prod_{k=1}^n U^{(k)} \right) \cdot P\beta^T \rightarrow \%$	
5 for $k = n$ to 1 by $k := k - 1$ do	
6 $\% \leftarrow P\beta^T := U^{(k)}P\beta^T$, where $U^{(k)} \equiv U^k$ by $S_k = \{i u_{ik} = 1\}$. $\rightarrow \%$	
7 $\left[\text{add } \beta_{i_k} \text{ to all other } \beta_i \text{ where } i \in S_k; \right.$	
$\% \leftarrow Q^T X^T := P\beta^T$. $\rightarrow \%$	
8 for $k = 1$ to n by $k := k + 1$ do	
9 $\left[\text{copy } \alpha_{j_k} := \beta_{i_k} \right.$	
10 return $\alpha = (\alpha_1, \dots, \alpha_n)$;	

we may assume that each $U^{(k)}$ is an $m \times m$ upper triangular matrix by extending

$$U^{(k)} \quad \text{as} \quad \begin{bmatrix} U^{(k)} & 0 \\ 0 & I_{m-n \times m-n} \end{bmatrix}. \quad (1.2.21)$$

Hence by the GE, $HX^T = \beta^T$ is transformed to $(LU) \begin{bmatrix} I_{n \times n} \\ 0 \end{bmatrix} (QX^T) = P\beta^T$, then is transformed into

$$\begin{bmatrix} I_{n \times n} \\ 0 \end{bmatrix} (QX^T) = \left(\prod_{k=1}^n U^{(k)} \right) \left(\prod_{k=n}^1 L^{(k)} \right) P\beta^T. \quad (1.2.22)$$

Once an LU -factorized system (1.2.22) is obtained, the unique solution α of $HX^T = \beta^T$ can be identified by computing the right-hand side of system (1.2.22) iteratively, first by *Forward Substitution* (FS) over the columns of L (from the first to the last column) then by *Backward Substitution* (BS) over columns of U (from the last to the first column of it). An exemplary pseudo-codes for the FS and BS is described in **Algorithm 1.2**.

Remark 1.2.1 (Row-wise FS and BS). L and U can be also row-wise factorized as a

Algorithm 1.3: Recovery of α by the row-wise FS and BS

```

1 Input:  $\beta^T$    Output:  $\alpha$ 

  /*<-- FS:  $\left(\prod_{k=n}^1 L^{(k)}\right) \cdot P\beta^T$  -->*/
2 for  $k = 1$  to  $n$  by  $k := k + 1$  do
3   /*<--  $P\beta^T := L^{(k)}P\beta^T$ . Note  $L^{(k)} \equiv L_k$ . -->*/
4   update  $\beta_{i_k} := \beta_{i_k} + \sum_{t=1}^{k-1} l_{k,t}\beta_{j_t}$ ;

  /*<-- BS:  $\left(\prod_{k=1}^n U^{(k)}\right) \cdot P\beta^T$  -->*/
5 for  $k = n$  to  $1$  by  $k := k - 1$  do
6   /*<--  $P\beta^T := U^{(k)}P\beta^T$ . Note  $U^{(k)} \equiv U_k$ . -->*/
7   update  $\beta_{i_k} := \beta_{i_k} + \sum_{t=n}^{k+1} u_{k,t}\beta_{j_t}$ ;

  /*<--  $Q^T X^T := P\beta^T$ . -->*/
8 for  $k = 1$  to  $n$  by  $k := k + 1$  do
9   copy  $\alpha_{j_k} := \beta_{i_k}$ .
10 return  $\alpha = (\alpha_1, \dots, \alpha_n)$ ;

```

product of elementary matrices. Since the transpose L^T is now an upper triangular matrix, $L^T = \prod_{k=1}^m \bar{L}^{(k)}$, where each $\bar{L}^{(k)}$ is formed by replacing the k^{th} column of $I_{m \times m}$ with the k^{th} column of L^T . Then

$$L = \left(\prod_{k=1}^m \bar{L}^{(k)} \right)^T = \prod_{k=m}^1 (\bar{L}^{(k)})^T, \quad (1.2.23)$$

where $(\bar{L}^{(k)})^T$ is formed by replacing the k^{th} row of $I_{m \times m}$ with the k^{th} row of L . Similarly, U can be obtained as a row-wise factorized form

$$U = \prod_{i=1}^n (\bar{U}^{(i)})^T, \quad (1.2.24)$$

where $(\bar{U}^{(i)})^T$ is now formed by replacing the i^{th} row of $I_{n \times n}$ with the i^{th} row U_i . The FS and BS in row-wise factorization is described in **Algorithm 1.3**. It should be emphasized that **Algorithm 1.3** uses precisely n rows of L and U . In contrast, **Algorithm 1.2** uses n columns of L and U for the recovery of α .

Remark 1.2.2 (Pivot Selections). Let us go back to the pivot selection in **Algorithm 1.1** of GE (see the line 7 of the algorithm). Through the pivoting rounds, let $|H(k-1)_i|$ denote the number of 1's of the row $H(k-1)_i$ after the round $k-1$ and refer to as the degree of $H(k-1)_i$. If we set the pivot selection as to choose a row of degree 1, then the GE is equivalent to the MPA (see [2, 5, 14]). In general, when rows of degree 1 are exhausted prematurely at round k , a row of degree 1 can be made by discarding columns in $H(k-1)$ in an appropriate manner. In this case, the GE is equivalent to the ALTA in [3, 4].

1.3 Low-Density Parity-Check Codes

In subsection 1.3.1, we introduce LDPC codes, the MPA in [5, 6] as a simple GE, and an systematic encoding of LDPC codes. In subsection 1.3.2, we introduce the performance analysis of LDPC codes under the MPA in [7]. By using the analysis, we show that the tornado sequence in [5] is a capacity approaching sequence. Further details of the performance analysis can be found in [7] and [28, ch. 2].

1.3.1 Encoding and Decoding Algorithm of LDPC Codes

For a given $m \times n$ matrix H over \mathbb{F}_2 , let $|H|$ denote the number of nonzero entries of H . We call $|H|$ as the density of H .

Definition 1.3.1 (LDPC Codes). For a given $m \times n$ matrix H over \mathbb{F}_2 with $m \leq n$, a binary parity-check code $C(H)$ is defined as the kernel space

$$C(H) = \{\alpha \in (\mathbb{F}_2^n) | H \cdot \alpha^T = 0\}. \quad (1.3.1)$$

We call α in $C(H)$ a *codeword*. If H is sparse, for an example, $|H| \leq cn \ln(n)$ for some constant $c > 0$, then we call $C(H)$ and H the LDPC code generated by H and an LDPC matrix, respectively.

Let H be an $m \times n$ LDPC matrix. Let us assume that rows of H are linearly independent, so that $\dim(\text{Ker}(H)) = n - m$. By GE, H can be transformed into a systematic form $G = [S_{m \times k}; I_{m \times m}]$, where $k = n - m$. Then since G is row-equivalent to H up to a column permutation, $C(G) = C(H)$. For a given $\alpha_I \in (\mathbb{F}_2^s)^k$, let $\alpha_P = S_{m \times k} \cdot \alpha_I^T$. Then $\alpha = (\alpha_I, \alpha_P)$ satisfies the kernel constraint $H\alpha^T = 0$, and thus,

$$(\mathbb{F}_2^s)^k \cong C(H) \quad \text{via} \quad E(G) = \begin{bmatrix} I_{k \times k} \\ S_{m \times k} \end{bmatrix}. \quad (1.3.2)$$

Hence by $\alpha = E(G)\alpha_I^T$, an α_I in $(\mathbb{F}_2^s)^k$ can be transformed to a codeword $\alpha \in C(H)$ in a form of $\alpha = (\alpha_I, \alpha_P)$. To obtain a row equivalent form $G = [S_{m \times k}; I_{m \times m}]$, however, the filling-in with 1 by GE makes the $|S_{m \times k}|$ proportional to n^2 . Thus, the computation for α_P by $S_{m \times k}\alpha_I^T$ in symbol additions is $O(n^2)$.

In general, the quadratic density of $S_{m \times k}$ can be significantly reduced when a row-equivalent G is replaced with a form $G = [S_{m \times k}; L_{m \times m}]$, where $L_{m \times m}$ is a lower triangular matrix. We recall that, from section 1.2, $L_{m \times m}^{-1} = \prod_{k=m}^1 L^{(k)}$, where $L^{(k)}$ is the elementary matrix formed by replacing the k^{th} column of $I_{m \times m}$ with the k^{th} column $(L_{m \times m})^k$. Therefore, once $S_{m \times k}\alpha_I^T$ is computed, α_P can be computed very efficiently by applying the FS in **Algorithm 1.3** to the product

$$\alpha_P^T := \left(\prod_{k=m}^1 L^{(k)} \right) S_{m \times k} \alpha_I^T. \quad (1.3.3)$$

Definition 1.3.2 (A Systematic Encoder). For a given $m \times n$ LDPC matrix H over \mathbb{F}_2 whose rows are linearly independent, let $G = [S_{m \times k}; L_{m \times m}]$ be an $m \times n$ matrix which is row-equivalent to H , and where the left block $L_{m \times m}$ is in a lower triangular form. Then for a given $\alpha_I \in (\mathbb{F}_2^s)^k$, the codeword $\alpha = (\alpha_I, \alpha_P)$ can be generated by

$$\begin{bmatrix} I_{k \times k} \\ L_{m \times m}^{-1} S_{m \times k} \end{bmatrix} \cdot \alpha_I^T = \begin{bmatrix} \alpha_I^T \\ \alpha_P^T \end{bmatrix}, \quad \text{where} \quad \alpha_P^T = L_{m \times m}^{-1} (S_{m \times k} \cdot \alpha_I^T). \quad (1.3.4)$$

Let $\text{En}(G) = \begin{bmatrix} I_{k \times k} \\ L_{m \times m}^{-1} S_{m \times k} \end{bmatrix}$. We call $\text{En}(G)$, α_I , and α_P a *systematic encoder* of

$C(H)$, a *systematic* part, and a *redundant* part, respectively. We refer to $R = \frac{k}{n}$ as the code rate, and $1 - R$ as the redundancy rate of $C(H)$.

If H is randomly generated with a certain row and column degree sequences, called capacity approaching sequences that will be introduced in the following section 1.3.2, then by the ALTA [3, 4], H can be permuted to a form $\bar{H} = \begin{bmatrix} A_{l \times n-l} & B_{l \times l} \\ C_{m-l \times n-l} & D_{m-l \times l} \end{bmatrix}$, where the right-top block B is a lower triangular matrix with l close to m . Multiplying by $S = \begin{bmatrix} I & 0 \\ -DB^{-1} & I \end{bmatrix}$, \bar{H} can be transformed to

$$S\bar{H} = \begin{bmatrix} A & B \\ \bar{C} & 0 \end{bmatrix} \quad \text{where } \bar{C} = -DB^{-1}A + C. \quad (1.3.5)$$

Then by Gauss-Jordan reduction [23, 24], the bottom-left block \bar{C} can be transformed into the form $[\bar{C}_k; I_{r \times r}]$, and hence, \bar{H} is row-equivalent to

$$\begin{bmatrix} A_k & A_r & B \\ \bar{C}_k & I_{r \times r} & 0 \end{bmatrix} \quad \text{or} \quad \begin{bmatrix} \bar{C}_k & I_{r \times r} & 0 \\ A_k & A_r & B \end{bmatrix}, \quad (1.3.6)$$

where $A = [A_k; A_r]$ and $k + r = n - l$. Notice that $\begin{bmatrix} I_{r \times r} & 0 \\ A_r & B \end{bmatrix}$ is now an $m \times m$ lower triangular matrix. Hence, by setting $L_{m \times m} := \begin{bmatrix} I_{r \times r} & 0 \\ A_r & B \end{bmatrix}$ and $S_{m \times k} := \begin{bmatrix} A_k \\ \bar{C}_k \end{bmatrix}$, $En(E) = \begin{bmatrix} I_{k \times k} \\ L_{m \times m}^{-1} S_{m \times k} \end{bmatrix}$ becomes a *systematic* encoder of $C(\bar{H})$. Also notice that the number of symbol additions to compute α_P by $En(G)$ is precisely $|S_{m \times k}| + |L_{m \times m}| - m$. An exemplary algorithm for the approximate lower triangulation of H is presented in **Algorithm 3.1** in section 3.2. Further details and a variety of greedy algorithms for the ALTA can be found at [3, 4].

Let α_e be a received sub-vector of α , when a codeword $\alpha \in C(H)$ was transmitted. Let $X = (x_1, \dots, x_{n_e})$ denote the erasure symbol vector, so that α can be expressed as (α_e, α_e) . Similarly, let $[N; M]$ be the rearrangement of columns of H associated with (α_e, X) . Thus, the kernel constraint system $H\alpha^T = 0$ is now expressed as

$$MX^T = \beta^T, \quad \text{where } \beta^T = N\alpha_e^T \quad (1.3.7)$$

The task of an LDPC decoder is in solving the consistent linear system (1.3.7) in the fastest way, and at the same time, with largest number of erasures n_e (but $n_e \leq m$) as possible. Obviously, the system has its unique solution $X = \alpha_e$ iff M has its full column rank.

In general, no matter what $\text{Rank}(M)$ is, all of the solutions of (1.3.7) (including free variables) can be identified by the LU factorization on M . Once a factorization of M is obtained by GE, say $PMQ^T = LU \begin{bmatrix} I_{n_e \times n_e} \\ 0 \end{bmatrix}$, solutions can be identified by first FS over L then BS over U with β as shown in **Algorithm 1.2**. In general, the computational complexity of the LU -factorization in bit operations (or bit-flips) is in $O(n_e^3)$, and the FS and BS together constitutes the complexity $O(n_e^2)$ in symbol additions of β . For a moment, let us now assume that M can be permuted to a lower triangular matrix L by a pair of row and column permutation (P, Q) , say $L = PMQ^T$. In that case, the erasure (symbol) vector X can be recovered by the simple FS over columns (or rows) of L with β as in **Algorithm 1.2** or **1.3**. Therefore, the number of symbol additions by the FS is less than $|M| + |N| = |H|$. Furthermore, if H is sparse, then M is sparse also, and therefore, the FS is very efficient.

Let us now describe the MPA, the fastest decoding algorithm of LDPC codes known so far. The MPA was designed for decoding of tornado codes in [5] for the first time, and is equivalent to a lower triangulation algorithm. We explain the MPA based on the exemplary pseudo-codes in **Algorithm 1.4**. The algorithm is based on the **Degree Reduction Rounds** that corresponds to the **Pivoting Rounds** of the LU -factorization in **Algorithm 1.1**. At each round k of the reduction (see **foreach** loop in **Algorithm 1.4**), starting from $M(0) = M$, let $M(k) = (m_{ij}^{(k)})$ denote the transformed matrix by adding the k^{th} pivot row $M(k-1)_{i_k}$ to the rows $M(k-1)_i$, where $i \in \bar{\mathcal{T}}$ and $m_{i,j_k}^{(k-1)} = 1$. Let $S_{i_k} = \{i \in \bar{\mathcal{T}} \mid m_{i,j_k}^{(k-1)} = 1\}$ (see the line 16 in the algorithm), and let $\deg(M(k)_i) = |M(k)_i|$, the number of 1's of $M(k)_i$. Then the row addition by $M(k)_i := M(k-1)_i + M(k-1)_{i_k}$ of the k^{th} pivoting round of

Algorithm 1.4: The Message Passing Algorithm on M

```

%<- Initialization: To identify  $M \rightarrow$ %
1 set  $\tilde{T} := [m]$ 
2 foreach (received symbol)  $\alpha_j \in \alpha_{\tilde{e}}$  do
3   | discard the column  $H^j$  from  $H$ ;
4 Input:  $M$    Output:  $(\sigma_l, \tau_l)$ .
5 foreach  $i \in \tilde{T}$  do
6   | identify  $\deg(M_i)$ ; //<-- i.e.  $\deg(M_i) = |M_i|$ .  -->//
7   | if  $\deg(M_i) = 0$  then
8   |   | discard  $\tilde{T} := \tilde{T} \setminus i$ ;

%<- Degree Reduction Rounds  $\Leftrightarrow$  Pivoting Rounds of GE -->%
9 while  $\tilde{T} \neq \emptyset$  do
10  //<-- Selection of the  $k^{th}$  Diagonal Entry of  $L$  -->//
11  if  $\exists i_k \in \tilde{T}$  such that  $\deg(M(k-1)_{i_k}) = 1$  then
12    | identify  $j_k$  with which  $m_{i_k j_k}^{(k-1)} = 1$ ;
13    | set  $\sigma_l(k) = i_k$  and  $\tau_l(k) = j_k$ ;
14    | discard  $\tilde{T} := \tilde{T} \setminus i_k$ ;
15  else goto FinalRound;

%<- Degree Reduction  $\Leftrightarrow \bar{E}^{(k)} M(k-1) \rightarrow$ %
16  foreach  $i \in \tilde{T}$  with which  $m_{i j_k}^{(k-1)} = 1$  do
17    | %<-  $\Leftrightarrow M(k)_i := M(k-1)_i + M(k-1)_{i_k}$ ; -->%
18    | reduce  $\deg(M(k)_i) := \deg(M(k-1)_i) - 1$ ;
19    | if  $\deg(M(k)_i) = 0$  then
20    |   | discard  $\tilde{T} := \tilde{T} \setminus i$ ; //<-- To discard null rows

20 FinalRound: //<-- Say the 'while' loop stops at  $k = l$ 
21 if  $l = n_e$  then
22   | return  $(\sigma_l, \tau_l)$ ; //<-- declare 'Decoding Success'
23 else
24   | return  $(\sigma_l, \tau_l)$ ; //<-- declare 'Decoding Failure'

```

Algorithm 1.1 (LU -factorization algorithm) is simplified as the row-degree reduction by $\deg(M(k)_i) := \deg(M(k-1)_i) - 1$ (see the lines 16 – 17 in the algorithm). The reason to this is in the fact that the MPA selects a pivot row from rows of degree

Algorithm 1.5: The recovery of $QX_{\bar{\mathcal{R}}}$ by the FS in **Algorithm 1.3**

```

1 Input:  $[N; M]$  and  $(\sigma_l, \tau_l)$    Output:  $X$ 
   /*<-- Initialization for  $\beta = N\alpha_{\bar{e}}^T$  -->*/
2 for  $i = 1$  to  $m$  by  $i := i + 1$  do
3   if  $i \in \sigma$  then
4     set  $\beta_i := N_i\alpha_{\bar{e}}^T$ ;
5   else
6     set  $\beta_i := 0$ ;

   /*<--  $QX^T = \left(\prod_{k=n_e}^1 L^{(k)}\right) P\beta^T$  -->*/
7 recover  $QX^T$  by using the FS in Algorithm 1.3 with  $P\beta^T$ ;
8 return  $QX^T$ ;

```

1 only, and thus, the resulting upper triangular matrix is simply the identity matrix $I_{m \times m}$. Then by $\bar{E}^{(k)}$ formed by replacing the i_k^{th} column of $I_{m \times m}$ with the $\chi_{S_{i_k}}^T$ as defined in (1.2.7), the Degree Reduction (the lines 16 – 19 in **Algorithm 1.4**) corresponds to $\bar{E}^{(k)}M(k-1)$ of GE. Similarly, at the last round l , if the MPA succeeds (i.e. $l = n_e$) then $M(l)$ can be expressed as

$$M(n_e) = \left(\prod_{k=n_e}^1 \bar{E}^{(k)} \right) M. \quad (1.3.8)$$

Then by using the permutation pair (P, Q) of (σ, τ) , which is extended from the returned (σ_l, τ_l) (see line 22 in **Algorithm 1.4** with the equations (1.2.14) and (1.2.15)), the initial system $MX^T = \beta^T$ is permuted to $(PMQ^T)(QX^T) = P\beta^T$, then is transformed to a product form

$$\begin{bmatrix} I_{n_e \times n_e} \\ 0 \end{bmatrix} QX^T = \left(\prod_{k=n_e}^1 L^{(k)} \right) P\beta^T, \quad (1.3.9)$$

where $L^{(k)}$ is now the elementary matrix formed by replacing the k^{th} row of $I_{m \times m}$ with the k^{th} row of the lower triangular matrix $L = PMQ^T$.

Once a lower triangulation of M is obtained by the MPA, the erasure vector QX^T

can be computed by the FS as described in **Algorithm 1.5** in section 1.2. In the algorithm, notice that for the recovery of the lost X , the number of symbol additions is less than $|H|$, since the row-wise FS uses precisely n_e rows of H .

Remark 1.3.1 (Further Development to S-MLDA). The MPA by **Algorithm 1.4** is designed to stop when rows of degree one are exhausted (see line 9 of the algorithm). If the lastly returned l is less than n_e , then a part of X can be recovered by replacing n_e with l in system (1.3.8). As a matter of fact, the stopping condition becomes the major defect of the codes when block lengths n are not large enough, say n is within several thousands. The reason to this is that, quite feasibly, the MPA stops prematurely (i.e. $l < n_e$) but M has its full column rank n_e . In chapter 2 and 3, this defect will be naturally removed by exploiting the ALTA in [3, 4] of the MLDA. The MLDA also has a couple of inefficiencies at the initialization step and with symbol additions, but it can solve the system $MX^T = \beta^T$ as long as $\text{Rank}(M) = n_e$. In the later chapters 2 and 3, the MLDA will be further developed into the S-MLDA by removing all possible inefficiencies of the MLDA.

1.3.2 Probability Density Evolutions on Degree Ensembles

The design of H that meet the following conditions in both density and rank property is the primary issue of LDPC codes:

1. For the time-efficiency of decoding codes, H should be designed as sparse as possible, at the same time, a randomly chosen $m \times n_e$ sub-matrix M of H ($n_e \leq m$) can be lower triangulated by the MPA with high probability;
2. M has its full column rank n_e , with n_e close to the row-dimension m as possible.

In this section, we introduce *Probability Density Evolution* that provides us a convenient tool for designing H that meet the two conditions above. Let us clarify several terms for the description of the density evolution.

Definition 1.3.3. Let 1_{st} denote the h_{st} , if $h_{st} = 1$.

Degree of 1_{st} : For a given 1_{st} , let $|H_s| = i$ and $|H^t| = j$. We say that a 1_{st} has its *row-degree i* and *column-degree j* .

Entry degree ensemble: Let define

$$\lambda_j = \frac{|\{1_{st} | \deg(H^t) = j\}|}{|H|} \quad \text{and} \quad \rho_i = \frac{|\{1_{st} | \deg(H_s) = i\}|}{|H|}. \quad (1.3.10)$$

Then let $\lambda(x) = \sum_{j \geq 1} \lambda_j x^{j-1}$ and $\rho(x) = \sum_{i \geq 2} \rho_i x^{i-1}$ the column-degree and row-degree distribution of entries of H , respectively. We call $(\lambda(x), \rho(x))$ the *entry-degree ensemble* of H .

Notice that, rather than the term x^i and x^j , each λ_j and ρ_i is associated with the term x^{j-1} and x^{i-1} , respectively. The reason to this will become clear soon. We also note that $\sum \lambda_j = \lambda(1) = 1$ and $\sum \rho_i = \rho(1) = 1$.

Let m_i and n_j be the number of rows and columns of degree i and j , respectively. Then $|H| = \sum i \cdot m_i = \sum j \cdot n_j$. Since $m_i = \frac{\rho_i |H|}{i}$ and $n_j = \frac{\lambda_j |H|}{j}$, the average row and column-degree a_r , a_c , respectively, is expressed as

$$a_r = \frac{|H|}{m} = \left(\sum \frac{\rho_i}{i} \right)^{-1} \quad \text{and} \quad a_c = \frac{|H|}{n} = \left(\sum \frac{\lambda_j}{j} \right)^{-1}. \quad (1.3.11)$$

Thus, $m = n(\frac{a_c}{a_r})$, and hence, the code rate $R = \frac{k}{n}$ can be expressed in terms of a_r and a_c as following

$$R = \frac{k}{n} = \frac{n-m}{n} = 1 - \frac{a_c}{a_r} = 1 - \frac{\sum \rho_i / i}{\sum \lambda_j / j} = 1 - \frac{(\int \rho)(1)}{(\int \lambda)(1)}. \quad (1.3.12)$$

We also note that from (1.3.11), once the average degrees a_r and a_c are chosen, then m is constrained as $m = n(\frac{a_c}{a_r})$.

Let H be an $m \times n$ random matrix that follows the entry-degree distribution $(\lambda(x), \rho(x))$, and let $C(H)$ be the LDPC code by H . Let $p_0 = \frac{ne}{n}$ be a fraction of random erasures when a codeword $\alpha \in C(H)$ was transmitted over BEC. One key

component for analyzing the performance of the MPA in erasure recovery is to study the initial erasure rate p_0 with which an $m \times n_e$ random sub-matrix M of H can be lower triangulated by the MPA. We first approach a probability density evolution in erasure rate p as a probability generating function associated with $(\lambda(x), \rho(x))$.

Let us now interpret the MPA as the following iterative rounds on H : Each round of the BPA consists of the following procedures:

Round k : If there exists known columns, then for every known column H^t , which was unknown and declared to be known at the previous round, and for every $1_{st} \in H^t$, the degree of H_s is reduced by 1: if $\deg(H_s) = 1$, the algorithm identifies the $1_{s't'}$ whose $H^{t'}$ is unknown by that moment and $H^{t'}$ is declared to be known at this round.

At round 0, a column H^j is declared to be known, if the associating symbol α_j was received, otherwise unknown. At each round k , we say that a 1_{st} is *unknown*, if the column H^t is unknown.

Theorem 1.3.1 (Density Evolution). *Let p_k be the probability that a 1 in H is unknown at the round k . Then*

$$p_{k+1} = p_0 \lambda(1 - \rho(1 - p_k)). \quad (1.3.13)$$

Proof. At the iteration round k , an unknown 1_{st} has its initial row-degree i with probability ρ_i , and the degree is reduced to 1 with probability $(1 - p_k)^{i-1}$, because all other 1's in the row H_s must be known by the round k . Similarly, 1_{st} is still unknown with probability $1 - (1 - p_k)^{i-1}$. Hence, at this round, the average probability that an unknown 1 is still unknown at the end of the round is given as

$$q_k = \sum \rho_i (1 - (1 - p_k)^{i-1}) = 1 - \rho(1 - p_k). \quad (1.3.14)$$

Now, an unknown 1_{st} of the round k has the column degree j with probability λ_j , and it is still unknown if the column H^t is initially lost (with probability p_0) and all

other 1's of H^t are still unknown. Thus, the probability that an unknown 1_{st} has a column-degree j and is still unknown at the round $k + 1$ is given as $p_0 \lambda_j(q_k)^{j-1}$. Therefore, at the begining of the round $k + 1$, a 1 is unknown with probability

$$p_{k+1} = \sum_j \lambda_j(p_0 q_k^{j-1}) = p_0 \sum_j \lambda_j q_k^{j-1} = p_0 \cdot \lambda(1 - \rho(1 - p_k)). \quad (1.3.15)$$

□

It should be notice that from (1.3.13), $n(p_k - p_{k+1}) \geq 1$ for the successful lower triangulation of M by the MPA. Then replacing by $p_k = x$, the MPA should satisfy the inequality

$$p_0 \cdot \lambda(1 - \rho(1 - x)) < x, \quad \forall x \in (0, p_0]. \quad (1.3.16)$$

For a given block length n , let $p^*(n)$ be the supremum of such p_0 satisfying (1.3.16). Then designing $(\lambda(x), \rho(x))$ such that its $p^*(n)$ is closed to the ideal limit $1 - R = \frac{m}{n}$ is the key part of designing LPDC codes, and not every $(\lambda(x), \rho(x))$ fulfills this property.

Example 1.3.1. Let $\lambda(x) = x^2$, $\rho(x) = x^5$. Then p^* can be obtained by considering the supremum of $\{p_0\}$ in which each p_0 satisfies

$$p_0(1 - (1 - x^5))^2 < x, \quad \forall x \in (0, p_0]. \quad (1.3.17)$$

It is shown that in $[4, 9]$, p^* is approximately 0.429 which is far from $\frac{m}{n} = 0.5$.

The inequality (1.3.16) is useful whether a known (λ, ρ) holds it or not. Let us call a entry degree ensemble (λ, ρ) as a *capacity approaching* sequence, if it holds the following condition: for a given $\epsilon > 0$, there exists D_0 such that for all $D \geq D_0$

$$(1 - R)(1 - \epsilon)\lambda_D(1 - \rho_D(1 - x)) < x, \quad \forall x \in (0, (1 - R)(1 - \epsilon)), \quad (1.3.18)$$

where $\lambda_D(x)$ and $\rho_D(x)$ consists of first D terms of $\lambda(x)$ and $\rho(x)$, respectively.

The first capacity approaching degree sequence for BEC is the tornado sequence discovered in [5] and [6]. Let $D := \lceil 1/\epsilon \rceil$, and let

$$\lambda_D(x) = \frac{1}{H(D)} \sum_{i \geq 2}^D \frac{1}{i-1} x^{i-1}, \quad H(D) = \sum_{i \geq 2}^D \frac{1}{i-1} \approx \ln(D), \quad (1.3.19)$$

$$\rho_D(x) = e^{\alpha(x-1)}, \quad \alpha = \frac{H(D)}{p_0}. \quad (1.3.20)$$

Plugging in (λ_D, ρ_D) into (1.3.16) asserts

$$p_0 \lambda_D(1 - \rho_D(1 - x)) \leq p_0 \lambda_D(1 - e^{-\alpha x}) < \frac{-p_0}{H(D)} \ln(e^{-\alpha x}) = x. \quad (1.3.21)$$

Hence, successful triangulation of M by the MPA is possible, if the fraction of erasures is no more than $\frac{H(D)}{\alpha}$. We note that by $(1 - R) \cdot (\int \lambda)(1) = (\int \rho)(1)$ in (1.3.12),

$$(1 - R) \frac{1}{H(D)} (1 - 1/(D + 1)) = \frac{1}{\alpha} (1 - e^{-\alpha}). \quad (1.3.22)$$

Thus $\frac{\alpha}{H(D)} = \frac{1-R}{1-e^{-\alpha}} (1 - 1/(D + 1))$ and is larger than $(1 - R)(1 - 1/D)$. Consequently,

$$(1 - R)(1 - 1/D) \lambda_D(1 - \rho_D(1 - x)) < x, \quad \forall x \in (0, (1 - R)(1 - 1/D)). \quad (1.3.23)$$

Many other capacity approaching sequences can be found in [12]. In practice, however, once a good degree sequence is found by linear search as in [5], many other sequences can be made by changing its fractions slightly. In chapter 3, we simulate LDPC codes under the S-MLDA generated by a slightly modified right-degree ensembles appeared in [4, 13].

1.4 Luby Transform Codes over BEC

In this section, we introduce LT codes invented by M. Luby [2], which was designed for multi- and broad-cast of multimedia over BEC without retransmission request of symbols over the Internet. For the design of LT codes, we introduce the Ideal Soliton

Distribution (ISD) and the Robust Soliton distribution (RSD). We then describe the ISD in two ways: first by Mackay's interpretation in [27, ch. 50], then second by Shokrollahi's interpretation in [14]. For the original design of the ISD, we refer readers to the LT paper [2]. In section 2.4, the RSD will be explained well by generalizing the Mackay's recursive formula (see **Lemma 2.4.1** at p. 52).

1.4.1 Encoding and Decoding Algorithms of LT codes

Let us first describe the LT transmission scheme over BEC. Suppose we would like to communicate an information data set I to receivers over BEC. We first subdivide the I into n symbols (or packets) of equal length s , say $\alpha = (\alpha_1, \dots, \alpha_n) \in (\mathbb{F}_2^s)^n$. Let us call α_i and α an input symbol and an input symbol vector, respectively. Now let $\rho(x) = \sum_{d=1}^n \rho_d x^d$ be a given probability distribution. The LT transmission scheme is as follows. For a given input symbol vector $\alpha \in (\mathbb{F}_2^s)^n$ (at an LT server), an LT encoder constantly generates syndrome symbols β_i 's by using the $\rho(x)$ and α in the following manner:

- 1) a degree d is randomly chosen with probability ρ_d , then a row $H_i \in \mathbb{F}_2^n$ of degree d is chosen at random from the $\binom{n}{d}$ possible choices;
- 2) the encoder computes $\beta_i := H_i \alpha^T$, it then transmits β_i over BEC.

The transmission stops when a receiver acquires a sufficient number of syndrome symbols. At the receiver end, if more than m syndrome symbols are received, where $m = (1 + \gamma)n$ for some $\gamma > 0$, then an LT decoder recovers α by solving the consistent linear system

$$HX^T = \beta^T, \quad \text{where } \beta = (\beta_1, \dots, \beta_m) \in (\mathbb{F}_2^s)^m \quad (1.4.1)$$

and H consists of rows H_i such that $H_i \alpha^T = \beta_i$.

Let us compare the systems (1.4.1) for LT codes and (1.3.7) for LDPC codes. In both LT and LDPC codes transmission scheme, the task of decoders is in solving the consistent linear systems (1.3.7) and (1.4.1) for their unique solution. Although the systems are almost same except the notations, there are significant differences between LT and LDPC code based transmission schemes.

In LDPC code based transmission scheme, for every instance of I , first, I is put into α_I in $(\mathbb{F}_2^s)^k$, where k is fixed by $k = \dim(\text{Ker}(H))$; second, α_I is transformed into a longer symbol vector $\alpha = (\alpha_I, \alpha_P)$ in $\text{Ker}(H)$; then lastly, symbols of α are transmitted over BEC till a receiver gets enough number of them, or every symbols of α is transmitted if a feedback is not available on the channel. The syndrome vector β of system (1.3.7) is computed by an LDPC decoder. Since H is already known to a decoder, an LDPC decoder can quickly identify system (1.3.7) by reading columns of H . However, since the M in system (1.3.7) is a sub-matrix that consists of columns of H , the row dimension of M is always fixed by the row-dimension of H , and therefore, k is fixed for every instance of I .

In LT transmission scheme, contrastingly, first, n can be chosen conveniently for each instance of an information data I ; second, I is simply put into an $\alpha \in (\mathbb{F}_2^s)^n$; then lastly, a syndrome vector β in system (1.4.1) is generated by an LT encoder and is transmitted over BEC. For every instance of α , however, the system (1.4.1) should be identified at the initialization step of decoding algorithms. To communicate the system (1.4.1) to a decoder, each H_i can be directly attached to β_i , or a decoder can generate H by using the same random generator of the LT encoder. Note that in both cases, the cost for communicating H in system (1.4.1) is not trivial. A variety of pseudo-random generators are available for the selection of a degree d and a row H_i of degree d . For an example, in the sections 2.6, 2.7, and 3.4, we use Mersenne Twister Algorithm [16] for the random selection of d and H_i .

Definition 1.4.1. (An LT code by $\rho(x)$) With a received symbol vector β from an

LT encoder, let H be the $m \times n$ matrix over \mathbb{F}_2 with $m \geq n$ such that $\beta_i = H_i \alpha^T$ for $i = 1, \dots, m$. We define the LT code generated by $\rho(x)$ as the set of all pairs $\{(\beta, H)\}$ in which each (β, H) forms a consistent linear system (1.4.1).

Assuming that rows of H follow the distribution $\rho(x)$ such as the RSD in [2], it is known that, if $m \geq (1 + \gamma)n$, $\exists \approx 0$, then a random $m \times n$ matrix H by $\rho(x)$ can be lower triangulated by the MPA, and hence, α can be recovered by the FS over the triangulated matrix (see [2, **Theorem 17**]). In reality, however, particularly for short block lengths n , say n is within several thousands, the triangulation by the MPA is not guaranteed, if γ is not large enough. We note that system (1.4.1) has the unique solution α iff $\text{Rank}(H) = n$. When $\text{Rank}(H) = n$, regardless of the failure of the MPA, system (1.4.1) can be solved for its unique solution α at least by a conventional GE on H . In section 2.2, we will generalize the MPA on H into the S-MLDA.

1.4.2 The Robust Soliton Degree Distribution

Similar to LDPC codes, the design of LT codes is focused on the design of a row-degree distribution $\rho(x)$ that meets two conditions:

1. A random $m \times n$ matrix H , generated by $\rho(x)$ with the row-dimension m as close to n as possible, can be lower triangulated by the MPA with high probability.
2. H is sparse as possible.

Through the degree-reduction rounds of the MPA on H , let us call the set of rows of reduced-degree 1 at each round k as a Ripple. The basic property required of a good degree distribution $\rho(x)$ is that, the number of rows of reduced degree 1 at each round k is greater than 1 but is as small as possible. If a size of the ripple is too large, then some of the rows of reduced degree 1 in the ripple, say $H^{(k)}_i$, may be same with the ones already in the ripple, so the check equation $H_i X^T = \beta_i$ of system (1.4.1)

is redundant. If the ripple is too small, then it may become empty, before the MPA finishes the lower triangulation of H .

The ISD $\rho(x)$ described next exhibits the ideal behavior in terms of the expected number of rows of a random H generated by $\rho(x)$, needed to recover α . Specifically speaking, one row of degree 1 in the ripple gives one row of reduced degree 1 for the next round of the degree-reduction process. In this sense, exactly n rows are needed to recover n input symbols by the MPA. The ISD, however, is quite fragile so that it is useless in practice. Even a small variance of the ripple through the degree-reduction rounds causes it to be empty. However, it gives many of the crucial ideas for the RSD described later.

Definition 1.4.2 (Ideal Solitan Distribution). The ISD $\rho(x) = \sum_{i=1}^n \rho_i x^i$ is defined as following:

$$\rho_i = \begin{cases} \frac{1}{n} & \text{for } i = 1, \\ \rho_i = \frac{1}{i(i-1)} & \text{for } i = 2, \dots, n \end{cases}. \quad (1.4.2)$$

Let H be an $m \times n$ matrix generated by the ISD $\rho(x)$. Then

$$|H| = \sum_{i=1}^n i(n\rho_i) = 1 + n \sum_{i=2}^n \frac{1}{i}. \quad (1.4.3)$$

Thus, the average row-degree $a_r = \frac{|H|}{n}$ can be approximated as $\ln(n)$ for large n .

Let us first give an easier interpretation for the ISD by using Mackay's recursive formula in [27, p592]. Let H be a random matrix generated by the ISD $\rho(x)$. Through the degree-reduction round t of the MPA, starting from $H(0) := H$, let $H(t)$ denote the residual matrix of $H(t-1)$ by discarding the row H_{i_t} and the column H^{j_t} , when the $1_{i_t, j_t}$ is selected as the t^{th} diagonal of a triangular matrix L by that round. Now, let $h_t(i)$ denote the expected number of rows of $H(t)$ having its reduced degree i after the reduction round t . We want to design a degree distribution by which a randomly generated H gives the ripple size $h_t(1) = 1$ for all $t \in \{0, 1, \dots, n-1\}$ through the reduction rounds of the MPA. At the reduction round t , for each degree $i > 1$, since

every row of H is generated in random, the probability that a 1 of a row in $H(t)$ having degree i is in the discarded column H^jt is $\frac{1}{n-t}$. Thus, the expected number of rows of $H(t)$, whose degree i is reduced to $i-1$ in $H(t+1)$ by the discarded column H^jt , is given as $\frac{i \cdot h_t(i)}{n-t}$. Similarly with $h_t(i+1)$, a total of $\frac{(i+1)h_t(i+1)}{n-t}$ rows of degree $i+1$ in $H(t)$ become rows of degree i in $H(t+1)$. Therefore, at the beginning of the round $t+1$, the expected number of rows of reduced-degree i is given as

$$h_{t+1}(i) = h_t(i) \left(1 - \frac{i}{n-t}\right) + \frac{(i+1)h_t(i+1)}{n-t}. \quad (1.4.4)$$

Setting by $h_{t+1}(1) = h_t(1) = 1$ for all t , we get $h_t(2) = \frac{n-t}{2}$ from (1.4.4). Then by the induction on (1.4.4),

$$h_t(i) = \frac{n-t}{i(i-1)}. \quad (1.4.5)$$

Particularly with $t = 0$, we obtain

$$h_0(i) = \begin{cases} \frac{n}{i(i-1)} & \text{for } i > 1, \\ 1 & \text{for } i = 1. \end{cases} \quad (1.4.6)$$

Then normalizing by $\rho_i := \frac{h_0(i)}{n}$, we obtain the ISD as

$$\rho = \left(\frac{1}{n}, \frac{1}{2}, \frac{1}{2 \cdot 3}, \dots, \frac{1}{i(i-1)}, \dots, \frac{1}{n(n-1)} \right). \quad (1.4.7)$$

In the following remark, we derive a differential equation and the ISD is approximated from the solution of the equation.

Remark 1.4.1 (Shokrollahi's Interpretation for the ISD). Let H be an $m \times n$ random matrix H generated by a row-degree distribution $\rho(x) = \sum_i \rho_i x^i$. At the degree-reduction round t of the MPA, the probability that a row of initial degree i has exactly one 1 in the residual matrix $H(t+1)$ is as following. Since every row of H is generated in random, the probability that exactly one 1 of the row is in the residual matrix $H(t+1)$ is $i \left(1 - \frac{t+1}{n}\right)$. Now, the probability that all other $i-1$ number of

1's of the row is in the t number of discarded columns at round t is $\left(\frac{t}{n}\right)^{i-1}$. Likewise, at the round $t + 1$, the probability that all other $i - 1$ number of 1's of the row is in the $t + 1$ number of discarded columns is given as $\left(\frac{t+1}{n}\right)^{i-1}$. Thus, the probability that the row of degree i in H becomes a row of degree 1 in $H(t + 1)$ after the round $t + 1$ is given as

$$i \left(1 - \frac{t+1}{n}\right) \left(\left(\frac{t+1}{n}\right)^{i-1} - \left(\frac{t}{n}\right)^{i-1} \right). \quad (1.4.8)$$

Hence in average, the probability that a row of H becomes of a row degree 1 in $H(t + 1)$ is given as

$$\begin{aligned} & \left(1 - \frac{t+1}{n}\right) \left(\sum_{i=1}^n \rho_i \cdot i \left(\frac{t+1}{n}\right)^{i-1} - \sum_{i=1}^n \rho_i \cdot i \left(\frac{t}{n}\right)^{i-1} \right) \\ &= \left(1 - \frac{t+1}{n}\right) (\rho'((t+1)/n) - \rho'(t/n)). \end{aligned} \quad (1.4.9)$$

Then by using $\rho'((t+1)/n) - \rho'(t/n) \approx \frac{1}{n} \rho''(t/n)$ for large n , the probability (1.4.10) can be approximated as

$$\left(1 - \frac{t+1}{n}\right) \frac{1}{n} \rho''(t/n). \quad (1.4.10)$$

We now assume that the row dimension m of H is very close to the column dimension n , so that the expected number of rows of degree 1 in $H(t + 1)$ is approximated as

$$m \cdot (1.4.10) \approx \left(1 - \frac{t+1}{n}\right) \rho''(t/n). \quad (1.4.11)$$

Then setting by $\left(1 - \frac{t+1}{n}\right) \rho''(t/n) = 1$ and $x = \frac{t}{n}$, we derive the differential equation

$$(1 - x) \rho''(x) = 1, \quad \text{for } 0 < x < 1. \quad (1.4.12)$$

Then using the initial condition $\rho(1) = 1$, the solution of the differential equation (1.4.12) is given as

$$\rho(x) = \sum_{i \geq 2} \frac{1}{i(i-1)} x^i. \quad (1.4.13)$$

The distribution above is similar to the ISD, except that the first term $\rho_1 = 0$ and it

has infinitely many terms.

As mentioned earlier, the ISD $\rho(x)$ in **Definition 1.4.2** is so weak because, a random H by the $\rho(x)$ has the expected ripple size as 1 through the degree-reduction round of the MPA; even a small variance of the ripple causes it to be empty. Furthermore, some of the columns of H could be null. A small modification on the ISD fixes this problem. The RSD in [2] ensures that the expected size of the ripple is large enough at each degree-reduction round of the MPA, so that it never becomes the empty set with high probability. The idea of the RSD is in the design of a row-degree distribution $\mu(x)$ by which an $m \times n$ random H gives the expected ripple size about $c \cdot \sqrt{n} \ln(n/\delta)$ for every reduction round of the MPA, where δ is the upper bound of the probability that the MPA on H fails to the complete lower triangulation of H and c is a constant of order 1. The intuition for the ripple size $c \cdot \sqrt{n} \ln(n/\delta)$ is come from the observation that, according to Luby's in [2], the probability of a random walk of length n deviates from its mean by more than $\sqrt{n} \ln(n/\delta)$ is at most δ . So, the RSD is designed with $h_0(1) = c\sqrt{n} \ln(n/\delta)$.

Definition 1.4.3 (Luby's Robust Solitan Distribution). The RSD $\mu(x) = \sum \mu_i x^i$ is defined as follows. Let $R = c \cdot \sqrt{n} \ln(n/\delta)$ for some constant $c > 0$. We first define $\tau(x = \sum \tau_i x^i)$ as following:

$$\tau_i = \begin{cases} R/(in) & \text{for } i = 1, \dots, n/R - 1 \\ R \ln(R/\delta)/n & \text{for } i = n/R \\ 0 & \text{for } i = n/R + 1, \dots, n. \end{cases} \quad (1.4.14)$$

We then add the ISD $\rho(x)$ to $\tau(x)$ and normalize it as the RSD $\mu(x) = \sum \mu_i x^i$ such that

$$\mu_i = (\rho_i + \tau_i)/\omega \quad \text{where} \quad \omega = \sum_i (\rho_i + \tau_i). \quad (1.4.15)$$

for all $i = 1, 2, \dots, n$.

Setting the number of syndrome symbols by $m = n\omega$ gives the expected number of syndrome symbols of degree i by

$$n \cdot \mu_i = n \cdot (\rho_i + \tau_i) = \begin{cases} \frac{n}{i(i-1)} + \frac{R}{i} & \text{if } i \leq \frac{n}{R} - 1 \\ \frac{R^2}{n} + R \ln(R/\delta) & \text{if } i = \frac{n}{R} \\ \frac{n}{i(i-1)} & \text{if } i \geq \frac{n}{R} \end{cases}. \quad (1.4.16)$$

Especially, notice that $m\mu_1 = 1 + R$ and $m \cdot \mu_2 = n/2 + R/2$.

We introduce the main theorem of the LT paper [2]. For the proof of the theorem, we refer readers to **Theorem 17** in the original paper [2].

Theorem 1.4.1 (**Theorem 17** in [2]). *The MPA on a random $m \times n$ matrix H , generated by the RSD $\mu(x)$, fails to recover the input symbol α with probability at most δ from a set of $m = n \cdot \omega$ syndrome symbols.*

Remark 1.4.2 (Further Works on the RSD). In section 2.4, replacing the initial condition $h_t(1) = 1$ with $h_t(1) = S + 1$ for some integer $S \geq 1$, we generalize Mackay's recursive formula (1.4.4) into (2.4.1). We will explain the Luby's RSD $\mu(x)$ as a particular case of the solution (2.4.4). To ensure the success of the MPA with high probability, it should be emphasized that the number of syndrome symbols (or the number of rows of H) should be large enough.

Remark 1.4.3 (Raptor Codes). In [14], Shokrollahi generalizes LT codes into Raptor codes by using pre-decoding stage on α . The main idea of Raptor codes is that, an input symbol vector α is protected by systematic LDPC codes or other codes in prior to LT encoding called "Pre-Coding". Then LT encoding is applied to the precoded input symbol vector. For further detail, see the raptor paper [14].

CHAPTER 2

The Maximum-Likelihood Decoding Algorithms of LT Codes

In this chapter, we first present the S-MLDA as an advanced form of the MLDA in [3]. We then present the design of LT degree distribution $\rho(x)$ by supplementing the RSD $\mu(x)$ with a small fraction of dense rows. Thus, a random H , generated by our designed $\rho(x)$, may fit for the S-MLDA. By using the Kovalenko's Rank Distribution in [21, 22, 29], we also present the rank distribution of random H generated by the $\rho(x)$. Simulation results, which show the viability of the proposed MLDA of LT codes, are also presented in section 2.6. In section 2.7, particularly, we substantiate that, by experimental results, LT codes from an arranged encoder matrix can achieve a stable overhead γ (for the successful S-MLDA) close to 0.

2.1 Introduction and Backgrounds

Let $\alpha = (\alpha_1, \dots, \alpha_n) \in (\mathbb{F}_2^s)^n$ be a given input symbol vector that we would like to communicate over BEC. In the LT based data transmission scheme, an LT encoder constantly generates syndrome symbols $\beta_i \in \mathbb{F}_2^s$ using a row-degree distribution $\rho(x) = \sum_{d=1}^n \rho_d x^d$ and α in the following manner:

1. a degree d is randomly chosen with probability ρ_d , then a row $H_i \in \mathbb{F}_2^n$ of degree d is chosen at random from the $\binom{n}{d}$ possible choices;

2. the encoder generates $\beta_i = H_i \alpha^T$, it then transmits β_i over BEC.

The transmission stops when a receiver acquires a sufficient number of syndrome symbols. At a receiver end, if more than $m = (1 + \gamma)n$ syndrome symbols are received for some $\gamma > 0$, then an LT decoder recovers the α by solving the consistent linear system

$$HX^T = \beta^T, \quad \beta = (\beta_1, \dots, \beta_m) \quad (2.1.1)$$

where H consists of rows H_i such that $H_i \alpha^T = \beta_i$. Assuming that rows of H follow the distribution $\rho(x)$ such as the RSD in [2], H can be lower triangulated by means of permuting rows and columns of H , as in the MPA [5] (or see **Algorithm 1.4**). Thus α can be recovered by the FS algorithms **Algorithm 1.2** and **1.3** over a triangulated matrix of H by the MPA. For short block lengths n , however, the success of the MPA (a triangulation of H by the MPA) is not guaranteed as $\gamma \rightarrow 0$.

Nonetheless, regardless of the MPA failure, system (2.1.1) has its unique solution iff $\text{Rank}(H) = n$. If $\text{Rank}(H) = n$, then system (2.1.1) can be solved efficiently by the MLDA suggested by Burshtein and Miller in [3]. Their MLDA exploits four major routines as following:

- 1 *ALTA*: By the ALTA on H (see [3,4] or **Algorithm 3.1**), a pair of row and column permutations (P, Q) of H is obtained with which $\bar{H} = PHQ^T = \begin{bmatrix} A & B \\ C & D \end{bmatrix}$, where B is in a lower triangular form. Thus, system (2.1.1) is permuted into

$$\begin{bmatrix} A & B \\ C & D \end{bmatrix} \begin{bmatrix} X_{\bar{R}}^T \\ X_{\bar{L}}^T \end{bmatrix} = \begin{bmatrix} \beta_u^T \\ \beta_l^T \end{bmatrix} \quad \equiv \quad \bar{H} \bar{X}^T = P \beta^T, \quad (2.1.2)$$

where $\bar{X}^T = QX^T = \begin{bmatrix} X_{\bar{R}}^T \\ X_{\bar{L}}^T \end{bmatrix}$ and $P\beta^T = \begin{bmatrix} \beta_u^T \\ \beta_l^T \end{bmatrix}$.

- 2 *BSR*: Multiplying by $S = \begin{bmatrix} B^{-1} & 0 \\ -DB^{-1} & I \end{bmatrix}$, called Back-Substitution of References

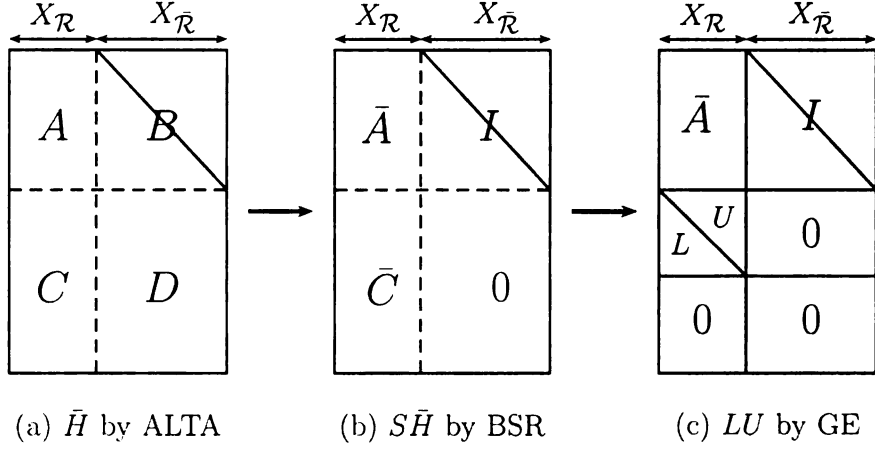


Figure 2.1. The MLDA on \bar{H}

(BSR), system (2.1.2) is transformed to

$$\begin{bmatrix} \bar{A} & I \\ \bar{C} & 0 \end{bmatrix} \begin{bmatrix} X_{\mathcal{R}}^T \\ X_{\bar{\mathcal{R}}}^T \end{bmatrix} = \begin{bmatrix} \bar{\beta}_u^T \\ \bar{\beta}_l^T \end{bmatrix} \Leftrightarrow (S\bar{H})\bar{X}^T = SP\beta^T, \quad (2.1.3)$$

where $\bar{A} = B^{-1}A$ and $\bar{C} = C - D\bar{A}$. (See **Figure. 2.1-(b)**).

3 *GE*: $X_{\mathcal{R}}$ is recovered by using on $\bar{C}X_{\mathcal{R}} = \bar{\beta}_l^T$. (See **Figure. 2.1-(c)**).

4 *Final FS (FFS)*: $X_{\bar{\mathcal{R}}}$ is recovered by $X_{\bar{\mathcal{R}}}^T = \bar{A}X_{\mathcal{R}} + \bar{\beta}_u^T$.

The core of the MLDA is in the novel combination of the ALTA and the BSR by S that reduces system (2.1.1) into a small system $\bar{C}X_{\mathcal{R}} = \bar{\beta}_l^T$ by means of a *partial GE* on \bar{H} over the columns of the triangular block $\begin{bmatrix} B \\ D \end{bmatrix}$ from the first to the last column of it. When $\bar{H} = \begin{bmatrix} B \\ D \end{bmatrix}$, the MLDA becomes the MPA. Of particular interest of the MLDA is in the ALTA on H , generated by the RSD. The prominence of Luby's RSD is not just in the perfect triangulation of H by the MPA for large block lengths n and overheads γ , but is also in the robustness of the RSD. A small perturbation on the RSD does not affect much the triangulation of H by the MPA. Furthermore, even for short n and γ close to 0, H can be permuted into an approximate triangular form \bar{H} in system (2.1.2) whose left-block $\begin{bmatrix} A \\ C \end{bmatrix}$ is very small. Thus, the GE on \bar{C} is very

efficient compared to the conventional GE on H . This advanced form of GE also can be found in [27, Ex-50.12] and the US patent [15]. Another MLDA using guessing strategies on $X_{\mathcal{R}}$ at a bit-level was developed in [17]. In the algorithm, whenever the triangulation of H (by the MPA) stops prematurely, a codeword bit, whose value has not been determined yet, is chosen and declared to be known (by assigning 1 or 0). The triangulation proceeds in this fashion, and the algorithm succeeds decoding as long as $\text{Rank}(H) = n$.

Let $q \times r$ be the matrix dimension of \bar{C} , where $q = \gamma n + r$. Note that $\text{Rank}(H) = n$ iff. $\text{Rank}(\bar{C}) = r$. If $\text{Rank}(\bar{C}) = r$, then the GE on \bar{C} can be performed by the LU -factorization algorithm **Algorithm 1.1** (see also [23, ch.7]) that returns \bar{C} in an LU -factorized form $\bar{C} = LU \begin{bmatrix} I_{r \times r} \\ 0 \end{bmatrix}$ such that L and U is a $q \times q$ lower and upper triangular matrix, respectively (see **Figure 2.1-(c)**). In fact, by the GE, L^{-1} and U^{-1} can be obtained in a product form of elementary matrices, so that

$$X_{\mathcal{R}}^T = (LU)^{-1} \bar{\beta}_l^T, \quad U^{-1} = \prod_{k=1}^r U^{(k)}, \quad L^{-1} = \prod_{k=r}^1 L^{(k)}, \quad (2.1.4)$$

where $L^{(k)}$ and $U^{(k)}$ are the elementary matrices formed by replacing the k^{th} row of $I_{q \times q}$ with the k^{th} row of L and U , respectively, and $\bar{\beta}_l$ is the one in system (2.1.3) whose symbols are associated with rows of \bar{C} . Even if $\text{Rank}(\bar{C}) < r$, free variables in $X_{\mathcal{R}}$ can be identified by the GE. Thus, system (2.1.1) can be solved for α by retransmitting the input symbols of free variables only. Otherwise, the deficiency may be further reduced by increasing a fraction of dense rows on H .

In this chapter, without explicit construction of the permuted \bar{H} in system (2.1.2), we first identify systems of the MLDA as an equivalent set of linear systems in a product form of elementary matrices via the permutations (P, Q) . Let $HQ^T = [H_{\mathcal{R}}; H_{\bar{\mathcal{R}}}]$, the rearrangement of columns of H associated with $(X_{\mathcal{R}}, X_{\bar{\mathcal{R}}})$. We identify system (2.1.2) from system (2.1.1) via (P, Q) . Then using $\bar{S} = P^T S P$, we interpret

system (2.1.3) (in the BSR) as

$$\bar{S}[H_{\mathcal{R}}; H_{\bar{\mathcal{R}}}] \bar{X}^T = \bar{S} \beta^T. \quad (2.1.5)$$

Let $\bar{H}_{\mathcal{R}} = \bar{S} H_{\mathcal{R}}$. For each k , $1 \leq k \leq q$, the k^{th} row \bar{C}_k of \bar{C} is identical to the i_k^{th} row $(\bar{H}_{\mathcal{R}})_{i_k}$ of $\bar{H}_{\mathcal{R}}$ for some i_k . Hence, we perform the LU -factorization over the row set $\{(\bar{H}_{\mathcal{R}})_{i_k}\}_{k=1}^q$ and obtain $m \times m$ matrices \bar{L} and \bar{U} that are identical to an L and U via a permutation pair (σ_r, τ_r) , respectively, and whose inverses are also in a product form of elementary matrices similar to the ones in system (2.1.4). Consequently, systems (2.1.3) and (2.1.4) together is equivalent to

$$\left[(\bar{L}\bar{U})^{-1} \cdot \bar{S} \cdot (H_{\mathcal{R}}^T) \right] \cdot \bar{X}^T = \left[(\bar{L}\bar{U})^{-1} \bar{S} \right] \cdot \beta^T. \quad (2.1.6)$$

Based on routines of the MLDA, we compute system (2.1.6) by exploiting the S-MLDA that consists of two major steps; *pre-decoding* and *post-decoding* as following.

1. *pre-decoding*:

- (a) compute the left-hand side of system (2.1.6);
- (b) discard the equations in system (2.1.1) that become null after the GE on $\bar{H}_{\mathcal{R}}$. (See null rows in the bottom of **Figure 2.1-(c)**).

2. *post-decoding*:

- (a) compute the right-hand side of system (2.1.6) by $\bar{\beta}^T := (\bar{L}\bar{U})^{-1} \bar{S} \beta^T$;
- (b) recover $X_{\mathcal{R}}$ from the $\bar{\beta}$ by looking at (σ_r, τ_r) ;
- (c) recover each x_k of $X_{\bar{\mathcal{R}}}$ using sparser of the k^{th} equations of the systems (2.1.2) and (2.1.3) as in below

$$B_k X_{\mathcal{R}}^T = \beta_k + A_k X_{\bar{\mathcal{R}}}^T \quad \text{or} \quad x_i = \bar{\beta}_k + \bar{A}_k X_{\bar{\mathcal{R}}}^T. \quad (2.1.7)$$

The major role of the *pre-decoding* step is to find out precisely n check equations, and at the same time, to compute all row operations on H that transform the H in system (2.1.1) into a form $\begin{bmatrix} \bar{A} & I \\ LU & 0 \\ 0 & 0 \end{bmatrix}$ as shown in **Figure 2.1-(c)**. Then the *post-decoding* is designed to recover $(X_{\mathcal{R}}, X_{\bar{\mathcal{R}}})$ by applying the same row operations without any redundant symbol additions on β .

For a higher probability $\Pr(\text{Rank}(H) = n)$, small fractions of dense rows are required in $\rho(x)$. Most of the dense rows, however, become null after the GE on \bar{C} . Thus, ahead of the *post-decoding* step, all of redundant rows should be discarded to avoid symbol additions of β over those rows. (See the curves in **Figure 2.3** removed by step 1b)).

\bar{A} in system (2.1.3) is not sparse in general. Furthermore, its column dimension r becomes larger as $\gamma \rightarrow 0$. However, the top part of \bar{A} is more likely sparser than the top of $[A; B]$. In fact, many rows in the top of \bar{A} are null or with degree one. On the other hand, the bottom of \bar{A} is denser than the the bottom of $[A; B]$. Hence, the alternative recovery by step 2c) by comparing the degrees $|\bar{A}_k|$ and $(|A_k| + |B_k|)$ improves the efficiency in symbol additions significantly. The improvement of the efficiency in symbol additions is presented in **Figure 2.3** (see green curves in the figure removed by the alternative recovery step 2c)).

The remainder of this chapter is organized as follows. In section 2.2, we elaborate on the systems (2.1.5) and (2.1.6) from the perspective of basic linear algebra over \mathbb{F}_2 . For exemplary pseudo-codes of the routines of the S-MLDA, see section 3.2. In section 2.3, we estimate the computational complexity of the S-MLDA in terms of the number of {sign, bit}-flips and symbol additions of β made by the *pre-decoding* and *post-decoding*, respectively. In section 2.4, we design a row-degree distribution $\rho(x)$ by supplementing a small fraction of dense rows for higher $\Pr(\text{Rank}(H) = n)$. We then analyze the rank-distribution of H by using Kovalenko's rank-distribution of random matrices [21, 22, 29] in section 2.5. In section 2.6, we present our simulation results

tested with LT codes, generated by a designed $\rho(x)$ in section 2.4 with block-lengths n , $10^3 \leq n \leq 10^4$, for the following scenarios:

1. the decoding error rates of the codes under the S-MLDA and the MPA to tell a stable overhead γ for the successful S-MLDA and MPA;
2. number of symbol additions by the *post-decoding* to tell the efficiency of the S-MLDA in symbol additions of β ;
3. fractions of references $\frac{r}{n}$ to tell the computational complexity of the *pre-decoding* in a bit-level;
4. the rank-deficiency $\eta = \dim(\text{Ker}(H))$.

We then substantiate, based on our experimental results, that a stable overhead γ for the successful S-MLDA is far better than the stable γ for the successful MPA, while the computational complexity of the S-MLDA in symbol additions maintains within few tens of n . In section 2.7, in the same scenarios 1) - 4) above, we present the performances of LT codes of short block-lengths n , $10^2 \leq n \leq 10^3$ generated by two encoding schemes:

E1) encoding by rows of a $(kn) \times n$ matrix M ;

E2) encoding by a random generation of rows of H .

Based on our experimental results, we also substantiate that, under the S-MLDA, LT codes generated by an arranged encoder matrix M also achieves a stable overhead γ for the successful S-MLDA close to 0. We then summarize the paper in section 2.8.

2.2 The Separated MLDA

We first outline several notations that are used in the remainder of the chapter. For a given $m \times n$ matrix K over \mathbb{F}_2 , we denote k_{ij} , K_i , K^j , and $|K|$ as the $(i, j)^{th}$ entry, the

i^{th} row, the j^{th} column, and the number of 1's of K , respectively. We use the notation 1_{ij} to indicate $k_{ij} = 1$. With the matrix dimension $m \times n$, let $[m] = \{1, 2, \dots, m\}$ and $[n] = \{1, 2, \dots, n\}$ the row and column index set of K , respectively. In the rest of the section, we verify the systems (2.1.5) and (2.1.6) in a product form of elementary matrices.

First, by flipping a known 1 into -1 through the diagonal extension of B , the ALTA (see [4, p644] or **Algorithm 3.1**) can be designed to obtain a set of ordered pairs $(\sigma_l, \tau_l) \subset [m] \times [n]$ such that

$$(\sigma_l, \tau_l) = (i_1, j_{r+1}) \succ \dots \succ (i_l, j_{r+l}), \quad n = l + r. \quad (2.2.1)$$

Thus, an index pair (i_s, j_{r+t}) in $\sigma_l \times \tau_l$ indicates the $(s, t)^{th}$ entry of the triangular B which is identical to $h_{i_s, j_{r+t}}$ in H . With the σ_l and τ_l , let $(\mathcal{R}, \bar{\mathcal{R}})$ and $(\mathcal{T}, \bar{\mathcal{T}})$ be the disjoint pair of $[n]$ and $[m]$, respectively, such that

$$\begin{aligned} \mathcal{R} &= \{j_1, \dots, j_r\}, \quad \bar{\mathcal{R}} = \tau_l = \{j_{r+1}, \dots, j_{r+l}\}, \\ \mathcal{T} &= \sigma_l = \{i_1, \dots, i_l\}, \quad \bar{\mathcal{T}} = \{i_{l+1}, \dots, i_m\}. \end{aligned} \quad (2.2.2)$$

With the pairs, we then set the permutations

$$\sigma : [m] \mapsto [m], \quad \sigma(i_k) = k, \quad \text{and} \quad \tau : [n] \mapsto [n], \quad \tau(j_k) = k. \quad (2.2.3)$$

The matrix representations of σ and τ , say P and Q^T , is obtainable by permuting rows of $I_{m \times m}$ and columns of $I_{n \times n}$ in the order of σ and τ , respectively (see also (1.2.5)). Associated with $(\mathcal{R}, \bar{\mathcal{R}})$, let $\bar{X} = (X_{\mathcal{R}}, X_{\bar{\mathcal{R}}})$ such that

$$X_{\mathcal{R}} = (x_{j_1}, \dots, x_{j_r}) \quad \text{and} \quad X_{\bar{\mathcal{R}}} = (x_{j_{r+1}}, \dots, x_{j_{r+l}}). \quad (2.2.4)$$

Then via (P, Q) , system (2.1.1) is permuted to system (2.1.2).

Second, multiplying by B^{-1} , the top system of (2.1.2) is brought into $X_{\bar{\mathcal{R}}}^T = \bar{A}X_{\mathcal{R}}^T + B^{-1}\beta_u^T$, then substituting the $X_{\bar{\mathcal{R}}}$ into the bottom system of (2.1.2) yields

$\bar{C}X_{\mathcal{R}}^T = B^{-1}\beta_u^T + \bar{A}\beta_l^T$ as shown in system (2.1.3). This step is the BSR in [3], and it can be accomplished by multiplying $S = \begin{bmatrix} B^{-1} & 0 \\ -DB^{-1} & I \end{bmatrix}$ to system (2.1.2) as shown in system (2.1.3). Note that $S^{-1} = \begin{bmatrix} B & 0 \\ D & I \end{bmatrix}$ and is a lower triangular matrix. Therefore, S can be decomposed as

$$S = \prod_{k=l}^1 S^{(k)} = S^{(l)}S^{(l-1)} \dots S^{(2)}S^{(1)}, \quad (2.2.5)$$

where each $S^{(k)}$ is the elementary matrix formed by replacing $(I_{m \times m})^k$ with \bar{H}^k and l is the number of columns of the left block $\begin{bmatrix} B \\ D \end{bmatrix}$. The computation of $S\bar{H}$ in system (2.1.3) is accomplished by the iteration

$$\bar{H} := S^{(k)}\bar{H}, \quad k = 1, 2, \dots, l. \quad (2.2.6)$$

Consequently, system (2.1.3) is expressed in a product form

$$\left[\left(\prod_{k=l}^1 S^{(k)} \right) \cdot \begin{bmatrix} A \\ C \end{bmatrix}; \left(\prod_{k=l}^1 S^{(k)} \right) \cdot \begin{bmatrix} B \\ D \end{bmatrix} \right] \bar{X}^T = \left(\prod_{k=l}^1 S^{(k)} \right) \cdot P\beta^T. \quad (2.2.7)$$

Notice that in the iterative BSR system above, since $S[\begin{smallmatrix} B \\ D \end{smallmatrix}]$ is always known as $\begin{bmatrix} I \\ 0 \end{bmatrix}$, the explicit computation of $S[\begin{smallmatrix} B \\ D \end{smallmatrix}]$ is redundant.

We now interpret the systems of the MLDA without explicit construction of the permuted $\bar{H} = \begin{bmatrix} A & B \\ C & D \end{bmatrix}$. First, since $\bar{H} = PHQ^T$, system (2.1.3) is expressed as

$$(SP)(HQ^T)\bar{X}^T = SP\beta^T \quad \equiv \quad (2.1.3). \quad (2.2.8)$$

Then multiplying by $P^{-1} = P^T$ in both sides,

$$(P^T SP)(HQ^T)\bar{X}^T = (P^T SP)\beta^T \quad \equiv \quad (2.2.8). \quad (2.2.9)$$

Rearranging columns of H associated with $(X_{\mathcal{R}}, X_{\bar{\mathcal{R}}})$, we then interpret the HQ^T in system (2.2.8) as $[H_{\mathcal{R}}; H_{\bar{\mathcal{R}}}]$ such that

$$H_{\mathcal{R}} = [H^{j_1}; \dots; H^{j_r}], \quad H_{\bar{\mathcal{R}}} = [H^{j_{r+1}}; \dots; H^{j_{r+l}}]. \quad (2.2.10)$$

Let $\bar{S} = P^T S P$, and let $\bar{S}^{(k)} = P^T S^{(k)} P$ formed by replacing the column $(I_{m \times m})^{ik}$ with the column $H^{j_{r+k}}$ via the k^{th} index pair (i_k, j_{r+k}) in (σ_l, τ_l) . Then by $P^{-1} = P^T$ and (2.2.5),

$$\bar{S} = P^T S P = \prod_{k=l}^1 P^T S^{(k)} P = \prod_{k=l}^1 \bar{S}^{(k)}. \quad (2.2.11)$$

Substituting the product (2.2.11) and $[H_{\mathcal{R}}; H_{\bar{\mathcal{R}}}]$ into system (2.2.9), the iterative BSR system (2.2.7) is transformed to

$$\left[\left(\prod_{k=l}^1 \bar{S}^{(k)} \right) H_{\mathcal{R}}; \left(\prod_{k=l}^1 \bar{S}^{(k)} \right) H_{\bar{\mathcal{R}}} \right] \bar{X}^T = \left(\prod_{k=l}^1 \bar{S}^{(k)} \right) \beta^T. \quad (2.2.12)$$

Note that, once S is known, $\bar{S} H_{\mathcal{R}}$ is always known as $P^T \begin{bmatrix} I \\ 0 \end{bmatrix}$, thus, the computation of $\bar{S} H_{\mathcal{R}}$ alone is enough for the computation of $\bar{S} H Q^T$ and is obtainable by the iteration

$$H_{\mathcal{R}} := \bar{S}^{(k)} H_{\mathcal{R}}, \quad k = 1, 2, \dots, l. \quad (2.2.13)$$

Let $\bar{H}_{\mathcal{R}} = \bar{S} H_{\mathcal{R}}$. Then since $\bar{C}_k = (\bar{H}_{\mathcal{R}})_{i_k}$ via $\sigma(i_k) = k$ for all $i_k \in \bar{\mathcal{T}}$, we replace the LU -factorization on \bar{C} as the factorization over the row set $\{(\bar{H}_{\mathcal{R}})_{i_k}\}_{k=1}^q$ with additional steps. At each pivoting round k of GE:

1. When a pivot $1_{s_k, t_k}$ is chosen for the k^{th} diagonal of LU , store the index pair $(s_k, t_k) \in \bar{\mathcal{T}} \times \mathcal{R}$ into (σ_r, τ_r) .
2. For each pivoted $1_{s, t_k}$ whose row index s is in $\bar{\mathcal{T}}$, flip it into $-1_{s, t_k}$ in $(\bar{H}_{\mathcal{R}})_s$.
3. Then discard the row $(\bar{H}_{\mathcal{R}})_{s_k}$ from the $\{(\bar{H}_{\mathcal{R}})_{i_k}\}$ updated up to that round.

After the GE, if $\text{Rank}(\bar{C}) = r$ then

$$(\sigma_r, \tau_r) = (s_1, t_1) \succ \dots \succ (s_r, t_r) \subset \bar{\mathcal{T}} \times \mathcal{R}. \quad (2.2.14)$$

Thus, for each $s_k \in \sigma_r$ and $t_j \in \tau_r$, the $-1_{s_k, t_j}$ or $1_{s_k, t_j}$ in $\bar{H}_{\mathcal{R}}$ can be identified as the 1_{kj} of an L or U in system (2.1.4), respectively. Notice that, if $\text{Rank}(\bar{C}) < r$ then free variables are precisely the x_j 's whose index j is in $\mathcal{R} \setminus \tau_r$.

Let $\bar{\beta}^T = \bar{S}\beta^T$. We note that, each $1_{k,j}$ in $L^{(k)}$ or $U^{(k)}$ in system (2.1.4) corresponds to the symbol addition $(\bar{\beta}_l)_k := (\bar{\beta}_l)_k + (\bar{\beta}_l)_j$, where $\bar{\beta}_l$ is the one in system (2.1.3). Similarly, the $-1_{s_k,t_j}$ or $1_{s_k,t_j}$ in $(\bar{H}_{\mathcal{R}})_{s_k}$ corresponds to the symbol addition $\bar{\beta}_{s_k} := \bar{\beta}_{s_k} + \bar{\beta}_{s_j}$ via the k^{th} row index s_k and the j^{th} row index s_j in σ_r . Therefore, each $L^{(k)}$ (or $U^{(k)}$) in system (2.1.4) corresponds to the $m \times m$ elementary matrix $\bar{L}^{(k)}$ (or $\bar{U}^{(k)}$), formed by placing those $-1_{s_k,t_j}$ (or $1_{s_k,t_j}$, respectively) as the $1_{s_k,s_j}$ in the row $(I_{m \times m})_{s_k}$. In this way, via (σ_r, τ_r) , the product form of L^{-1} and U^{-1} in system (2.1.4) corresponds to the products

$$\bar{L}^{-1} = \prod_{k=r}^1 \bar{L}^{(k)} \Leftrightarrow L^{-1}, \quad \bar{U}^{-1} = \prod_{k=1}^r \bar{U}^{(k)} \Leftrightarrow U^{-1}. \quad (2.2.15)$$

Consequently, via (P, Q) and (σ_r, τ_r) , the systems (2.1.3) and (2.1.4) together, as shown in system (2.1.6), is combined as

$$\bar{U}^{-1} \bar{L}^{-1} \bar{S} [H_{\mathcal{R}}; H_{\bar{\mathcal{R}}}] \cdot \bar{X}^T = \bar{U}^{-1} \bar{L}^{-1} \bar{S} \beta^T. \quad (2.2.16)$$

where \bar{S} , \bar{U}^{-1} , and \bar{L}^{-1} are in a product form of elementary matrices as shown in (2.2.15) and (2.2.11). In other words, this means that, by representing the (σ_r, τ_r) as the permutation matrix pair (P_r, Q_r) on $\bar{H}_{\mathcal{R}}$,

$$\bar{U}^{-1} \bar{L}^{-1} \bar{H}_{\mathcal{R}} = P_r^T P^T \begin{bmatrix} \bar{A} \\ I_{r \times r} \\ 0 \end{bmatrix} Q_r. \quad (2.2.17)$$

In the *pre-decoding* step in section 2.1, the step 1a) can be summarized as the following diagram

$$H \xrightarrow[\substack{ALTA \\ (\sigma_l, \tau_l)}]{} H_{\mathcal{R}}, (\sigma_l, \tau_l) \xrightarrow[\substack{BSR \\ (2.2.13)}]{} \bar{H}_{\mathcal{R}} \xrightarrow[\substack{GE \\ (2.2.15)}]{} \bar{L} \bar{U}. \quad (2.2.18)$$

Then all redundant check equations $H_i X^T = \beta_i$, whose row index i is in $\bar{\mathcal{T}} \setminus \sigma_r$, are discarded at the step 1b). The right-hand side of system (2.2.16) is computed by the *post-decoding* step as described in section 2.1. We note that, in (2.1.7) of the

alternative recovery step, each \bar{A}_k , A_k , and B_k is identical to $(\bar{H}_{\mathcal{R}})_{i_k}$, $(H_{\mathcal{R}})_{i_k}$, and $(H_{\mathcal{R}})_{i_k}$ via $\sigma(i_k) = k$, respectively.

2.3 Computational Complexities of the MLDA

In practice, several supplemental instructions are indispensable for the efficiency of the S-MLDA, for examples, degree reductions through the rounds of the ALTA and updating the (σ_l, τ_l) in the ALTA, and so on. Since a symbol addition or a {sign, bit}-flip accompanies those instructions within a constant time, we estimate the computational complexity of the S-MLDA by counting the number of {sign, bit}-flips and symbol additions encountered during the *pre-decoding* and *post-decoding*, respectively. Through the section, let us assume that $\mathcal{R} \neq \emptyset$.

We first estimate the complexity of the *pre-decoding* stage. By the ALTA, every 1 in H is flipped into -1 to obtain a permutation index pair (σ_l, τ_l) in equation (2.2.1). Hence the number of sign-flips by the ALTA is $|H|$. While computing $\bar{H}_{\mathcal{R}}$ by the BSR iteration (2.2.13), each 1 in $H_{\mathcal{R}}$, except in the diagonal of B , corresponds to one addition of rows in $H_{\mathcal{R}}$. Thus, the number of bit-flips by the BSR is proportional to $r(|H_{\mathcal{R}}| - n + r)$, where r is the number of columns of $H_{\mathcal{R}}$. By the GE on $\bar{H}_{\mathcal{R}}$, when a pivot $1_{s_k, t_k}$ is chosen at round k , the row $(\bar{H}_{\mathcal{R}})_{s_k}$ is added to the rows of \bar{T} of which the t_k^{th} column entry is 1. Since $|(\bar{H}_{\mathcal{R}})_{s_k}| \leq (r - k)$ and $|\bar{T}| \leq (r - k) + \gamma n$ at the round k , the number of {sign, bit}-flips together is less than

$$\sum_{k=1}^r |(\bar{H}_{\mathcal{R}})_{s_k}| |\bar{T}| = c \sum_{k=1}^r k(k + \gamma n) \quad (2.3.1)$$

for some constant c . In fact, our simulations exhibit that, at round k ,

$$|(\bar{H}_{\mathcal{R}})_{s_k}| \leq \frac{r - k}{2} \quad \text{and} \quad |\bar{T}| \leq \frac{(r - k) + \gamma n}{2}, \quad (2.3.2)$$

and thus, $c \leq \frac{1}{4}$. Hence in aggregate, the number of {sign, bit}-flips from the *pre-*

decoding step can be upper estimated as

$$\underbrace{|H|}_{ALTA} + \underbrace{r|H_{\bar{\mathcal{R}}}|}_{BSR} + \underbrace{\frac{1}{2}\gamma c n r^2 + \frac{1}{3}c r^3}_{GE}. \quad (2.3.3)$$

When $r \approx \epsilon n$ for large n and small $\epsilon > 0$, the estimate (2.3.3) may be dominated by either $(\gamma\epsilon^2)n^3$ or $(\epsilon^3)n^3$. Hence, in this case, the computational complexity of the *pre-decoding* step in a bit-level is $O(n^3)$ but with small constant factors ϵ^3 and $\gamma\epsilon^2$ (see [3, p. 4]).

Let us now estimate the number of symbol additions made by the *post-decoding* stage. We recall the removal of all redundant rows (including most of the dense rows) by the step 1a) of the *pre-decoding* and the alternative recovery step by 2c) of the *post-decoding* in section 2.1. At the initialization step, since only the $\bar{\beta}_i$'s, whose row index i is in $\sigma_l \cup \sigma_r$, are copied from β and since $|\sigma_l \cup \sigma_r| = n$, precisely, n copies of β_i 's are made for $\bar{\beta}$. For the computation of $\bar{\beta}^T = \bar{U}^{-1}\bar{L}^{-1}\bar{S}\beta^T$ in system (2.1.6), due to the removal of redundant dense rows, significantly less than $\frac{|H_{\bar{\mathcal{R}}}|}{1+\gamma}$ and less than r^2 in number of symbol additions of $\bar{\beta}$ are made by \bar{S} and $\bar{U}^{-1}\bar{L}^{-1}$, respectively. Now let $d_i = \min\{|\bar{H}_{\mathcal{R}}|_i, |H_i|\}$. By the alternative recovery in (2.1.7), a total of $d = \sum_{k=1}^l d_{i_k}$ symbol additions are made for the recovery of $X_{\bar{\mathcal{R}}}$, that is also significantly less than $\frac{|H|}{1+\gamma}$. Hence in total, the number of symbol additions from the *post-decoding* is significantly less than

$$n + \frac{|H| + |H_{\bar{\mathcal{R}}}|}{1 + \gamma} + r^2. \quad (2.3.4)$$

We now assume that the original MLDA in [3] is used for the recovery of X . (Recall 2. *BSR*, 3. *GE*, and 4. *FFS* of the original MLDA section 2.1). In the *pre-decoding* step, only the {sign, bit}-flips that correspond to one row-addition constitute symbol additions of β or $\bar{\beta}$. First, a total of $|H_{\bar{\mathcal{R}}}|$ by the *BSR* and less than $\sum_{k=1}^r (\gamma n + k)$ row additions by the *GE* on $\bar{H}_{\mathcal{R}}$, are made for the recovery of $X_{\mathcal{R}}$. Then for the recovery of $X_{\bar{\mathcal{R}}}$, a total of $|\bar{A}|$ symbol additions are made by the *FFS*. Hence in aggregate, the

number of symbol additions by the original MLDA is upper estimated as

$$|H_{\mathcal{R}}| + |\bar{A}| + r(\gamma n + r). \quad (2.3.5)$$

In **Figure 2.3**, we show that, due to the heavy density $|\bar{A}|$ and dense rows in H , the estimate (2.3.5) is much larger than the one made by the *post-decoding* stage of the S-MLDA. (Compare black and red curves in **Figure 2.3**).

2.4 Degree Distribution Design with Dense Rows

In this section, we first go over LT degree distribution design with Mackay's simple recursion formula in [27, ch. 9]. A fine analysis of the design in a continuous frame can be consulted in Shokrollahi's works in [14].

With a designed distribution $\mu(x)$, we then alter $\mu(x)$ into a $\rho(x)$ by supplementing a small fraction of dense rows of degree $\frac{n}{2}$. We then analyze rank properties of an $m \times n$ random H generated by $\rho(x)$. By doing so, we provide a simple way for the appropriate value of $\rho_{n/2}$.

Let us first consider the diagonal extension of the MPA on H in [4, p. 644]. Starting from $H(0) := H$, at the t^{th} extension step, let $1_{i_t, j_t}$ be selected as the t^{th} diagonal entry of $L = PHQ^T$, and let $H(t)$ denote the residual matrix by discarding the j_t^{th} column and the i_t^{th} row from $H(t-1)$.

Lemma 2.4.1. *Let $h_t(d)$ denote the expected number of rows of degree d in $H(t)$. Then after the $(t+1)^{th}$ diagonal extension step, $H(t+1)$ has*

$$h_{t+1}(d) = \begin{cases} (h_t(1) - 1) \left(1 - \frac{1}{n-t}\right) + \frac{2h_t(2)}{n-t}, & \text{if } d = 1, \\ h_t(d) \left(1 - \frac{d}{n-t}\right) + \frac{(d+1)h_t(d+1)}{n-t}, & \text{otherwise} \end{cases}. \quad (2.4.1)$$

Proof. Once a 1 is chosen for the $(t+1)^{th}$ diagonal entry of L , say $1_{i_{t+1}, j_{t+1}}$ is chosen for the diagonal entry, the i_{t+1}^{th} row counted in $h_t(1)$ is automatically discarded,

thus, there remains $h_t(1) - 1$ rows of degree 1 in $H(t)$. When the j_{t+1}^{th} column is discarded from $H(t)$, each of the $h_t(1) - 1$ remained rows of degree 1 has 1 in the discarded column with probability $\frac{1}{n-t}$. Hence, about $(h_t(1) - 1)\frac{1}{n-t}$ rows of degree 1 become rows of degree 0, and therefore, about $(h_t(1) - 1)(1 - \frac{1}{n-t})$ rows of degree 1 in $H(t)$ still remain as rows of degree 1 in $H(t+1)$. Now, a row of degree 2 in $H(t)$ has a 1 in the discarded column $H^{j_{t+1}}$ with probability $\frac{2}{n-t}$. This implies that about $h_t(2)\frac{2}{n-t}$ rows of degree 2 in $H(t)$ become rows of degree 1 in $H(t+1)$ after the diagonal extension. Therefore,

$$h_{t+1}(1) = (h_t(1) - 1) \left(1 - \frac{1}{n-t}\right) + \frac{2h_t(2)}{n-t}. \quad (2.4.2)$$

In case of $d \geq 2$, when the j_{t+1}^{th} column is discarded from $H(t)$, a row of degree d in $H(t)$ has a 1 in the column with probability $\frac{d}{n-t}$. This implies that $h_t(d)(1 - \frac{d}{n-t})$ rows of degree d in $H(t)$ still remain as rows of degree d in $H(t+1)$, at the same time, about $h_t(d+1)\frac{d+1}{n-t}$ rows of degree $d+1$ in $H(t)$ become rows of degree d in $H(t+1)$. Then the sum of the two cases asserts the $h_{t+1}(d)$ in (2.4.1). \square

Theorem 2.4.1. *Expecting the ripple size by $h_t(1) = S + 1$ for all $t = 1, 2, \dots, n$,*

$$h_t(d) = \frac{n-t}{d(d-1)} + \frac{S}{d}, \quad d > 1. \quad (2.4.3)$$

In particular, when $t = 0$,

$$h_0(d) = \begin{cases} S + 1, & \text{for } d = 1, \\ \frac{n}{d(d-1)} + \frac{S}{d}, & \text{otherwise} \end{cases}. \quad (2.4.4)$$

Proof. Setting by $h_t(1) = h_{t+1}(1) = S + 1$, the first recursion in (2.4.1) yields $h_t(2) = \frac{n-t}{2} + \frac{S}{2}$. Let us assume that $h_t(d) = \frac{n-t}{d(d-1)} + \frac{S}{d}$ for all $d \geq 3$. Plugging in $h_{t+1}(d) = \frac{n-(t+1)}{d(d-1)} + \frac{S}{d}$ into the second recursion in (2.4.1) then simplifying it into $\frac{n-t}{d} + S = (d+1)h_t(d+1)$ yields the estimate (2.4.3). (2.4.4) is obvious by $t = 0$. \square

Let us now set m the number rows of H by

$$m = \sum_{d=1}^n h_0(d) \approx n + S(1 + \ln(n)). \quad (2.4.5)$$

Thus, from the right-hand side of (2.4.5), $\gamma \approx \frac{S(1+\ln(n))}{n}$. Then normalizing by m , we derive the RSD as

$$\mu(x) = \sum_{d=1}^n \mu_d x^d, \quad \text{where} \quad \mu(d) = \frac{h_0(d)}{m}. \quad (2.4.6)$$

The RSD in [2] can be thought of as a special case of (2.4.4) with $S = c \ln(n/\delta) \sqrt{n}$ and $h_0(1) = S + 1$ as below

$$h_0(d) = \begin{cases} \frac{n}{d(d-1)} + \frac{S}{d}, & 1 < d < \frac{n}{S}, \\ \frac{n}{d(d-1)} + S \ln(S/\delta), & d = \frac{n}{S}, \\ \frac{n}{d(d-1)}, & \text{otherwise.} \end{cases} \quad (2.4.7)$$

Theorem-17 in the original LT paper [2] guarantees that when an $m \times n$ matrix H is randomly generated by the RSD in (2.4.7) with $m \approx n + S(\ln(S/\delta) + \ln(n/S))$, the LT system (2.1.1) has its unique solution and can be solved by the MPA with probability at least $1 - \delta$.

The triangulation of H by the MPA depends on the constraint $h_t(1) \geq 1$ for all t . In particular, for short block lengths n , γ should be increased for the successful MPA with high probability. For a stable ripple design, see Shokrollahi's Raptor paper [14, pp. 21-22]. In contrast, the success of the S-MLDA depends on $\text{Rank}(\bar{C}) = r$. The efficiency of the S-MLDA in computational complexity may be degraded due to the GE on \bar{C} . However, the fraction of references $\frac{r}{n}$ is quite small for all $\gamma > 0$, and hence, the GE on \bar{C} in a bit-level may not be a major drawback to the efficiency. (See **Figure 2.4**)

Let H be an $m \times n$ random matrix over \mathbb{F}_2 generated by a row-degree distribution

$\rho(x) = \sum \rho_d x^d$. We first estimate the column-degree distribution of H , say $\lambda(x) = \sum_{d=0}^m \lambda_d x^d$. Then with λ_0 , the fraction of null columns of H , we estimate a lower bound of the density $|H|$ to keep λ_0 small.

Lemma 2.4.2. *Let $m = (1 + \gamma)n$. Then the column-degree distribution of H can be estimated as*

$$\lambda(x) = \prod_{d=1}^n \left(\left(1 - \frac{d}{n} \right) + \frac{d}{n} x \right)^{(1+\gamma)n\rho_d}. \quad (2.4.8)$$

Proof. For a given row H_i , let $d_i = |H_i|$. Since H_i is randomly chosen from $\binom{n}{d_i}$ possible choices, entries of H_i follow the distribution

$$f_{d_i}(x) = \left(1 - \frac{d_i}{n} \right) + \frac{d_i}{n} x, \quad \frac{d_i}{n} = \Pr(h_{ij} = 1). \quad (2.4.9)$$

Let $H(k)$ denote the sub-matrix that consists of the first k rows of H . Obviously, the column-degree distribution of $H(1)$ is $f_{d_1}(x)$. Assume that columns of $H(k)$ follow the degree distribution $\prod_{i=1}^k f_{d_i}(x) = \sum_{j=0}^k a_j x^j$. Now let $H(k+1)$ be an expansion of $H(k)$ supplemented with a random row of degree d_{k+1} . Then, a column of $H(k+1)$ was a column of degree j in $H(k)$ with probability a_j and it becomes a column of degree $j+1$ in $H(k+1)$ with probability $\frac{d_{k+1}}{n}$. This case happens with the probability $a_j \frac{d_{k+1}}{n}$. Similarly, a column of $H(k+1)$ was in degree $j+1$ in $H(k)$ with probability a_{j+1} and it remains as a column of degree $j+1$ in $H(k+1)$ again with probability $(1 - \frac{d_{k+1}}{n})$. This case happens with probability $a_{j+1}(1 - \frac{d_{k+1}}{n})$. Since these two cases are all cases for a column in $H(k+1)$ being degree $j+1$, a column of $H(k+1)$ has degree $j+1$ with probability

$$a_j \left(\frac{d_{k+1}}{n} \right) + a_{j+1} \left(1 - \frac{d_{k+1}}{n} \right), \quad (2.4.10)$$

which is exactly the coefficient of the term x^{j+1} of the product

$$f_{d_{k+1}}(x) \sum_{j=0}^k a_j x^j = \prod_{i=1}^{k+1} f_{d_i}(x). \quad (2.4.11)$$

Hence, by the induction, H has the column-degree distribution

$$\lambda(x) = \prod_{i=1}^m f_{d_i}(x). \quad (2.4.12)$$

Now, H follows the row-degree distribution $\rho(x)$ with $m = (1 + \gamma)n$, and the number of rows of degree d is $(1 + \gamma)n\rho_d$. Therefore, rearranging the product (2.4.12) with respect to d then substituting (2.4.9) into (2.4.12) asserts (2.4.8). \square

Theorem 2.4.2. *Let a_r be the average row-degree of H . Then the fraction of null columns of H , λ_0 , is estimated as*

$$\lambda_0 = \prod_{d=1}^n (1 - d/n)^{n(1+\gamma)\rho_d} \approx e^{-a_r(1+\gamma)}. \quad (2.4.13)$$

Proof. By **Lemma 2.4.2**, the fraction of columns of degree k is given as $\lambda_k = \frac{\lambda^{(k)}(0)}{k!}$, and hence, the product form in (2.4.13) is clear by $\lambda_0 = \lambda(0)$. For sufficiently large n , since

$$(1 - d/n)^{n(1+\gamma)\rho_d} \approx e^{-d\rho_d n(1+\gamma)}, \quad (2.4.14)$$

the product can be approximated as $e^{-(1+\gamma)\sum d\rho_d}$. Then since $a_r = \sum d\rho_d$, we conclude (2.4.13). \square

For the unique solution of system (2.1.1) with a destined δ , $0 < \delta < 1$, the number of null columns in estimate is given as $n\lambda_0 = ne^{-(1+\gamma)a_r}$, and thus, a row-degree distribution $\rho(x)$ should hold the inequality $ne^{-(1+\gamma)a_r} < \delta < 1$. Therefore

$$a_r \geq \frac{\ln(n/\delta)}{(1+\gamma)} \quad \text{or} \quad |H| \geq n \ln(n/\delta). \quad (2.4.15)$$

To meet this inequality, we note that dense fractions in $\rho(x)$ are indispensable.

Most of the fractions of the RSD $\mu(x)$ in (2.4.6) are too small to get $\lfloor n\mu_d \rfloor \geq 1$. For an example, with short block lengths n in a few thousands, $\lfloor n\mu_d \rfloor = 0$ for $d > 60$. We recall that dense fractions are indispensable to meet the inequality (2.4.15). For

this reason, we alter the $\mu(x)$ into $\rho(x)$ having 3 parts:

$$\rho(x) = \sum_{d \in \mathcal{D}_1} \rho_d x^d + \sum_{d \in \mathcal{D}_2} \rho_d x^d + \rho_{n/2} x^{n/2}, \quad (2.4.16)$$

in the following manner:

1. $\rho_d \approx \mu_d$ for $d \in \mathcal{D}_1 = \{1, \dots, 20\}$;
2. $0.001 \leq \rho_d \leq 0.01$ for $d \in \mathcal{D}_2$, where \mathcal{D}_2 consists of few tens of degrees d such that $20 \leq d \leq 400$;
3. $0.001 \leq \rho_{n/2} \leq 0.005$.

The idea of the $\rho(x)$ is as following. First of all, the fraction $\rho_{n/2}$ is for higher $\Pr(\text{Rank}(H) = n)$. Second, \mathcal{D}_1 is for smaller $\frac{r}{n}$ with the constraint $\sum_{d \in \mathcal{D}_1} \mu_d \geq 0.9$ based on equation (2.4.6). Lastly, based on the density constraint (2.4.15), \mathcal{D}_2 is aimed to hold $\sum_{\mathcal{D}_1 \cup \mathcal{D}_2} d \rho_d \geq \frac{\ln(n/\delta)}{1+\gamma}$. Thus, hopefully, the $[A, B]$ of \bar{H} in system (2.1.2) consists of rows of degree d in $\mathcal{D}_1 \cup \mathcal{D}_2$ only.

One would set $\rho(x) := \rho_{\mathcal{D}_1 \cup \mathcal{D}_2}(x) / \rho_{\mathcal{D}_1 \cup \mathcal{D}_2}(0)$ that meets the inequality (2.4.15) with a small δ . It seems to the authors that, even with a large γ , an $m \times n$ random matrix H generated by the $\rho(x)$ does not achieve the probability $\Pr(\text{Rank}(H) = n)$ close to 1. For an example, in **Figure 2.10**, rank deficient cases happen just one or two times (out of 1000 random constructions of H by the $\rho(x)$) with deficiency $\dim(\text{Ker}(H)) = 1$ or 2. In contrast, when the $\rho(x)$ is supplemented with $\rho_{n/2} := 0.005$, the small deficiencies are gracefully removed up to $1 + \gamma \approx 1.008$.

Since $|H|$ is increased by $\frac{n^2}{2}(1 + \gamma)\rho_{n/2}$ with the fraction $\rho_{n/2}$ alone, a slight increment on $\rho_{n/2}$ may cause $|H|$ much heavier than desired ones. Let us now present a simple way for an appropriate value of $\rho_{n/2}$ with an estimated $\dim(\text{Ker}(H))$ through simulations. Let H be an $m \times n$ random matrix generated by the $\rho_{\mathcal{D}_1 \cup \mathcal{D}_2}(x) = \sum_{\mathcal{D}_1 \cup \mathcal{D}_2} \rho_d x^d$. For a given $V \in \mathbb{F}_2^n$, let $V^\perp = \{X \in \mathbb{F}_2^n | V \cdot X = 0\}$ the complement

space of V . With the row space $\text{RS}(H) = \{\sum_{i \in I} H_i | \forall I \subset [m]\}$, let $\text{Rank}(H) = \dim(\text{RS}(H))$.

Theorem 2.4.3. *Let \tilde{H} be an expansion of H supplemented with $m_{n/2}$ random rows of degree $\frac{n}{2}$. Then,*

$$\Pr\left(\text{Rank}(\tilde{H}) < n\right) \leq \frac{1}{2^{m_{n/2}-\eta}}, \quad \text{where } \eta = \dim(\text{Ker}(H)). \quad (2.4.17)$$

Proof. We first note that, from basic linear algebra, $\text{RS}(H) \subset V^\perp$ iff. $V \in \text{Ker}(H)$, and $\text{Ker}(\tilde{H}) \subset \text{Ker}(H)$. We also note that $\text{Rank}(\tilde{H}) < n$ iff. $\text{RS}(\tilde{H}) \subset V^\perp$ for some nonzero $V \in \text{Ker}(H)$. For each nonzero $V \in \text{Ker}(H)$, since supplemented dense rows are chosen in random,

$$\Pr(\text{RS}(\tilde{H}) \subset V^\perp) \leq 2^{-m_{n/2}}. \quad (2.4.18)$$

Then since $|\text{Ker}(H)| = 2^\eta$, the inequality (2.4.17) is clear by the sum of the probabilities (2.4.18) for all $V \in \text{Ker}(H)$. \square

The author of the thesis is not aware of any closed form of mathematical estimates for η . Nonetheless, for a given $\rho_{\mathcal{D}_1 \cup \mathcal{D}_2}(x) = \sum_{d \in \mathcal{D}_1 \cup \mathcal{D}_2} \rho_d x^d$, η can be estimated by extensive simulations of the S-MLDA very efficiently. Then by increasing $m_{n/2}$ in (2.4.17), one would achieve $\eta = 0$ with high probability. Thus, $\rho_{n/2}$ may be assigned by $\rho_{n/2} := \frac{m_{n/2}}{(1+\gamma)n}$ with an appropriate γ .

2.5 The Rank Distributions of \tilde{H}

In this section, we introduce Kovalenko's rank distribution theorem over finite fields [21,22,29]. We then derive a finite version of the rank distribution of the supplemented matrix \tilde{H} in **Theorem 2.4.3**.

We first introduce the limit version of the rank distribution theorem. Let H be

an $(n + k) \times n$ random matrix over a finite field \mathbb{F}_q with the density constraint

$$\frac{\ln(n) + x}{n} \leq \Pr(h_{ij} \neq 0) \leq 1 - \frac{\ln(n) + x}{n}, \quad (2.5.1)$$

where $x \rightarrow \infty$ arbitrarily slowly. Kovalenko and Levitskaya [21] showed that, as $n \rightarrow \infty$, for any fixed integers k and s with $k + s \geq 0$,

$$\lim_{n \rightarrow \infty} \Pr(\text{Rank}(H) = n - s) = \frac{1}{q^{s(k+s)}} \frac{\prod_{i=s+1}^{\infty} (1 - \frac{1}{q^i})}{\prod_{i=1}^{k+s} (1 - \frac{1}{q^i})}. \quad (2.5.2)$$

For an example, with $(q = 2, k = 30, s = 0)$,

$$\lim_{n \rightarrow \infty} \Pr(\text{Rank}(H) = n) = \prod_{i=31}^{\infty} \left(1 - \frac{1}{2^i}\right) \quad (2.5.3)$$

which is very close to 1. In [22], Cooper further improved that, under the assumption that H has no zero columns, the distribution (2.5.2) is also true when the condition in (2.5.1) is weakened as $x \rightarrow -\infty$. Note that the distribution (2.5.2) is not directly applicable to a random H generated by a $\rho(x)$ in (2.4.16), because both the row and column-degree distributions of H may not meet the density constraint in (2.5.1).

Let H be an $m \times n$ random matrix over \mathbb{F}_2 generated by the $\rho_{\mathcal{D}_1 \cup \mathcal{D}_2}(x)$ with rank-deficiency $\eta = \dim(\text{Ker}(H))$. Starting from $H(0) = H$, let $H(k)$ be an expansion of $H(k-1)$ supplemented with a random row of degree $\frac{n}{2}$. Now let $\zeta_{k,\eta}(\omega) = \Pr(\text{Rank}(H(k)) = (n - \eta) + \omega)$ where $0 \leq \omega \leq \eta$.

Proposition 2.5.1. *The rank-distribution of $H(k+1)$ follows*

$$\zeta_{k+1,\eta}(\omega) = \zeta_{k,\eta}(\omega - 1) \left(1 - \frac{1}{2^{\eta - \omega + 1}}\right) + \zeta_{k,\eta}(\omega) \frac{1}{2^{\eta - \omega}}. \quad (2.5.4)$$

Proof. Only two cases are possible for $\text{Rank}(H(k+1)) = n - (\eta - \omega)$; either $\text{Rank}(H(k)) = n - (\eta - \omega + 1)$ or $\text{Rank}(H(k+1)) = n - (\eta - \omega)$. If $\text{Rank}(H(k)) = n - (\eta - \omega + 1)$, then $\text{Rank}(H(k+1)) = n - \eta + \omega$ iff. the supplemented row is not in $\text{Ker}(H(k))$. Since $\dim(\text{Ker}(H(k))) = \eta - \omega + 1$ in this case, $\text{Rank}(H(k+1)) =$

$n - (\eta - \omega)$ with probability $\zeta_{k,\eta}(\omega - 1) \left(1 - \frac{1}{2^{\eta - \omega + 1}}\right)$. If $\text{Rank}(H(k)) = n - (\eta - \omega)$, then $\text{Rank}(H(k + 1)) = n - (\eta - \omega)$ iff. the supplemented row is in $\text{Ker}(H(k))$. Since $\dim(\text{Ker}(H(k))) = \eta - \omega$ in this case, $\text{Rank}(H(k + 1)) = n - (\eta - \omega)$ with probability $\zeta_{k,\eta}(\omega) \frac{1}{2^{\eta - \omega}}$. Then the sum of the two probabilities asserts (2.5.4). \square

We now derive the solution of (2.5.4), a finite version of rank-distribution of $H(k)$.

Lemma 2.5.1. *Let $\omega \leq k$, $\omega \leq \eta$, and let $l = k - \omega$. Then*

$$\zeta_{k,\eta}(\omega) = \prod_{i=1}^{\omega} \left(1 - \frac{1}{2^{\eta+1-i}}\right) \frac{1}{2^{l(\eta-\omega)}} \cdot S(\omega, l), \quad (2.5.5)$$

where

$$S(\omega, l) = \sum_{i_1=0}^{\omega} \sum_{i_2=0}^{i_1} \cdots \sum_{i_l=0}^{i_{l-1}} \frac{1}{2^{i_1+i_2+\cdots+i_l}}. \quad (2.5.6)$$

Proof. Although the same arguments of **Theorem 3.2.1** in [29, p. 126] can be used for the probability (2.5.5), we derive the probability (2.5.5) by using (2.5.4) and the mathematical induction. If $\omega = 0$ or $l = 0$, then clearly

$$\zeta_{k,\eta}(0) = \frac{1}{2^{k\eta}} \quad \text{and} \quad \zeta_{k,\eta}(k) = \prod_{i=1}^k \left(1 - \frac{1}{2^{\eta+1-i}}\right). \quad (2.5.7)$$

Assume that (2.5.5) is true for k and ω . We show that by using (2.5.4), the distribution (2.5.5) also holds for $k + 1$ and ω . Let $l = k - \omega$. First, $\zeta_{k,\eta}(\omega - 1) \left(1 - \frac{1}{2^{\eta - \omega + 1}}\right)$ in (2.5.4) is expressed as

$$\prod_{i=1}^{\omega} \left(1 - \frac{1}{2^{\eta+1-i}}\right) \frac{1}{2^{(l+1)(\eta-\omega+1)}} \cdot S(\omega - 1, l + 1), \quad (2.5.8)$$

and $S(\omega - 1, l + 1)$ in the above can be expressed as

$$S(\omega - 1, l + 1) = \sum_{\omega \geq i_1 \geq \cdots \geq i_{l+1} \geq 1} 2^{-\sum_{s=1}^{l+1} i_s}. \quad (2.5.9)$$

Similarly, the $\zeta_{k,\eta}(\omega) \frac{1}{2^{\eta-\omega}}$ in (2.5.4) is expressed as

$$\prod_{i=1}^{\omega} \left(1 - \frac{1}{2^{\eta+1-i}}\right) \frac{1}{2^{(l+1)(\eta-\omega)}} \cdot S(\omega, l), \quad (2.5.10)$$

where $S(\omega, l)$ can be expressed as

$$S(\omega, l) = \sum_{\omega \geq i_1 \geq \dots \geq i_l \geq (i_{l+1}=0)} 2^{-\sum_{s=1}^{l+1} i_s}. \quad (2.5.11)$$

It can then be seen that, from (2.5.9) and (2.5.11),

$$S(\omega - 1, l + 1) + S(\omega, l) = S(\omega, l + 1). \quad (2.5.12)$$

Therefore, the sum of (2.5.8) and (2.5.10) gives $\zeta_{k+1,\eta}(\omega)$ as

$$\prod_{i=1}^{\omega} \left(1 - \frac{1}{2^{\eta+1-i}}\right) \frac{1}{2^{(l+1)(\eta-\omega)}} S(\omega, l + 1), \quad (2.5.13)$$

where

$$S(\omega, l + 1) = \sum_{\omega \geq i_1 \geq \dots \geq i_{l+1} \geq 0} \frac{1}{2^{i_1 + i_2 + \dots + i_{l+1}}}. \quad (2.5.14)$$

□

Theorem 2.5.1. *With the same notations in Lemma 2.5.1, let $\tilde{H} = H(k)$. Then with $\omega = \eta$ and $k \geq \eta$, as a particular case of the distribution (2.5.5),*

$$\Pr(\text{Rank}(\tilde{H}) = n) = \prod_{i=1}^{\eta} \left(1 - \frac{1}{2^{\eta+1-i}}\right) \cdot S(\eta, l). \quad (2.5.15)$$

The author is not aware of any simpler form of $S(\eta, l)$. Furthermore, it is difficult to compute $S(\eta, l)$, because it requires an l -dimensional array for the computation of the multiple sum. Hence in practice, once a rank-deficiency η is estimated from simulations, the lower-bound (2.4.17) is more practical for the value of $\rho_n/2$. Nonetheless, it is straightforward to see that, by **Theorem 2.4.17**, **Theorem 2.5.1**, and by

$$l = k - \eta,$$

$$S(\eta, l) \geq \left(1 - \frac{1}{2^l}\right) \prod_{i=1}^{\eta} \left(1 - \frac{1}{2^{\eta+1-i}}\right)^{-1}. \quad (2.5.16)$$

Remark 2.5.1. So far, we assumed that \tilde{H} is an expansion of H by supplementing random rows of degree $\frac{n}{2}$. In fact, **Lemma 2.5.1** is also true when supplemented rows are randomly chosen from 2^n possible choices (see the proof of **Theorem 3.2.1** in [29]). We may anticipate that the distribution (2.5.5) also holds, if supplemented rows meet the density constraint (2.5.1). Heuristically, the anticipation is reasonable in the following sense. Let $\text{Rank}(H(k)) = n - (\eta - \omega)$. By GE, $H(k)$ can be transformed to a row-equivalent form $[A; I]$ where I is the identity matrix of size $n - (\eta - \omega)$. Then pivoting the supplemented row of $H(k+1)$ with diagonal entries of the I , let the supplemented row be transformed to $(A_{k+1}; 0)$, and thus, $H(k+1)$ is now row-equivalent to $\begin{bmatrix} A & I \\ A_{k+1} & 0 \end{bmatrix}$. Then $\text{Rank}(H(k+1)) = 1 + \text{Rank}(H(k))$ as long as $A_{k+1} \neq 0$. Now, if we assume that A_{k+1} is a random vector in $\mathbb{F}_2^{\eta-\omega}$ where $\eta - \omega = \dim(\text{Ker}(H(k)))$, then $A_{k+1} \neq 0$ with probability $1 - \frac{1}{2^{\eta-\omega}}$.

Remark 2.5.2. When $q = 2$, the limit distribution (2.5.2) can be derived from **Lemma 2.5.1** by setting $H(0) = \emptyset$ (so that $\eta = n$) and $\omega = n - s$. To see this, we first rearrange the sum $S(\omega, l)$ in (2.5.5) as

$$S(\omega, l) = \sum_{i_l=0}^{\omega} 2^{-i_l} \sum_{i_{l-1}=i_l}^{\omega} 2^{-i_{l-1}} \dots \sum_{i_1=i_2}^{\omega} 2^{-i_1}. \quad (2.5.17)$$

Since $\omega \rightarrow \infty$ as $n \rightarrow \infty$, the rearrangement can be simplified as a product form

$$\begin{aligned} \lim_{n \rightarrow \infty} S(\omega, l) &= \left(1 - \frac{1}{2}\right)^{-1} \sum_{i_l=0}^{\infty} 2^{-i_l} \dots \sum_{i_3 \leq i_2}^{\infty} 2^{-2i_2} \\ &\quad \vdots \quad \quad \quad \vdots \quad \quad \quad \vdots \\ &= \left(1 - \frac{1}{2}\right)^{-1} \dots \left(1 - \frac{1}{2^{l-1}}\right)^{-1} \sum_{i_l=0}^{\infty} 2^{-li_l} \\ &= \prod_{i=1}^l \left(1 - \frac{1}{2^i}\right)^{-1}. \end{aligned} \quad (2.5.18)$$

Then plugging in the product (2.5.18) into the distribution (2.5.13) gives the limit distribution (2.5.2). (See also [29, p130]).

We now close this section by providing a formal approach to an estimate of $|\text{Ker}(H)|$. Let H be an $m \times n$ random matrix over \mathbb{F}_2 generated by a $\rho(x)$ in (2.4.6), and let X be a nonzero random vector in \mathbb{F}_2^n with $|X| = k$. Then the number of solutions of system (2.1.1) is exactly $|\text{Ker}(H)|$. Now, for a given random row H_i of degree d_i ,

$$\Pr(H_i \cdot X = 0) = \frac{1 + \left(1 - \frac{2k}{n}\right)^{d_i}}{2}. \quad (2.5.19)$$

Then since H has $m\rho_d$ rows of degree d and each of them is generated independently from all other rows,

$$\Pr(X \in \text{Ker}(H)) = \prod_{\rho_d \neq 0} \left(\frac{1 + (1 - 2k/n)^d}{2} \right)^{m\rho_d}. \quad (2.5.20)$$

Next, X can be chosen in total $\binom{n}{k}$ different ways, therefore, the expectation of $|\text{Ker}(H)|$ is given as

$$\frac{1}{2^m} \sum_{k=0}^n \binom{n}{k} \prod_{\rho_d \neq 0} \left(1 + (1 - 2k/n)^d \right)^{m\rho_d}. \quad (2.5.21)$$

If a fine approximation of (2.5.21) is possible, then an appropriate value of $\rho_{n/2}$ can be assigned for $\Pr(|\text{Ker}(H)| = 1) \approx 1$ without much difficulty. It seems to the author that, the equation (2.5.21) is more likely blown up to ∞ by a small increment on ρ_d .

2.6 Simulation Results

In this section, we provide our experimental results simulated with LT codes of block-lengths n , $10^3 \leq n \leq 10^4$, generated by a row-degree distribution $\rho(x)$ in (2.4.6) supplemented with the dense fraction $\rho_{n/2} = 0.005$. Particularly in **Figure 2.2**, we substantiate that, under the S-MLDA, our designed LT codes can achieve the

\mathcal{D}_1	$(\rho_d)_{d=1}^6 = (0.015, 0.47, 0.164, 0.074, 0.047, 0.032)$ $(\rho_d)_{d=7}^{12} = (0.023, 0.017, 0.013, 0.011, 0.009, 0.008)$
\mathcal{D}_2	$\rho_d = 0.004$ for $13 \leq d \leq 20$ $\rho_d = 0.002$ for $21 \leq d \leq 30$ $\rho_d = 0.001$ for $31 \leq d \leq 70$ $\rho_d = 0.004$ for $d = 71, 72, 141, 260, 350$
\mathcal{D}_3	$\rho_{n/2} = 0.005$

Table 2.1. The row-degree distribution $\rho(x)$

performance in overhead γ (for the successful S-MLDA) close to 0. In **Figure 2.4**, we also provide the fraction of references $r_f = \frac{r}{n}$ which shows the computational efficiency of the *pre-decoding* with respect to $r = \epsilon n$. (Recall the small constant factor ϵ in (2.3.3) at p. 51).

The spectrum of our simulation is as follows. First, we generate a distribution $\mu(x)$ in (2.4.6) with $S = 15$ and $n = 10^3$. Based on (2.4.16) and the conditions 1) - 3) therein, we alter the $\mu(x)$ into the $\rho(x)$ as shown in **Table 2.1**. Second, we tested LT codes under the S-MLDA for the 10 block lengths from $n = 10^3$ to $n = 10 \cdot 10^3$. For each fixed n , a row dimension m is increased by 1 from $m := n$ up to $m := (1.2)n$. Then for each matrix dimension $m \times n$, an $m \times n$ random matrix H by the row-degree distribution $\rho(x)$ is constructed 100 times by generating its rows using Mersenne Twister algorithm [16] on $[n]$. Then for each instance of H , the S-MLDA is tested with a nonzero input symbol vector α in $(\mathbb{F}_2^s)^n$ for the following scenarios:

1. decoding failure rate of codes under the S-MLDA and the MPA;
2. number of symbol additions by the S-MLDA and the original MLDA in [3];
3. fractions of references $\frac{r}{n}$;
4. rank-deficiency $\eta = \dim(\text{Ker}(H))$.

In **Figure 2.2**, for each fixed n , a black and gray curve $\text{DFR} = f(1 + \gamma)$ shows the decoding failure rate (DFR) of codes under the S-MLDA and the MPA, respectively.

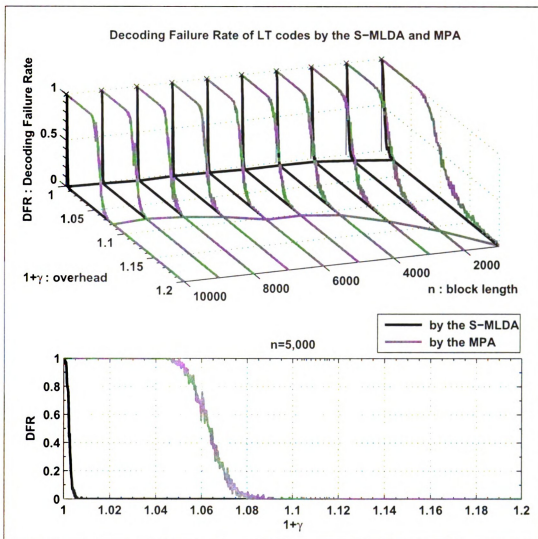


Figure 2.2: Performances of LT Codes in Decoding Failure Rates (DFR) (under the S-MLDA (black curves 1) and the MPA (gray curves 2). The codes are generated by the $\rho(x)$ in Table 2.1 over the block lengths from $n = 1,000$ to $10,000$

For examples, when $n = 5,000$ and $1 + \gamma = 1.01$ with 100 trials of code constructions (see the bottom figure in Figure 2.2), the gray point $(1.01, 1)$ indicates that the MPA never succeeds for the recovery of α , in contrast, the black point $(1.01, 0)$ indicates that the S-MLDA never fails to recover α . We observe that, for any $1 + \gamma > 1.007$, the DFR by the S-MLDA is 0. Even if $1 \leq 1 + \gamma \leq 1.007$, our simulations also exhibit that $\text{Rank}(H) \geq n - 12$ (see the bottom figure in Figure 2.5). Therefore, a small

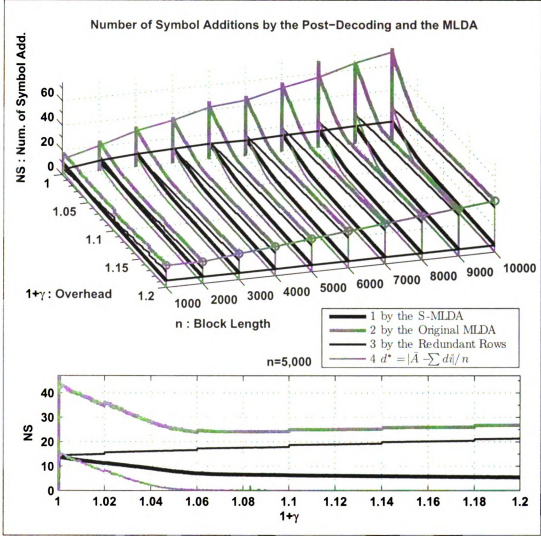


Figure 2.3. Number of Symbol Additions by the Post-Decoding and the MLDA in [3]

increment in $\rho_{n/2}$ (or γ) based on the inequality (2.4.17) or the rank-distribution (2.5.15) may increase $\Pr(\text{Rank}(H) = n)$ very close to 1. Also notice that, by the MPA, $\text{DFR} > 0$ for any $1 + \gamma < 1.08$ and n . Therefore, the desirable γ for the successful decoding of codes by the MPA should be greater than 0.08.

In Figure 2.3, for each fixed n , a black and gray curve 1 and 2 represents the number of symbol additions divided by n , denoted as NS, made from the *post-decoding* and the original MLDA in [3], respectively. Similarly, the black and gray curve 3 and

4 indicates the NS made by the redundant equations $H_i X^T = \beta_i$ whose index i is in $\bar{\mathcal{T}} \setminus \sigma_r$ and by the $d^* = ||\bar{A}| - \sum_{k=1}^l d_i|/n$, as shown in p. 52, respectively. (Recall the alternative recovery and the removal of redundant rows in section 2.1). When $n = 5,000$ and $1 + \gamma \approx 1.01$, for examples, the point $(1.01, 12)$ on the black curve indicates that, approximately, $12 \cdot 5,000$ symbol additions is made by the *post-decoding*, and the point $(1.01, 39)$ on the gray curve corresponds to $39 \cdot 5,000$ symbol additions made by the original MLDA in [3]. Similarly, the black point $(1.01, 15)$ on the curve 3 and the gray point $(1.01, 11)$ on the curve 4 corresponds to $15 \cdot 5,000$ and $(|\bar{A}| - d) = 11 \cdot 5,000$ redundant symbol additions, respectively. It can be observed that the black curve 1 is mainly contributed by sparse fractions of degree d in $\mathcal{D}_1 \cup \mathcal{D}_2$, however, the gray curve 2 is mainly contributed by the dense fraction $\rho_{n/2} = 0.005$ and $|\bar{A}|$. From the figure, observe that the gray curve 2 is much larger than the black curve 1.

In **Figure 2.4**, for each fixed block length n , a black curve shows the fraction of references (FR), $\text{FR} = \frac{r}{n}$, where r is the column dimension of $H_{\mathcal{R}}$ (or $[\frac{A}{C}]$). When $n = 5,000$ and $1 + \gamma = 1.01$, for instance, the point $(1.01, 0.025)$ indicates that the \bar{C} in system (2.1.3) has its matrix dimension about 150×125 which is very small compared to the matrix dimension of H , $5,050 \times 5,000$. Notice that for any γ and n , $\text{FR} \leq 0.041$ that substantiates the very small constant factor in the complexity of the GE on \bar{C} (see also [3, p. 4]). (To see the lower complexity, compute the upper bounds (2.3.3) and (2.3.4) with $r = 0.041n$).

In **Figure 2.5**, each curve represents the rank-deficiency $\eta = \dim(\text{Ker}(H))$, or the number of free variables under the S-MLDA. For an example, when $n = 5,000$ (see the bottom figure in **Figure 2.5**), the point $(1 + \gamma = 1, \eta = 11)$ indicates that the S-MLDA fails to recover α with $\text{DFR} = 1$ (see the bottom figure in **Figure 2.2**) and the rank deficiency is approximately 11. Even when $\eta > 0$, the GE on $\bar{H}_{\mathcal{R}}$ (or \bar{C}) identifies the all the free variables, and thus, α is obtainable by retransmitting the

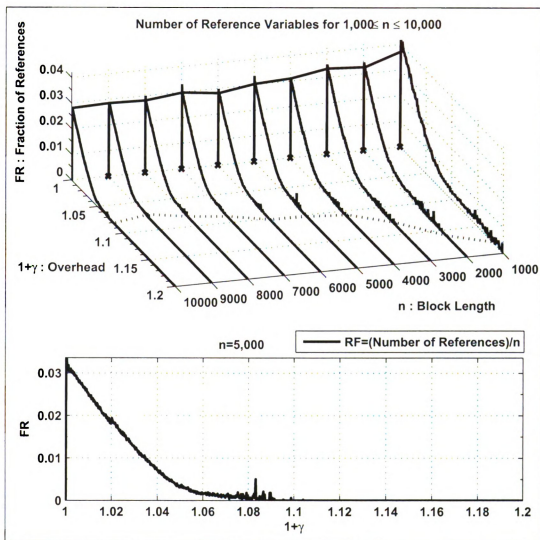


Figure 2.4. Fraction of References

input symbols of free variables only.

Figure 2.6 shows how the dense fraction $\rho_{n/2} = 0.005$ gracefully improves the DFR of the S-MLDA to 0 with $\Pr(\text{Rank}(H) = n)$ close to 1. In the bottom figure, a black curve shows the DFR of the S-MLDA on H , generated by $\rho(x)$. Similarly, the gray curve shows the DFR of the S-MLDA on H , generated by $\rho_{\mathcal{D}_1 \cup \mathcal{D}_2}(x)$. In both cases, ρ_d 's are taken from Table 2.1. For the simulation, starting from $m := n$ to $m := (1.2)n$, a row dimension m is increased by 1, and for each matrix dimension

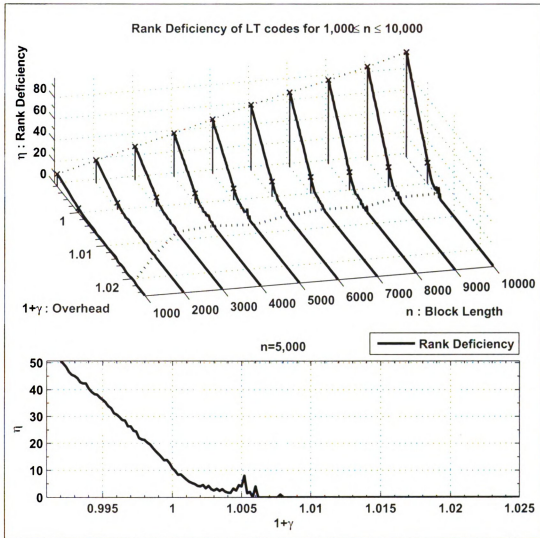


Figure 2.5. The Rank-Deficiency $\eta = \dim(\text{Ker}(H))$ (or the Number of Free Variables)

$m \times n$, an $m \times n$ matrix H is generated 1000 times. Then the S-MLDA is tested for each instance of H and $\alpha \neq 0$. When H is constructed by $\rho_{\mathcal{D}_1 \cup \mathcal{D}_2}$ (see the gray curves), although the DFR and η is small, rank-deficient cases occur constantly up to $1 + \gamma = 1.1$, then sporadically as $1 + \gamma$ increases to 1.2. Contrastingly, when H is supplemented with $\rho_{n/2} = 0.005$, the small deficiencies are gracefully removed (i.e. $\eta = 0$) up to $1 + \gamma \leq 1.008$ (see the black curves). From the bottom figure, observe that the DFR of the S-MLDA around $1 + \gamma = 1.0076$ decreases to 0 dramatically

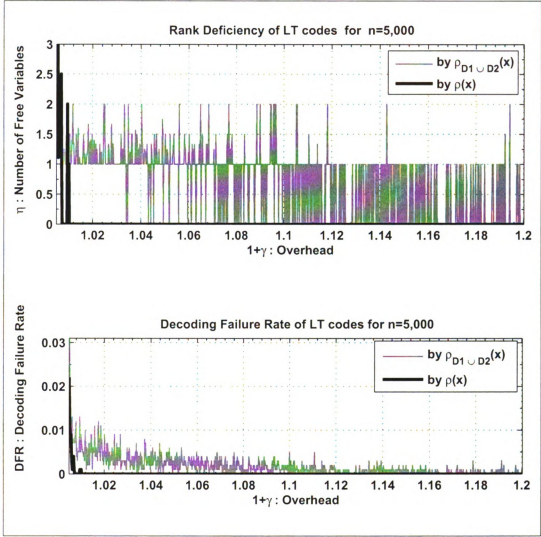


Figure 2.6. Rank Deficiency (top figure) and DFR (bottom figure) of LT codes of $n = 5,000$, generated by $\rho(x)$ and $\rho_{D_1 \cup D_2}$

by a slight increment on γ .

2.7 LT Codes of Short Block Lengths From an Arranged Encoder Matrix

In this section, we present performances of LT codes of short block lengths n , $10^2 \leq n \leq 10^3$. For higher $\Pr(\text{Rank}(H) = n)$, we design LT codes with the larger dense

fraction $\rho_{n/2} = 0.082$. Particularly, we substantiate that, under the S-MLDA, LT codes generated by an arranged encoder matrix M can achieve a stable overhead γ for the successful S-MLDA close to 0, while the S-MLDA maintains its complexity of decoding in symbol additions within few tens of n .

When H is designed by the RSD with short block lengths n within several hundreds, we (the author Ki-Moon Lee and Hayder Radha) observed that the nonzero rank deficiency $\eta = \text{Ker}(H)$ is feasibly happened but is less than a few. A small deficiency η can be removed by using the pre-coding strategies as in Raptor codes [14] but with a small degradation in overheads γ . In contrast, supplementing dense fractions in $\mu(x)$ can increase $\Pr(\text{Rank}(H) = n)$ rapidly without extra costs in γ .

In the earlier work in section 2.6, we designed LT codes with dense rows that fit for the block lengths n , $10^3 \leq n \leq 10^4$, under the S-MDLA. Particularly, a very small dense fraction $\rho_{n/2} = 0.005$ was supplemented to $\mu(x)$, and a stable γ for the successful S-MLDA was less than 0.01 for n a few thousands. With shorter block lengths n in several hundreds, however, it seems to us that much larger $\rho_{n/2}$ is required for a stable γ . In this section, we design a distribution $\rho(x)$ with $\rho_{n/2} = 0.082$.

Dense rows may cause a nontrivial drawback in communicating H to a decoder. Note that, for every instance of α , a decoder should identify H by using the same random generator of an encoder. Otherwise, each H_i should be directly delivered to a receiver attached on syndrome symbols of β . In both cases, due to dense rows, the cost in communicating H may not be trivial. Dense rows may also degrade the computational efficiency of the MLDA in [3] seriously. Nonetheless, even with the γ very close to 0, H can be approximate lower triangulated by the ALTA in [3, 4]. Furthermore, most of the dense rows become redundant. Therefore, those redundant rows should be identified so that symbol additions over the redundant rows can be removed by the S-MLDA.

In this section, we design LT codes for short block lengths n in two perspectives.

First, we alter a designed RSD $\mu(x)$ into a $\rho(x) = \sum \rho_d x^d$ by supplementing a dense fraction $\rho_{n/2}$ for the higher $\Pr(\text{Rank}(H) = n)$. Thus, even for short block lengths n and γ close to 0, a check matrix H of system (2.1.1) may have its full column rank n with high probability. Second, to communicate H to a decoder efficiently, we use a $(kn) \times n$ encoder matrix M over \mathbb{F}_2 whose row-degree distribution follows a designed $\rho(x)$. Therefore (at a receiver end), an LT decoder can quickly identify H by reading rows of M without extra cost in communicating H to a decoder.

In this encoding scheme, however, a row-degree distributions of H may be deviated from the designed $\rho(x)$, the distribution of the encoder M , seriously. Therefore, the random features of the original encoding scheme should be limited. In particular, a stable overhead γ of codes for the successful MPA may be degraded seriously. Nonetheless, our simulation exhibits that the degradation does not affect the performance in γ under the S-MLDA. (Compare the black and gray curves in **Figure 2.7**). Due to the fixed M , the fixed block-length n could be also a drawback to the flexibility of block lengths n . However, if changes in n are not so severe, then this constraint can be negotiated by using shortening techniques with null symbols on α and resizing the symbol-size s .

In this section, focusing on LT codes generated by an arranged encoder matrix M , we simulate LT codes of 10 block lengths n from $n = 100$ to 1000 in the following spectrum. First, we construct a row-degree distribution $\tilde{\rho}(x)$ in the following way:

1. A $\mu(x)$ in (2.4.6) is generated with $S = 15$ and $n = 10^3$.
2. Then fractions of $\rho(x)$ in (2.4.16) are given as:
 - (a) $\rho_d = \mu_d$ where $d \in \mathcal{D}_1 = \{1, 2, \dots, 20\}$ (for ρ_d , see **Table 2.1** at p. 64);
 - (b) $\rho_{60} = 0.02$ with $\mathcal{D}_2 = \{60\}$;
 - (c) $\rho_{n/2} = 0.082$.
3. Then $\rho(x)$ is normalized as $\tilde{\rho}(x) = \rho(x)/\rho(0)$.

Second, we arrange a $(5n) \times n$ encoder matrix M by using PEG algorithm [13] with the row-degree distribution $\tilde{\rho}(x)$. Then an $m \times n$ matrix H is generated in two ways:

E1) Rows of H are chosen in random from M .

E2) Rows of H are randomly generated by $\tilde{\rho}(x)$ using Mersenne Twister Algorithm [16] on $[n]$.

Then a syndrome symbol β_i is generated by $\beta_i = H_i \cdot \alpha^T$ and transmitted over BEC.

In both encoding schemes E1 and E2, for each fixed n :

1. the row dimension m is increased by 1 from $m := n$ up to $m := (1.3)n$;
2. for each matrix dimension $m \times n$, an $m \times n$ matrix H is constructed 1,000 times;
3. for each instance of H and $\alpha \neq 0$, the S-MLDA is tested.

Let us now present and compare their d_{err} and computational complexities based on our extensive simulation results.

In **Figure 2.7**, for each fixed n , a black curve 1 and a gray curve 2 represent the DFR of codes by the S-MLDA and the MPA, respectively, on H constructed by E1. Similarly, a black curve 3 and a gray curve 4 are the DFR made by the S-MLDA and the MPA, respectively, on H constructed by E2. When $n = 500$ and $1 + \gamma = 1.02$ in the bottom figure, for an example, the point $(1.02, 1)$ on the gray curves 2 and 4 indicates that the MPA never succeeds for the recovery of α . In contrast, the point $(1.02, 0)$ on the black curves 1 and 3 indicates that the S-MLDA never fails for the recovery of α with 1,000 constructions of H . It can be observed that, for each n , the DFR of the S-MLDA on H , generated by E1, is slightly better than the DFR of the S-MLDA on H , generated by E2. In contrast, as $1 + \gamma$ grows to 1.3, the DFR of the MPA on H , generated by E2, is better than the DFR on H , generated by E1. This substantiates that, under the S-MLDA, LT codes by an arranged encoder matrix M can also achieve the performance in γ for the successful S-MLDA close to 0, when M

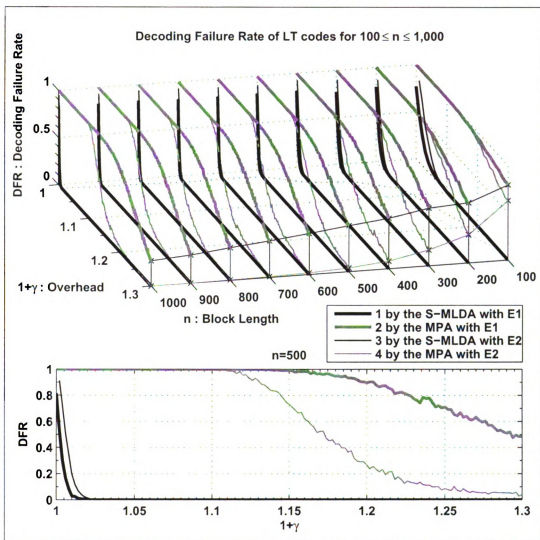


Figure 2.7. Performances in Decoding Error Rates under the S-MLDA and the MPA

is supplemented with a small fraction of dense rows. However, the performance γ for the successful MPA may be degraded seriously.

In **Figure 2.8**, for each fixed n , each curve represents the number of symbol additions divided by n , denoted as NS in the figure, by the S-MLDA and the MLDA:

1. A black curve 1 represents the NS made by the *post-decoding* step of the S-MLDA;
2. A gray curve 2 is the NS made by the original MLDA in [3];

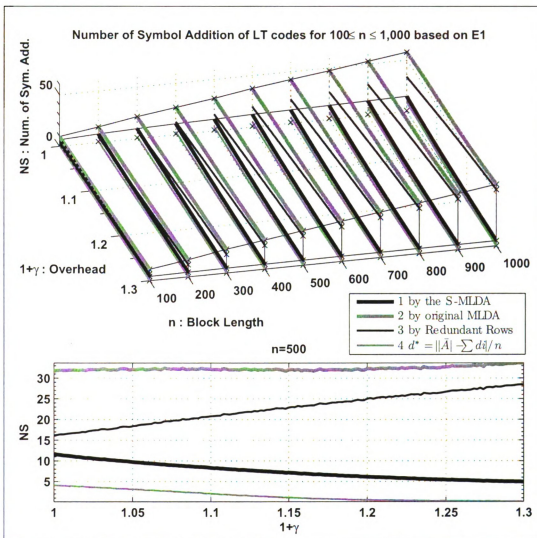


Figure 2.8. Number of Symbol Additions made by the post-decoding and the original MLDA based on the encoding scheme E1

3. A black curve 3 is the NS made from redundant rows (recall the removal of all redundant rows in the *pre-decoding*);
4. Lastly, a gray curve 4 shows the NS by the difference $d^* = \frac{||\tilde{A}|| - d}{n}$ (recall the alternative recovery in section 2.3).

When $n = 500$ and $1 + \gamma = 1.15$ (see the bottom figure), for an example, the point $(1 + \gamma = 1.15, n_s = 7)$ on the black curve 1 and the gray point $(1.15, 35)$ on the

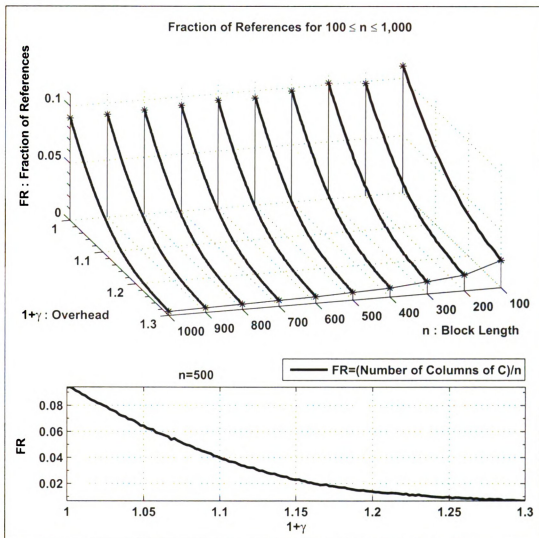


Figure 2.9. Fraction of References based on the Encoding Scheme E1

gray curve 2 indicates that about $7 \cdot 500$ and $35 \cdot 500$ symbol additions is made by the *post-decoding* and by the original MLDA, respectively. As γ grows, the NS by the *post-decoding* step is mainly contributed by the sparse fractions of $\tilde{\rho}(x)$ and a few dense rows of degree $\frac{n}{2}$. Observe that, for any instance of n and γ , the NS by the *post-decoding* is less than 17. In contrast, the NS by the original MLDA is mainly come from the dense fraction $\tilde{\rho}_{n/2}$. Due to the dense fractions, notice that, as n increases, the NS by the redundant rows (the black curves 3) far exceeds the NS made by the

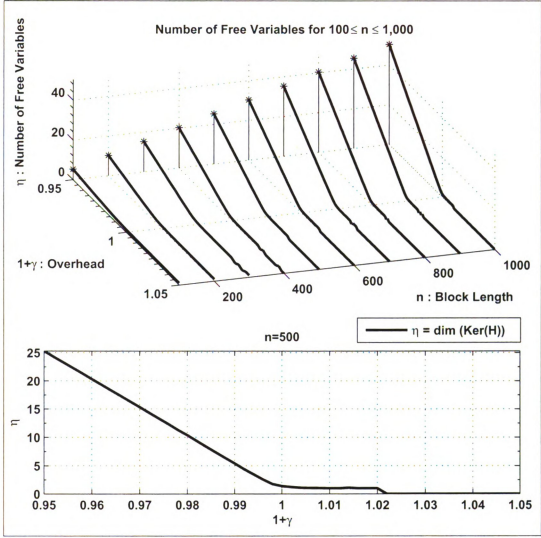


Figure 2.10. Number of Free Variables based on the Encoding Scheme E1

post-decoding.

In **Figure 2.9**, for each fixed n , a black curve indicates the fraction of references, denoted as FR in the figure, $\text{FR} = \frac{r}{n}$. When $n = 500$ and $1 + \gamma = 1.1$, for instance, the point $(1 + \gamma = 1.1, r_f = 0.04)$ says that \tilde{C} has its matrix dimension about 20×70 that is much smaller than 550×500 , the dimension of H . It can be seen that, for any instance of $1 + \gamma$ and n , $\text{FR} \leq 0.12$. Thus, the performance of the GE on \tilde{C} in computational complexity is actually very efficient.

In **Figure 2.10**, for each fixed n , a black curve η represents the number of free variables under the S-MLDA. For an example, when $n = 500$, the point $(1 + \gamma = 1, \eta = 1.7)$ (see the bottom figure in **Figure 2.10**) says that the deficiency in rank is approximately 1.7. From the figure, even the case when $m < n$, it should be noticed that $\eta \leq (n - m) + 3$. We recall that, if $\eta > 0$, the GE on \tilde{C} identifies the η free variables, and thus, α is obtainable by retransmitting symbols of free variables only.

2.8 Conclusions

In section 2.2, we introduce the S-MLDA as an advanced form of the MLDA in [3]. In section 2.3, we then estimate the complexity of the S-MLDA which is very efficient compared to the original MLDA. In section 2.4, we present a simple design of LT degree distributions with a small fraction of dense rows. We then demonstrate rank properties of a random H (including Kovalenko's rank-distribution), generated by our designed row-degree distribution $\rho(x)$. Simulation results in terms of code performances in both stable overheads γ and number of symbol additions are presented in section 2.6. Through the simulation, we provide the evidences that LT codes of block lengths n from $n = 1,000$ to $10,000$ can achieve stable overhead γ for the successful S-MLDA very close to 0, while the S-MLDA maintains its computational complexity in symbol addition within few tens of n . Lastly, in section 2.7, we also present experimental evidences which substantiate that, for short block lengths n , $100 \leq n \leq 1,000$, LT codes from an arranged encoder matrix M can also achieve performance in γ close to 0, when M is supplemented with small fraction of dense rows.

CHAPTER 3

The Maximum-Likelihood Decoding Algorithms of LDPC Codes

In this chapter, the same S-MLDA developed for LT codes in section 2.2 is applied for the decoding of BEC based LDPC codes. Clear improvements based on the S-MLDA over current LDPC decoding algorithms is demonstrated. We also present experimental results which demonstrate that, under the S-MLDA, LDPC codes can achieve performance in erasure rate p (for the successfull S-MLDA) very close to 0, while the algorithm maintains the computational complexity in symbol additions within few tens of block lengths n .

3.1 Introduction and Backgrounds

Typically in BEC based LDPC codes, as defined in **Definition 1.3.1**, an LDPC code $C(H)$ is the kernel space

$$\text{Ker}(H) = \{\alpha \in (\mathbb{F}_2^s)^n \mid H \cdot \alpha^T = 0\}, \quad (3.1.1)$$

where H is an $m \times n$ matrix over \mathbb{F}_2 . In general, for any given $\alpha_I \in (\mathbb{F}_2^s)^k$, a codeword α is generated in a systematic form $\alpha = (\alpha_I; \alpha_P)$ such that $\alpha_P = L_{m \times m}^{-1} S_{m \times k} \alpha_I^T$ as shown in (1.3.4). When α is transmitted over a BEC, the overall codeword vector α is expressed as $\alpha = (\alpha_{\bar{e}}, \alpha_e)$ where $\alpha_{\bar{e}}$ and α_e represents the received and the lost

part of α , respectively. Let n_e and $n_{\bar{e}}$ denote the number of symbols of α_e and $\alpha_{\bar{e}}$, respectively, and let $X = \alpha_e$. Associating the columns of H with the expression $(\alpha_{\bar{e}}, X)$, let $[N; M]$ be the rearrangement of H . Then the kernel space constraint $H\alpha^T = 0$ is now expressed as the consistent linear system

$$MX^T = \beta^T \quad \text{where} \quad \beta^T = N\alpha_{\bar{e}}^T \quad (3.1.2)$$

Obviously, the system has the unique solution α_e iff. $\text{Rank}(M) = n_e$.

When H is designed with a good degree sequence, for examples the sequences in [4, 13], then the unique solution of the system (3.1.2) can be solved by the MPA [5]. For short block lengths n , however, the successful triangulation of M by the MPA is not guaranteed as the erasure rate $p = \frac{n_e}{n}$ approaches to the ideal limit $1 - R = \frac{m}{n}$, where $R = \frac{k}{n}$ and $m = n - k$. (See the DFR of codes by the MPA in **Figure 3.1**).

Once the initial system (3.1.2) is identified, the problem of solving the system is same as the one of solving the system (2.1.1) of decoding LT codes, except that the M in (3.1.2) consists of columns of H in (3.1.1) and β is formed by $\beta^T = N\alpha_{\bar{e}}^T$. Therefore, replacing the LT check matrix H in system (2.1.1) with M , the ALTA designed for the H is directly applicable to the M in system (3.1.2). After the ALTA on M , replacing $HQ^T = [H_{\mathcal{R}}; H_{\bar{\mathcal{R}}}]$ in system (2.1.6) with $MQ^T = [M_{\mathcal{R}}; M_{\bar{\mathcal{R}}}]$ and applying the same BSR and GE developed in section 2.2, the S-MLDA system (2.1.6) for LT codes can be modeled for the decoding of LDPC codes as following

$$(\bar{L}\bar{U})^{-1}\bar{S} \cdot [M_{\mathcal{R}}; M_{\bar{\mathcal{R}}}]\bar{X}^T = (\bar{L}\bar{U})^{-1}\bar{S} \cdot \beta^T. \quad \Leftrightarrow \quad (2.1.6) \quad (3.1.3)$$

Based on the MLDA [3] and the system (3.1.3), the same *pre-decoding* and *post-decoding* step developed in section 2.1 and section 2.2 are directly applicable for solving system (3.1.3).

Although the S-MLDA for both LT and LDPC codes are almost same, the contributions to the efficiency in number of symbol additions by the *pre-* and *post-decoding*

are quite different. In LT codes, the improvement in symbol additions with β contributed by the step 1b) of the *pre-decoding* (or the removal of redundant equations) is significant, because most of dense rows become null after the GE. On the other hand, the improvement by the alternative recovery in step 2c) is relatively small, because the fraction of reference $\frac{r}{n}$ is quite small. In LDPC codes, contrastingly, the removal of redundant equations by the step 1b) is less significant than the one for LT codes, because, in general, the M in system (3.1.2) has no dense rows. On the other hand, due to the large reference fractions $\frac{r}{n}$ (or the number of columns of $\begin{bmatrix} A \\ C \end{bmatrix}$ in system (2.1.2)), $|\bar{A}|$ is much larger than $|A| + |B|$. Therefore, in LDPC codes, the alternative recovery by the step 2c) is indispensable for the efficiency of the *post-decoding*. The serious degradation of the efficiency in symbol additions when \bar{A} alone is used for the FFS is presented in **Figure 3.2**.

Another significant difference between LT and LDPC codes is in the time-efficiency of the initialization step of the S-MLDA. In LDPC codes, a fixed H is used for every instance of α , and thus, the decoder can quickly setup the initial system (3.1.2) by reading the columns of H . In contrast, based on the original LT transmission scheme, for every instance of a received encoding symbol vector β , rows of H should be generated by using the same random generator of an encoder to setup the system (2.1.1). This random generation of H , as a matter of fact, results in a nontrivial drawback to the time-efficiency of the S-MLDA. Otherwise, rows of H should be transmitted to receivers attached on syndrome symbols of β that requires nontrivial costs in symbol-size.

In LDPC codes, particularly when H in (3.1.1) is designed with capacity approaching degree sequences, for an example the tornado sequence in [5], the a stable erasure rate $p = \frac{n_e}{n}$ for the successful S-MLDA is very close to the ideal limit $\frac{m}{n}$. This implies that, with high probability, the S-MLDA can recover the lost symbol vector $X = \alpha_e$ as long as it acquires more than $(1 - p)n$ symbols which is very close to k ,

the number of symbols of the information part α_I . Due to the fixed code rate $R = \frac{k}{n}$, however, the number of symbols of a systematic part α_I is constrained by a fixed k . Thus, depending on the size of source data I , say $|I|$, the fixed code rate R could be a serious drawback in LDPC codes based transmission scheme. Nonetheless, assuming that a symbol size s is flexible to choose, this rate constraint can be negotiated by selecting an appropriate symbol-size s . For an example, s can be chosen by $s \approx \frac{|I|}{k}$. Then the rate constraint may be further improved, if necessary, by plugging in null symbols into a systematic part α_I , called *shortening codes*.

The remainder of the chapter is focused on the following subjects. In section 3.2, using the same arguments of the S-MLDA for LT codes developed in section 2.2, we derive the S-MLDA system (3.1.3) for the decoding of LDPC codes. In this section, we also provide exemplary pseudo-codes for routines of the S-MLDA. In section 3.3, with the same manners in section 2.3, we estimate the computational complexities of the S-MLDA with respect to the number of {sign, bit}-flips and symbol additions made by the *pre-* and *post-decoding* step, respectively. We also compare the number of symbol additions made by the *post-decoding* step and the original MLDA in [3]. In section 3.4, we present the simulation results tested with $\frac{1}{2}$ -rate PEG codes [13] under the S-MLDA for 10 block lengths n from $n = 2,000$ to $20,000$ by $2,000$, for the performances of codes for the following scenarios:

1. the performances of the S-MLDA and the MPA in decoding failure rate;
2. the complexity of the *post-decoding* in symbol additions;
3. the fraction of references $\frac{r}{n}$ to tell the complexity of the GE on \bar{C} ;
4. the rank-deficiencies $\eta = \dim(\text{Ker}(M))$.

We then summarize the chapter in section 3.5.

3.2 The S-MLDA Design with LDPC Codes

In this section, the S-MLDA system (3.1.3) is clarified in detail for the decoding of LDPC codes. For each routine of the S-MLDA, an exemplary pseudo-code is also provided. Corresponding to the expression $(\alpha_{\bar{e}}, X)$ in system (3.1.2), we denote \mathcal{E} and $\bar{\mathcal{E}}$ as the index set of X and $\alpha_{\bar{e}}$, respectively. Thus, the M in system (3.1.2) has the row index set $[m]$ and the column index set \mathcal{E} . For other notations used in this section, see the first paragraph in section 2.2.

By the ALTA on M , first of all, a set of successive pairs in $[m] \times \mathcal{E}$ is obtained for the triangular block B such that

$$(\sigma_l, \tau_l) = (i_1, j_{r+1}) \succ \cdots \succ (i_l, j_{r+l}), \quad r = n - l. \quad (3.2.1)$$

Then for each index pair $(s, t) \in \sigma_l \times \tau_l$, the $(s, t)^{th}$ entry of B can be identified by reading the $(i_s, j_{r+t})^{th}$ entry of H . An exemplary pseudo-code for the ALTA is described in **Algorithm 3.1**. In the algorithm, we design the ALTA as the iteration between the sub-routines **MPA()** and **Referencing()** that accompany **Sign-Flip()**. In the algorithm, whenever the triangulation of B by the **MPA()** stops prematurely, a column that is not joined into $\begin{bmatrix} B \\ D \end{bmatrix}$ nor $\begin{bmatrix} A \\ C \end{bmatrix}$ by that round, is chosen and declared to be a column of $\begin{bmatrix} A \\ C \end{bmatrix}$ by **Referencing()**. The triangulation of B proceeds in this fashion, untill all columns of M are joined into either $\begin{bmatrix} A \\ C \end{bmatrix}$ or $\begin{bmatrix} B \\ D \end{bmatrix}$. Many other strategies for **Referencing()** can be found in [3, 4].

With the returned (σ_l, τ_l) from the ALTA (see line 9 in the algorithm), let $\mathcal{E} = (\mathcal{R}, \bar{\mathcal{R}})$ and $[m] = (\mathcal{T}, \bar{\mathcal{T}})$ the disjoint pair of $[m]$ and \mathcal{E} , respectively, such that

$$\mathcal{R} = \mathcal{E} \setminus \tau_l = \{j_1, \dots, j_r\}, \quad \bar{\mathcal{R}} = \tau_l = \{j_{r+1}, \dots, j_{r+l}\}, \quad (3.2.2)$$

$$\mathcal{T} = \sigma_l = \{i_1, \dots, i_l\}, \quad \bar{\mathcal{T}} = [m] \setminus \sigma_l = \{i_{l+1}, \dots, i_m\}. \quad (3.2.3)$$

By extending the (σ_l, τ_l) into a row and column permutation pair (σ, τ) of M such

Algorithm 3.1: The ALTA on M

```

1 Input:  $H$  and  $(\bar{\mathcal{E}}, \mathcal{E})$  Output:  $(\sigma_l, \tau_l)$ 

   /*<-- Initialization: -->*/
2 foreach  $j \in \bar{\mathcal{E}}$  do
3   | Sign-Flip() with  $H^j$ ;

   /*<-- Triangulation: -->*/
4 while  $\mathcal{E} \neq \emptyset$  do
5   | if  $\mathfrak{R} = \emptyset$  then
6   |   | Referencing();
7   | else
8   |   | MPA();
9 return  $(\sigma_l, \tau_l)$ ; Exit the Algorithm;

   /*<-- Sub Routines: -->*/
10 MPA():
11 while  $\mathfrak{R} \neq \emptyset$  do
12   | foreach  $(i, j) \in \mathfrak{R}$ ; do
13   |   | if  $j \in \mathcal{E}$  then
14   |   |   | reduce  $\mathcal{E} := \mathcal{E} \setminus j$ ;
15   |   |   | Sign-Flip() with  $H^j$ ;
16   |   |   | update  $(\sigma_l, \tau_l) := (\sigma_l, \tau_l) \cup (i, j)$ ;
17   |   | reduce  $\mathfrak{R} := \mathfrak{R} \setminus (i, j)$ ;

18 Referencing():
19 choose an  $H_i$  such that  $|H_i| = \min\{|H_s| > 0\}$ ;
20 while  $|H_i| > 1$  do
21   | choose a  $j$  such that  $1_{ij} \in H_i$ ;
22   | Sign-Flip() with  $H^j$ ;

23 Sign-Flip():
24 foreach  $1_{ij} \in H^j$  do
25   | flip  $\text{sign}(1_{ij}) := -1$ ;
26   | reduce  $|H_i| := |H_i| - 1$ ;
27   | if  $|H_i| = 1$  then
28   |   | find the  $(s, t)$  such that  $1_{st} \in H_i$  and  $\text{sign}(1_{st}) = 1$ ;
29   |   | update  $\mathfrak{R} := \mathfrak{R} \cup (s, t)$ ;

```

that

$$\sigma : [m] \mapsto [m], \sigma(i_k) = k, \quad \text{and} \quad \tau : \mathcal{E} \mapsto \mathcal{E}, \tau(j_k) = k, \quad (3.2.4)$$

the permutation matrix P and Q^T of (σ, τ) can be formed by permuting rows of $I_{m \times m}$ and columns of $I_{n_e \times n_e}$ in the order of σ and τ , respectively. Let $\bar{M} = PMQ^T = \begin{bmatrix} A & B \\ C & D \end{bmatrix}$ as shown in **Figure 2.1-(a)**. Then for each $(s, t) \in [m] \times \mathcal{E}$, the $(s, t)^{th}$ element of \bar{M} is exactly the $(i_s, j_t)^{th}$ element of H in (3.1.1). Let $\bar{X}^T = Q^T X^T = \begin{bmatrix} X_{\mathcal{R}}^T \\ X_{\bar{\mathcal{R}}}^T \end{bmatrix}$ where

$$X_{\mathcal{R}} = [x_{j_1}, \dots, x_{j_r}] \quad \text{and} \quad X_{\bar{\mathcal{R}}} = [x_{j_{r+1}}, \dots, x_{j_{r+l}}]. \quad (3.2.5)$$

Then by $Q^T = Q^{-1}$, similar to system (2.1.2), system $PMX^T = P\beta^T$ is interpreted as $\bar{M}\bar{X}^T = P\beta^T$.

Let us now rearrange columns of MQ^T , as in the order of $(\mathcal{R}, \bar{\mathcal{R}})$, into two parts

$$M_{\mathcal{R}} = [H^{j_1}, \dots, H^{j_r}] \quad \text{and} \quad M_{\bar{\mathcal{R}}} = [H^{j_{r+1}}, \dots, H^{j_{r+l}}]. \quad (3.2.6)$$

Then by using $S^{-1} = \begin{bmatrix} B & 0 \\ D & I \end{bmatrix}$ which is in a lower triangular form, S can be factorized into a product form of elementary matrices such that

$$S = \prod_{k=l}^1 S^{(k)} = S^{(l)} S^{(l-1)} \dots S^{(2)} S^{(1)}, \quad (3.2.7)$$

where each $S^{(k)}$ is formed by replacing the column $(I_{m \times m})^k$ with the k^{th} column \bar{M}^k and l is the number of columns of the triangular block B . (See equation (1.2.23) at p. 19). With the product form (3.2.7), $S\bar{M}$ can be computed by the iteration

$$\bar{M} := S^{(k)} \bar{M}, \quad k = 1, 2, \dots, l. \quad (3.2.8)$$

Because the S-MLDA does not construct the permuted $\bar{M} = \begin{bmatrix} A & B \\ C & D \end{bmatrix}$ explicitly, the product form (3.2.7) should be interpreted appropriately via (P, Q) into an equivalent form. With the same way for (2.2.11), let $\bar{S} = P^T S P$ and let $\bar{S}^{(k)} = P^T S^{(k)} P$, $k = 1, 2, \dots, l$, formed by replacing the column $(I_{m \times m})^k$ with the column $H^{j_{r+k}}$.

Algorithm 3.2: The BSR on $M_{\mathcal{R}}$ by $\prod_{k=l}^1 \bar{S}^{(k)} \cdot M_{\mathcal{R}}$	
1	Input: $M_{\mathcal{R}}$ Output: $\bar{M}_{\mathcal{R}}$
2	foreach $(i_k, j_{r+k}) \in (\sigma_l, \tau_l)$ <i>as in the order</i> do
	$\% \leftarrow M_{\mathcal{R}} := \bar{S}^{(k)} M_{\mathcal{R}} \rightarrow \%$
3	foreach $1_{i, j_{r+k}} \in H^{j_{r+k}}, i \neq i_k$ do
4	add $(M_{\mathcal{R}})_i := (M_{\mathcal{R}})_i + (M)_{i_k};$
5	return $M_{\mathcal{R}}; // \leftarrow \bar{M}_{\mathcal{R}}$

Then (3.2.7) is now transformed to

$$\bar{S} = P^T S P = \prod_{k=l}^1 P^T S^{(k)} P = \prod_{k=l}^1 \bar{S}^{(k)} \quad (3.2.9)$$

Consequently, similar to the one in system (2.1.3), the BSR system $S\bar{M}\bar{X}^T = P\beta^T$ is now transformed to

$$\left[\left(\prod_{k=l}^1 \bar{S}^{(k)} \right) M_{\mathcal{R}}; \left(\prod_{k=l}^1 \bar{S}^{(k)} \right) M_{\bar{\mathcal{R}}} \right] X^T = \left(\prod_{k=l}^1 \bar{S}^{(k)} \right) \beta^T \quad \Leftrightarrow \quad (2.1.3). \quad (3.2.10)$$

Let $\bar{M}_{\mathcal{R}} = \bar{S} M_{\mathcal{R}}$ and $\bar{M}_{\bar{\mathcal{R}}} = \bar{S} M_{\bar{\mathcal{R}}}$. Notice that, since $\bar{M}_{\bar{\mathcal{R}}} = P^T \begin{bmatrix} I \\ 0 \end{bmatrix}$, the computation of $\bar{M}_{\bar{\mathcal{R}}}$ is enough to set up system (3.2.10) and is accomplished by the iteration

$$M_{\mathcal{R}} := \bar{S}^{(k)} M_{\mathcal{R}}, \quad k = 1, 2, \dots, l. \quad (3.2.11)$$

An exemplary pseudo-code for the BSR iteration (3.2.11) is described in **Algorithm 3.2**. For the GE on $\bar{M}_{\mathcal{R}}$ later, we assume that $M_{\mathcal{R}}$ is constructed explicitly in a ternary format $\{-1, 0, 1\}$.

Note that the S-MLDA does not construct the $S\bar{M} = \begin{bmatrix} \bar{A} & I \\ \bar{C} & 0 \end{bmatrix}$ of system (2.1.3). Therefore, the GE on \bar{C} should be designed to perform the pivoting processes on the set of rows $\{(\bar{M}_{\mathcal{R}})_i | i \in \bar{\mathcal{T}}\}$, which is equivalent to \bar{C} via P . At the end, the GE returns an updated $\bar{M}_{\mathcal{R}}$ that consists of $\{-1, 1, 0\}$ with a set of successive pairs

$$(\sigma_r, \tau_r) = (s_1, t_1) \succ \dots \succ (s_r, t_r) \subset \bar{\mathcal{T}} \times \mathcal{R}. \quad (3.2.12)$$

Algorithm 3.3: The GE on $\bar{M}_{\mathcal{R}}$ with \bar{T} and \mathcal{R}

```

1 Input:  $\bar{M}_{\mathcal{R}}$ ,  $\bar{T}$ , and  $\mathcal{R}$ .      Output:  $\bar{M}_{\mathcal{R}}$  and  $(\sigma_r, \tau_r)$ .
   /*<-- Initialization:  -->*/
2 foreach  $i \in \bar{T}$  do
3   if  $(\bar{M}_{\mathcal{R}})_i = 0$  then
4     discard  $\bar{T} := \bar{T} \setminus i$ ;

   /*<-- General Rounds:  -->*/
5 while  $\bar{T} \neq \emptyset$  do
   /*<-- Pivot Selection:  -->*/
6   choose  $\exists s_k \in \bar{T}$  such that  $|(\bar{M}_{\mathcal{R}})_{s_k}| = \min_{s \in \bar{T}} \{|\bar{M}_s|\}$ ;
7   select  $\exists l_{s_k, t_k} \in (\bar{M}_{\mathcal{R}})_{s_k}$ ;
8   insert  $(\sigma_r, \tau_r) := (\sigma_r, \tau_r) \cup (s_k, t_k)$ ;

   /*<-- Pivoting:  -->*/
9   foreach  $i \in \bar{T}$  such that  $(\bar{m}_{\mathcal{R}})_{i, t_k} = 1$  do
10    flip  $l_{i, t_k}$  into  $-l_{i, t_k}$ ;
11    add  $(\bar{M}_{\mathcal{R}})_i := (\bar{M}_{\mathcal{R}})_i + (\bar{M}_{\mathcal{R}})_{s_k}$ ;
12    if  $|(\bar{M}_{\mathcal{R}})_i| = 0$  then
13      discard  $\bar{T} := \bar{T} \setminus i$ ; //<-- To discard null rows

   /*<-- Discarding:  -->*/
14  discard  $\bar{T} := \bar{T} \setminus s_k$ ;
15 return  $(\sigma_r, \tau_r)$  and  $\bar{M}_{\mathcal{R}}$ ;

```

After the GE, an entry l_{ij} of L or u_{ij} of U can be identified from the $(s_i, t_j)^{th}$ entry of the $\bar{M}_{\mathcal{R}}$ where $s_i \in \sigma_r$ and $t_j \in \tau_r$. While computing $X_{\mathcal{R}}$ by the FS over rows of L then the BS over rows of U in system (2.1.6), each $1_{k,j}$ of $L^{(k)}$ or $U^{(k)}$ corresponds to the symbol addition $(\beta_l)_k := (\beta_l)_k + (\beta_l)_j$. Let $\bar{\beta}^T = \bar{S}\beta^T$. This addition should be interpreted as the symbol addition on $\bar{\beta}^T = \bar{S}\beta^T$ with $\bar{M}_{\mathcal{R}}$ by looking at (σ_r, τ_r) as in the following way. Each $1_{k,j}$ of $L^{(k)}$ or $U^{(k)}$ is recoded as the $-1_{s_k, t_j}$ or $1_{s_k, t_j}$ in $(\bar{M}_{\mathcal{R}})_{s_k}$, respectively. Therefore, the $-1_{s_k, t_j}$ or $1_{s_k, t_j}$ corresponds to the symbol addition $\bar{\beta}_{s_k} := \bar{\beta}_{s_k} + \bar{\beta}_{s_j}$ via s_k, s_j in σ_r , and t_j in τ_r . In this way, $L^{(k)}$ and $U^{(k)}$ can be interpreted as $\bar{L}^{(k)}$ and $\bar{U}^{(k)}$, which is the $m \times m$ elementary matrix formed by replacing those $-1_{s_k, t_j}$ and $1_{s_k, t_j}$ as the $1_{s_k, s_j}$ in the row $(I_{m \times m})_{s_k}$, respectively.

Therefore, the GE on $\bar{M}_{\mathcal{R}}$ is equivalent to the factorization

$$\bar{L}^{-1} = \prod_{k=r}^1 \bar{L}^{(k)}, \quad \bar{U}^{-1} = \prod_{k=1}^r \bar{U}^{(k)}, \quad (3.2.13)$$

where r is the number of columns of $\bar{M}_{\mathcal{R}}$. Consequently, multiplying the product form (3.2.13) into the BSR system (3.2.10) verifies the S-MLDA system (3.1.3). An exemplary pseudo-code for the GE on $\bar{M}_{\mathcal{R}}$ is described in **Algorithm 3.3**. For the recovery of $X_{\mathcal{R}}$, initializing by $x_{t_k} := \beta_{s_k}$ for each $(s_k, t_k) \in (\sigma_r, \tau_r)$ in advance replaces the symbol addition $\bar{\beta}_{s_k} := \bar{\beta}_{s_k} + \bar{\beta}_{s_j}$ into $x_{t_k} := x_{t_k} + x_{t_j}$ (see FS/BS in **Algorithm 3.4**).

Let us now go back to the GE to remove redundant symbol additions in system (3.1.3). It can be observed that, for each $i \in \bar{\mathcal{T}} \setminus \sigma_r$, the row $(M_{\mathcal{R}})_i$ is nullified by the GE on $\bar{M}_{\mathcal{R}}$, and thus, any symbol additions made with the syndrome symbol $\bar{\beta}_i$ is completely redundant. These redundant symbol additions can be removed by discarding the equations $M_i X^T = \beta_i$ in system (3.1.2) for all $i \in \bar{\mathcal{T}} \setminus \sigma_r$ by the step 1b) of the *pre-decoding*. In LDPC codes, since rows of H are sparse, these redundant additions may not be a serious drawback to the efficiency of the S-MLDA. In LT codes, however, a small fraction of dense rows in H is required to ensure that $|H^j| \geq 1$ for all $j \in [n]$, and most of them become redundant after the GE on $H_{\mathcal{R}}$ (see [19, p.4] and [27, Ex-50.5]). Therefore, ahead of the *post-decoding*, removal of the redundant equations is essential for the efficiency of the S-MLDA for LT codes. The computation of $\bar{U}^{-1} \bar{L}^{-1} \bar{S} \bar{\beta}^T$ is described in **Algorithm 3.4**. Note that, if $\mathcal{R} = \emptyset$, then the lost symbol vector X is simply recovered by the BSR iteration $\bar{\beta}^T := \bar{S} \bar{\beta}^T$ as in **Algorithm 3.4**.

In the alternative recovery by (2.1.7), it can be seen that, for each $i_k \in \sigma_l$, M_{i_k} and $(\bar{M}_{\mathcal{R}})_{i_k}$ corresponds to (A_k, B_k) and \bar{A}_k , respectively. Hence, by looking at $(i_k, j_k) \in (\sigma_l, \tau_l)$, each x_{j_k} of $X_{\bar{\mathcal{R}}}$, in the order of τ_l , is recovered by either $M_{i_k} X_{\bar{\mathcal{R}}}^T = \beta_{i_k} + (M_{\mathcal{R}})_{i_k} X_{\mathcal{R}}^T$ or by $x_{j_k} = \bar{\beta}_{i_k} + (\bar{M}_{\mathcal{R}})_{i_k} X_{\mathcal{R}}^T$. The $\bar{M}_{\mathcal{R}}$ (or \bar{A}) returned from

Algorithm 3.4: The computation of $\bar{U}^{-1}\bar{L}^{-1}\bar{S}^{(k)} \cdot \beta^T$

```

1 Input:  $\alpha_{\bar{e}}$  Output:  $\beta, \bar{\beta}, X_{\mathcal{R}}$ .

   /*<-- Initialization: -->*/
2 foreach  $i \in \sigma_l \cup \sigma_r$  do
3   set  $\beta_i := N_i \cdot \alpha_{\bar{e}}^T$ ; //<-recall  $\beta^T = N\alpha_{\bar{e}}$ 
4   copy  $\bar{\beta}_i := \beta_i$ ; //<-for the alternative recovery

   /*<--  $\bar{\beta}^T := \bar{S} \cdot \beta^T$ : -->*/
5 foreach  $(i_k, j_{r+k}) \in \sigma_l \times \tau_l$  as in the order do
6   /*<--  $\bar{\beta}^T := \bar{S}^{(k)} \cdot \bar{\beta}^T$ : -->*/
7   foreach  $1_{i, j_{r+k}} \in H^{j_{r+k}} \ i \neq i_k$  do
8     add  $\bar{\beta}_i := \bar{\beta}_i + \bar{\beta}_{i_k}$ ;

   /*<-- Initialization: -->*/
9 foreach  $(s_k, t_k) \in (\sigma_r, \tau_r)$  as in the order do
10  copy  $x_{t_k} := \bar{\beta}_{s_k}$ ;

   /*<-- FS by  $\bar{L}^{-1} = \prod_{k=r}^1 \bar{L}^{(k)}$  -->*/
11 foreach  $(s_k, t_k) \in (\sigma_r, \tau_r)$  as in the order do
12  /*<--  $\bar{\beta}^T := \bar{L}^{(k)} \bar{\beta}^T$  -->*/
13  foreach  $-1_{s_k, j} \in (\bar{M}_{\mathcal{R}})_{s_k}$  do
14    add  $x_{t_k} := x_{t_k} + x_j$ ;

   /*<-- BS by  $\bar{U}^{-1} = \prod_{k=1}^r \bar{U}^{(k)}$  -->*/
15 foreach  $(s_k, t_k) \in (\sigma_r, \tau_r)$  as in the reversed order do
16  /*<--  $\bar{\beta}^T := \bar{U}^{(k)} \bar{\beta}^T$  -->*/
17  foreach  $1_{s_k, j} \in (\bar{M}_{\mathcal{R}})_{s_k}, j \neq t_k$  do
18    add  $x_{t_k} := x_{t_k} + x_j$ ;
19 return  $\beta, \bar{\beta}$ , and  $X_{\mathcal{R}}$ ;

```

the BSR is not sparse in general. The top part of \bar{A} is more likely sparser than the top part of $[A; B]$. On the other hand, the bottom part of \bar{A} is much denser than the bottom part of $[A; B]$. Therefore, selecting a sparser equation by comparing the degrees $|M_{i_k}|$ and $|\bar{M}_{i_k}|$ may improve the efficiency of the S-MLDA in number of symbol additions significantly. An exemplary algorithm for the FFS is described in **Algorithm 3.5**. The overall S-MLDA is summarized in **Algorithm 3.6**.

Algorithm 3.5: The recovery of $X_{\mathcal{R}}$ by FFS

```

1 Input:  $X_{\mathcal{R}}$  Output:  $X_{\mathcal{R}}$ .
2 foreach  $(i_k, j_k) \in (\sigma_l, \tau_l)$  as in the order do
    /*<-- Use  $B_i X_{\mathcal{R}}^T = \beta_i + A_i X_{\mathcal{R}}^T$  -->*/
3   if  $|M_{i_k}| < |(\bar{M}_{\mathcal{R}})_{i_k}|$  then
4     copy  $x_{j_{r+k}} := \beta_{i_k}$ ; /*<-- not  $\bar{\beta}_{i_k}$ 
5     foreach  $1_{i_k, j} \in M_{i_k}, j \neq j_{r+k}$  do
6       add  $x_{j_{r+k}} := x_{j_{r+k}} + x_j$ ;
    /*<-- Use  $X_{\mathcal{R}}^T = \beta_i + \bar{A}_i X_{\mathcal{R}}^T$  -->*/
7   else
8     copy  $x_{j_{r+k}} := \bar{\beta}_{i_k}$ ; /*<-- not  $\beta_{i_k}$ 
9     foreach  $1_{i_k, j} \in (\bar{M}_{\mathcal{R}})_{i_k}$  do
10      add  $x_{j_{r+k}} := x_{j_{r+k}} + x_j$ ;
11 return  $X_{\mathcal{R}}$ ;

```

Algorithm 3.6: The overall S-MLDA

```

1 Input:  $[\alpha_{\bar{e}}, X]$  Output:  $[X_{\mathcal{R}}, X_{\mathcal{R}}]$ 
2 do ALTA by Algorithm 3.1;
3 if  $\mathcal{R} = \emptyset$  then
4   recover  $\bar{X}$  by Algorithm 3.4;
5   return  $X$ ;
6   exit the S-MLDA;
7 construct  $M_{\mathcal{R}}$  with  $\mathcal{R}$ ;
8 do the BSR by Algorithm 3.2;
9 do the GE by Algorithm 3.3;
10 if  $\mathcal{R} \setminus \tau_r \neq \emptyset$  then
11   return the free variables  $\mathcal{R} \setminus \tau_r$ ;
12   exit the S-MLDA;
13 recover  $X_{\mathcal{R}}$  by Algorithm 3.4;
14 recover  $X_{\mathcal{R}}$  by Algorithm 3.5;
15 return  $[X_{\mathcal{R}}, X_{\mathcal{R}}]$ ;
16 exit the S-MLDA;

```

3.3 The Complexity of the MLDA

Similar to LT codes in section 2.3, we estimate the computational complexity of the S-MLDA by counting the number of {sign, bit}-flips and symbol additions made by

the *pre-decoding* and the *post-decoding*, respectively. Throughout this section, we assume that $\mathcal{R} \neq \emptyset$.

Let us first estimate the complexity of the *pre-decoding*. By the ALTA based on **Algorithm 3.1**, total $|N|$ number of 1's is flipped into -1 to set up M , and then every 1 in M is eventually flipped into -1 . Hence, the complexity of the ALTA in sign-flip is proportional to $|H|$. While computing $\bar{M}_{\mathcal{R}} = \bar{S}M_{\mathcal{R}}$ by the BSR iteration (3.2.11), based on **Algorithm 3.2**, each 1 in $M_{\bar{\mathcal{R}}}$ (or $\begin{bmatrix} B \\ D \end{bmatrix}$), except in the diagonal of B , corresponds to one row addition in $M_{\mathcal{R}}$ (or $\begin{bmatrix} A \\ C \end{bmatrix}$). Therefore, the BSR constitutes the complexity proportional to $r(|M_{\bar{\mathcal{R}}}| - n + r)$, where r is the number of columns of $M_{\mathcal{R}}$. By the GE on $\bar{M}_{\mathcal{R}}$, based on **Algorithm 3.3**, when a pivot $1_{s_k, t_k}$ is chosen at each pivoting round k , the row \bar{M}_{s_k} is added to the rows of \bar{T} whose t_k^{th} column entry is 1. Since $|\bar{M}_{s_k}| \leq (r - k)$ and $|\bar{T}| \leq (m - l - k)$ at round k , the number of {sign, bit} -flips together is less than

$$\sum_{k=1}^r |\bar{M}_{s_k}| |\bar{T}| = c \sum_{k=1}^r k((1 - R - p)n + k), \quad (3.3.1)$$

where $R = \frac{m}{n}$ and c is a constant less than 1. In practice, simulations exhibit that, at each round k , $|\bar{T}| \leq \frac{m-l-k}{2}$ and $|\bar{M}_{s_k}| \leq \frac{r-k}{2}$, thus, $c \leq \frac{1}{4}$. Hence in total, the number {sign, bit}-flips by the *pre-decoding* step is less than

$$|H| + r|M_{\bar{\mathcal{R}}}| + c \sum_{k=1}^r k((1 - R - p)n + k). \quad (3.3.2)$$

When $r \approx \epsilon n$ with a small fraction $\epsilon > 0$, we may assume that, in general, the estimate (3.3.2) is dominated by either $(1 - R - p)\epsilon^2 n^3$ or $\epsilon^3 n^3$, so that as shown in [3, p. 4], the overall complexity of the *pre-decoding* step is $O(n^3)$ but with a very small constant factor ϵ^3 or $(1 - R - p)\epsilon^2$. For an example, as presented in **Figure 3.3**, the fraction of references $\frac{r}{n}$ from simulations of the S-MLDA is less than 0.032.

Second, let us now estimate the number of symbol additions made by the *post-decoding* step. We notice that in **Algorithm 3.4**, precisely, n_e number of β_i 's,

whose i is in $\sigma_l \cup \sigma_r$, are constructed to set up β and $\bar{\beta}$ at the initialization step of the algorithm. Let $p = \frac{nc}{n}$. Then approximately, $\frac{p}{1-R}(|N| + n)$ number of symbol additions of $\alpha_{\bar{e}}$ is made for β and $\bar{\beta}$ at the initialization step. For the computation of $\bar{S}\beta^T$, approximately, $\frac{p}{1-R}|M_{\bar{\mathcal{R}}}|$ symbol additions of $\bar{\beta}$ are made. Then for the recovery of $X_{\bar{\mathcal{R}}}$, total of $(|L| + |U| - 2r)$ symbol additions of $\bar{\beta}$ is made by $\bar{U}^{-1}\bar{L}^{-1}$. Now for each $i \in \sigma_l$, let $d_i = \min\{|\bar{M}_{\mathcal{R}}|_i, |M_i|\}$. By the alternative recovery step (based on **Algorithm 3.5**), total $d = \sum_{k=1}^l d_{i_k}$ symbol additions is made for the recovery of $X_{\bar{\mathcal{R}}}$, and $d < \frac{p}{1-R}|M|$. Hence in total, the number of symbol additions made by the *post-decoding* is less than

$$\frac{p}{1-R} (|H| + |M_{\bar{\mathcal{R}}}| + n) + r^2. \quad (3.3.3)$$

Lastly, let us estimate the number of symbol additions by the original MLDA in [3]. Let the system $X_{\bar{\mathcal{R}}} = \bar{\beta}^T + \bar{A}X_{\mathcal{R}}^T$ alone be used for the recovery of $X_{\bar{\mathcal{R}}}$, as showed in 4 *FFS* in section 2.1. At the *pre-decoding* step, note that only the {sign, bit}-flips that correspond to a row addition constitute one symbol addition of β or $\bar{\beta}$. Foremost, total of $|N| + |M_{\bar{\mathcal{R}}}|$ symbol additions by the BSR and less than $|U| - r + \sum_{k=0}^{r-1} |\bar{T}|$ row additions by the GE and U^{-1} are made for the recovery of $X_{\bar{\mathcal{R}}}$. Then for the recovery of $X_{\bar{\mathcal{R}}}$, total of $|\bar{M}_{\mathcal{R}}| + l$ (or $|\bar{A}| + l$) symbol additions is made. Hence in total, the number of symbol additions by the original MLDA in [3] is less than

$$|N| + |M_{\bar{\mathcal{R}}}| + |\bar{M}_{\mathcal{R}}| + r^2 + (1 - R - p)nr. \quad (3.3.4)$$

In the following section, we substantiate that, by experimental results, the estimate (3.3.4) is dominated by $|\bar{M}_{\mathcal{R}}|$ (or $|\bar{A}|$) as $n_e \rightarrow m$. (See green curves in **Figure 3.2** that should be removed by the alternative recovery step).

3.4 Simulations

In this section, we present the simulation results of the S-MLDA tested with $\frac{1}{2}$ -rate PEG LDPC codes [13]. We then demonstrate, based on the experimental results, that some LDPC codes under the S-MLDA can achieve performance in stable erasure rates p (for the successful S-MLDA) very close to the ideal limit $\frac{m}{n}$, while the decoding complexity of the *post-decoding* in symbol additions maintains within few tens of n .

The simulation is based on the following spectrum. First of all, for each block length n , a check matrix H is arranged in advance by using PEG software [27] that provides a larger local minimum cycle of columns in the best effort of the greedy algorithm in [13]. The row and column-degree distribution (λ, ρ) of H , whose average row-degree is $a_r = 8.33$, is as follows

$$\begin{aligned}\lambda(x) &= 0.4578x^2 + 0.3238x^3 + 0.0214x^4 + 0.0593x^6 + 0.0389x^7 \\ &\quad + 0.0248x^8 + 0.0088x^9 + 0.0177x^{19} + 0.0475x^{20}, \\ \rho(x) &= 0.6708x^8 + 0.3292x^9.\end{aligned}\tag{3.4.1}$$

Using the ALTA in [4], if necessary, we convert H into an approximate triangular generator matrix $G = [S_{m \times k}; L_{m \times m}]$ to obtain a codeword α in a systematic form $\alpha = (\alpha_I, \alpha_P)$, where $\alpha_P = L_{m \times m}^{-1} S_{m \times k} \alpha_I^T$. Second, the tested block-lengths n are from $n = 2,000$ to $20,000$ by $2,000$. For each n , starting from $n_e = (0.5)n$ to $n_e = (0.4)n$, the number of losses n_e is given by the decrement $n_e := n_e - 1$. Then for each pair (n, n_e) , the S-MLDA is tested 100 times by assigning n_e random losses on a codeword α by using the Mersenne Twister algorithm [16] on $[n]$ to measure the performances of codes under the S-MLDA in the following scenarios:

1. decoding failure rate of codes under the S-MLDA and the MPA;
2. number of symbol additions by the *post-decoding* and the original MLDA [3];
3. fractions of references $\frac{r}{n}$;

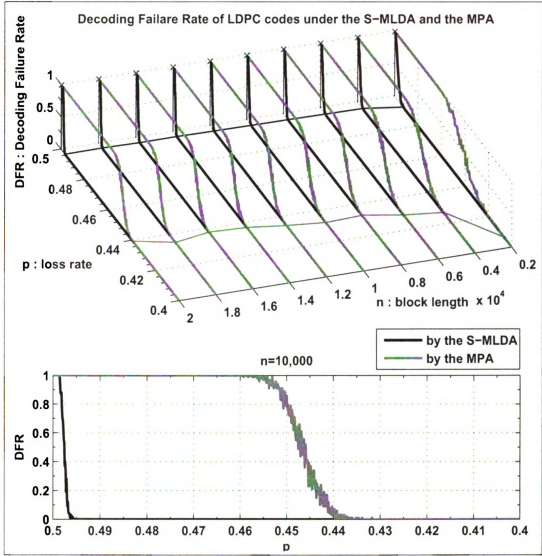


Figure 3.1. Decoding Failure Rate of LDPC codes under the S-MLDA (black curves) and the MPA (gray curves) for block lengths $2,000 \leq n \leq 20,000$

4. rank-deficiency $\eta = \dim(\text{Ker}(H))$.

In **Figure 3.1**, for each fixed n , a curve represents the decoding failure rate of LDPC codes (denoted as DFR in the figure) in 100 trials of decoding by the S-MLDA (black curves) and by the MPA (gray curves). For an example, when $n = 10,000$ (see the bottom figure in **Figure 3.1**), the red point ($p = 0.49$, $\text{DFR} = 1$) indicates that the MPA never succeeds to recover $pn = 4900$ random losses of α . In contrast,

the blue point $(0.49, 0)$ indicates that the S-MLDA always succeeds for the recovery of $pn = 4900$ random losses. It can be seen that, when $n = 10,000$, the DFR of the S-MLDA is 0 up to $p \leq 0.496$. In contrast, the DFR of the MPA is always greater than 0 for all $p \geq 0.435$ and is 1 for all $p > 0.458$. It can be also observed that, from the top figure, for any pair (n, p) where $p \leq 0.492$, the DFR by the S-MLDA decreases to 0 dramatically by a slight decrement in p . Thus, the maximum loss rate $p = \frac{n\epsilon}{n}$ recoverable by the S-MLDA can be projected to the ones close to the ideal limit $1 - R = 0.5$.

Figure 3.2 shows the number of symbol additions made by the *post-decoding* (based on **Algorithm 3.4** and **3.5**) and the original MLDA in [3]. For each fixed n , curves represent the number of symbol additions divided by n , denoted as NS in the figure. First, a black curve 1 and a gray curve 2 indicates the NS made by the *post decoding* and the original MLDA in [3], respectively. For instance, when $n = 10,000$ (see the bottom figure in **Figure 3.2**), the point $(p = 0.49, n_s = 9)$ on the black curve 1 indicates that, approximately, $(NR)n = 90,000$ symbol additions is made by the *post-decoding*, and the red point $(0.49, 36)$ corresponds to, approximately, 360,000 symbol additions by the original MLDA. Similarly, a gray curve 3 indicates the difference

$$d^* = \frac{1}{n} \left| |\bar{M}_{\mathcal{R}}| - \sum_{k=1}^l d_{i_k} \right|, \quad (3.4.2)$$

where $d_{i_k} = \min\{|M_{i_k}|, |(\bar{M}_{\mathcal{R}})_{i_k}|\}$, $i_k \in \sigma_l$. (Recall the alternative recovery step 2c) in section 2.1). It can be observed that, as $p \rightarrow 0.5$, a gray curve 2 is more likely parallel to a gray curve 3. This tells that, as p approaches to $1 - R = 0.5$, the number of symbol additions by the original MLDA is dominated by $|\bar{M}_{\mathcal{R}}|$ (or $|\bar{A}|$). In the top figure, notice that the NS by the *post-decoding* is less than 17 for any instance of (n, p) . In contrast, as n grows and $p \rightarrow 0.5$, the red NS by the original MLDA far exceeds the one made by the S-MLDA. This substantiates that, as $p \rightarrow 0.5$, the alternative recovery step of the *post-decoding* significantly improves the decoding

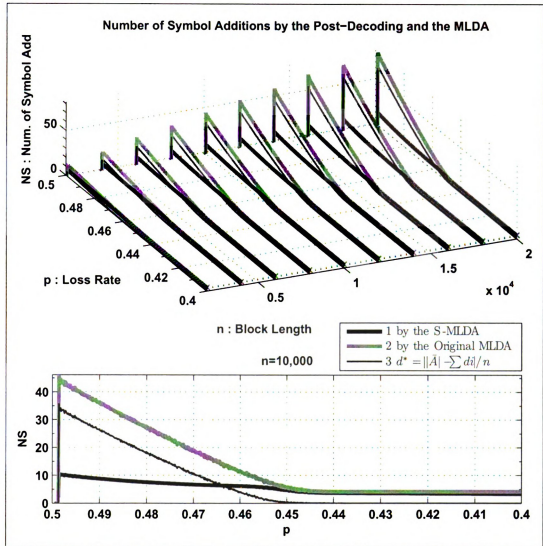


Figure 3.2. Number of Symbol Additions made by the Post-Decoding and the original MLDA

efficiency in symbol additions. Lastly, we recall the removal of redundant equations by the step 1b) of the *pre-decoding* step. In the figure, for any pair (n, p) , the NR made by the redundant rows is too small to tell. The reason to this is in the fact that redundant rows are sparse with average degree $a_r = 8.33$ and the number of those rows is $(1 - R - p)n$. Nonetheless, the number of redundant symbol additions over the those rows is about $8.33(1 - R - p)n$ and is not trivial.

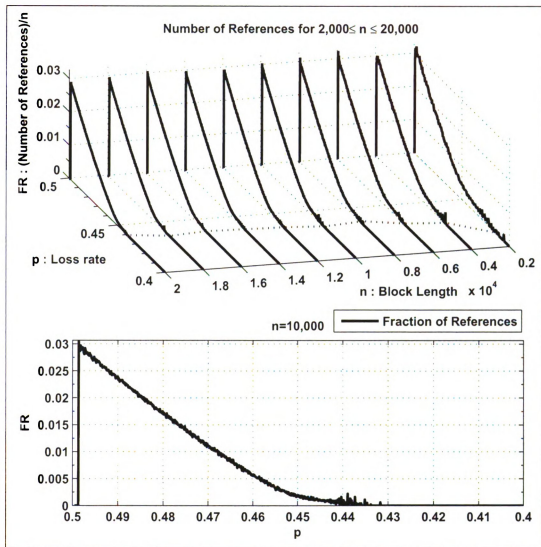


Figure 3.3. Number of References

In **Figure 3.3**, for each fixed n , a black curve represents the fraction of references $FR = \frac{r}{n}$, where r is the number of columns of $M_{\mathcal{R}}$. For an example, when $n = 10,000$ (the bottom figure in **Figure 3.3**), the point $(p = 0.49, r_f = 0.023)$ indicates that $r \approx n(FR) = 230$. It can be observed that, from the top figure, $FR \leq 0.032$ for any instance of (n, p) . This substantiates the very small constant fraction in the complexity of the GE on \bar{C} . (See the last paragraph in section 3.3 or [3, p. 4]). Therefore, with such small fraction of references, the GE on $\bar{M}_{\mathcal{R}}$ (or \bar{C}) may not be

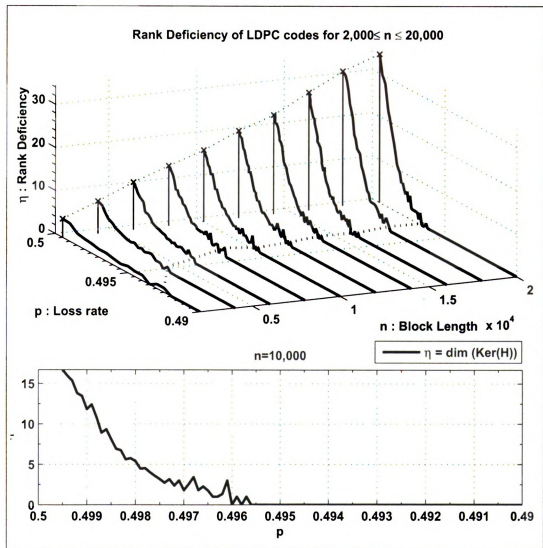


Figure 3.4. Number of Free Variables by the S-MLDA

a drawback to the overall complexity of the S-MLDA.

In Figure 3.4, for each fixed n , a black curve $\eta = f(p)$ represents the number of free variables when the S-MLDA fails to recover the lost $X = \alpha_e$. For an example, when $n = 10,000$, the point $(p = 0.498, \eta = 5)$ (see the bottom figure in Figure 3.4) says that the S-MLDA fails to recover the lost α_e with probability DFR = 0.81 (see Figure 3.1) and the rank-deficiency in this case is approximately 5. Thus, even the failure cases of the S-MLDA, α is obtainable by retransmitting less than 5 symbols

of free variables only.

3.5 Conclusions

Through the sections (3.1) and (3.2), we provide mathematical models and exemplary algorithms of the routines of the S-MLDA for the decoding of BEC based LDPC codes. In section 3.3, we estimate the complexity of the proposed S-MLDA. In section 3.4, we present the simulation results of the S-MLDA tested with $\frac{1}{2}$ -rate PEG-LPDC codes for the performances in decoding failure rates and computational complexities. Particularly in the section, we substantiate that the PEG-LDPC codes under the S-MLDA can achieve performance in erasure recovery rate very close to the ideal limit $1 - R$, while the S-MLDA maintains the complexity of the post-decoding in symbol additions within few tens of n and the fraction of references $\frac{r}{n}$ less than 0.032.

INDEX

*LIST OF ABBREVIATIONS

ALTA, 7
 BEC, 2
 BS, 18
 BSR, 41
 DFR, 64
 FFS, 41
 FR, 67
 FS, 18
 GE, 3
 ISD, 31
 LDPC, 4
 LT, 5
 MLDA, 7
 MPA, 6
 NS, 66
 RSD, 31
 S-MLDA, 8

*LIST OF NOTATIONS

$(\mathcal{R}, \bar{\mathcal{R}})$, 46
 $(\mathcal{T}, \bar{\mathcal{T}})$, 46
 (σ_l, τ_l) , 46
 1_{ij} , 46
 $C(H)$, 20
 $H(t)$, 52
 $X\mathcal{R}$, 46
 $X_{\bar{\mathcal{R}}}$, 46
 $[m]$, 46
 $[n]$, 46
 \bar{X} , 46
 $\text{Ker}(H)$, 3
 $\text{RS}(H)$, 58
 a_r , 56
 $\text{Img}(H)$, 12

ALTA, 7, 40, 80, 83
 altered $\rho(x)$, 57
 alternative recovery, 44, 81, 88, 95
 BEC, 2
 BS, 18
 Row-wise, 18
 BSR, 41, 80
 capacity approaching, 29
 code
 LDPC, 20
 LT, 32
 raptor, 38, 54
 tornado, 30
 codeword, 2, 20
 decoder, 2
 degree
 column, 27
 row, 27
 Degree Reduction, 23
 Density Evolution, 26
 DFR, 64, 73, 94
 elementary row operation, 13
 encoder, 2
 entry-degree ensemble, 27
 erasure, 2
 FFS, 41
 FR, 67, 77, 97
 FS, 18
 Row-wise, 18
 GE, 3, 11, 41, 80

- ISD, 31, 34
- LDPC, 4
 - PEG, 7
- LT, 5
 - transmission, 31
- LU-factorization, 11, 15
- matrix
 - elementary, 14
 - LDPC, 21
 - Lower Triangular, 12
- Mersenne Twister algorithm, 64, 93
- MLDA, 7
 - Burshtein and Miller's, 40
 - S-MLDA, 8
 - Separated, 43
- MPA, 6, 54
- NS, 66, 74, 75, 95
- optimal, 7
- overhead, 39, 41
- post-decoding, 43, 81
- pre-decoding, 43, 81
- rank deficiency, 57, 78
- rank distribution, 58
 - Cooper, 59
 - Kovalenko, 59
- rate
 - code, 22
 - erasure, 7
 - redundancy, 22
- redundant, 22
- redundant rows, 44, 75, 88, 96
- Ripple, 33
- RSD, 31, 37
 - definition of, 54
- S-MLDA, 80, 83
- symbol, 2
 - information symbol vector, 2
 - syndrome, 11
 - syndrome vector, 11
 - systematic, 22

BIBLIOGRAPHY

- [1] R. G. GALLAGER, *Low Density Parity-Check Codes*, MIT Press, Cambridge, MA, 1963. 5, (1969), 399-410.
- [2] M. LUBY, *LT Codes*, 43rd Annual IEEE Symposium on Foundations of Computer Science, 2002.
- [3] David Burshtein, Gadi Miller, *An Efficient Maximum-Likelihood Decoding of LDPC Codes Over the Binary Erasure Channel*, IEEE Trans. Inform. Theory, 50:2837-2844, 2004.
- [4] T. RICHARDSON, R. URBANKE, *Efficient Encoding of Low-Density Parity-Check Codes*, IEEE Trans. Inform. Theory, 47:638-656, 2001.
- [5] M. LUBY, M. MITZENMACHER, A. SHOKROLLAHI, D. SPIELMAN., *Efficient Erasure Correcting Codes.*, IEEE Transl Inform. Theory, 47:569-584, 2001.
- [6] M. LUBY, M. MITZENMACHER, A. SHOKROLLAHI, D. SPIELMAN., *Practical Loss-resilient Codes*, In Proceedings of the 29th annual ACM Symposium on Theory of Computing, pages 150-159, 1997.
- [7] A. SHOKROLLAHI, *LDPC Codes: An Introduction*, J. Diff. Eqs., 5, (1999), 399-410.
- [8] D. MACKAY AND R. NEAL, *Good codes based on very sparse matrices*, Cryptography and Coding, 5th IMA Conference, Dec. 1995.
- [9] L. BAZZI, T. RICHARDSON, R. URBANKE, *Exact thresholds and optimal codes for the binary symmetric channel and Gallager's decoding algorithm A*, IEEE Trans. Inform. Theory, 47, 2001.
- [10] L. RIZZO, *Effective Erasure Codes for Reliable Computer Communication Protocols*, Comput. Commun. Rev., vol 27, pp.24-36, Apr. 1997.
- [11] L. RIZZO, L. VICISANO, *A Reliable Multicast Data Distribution Protocol Based on Software FEC Techniques*, in Proc. HPCS, June 1997
- [12] P. OSWALD AND A. SHOKROLLAHI, *Capacity-Achieving Sequences for the Erasure Channel*, IEEE Trans. Inform. Theory, 48:3017-3028, 2002.

- [13] H. Xiao, A. H. Banihashemi, *Improved Progressive-Edge-Growth (PEG) Construction of Irregular LDPC Codes*, IEEE Comm. Letters, Vol 8, No.12, December, 2004.
- [14] A. SHOKROLLAHI, *Raptor codes*, Digital Fountain, Inc., Tech. Rep. DF2003-06-001, June 2003.
- [15] A. Shokrollahi, S. Lassen, and R. Karp, *Systems and Processes for Decoding Chain Reaction Codes Through Inactivation* U.S. Patent 1,856,263, Feb. 15, 2005.
- [16] MAKATO MATSUMOTO AND TAKUJI NISHIMURA *Mersenne Twister: A 623-Dimensionally Equidistributed Uniform Pseudo-Random Number Generator*, ACM Trans. on Modeling and Computer Simulation, Vol. 8, No.1, Jan 1998, p3-30.
- [17] Hossein Pishro-Nik and Faramarz Fekri, *On Decoding of Low-Density Parity-Check Codes Over the Binary Erasure Channel* IEEE Trans. Inform. Theory, 50:439-454, Mar. 2004.
- [18] KI-MOON LEE, HAYDER RADHA, *The Design of Maximum-Likelihood Decoding Algorithms of LDPC Codes over Binary Erasure Channels*, Accepted to CISS 2007.
- [19] KI-MOON LEE AND HAYDER RADHA, *The Maximum Likelihood Decoding of LT codes and Degree Distribution Design with Dense Fractions*, Accepted to ISIT 2007.
- [20] KI-MOON LEE, HAYDER RADHA, AND HO-YOUNG CHEONG, "LT Codes from an Arranged Encoder Matrix and Degree Distribution Design with Dense Rows", submitted to Allerton Conference 2007.
- [21] I. N. KOVALENKO, A. A. LEVITSKAYA AND M. N. SAVCHUK, *Selected Problems in Probabilistic Combinatorics (in Russian)*, Naukova Dumka, Kyiv 1986.
- [22] C. COOPER, *On the rank of random matrices*, Random Structures and Algorithms, 1999.
- [23] Steven J. Leon, *Linear Algebra with Application*, Prentice Hall 6th Edition.
- [24] Steven Roman, *Advanced Linear Algebra*, Springer 2nd Edition 2005.
- [25] Charles G. Cullen, *Matrices and Linear Transformations*, Addison-Wesely 2nd Edition 1990.
- [26] Sheldon Axler, *Linear Algebra Done Right*, Springer 2nd Edition 1999.
- [27] DAVID J.C MACKAY, *Information Theory, Inference and Learning Algorithms*, Cambridge University Press 2003.

- [28] T. RICHARDSON, R. URBANKE, *Modern Coding Theory*, This books is available at <http://lthcwww.epfl.ch/>
- [29] V. F. Kolchin. *Random Graphs*, Cambridge University Press 1999.

MICHIGAN STATE UNIVERSITY LIBRARIES



3 1293 02956 0376