Doctoral Dissertation

Construction and Decoding of Complex Low Density Lattice Codes

Warangrat Wiriya

Supervisor : Professor Brian Michael Kurkoski

Graduate School of Advanced Science and Technology
Japan Advanced Institute of Science and Technology
[Information science]
March 2023

Reviewed by

Professor Tadashi Wadayama

Professor Kaneko Mineo

Associate Professor Lim Yuto

Associate Professor Sakti Sakriani

Abstract

Coded modulation increases the spectral efficiency of wireless communication systems. Lattice codes are elegant and powerful structures for coded modulation that not only can achieve the capacity of the additive white Gaussian noise (AWGN) channel, but also are a key ingredient to many multi-terminal schemes that exploit linearity properties [1–3]. There is an always-increasing demand for increased spectrum efficiency, massive connectivity and higher data rates; in post-5G or 6G wireless networks, lattice codes are a potential candidate to achieve these goals [4–7].

Low-density lattice codes (LDLC) defined over the real numbers is one type of lattice codes which show the decoded efficiently in high-dimensional Euclidean space, and error free decoding is possible within 0.6 dB of the unconstrained power channel capacity [8]. These real-valued LDLC were extended to the complex numbers. Such complex low-density lattice codes (CLDLC) provide several advantages for future wireless network and also outperforms real LDLC. However, belief propagation (BP) decoding of CLDLC confronts the same issue as real LDLC, that an infinite Gaussian mixture must be approximated for the decoder implementation.

This dissertation provides three contributions for complex low-density lattice codes (CLDLC). First, a decoding algorithm for CLDLC using a likelihood-based reliability function is used to determine the number of complex Gaussian functions at the variable node. This allows each message to be approximated by a variable number of Gaussians depending upon its reliability. An upper bound on the Kullback-Leibler (KL) divergence of the approximation is formed to find selection thresholds via linear regression. Second, a construction of complex low-density lattice codes (CLDLC) using Eisenstein integers is given. Third, a generalized CLDLC Latin square construction and a corresponding condition for convergence of variances under belief propagation (BP) decoding is given. The proposed CLDLC decoding algorithm has higher performance and lower complexity compared to existing algorithms. When the reliability-based algorithm is applied to Eisenstein integer CLDLC decoding, the complexity is reduced to $\mathcal{O}(n \cdot t \cdot 1.35^{d-1})$ at volume-to-noise ratio of 6 dB, for lattice dimension n, with degree d inverse generator matrix and t decoding iterations. Decoding CLDLC using Eisenstein integers has lower complexity than CLDLC using Gaussian integers when $n \geqslant 49$.

In addition, this dissertation has a contribution on low-density parity-check code (LDPC) decoders for NAND flash memory. The data read system for NAND flash memory can be modeled by a discrete memoryless channel (DMC) with unknown channel transition probabilities. However, LDPC decoders need a channel estimate, and incorrect channel estimation degrades the performance of LDPC decoder. This abstract proposes using the expectation maximization (EM) algorithm to estimate channel transition probabilities, needed to compute log-likelihood ratios (LLRs) for the LDPC decoders. At word-error rate 10^-5, the performance of the EM system was only 0.02 dB loss compared to the system that knows the channel exactly.

Keywords: Complex low-density lattice codes, Belief propagation decoder, Eisenstein integers, Low-density parity-check codes, NAND flash memory.

Acknowledgments

First of all, I would like to express my deepest appreciation to my advisor Professor Brian M. Kurkoski for supporting my Ph.D study and research, for the knowledge, motivation, inspiration, and his patience during the long journey of my Ph.D program.

Besides my advisor, I would like to thank my dissertation committee: Professor Tadashi Wadayama, Professor Kaneko Mineo, Associate Professor Lim Yuto, and Associate Professor Sakti Sakriani for all constructive comments on my dissertation.

Next, I would like to thank the doctoral research fellow (DRF) from JAIST, KDDI foundation scholarship, and JSPS scientific research Kakenhi Kiban Kenkyu for research grants and scholarships.

Finally, I very much appreciate my family in Thailand, my father, my mother, and my sister for being my strength. Thank you to my friends in JAIST for helping and supporting me: Tseng Bo-yu, Ashmawy Arm, and Omran Hindawi.

Contents

List of Figures

Chapter 1

Introduction

1.1 Background and Motivation

Starting from the 1890s, a famous wireless telegraphy experiment was conducted by Nobel Laureate G. Marconi [10]. This was the first time that the signals were transmitted wirelessly using electromagnetic waves. It took around 90 years before this technology was turned into commercial applications in which people can connect with each other in real-time. 1G-6G technologies are as follows.

1) The first-generation of mobile communications (1G) was introduced for voice communications in the 1980s, which transmits information via analog signals, the data throughput is around 2.4 Kbps [11].

2) In the 1990s, the second-generation (2G) of mobile communications was introduced. In this technology, division multiple access (TDMA) and code division multiple access (CDMA) are digital modulation techniques which can carry both voice and short message services, the data throughput is around 64 Kbps [12].

3) In 2000, the 3G network was launched for communications and internet access at a speed of at least 2 Mbps.

4) In 2010, the fourth generation (4G) network was launched and it can support high-speed, high-quality, and high-capacity services with a speed of at less than 1 Gbps [12].

5) Currently, we are in the era of fifth-generation (5G) of mobile communication networks. This 5G was launched in 2019 and the international telecommunication union (ITU) has defined three major application scenarios for 5G new radio (NR): enhanced mobile broadband (eMBB), massive machine type communication (mMTC), and ultra-reliable low latency communication (URLLC). 5G is expected to obtain a data rate of 1-10 Gbps and a latency of one millisecond [13].

6) Moving towards 2030 and beyond, 6G will be deployed to provide a truly global coverage with an extreme high data (100 Gbps at the peak data rate), an extreme low latency, an extreme high reliability and an enormous number of connectivity [14].

There is an always-increasing demand for massive connectivity and higher data rates. One solution to achieve this goal is increase spectrum efficiency using coded modulation. Lattice codes are elegant and powerful structures for coded modulation compared to the current quadrature amplitude modulation (QAM). Lattice codes is not only can achieve the capacity of the additive white Gaussian noise (AWGN) channel, but also are a key ingredient to many multi-terminal schemes (can support massive connectivity) that exploit linearity properties [1–3]. Therefore, lattice codes are a potential candidate to achieve these goals [4–7]. For example, lattice codes are used in compute and forward (CF) which is a well-known relay paradigm in wireless communications. In post-5G or 6G wireless network, CF may play a role in multi-user multi-relay [15] or massive multi-input multi-output (MIMO) system in low earth orbit (LEO) satellite communication system [16].

The CF strategy, which is based on lattice codes, was proposed by Nazer and Gastpar [17]. This strategy implements network coding where the receivers decode finite-field linear combinations of transmitted messages, instead of decoding the transmitted messages in the traditional decode-and-forward paradigm. A receiver that receives a sufficient number of linear combinations can recover the transmitted messages by solving a system of independent linear equations that relate the transmitted messages. [18] extend the nested lattice codes over integers for CF paradigm of Nazer and Gastpar to the use of nested lattice codes over Eisenstein integers. The results show that both the outage performance and error-correcting performance of nested lattice codebooks over Eisenstein integer outperforms lattice codebooks over integers considered by Nazer and Gastpar with no additional computational complexity.

Low-density lattice codes (LDLC) is one type lattice codes which can be decoded efficiently in a large dimension. LDLC defined over the real numbers were proposed by Sommer et al. [19]. These lattice codes can be encoded and decoded efficiently in high-dimensional Euclidean space, and error-free decoding is possible within 0.6 dB of the unconstrained power channel capacity [8].

These real-valued LDLC were extended to the complex numbers by Yona and Feder [20]. Such complex low-density lattice codes (CLDLC) provide several advantages. For instance, CLDLC are naturally matched to complex-valued channels, and they are a suitable construction for the compute-and-forward paradigm as an alternative strategy for wireless networks [21]. Another advantage is that CLDLC increases the coding gain compared to real LDLC [20]. An n-dimensional CLDLC is not simply a $2n$-dimensional real LDLC, but rather the CLDLC is directly generated over the domain of complex numbers. CLDLC outperforms $2n$-dimensional real LDLC because the real-valued parity check matrix normally suffers from short loops.

1.2 Problems to be Solved

LDLC and CLDLC lattices have a sparse inverse generator matrix. Based on this property, LDLC and CLDLC can be decoded on principles similar to low-density parity-check (LDPC) codes using a belief propagation (BP) decoding algorithm [22], where the messages in the iterative processing of LDLC and CLDLC are real number and complex number Gaussian functions, respectively, and the algorithm has complexity which is linear in lattice dimension n. BP decoding of LDLC and CLDLC confront the same issue that are an infinite Gaussian mixture must be approximated for the decoder implementation.

1.3 Related Works

In this section, we summarize existing works that proposed to approximate the infinite Gaussian mixture in BP decoding.

1.3.1 Related Works to LDLC

For the case of real-valued LDLC, in 2008, [23] introduced a parametric LDLC decoding algorithm employing GMR algorithm to approximate the Gaussian mixture messages. All possible pairs of Gaussians on a list are searched and the closest pairs are replaced with a single Gaussian. Further, using single Gaussians as the messages between the variable and check nodes leads to reduced memory requirements with minor performance penalty [8]. Yona and Feder [24] presented a parametric LDLC decoder based on real numbers where the Gaussian mixture approximation algorithm is similar to [20] in the complex lattice case. In 2016, [25] proposed the three/two Gaussian decoding algorithm to reduce the number Gaussian mixtures in LDLC. The three/two Gaussian decoding algorithm eliminated searching and sorting, which were used by previous works [8, 23, 24], conversely, the infinite Gaussian mixtures are approximated by three or two Gaussians. Each variable message is computed based on 3^{d-1} and 2^{d-1} Gaussian functions for three and two Gaussians decoding algorithm, respectively, where d is the degree of the inverse generator matrix. [26] reduced the complexity of the three/two Gaussians decoding algorithm by calculating each variable message based on $2d - 2$ Gaussian functions. Recently, an innovative real-valued LDLC decoder was given by Wang and Mao, which uses list sphere decoding at the variable node, but with the corresponding complexity of sphere decoding [27].

1.3.2 Related Works to CLDLC

For lattice codes based on Gaussian integers, in the complex-valued LDLC case, Yona and Feder [20] presented a CLDLC decoder employing the Gaussian mixture reduction (GMR) algorithm. Their decoder sorts the whole list of Gaussian mixtures and the Gaussians which satisfy a given condition will be grouped. Each incoming message at the variable node is approximated by 9 complex Gaussian functions. The GMR algorithm must be performed at every multiplication at the variable node, on each iteration. Due to the large number of approximated Gaussian functions, the complicated GMR algorithm, and the large number of uses of the GMR algorithm at the variable node, this proposed algorithm may not be suitable for hardware implementation. That paper also extended a condition on convergence of the variances during BP decoding of Latin square LDLCs from the real case [19] to the complex case, but did not give a proof.

1.4 Goal and Contributions

The goal of this dissertation is to develop low-complexity construction and decoding of CLDLC. To achieve this goal, we first started our development on the real-valued LDLC, then extended it to CLDLC.

For LDLC, we propose reliability-based decoding of low-density lattice codes (LDLC). We define the reliability of the check-to-variable messages for two purposes. The first one is to choose to approximate the infinite Gaussian mixtures by one or two Gaussian. The reliability of each check-to-variable message is calculated. If there is higher reliability than a decided threshold value, one Gaussian will be selected; otherwise two Gaussians will be used. The other purpose is for the updating sequence of variable nodes of the parametric shuffled BP (SBP) decoding algorithm [28]. The parametric SBP increases the convergence speed. The updating sequence of SBP follows the order of reliability of the check-to-variable messages from high to low. The numerical results show that the proposed algorithm gives superior performance and lower complexity compared to two or three Gaussian decoding algorithm [25].

The major advantage of this work is it reduces the number of the Gaussian mixtures at the variable node and increases the convergence speed compared to the two or three Gaussians decoding algorithm [25]. Another advantage is the proposed algorithm shows that: 1) the single Gaussian can be used, 2) each variable node input does not need to be expanded using a constant number of integers, and 3) without SBP algorithm, the maximum number of Gaussian approximation equals two is not sufficient to maintain a good probability of symbol error performance. A higher number of Gaussian in the approximation is required.

For CLDLC, this dissertation has three major contributions. The first is reliability-based CLDLC decoding which is extended from LDLC. Similar to LDLC, in reliability-based decoding, the number of Gaussians in the check-to-variable message is adaptively selected depending upon the reliability. It provides a small number of Gaussians in each check-to-variable message; in particular, it allows the use of a single Gaussian when the message has high reliability. Fewer Gaussians means lower decoder complexity. However, we know from the real-valued case that two Gaussians approximation is not sufficient to maintain a good performance when the messages have low reliability. In CLDLC, more than two Gaussians can be selected for low-reliability messages. Reliability-based decoding of CLDLC also uses a threshold to select the number of Gaussians which is similar to LDLC, but two thresholds are defined to allow us selected more than two Gaussians. In addition, this threshold can be found using the Kullback-Leibler (KL) divergence between the approximation and the true distribution, but explicitly computing the divergence is inefficient. Instead, we form an upper bound on this KL divergence, and linear regression is used to efficiently estimate this threshold.

The second contribution is a CLDLC construction using Eisenstein integers (EI). This new construction reduces the complexity of message-passing decoders where an infinite Gaussian mixture is represented by a finite mixture. The advantage of the Eisenstein integers over the Gaussian integers is that the hexagonal Voronoi cells of the Eisenstein integer lattice has the tightest packing in two dimensions. As a result, the quality of the message-passing approximation is improved and mixtures have a smaller number of Gaussians. This smaller number of Gaussians leads to lower decoder complexity. While the extension to Eisenstien integers is straightforward, the contribution is that the Eisenstien integer construction lowers decoding complexity, compared to Gaussian integers.

Reliability-based decoding is applied to decoding both Eisenstein integer and Gaussian integer CLDLC constructions in this paper. For decoding GI-CLDLC, reliability-based decoding has lower complexity, and slightly better performance, than the previous CLDLC decoder of Yona and Feder [20]. Moreover, EI-CLDLC decoding has lower complexity than GI-CLDLC decoding for dimension $n \geqslant 49$.

The last contribution is a generalized CLDLC construction we call *relaxed Latin square*. Similar to real-number LDLC, CLDLC has a condition for exponential convergence of messages under belief-propagation decoding, specifically a condition on the matrix coefficients of the inverse generator matrix. We give a new convergence condition for the relaxed Latin square construction, which provides some guidance for designing the inverse generator matrix for a broader range of CLDLC than previously possible. This generalizes results in [19, 20]. The advantage of the relaxed Latin square is that each row and column does not require to have the same coefficients and weight d; or, it can be seen as an irregular CLDLC. Based on this property, we can design the triangular structure which is good for low complexity and fast encoding process, and it is suitable for the shaping operations.

1.5 Notation and Definitions

We use $\mathbb{C}$, $\mathbb{R}$ and $\mathbb{Z}$ to denote the fields of complex numbers, real numbers and integers, respectively. We use boldface lowercase and boldface uppercase to denote column vectors and matrices, e.g. $\mathbf{x}$ and $\mathbf{H}$, respectively. We use three interchangeable ways of writing complex numbers: $z = a + bi$, vector $\mathbf{z} = [a\ b]^T$, and $re^{i\theta}$ where $r = \sqrt{a^2 + b^2}$ and $\theta = \tan^{-1}(\frac{b}{a})$. Also, $x_{Re} = a$ and $x_{Im} = b$ denote the real and imaginary parts of the complex number x. The conjugate transpose or Hermitian transpose of $\mathbf{G}$ is $\mathbf{G}^{\dagger}$, and the complex conjugate of c is c^*. The n-by-n identity matrix is $\mathbf{I}_n$.

1.6 Organization

This dissertation is organized as follows. Chapter 2 describes a basic knowledge of lattice codes and fundamentals of CLDLC which are included CLDLC construction, BP decoder and the infinite Gaussian mixture in the BP decoding. Chapter 3 introduces a new CLDLC construction called relaxed Latin square, and a proof of variance convergence are also provided. Chapter 4 explains the proposed reliability-based decoding of real-valued LDLC. Chapter 5 explains the proposed reliability-based decoding of CLDLC Using Gaussian and Eisenstein integers. Chapter 6 shows the minor research which is expectation maximization (EM) algorithm for discrete memoryless channel (DMC) estimation in NAND flash memory. Chapter 7 summarizes the dissertation. The last chapter is about future works. In addition, the proof of the convergence condition for the relaxed Latin square construction and the upper bound on KL divergence are given in Appendix A and B, respectively.

Chapter 2

Complex Low-Density Lattice Codes (CLDLC)

The main role of this chapter is to introduce the fundamentals of complex low-density lattice codes (CLDLC). However, before explaining CLDLC, we provide basic knowledge of lattice codes which can help readers for understanding CLDLC easier.

LDLC defined over the real numbers was proposed by Sommer et al. [19]. These lattice codes can be encoded and decoded efficiently in high-dimensional Euclidean space, and error-free decoding is possible within 0.6 dB of the unconstrained power channel capacity [19].

Yona and Feder [20] described complex low-density lattice codes (CLDLC) which extend LDLC principles to lattices defined over the Gaussian (complex) integers [20]. CLDLC are naturally matched to complex-valued channels and they are a candidate for the compute-and-forward (C&F) relaying CLDLC increases the coding gain for small dimensional lattices compared to real LDLC [20]. An n-dimensional CLDLC is not simply a $2n$-dimensional real LDLC, but rather the CLDLC is directly generated over the domain of complex numbers. CLDLC outperforms $2n$-dimensional real LDLC because the real-valued parity check matrix normally suffers from short loops. Similar to the real LDLC case, CLDLC lattices have a sparse inverse generator matrix. Based on this property, CLDLC lattices can be decoded using belief propagation (BP) algorithm [22], however, the messages in the iterative processing of CLDLC are complex Gaussian functions. Due to the BP decoding, CLDLC confronts the same issue as real LDLC, which is that an infinite complex Gaussian mixture functions occurs in the iterative decoding process. This infinite complex Gaussian mixtures must be approximated for any implementation. The infinite Gaussian mixture approximated is presented in Chapter 5.

2.1 Definition of lattices

An n-dimensional lattice Λ is a discrete additive subgroup of Euclidean space $\mathbb{R}^n$. Since $\mathbb{R}^n$ is a vector space and Λ is a subgroup, so Λ is also vector subspace. Consequently, a lattice has the following properties.

- If lattice points $\mathbf{x}$, $\mathbf{y} \in \Lambda$, then $\mathbf{x} + \mathbf{y} \in \Lambda$.
- The zero point is a lattice point.
- A lattice Λ is a set of infinite size. For example, assume $\mathbf{x} \in \Lambda$, then $\mathbf{x}+\mathbf{x}+\cdots+\mathbf{x} \in \Lambda$.
- Assume $\mathbf{x} \in \Lambda$, and $a \in \mathbb{Z}$ is an integer, then $a \cdot \mathbf{x} \in \Lambda$.

In summary, lattice Λ is group under vector addition, and it is also Euclidean-space code.

2.2 Generator Matrix

An $n \times n$ generator matrix $\mathbf{G}$ whose columns are the basis vectors $\mathbf{g}_1, \mathbf{g}_2, \ldots, \mathbf{g}_n$ is the generator matrix of the lattice

$$\mathbf{G} = \begin{bmatrix} | & | & & | \\ \mathbf{g}_1 & \mathbf{g}_2 & \dots & \mathbf{g}_n \\ | & | & & | \end{bmatrix}. \tag{2.1}$$

Since a lattice λ is an n-dimensional subspace of $\mathbb{R}^n$, this space can be spanned by a set of n linearly independent basis vectors $\mathbf{g}_1, \mathbf{g}_2, \dots, \mathbf{g}_n$ where,

$$\mathbf{g} = \begin{bmatrix} g_1 \\ g_2 \\ \vdots \\ g_n \end{bmatrix} \tag{2.2}$$

is a column vector representing a point in $\mathbb{R}^n$.

A vector $\mathbf{x}$ is a lattice point if $\mathbf{x}$ can be formed as a linear combination of the basis vectors scaled by integers $b_i \in \mathbb{Z}$, where $i = 1, 2, \dots, n$. The lattice point $\mathbf{x}$ can be expressed as

$$\mathbf{x} = \mathbf{g}_1 b_1 + \mathbf{g}_2 b_2 + \dots + \mathbf{g}_n b_n. \tag{2.3}$$

Or it can be represented in the matrix form as:

$$\mathbf{x} = \mathbf{G}\mathbf{b}. \tag{2.4}$$

For example, the 2×2 hexagonal lattice, denoted as A_2 has the generator matrix

$$\mathbf{g} = \begin{bmatrix} \frac{\sqrt{3}}{2} & 0 \\ \frac{1}{2} & 1 \end{bmatrix}. \tag{2.5}$$

The A_2 lattice can be construct as in Fig. 2.1. The blue dots are the lattice point, and the blue lines are the Voronoi region of the lattice. Other property of the lattice is the kissing number, which is the number of closest neighbors to the lattice point. Lattice A_2 has kissing number equals six.

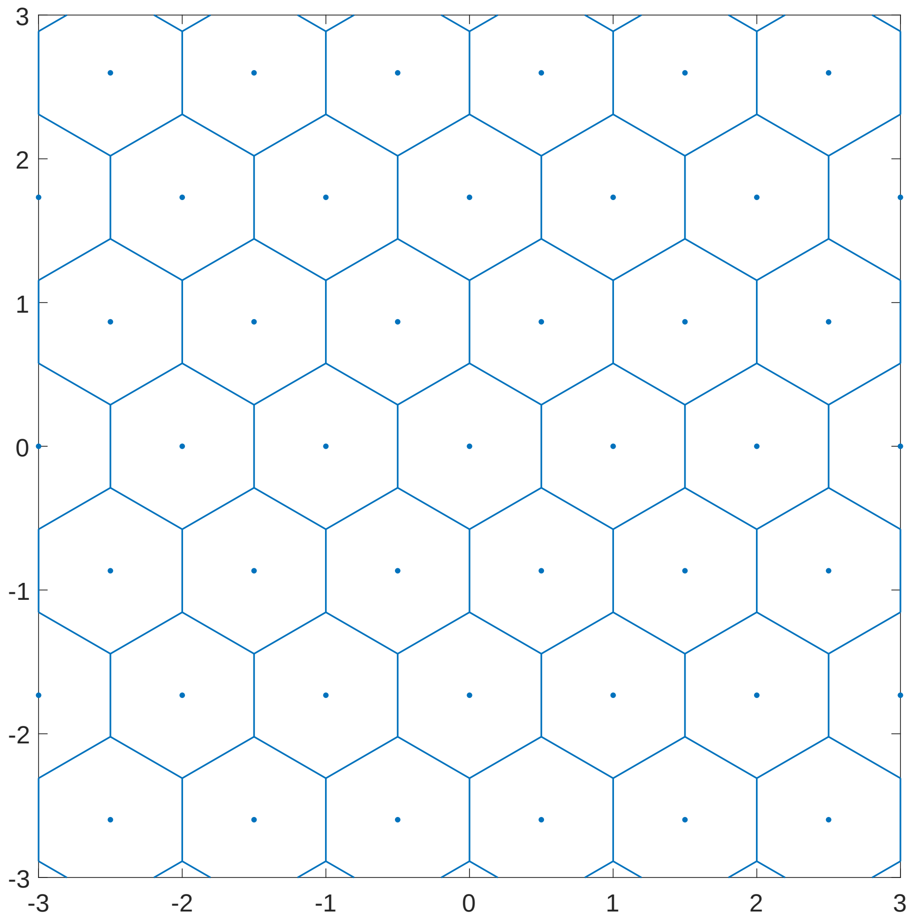

Figure 2.1: 2×2 hexagonal lattice A_2, the blue dots are the lattice points, and the blue lines are the Voronoi region of the lattice. Each lattice point of the A_2 lattice has 6 neighbors or the kissing number is 6.

The Voronoi region of a lattice point is the set of points in $\mathbb{R}^n$ are closer to a lattice point than to any other lattice point. The volume of the Voronoi region defined as $V(\Lambda)$ equals

$$V(\Lambda) = |\det(\mathbf{G})|. \tag{2.6}$$

For a lattice Λ, a fundamental region $\mathcal{V} \subset \mathbb{R}^n$ is a shape that is shifted by each lattice point in Λ will cover the whole real space $\mathbb{R}^n$ exactly. It can be express as:

$$\mathbb{R}^n = \bigcup_{\mathbf{x}\in\Lambda} \mathcal{V} + \mathbf{x} \ \text{ and} \tag{2.7}$$

$$\{\mathcal{V} + \mathbf{x}\} \cap \{\mathcal{V} + \mathbf{y}\} = \phi, \tag{2.8}$$

for any $\mathbf{x} \neq \mathbf{y}$. Any point $\mathbf{y} \in \mathbb{R}^n$ is in one fundamental region exactly.

2.3 Lattice Properties

2.3.1 Minimum Distance

A lattice is a Euclidean-space code, so it has a squared minimum distance d^2_{min}. Since a lattice is linear, d^2_{min} is given by:

$$d^2_{min} = \min_{\mathbf{x}\in\Lambda\backslash\mathbf{0}} ||\mathbf{x}||. \tag{2.9}$$

If any two points $\mathbf{x}$, $\mathbf{y}$ satisfy $||\mathbf{x}-\mathbf{y}^2|| = d^2_{min}$, then another lattice point $\mathbf{z} = \mathbf{x}-\mathbf{y}$ satisfies $||\mathbf{z}||^2 = d^2_{min}$ by linearity.

The squared minimum distance d^2_{min} of a lattice Λ, and the squared minimum distance d'^2_{min} of a scaled version $\Lambda' = k\Lambda$ are related by:

$$d^2_{min} = k^2 d^2_{min}. \tag{2.10}$$

A lattice could be scaled arbitrarily largely to increase d^2_{min}. The disadvantage is that this reduce the density of point (or increase the average power transmission for lattice codes) as well.

2.3.2 Coding Gain

The coding gain normalizes the minimum distance and is a useful measure of lattice's suitability for error-correction.

For an n- dimensional lattice Λ with squared minimum distance d_{min}^2, the coding gain is [29]:

$$\gamma = \frac{d_{min}^2}{V(\Lambda)^{2/n}}. \tag{2.11}$$

The coding gain is independent of lattice scaling. Coding gain can be interpreted as the reduction in average energy for using Λ as the codebook, over using the uncoded integer lattice $\mathbb{Z}^n$.

2.3.3 Packing radius and Packing Density

The packing radius ρ is half the minimum distance d_{min}. It can be expressed as:

$$\rho = \frac{d_{min}^2}{4}. \tag{2.12}$$

The packing density Δ of a lattice is the fraction of the whole space occupied by the spheres:

$$\Delta = \frac{\text{volume of one sphere}}{\text{volume of the fundamental region}}, \tag{2.13}$$

which is:

$$\Delta = \frac{V_n \rho^n}{V(\Lambda)}, \tag{2.14}$$

where ρ is the packing radius and V_n is the volume of a unit sphere in n-dimensions, given by:

$$V_n = \frac{2\pi^{2/n}}{n\Gamma(n/2)}. \tag{2.15}$$

Note that Γ is the gamma function.

The packing density Δ is related to the coding gain γ as the following:

$$\Delta^{2/n} = \frac{V_n}{4}\gamma. \tag{2.16}$$

2.3.4 Lattice Quantization

Lattice quantization or lattice decoding finds the lattice point $\mathbf{x} \in \Lambda$ which is closest to an arbitrary $\mathbf{y} \in \mathbb{R}^n$. Quantization refers to approximating an arbitrary real-valued vector $\mathbf{y}$ by a lattice point $\mathbf{x} \in \Lambda$. Decoding refers to finding the lattice point closest to a lattice point plus noise, $\mathbf{y} = \mathbf{x} + \mathbf{z}$. In either case, $\mathbf{x}$ is the lattice point which is closest to $\mathbf{y}$ in the Euclidean distance. $\mathbf{x}$ can be calculated from

$$\mathbf{x} = \arg\min_{\lambda \in \Lambda} ||\mathbf{y} - \lambda||^2. \tag{2.17}$$

2.4 CLDLC and Unconstrained Power AWGN System

An n-dimensional complex lattice Λ is a discrete additive subgroup of $\mathbb{C}^n$, defined by a non-singular square generator matrix $\mathbf{G}$. A lattice point $\mathbf{x}$ is an integral linear combination of basis vectors in $\mathbf{G}$. Each lattice point is constructed from:

$$\mathbf{x} = \mathbf{G}\mathbf{b}. \tag{2.18}$$

Complex-valued lattices are a generalization of real-valued lattices. In CLDLC, the inverse of $\mathbf{G}$, $\mathbf{H} = \mathbf{G}^{-1}$ is restricted to be sparse to develop a linear-time iterative decoding scheme [19], where $\mathbf{H}$ is called the check matrix or inverse generator matrix. For encoding, CLDLC can use the Jacobi method as in the real-valued LDLC case [19]. In this dissertation. we consider two types of complex integers $\mathbf{b}$, Gaussian integers (GI) $\mathbb{Z}[i]$ ($i = \sqrt{-1}$) and Eisenstein integers (EI) $\mathbb{Z}[\omega]$ where $\omega = \frac{-1+i\sqrt{3}}{2}$. A vector of GIs is written $\mathbf{b} = (b_{1,Re}+ib_{1,Im}, b_{2,Re}+ib_{2,Im}, ..., b_{n,Re}+ib_{n,Im})$, and a vector of EIs is written $\mathbf{b} = (b_{1,Re}+\omega b_{1,Im}, b_{2,Re}+\omega b_{2,Im}, ..., b_{n,Re}+\omega b_{n,Im})$, where $b_{Re}, b_{Im} \in \mathbb{Z}^n$.

The Voronoi region of a lattice point is the set of points in $\mathbb{C}^n$ that are closest to the lattice point. For a square generator matrix $\mathbf{G}$, the Voronoi region volume for lattices with Gaussian integers equals

$$V(\Lambda) = \det(\mathbf{G}^{\dagger}\mathbf{G}). \tag{2.19}$$

For lattices with Eisenstein integers $V(\Lambda)$ equals

$$V(\Lambda) = \left(\frac{\sqrt{3}}{2}\right)^n \det(\mathbf{G}^{\dagger}\mathbf{G}) \tag{2.20}$$

[30]. We consider here the unconstrained power system. This paper uses the volume-to-noise ratio (VNR) as an analog to the signal-to-noise ratio. The VNR for Gaussian and Eisenstein integer lattices is defined as:

$$VNR = \frac{V(\Lambda)^{2/n}}{2\pi e \sigma^2}. \tag{2.21}$$

The Poltyrev capacity [31], or unconstrained lattice capacity, is $\sigma^2 = \frac{1}{2\pi e}$ [32], which corresponds to $VNR = 0$ dB. In the sequel we normalize the generator matrix $\mathbf{G}$ or its determinant such that $V(\Lambda) = 1$.

Since we consider the unconstrained lattice only, a complex lattice point $\mathbf{x} \in \Lambda$ is transmitted over a complex additive white Gaussian noise (CAWGN) channel, and the received sequence $\mathbf{y} = (y_1, y_2, ..., y_n) \in \mathbb{C}^n$ is:

$$\mathbf{y} = \mathbf{x} + \mathbf{z} \tag{2.22}$$

where the vector $z_j \sim \mathcal{CN}(\mathbf{0}, \sigma^2\mathbf{I}_2)$ is complex additive Gaussian noise. $\mathcal{CN}$ denotes complex-normal distribution, σ^2 is noise variance of each element of complex number. The maximum likelihood lattice point estimate $\hat{\mathbf{x}}$ is:

$$\hat{\mathbf{x}} = \arg\max_{\mathbf{x}\in\Lambda} \Pr(\mathbf{y}|\mathbf{x}). \tag{2.23}$$

If $\mathbf{x} = \hat{\mathbf{x}}$ the correct codeword is received, or an error occurred otherwise.

2.5 Inverse Generator Matrix/Check Matrix H

Similar to low-density parity-check (LDPC) codes, CLDLC codes are also block codes with inverse generator matrix or check matrices $\mathbf{H}$ that contain only a very small number of non-zero entries. This sparseness of $\mathbf{H}$ is essential for an iterative decoding complexity that increases linearly with the code length. The biggest difference between LDPC and CLDLC codes is LDPC codes design is based on the finite file $\mathbf{GF}(2)$ while CLDLC is based on the selection of the $h_1, h_2, \ldots, h_i$ coefficients , where $h_i \in \mathbb{C}$.

One of the well-known real number LDLC constructions is the Latin square construction for check matrix $\mathbf{H}$ which was proposed by Sommer et al. [19] and extended by Yona and Feder [20] to CLDLC. A Latin square CLDLC can be constructed by designing a check matrix $\mathbf{H}$ having constant row and column weight d. Let

$$\mathbf{h} = [1, h_2, h_3, ..., h_d] \tag{2.24}$$

be a generating sequence, where $h_i \in \mathbb{C}$. The non-zero entries of each row and each column are $h_1, h_2, h_3, ..., h_d$, so that each row (respectively, column) is a permutation of any other row (column), except for complex rotations, including sign changes. In addition, $\mathbf{H}$ should be 4-cycle free. An n dimensional CLDLC can be considered as a regular construction if all the row degrees and column degrees of the check matrix are equal to a common degree d.

A sufficient condition [19] to achieve exponential convergence of the message variance is to select the generator sequence such that:

$$\alpha = \frac{\sum_{j=2}^{d} |h_j|^2}{|h_1|^2} < 1, \tag{2.25}$$

where $|\cdot|$ denotes the Euclidean norm [19]. To achieve the condition in (2.25), a generator sequence $\mathbf{h}$ must satisfy $|h_1| \geqslant |h_2| \geqslant ... \geqslant |h_d|$. For example, the matrix $\mathbf{H}$ is a parity check matrix of Latin square CLDLC with lattice dimension $n = 8$, degree $d = 3$, the generating sequence is $|h_1| = 1$ and $|h_2| = |h_3| = \frac{1}{\sqrt{d}}$, and in the sequel, we normalize $V(\Lambda) = 1$. An example check matrix $\mathbf{H}$ is:

$$\mathbf{H} = \begin{bmatrix} 1 & 0 & 0 & h_2 & 0 & 0 & h_3 & 0 \\ h_2^* & 1 & 0 & 0 & 0 & 0 & 0 & h_3 \\ 0 & -h_3^* & 1 & 0 & 0 & 0 & 0 & h_2 \\ 0 & 0 & -h_3^* & 1 & 0 & 0 & h_2 & 0 \\ -h_3 & 0 & 0 & 0 & 1 & h_2 & 0 & 0 \\ 0 & 0 & -h_2^* & h_3 & 0 & 1 & 0 & 0 \\ 0 & -h_2^* & 0 & 0 & h_3 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & h_2 & h_3 & 0 & 1 \end{bmatrix}, \tag{2.26}$$

where the generating sequence $\mathbf{h}$ in the rectangular coordinate form is $\{1, \sqrt{\frac{2}{9}} + i\sqrt{\frac{1}{9}}, \sqrt{\frac{2}{9}} + i\sqrt{\frac{1}{9}}\}$.

2.6 Bipartite Graph

Similar to low-density parity-check (LDPC) codes, CLDLC can be decoded using a bipartite graph: it is a graph with variable nodes at one side and check nodes at the other side. Figure 2.2 (i) represents the bipartite graph which is constructed from the check matrix $\mathbf{H}$ in (2.26). V_j and C_i represent the check node i and the variable node j, respectively. Each variable node corresponds to a single element of the lattice codeword $\mathbf{x} = \mathbf{Gb}$ in (2.18). Each row of the check matrix $\mathbf{H}$ corresponds to a check node which is represented the equation of the form $\sum_k h_k x_{i_k} = integer$, where i_k denotes the positions of non-zero elements at the corresponding row of the check matrix $\mathbf{H}$, h_k are the values of $\mathbf{H}$ at these non-zero element locations and the *integer* at the right-hand side is unknown.

Figure 2.2 (a) - (h) give an illustration of how to construct a bipartite graph from (2.26). For example, at the first row of the check matrix, $\mathbf{H}$ corresponding to the check node C_1, positions 1, 4, and 7 are non-zero elements which are 1, h_2 and h_3 respectively. The check node C_1 will be connected to the non-zero element positions which are the variable node V_1, V_4, and V_7, then each edge is assigned value 1, h_2 and h_3, respectively. Edges of check node C_2 to C_8 can be constructed in the similar way. Finally, Figure 2.2 (a) to (h) are combined together and the completed bipartite graph is shown in Figure 2.2 (i).

On the bipartite graph, a cycle is a sequence of connected nodes which starts and ends at the same node in the graph. The length of a cycle is the number of edges it contains, and it is always an even number. Generally, longer cycles yield better error rate performance of decoding because there are more nodes that can exchange the information. A 4-cycle exists when two variable nodes are both connected to the same pair of check nodes degrade the decoding performance, so it should be avoided.

2.7 Belief Propagation Decoder

In this section the decoding algorithm for CLDLC lattices is described. The decoding algorithm is based on the belief propagation (BP) algorithm.

Belief propagation is also known as sum–product message passing algorithm. This technique was invented in 1982 by Pearl [33], and it is commonly used in artificial intelligence and information theory, and has demonstrated experimental success in numerous applications such as low-density parity-check codes, turbo codes, free energy approximation, and satisfiability. We called this algorithm the message-passing because their operation can be described by passing of messages along the edges of a bipartite graph. Each bipartite graph node works individually, having access only to the messages on the edges connected to it. Message-passing algorithms are a type of iterative decoding algorithm; the messages pass back and forward between the variable nodes and check nodes iteratively until reaching the target number of iterations. In addition, they are named for sum–product because the check node implements the summation and the variable node the product.

In 2008, Sommer et al. introduced BP decoder for real-valued LDLC, and the messages between variable and check nodes are the probability density function (pdf) of Gaussian function (in the case that channel is Additive white Gaussian noise (AWGN)). This decoder has very high complexity. Then, [23] introduced a parametric LDLC decoding algorithm the pdf of Gaussian function can be represented by 3 parameters which are mean, variance and amplitude. Further, using

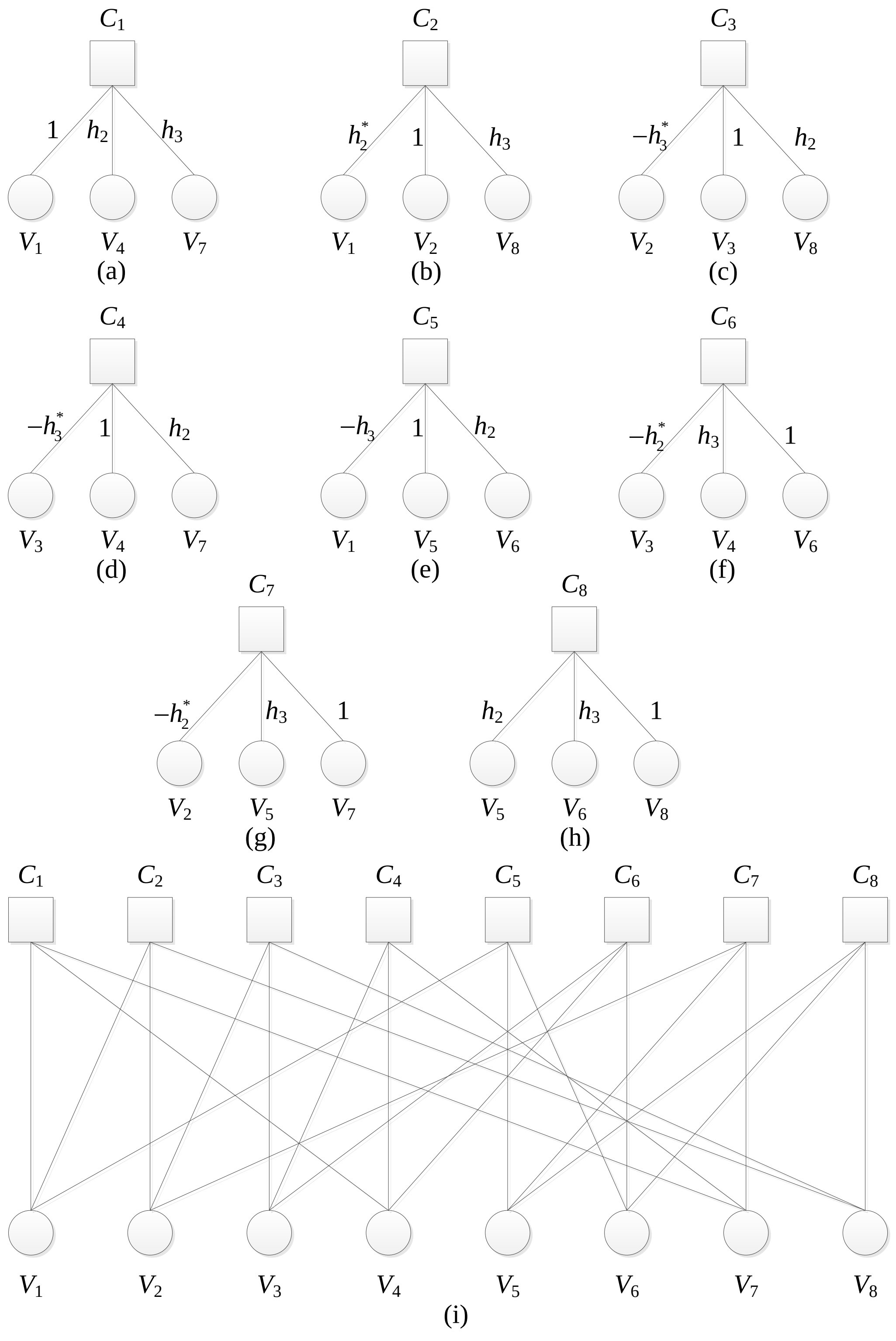

Figure 2.2: The bipartite graph of an CLDLC constructed from (2.26).

single Gaussians as the messages between the variable and check nodes leads to reduced memory requirements with minor performance penalty [8]. Yona and Feder [20] extended the parametric BP decoder based on the real-valued LDLC to the complex numbers for CLDLC decoder in 2010.

2.7.1 Operations on Complex Gaussian Mixtures

This section describes operations on complex Gaussian functions which are messages in belief propagation decoding, described in Section 2.7.2. The probability density function (pdf) of a complex Gaussian function with 2×1 mean vector $\mathbf{m}$ and 2×2 covariance matrix $\mathbf{V}$ is:

$$\mathcal{N}(\mathbf{z};\mathbf{m},\mathbf{V}) = \frac{1}{2\pi\sqrt{|\mathbf{V}|}} e^{-\frac{1}{2}(\mathbf{z}-\mathbf{m})^T\mathbf{V}^{-1}(\mathbf{z}-\mathbf{m})}. \tag{2.27}$$

From the AWGN channel assumption, the BP messages are Gaussian mixtures. The message $f(\mathbf{z})$ is a mixture of N Gaussians,

$$f(\mathbf{z}) = \sum_{j=1}^{N} c_j \mathcal{N}(\mathbf{z};\mathbf{m}_j,\mathbf{V}_j), \tag{2.28}$$

where $c_j \geq 0$ are mixing coefficients with $\sum_{j=1}^{N} c_j = 1$, $\mathbf{m}_j$ and $\mathbf{V}_j$ are mean and covariance of the Gaussian mixture. In this way, each Gaussian mixture can be described by a set of triples $(\mathbf{m}_1,\mathbf{V}_1,c_1),...,(\mathbf{m}_N,\mathbf{V}_N,c_N)$.

Let $f(\mathbf{z}) = \sum_{j=1}^{N} f_j(\mathbf{z})$ and $g(\mathbf{z}) = \sum_{k=1}^{M} g_k(\mathbf{z})$ be two Gaussian mixtures. Their product is a mixture of NM Gaussians. Each mixture element is the pairwise product of two components $f_j(\mathbf{z}) = c_1\mathcal{N}(\mathbf{z};\mathbf{m}_1,\mathbf{V}_1)$ and $g_k(\mathbf{z}) = c_2\mathcal{N}(\mathbf{z};\mathbf{m}_2,\mathbf{V}_2)$, a Gaussian $s(\mathbf{z}) = c\mathcal{N}(\mathbf{z};\mathbf{m},\mathbf{V})$ with mean $\mathbf{m}$, variance $\mathbf{V}$ and mixing coefficient c given by:

$$\mathbf{V} = (\mathbf{V}_1^{-1} + \mathbf{V}_2^{-1})^{-1}, \tag{2.29}$$

$$\mathbf{m} = \mathbf{V}(\mathbf{V}_1^{-1}\mathbf{m}_1 + \mathbf{V}_2^{-1}\mathbf{m}_2), \tag{2.30}$$

$$c = \frac{c_1 c_2}{2\pi\sqrt{|\mathbf{V}_1 + \mathbf{V}_2|}} e^{-\frac{1}{2}(\mathbf{m}_1-\mathbf{m}_2)^T(\mathbf{V}_1+\mathbf{V}_2)^{-1}(\mathbf{m}_1-\mathbf{m}_2)}. \tag{2.31}$$

Let $f(\mathbf{z})$ be a Gaussian mixture. The moment matching approximation (MM) which minimizes the Kullback-Leiber divergence between $f(\mathbf{z})$ and $q(\mathbf{z})$ is used to approximate $f(\mathbf{z})$ with a single Gaussian $q(\mathbf{z})$ [34]. The MM approximation finds the single Gaussian $q(\mathbf{z})$ which has the same mean $\mathbf{m}$ and variance $\mathbf{V}$ as $f(\mathbf{z})$, given by:

$$\mathbf{m} = \sum_{j=1}^{N} c_j \mathbf{m}_j, \tag{2.32}$$

$$\mathbf{V} = \sum_{j=1}^{N} c_j(\mathbf{V}_j + (\mathbf{m}_j - \mathbf{m})(\mathbf{m}_j - \mathbf{m})^T). \tag{2.33}$$

This operation is denoted as:

$$q(\mathbf{z}) = MM(f(\mathbf{z})). \tag{2.34}$$

2.7.2 Parametric Belief Propagation Decoder

BP decoding of CLDLC confronts the common issue as real LDLC, that there is an infinite Gaussian mixture in the iterative decoding. This infinite Gaussian mixture must be approximated for the decoder implementation. Due to this issue, this dissertation proposes to approximate the infinite complex Gaussian mixture function with a finite number of complex Gaussian functions using the reliability of check-to-variable messages and a threshold function. This approximation takes place at the variable node. This subsection describes the overall picture of the CLDLC parametric BP decoding algorithm. Details of the approximation at the variable node is explained in Chapter 5.

CLDLC parametric BP decoding can be performed on a bipartite graph, where rows of $\mathbf{H}$ correspond to check nodes, and columns of $\mathbf{H}$ correspond to variable nodes. The variable-to-check messages are $q_k(z)$ and the check-to-variable messages are $R_k(z)$. The proposed CLDLC reliability parametric BP decoding algorithm is as follows.

- *Initialization*: For the AWGN channel (3.3), the initial variable-to-check message is

$$y_i(\mathbf{z}) = \mathcal{N}(\mathbf{z}; \mathbf{y}_i, \sigma^2 \mathbf{I}_2), \tag{2.35}$$

  for $i = 1, 2, ..., n$. Received messages are sent to connected check nodes in the initial step.

- *Check-to-variable message*: The check node incoming messages are $k = 1, ..., d-1$ single Gaussians and output is $k = d$. The corresponding non-zero coefficients from $\mathbf{h}$ are $h_1, ..., h_d$. The output $R_d(\mathbf{z})$ is:

$$R_d(\mathbf{z}) = \mathcal{N}(\mathbf{z}; \mathbf{m}_d, \mathbf{V}_d). \tag{2.36}$$

  where mean $\mathbf{m}_d$ and covariance matrix $\mathbf{V}_d$ can be found from

$$\mathbf{m}_d = -[h_d]^{-1} \sum_{k=1}^{d-1} [h_k] \mathbf{m}_k, \tag{2.37}$$

$$\mathbf{V}_d = \sum_{k=1}^{d-1} [\frac{h_k}{h_d}] \mathbf{V}_k [\frac{h_k}{h_d}]^T \tag{2.38}$$

  with $[\frac{h_k}{h_d}] = \begin{bmatrix} Re\{\frac{h_k}{h_d}\} & -Im\{\frac{h_k}{h_d}\} \\ Im\{\frac{h_k}{h_d}\} & Re\{\frac{h_k}{h_d}\} \end{bmatrix}$.

- *Variable-to-check message*: The messages $R_k(z)$ coming from the check nodes are expanded to a periodic function $\widetilde{R}_k(z)$, but with a finite number of Gaussians depending on their reliability. This expansion or the approximation number of finite Gaussian will be described in Chapter 5. Then, the moment matching approximation MM is used to find the single-Gaussian message $f_d(z)$ sent back to the check node:

$$f_d(\mathbf{z}) = y_i(\mathbf{z}) \prod_{k=1}^{d-1} \widetilde{R}_k(\mathbf{z}), \text{ and} \tag{2.39}$$

$$q_d(\mathbf{z}) = MM(f_d(\mathbf{z})). \tag{2.40}$$

Efficient implementation of the variable node is one of the contributions of this dissertation. Section 5.1 shows how to select Gaussians near the channel value y_i. The specific number of Gaussians selected should be as small as possible, and are selected using reliability as shown in Section 5.2.1. Section 5.2.3 gives the variable node function in detail.

The variable node computes the product of mixtures of Gaussians using (2.29)–(2.31). The outputs of variable node which are Gaussian mixtures will be approximated by a single Gaussian using MM approximation using (2.32)–(2.34), before sending the messages to check nodes.

- *Final decision*: At the last iteration the product without omitting any message is performed:

$$q_i^{final}(\mathbf{z}) = y_i(\mathbf{z}) \prod_{k=1}^{d} \widetilde{R}_k(\mathbf{z}). \tag{2.41}$$

The lattice point estimate $\hat{\mathbf{x}}$ and the integer estimate $\hat{\mathbf{b}}$ are:

$$\hat{\mathbf{x}}_i = \arg\max_{\mathbf{z}} q_i^{final}(\mathbf{z}), \text{ and} \tag{2.42}$$

$$\hat{\mathbf{b}} = \lfloor \mathbf{H}\hat{\mathbf{x}} \rceil. \tag{2.43}$$

Note that a forward-backward recursion is applied at the variable node to reduce the number of operations inside variable node. This recursion is similar to the existing forward-backward algorithm of [25] (real-valued case) but it is different [20] (complex-valued case) in how the channel value $\mathbf{y}_i$ is handled, in [25] the channel message $\mathbf{y}_i$ is multiplied at the last step of the recursion while in [20] the channel message $\mathbf{y}_i$ is multiplied at the first step of the recursion.

The forward-backward recursion is done as follows :

1) The forward recursion defined as:

$$\alpha_k(\mathbf{z}) = \alpha_{k-1}(\mathbf{z}) \cdot \widetilde{R}_k(\mathbf{z}) \tag{2.44}$$

for $k = 2, 3, \ldots, d-1$ with $\alpha_1(\mathbf{z})$ initialized as equal to $\widetilde{R}_1(\mathbf{z})$.

2) The backward recursion $\beta_k(\mathbf{z})$ is computed, for $k = d, d-1, d-2, \ldots, 1$ as:

$$\beta_{k-1}(\mathbf{z}) = \beta_k(\mathbf{z}) \cdot \widetilde{R}_{k-1}(\mathbf{z}), \tag{2.45}$$

with $\beta_k(\mathbf{z})$ initialized as the approximation of $\widetilde{R}_d(\mathbf{z})$.

3) Then combining the forward and backward recursion, we get:

$$\widetilde{f}_d(\mathbf{z}) = \alpha_k(\mathbf{z}) \cdot \beta_{k-1}(\mathbf{z}). \tag{2.46}$$

4) Finally, the single complex Gaussian output of the variable node is calculated by using the moment matching approximation:

$$f_d(\mathbf{z}) = MM(y_i(\mathbf{z}) \cdot \widetilde{f}_d(\mathbf{z})). \tag{2.47}$$

2.8 Concluding Remarks

This chapter starts with the basic knowledge of lattice codes and lattice properties. Then, we describe the fundamental knowledge of complex low-density lattice codes (CLDLC) in this chapter. Similar to low-density parity-check (LDPC) codes, CLDLC codes are also block codes with inverse generator matrix or check matrices **H** that contain only a very small number of non-zero entries. CLDLC lattices have a sparse inverse generator matrix. Based on this property, CLDLC can be decoded on principles similar to LDPC codes using a belief propagation (BP) decoding algorithm. The biggest difference is CLDLC defined over the complex numbers while LDPC codes design is based on the finite file $\mathbf{GF}(2)$. Since CLDLC is definded based on the complex number, the messages in the iterative decoding process are the probability density functions (pdf) of complex Gaussian for each lattice symbol, while the messages in the iterative decoding process of LDPC are the log likelihood ratios (LLR).

For the encoding part, we start from how to construct lattice points, inverse generator matrix/check matrix, and the unconstrained power AWGN system.

For decoding part, we introduce the bipartite graph and parametric belief propagation (BP) decoder. Especially, the BP decoding process illustrates how the infinite complex Gaussian mixtures happen.

Chapter 3

Relaxed Latin Square Construction for CLDLC

In this chapter, we propose a new CLDLC construction called *relaxed Latin square* for a check matrix $\mathbf{H}$. This construction is extended from the Latin square construction which was proposed by Sommer et al. for real number LDLC [19] and it was proposed by Yona and Feder [20] for CLDLC. The Latin square matrix for LDLC [19] and CLDLC [20] can be constructed by designing a check matrix $\mathbf{H}$ having constant row and column weight d, or this construction can be considered as a regular LDLC and CLDLC. Unlike existing works, the relaxed Latin square that we proposed has fewer restrictions for the coefficient and weight design. Each row and column does not require to have the same coefficients and weight d; or, it can be seen as an irregular CLDLC.

Sommer et al. [19] provided a condition for exponential convergence of message variances under belief-propagation decoding for real-number LDLC, specifically a condition on the matrix coefficients of the check matrix $\mathbf{H}$. For CLDLC, [20] extended the condition on the convergence of the variances during BP decoding of Latin square LDLCs from the real case [19] to the complex case, but did not give a proof. We give a new convergence condition for the relaxed Latin square construction, which provides some guidance for designing the inverse generator matrix for a broader range of CLDLC than previously possible. The proof of exponential convergence of variances under belief-propagation decoding is also provided in Appendix A.

Since the variance convergence proof of the relaxed Latin square construction of CLDLC is extended from the Latin square construction of real-valued LDLC [19]. The Latin square construction, the belief-propagation decoding, and the variance convergence of real-valued LDLC will be introduced as a background at the beginning of this chapter. Afterward, the relaxed Latin square construction of CLDLC, its convergence conditions, and the proof of exponential convergence of variances will be explained

3.1 Real-Valued LDLC

Construction and decoding of real-valued LDLC are similar to CLDLC that have been introduced in Section 2.4, but it is defined on the real numbers $\mathbb{R}$.

An n-dimensional real-valued lattice Λ is a discrete additive subgroup of $\mathbb{R}^n$, defined by a non-singular square generator matrix $\mathbf{G}$ whose column are the basis vectors. A lattice point $\mathbf{x}$ is the set of all linear combinations of basis vectors in $\mathbf{G}$. Each lattice point is constructed from:

$$\mathbf{x} = \mathbf{G}\mathbf{b}. \tag{3.1}$$

$\mathbf{b}$ is an n-dimensional vector of integers, and the check matrix is $\mathbf{H} = \mathbf{G}^{-1}$. The Voronoi region volume equals

$$V(\Lambda) = \det(\mathbf{G}). \tag{3.2}$$

We consider here the unconstrained power system. An LDLC lattice point $\mathbf{x} \in \Lambda$ is transmitted over an additive white Gaussian noise (AWGN) channel, and the received sequence $\mathbf{y} = (y_1, y_2, ..., y_n) \in \mathbb{R}^n$ is:

$$\mathbf{y} = \mathbf{x} + \mathbf{z} \tag{3.3}$$

where the vector $\mathbf{z} \sim \mathcal{N}(0, \sigma^2)$ is an additive Gaussian noise. The Gaussian probability distribution is defined as:

$$\mathcal{N}(z; m, \sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(z-m)^2}{2\sigma^2}}. \tag{3.4}$$

The probability density function (pdf) of a real-valued Gaussian function has mean m and variance σ^2.

Since we consider here the unconstrained power system. The volume-to-noise ratio (VNR) is defined as:

$$VNR = \frac{V(\Lambda)^{2/n}}{2\pi e \sigma^2}. \tag{3.5}$$

The Poltyrev capacity, or unconstrained lattice capacity, is $\sigma^2 = \frac{1}{2\pi e}$ [32], which corresponds to $VNR = 0$ dB.

3.1.1 Latin Square Construction of Real-Valued LDLC

Sommer et al. introduced the Latin square construction for the check matrix $\mathbf{H}$ for LDLC. Every row and column of an n-dimensional Latin square matrix has the same d nonzero real-valued coefficients (except for sign changes). A sufficient condition to achieve exponential convergence of the message variance in the BP decoder is to select the generator sequence $h_1 \geqslant h_2 \geqslant ... \geqslant h_d$ such that:

$$\alpha = \frac{\sum_{j=2}^{d} h_j^2}{h_1^2} < 1. \tag{3.6}$$

For example, the matrix

$$\mathbf{H} = \begin{bmatrix} 0 & -0.8 & 0 & -0.5 & 1 & 0 \\ 0.8 & 0 & 0 & 1 & 0 & -0.5 \\ 0 & 0.5 & 1 & 0 & 0.8 & 0 \\ 0 & 0 & -0.5 & -0.8 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0.5 & 0.8 \\ 0.5 & -1 & -0.8 & 0 & 0 & 0 \end{bmatrix}, \tag{3.7}$$

is a check matrix $\mathbf{H}$ of the Latin square LDLC with lattice dimension $n = 6$, degree $d = 3$ and generating sequence $h_1 = 1$, $h_2 = 0.8$, $h_3 = 0.5$. α can be calculated as this example:

$$\alpha = \frac{h_2^2 + h_3^2}{h_1^2} = \frac{0.8^2 + 0.5^2}{1^2} = 0.89 < 1. \tag{3.8}$$

This check matrix $\mathbf{H}$ should be further normalized by the constant $\sqrt[n]{|\det(\mathbf{H})|}$ in order to have $|\det(\mathbf{H})| = |\det(\mathbf{G})| = 1$, as required by the Definition 1 in [19].

3.1.2 Belief-Propagation Decoding of Real-Valued LDLC

The LDLC BP decoding algorithm proposed by Sommer et al [19] the messages are real pdf functions over the interval $(-\infty, \infty)$, whereas the parametric BP decoding algorithm of CLDLC in Section 2.7.2 represents the pdf of the complex Gaussian by 4 parameters which are mean, variance, covariance, and message coefficient. The LDLC decoder algorithm presented in [1] is as follows:

Denote the variable nodes by $v_1, v_2, \ldots, v_n$ and the check nodes by $c_1, c_2, \ldots, c_n$.

- *Initialization*: each variable node v_k sends to all its check nodes a Gaussian function message $f(z)$

  $$f_k(z) = y_k(z) = \mathcal{N}(z; y_k, \sigma^2), \tag{3.9}$$

  for $k = 1, 2, 3, \ldots n$.

- *Check-to-variable message*: The messages that the check node c_k send back to the variable node are calculated in three steps.

  1) The convolution step: all messages except $f_j(z)$, are convolved after expanding each message by its generator sequence h:

  $$\widetilde{p}_j(z) = f_1\left(\frac{z}{h_1}\right) \circledast \cdots \circledast f_{j-1}\left(\frac{z}{h_{j-1}}\right) \circledast f_{j+1}\left(\frac{z}{h_{j+1}}\right) \circledast \cdots \circledast f_d\left(\frac{z}{h_d}\right), \tag{3.10}$$

  where $j = 1, 2, \ldots, d$ and $\circledast$ denotes convolution.

2) The stretching step: The messages $\widetilde{p}_j(z)$ are stretched by $-h_j$ to:

$$p_j(z) = \widetilde{p}_j(-h_j z). \tag{3.11}$$

3) The periodic extension: The message $p_j(z)$ is extended to a periodic function with period $\frac{1}{|h_j|}$:

$$R_j(z) = \sum_{i=-\infty}^{\infty} p_j \left(z - \frac{i}{h_j} \right). \tag{3.12}$$

- *Variable-to-check message*: each variable node v_k sends a message back to each check node that is connected to it. For a variable node v_k, it is assumed that there are variable-to-check messages $c_{k_1}, c_{k_2}, \dots, c_{k_d}$ are sent to the check nodes, where d is the appropriate column weight of the check matrix $\mathbf{H}$. The message c_{k_e} that is sent back to the check node is calculated in two following steps.

1) The product step is

$$\widetilde{f}_j(z) = e^{-\frac{(y_k - z)^2}{2\sigma^2}} \prod_{l=1,l}^{d} R_l(z). \tag{3.13}$$

2) The normalization step is

$$f_j(z) = \frac{\widetilde{f}_j(z)}{\int_{-\infty}^{\infty} \widetilde{f}_j(z) dx} \tag{3.14}$$

These steps are repeated until reaching the desired number of iterations.

- *Final decision*: After completing the desired number of iterations, the integer information vector $\mathbf{b}$ is estimated. First, the final pdfs of the codeword elements is

$$\widetilde{f}_{final}(z) = e^{-\frac{(y_k - z)^2}{2\sigma^2}} \prod_{l=1}^{d} R_l(z). \tag{3.15}$$

Then, estimate each codeword by finding the peak of its PDF as

$$\hat{\mathbf{x}} = \arg\max_{\mathbf{z}} \widetilde{f}_{final}(z). \tag{3.16}$$

Finally, we estimate **b** as

$$\hat{\mathbf{b}} = \lfloor \mathbf{H}\hat{\mathbf{x}} \rceil. \tag{3.17}$$

The $\lfloor \cdot \rceil$ denotes rounding to the closest integer operation.

Note that for both variable-to-check messages and check-to-variable messages, all Gaussians that in a given mixture have the same variance.

3.1.3 Convergence of Variances

The messages in the iterative decoding process are probability density functions (pdfs) for each lattice symbol. The convergence of the message variances to zero implies that the Gaussians approach impulses, this is an important property to decode successfully. The convergence rate, slow or fast depends on the design of the check matrix **H**. If the design of the generator sequences $h_1, h_2, \ldots h_d$ follow (3.6), the convergence rate of message variances can be converged exponentially. If $\alpha \geq 1$ the variances may still converge, but the convergence rate may be as slow as $\mathcal{O}(1/t)$ at iteration t [19].

In this section, we introduce the proof of the variance convergence for the LDLC lattice decoder in [19].

In the Latin square LDLC construction, there are d edge types corresponding to the d coefficients. For any iteration t, the variable-to-check messages (3.14) that are sent along edges with the same absolute value have the same variance.

Let $v_i^{(t)}$ be the mixture variance for the input of a check node along the edge for h_i, for $i = 1, 2, 3, \ldots, d$, and the mixture variance of the outgoing message to variable node is u_i.

Consider $d = 4$ and the convergence of v_2. The variance at the variable node output is given by:

$$\frac{1}{v_2^{(t+1)}} = \frac{1}{u_1} + \frac{1}{u_3} + \frac{1}{u_4} + \frac{1}{\sigma^2} \geq \frac{1}{u_1}, \tag{3.18}$$

where $u_i \geq 0$ is assumed.

For the check node, the mixture variance of the outgoing massage u_1 is:

$$u_1 = \frac{h_2^2 v_2 + h_3^2 v_3 + h_4^2 v_4}{h_1^2}. \tag{3.19}$$

If we assume that $v_1 \geq v_2 \geq v_3 \geq v_4$, (3.19) also can be written as:

$$u_1 \leq \frac{h_2^2 + h_3^2 + h_4^2}{h_1^2} \cdot v_2 = \alpha v_2. \tag{3.20}$$

Then, (3.18) and (3.20) are combined. Finally, we get

$$\frac{1}{v_2^{(t+1)}} \geq \frac{1}{u_1}, \tag{3.21}$$

$$\frac{1}{v_2^{(t+1)}} \geq \frac{1}{\alpha v_2^{(t)}}, \tag{3.22}$$

$$v_2^{(t+1)} \leq \alpha v_2^{(t)}. \tag{3.23}$$

Since $v_2^{(1)} = \alpha^2$, the sequence of $v_2^{(t)}$ is going to approach to zero exponentially in the t iterations. To obtain the exponential convergence of the decoding α must satisfy this condition $\alpha < 1$. Fig 3.1 shows the message propagation for variance converge analysis.

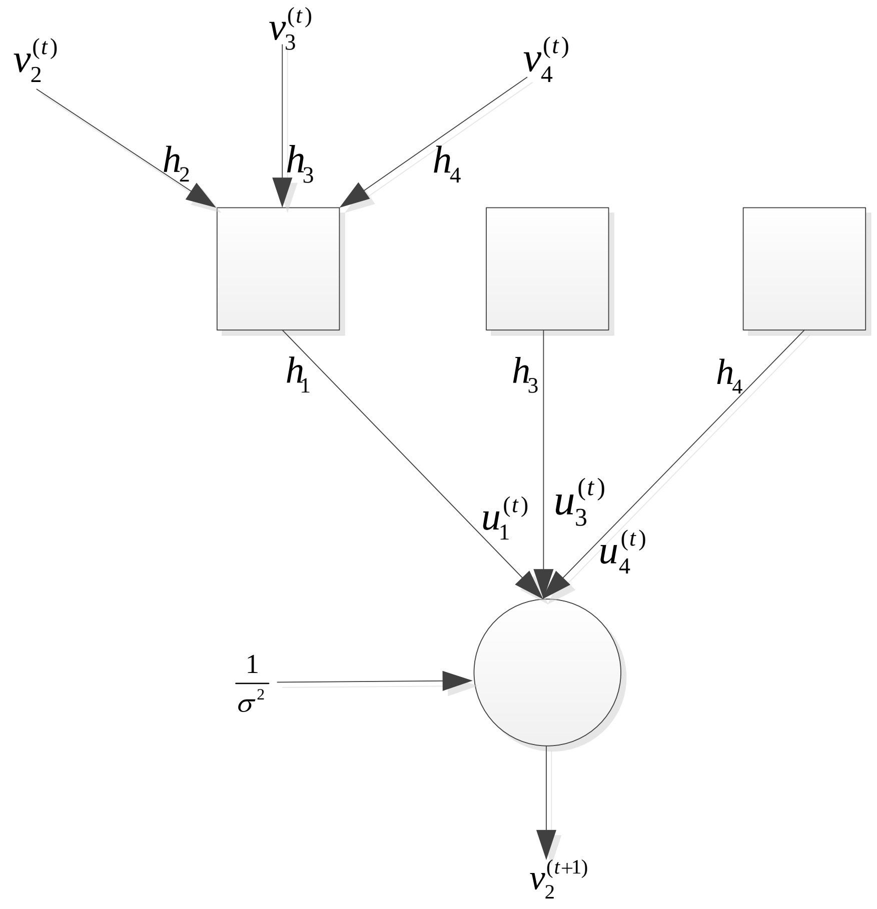

Figure 3.1: Message propagation for variance converge analysis at variance $v_2^{(t+1)}$.

3.2 Relaxed Latin Square of CLDLC

Yona and Feder [20] extended the Latin square construction for the real-valued LDLC [19] to CLDLC. Even though, the Latin square construction of CLDLC have been introduced in Section 2.5. But to make readers understand and see the differences between the Latin square and the relaxed Latin square construction of CLDLC clearly, in this section, we make the summary of Latin square construction again before explaining about the relaxed Latin square construction.

A Latin square CLDLC can be constructed by designing a check matrix $\mathbf{H}$ having constant row and column weight d. Let $\mathbf{h} = [h_1, h_2, h_3, ..., h_d]$ be a generating sequence, where $h_i \in \mathbb{C}$ and $i = 1, 2, .., d$. The non-zero entries of each row and each column are $h_1, h_2, h_3, ..., h_d$, so that each row (respectively, column) is a permutation of any other row (column), except for complex rotations, including sign changes. This construction can be considered as a regular construction.

The belief-propagation decoder described in Subsection 2.7.2 uses single-Gaussian messages as an approximation. In exact belief-propagation decoding, the messages are a mixture of an infinite number of Gaussians, and the variances of all Gaussians in a given mixture are the same, say v_t for edge t. Sommer et al. showed that a sufficient condition to achieve exponential convergence of the message variance in the BP decoder is to select the generator sequence $|h_1| \geqslant |h_2| \geqslant ... \geqslant |h_d|$ such that $\alpha < 1$, see (2.25) in Section 2.5. The convergence is exponential in the sense that $v_t \leq \alpha^i \sigma^2$, where ℓ is the iteration number and where v_t is the mixtures' variances at the variable node output for the edges associated with edge h_t, for $t = 2, 3, \ldots, d$. This was proved for real LDLC lattices [19, Theorem 1], but not for CLDLC lattices [20].

We propose a more general construction of CLDLC called *Relaxed Latin Square* which has fewer restrictions for the coefficient and weight design. The relaxed Latin square can be seen as an irregular CLDLC. Moreover, we give a new convergence condition for the relaxed Latin square construction.

Relaxed Latin Square: In row i of $\mathbf{H}$, let $h'_{1,i}, h'_{2,i}, \ldots, h'_{d_i,i}$ be the non-zero coefficients having degree d_i, where $|h'_{1,i}| > |h'_{2,i}| \geq \cdots \geq |h'_{d_i,i}|$; except for the strict inequality, this is a labeling and not a restriction on the values. Each column of $\mathbf{H}$ has one coefficient of h_1 type and one coefficient of h_2 type; in column j the remaining elements are less than h_2.

Proposition 1 *For $i = 1, \ldots, n$, let α_i be the row alpha which is represented the convergence of variance in each row:*

$$\alpha_i = \frac{1}{|h'_{i,1}|^2} \sum_{k=2}^{d_i} |h'_{k,i}|^2. \tag{3.24}$$

For the relaxed Latin square construction, if all $\alpha_i < 1$ for $i = 1, 2, \ldots, n$ then the variable node message variances $v_{t,j}$ on edges $h_{2,j}, \ldots, h_{d,j}$ satisfy for $t = 2, 3, \ldots, d$:

$$v_{t,j} \leq \alpha_{\max}^{\ell} \sigma^2, \tag{3.25}$$

where $\alpha_{\max}$ is maximum of $\alpha_1, \alpha_2, \ldots, \alpha_n$.

In other words, the variances converge exponentially fast in iterations if (3.24) is satisfied. The proof is in Appendix A. An example of a Latin square check matrix and relaxed Latin square check matrix $\mathbf{H}_{Latin}$ and $\mathbf{H}_{RelaxedLatin}$ are shown in (3.26) and (3.27) , respectively. $\mathbf{H}_{Latin}$ has constant row and column weight 3, and $\mathbf{H}_{RelaxedLatin}$ has row and column weight between 2–4.

$$\mathbf{H}_{Latin} = \begin{bmatrix} h_1 & 0 & 0 & h_2 & 0 & 0 & h_3 & 0 \\ h_2 & h_1 & 0 & 0 & 0 & 0 & 0 & h_3 \\ 0 & h_3 & h_1 & 0 & 0 & 0 & 0 & h_2 \\ 0 & 0 & h_3 & h_1 & 0 & 0 & h_2 & 0 \\ h_3 & 0 & 0 & 0 & h_1 & h_2 & 0 & 0 \\ 0 & 0 & h_2 & h_3 & 0 & h_1 & 0 & 0 \\ 0 & h_2 & 0 & 0 & h_3 & 0 & h_1 & 0 \\ 0 & 0 & 0 & 0 & h_2 & h_3 & 0 & h_1 \end{bmatrix}, \tag{3.26}$$

$$\mathbf{H}_{RelaxedLatin} = \begin{bmatrix} h_1 & 0 & 0 & h_2 & 0 & 0 & h_3 & 0 \\ h_3 & 0 & 0 & 0 & h_2 & h_1 & 0 & 0 \\ 0 & 0 & h_1 & 0 & 0 & h_2 & 0 & h_3 \\ 0 & 0 & h_2 & h_1 & 0 & 0 & 0 & 0 \\ 0 & h_1 & 0 & 0 & h_3 & 0 & 0 & h_2 \\ 0 & h_2 & 0 & h_3 & 0 & 0 & h_1 & 0 \\ 0 & 0 & h_3 & 0 & h_1 & 0 & h_2 & 0 \\ h_2 & 0 & h_4 & 0 & 0 & 0 & 0 & h_1 \end{bmatrix}. \tag{3.27}$$

3.3 Triangular Structure of Relaxed Latin Square CLDLC Based on Modified Array Low-Density Parity-Check (LDPC) Codes

In section 3.2, we introduced a new CLDLC construction called the relaxed Latin square which can be considered as an irregular CLDLC. Each row and column of the check matrix **H** allows to has the differences of coefficient and weight design. Based on this property, the triangular structure of the check matrix **H** is possible.

The triangular structure is desirable for low complexity and fast encoding process, it is also suitable for the shaping operations. The Gaussian elimination could be used to obtain the triangular structure, but this method increase the complexity of encoding process as the dimension increases, so this method is not practical for the hardware implementation. The modified array low-density parity-check (LDPC) codes [35], [36] have the triangular structure, and it can be applied to the relaxed Latin square construction of CLDLC to obtain the triangular structure.

Since the modified array codes is designed based on the quasi-cyclic LDPC (QC-LDPC) codes. First, the QC-LDPC codes are introduced, follows by the modified array LDPC codes. Finally, the triangular structure of relaxed Latin square matrix which is designed based on the modified array LDPC codes are illustrated.

3.3.1 Quasi-Cyclic LDPC Codes

LDPC codes first is discovered by Galager [37], then discovered again by Sipser et al. [38] and Mackey et al. [39]. After rediscovering, LDPC codes have created much interest since they are show remarkable performance close to the Shannon limit over additive white Gaussian noise (AWGN) channels when the code length or the parity check matrix size is large.

However, the large parity check matrix requires a significant amount of memory to store it. Quasi-cyclic LDPC (QC-LDPC) codes are one of a good candidate to solve the memory problem. This is because the parity check matrix of QC-LDPC consists of small square blocks which are circulant permutation matrices. Let $\boldsymbol{P}$ be the $L \times L$ permutation matrix given by

$$\boldsymbol{P} = \begin{bmatrix} 0 & 1 & 0 & \dots & 0 \\ 0 & 0 & 1 & \dots & 0 \\ \vdots & \vdots & \vdots & & \vdots \\ 0 & 0 & 0 & \dots & 1 \\ 1 & 0 & 0 & \dots & 0 \end{bmatrix}. \tag{3.28}$$

Note that $\boldsymbol{P}^i$ is just the circulant matrix which shifts that identity matrix $\boldsymbol{I}$ to the right by i times for any integer i $0 \leq i \geq L$.

Assume $\mathbf{H}$ is the $mL \times nL$ parity check matrix which is defined by

$$\mathbf{H} = \begin{bmatrix} \boldsymbol{P}^{a_{11}} & \boldsymbol{P}^{a_{12}} & \dots & \boldsymbol{P}^{a_{1(n-1)}} & \boldsymbol{P}^{a_{1n}} \\ \boldsymbol{P}^{a_{21}} & \boldsymbol{P}^{a_{22}} & \dots & \boldsymbol{P}^{a_{2(n-1)}} & \boldsymbol{P}^{a_{2n}} \\ \vdots & \vdots & \dots & \vdots & \vdots \\ \boldsymbol{P}^{a_{m1}} & \boldsymbol{P}^{a_{m2}} & \dots & \boldsymbol{P}^{a_{m(n-1)}} & \boldsymbol{P}^{a_{mn}} \end{bmatrix}, \tag{3.29}$$

where $a_{ij} \in \{0, 1, \dots, L-1\}$.

3.3.2 Modified Array LDPC Codes

Eleftheriou et al. [35] proposed a modified array code with the following parity check matrix for an efficient encoding:

$$\mathbf{H} = \begin{bmatrix} \boldsymbol{I} & \boldsymbol{I} & \boldsymbol{I} & \dots & \boldsymbol{I} & \dots & \boldsymbol{I} \\ 0 & \boldsymbol{I} & \boldsymbol{P} & \dots & \boldsymbol{P}^{(j-2)} & \dots & \boldsymbol{P}^{(k-2)} \\ 0 & 0 & \boldsymbol{I} & \dots & \boldsymbol{P}^{2(j-3)} & \dots & \boldsymbol{P}^{2(k-3)} \\ \vdots & \vdots & \vdots & \dots & \vdots & \dots & \vdots \\ 0 & 0 & \dots & 0 & \boldsymbol{I} & \dots & \boldsymbol{P}^{(j-1)(k-1)} \end{bmatrix}, \tag{3.30}$$

where $\mathbf{0}$ is the $L \times L$ zero or null matrix. This modified array can be considered as an irregular QC-LDPC code which is defined by three parameters; a prime number q and two integers j and k such that $q \geq k \geq j$, where j and k represent the number of non-zero elements by row and column respectively, called the row and column weight or degree of the parity check matrix.

Due to the upper triangular form of the modified array matrix, it can be encoded efficiently and also easily checked that there is no cycles of length 4 in the corresponding bipartite graph or Tanner graph.

3.3.3 Modified Array CLDLC

This subsection represents the triangular structure of relaxed Latin square matrix which is designed based on the modified array LDPC codes. We call this new construction as *Modified Array CLDLC*.

Since the relaxed Latin square matrix is square matrix, the maximum row weight and column weight equals d for the modified array CLDLC. The modified array CLDLC can be constructed as the following steps.

1) Generate the modified array code as in (3.30) with 4 cycle free based on the finite file $\mathbf{GF}(2)$.

2) The non-zero elements are replaced by $h_1, h_2, h_3, ..., h_d$ with random sign. The design of $h_1, h_2, h_3, ..., h_d$ must follow the convergence of variance in

(3.24). The array LDLC lattice inverse generator matrix H is:

$$\mathbf{H} = \begin{bmatrix} h_1\boldsymbol{I} & h_2\boldsymbol{I} & h_3\boldsymbol{I} & h_4\boldsymbol{I} & \dots & h_d\boldsymbol{I} \\ \mathbf{0} & h_1\boldsymbol{I} & h_2\boldsymbol{P} & h_3\boldsymbol{P}^2 & \dots & h_{d-1}\boldsymbol{P}^{(d-2)} \\ \mathbf{0} & \mathbf{0} & h_1\boldsymbol{I} & h_2\boldsymbol{P}^2 & \dots & h_{d-2}\boldsymbol{P}^{(d-1)} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & h_1\boldsymbol{I}^2 & \dots & h_{d-3}\boldsymbol{P}^{(d-3)} \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & h_1\boldsymbol{I} \end{bmatrix}. \qquad (3.31)$$

Even if the triangular structure is good for encoding and shaping operations, but the codeword components whose column degree is low have less protection. For example, the first column with only one non-zero element is uncoded because it only participates in a single check equation. One possible solution is to increase the protection of the less protected elements by increasing the power of the less protected elements. This can be done by scaling those elements by some factor [40].

3.4 Concluding Remarks

In this chapter, we propose a new CLDLC construction called *relaxed Latin square* for a check matrix **H**. This construction is extended from the Latin square construction. The advantage of the relaxed Latin square is that each row and column does not require to have the same coefficients and weight d; or, it can be seen as an irregular CLDLC. The proof of exponential convergence of variances under belief-propagation decoding is also provided. In addition, we also provided the triangular structure of relaxed Latin square CLDLC which is designed based on the modified array LDPC Codes. The triangular structure is good for low complexity and fast encoding process, and it is suitable for the shaping operations.

Chapter 4

Reliability-Based Decoding of Real-Valued LDLC

Before going to the reliability-based decoding of CLDLC which is the main theme of this dissertation. We first would like to introduce the reliability-based decoding of the real-valued LDLC in this chapter. Then, we extend the reliability-based decoding to CLDLC in Chapter 5. The reliability-based decoding of the real-valued LDLC can help readers to understand the complex number case easier.

LDLC codes are lattices which are constructed and decoded on principles similar to low-density parity-check (LDPC) codes. They can be decoded in large dimensions and error-free decoding is possible within 0.6 dB of the unconstrained-power channel capacity [8]. These lattice codes have sparse parity-check matrices. From this property, LDLC can be decoded using linear complexity iterative decoding based on belief propagation (BP) which is similar to LDPC codes. In iterative decoding of LDLC codes, the messages are continuous functions. When the channel is AWGN, these messages are a mixture of Gaussian functions. The number of these Gaussian functions grows quite rapidly as the number of iterations increase. For any implementation, the number of Gaussian mixtures must be reduced. In prior work [19], the Gaussian mixtures were quantized. However, this work still has high computation complexity and requires large storage. In [23] , a parametric

LDLC decoding algorithm employing Gaussian mixture reduction (GMR) to approximate the messages was presented. The messages passed between the variable and check nodes were represented by triples of three parameters, mean, variance, and mixing coefficient. All possible pairs of Gaussian on a list are searched and the closet pairs are replaced with a single Gaussian using Kullback-Leiber divergence. Further, using single Gaussians as the messages between the variable and check nodes leads to reduced memory requirements with minor performance penalty [8]. [25] proposed the three/two Gaussians parametric decoding algorithm. This work used the same parametric decoding algorithm as [8], [23], but the infinite Gaussian mixtures are approximated by three or two Gaussians at variable node. The results showed that the three or two Gaussians yield the better performance and lower complexity compared to the GMR algorithm.

We define the reliability of the check-to-variable messages for two purposes. The first one is to choose to approximate the infinite Gaussian mixtures by one or two Gaussian. The reliability of each check-to-variable message is calculated. If there is higher reliability than a decided threshold value, one Gaussian will be selected; otherwise, two Gaussians will be used. The other purpose is for the updating sequence of variable nodes of the parametric shuffled BP (SBP) decoding algorithm [28]. The parametric SBP increases the convergence speed of the BP decoder. Each variable node will average the reliability of their input messages, and the variable node which has highest reliability will be updated first.

The major advantage of this work is it reduces the number of the Gaussian mixtures at the variable node and increases the convergence speed compared to the two or three Gaussians decoding algorithm [25]. Another advantage is the proposed algorithm shows that: 1) the single Gaussian can be used, 2) each variable node input does not need to be expanded using a constant number of integers, and 3) without SBP algorithm, the maximum number of Gaussian approximation equals two is not sufficient to maintain a good probability of symbol error performance. A higher number of Gaussian in the approximation is required.

The basic theory of LDLC is described in Section 3.1. And this chapter is organized as follows. Section 4.1 describes the differences between BP and

SBP decoder. Section 4.2 describes operations over Gaussian mixtures and the moment matching approximation. Section 4.3 gives the explanation about one or two Gaussian approximation. Section 4.4 describes the reliability of the check-to-variable messages and how to select one or two Gaussian based on their reliability. In addition, the updating sequence of the variable nodes of the parametric SBP algorithm based on the input reliability is included in this section. The last section, Section 4.5 gives the numerical results.

4.1 BP and SBP Decoder

Similar to low-density parity check (LDPC) codes, the LDLC decoder uses a bipartite graph for the iterative decoding. The bipartite graph of LDLC consists of variable nodes at one side and check nodes at the other side. Each variable node corresponds to a single element of the codeword $\mathbf{x} = \mathbf{G}\mathbf{b}$. Each check node corresponds to a check equation, $\sum_k h_k x_{i_k} = \textit{integer}$, where i_k are the positions of the non-zero elements at the corresponding row of $\mathbf{H}$, h_k are the values of $\mathbf{H}$ at these locations and the *integer* is unknown. An edge connects check node i and variable node j if and only if $H_{i,j} \neq 0$. The iterative decoding of LDLC passes messages over the bipartite graph called belief-propagation (BP) decoding, in each iteration the check nodes send messages to the variable nodes along the edges of bipartite graph and vice versa. The messages between check nodes and variable nodes are real functions while the messages of LDPC codes are scalar values (e.g., log likelihood ratio (LLR) of a bit). In the standard BP algorithm, every variable-to-check messages or check-to-variable messages can be computed in parallel. (All variable nodes pass messages to the connected check nodes in the same time and vice versa.)

Shuffled belief-propagation (SBP) decoding is a sequential belief propagation algorithm which can speed up the convergence of BP decoding [28]. In the SBP algorithm, the initialization, stopping criterion test and output steps remain the same as BP algorithm. The difference between the two algorithms lies in the updating procedure of the variable nodes. While BP decoding is performed in parallel, SBP sequentially updates the variable nodes. An example is shown in

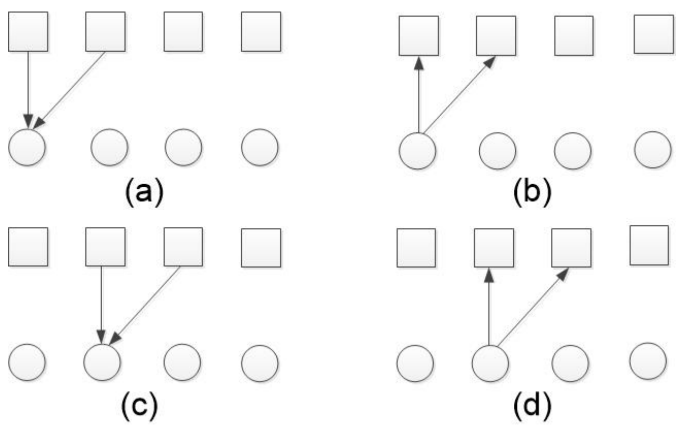

Figure 4.1: Updated sequence of shuffled BP decoding

Fig. 4.1. Initially, all channel messages are passed to the check nodes. The next step (a) is the check-to-variable message. The check nodes which are connected to the first variable node sends the messages to the first variable node. In computing the variable-to-check message (b), the first variable node returns the messages to the connected check nodes and the check nodes recalculate the messages. The remaining variable nodes have the same step as the first variable node. After the last variable node is updated, this completes 1 iteration. In this way, the latter variable nodes will have more information or reliability. However, the updating order of variable nodes can be changed. (It is not necessary that the first variable node must update first.) For decoding of LDLCs, a good sequence for updating is the key for a good performance. We will explain the updating sequence for the variable nodes in Section 4.4 .

4.2 Operations on Gaussian Mixtures

The LDLC decoding algorithm estimates PDFs: $f_{x_i|\,\mathbf{y}}(x|\mathbf{y})$, $i = 1, ..., n.)$ using a message passing algorithm over the bipartite graph. Since the variables x_i are continuous, the message are functions. A Gaussian with mean m and variance v is denoted as:

$$\mathcal{N}(z; m, v) = \frac{1}{\sqrt{2\pi v}} e^{-\frac{(z-m)^2}{2v}}. \tag{4.1}$$

For the AWGN channel, these functions are Gaussian mixtures. The message $f(z)$ is a mixture of N Gaussians

$$f(z) = \sum_{i=1}^{N} c_i \mathcal{N}(z; m_i, v_i) \tag{4.2}$$

where $c_i \geq 0$ are the mixing coefficients with $\sum_{i=1}^{N} c_i = 1$. In this way, each Gaussian mixture can be described by a set of triples, where each triple consists of three parameters: mean, variance, and mixing coefficient, $(m_1, v_1, c_1), ..., (m_N, v_N, c_N)$.

4.2.1 Product over Gaussian mixtures

Since the variable node output can be calculated from the product of input messages, the product of two Gaussian mixtures will be described. Let $f(z) = \sum_{i=1}^{N} f_i(z)$ and $g(z) = \sum_{i=1}^{N} g_i(z)$ be two Gaussian mixtures. The product of two components $f_i(z) = c_1\mathcal{N}(z; m_1, v_1)$ and $g_i(z) = c_2\mathcal{N}(z; m_2, v_2)$ is a single Gaussian $s(z) = c\mathcal{N}(z; m, v)$ with mean m, variance v and mixing coefficient c given by:

$$\frac{1}{v} = \frac{1}{v_1} + \frac{1}{v_2}, \tag{4.3}$$

$$\frac{m}{v} = \frac{m_1}{v_1} + \frac{m_2}{v_2}, \tag{4.4}$$

$$c = \frac{c_1 c_2}{\sqrt{2\pi(v_1 + v_2)}} \exp^{-\frac{(m_1 - m_2)^2}{2v_1 + 2v_2}}. \tag{4.5}$$

4.2.2 Moment Matching Approximation for Gaussian Mixtures

The operation over variable is described in this subsection. Let $f(z)$ be the Gaussian mixture. To reduce the complexity, the messages which are sent back to the check nodes or the output of variable node must be a single Gaussian $q(z) = \mathcal{N}(z; m, v)$. The moment matching approximation (MM) which minimizes the Kullback-Leiber divergence between $f(z)$ and $q(z)$ is used to approximate the Gaussian mixture $f(z)$ to a single Gaussian $q(z)$ [41]. The MM finds the single

Gaussian $q(z)$ which has the same mean m and variance v as $f(z)$. The mean m and variance v is given by:

$$m = \sum_{i=1}^{N} c_i m_i, \tag{4.6}$$

$$v = \sum_{i=1}^{N} c_i(v_i + m_i^2) - m^2. \tag{4.7}$$

This operation is denoted as:

$$q(z) = MM(f(z)). \tag{4.8}$$

4.3 One or Two Gaussian Approximation

The check-to-variable message over a set of integers $\mathcal{B}$, given by (4.9). In principle it should be computed over all integers, but it is not convenient in practice, so the number of integers must be reduced. We assume that the channel message $Y(z)$ is a single Gaussian which has mean m_a and variance $v_a = \sigma^2$. The infinite periodic Gaussians $\widetilde{R}(z)$ with period $\frac{1}{h}$ for $h \in \mathbf{h}$ has mean m_c and variance v_c. Consider the multiplication between the channel message $Y(z)$ and and the infinite periodic Gaussians $\widetilde{R}(z)$. The periodic Gaussians far from the channel message have near-zero mixing coefficients and can be safely ignored. Therefore, $\widetilde{R}(z)$ can be restricted using some finite integer set $\mathcal{B}$ which are near the channel message as,

$$R(z) = \sum_{i \in \mathcal{B}} \mathcal{N}(z; m_c + \frac{i}{h}, v_c), \tag{4.9}$$

Having a good approximation at variable node leads to a good performance in the parametric LDLC decoder. The idea is to approximate an infinite Gaussian mixture $Y(z)\widetilde{R}(z)$ with $Y(z)R(z)$, which consists of a finite number of Gaussian. [25] showed that the approximation in the tails of Gaussian function is very important. They proposed $|\mathcal{B}| = 2$ or 3 to approximate the infinite Gaussian and shows that the decoder will lose the performance if $|\mathcal{B}| = 1$ or MM approximation is applied.

In this work, we show that the finite number of integer $|\mathcal{B}| = 1$ or 2 is possible to use and gives the better performance compared to [25]. We will explain this in the next section. At the considered variable node, each incoming check-to-variable message will be expand by two Gaussians or can be used as the single Gaussian without expansion.

Here the two cases of $|\mathcal{B}| = 1$ and $|\mathcal{B}| = 2$ Gaussians near m_a are considered. Let the single Gaussian set be $\mathcal{B} = \{b_0\}$, and the two Gaussians set be $\mathcal{B} = \{b_1, b_2\}$ where $b_2 = b_1 + 1$.

For the single Gaussian, b_0 is

$$b_0 = \lceil -h(m_c - m_a) \rfloor. \tag{4.10}$$

The resulting Gaussian mixture is:

$$R(z) = \mathcal{N}(z; m_c + \frac{b_0}{h}, v_c), \tag{4.11}$$

For the two-Gaussian set, two integer are selected. The first one is less than a noninterger estimate, and the other one is more than a noninteger estimate. b_1 can be found as:

$$b_1 = \lfloor -h(m_c - m_a) \rfloor. \tag{4.12}$$

The resulting Gaussian mixture is:

$$R(z) = \mathcal{N}(z; m_c + \frac{b_1}{h}, v_c) + \mathcal{N}(z; m_c + \frac{b_2}{h}, v_c). \tag{4.13}$$

4.4 Reliability-Based Parametric LDLC Decoding

In this section, we present the reliability-based parametric SBP decoding algorithm. The number of Gaussians for the periodic expansion and updating order of SBP both use the reliability of the incoming check-to-variable messages. Therefore, the reliability calculation is described first, followed by the parametric SBP decoding algorithm.

4.4.1 Reliability of Check-to-Variable Messages

In this part, we describe the reliability of the incoming check-to-variable message. The reliability is based on the likelihood of the mean and the nearest integer of the check-to-variable message. The proof is similar to CLDLC, it is given in Section 5.2.1. We use (4.10) to calculate the integer a, $a \in \mathbb{Z}$ nearest to $b \in \mathbb{R}$. b and a represent the mean and its integer approximation, respectively. When the channel is noiseless, a will be equal to b. Therefore, the difference between a and b can represent the intensity of the Gaussian noise or the reliability of the message. If there is a large difference between a and b, it means that the message has low reliability or high intensity of Gaussian noise. The reliability of each message can be calculated as follows,

$$\mathit{Reliability} = \frac{1}{|a-b|} = \frac{1}{|\lceil -h_d(m_d - m_a)\rfloor + (h_d(m_d - m_a))|}. \tag{4.14}$$

4.4.2 Reliability-Based Parametric SBP Decoder

The proposed decoding algorithm is presented here. The SBP algorithm decodes over the bipartite graph that was described in Section II-C. There are $n \cdot d$ variable-to-check messages $q_k(z)$, and $n \cdot d$ check-to-variable messages $\widetilde{R}_k(z), k = 1, 2, ...n \cdot d$. Since the periodic expansion step takes place at the variable node, the messages are the mixtures of Gaussians. The variable node computes the product of the incoming messages from (4.3)-(4.5). To reduce the memory requirement between variable and check nodes, the outputs of variable node which are Gaussian mixtures will be approximated by a single Gaussian using MM approximation from (4.6)-(4.8), before sending the messages to check nodes. Single Gaussians are represented by its mean and variance called parametric decoding.

The parametric iterative decoding algorithm is as follows:

- *Initialization*: According to (3.3), the channel is AWGN, the initial variable-to-check message is

$$y_i(z) = \mathcal{N}(z; y_i, \sigma^2), \tag{4.15}$$

for $i = 1, 2, ..., n$. All received messages are sent to the check nodes in initial step.

- *Check-to-variable message*: The incoming messages of check nodes are $k = 1, ..., d-1$ single Gaussians and output d. The corresponding non-zero coefficients from $\mathbf{H}$ are $h_1, ..., h_d$. The output $\widetilde{R}_d(z)$ is:

  $$\widetilde{R}_d(z) = \mathcal{N}(z; m_d, v_d), \tag{4.16}$$

  where,

  $$m_d = -\frac{\sum_{k=1}^{d-1} h_k m_k}{h_d}, \text{ and } v_d = -\frac{\sum_{k=1}^{d-1} h_k^2 v_k}{h_d^2}. \tag{4.17}$$

- *Variable node update sequence*: We add this step to sequence the variable nodes for updating. $n \cdot d$ outgoing messages from the check nodes will be calculate $Reliability_k$, $k = 1, 2, ...n \cdot d$ by (4.14). Then, find the sequence for updating variable node. Consider the variable node $v_i, i = 1, 2, ..., n$ has the inputs from the check nodes $k = 1, ..., d$. The average *Reliability* of v_i can be calculate by,

  $$Avg\_Reliability\_v_i = \frac{\sum_{k=1}^{d} Reliability_k}{d}, \tag{4.18}$$

  $Avg\_Reliability\_v_i$ will be sequenced from high to low. The updated order of the variable nodes start from the variable node which has the highest $Avg\_Reliability\_v$ to the variable node which has the lowest $Avg\_Reliability\_v$. The SBP decoding algorithm was described in Section II-C. Note that we can use the average reliability to find the updated sequence of SBP decoding algorithm because the reliability is always a positive value, so it is not necessary to use the higher complexity algorithm for the hardware implementation such as the root mean square (RMS) to find the updated sequence of SBP.

- *Variable-to-check message*: The messages coming from the check node are single Gaussians $\mathcal{N}(z; m_d, v_d)$. Then the messages are expanded to a periodic function with period $1/|h_d|$, if $b \in \mathcal{B}$. The periodic Gaussian is:

$$R_d(z) = \sum_{b \in \mathcal{B}} \mathcal{N}(z; m_d(b), v_d), \tag{4.19}$$

where the mean m of each Gaussian is

$$m_d(b) = m_d + \frac{b}{h_d}. \tag{4.20}$$

The set $\mathcal{B}$ represents a subset of the integer. The input messages of the variable node will select $|\mathcal{B}| = 1$ if $Reliability_k > Threshold$, and if $Reliability_k < Threshold$ then $|\mathcal{B}| = 2$ is used. (We set the threshold value for choosing the number of expansion.) The message $f_d(z)$ sent back to the check node is a single Gaussian approximated by:

$$f_d(z) = y_i(z) \prod_{k=1}^{d-1} R_k(z), \tag{4.21}$$

$$q_d(z) = MM(f_d(z)), \tag{4.22}$$

- *Final decision*: At the last iteration the product without omitting any message is performed:

$$q_i^{final}(z) = \mathcal{N}(z; y_i, \sigma^2) \prod_{k=1}^{d} R_k(z). \tag{4.23}$$

And the the lattice point estimate $\hat{\mathbf{x}}$ and the integer information estimate $\hat{\mathbf{b}}$ are

$$\hat{x}_i = \arg\max_z q_i^{final}(z), \text{ and } \hat{\mathbf{b}} = \lfloor \mathbf{H}\hat{\mathbf{x}} \rceil. \tag{4.24}$$

4.5 Numerical Results

The reliability-based parametric LDLC decoding was evaluated numerically on the unconstrained input power AWGN channel. We compared three algorithms:

1. two or three Gaussians decoding [25], 2. the proposed reliability-based decoding *without* SBP and 3. the proposed reliability-based decoding *with* SBP. Lattices with dimension $n = 100$ and 1000 were used, and the matrix H had row and column weights d of 5 and 7, respectively. The inverse generator matrix was created with the generator sequence $\mathbf{h} = \{1, \frac{1}{\sqrt{d}}, ..., \frac{1}{\sqrt{d}}\}$, and the matrix was normalized to have $|\det(\mathbf{G})| = 1$.

To find the threshold, we start with a small-valued of the threshold to obtain, then test the decoder symbol error rate (SER). Afterward, gradually increase the threshold and test the SER again. This process is repeated until the decoder SER has noticeable degradation, and that point will be set as the threshold. (when $Threshold$ equals 0, all check-to-variable messages are expanded using $|\mathcal{B}| = 2$) we select $Threshold = 0.2$ as a compromise between performance and complexity. Fig. 4.2 shows the average number of Gaussian that we use in the periodic expansion versus $Threshold$. The average number of Gaussians over the VNR range is 1.4 when we select $Threshold = 0.2$. The complexity of the proposed algorithm depends on the lattice dimension n,the number of iterations t, number of the Gauusians in the periodic expansion, and the degree of the inverse generator matrix d. Assuming that the selection of one and two Gaussians is independent, the complexity of the proposed decoder is $\mathcal{O}(n \cdot t \cdot 1.4^{d-1})$. For comparison, the complexity of the two or three Gaussians are $\mathcal{O}(n \cdot t \cdot 2^{d-1})$ and $\mathcal{O}(n \cdot t \cdot 3^{d-1})$.

In Fig.4.3, the average number of iterations required for decoder convergence is shown. The number of iterations recedes when the VNR increases and the proposed algorithm lowers the number of iterations required for convergence.

Fig.4.4 shows the probability of symbol error for the proposed decoding algorithms vs. VNR. The proposed reliability-based decoding with SBP decoding outperforms the two or three Gaussians decoder. When the probability of symbol error equals 10^{-4} and $n = 100$ and 1000, the proposed algorithm gains 0.25 dB and 0.2 dB, respectively. If we compare to the proposed reliability-based decoding without SBP decoding, the reliability-based SBP decoding gains 0.5 dB and 0.3 dB for $n = 100$ and 1000, respectively.

4.6 Concluding Remarks

This chapter proposes reliability-based parametric decoding of low-density lattice codes (LDLC). We define the reliability of the check-to-variable messages for two purposes. The first one is to choose to approximate the infinite Gaussian mixtures by one or two Gaussian. The reliability of each check-to-variable message is calculated. If there is higher reliability than a decided threshold value, one Gaussian will be selected; otherwise two Gaussians will be used. The other purpose is for the updating sequence of variable nodes of the parametric shuffled BP (SBP) decoding algorithm. The parametric SBP increases the convergence speed. The updating sequence of SBP follows the order of reliability of the check-to-variable messages from high to low.

The numerical results show that the proposed algorithm gives superior performance and lower complexity compared to two or three Gaussian decoding algorithm. At a probability of symbol error equal 10^{-4} and $n = 100$ and 1000, the proposed algorithm gains 0.25 and 0.2 dB, respectively. Moreover, the complexity of the proposed decoder is $\mathcal{O}(n \cdot t \cdot 1.4^{d-1})$ which is lower than the two or three Gaussians [25] which have the complexity $\mathcal{O}(n \cdot t \cdot 2^{d-1})$ and $\mathcal{O}(n \cdot t \cdot 3^{d-1})$.

From the reliability-based LDLC decoding, we know that:

1) the single Gaussian can be used,
2) each variable node input does not need to be expanded using a constant number of integers, and
3) without SBP algorithm, the maximum number of Gaussian approximation equals two is not sufficient to maintain a good probability of symbol error performance. A higher number of Gaussian in the approximation is required.

From this summary, we extended the reliability-based decoding to the complex number case in the next chapter.

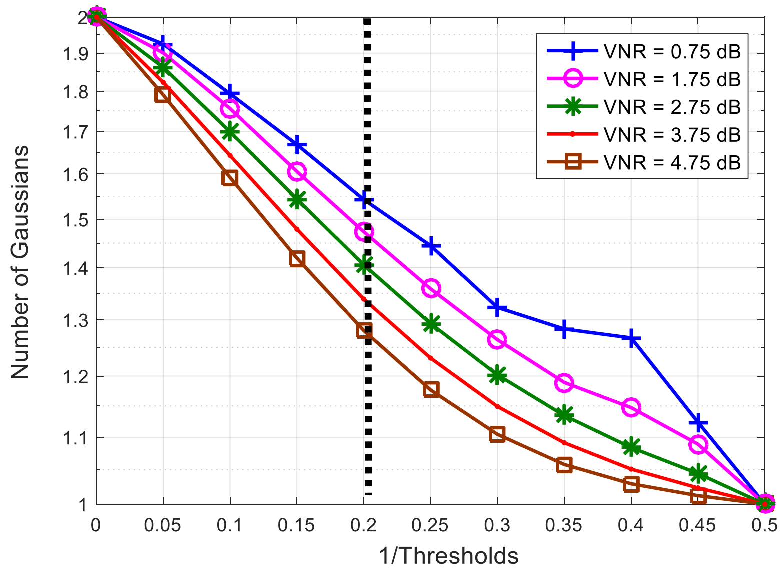

Figure 4.2: Threshold vs number of Gaussian for lattice dimension $n = 100$ and degree $d = 5$

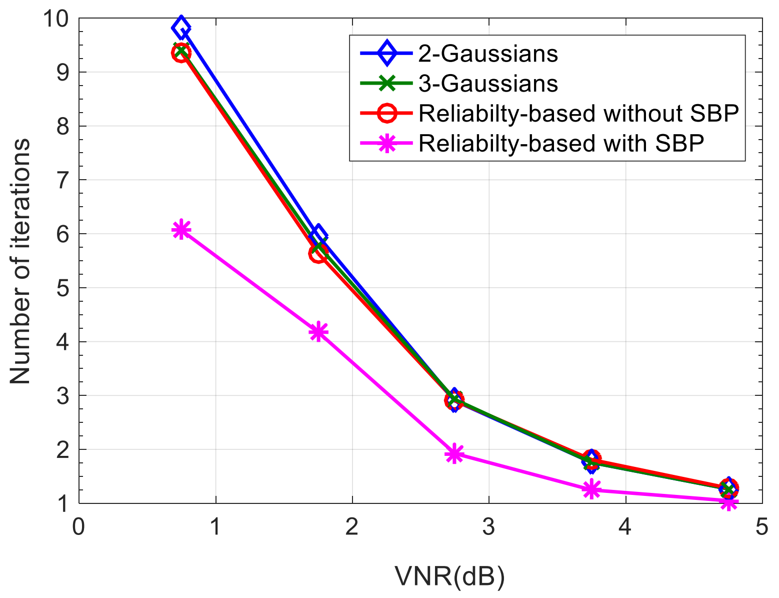

Figure 4.3: Average number of iterations required for decoder convergence in term of VNR for lattice dimension $n = 100$ and degree $d = 5$

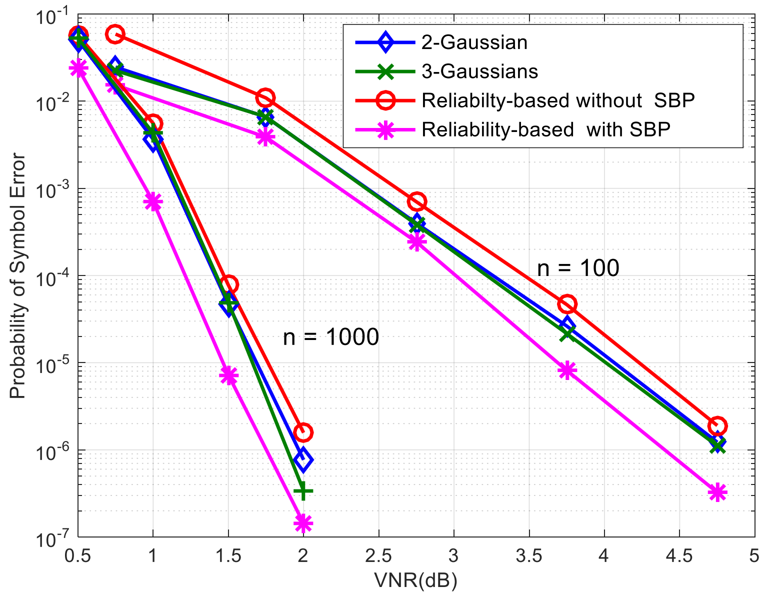

Figure 4.4: The performance comparison in term of VNR vs probability of symbol error

Chapter 5

Reliability-Based Decoding of CLDLC Using Gaussian and Eisenstein Integers

As we introduced in Section 2.7.2, BP decoding of CLDLC confronts the infinite Gaussian mixture issue in iterative decoding, and this infinite Gaussian mixture must be approximated for the decoder implementation. This chapter describes an approximation of an infinite Gaussian mixture using a finite Gaussian mixture; this is done for both CLDLC based on Gaussian integers and Eisenstein integers. This approximation is used at the variable node of the belief-propagation decoding algorithm for CLDLC lattices in Section 2.7.2.

The CLDLC decoding algorithm presented by Yona and Feder [20] employs Gaussian mixture reduction (GMR) algorithm. The decoder sorts the list of Gaussian mixtures and the Gaussians which satisfy a given condition will be grouped. Each incoming message of the variable node is approximated by 9 complex Gaussian functions. The GMR algorithm is be performed at every multiplication, at each variable node, on each iteration. Due to the large number of Gaussian functions, the complicated GMR algorithm, and the large number of its uses, this algorithm may not be suitable for hardware implementation.

To approximate the infinite Gaussian mixture to a finite number, we propose reliability-based CLDLC decoding. In reliability-based decoding, the number of Gaussians in the check-to-variable message is adaptively selected depending upon the reliability. We define a likelihood-based reliability function and use it to determine the number of complex Gaussians at the variable node. This leads to an optimized number of Gaussians in each check-to-variable message. For example, when the message has high reliability a single Gaussian is selected, otherwise a greater number of Gaussian is selected for the message which has lower reliability. To determine that the check-to-variable message has high or low reliability, we define a threshold function which can be found using the Kullback-Leibler (KL) divergence between the approximation and the true distribution, but explicitly computing the divergence is inefficient. Instead, we form an upper bound on this KL divergence, and linear regression is used to efficiently estimate this threshold.

In addition, we proposed a CLDLC construction using Eisenstein integers (EI) in this chapter. The advantage of Eisenstein integers over Gaussian integers is that Eisenstein integers give a more accurate approximation for a fixed number of Gaussians in the mixture. In the approximation, Gaussians with means closest to the channel value, corresponding to either Gaussian integers or Eisenstein integers, are selected because these have the highest likelihood. If a fixed number of Gaussians is considered, for example three Gaussians, then three Eisenstein integers will together give a higher likelihood than three GI because Eisenstein integers have hexagonal packing, which is tighter than the cubic packing of GI (the hexagonal packing is the tightest in two dimensions). Due to the tighter packing, the distance to the channel value will be lower on average.

5.1 Gaussian Approximation for CLDLC Decoding

In CLDLC, the check-to-variable message is an infinite complex Gaussian mixture, which for GI is given by

$$R(\mathbf{z}) = \sum_{j \in \mathbb{Z}[i]} \mathcal{N}\left(\mathbf{z}; \mathbf{m}_c + \frac{j}{h}, \mathbf{V}_c\right), \tag{5.1}$$

and for EI, the sum is over $\mathbb{Z}[\omega]$. The infinite periodic Gaussians $R(\mathbf{z})$ with period $1/h$ has mean $\mathbf{m}_c$ and variance $\mathbf{V}_c$. The channel message $Y(\mathbf{z})$ is a single complex Gaussian function with mean $\mathbf{m}_a$ and variance $\mathbf{V}_a = \sigma^2\mathbf{I}_2$. The exact product of $Y(\mathbf{z})$ and $R(\mathbf{z})$ at the variable node is also an infinite complex Gaussian mixture. This infinite Gaussian mixture $Y(\mathbf{z})R(\mathbf{z})$ must be reduced to a finite number of Gaussians $Y(\mathbf{z})\widetilde{R}(\mathbf{z})$ in practice. $\widetilde{R}(\mathbf{z})$ is the summation in (5.1) restricted to some finite subset $\mathcal{B}$, given by

$$\widetilde{R}(\mathbf{z}) = \sum_{j \in \mathcal{B}} \mathcal{N}\left(\mathbf{z}; \mathbf{m}_c + \frac{j}{h}, \mathbf{V}_c\right), \tag{5.2}$$

where $\mathcal{B} \subset \mathbb{Z}[i]$ in the GI case and $\mathcal{B} \subset \mathbb{Z}[\omega]$ in the EI case.

Choosing a finite subset of integers $\mathcal{B}$ of $\widetilde{R}(\mathbf{z})$ which is sufficient to represent the infinite Gaussian mixtures $R(\mathbf{z})$ is the key for an accurate approximation. The periodic Gaussians far from the channel message have near-zero mixing coefficients and can be safely ignored. Therefore, $R(\mathbf{z})$ can be restricted using some finite integer set $\mathcal{B}$ which are near the channel message $Y(\mathbf{z})$. How to select $\mathcal{B}$ is the subject of the next two subsections.

5.1.1 Gaussian Integer Approximation

We present an approximation of $\widetilde{R}(z)$ based on a subset of Gaussian integers $\mathcal{B} \subset \mathbb{Z}[i]$ at the variable node in this subsection. The three cases of $|\mathcal{B}| = 1, |\mathcal{B}|$ $= 2$ and $|\mathcal{B}| = 4$ Gaussians near $Y(\mathbf{z})$ are considered. Each case of $|\mathcal{B}|$ is used to approximate check-to-variable messages which have different reliability. If a check-to-variable message has high reliability, $|\mathcal{B}| = 1$ is sufficient to represent the infinite Gaussian mixtures. But if the check-to-variable message has intermediate or low reliability $|\mathcal{B}| = 2$ or 4 will be applied, respectively.

For the high-reliability case, let the single Gaussian set be $\mathcal{B} = \{\mathbf{b}_0\}$, where $\mathbf{b}_0$ is

$$\mathbf{b}_0 = \lceil -[\,h]\,(\mathbf{m}_c - \mathbf{m}_a) \rfloor, \tag{5.3}$$

and $[\,h] = \begin{bmatrix} Re\{h\} & -Im\{h\} \\ Im\{h\} & Re\{h\} \end{bmatrix}$. The resulting Gaussian mixture is

$$\widetilde{R}(\mathbf{z}) = \mathcal{N}(\mathbf{z}; \mathbf{m}_c + [h]^{-1}\mathbf{b}_0, \mathbf{V}_c). \tag{5.4}$$

For intermediate reliability, two Gaussian integers are selected, $\mathcal{B} = \{\mathbf{b}_0, \mathbf{b}_1\}$. $\mathbf{b}_0$ is found from (5.3). Define $\widetilde{\mathbf{b}}_0 = -[\,h]\,(\mathbf{m}_c - \mathbf{m}_a)$ as the soft value of $\mathbf{b}_0$. $\mathbf{b}_1$ is found from the minimum Euclidean distance between $\widetilde{\mathbf{b}}_0$ and the eight integers nearest $\mathbf{b}_0$. Define $\zeta_0 = \mathbf{b}_0 + \{[-1,-1]^T, [-1,0]^T, [-1,1]^T, [0,-1]^T, [0,1]^T, [1,-1]^T, [1,0]^T, [1,1]^T\}$ so that $\mathbf{b}_1$ is:

$$\mathbf{b}_1 = \min_{\mathbf{x} \in \zeta_0} ||\mathbf{x} - \widetilde{\mathbf{b}}_0||^2. \tag{5.5}$$

The resulting Gaussian mixture is

$$\widetilde{R}(\mathbf{z}) = \mathcal{N}(\mathbf{z}; \mathbf{m}_c + [h]^{-1}\mathbf{b}_0, \mathbf{V}_c) + \mathcal{N}(\mathbf{z}; \mathbf{m}_c + [h]^{-1}\mathbf{b}_1, \mathbf{V}_c). \tag{5.6}$$

For low reliability, four Gaussian integers are selected, $\mathcal{B} = \{\mathbf{b}_0, \mathbf{b}_1, \mathbf{b}_2, \mathbf{b}_3\}$. $\mathbf{b}_0$ and $\mathbf{b}_1$ can be found from equation (5.3) and (5.5). $\mathbf{b}_2$ can be computed as

$$\mathbf{b}_2 = \min_{\mathbf{x} \in \zeta_1} ||\mathbf{x} - \widetilde{\mathbf{b}}_0||^2, \tag{5.7}$$

where $\zeta_1 = \zeta_0 - \mathbf{b}_1$, and $-$ denotes set subtraction. $\mathbf{b}_3$ can be found as,

$$\mathbf{b}_3 = \min_{\mathbf{x} \in \zeta_2} ||\mathbf{x} - \widetilde{\mathbf{b}}_0||^2, \tag{5.8}$$

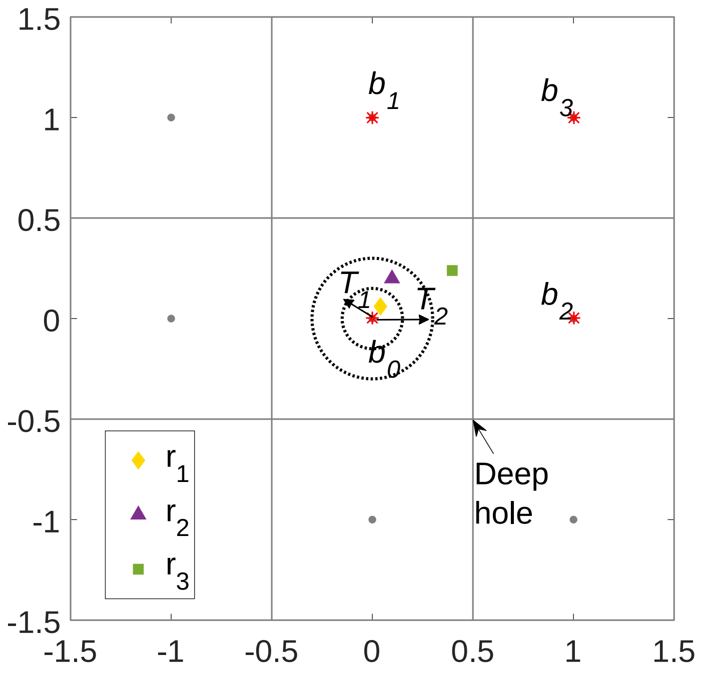

Figure 5.1: Gaussian approximation based on reliability of check-to-variable message for Gaussian integers for three example check-to-variable messages r_1, r_2 and r_3.

where $\zeta_2 = \zeta_1 - \mathbf{b}_2$. The resulting Gaussian mixture is

$$\begin{aligned}\widetilde{R}(\mathbf{z}) = \mathcal{N}(\mathbf{z};\mathbf{m}_c + [h]^{-1}\mathbf{b}_0, \mathbf{V}_c) + \mathcal{N}(\mathbf{z};\mathbf{m}_c + [h]^{-1}\mathbf{b}_1, \mathbf{V}_c) +\\ \mathcal{N}(\mathbf{z};\mathbf{m}_c + [h]^{-1}\mathbf{b}_2, \mathbf{V}_c) + \mathcal{N}(\mathbf{z};\mathbf{m}_c + [h]^{-1}\mathbf{b}_3, \mathbf{V}_c).\end{aligned} \tag{5.9}$$

For Gaussian integers, example $\mathbf{b}_0, \mathbf{b}_1, \mathbf{b}_2$ and $\mathbf{b}_3$ are shown in Fig. 5.1. In summary, $\mathbf{b}_0$ is the closest to the soft value $\widetilde{\mathbf{b}}_0$, $\mathbf{b}_1$ is second closest, etc. In the figures, T_1 and T_2 are thresholds to select between high, intermediate and low reliability, and will be described in Section 5.2.

5.1.2 Eisenstein Integer Approximation

We propose an approximation of $\widetilde{R}(z)$ based on a subset of Eisenstein integers $\mathcal{B} \subset \mathbb{Z}[\omega]$ in this subsection. The approximation is made at the variable node, as in case of Gaussian integers. Three cases of $|\mathcal{B}| = 1, |\mathcal{B}| = 2$ and $|\mathcal{B}| = 3$ Gaussians near $Y(\mathbf{z})$ are considered. For check-to-variable messages with high reliability, a single Gaussian $\mathcal{B} = \{\mathbf{b}_0\}$ is selected, which can be found using coset decoding of the hexagonal lattice [29] as follows:

1. Calculate
$$\widetilde{\mathbf{b}}_0 = -[h]\,(\mathbf{m}_c - \mathbf{m}_a) \tag{5.10}$$

2. Let Λ_1 be a scaled integer lattice with generator matrix $\mathbf{G} = \begin{bmatrix} 1 & 0 \\ 0 & \sqrt{3} \end{bmatrix}$. Λ_1 is the direct sum $\mathbb{Z} \oplus \sqrt{3}\mathbb{Z}$— this allows us to quantize in each dimension independently. Quantize $\widetilde{\mathbf{b}}_0$ to the nearest point in Λ_1:
$$\mathbf{z}_0 = Q_{\Lambda_1}(\widetilde{\mathbf{b}}_0) = [\lceil \widetilde{b}_{0,Re} \rfloor \;\; \sqrt{3} \cdot \lceil \widetilde{b}_{0,Im}/\sqrt{3} \rfloor]^T, \tag{5.11}$$

3. Find the point $\mathbf{z}_1$ in the coset $\Lambda_1 + \mathbf{s}$ closest to $\widetilde{\mathbf{b}}_0$ as:
$$\mathbf{z}_1 = Q_{\Lambda_1}(\widetilde{\mathbf{b}}_0 - \mathbf{s}) + \mathbf{s}, \tag{5.12}$$
where $\mathbf{s} = [\frac{1}{2} \; \frac{\sqrt{3}}{2}]^T$.

4. The point $\mathbf{b}_0$ closest to $\widetilde{\mathbf{b}}_0$ is:
$$\mathbf{b}_0 = \begin{cases} \mathbf{z}_0, & \text{if } ||\mathbf{z}_0 - \widetilde{\mathbf{b}}_0||^2 \leqslant ||\mathbf{z}_1 - \widetilde{\mathbf{b}}_0||^2 \\ \mathbf{z}_1, & \text{if } ||\mathbf{z}_0 - \widetilde{\mathbf{b}}_0||^2 > ||\mathbf{z}_1 - \widetilde{\mathbf{b}}_0||^2 \end{cases}. \tag{5.13}$$

The resulting Gaussian mixture is $\widetilde{R}(\mathbf{z}) = \sum_{j \in \{\mathbf{b}_0\}} \mathcal{N}(\mathbf{z}; \mathbf{m}_c + [h]^{-1} j, \mathbf{V}_c)$.

$$\widetilde{R}(\mathbf{z}) = \mathcal{N}(\mathbf{z}; \mathbf{m}_c + [h]^{-1}\mathbf{b}_0, \mathbf{V}_c). \tag{5.14}$$

For intermediate reliability, two Eisenstein integers are selected, $\mathcal{B} = \{\mathbf{b}_0, \mathbf{b}_1\}$; $\mathbf{b}_0$ is from (5.10)–(5.13) and $\mathbf{b}_1$ is found from the minimum Euclidean distance between $\widetilde{\mathbf{b}}_0$ and the six integers nearest $\mathbf{b}_0$. Define as ε_0, where $\widetilde{\mathbf{b}}_0 = -[\,h\,]\,(\mathbf{m}_c - \mathbf{m}_a)$ and $\varepsilon_0 = \mathbf{b}_0 + \{[-1,0], [1,0], [-\frac{1}{2}, \frac{\sqrt{3}}{2}], [\frac{1}{2}, \frac{\sqrt{3}}{2}], [-\frac{1}{2}, -\frac{\sqrt{3}}{2}], [\frac{1}{2}, -\frac{\sqrt{3}}{2}]\}$ so that $\mathbf{b}_1$ is:

$$\mathbf{b}_1 = \min_{\mathbf{x} \in \varepsilon_0} ||\mathbf{x} - \widetilde{\mathbf{b}}_0||^2. \tag{5.15}$$

The resulting Gaussian mixture is

$$\widetilde{R}(\mathbf{z}) = \mathcal{N}(\mathbf{z}; \mathbf{m}_c + [h]^{-1}\mathbf{b}_0, \mathbf{V}_c) + \mathcal{N}(\mathbf{z}; \mathbf{m}_c + [h]^{-1}\mathbf{b}_1, \mathbf{V}_c). \tag{5.16}$$

For low reliability, three Eisenstein integers are selected, $\mathcal{B} = \{\mathbf{b}_0, \mathbf{b}_1, \mathbf{b}_2\}$, where $\mathbf{b}_0$ and $\mathbf{b}_1$ can be found as above, and $\mathbf{b}_2$ can be computed as

$$\mathbf{b}_2 = \min_{\mathbf{x} \in \varepsilon_1} ||\mathbf{x} - \widetilde{\mathbf{b}}_0||^2, \tag{5.17}$$

where $\varepsilon_1 = \varepsilon_0 - \mathbf{b}_1$. The resulting Gaussian mixture is

$$\begin{aligned}\widetilde{R}(\mathbf{z}) = \mathcal{N}(\mathbf{z}; \mathbf{m}_c + [h]^{-1}\mathbf{b}_0, \mathbf{V}_c) + \mathcal{N}(\mathbf{z}; \mathbf{m}_c + [h]^{-1}\mathbf{b}_1, \mathbf{V}_c) + \\ \mathcal{N}(\mathbf{z}; \mathbf{m}_c + [h]^{-1}\mathbf{b}_2, \mathbf{V}_c).\end{aligned} \tag{5.18}$$

For Eisenstein integers, example $\mathbf{b}_0, \mathbf{b}_1$, and $\mathbf{b}_2$ of Eisenstein integers are shown in Fig. 5.2. In the figures, T_1 and T_2 are thresholds to select between high, intermediate and low reliability, and will be described in Section 5.2.

The maximum value for $|\mathcal{B}|$ of 4 and 3 for Gaussian and Eisenstein integers is related to the number of lattice points connected to a deep hole. The deep hole is the furthest point from the lattice point on the Voronoi cell and are shown in Fig. 5.1 and 5.2. If the check-to-variable message r is at the deep hole in the worst case, the number of high-likelihood integers $|\mathcal{B}|$ near r is 4 and 3 for Gaussian and Eisenstein integers, respectively.

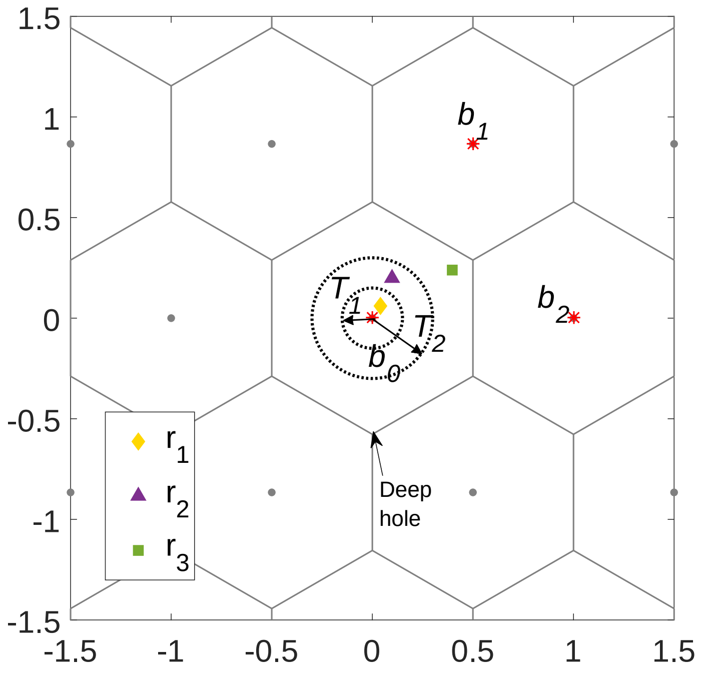

Figure 5.2: Gaussian approximation based on reliability of check-to-variable message for Eisenstein integers for three example check-to-variable messages r_1, r_2 and r_3.

5.2 Reliability-Based Implementation at Variable Nodes

This section describes reliability-based selection of Gaussians to be used at the variable node in Section 2.7.2. From Section 5.1, the infinite complex Gaussian mixtures can be approximated by 1, 2 or 4 Gaussians in the GI case, or 1, 2 or 3 Gaussians in the EI case. This section addresses how to select the number of Gaussians. We define the reliability, $1/\rho$ of the check-to-variable messages which is used to choose the number of Gaussians for each incoming message at the variable node. Two thresholds $T_1 < T_2$ will be set at the variable node. If the reliability is high, that is ρ satisfies $\rho \leq T_1$, then one Gaussian will be selected. If the reliability is intermediate, that is, ρ satisfies $T_1 < \rho \leq T_2$, then two Gaussian will be selected. Otherwise, the reliability is low and four Gaussians or three Gaussians will be selected for Gaussian integers or Eisenstein integers, respectively. The thresholds T_1, T_2 are selected such that the Kullback-Leiber divergence of the resulting approximation is no greater than a target value.

5.2.1 Reliability Function

The reliability is based on the likelihood of the soft check-to-variable message $\widetilde{\mathbf{b}}_0 = -[\,h]\,(\mathbf{m}_c - \mathbf{m}_a)$ given its nearest integer approximation $\mathbf{b}_0$, as given by (5.3). When the channel is noiseless, $\widetilde{\mathbf{b}}_0$ will be equal to $\mathbf{b}_0$, and the highest likelihood value is obtained. The likelihood of $\widetilde{\mathbf{b}}_0$ given $\mathbf{b}_0$ can represent the inverse magnitude of the noise, or the reliability of the message. If there is a large difference between $\widetilde{\mathbf{b}}_0$ and $\mathbf{b}_0$, it means that the message has low reliability or high noise magnitude. The likelihood is:

$$\Pr(\widetilde{\mathbf{b}}_0|\mathbf{b}_0) = \frac{1}{2\pi\sqrt{|\mathbf{V}|}} e^{-\frac{1}{2}(\widetilde{\mathbf{b}}_0-\mathbf{b}_0)^T\mathbf{V}^{-1}(\widetilde{\mathbf{b}}_0-\mathbf{b}_0)}. \tag{5.19}$$

For any given Gaussian with variance $\mathbf{V}$, $\widetilde{\mathbf{b}}_0$ and $\mathbf{b}_0$ are the only variables in (5.19), the other parameters are constant. Therefore, the reliability $1/\rho$ depends on $\widetilde{\mathbf{b}}_0$ and $\mathbf{b}_0$ only. We define ρ as the relevant part of the likelihood:

$$\rho = |\widetilde{\mathbf{b}}_0 - \mathbf{b}_0| = |(-[h]\,(\mathbf{m}_c - \mathbf{m}_a)) - \lceil -[h]\,(\mathbf{m}_c - \mathbf{m}_a) \rfloor|, \tag{5.20}$$

which satisfies $0 \leq \rho \leq 1$. A smaller value of the magnitude of the Gaussian noise means higher reliability, so the reliability is written as $1/\rho$. Fig. 5.1-(a) and (b) show three example check-to-variable messages r_1, r_2 and r_3 which have reliabilities ρ_1, ρ_2 and ρ_3, respectively; both Gaussian integers and Eisenstein integers are shown. $\rho_1 < \rho_2 < \rho_3$ represent low, intermediate and high intensity of the Gaussian noise, respectively. The first message with the smallest ρ_1 has the highest reliability $1/\rho_1$, the second and third message with ρ_2 and ρ_3 have intermediate and low reliability, $1/\rho_2$ and $1/\rho_3$, respectively.

5.2.2 Threshold and Bound on Kullback-Leibler Divergence

This subsection describes how to obtain the thresholds T_1, T_2, using the Kullback-Leibler (KL) divergence. In particular, the smallest number of Gaussians are chosen such that an estimate of the resulting KL divergence does not exceed a target value. The KL divergence between two Gaussian mixtures does not have a closed-form solution in general. Our approach is to form an upper bound on the KL divergence and use this to upper bound the thresholds, T_1, T_2. Then, linear regression is used as an approximation.

The KL divergence represents the similarity between two probability density functions (pdfs) [42, 43]. If the two pdfs are the same, the divergence is zero. Our main interest is the KL divergence between the infinite Gaussian mixture $Y(\mathbf{z})R(\mathbf{z})$ and the finite Gaussian mixture $Y(\mathbf{z})\widetilde{R}(\mathbf{z})$:

$$D(Y(\mathbf{z})R(\mathbf{z})||Y(\mathbf{z})\widetilde{R}(\mathbf{z})) = \int_{\mathbb{C}} Y(\mathbf{z})R(\mathbf{z}) \log \frac{Y(\mathbf{z})R(\mathbf{z})}{Y(\mathbf{z})\widetilde{R}(\mathbf{z})} d\mathbf{z}, \tag{5.21}$$

where $Y(\mathbf{z})$ is channel message with mean m_a and variance $\mathbf{V}_a$. $R(\mathbf{z})$ and $\widetilde{R}(\mathbf{z})$ are given in (5.1) and (5.2). The KL divergence of the Gaussian approximation depends on 5 parameters: h, m_c, m_a, $\mathbf{V}_c$ and $\mathbf{V}_a$. If h, $\mathbf{V}_c$ and $\mathbf{V}_a$ are fixed, the KL divergence depends on $m_c - m_a$ only.

Let κ denote the maximum allowed KL divergence in (5.21) — we seek approximations with KL divergence not greater than κ. Without loss of generality, set $m_a = 0$ and $b_0 = 0$; then the KL divergence depends on m_c only. Fig. 5.3 shows the KL divergence between one Gaussian $Y(\mathbf{z})$ with $\mathbf{V}_a = 0.025 \cdot \mathbf{I}_2$ and a four-Gaussian mixture $\widetilde{R}(\mathbf{z})$ with $\mathbf{V}_c = 0.015 \cdot \mathbf{I}_2$ and $h = 1$.

The region $\mathcal{R}$ are the values of m_c where the KL divergence is less than or equal to the target value κ, that is:

$$\mathcal{R} = \{m_c \in \mathbb{C} \mid KL(m_c) \leq \kappa\}. \tag{5.22}$$

The threshold T is the radius of the largest disc that is fully contained in $\mathcal{R}$. The region $\mathcal{R}$ is not a disc, but we restrict the effective region to be the disc $|m_c| \leq T$, for ease of computation. Interpret T as T_1 if $\widetilde{R}(\mathbf{z})$ is one Gaussian, and as T_2 if $\widetilde{R}(\mathbf{z})$ is two Gaussians.

Next, in order to bound the KL divergence, assume that $\mathbf{V}_c$ and $\mathbf{V}_a$ have covariance zero, and the larger of the two variances is used as this has the greater effect on the KL divergence. Accordingly, the maximum element of the covariance matrix $\mathbf{V}_c$ and $\mathbf{V}_a$ is $v_{c,\max}$ and $v_{a,\max}$ respectively.

Since the target is to find the thresholds T_1 (one Gaussian) and T_2 (two Gaussians), $\widetilde{R}(\mathbf{z})$ is an l-Gaussian mixture where l equals one or two. To form a bound, k Gaussians are selected from $R(\mathbf{z})$ where $k \in \mathbb{Z}^+$ (ideally, $k \to \infty$, but since most resulting Gaussians have near-zero mixing coefficients, it is more effective to consider k most relevant Gaussians). Using these assumptions, an upper bound on (5.21) can be formed.

Proposition 2 *Let $R_k(\mathbf{z})$ consist of k Gaussians with mean m_c as in* (5.1)*. If $\widetilde{R}(\mathbf{z})$ is a single Gaussian approximation, then:*

$$D(Y(\boldsymbol{z})R(\boldsymbol{z})||Y(\boldsymbol{z})\widetilde{R}(\boldsymbol{z})) \leq \frac{(k-1)v_{a,max}}{2|h|^2(v_{c,max}+v_{a,max})\left(1+e^{\frac{-2(m_{c,Re}h_{Re}-m_{c,Im}h_{Im})+1}{2|h|^2(v_{c,max}+v_{a,max})}}\right)}. \tag{5.23}$$

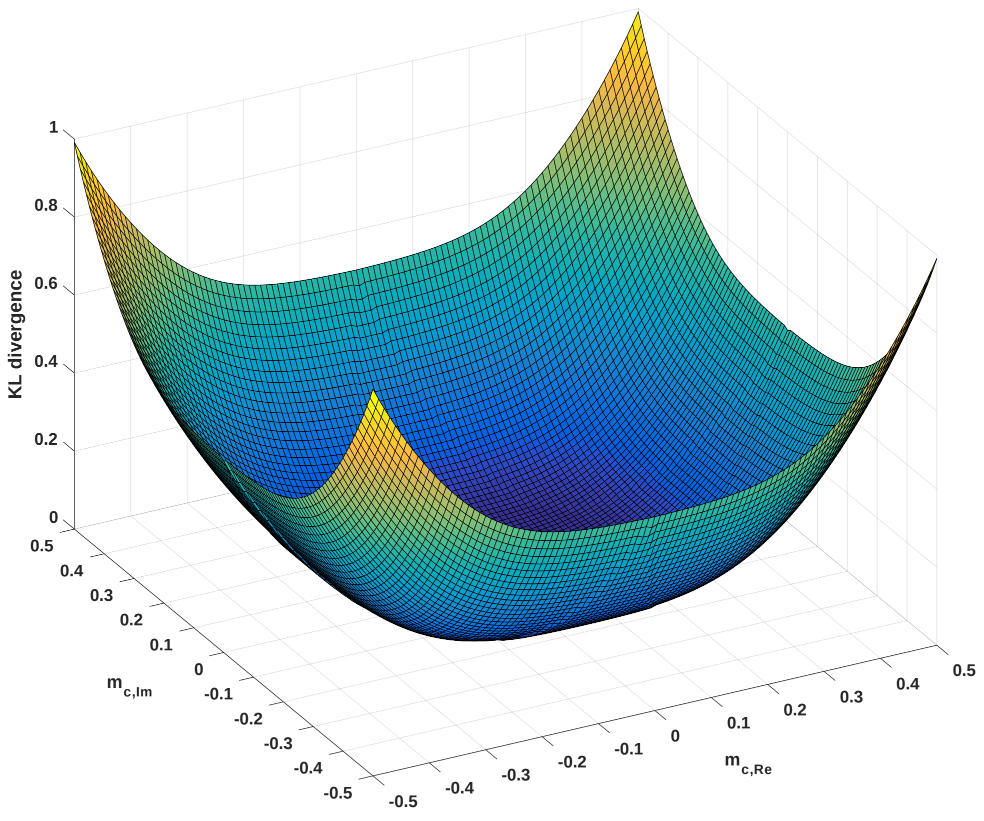

Figure 5.3: KL divergence of Gaussian integer between 1 and 4 Gaussian when $|h| = 1$, $\mathbf{V}_a = 0.025 \cdot \mathbf{I}_2$, $\mathbf{V}_c = 0.015 \cdot \mathbf{I}_2$.

If $\widetilde{R}(\mathbf{z})$ is a two Gaussian approximation, then:

$$D(Y(\boldsymbol{z})R(\boldsymbol{z})||Y(\boldsymbol{z})\widetilde{R}(\boldsymbol{z})) \leq \frac{(k-1)v_{a,max}}{|h|^2(v_{c,max}+v_{a,max})\left(1+e^{\frac{2(m_{c,Re}h_{Re}-m_{c,Im}h_{Im})-1}{2|h|^2(v_{c,max}+v_{a,max})}}\right)^2}. \tag{5.24}$$

The proof is given in Appendix B.

This can be used to form an upper bound on T_1 and T_2. The value of T_1 and T_2 is $0 \leq \{T_1, T_2\} \leq 1/h$ because T_1 and T_2 are calculated from the check-to-variable message after stretching by $1/h$. We are interested in the value of m_c for which the bound attains κ; this value is T_1 ($l = 1$) or T_2 ($l = 2$). In (5.21), $D(Y(\mathbf{z})R(\mathbf{z})||Y(\mathbf{z})\widetilde{R}(\mathbf{z}))$ has the worst case when $m_{c,Re} = m_{c,Im}$, so we assume $m_{c,Re} = m_{c,Im}$ in (5.23) and (5.24). We have T_1 as:

$$T_1 = \frac{|h|}{(h_{Re}-h_{Im})}\left(\frac{1}{2} - |h|^2(v_{a,max}+v_{c,max})\log\left(\left(\frac{(k-1)(v_{a,max})}{2\kappa|h|^2(v_{c,max}+v_{a,max})}\right)-1\right)\right), \tag{5.25}$$

For the two-Gaussian approximation, we have T_2 as:

$$T_2 = \frac{|h|}{(h_{Re}-h_{Im})}\left(\frac{1}{2} + |h|^2(v_{a,max}+v_{c,max})\log\left(\sqrt{\left(\frac{(k-1)(v_{a,max})}{\kappa|h|^2(v_{c,max}+v_{a,max})}\right)-1}\right)\right). \tag{5.26}$$

Note that from (5.20), $0 \leq \rho \leq 1$ applies to the check-to-variable message before stretching by $1/h$. So that T_1 and T_2 is comparable with ρ, the scalar $|h|$ is included in (5.25) and (5.26).

While (5.25) and (5.26) is not linear, we observed that it is roughly linear for parameters of interest, so linear regression is used to estimate T_1 and T_2. Fig. 5.4 and 5.5 show an example of T_1 and T_2 obtained from the upper bound and its linear approximation for Gaussian integer approximation.

For Gaussian integer decoding, the linear regression was found as:

$$T_1 = 0.4251 - 4.9594 v_{a,\max} - 0.6407 v_{c,\max} - 24.0404 v_{a,\max} v_{c,\max}, \tag{5.27}$$

$$T_2 = 0.6141 - 7.8082 v_{a,\max} - 1.4818 v_{c,\max} - 0.9999 v_{a,\max} v_{c,\max}, \tag{5.28}$$

respectively. For the Eisenstein integer decoding, the linear regression was found as:

$$T_1 = 0.4442 - 5.6516 v_{a,\max} - 1.2379 v_{c,\max} - 10.3436 v_{a,\max} v_{c,\max}, \tag{5.29}$$

$$T_2 = 0.4326 - 5.5644 v_{a,\max} - 1.0497 v_{c,\max} + 4.2929 v_{a,\max} v_{c,\max}, \tag{5.30}$$

respectively.

The constant value κ is set at 10^{-2}. This value was chosen to give a favorable performance-complexity trade off. The constant κ can be obtained from a numerical search of KL divergence. We start with small-valued of KL divergence, and use this value as the threshold to obtain T_1 and T_2 , then test the decoder symbol error rate (SER). Afterward, gradually increase it to obtain another T_1 and T_2 and test the SER again. This process is repeated until there is noticeable degradation in decoder SER, as determined by substantial simulations. That point will be set as κ, to optimize the trade off between complexity and performance. For GI and EI reliability-based CLDLC decoding, we set the constant value $\kappa = 10^{-2}$,

For the number of Gaussians in $R(\mathbf{z})$, k, we selected $k = 4$ and 3 for Gaussian integer and Eisenstein integer, respectively. In order to implement a practical algorithm, the same threshold for all check-to-variable messages independently of coefficient h is desired, so $h = 1$ was selected because this gives the smallest (i.e. most pessimistic) threshold value.

5.2.3 Gaussian Approximation at Variable Node

This section describes the approximation at the variable node in Section 2.7.2. Unlike existing decoding algorithms based real and complex numbers [8,19,20,23–26], the variable node function of the proposed algorithm adaptively selects the

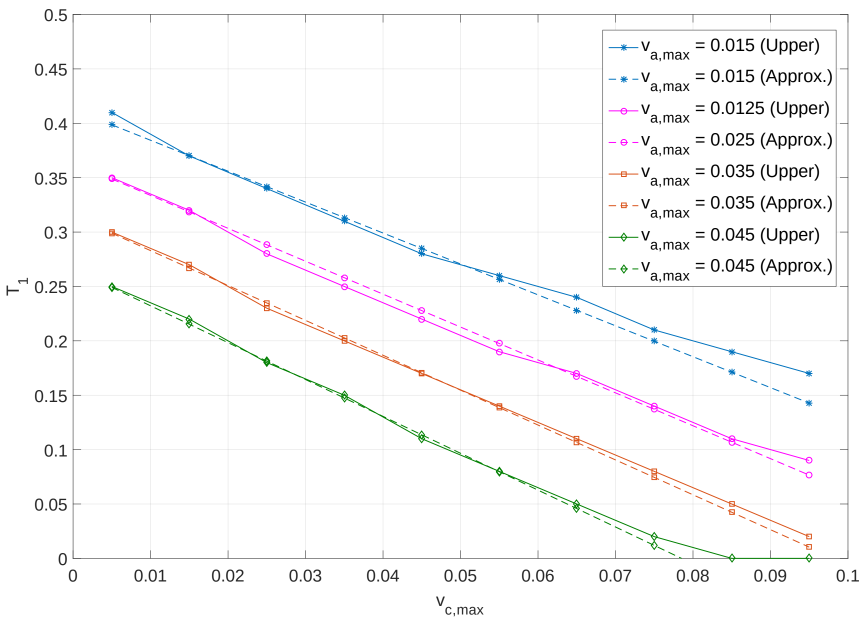

Figure 5.4: T_1 of Gaussian integer approximation, for $k = 4$ and $\kappa = 10^{-2}$. Solid lines represent T_1 calculated from the upper bound. Dashed lines represent T_1 approximated by linear regression.

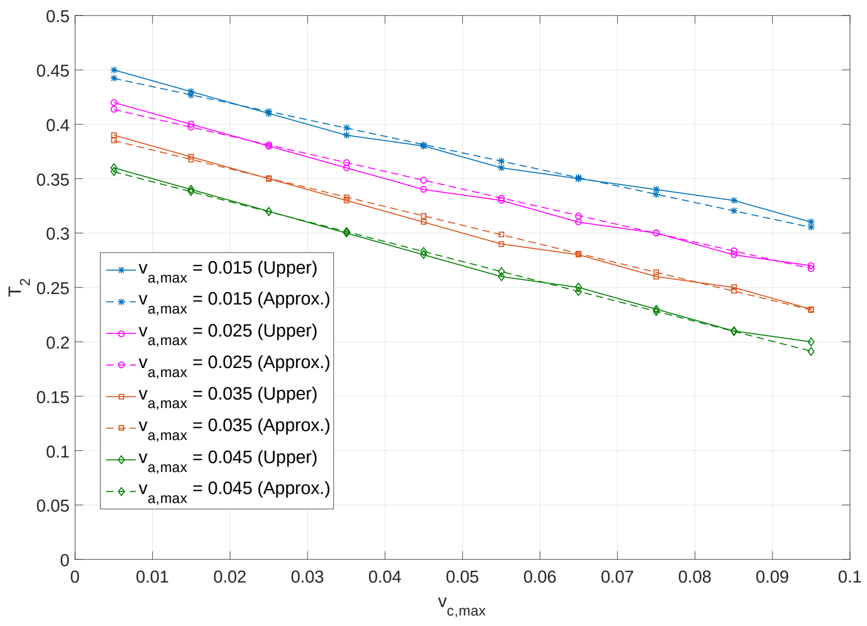

Figure 5.5: T_2 of Gaussian integer approximation, for $k = 4$ and $\kappa = 10^{-2}$. Solid lines represent T_2 calculated from the upper bound. Dashed lines represent T_2 approximated by linear regression.

number of Gaussians using the reliability of check-to-variable message. At each iteration, the variable node function is as follows.

- *Step 1, calculate reliability*: At a variable node, the input messages are $R_i(z)$ with mean $\mathbf{m}_i$ and variance $\mathbf{V}_i$, for $i = 1, 2, \ldots, d$. Find the corresponding reliabilities $\rho_1, \rho_2, ..., \rho_d$ using (5.20).

- *Step 2, find the number of Gaussians in message expansion*: For $i = 1, 2, \ldots, d$, the reliability ρ_i of each message is compared to the threshold T_1 and T_2. If the reliability satisfies $\rho_i \leqslant T_1$, then the number of Gaussians in the message expansion is $|\mathcal{B}_i| = 1$. If $T_1 < \rho_i \leqslant T_2$, then $|\mathcal{B}_i| = 2$. Otherwise, $|\mathcal{B}_i| = 3$ or 4 for Gaussian or Eisenstein integers integers, respectively.

- *Step 3, message expansion*: Each message $R_1(z), R_2(z), ..., R_d(z)$ is expanded to a periodic function using $|\mathcal{B}_i|$ Gaussians from step 2. The periodic Gaussian is
$$\widetilde{R}_i(\mathbf{z}) = \sum_{b \in \mathcal{B}_i} \mathcal{N}(\mathbf{z}; \mathbf{m}_i(\mathbf{b}), \mathbf{V}_i). \tag{5.31}$$
where the mean of each Gaussian is:
$$\mathbf{m}_i(\mathbf{b}) = \mathbf{m}_i + [h_i]^{-1}\mathbf{b}. \tag{5.32}$$
The set $\mathcal{B}_i$ for Gaussian and Eisenstein integers is found in Section 5.1 and 5.1.2, respectively.

In Step 2, the maximum number of Gaussians in the expansion $|\mathcal{B}|$ for Eisenstein integers is lower than for Gaussian integers. This is because Eisenstein integers have hexagonal Voronoi cells which has the tightest packing in two dimensions. Fig. 5.1 and 5.2 illustrate the case of check-to-variable message has high intensity of Gaussian noise r_3. Only 3 Eisenstein integers ($\mathbf{b}_0, \mathbf{b}_1$ and $\mathbf{b}_2$) are sufficient to cover all possibilities of check-to-variable message while Gaussian integers require up to 4 integers ($\mathbf{b}_0, \mathbf{b}_1, \mathbf{b}_2$ and $\mathbf{b}_3$) to cover all possibilities.

In Step 3, a question that might arise is how many Gaussians are sufficient for the expansion or approximation. To answer this question, we calculate the reliability ρ of check-to-variable messages and set the threshold T_1 and T_2 for choosing the number of expansion $|\mathcal{B}|$. The set $\mathcal{B}$ represents a subset of the Gaussian integer or Eisenstein integer. For Gaussian integer, the input messages of the variable node will select $|\mathcal{B}| = 1$ if the reliability $\rho \leqslant T_1$ (high reliability), then $|\mathcal{B}| = 2$ if $T_1 < \rho \leqslant T_2$ (medium reliability), otherwise $|\mathcal{B}| = 4$ is used (low reliability). For example, Fig. 5.1 shows the periodic expansion based on Gaussian integer. The first message r_1 with the smallest ρ_1, the value of ρ_1 is less than T_1 so, the single Gaussian set $\mathcal{B} = \{\mathbf{b}_0\}$ will be used, where $\mathbf{b}_0$ can be calculated from (5.3). In the second case, consider the message r_2 with ρ_2, the value of ρ_2 is between T_1 and T_2 so, the two Gaussian set $\mathcal{B} = \{\mathbf{b}_0, \mathbf{b}_1\}$ will be used, where $\mathbf{b}_1$ can be calculated from (5.5). And the last one, consider the message r_3 with ρ_3, the value of the value of ρ_3 is more than T_2 so, the four Gaussian set $\mathcal{B} = \{\mathbf{b}_0, \mathbf{b}_1, \mathbf{b}_2, \mathbf{b}_3\}$ will be used, where $\mathbf{b}_2, \mathbf{b}_3$ can be calculated from (5.7) and (5.8).

In the case of Eisenstein integers, the input messages to the variable node will select $|\mathcal{B}| = 1$ if the reliability $\rho \leqslant T_1$, then $|\mathcal{B}| = 2$ if $T_1 < \rho \leqslant T_2$, otherwise $|\mathcal{B}| = 3$ is used. Fig. 5.2 shows the periodic expansion based on Eisenstein integer. The periodic expansion concept is similar to the Gaussian integer case but, in the last case, when ρ_3 is more than T_2, the three Gaussian set $\mathcal{B} = \{\mathbf{b}_0, \mathbf{b}_1, \mathbf{b}_2\}$ will be used, where $\mathbf{b}_0, \mathbf{b}_1, \mathbf{b}_2$ can be calculated from (5.10)-(5.13), (5.15) and (5.17), respectively.

5.3 Numerical Results

5.3.1 Error Rate for Reliability-Based Decoder

The error rate of the reliability-based parametric CLDLC decoder was evaluated on the unconstrained input power complex AWGN channel. We compared five decoding algorithms: 1) real-valued LDLC with the 3 Gaussian decoder [25], 2) CLDLC based on GMR algorithm with 9 Gaussians [20]; CLDLC decoding with a fixed number of Gaussians: 3) 4 Gaussians (4 GI-CLDLC) and 4) 2 Gaussians CLDLC (2 GI-CLDLC), which are the extension of real-valued decoding [25] to the complex case. Finally, 5) the proposed reliability-based CLDLC decoder. Here, complex-valued lattices with dimension $n = 8$, 49, 500 and 5,000 were used, and the matrix $\mathbf{H}$ had row and column weights d of 3, 3, 5 and 7, respectively. The elements of the GI and EI vector $\mathbf{b}$ in equation (2.18) are randomly chosen from the set of integers $b_{Re}, b_{Im} \in \mathcal{Z}$, where $\mathcal{Z} = \{-10, -9, \ldots, 9, 10\}$. For 9 Gaussians with GMR algorithm, we follow the settings of [20] using $M = 10, K = 3, VarRangeLen = 0.4$ and $CheckRangeLen = 0.05$.

For real-valued LDLC lattices, dimension $n = 16$, 100, 1,000 and 10,000 was used, and the row and column weights are the same as CLDLC and the elements of the real integer vector $\mathbf{b}$ are randomly chosen from $\mathcal{Z}$. Note that we choose the three-Gaussian LDLC decoding algorithm [25] for decoding real-valued LDLC because this decoding algorithm yields similar performance to the quantized decoding algorithm but with less complexity. The quantized decoding algorithm proposed by [19] in 2008. This algorithm give very good performance but the Gaussian mixtures are quantized, so leads to high computational complexity, it also requires large storage.

The inverse generator matrix was created with the generator sequence $\mathbf{h} = \{1, \frac{1}{\sqrt{d}}, ..., \frac{1}{\sqrt{d}}\}$, and the matrix was normalized such that $V(\Lambda) = 1$. At each VNR, we simulated until the number of symbol errors and word errors reach 1,000 and 500 at least, respectively (must satisfy both conditions). The maximum number of iterations for the belief propagation (BP) decoder is 50 iterations.

The constant κ can be obtained from a numerical search of KL divergence as described in Section 5.2.2. We start with small-valued of KL divergence, and use this value as the threshold to obtain T_1 and T_2 , then test the decoder symbol error rate (SER). Afterward, gradually increase it to obtain another T_1 and T_2 and test the SER again. This process is repeated until the decoder SER has noticeable degradation, and that point will be set as κ for trade off between complexity and performance. For GI and EI reliability-based CLDLC decoding, we set the constant value $\kappa = 10^{-2}$, and T_1 and T_2 can be calculated from (5.27)–(5.30).

Fig. 5.6 shows the symbol error rate of each decoding algorithm for CLDLC based on Gaussian integers (a symbol error occurs when $\widehat{b}_i \neq b_i$). The result shows that reliability-based decoding gives the best performance, when $n \leqslant 500$. In addition, CLDLC outperforms real-valued LDLC when $n \leqslant 500$. For $n = 5000$ all CLDLC decoding algorithms based on Gaussian integers yield the same performance as the real-valued LDLC.

Fig. 5.7 shows the symbol-error rate of decoding CLDLC based on Eisenstein integers. We also compared five algorithms: 1) 3 Gaussians LDLC, 2) 7 EI-CLDLC, 3) 3 EI-CLDLC, 4) 2 EI-CLDLC, and 5) EI reliability-based CLDLC decoder. The general tendency of the results are similar to CLDLC based on Gaussian integers. The reliability-based decoding algorithm based on Eisenstein integers gives the best performance among four CLDLC decoding algorithms, and EI-CLDLC outperforms the real-valued LDLC when $n \leqslant 500$.

Fig. 5.8 shows the word error rate (WER) comparison between GI and EI reliability-based CLDLC and polar lattices [9]. This result shows that GI and EI reliability-based CLDLC outperforms polar lattices by 0.5 and 0.6 dB at WER = 10^{-4}. The decoding complexity is mentioned in the next subsection.

5.3.2 Reliability-Based Decoder Complexity

The complexity of the proposed reliability-based decoding algorithm and existing algorithms are described in this subsection.

For GMR decoding of CLDLC lattices [20], the storage requirement needed after the GMR algorithm is $\mathcal{O}(n \cdot d \cdot M)$. The computational complexity is $\mathcal{O}(n \cdot d \cdot t \cdot K^2 \cdot M^3)$, and is dominated by sorting and searching in tables, where n is the lattice dimension, d is the degree of the inverse generator matrix, t is the number of iterations, K is the number of replications and M is the number of Gaussians in each list. $K = 3$ and $M = 10$ were used in [20].

[25] proposed the three/two Gaussians decoding algorithm to reduce the number Gaussian mixtures in LDLC (real number case). It is straightforward to extend this algorithm to the complex case; we call this GI-CLDLC and EI-CLDLC decoding. The computational complexity of GI-CLDLC are $\mathcal{O}(n \cdot t \cdot 2^{d-1})$ and $\mathcal{O}(n \cdot t \cdot 4^{d-1})$ for 2 Gaussian replications and 4 Gaussian replications, respectively.

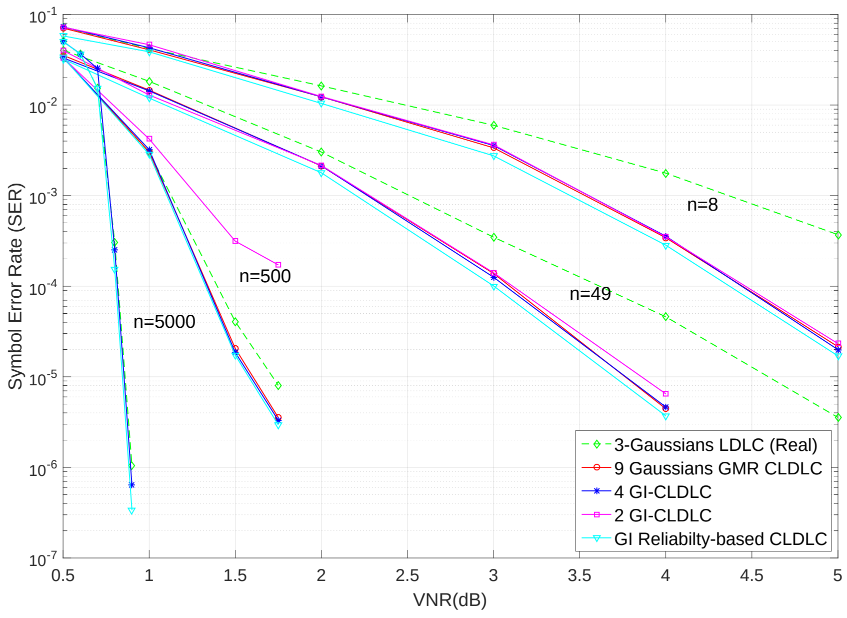

Figure 5.6: Performance comparison in terms of VNR vs symbol error rate of GI-CLDLC

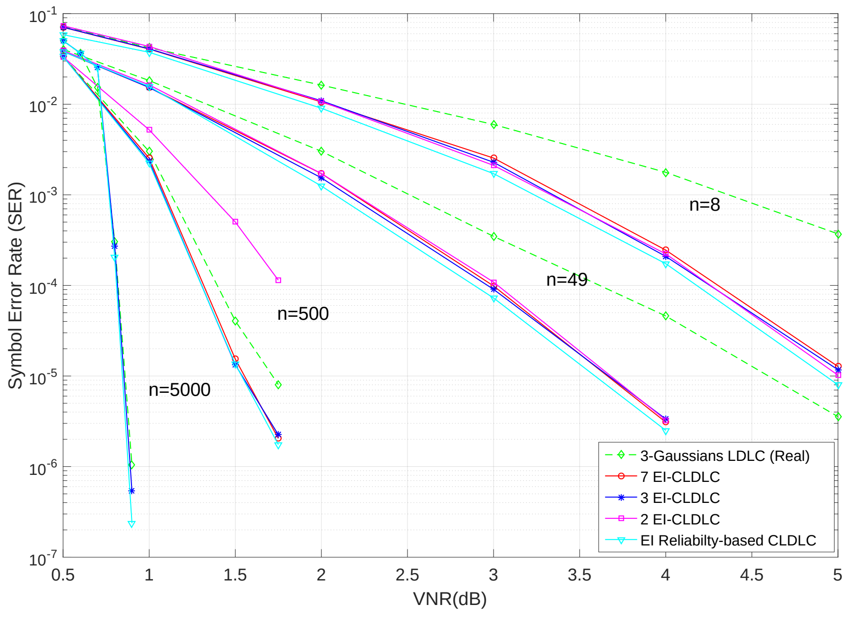

Figure 5.7: The performance comparison in terms of VNR vs symbol error rate of GI-CLDLC

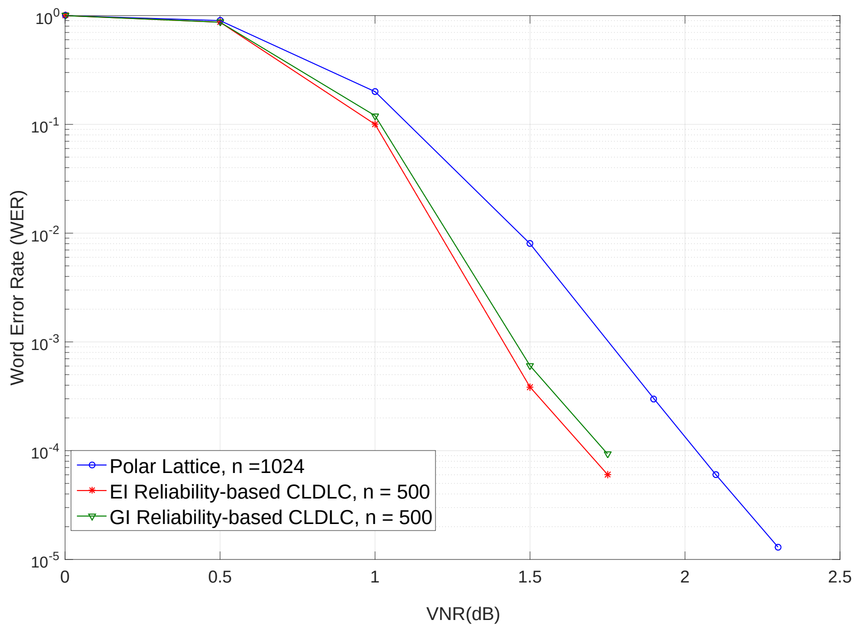

Figure 5.8: The performance comparison between GI and EI reliability-based CLDLC with Polar lattices [9] in terms of VNR vs word error rate.

For EI-CLDLC, the complexity are $\mathcal{O}(n \cdot t \cdot 2^{d-1})$ and $\mathcal{O}(n \cdot t \cdot 3^{d-1})$ for 2 Gaussian replications and 3 Gaussian replications, respectively. After the moment matching (MM) step, the storage requirement needed is $5 \cdot n \cdot d$ because the message passed between the check and variable nodes are single complex Gaussian functions which are represented by five parameters, 2×1 mean vector $\mathbf{m}$ and 2×2 covariance matrix $\mathbf{V}$.

The computational complexity of the proposed reliability-based decoding algorithm is $\mathcal{O}(n \cdot t \cdot |\widetilde{\mathcal{B}}|^{d-1})$, where $|\widetilde{\mathcal{B}}|$ is an average number of Gaussians in the periodic expansion step. This complexity analysis assumes that $|\mathcal{B}|$ is independent for each edge. While not strictly independent, the sparse graph is locally tree-like, making this a good approximation. After the MM algorithm, the storage requirement is $5 \cdot n \cdot d$. $|\widetilde{\mathcal{B}}|$ as a function of the 3 parameters n, d and VNR is shown in Fig. 5.9. For example, for a fixed VNR, we can see that $|\widetilde{\mathcal{B}}|$ decreases when n and d increases. On the other hand, for a fixed n and d, $|\widetilde{\mathcal{B}}|$ decreases when VNR increases. The mean of $|\widetilde{\mathcal{B}}|$ ranges from 3.75 (at VNR = 0.5 dB) to

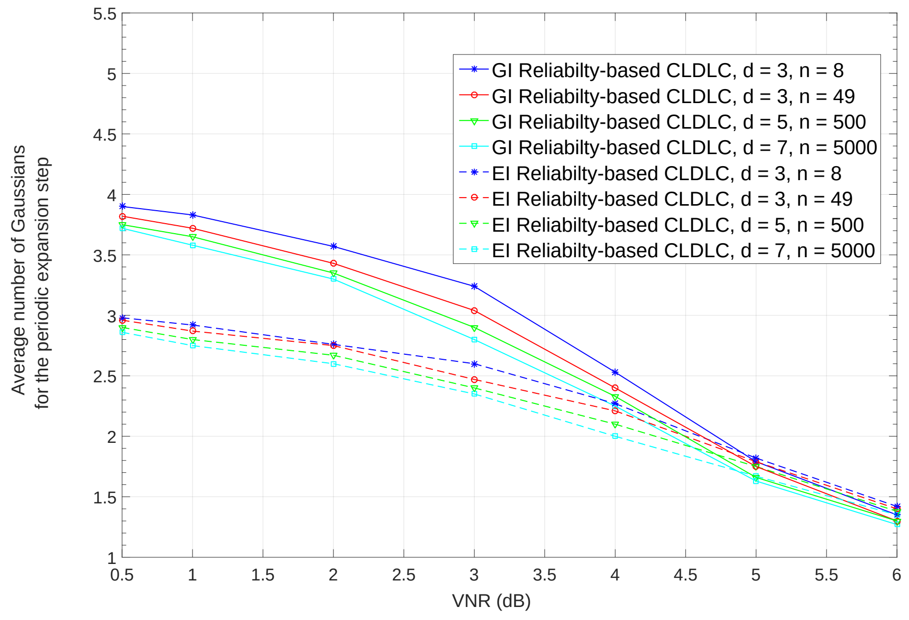

Figure 5.9: Average number of Gaussians $|\mathcal{B}|$ of the reliability-based parametric decoder in the periodic expansion step vs VNR for each lattice dimension, solid lines and dash lines show GI-CLDLC and EI-CLDLC, repectively.

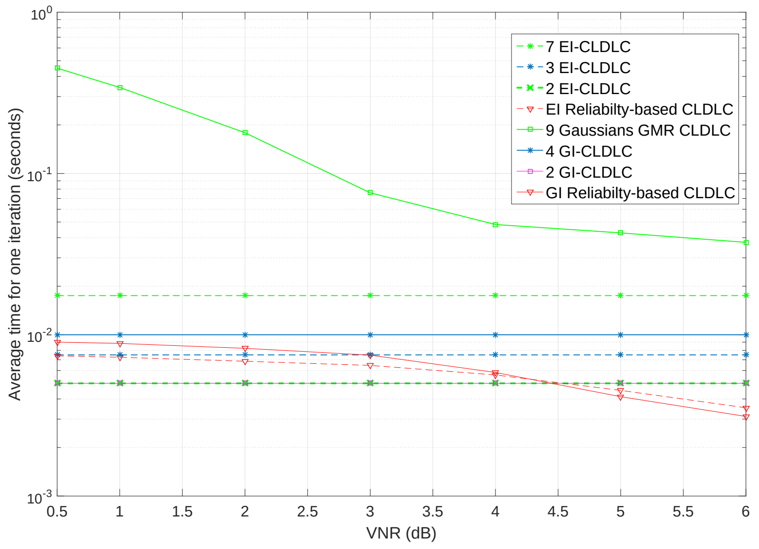

Figure 5.10: Time comparison between 9 Gaussians GMR, 4 GI-CLDLC, 2 GI-CLDLC, GI reliability-based CLDLC, 7 EI-CLDLC, 3 EI-CLDLC, 2 EI-CLDLC and EI reliability-based CLDLC, for CLDLC dimension $n = 8$ and degree $d = 3$.

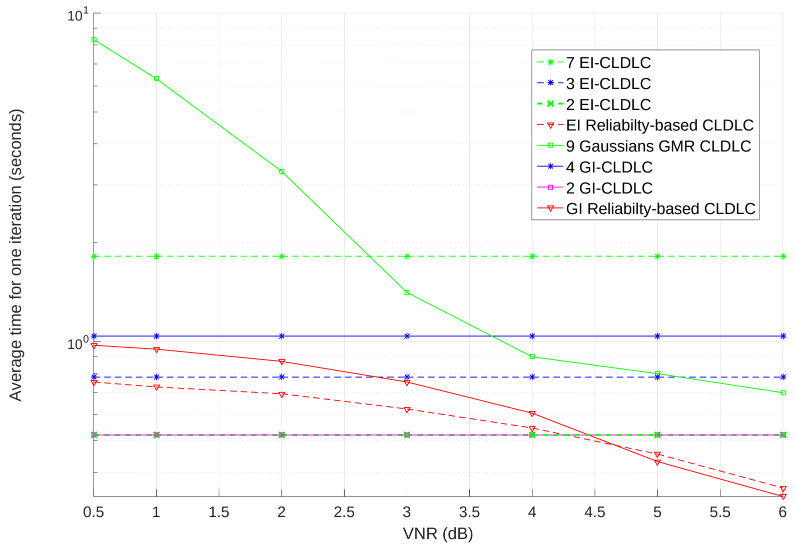

Figure 5.11: Time comparison between 9 Gaussians GMR, 4 GI-CLDLC, 2 GI-CLDLC, GI reliability-based CLDLC, 7 EI-CLDLC, 3 EI-CLDLC, 2 EI-CLDLC and EI reliability-based CLDLC, for CLDLC dimension $n = 500$ and degree $d = 5$.

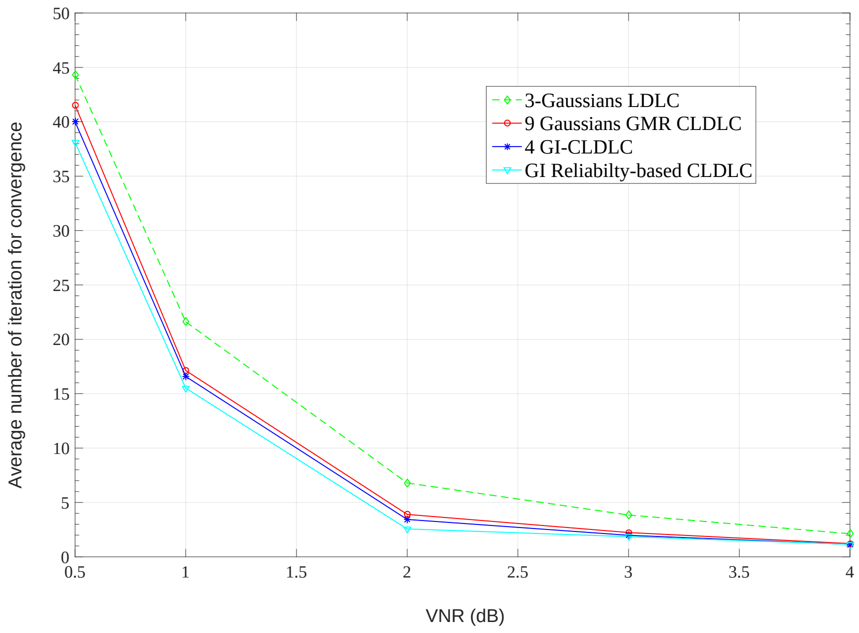

Figure 5.12: Average number of iterations required for GI-CLDLC decoder convergence in terms of VNR, for CLDLC demension $n = 500$ and degree $d = 5$.

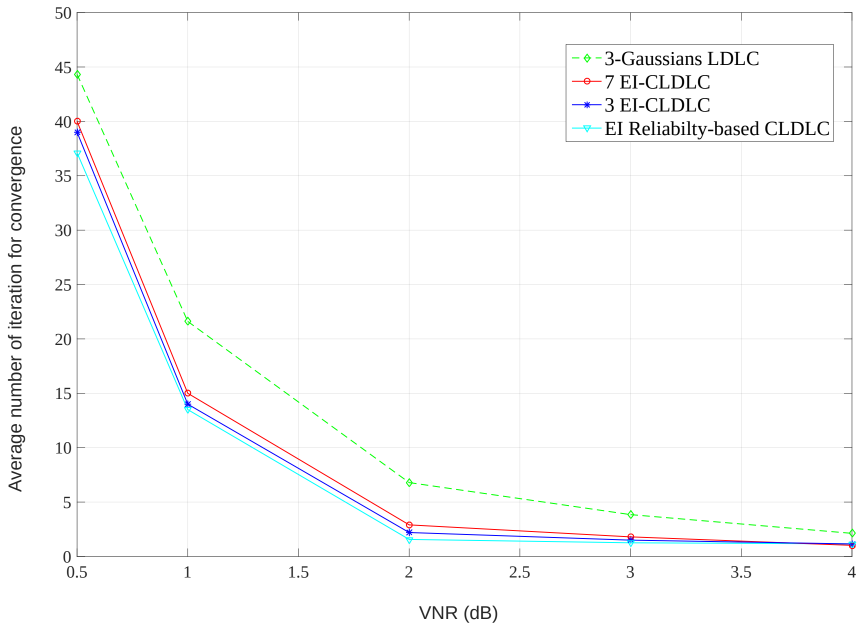

Figure 5.13: Average number of iterations required for EI-CLDLC decoder convergence in terms of VNR, for CLDLC demension $n = 500$ and degree $d = 5$.

1.35 (at VNR = 6 dB) Gaussians on average for GI reliability-based, and 2.9 (at VNR = 0.5 dB) to 1.38 (at VNR = 6 dB) Gaussians on average for EI reliability-based. Significantly, when GI and EI reliability-based decoding are compared, the EI decoder has lower complexity at $n = 49, 500$ and 5000, when the VNR range is 1–4 dB. This is due to the tighter packing of the Eisentein integers which allows fewer Gaussians for a good approximation, as was discussed earlier.

Polar lattices [9] have lower decoding complexity than CLDLC lattices. The overall decoding complexity is $\mathcal{O}(rn \log n)$, where r is the number of levels of lattice partitions, $r = 2$ in this paper.

The computation time of the reliability-based decoder based on Gaussian integers and Eisenstein integers, 9 Gaussians GMR algorithm, 4 GI-CLDLC, 2 GI-CLDLC, 7 EI-CLDLC, 3 EI-CLDLC and 2 EI-CLDLC are shown in Fig. 5.10 and 5.11. Here, we consider two lattice dimensions $n = 8$ and 500 with degree $d = 3$ and 5, respectively. For $n = 8$, 2 GI-CLDLC and 2 EI-CLDLC give comparable performance to other algorithms, as was shown in Fig. 5.6 and 5.7, and also give the lowest computation time when VNR < 4.5 dB. When VNR $\geqslant 4.5$ dB reliability-based decoders provide the lowest computation time.

For $n = 500$, Fig. 5.11 shows that 2 GI-CLDLC and 2 EI-CLDLC give the lowest computation time when VNR < 4.5 and 4.25 dB, respectively. However, the 2 Gaussians expansion loses performance when $n \geqslant 49$, as was shown in Fig. 5.6 and 5.7. Therefore, reliability-based decoders provide the lowest computation time that also have good decoder error rates when $n \geqslant 49$. In addition, if we compare GI reliability-based decoding and EI reliability-based decoding, EI shows lower computation time when VNR $\leqslant 4.6$ dB, again because the EI integers provide a good approximation with fewer Gaussians.

In Fig. 5.12 and 5.13, the average number of iterations required for decoder convergence is shown. We took a sample of 10,000 converged codewords (non-converging codewords are ignored) and evaluate the mean of the number of iterations required. The number of iterations reduces when the VNR increases. Fig. 5.12 and 5.13 show the number of iterations for GI decoding and EI decoding, respectively. The complex-valued LDLC decoder needs fewer iterations for convergence compared to the real-valued LDLC. Both GI and EI reliability-based decoding require the fewest iterations for convergence.

5.4 Concluding Remarks

We propose a construction of CLDLC based on Eisenstein integers, to reduce the complexity of CLDLC decoding. We defined the likelihood-based reliability of the check-to-variable messages and the threshold for choosing the number of finite Gaussian for each incoming message at the variable node. This allows each message to be approximated by a variable number of Gaussians depending upon its reliability; a single Gaussian when the message has high reliability leads to low complexity and good performance. The threshold can be found using the Kullback-Leibler (KL) divergence between the approximation and the true distribution, but explicitly computing the divergence is inefficient. Instead, we form an upper bound on this KL divergence, and linear regression is used to efficiently estimate this threshold.

We compared 6 algorithms: 1) 9 Gaussians with GMR algorithm based on Gaussian integer [20], 2) 7 EI-CLDLC (the maximun number of Eisenstein integer approximation), and the extended algorithms from [25] (real number case) to the complex case which are 3) 4 GI-CLDLC, 4) 2 GI-CLDLC, 5) 3 EI-CLDLC, 6) 3 EI-CLDLC.

Our results show that the reliability-based decoding algorithm for Eisenstein integers gives the lowest complexity when $n \geqslant 49$. In addition, reliability-based decoding algorithms based on both Eisenstein integers and Gaussian integers shows the best performance when $n \leqslant 500$. Eisenstein integers provides lower complexity than Gaussian integers because the hexagonal Voronoi cells of the Eisenstein integer lattice has the tightest packing in two dimensions, leading to a higher reliability than Gaussian integers for the same fixed number of Gaussians in the approximation.

Chapter 6

Research on LDPC: EM Algorithm for DMC Channel Estimation in NAND Flash Memory

NAND flash memory systems read data by setting a threshold to obtain binary channel outputs, but the allowed number of reads is limited. Due to manufacturing variations, device aging and high storage temperature, the true underlying channel parameters may be difficult to model accurately. Thus, the channel can be approximated by a discrete memoryless channel (DMC) with unknown channel transition probabilities and a small number of outputs. Low-density parity-check (LDPC) decoders require a channel estimate, and incorrect channel estimation degrades the word-error rate (WER) of the LDPC decoder.

In this abstract, the expectation maximization (EM) algorithm is used to estimate unknown DMC transition probabilities. In general, the EM algorithm estimates the parameters of a statistical model. The LDPC decoder plays a key role, providing a priori estimates of the code bits, required by the EM algorithm.

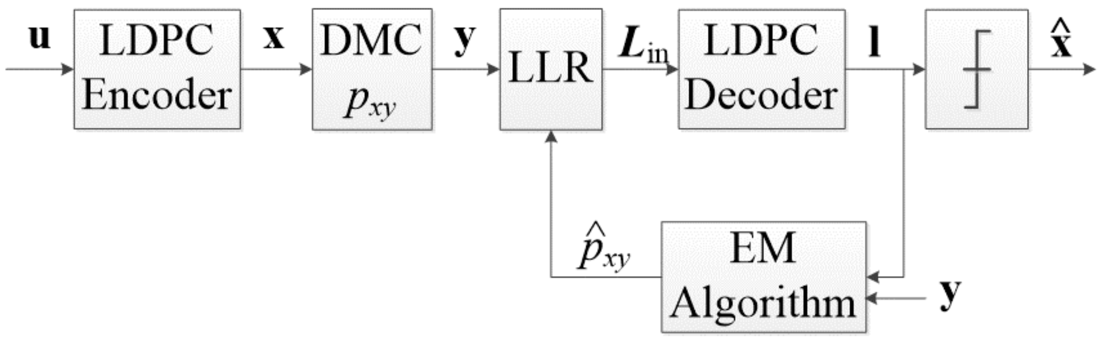

Figure 6.1: Block diagram for an LDPC-coded NAND flash memory system. The decoder uses the EM algorithm in turbo equalization-style setting.

The only previous work we are aware of applying the EM algorithm to DMC estimation is by Boutros et al. [44]. This work also includes an LDPC code to aid estimation. However, only the symmetrical errors-and-erasures channel with three outputs was considered, whereas our aim is to consider general DMC channels. [45] proposed alternative methods using ML, MAP or MSE for estimating the flash memory error model. However, those metrics are calculated based on the number of bits that were read from the same voltage bin and participate in unsatisfied parity check equations, whilst the EM algorithm does not need above information. Recurrent neural networks (RNNs) can also be applied to this problem; the method proposed in [46] is targeted at NAND flash memories, but requires a large amount of training data, whereas the EM algorithm requires no training data.

6.1 EM algorithm

Consider a statistical model which produces an observed sequence $\mathbf{y}$, and has an unobserved sequence $\mathbf{x}$. These are generated by a set of unknown parameters, represented by the vector $\mathbf{p}$. Maximum likelihood estimation $\mathbf{p}^* = \arg\max_{\mathbf{p}} \log \Pr[\mathbf{y}|\mathbf{p}]$ is usually computationally intractable.

The EM algorithm is an iterative method to estimate the unknown channel parameters $\mathbf{p}$. The following is the general EM algorithm [47, p. 441]:

1. Choose initial $\mathbf{p}^{\text{old}}$.
2. **E Step** Evaluate $\Pr(\mathbf{x}|\mathbf{y}, \mathbf{p}^{\text{old}})$.
3. **M Step** Evaluate

$$\mathbf{p}^{\text{new}} = \arg\max_{\mathbf{p}} \sum_{\mathbf{x}} \Pr(\mathbf{x}|\mathbf{y}, \mathbf{p}^{\text{old}}) \log \Pr(\mathbf{x}, \mathbf{y}|\mathbf{p}) \tag{6.1}$$

4. Check for convergence of the log likelihood or the parameter values. If convergence is not achieved, then $\mathbf{p}^{\text{old}} \leftarrow \mathbf{p}^{\text{new}}$ and go to Step 2.

With each iteration, the log-likelihood will increase until it approaches a local maximum.

While the EM algorithm can be applied to general DMCs, for clarity the description is given for a general two-output channel; this may be termed the binary asymmetric channel (BAC). The BAC has error probability $[1|0] = p_0$ and $[0|1] = p_1$. The channel input sequence is $= (x_1, x_2, \ldots, x_n)$ and the channel output sequence is $= (y_1, y_2, \ldots, y_n)$. The a priori information is $\underline{l} = (l_1, l_2, \ldots, l_n)$ where $l_i = \Pr[_i = 0]$. The goal is to estimate the BAC error probability $\widehat{p}_0, \widehat{p}_1$ from $_i$ and $l_1, \ldots, l_n$ using the EM algorithm.

For the E Step, use the initial estimates $\widehat{p}_0 \leftarrow p_0^{\text{old}}$ and $\widehat{p}_1 \leftarrow p_1^{\text{old}}$, and compute the function $q(x, y, l)$ given by:

$$= \begin{cases} \frac{(1-\widehat{p}_0)l}{(1-\widehat{p}_0)l+\widehat{p}_1(1-l)} & (x,y) = (0,0) \\ \frac{\widehat{p}_1(1-l)}{(1-\widehat{p}_0)l+\widehat{p}_1(1-l)} & (x,y) = (1,0) \\ \frac{\widehat{p}_0 l}{\widehat{p}_0 l+(1-\widehat{p}_1)(1-l)} & (x,y) = (0,1) \\ \frac{(1-\widehat{p}_1)(1-l)}{\widehat{p}_0 l+(1-\widehat{p}_1)(1-l)} & (x,y) = (1,1) \end{cases} \tag{6.2}$$

For the M Step, perform the following optimization:

$$\widehat{p}_0, \widehat{p}_1 = \arg\max_{p_0,p_1} \sum_{i=1}^{n} \sum_{x} q(x_i, y_i, l_i) \log[x_i, y_i]. \tag{6.3}$$

Using Lagrange multipliers, the estimates $\widehat{p}_0$ and $\widehat{p}_1$ are:

$$\widehat{p}_0 = \frac{\sum_{i:y_i=1} q(0,1,l_i)}{\sum_{i:y_i=1} q(0,1,l_i) + \sum_{i:y_i=0} q(0,0,l_i)} \tag{6.4}$$

$$\widehat{p}_1 = \frac{\sum_{i:y_i=0} q(1,0,l_i)}{\sum_{i:y_i=0} q(1,0,l_i) + \sum_{i:y_i=1} q(1,1,l_i)} \tag{6.5}$$

6.2 LDPC-coded NAND flash memory system

Consider the use of the EM algorithm in the following LDPC-coded system in fig:blockDiagram. Information **u** is encoded to an LDPC codeword **x** of an (n, k) LDPC code. Then, the codeword **c** is programmed to each cell of NAND flash memory. To model possible charge leakage, a constant offset $a \geq 0$ is subtracted from programmed levels. Additive white Gaussian noise z_i, with mean 0 and variance , is added to the signal $\widetilde{y}_i = x_i + z_i - a$, for $i = 1, \ldots, n$. The SNR definition is 1/. To model the NAND flash read process, $\widetilde{y}$ is quantized to two levels by a threshold at 0, producing two channel output levels. This is a binary asymmetric channel (BAC) where the error probabilities p_0, p_1 depend on a, σ^2.

Numerical evaluations are performed using this channel. The (4608,4096) and (36864,32768) LDPC codes have rate 8/9, column weight of 3, and the row weight varies from 26–29 and 26–28, respectively. This LDPC code is constructed using the progressive-edge-growth (PEG) algorithm [48]. For the LDPC decoder, we use the belief propagation (BP) algorithm. Numerical results for three systems are shown in fig:vnr-vs-wer.

In the **nonblind system**, the detector knows p_{xy} exactly and these are used in channel estimation. Decoding is performed for $I_{\text{LDPC}} = 30$ and 90 iterations. This represents the ideal case, since the channel is known.

In the **blind system**, the detector does not know p_{xy}. Instead, it assumes a symmetric channel with low SNR, specifically $\widehat{a} = 0$ and $\widehat{SNR} = 5$ dB. Decoding is performed for $I_{\mathrm{LDPC}} = 30$ iterations.

In the **EM system**, the decoder uses the EM algorithm in a turbo equalization-style setting, as shown in fig:blockDiagram. Initially, the decoder has no knowledge of p_{xy} and makes an initial estimate $\widehat{p}_{xy}$. This is used to compute LLRs, $L_{\mathrm{in}} = \log \frac{\Pr[x=0|y_i]}{\Pr[x=1|y_i]} = \log \frac{\widehat{p}_{0y_i}}{\widehat{p}_{1y_i}}$. Using these, the LDPC decoder operates for I_{LDPC} iterations, and produces soft outputs $\underline{l}$. Next the EM algorithm performs channel estimation. It has inputs , a priori information $\underline{l}$ and the old estimate $\widehat{p}_{xy}^{\mathrm{old}}$. It operates for I_{EM} iterations. It outputs a new channel estimate $\widehat{p}_{xy}^{\mathrm{new}}$. This is used to improve the computation of the LLRs, which is used for the next I_{LDPC} decoding iterations.

In the numerical results, the EM system initially assumes the same conditions as the blind system. Three iteration schedules are shown, $I_{\mathrm{LDPC}} \times I_{\mathrm{turbo}} = 10 \times 3, 15 \times 2$ and 30×3. The EM algorithm operates for $I_{\mathrm{EM}} = 8$ iterations. The results show that the 15×2 and 30×3 EM system give the closest WER to the nonblind system with $I_{LDPC} = 30$ and 90 iterations, respectively. The 10×3 EM system loses the performance around 0.1 dB due to worse estimation of $\widehat{p}_{xy}$.

6.3 Concluding Remarks

The main conclusion from the numerical results is that the EM algorithm can estimate unknown channel error probabilities well enough to provide the similar WER as the nonblind detector which knows the channel exactly. This is about 0.3 dB better than the blind system which has no channel knowledge.

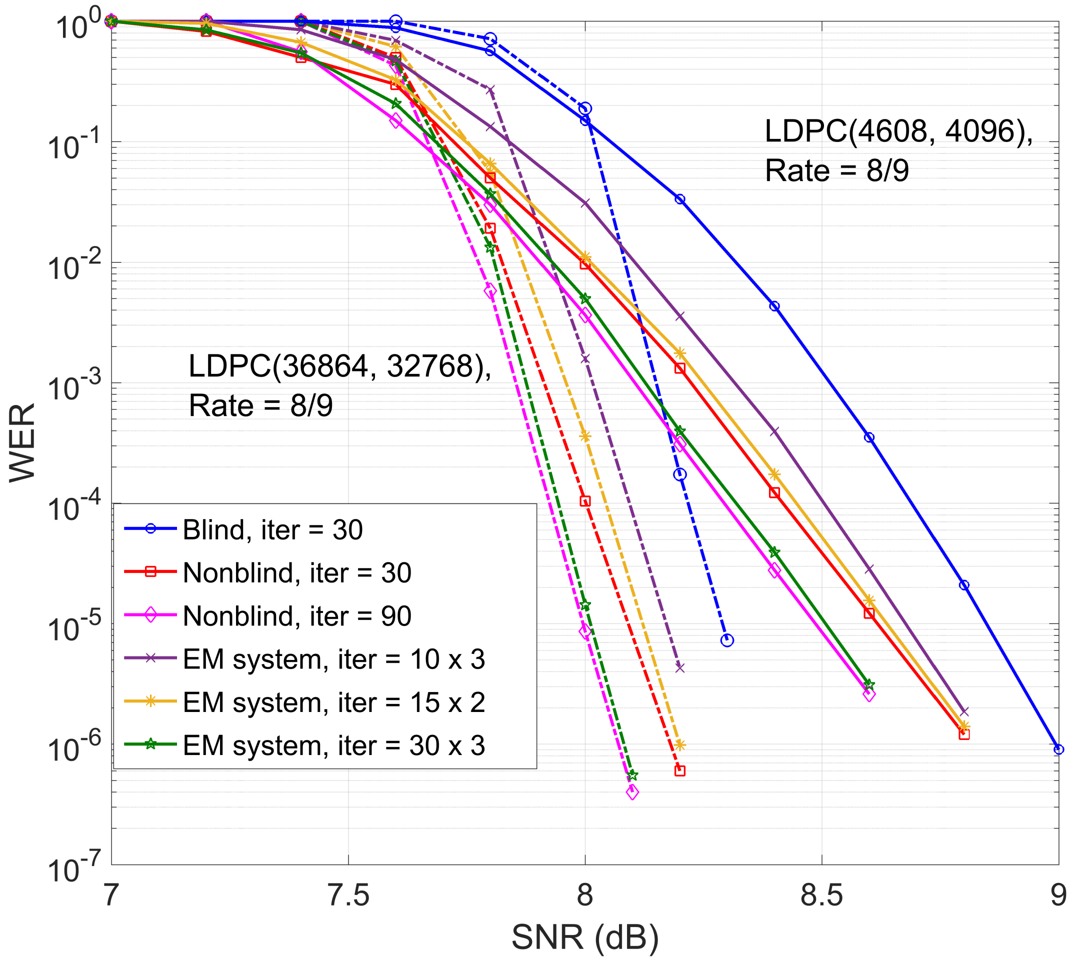

Figure 6.2: WER results for channel with offset $a = 0.5$. Rate 8/9 LDPC code with $n = 4608$ (solid lines) and 36864 (dashed lines).

Chapter 7

Conclusion

We proposed a construction of CLDLC based on Eisenstein integers, to reduce the complexity of CLDLC decoding. We defined the reliability of the check-to-variable messages for choosing the number of finite Gaussian for each incoming message at the variable node. Each incoming message at the variable node can be approximated by a varying number of finite Gaussians, depending on its reliability. Therefore, the number of finite Gaussians will be minimized for each incoming message, reducing complexity. The reliability-based decoding algorithm was applied to CLDLC based on Gaussian integers as well.

We compared 6 algorithms: 1) 9 Gaussians with GMR algorithm based on Gaussian integer [20], 2) 7 EI-CLDLC (the maximun number of Eisenstein integer approximation), and the extended algorithms from [25] (real number case) to the complex case which are 3) 4 GI-CLDLC, 4) 2 GI-CLDLC, 5) 3 EI-CLDLC, 6) 3 EI-CLDLC. Our results show that the reliability-based decoding algorithm for Eisenstein integers gives the lowest complexity when $n \geqslant 49$. In addition, reliability-based decoding algorithms based on both Eisenstein integers and Gaussian integers shows the best performance when $n \leqslant 500$. Eisenstein integers provides lower complexity than Gaussian integers because the hexagonal Voronoi cells of the Eisenstein integer lattice has the tightest packing in two dimensions, leading to a higher reliability than Gaussian integers for the same fixed number of

Gaussians in the approximation.

For CLDLC construction, we propose a new CLDLC construction called *relaxed Latin square* which can be considered as an irregular CLDLC and also provide a new convergence condition.Based on this new construction, we introduce the triangular structure of relaxed Latin square CLDLC which is designed based on the modified array LDPC Codes. The triangular structure is good for low complexity and fast encoding process, and it is suitable for the shaping operations.

Research on LDPC: For minor research, we proposes using the expectation maximization (EM) to estimate channel transition probabilities of NAND flash memory. The numerical results is that the EM algorithm can estimate unknown channel error probabilities well enough to provide the similar WER as the nonblind detector which knows the channel exactly. This is about 0.2 dB better than the blind system which has no channel knowledge.

Chapter 8

Future Works

For CLDLC, the future works are as follows:

1) For a new CLDLC construction called *relaxed Latin square* for a check matrix **H**, this construction can be used to design the triangular structure which is good for low complexity and fast encoding process, and it is suitable for the shaping operations. However, the message in the corner of the triangular structure will have less protection because the row/column weight is only one. This will lead to performance loss compared to the Latin square structure. Finding the method to protect the message in the conner is necessary. In addition, since the relaxed Latin square is the regular CLDLC, an optimal row and column weight are still unknown. These two topics are future works for the relaxed Latin square matrix.

2) For CLDLC decoder, the constant κ can be obtained from a numerical search of KL divergence. The new method to find κ is the future work.

For EM algorithm for DMC channel, the future work is that we plan to develop a fixed-point implementation of the EM estimation algorithm.

Appendix A

Convergence Analysis of CLDLC for Latin Squares

Proof of Proposition 1 For transmission over the complex AWGN channel, the variance $\mathbf{Y}$ has zero covariance. The messages between the variable nodes and check nodes are a mixture of complex Gaussians. The central observation is that when no approximations are performed, all the Gaussians in the mixture have the same variance, and for each Gaussian has zero covariance, and real variance component and complex variance component are equal. At check node i, the incoming message variance on the h_k edge is $\mathbf{V}_{k,i}$. At variable node j, the incoming message variance on the h_k edge is $\mathbf{U}_{k,j}$, expressed as:

$$\mathbf{Y}_j = \begin{bmatrix} \sigma^2 & 0 \\ 0 & \sigma^2 \end{bmatrix}, \tag{A.1}$$

$$\mathbf{V}_{k,i} = \begin{bmatrix} v_{k,i} & 0 \\ 0 & v_{k,i} \end{bmatrix}, \tag{A.2}$$

$$\mathbf{U}_{k,j} = \begin{bmatrix} u_{k,j} & 0 \\ 0 & u_{k,j} \end{bmatrix}. \tag{A.3}$$

At check node i, only the h_1 output from the check node contributes to the variance convergence proof. At check node i the variance has zero covariance:

$$\mathbf{U}_{1,i} = \sum_{k=2}^{d_i} [\frac{h'_{k,i}}{h'_{1,i}}]\mathbf{V}_{k,i}[\frac{h'_{k,i}}{h'_{1,i}}]^T \tag{A.4}$$

$$= \begin{bmatrix} \sum_{k=2}^{d} \left(Re\{\frac{h'_{k,i}}{h'_{1,i}}\}^2 + Im\{\frac{h'_{k,i}}{h'_{1,i}}\}^2 \right) v_{k,i} & 0 \\ 0 & \sum_{k=2}^{d} \left(Re\{\frac{h'_{k,i}}{h'_{1,i}}\}^2 + Im\{\frac{h'_{k,i}}{h'_{1,i}}\}^2 \right) v_{k,i} \end{bmatrix}. \tag{A.5}$$

This can be written as

$$u_{1,i} = \frac{1}{\left(\sqrt{{h'_{1,i,Re}}^2 + {h'_{1,i,Im}}^2}\right)^2} \sum_{k=2}^{d} \left(\sqrt{{h'_{k,i,Re}}^2 + {h'_{k,i,Im}}^2}\right)^2, \tag{A.6}$$

$$v_{k,i} = \frac{1}{|h'_{1,i}|^2} \sum_{k=2}^{d} |h'_{k,i}|^2 v_{k,i}. \tag{A.7}$$

Here and following, $h'_{k,i}$ is the coefficient in row i that has label h_k and $h''_{k,j}$ is the coefficient in column j that has label h_k.

Recall that for the relaxed Latin square design, at check node i, $|h'_{1,i}| \geq |h'_{2,i}| \geq |h'_{t,i}|$ for $t = 3, \ldots, d_i$, and at variable node j, $|h''_{1,j}| \geq |h''_{2,j}| \geq |h''_{k,j}|$ for $k = 3, \ldots, d_j$. Using the above check node rule, it can be shown that, for $t = 3, \ldots, d$, if the check node inputs satisfy $v_{1,i} \geq v_{2,i} \geq v_{t,i}$, then the check node outputs satisfy $u_{1,i} \leq u_{2,i} \leq u_{t,i}$, for all check nodes i. Using this, $u_{1,i}$ as in (A.5) may be bounded by:

$$u_{1,i} \leq \alpha_i v_{2,i}. \tag{A.8}$$

At variable node j, the output $\mathbf{V}_{t,j}$ is:

$$\mathbf{V}_{t,j} = \left(\sum_{t'=1\backslash t}^{d} \mathbf{U}_{t,j}^{-1} + \mathbf{Y}_j^{-1} \right)^{-1}, \tag{A.9}$$

where $\mathbf{V}_{t,j}$ has zero covariance in the form of (A.3) and the variance $v_{t,j}$ is:

$$\frac{1}{v_{t,j}} = \sum_{t'=1\backslash t}^{d-1} \frac{1}{u_{t',j}} + \frac{1}{\sigma^2}, \tag{A.10}$$

from which $v_{t,j} \leq u_{t,j}$ for $t \geq 2$ holds. This can also be used to show that if degree-d_j variable node j has inputs satisfing $u_{1,j} \leq u_{2,j} \leq u_{t,k}$, for $t = 3, \ldots, d_j$, then variable node outputs satisfy $v_{1,j} \geq v_{2,j} \geq v_{t,j}$. Since initially $v_{t,i} = \sigma^2$, then by induction $v_{1,i} \geq v_{2,i} \geq v_{t,i}$ holds generally.

The above can be used to show that $v_{2,i'} \leq \alpha_i v_{2,i}$, where i is the check node on iteration ℓ and i' is the check node on iteration $\ell + 1$. Let $\alpha_{\max}$ be the maximum of $\alpha_1, \alpha_2, \ldots, \alpha_n$. Then:

$$v_{t,i} \leq \alpha_{\max}^{\ell} \sigma^2 \tag{A.11}$$

for $t = 2, 3, \ldots, d_i$, where the initial condition of $v_{t,i} = \sigma^2$ and $v_{2,j} \geq v_{t,j}$ was used.

Appendix B

Proof of Proposition 2

The proof of Proposition 2 is given. The check-to-variable messages $R(\mathbf{z})$ and $\widetilde{R}(\mathbf{z})$ have mean $\mathbf{m}_c$, variance $\mathbf{V}_c$, where the k-Gaussian mixture $R(\mathbf{z})$ expressed as:

$$R(\mathbf{z}) = \sum_{i=0}^{k-1} \mathcal{N}(\mathbf{z}; \mathbf{m}_c + \mathbf{b}_i/h, \mathbf{V}_c) \tag{B.1}$$

and the l-Gaussian approximation is $\widetilde{R}(\mathbf{z})$, expressed as:

$$\widetilde{R}(\mathbf{z}) = \sum_{j=0}^{l-1} \mathcal{N}(\mathbf{z}; \mathbf{m}_c + \mathbf{b}_j/h, \mathbf{V}_c). \tag{B.2}$$

The channel message has mean $\mathbf{m}_a$, variance $\mathbf{V}_a$, expressed as $Y(\mathbf{z}) = \mathcal{N}(\mathbf{z}; \mathbf{m}_a, \mathbf{V}_a)$. And it is assumed that $\mathbf{V}_c$ and $\mathbf{V}_a$ are diagonal covariance matrices.

Define f and g as $Y(\mathbf{z})R(\mathbf{z})$ and $Y(\mathbf{z})\widetilde{R}(\mathbf{z})$, respectively, and these products are:

$$f = Y(\mathbf{z})R(\mathbf{z}) = \sum_{i=0}^{k-1} \pi_i \underbrace{\mathcal{N}(\mathbf{z}; \mathbf{m}_i, \mathbf{V})}_{f_i}, \text{ and} \tag{B.3}$$

$$g = Y(\mathbf{z})\widetilde{R}(\mathbf{z}) = \sum_{j=0}^{l-1} \omega_j \underbrace{\mathcal{N}(\mathbf{z}; \mathbf{m}_j, \mathbf{V})}_{g_j}, \tag{B.4}$$

where $\mathbf{V}, \mathbf{m}_i, \pi_i$ and ω_j are:

$$\mathbf{V} = (\mathbf{V}_c^{-1} + \mathbf{V}_a^{-1})^{-1}, \tag{B.5}$$

$$\mathbf{m}_i = \mathbf{V}(\mathbf{V}_c^{-1}(\mathbf{m}_c + \mathbf{b}_i/h)), \tag{B.6}$$

$$\pi_i' = \frac{1}{2\pi\sqrt{|\mathbf{V}_c + \mathbf{V}_a|}} e^{-\frac{1}{2}(\mathbf{m}_c + \mathbf{b}_i/h)^T(\mathbf{V}_c+\mathbf{V}_a)^{-1}(\mathbf{m}_c+\mathbf{b}_i/h)}, \tag{B.7}$$

$$\pi_i = \frac{\pi_i'}{\sum_{i=0}^{k-1} \pi_i'}, \tag{B.8}$$

$$\omega_j = \frac{\pi_j'}{\sum_{j=0}^{l-1} \pi_j'}. \tag{B.9}$$

Without loss of generality, $\pi_0' \geq \pi_1' \geq \cdots \geq \pi_{k-1}'$ and $\mathbf{m}_a = 0$ is assumed, which implies $|\mathbf{m}_0| \leq |\mathbf{m}_1| \leq \cdots \leq |\mathbf{m}_{k-1}|$.

Then, the KL divergence between $Y(\mathbf{z})R_k(\mathbf{z})$ and $Y(\mathbf{z})\widetilde{R}(\mathbf{z})$ can be expressed as:

$$D(Y(\mathbf{z})R(\mathbf{z})||Y(\mathbf{z})\widetilde{R}(\mathbf{z})) = D(\sum_{i=0}^{k-1} \pi_i f_i || \sum_{j=0}^{l-1} \omega_j g_j). \tag{B.10}$$

This can be upper bounded using the variational upper bound on KL divergence [43]:

$$D(Y(\mathbf{z})R(\mathbf{z})||Y(\mathbf{z})\widetilde{R}(\mathbf{z})) \leq \sum_{i=0}^{k-1} \sum_{j=0}^{l-1} \pi_i \omega_j D(f_i||g_j). \tag{B.11}$$

In general, the KL divergence between two complex Gaussian functions $\hat{f}$ and $\hat{g}$ has a closed formed expression,

$$D(\hat{f}||\hat{g}) = \frac{1}{2}\left(\log\frac{|\mathbf{V}_{\hat{g}}|}{|\mathbf{V}_{\hat{f}}|} + \mathrm{Tr}[\mathbf{V}_{\hat{g}}^{-1}\mathbf{V}_{\hat{f}}] - 2 + (\mathbf{m}_{\hat{f}} - \mathbf{m}_{\hat{g}})^T\mathbf{V}_{\hat{g}}^{-1}(\mathbf{m}_{\hat{f}} - \mathbf{m}_{\hat{g}})\right). \tag{B.12}$$

Here, f and g have the same variance $\mathbf{V}$. Then, $\mathbf{V}$ and $\mathbf{m}$ in (B.5) and (B.6) are substituted into (B.12), so that:

$$\pi_i\omega_j D(f_i||g_j) = \begin{cases} 0, & \text{if } i = j \\ \pi_i\omega_j\frac{v_a|\mathbf{m}_i-\mathbf{m}_j|^2}{2(v_c+v_a)}, & \text{if} i \neq j \end{cases}. \tag{B.13}$$

Equation (B.11) has $l(k-1)$ non-zero terms, and $\pi_0\omega_1 D(f_0||g_1) = \pi_1\omega_0 D(f_1||g_0)$ are equal and the greatest, so the term $\pi_0\omega_1$ determines the upper bound.

For single Gaussian approximation $\omega_0 = 1$, and π_1 is given by:

$$\pi_1 = \frac{\pi_1'}{\sum_{i=0}^{k-1}\pi_i'} \tag{B.14}$$

$$= \left(\frac{e^{-\frac{|\mathbf{m}_c+\mathbf{b}_1/h|^2}{2(v_c+v_a)}}}{\sum_{i=0}^{k-1} e^{-\frac{|\mathbf{m}_c+\mathbf{b}_i/h|^2}{2(v_c+v_a)}}}\right), \tag{B.15}$$

Now make the restriction to $k = 2$

$$\pi_1 \leq \left(\frac{e^{-\frac{|\mathbf{m}_c+\mathbf{b}_1/h|^2}{2(v_c+v_a)}}}{e^{-\frac{|\mathbf{m}_c+\mathbf{b}_0/h|^2}{2(v_c+v_a)}} + e^{-\frac{|\mathbf{m}_c+\mathbf{b}_1/h|^2}{2(v_c+v_a)}}}\right) \tag{B.16}$$

$$= \frac{1}{1 + e^{\frac{-(\mathbf{m}_c h+\mathbf{m}_c^* h^*)+1}{2|h|^2(v_c+v_a)}}} \tag{B.17}$$

$$= \frac{1}{1+e^{\frac{-2(m_{c,Re}h_{Re}-m_{c,Im}h_{Im})+1}{2|h|^2(v_c+v_a)}}} \tag{B.18}$$

where $\mathbf{b}_0 = 0$ and $\mathbf{b}_1$ is any integer at the minimum Euclidean distance of 1, $|\mathbf{b}_1| = 1$, so without loss of generality, take $\mathbf{b}_1 = -1$. Then, (B.18) is substituted into (B.13), where $|\mathbf{m}_i - \mathbf{m}_j|^2$ is upper bounded by $|\mathbf{m}_i - \mathbf{m}_j|^2 \leq 1/|h|^2$. Form the upper bound from (B.11) by replacing the $(k-1)$ non-zero terms with the greatest term, so the upper bound of single Gaussian approximation is:

$$D(Y(\mathbf{z})R(\mathbf{z})||Y(\mathbf{z})\widetilde{R}(\mathbf{z})) \leq \frac{(k-1)v_a}{2|h|^2(v_c+v_a)|(1+e^{\frac{2(m_{c,Re}h_{Re}-m_{c,Im}h_{Im})+1}{2|h|^2(v_c+v_a)}})}. \tag{B.19}$$

For two Gaussian approximation, the term $\pi_0\omega_1$ can be written as:

$$\pi_0\omega_1 = \frac{\pi'_0}{\sum_{i=0}^{k-1}\pi'_i}\frac{\pi'_1}{\sum_{j=0}^{l-1}\pi'_j} \tag{B.20}$$

$$= \Big(\frac{e^{-\frac{|\mathbf{m}_c+\mathbf{b}_0/h|^2}{2(v_c+v_a)}}}{\sum_{i=0}^{k-1} e^{-\frac{|\mathbf{m}_c+\mathbf{b}_i/h|^2}{2(v_c+v_a)}}}\Big)\Big(\frac{e^{-\frac{|\mathbf{m}_c+\mathbf{b}_1/h|^2}{2(v_c+v_a)}}}{\sum_{i=0}^{l-1} e^{-\frac{|\mathbf{m}_c+\mathbf{b}_i/h|^2}{2(v_c+v_a)}}}\Big). \tag{B.21}$$

$$\pi_0\omega_1 \leq \left(\frac{e^{-\frac{|\mathbf{m}_c+\mathbf{b}_0/h|^2}{2(v_c+v_a)}}}{\sum_{i=0}^{l-1} e^{-\frac{|\mathbf{m}_c+\mathbf{b}_i/h|^2}{2(v_c+v_a)}}}\right)^2. \tag{B.22}$$

Make the restriction to $l = 2$

$$\pi_0\omega_1 \leq \left(\frac{e^{-\frac{|\mathbf{m}_c+\mathbf{b}_0/h|^2}{2(v_c+v_a)}}}{e^{-\frac{|\mathbf{m}_c+\mathbf{b}_0/h|^2}{2(v_c+v_a)}} + e^{-\frac{|\mathbf{m}_c+\mathbf{b}_1/h|^2}{2(v_c+v_a)}}}\right)^2 \tag{B.23}$$

$$= \left(\frac{1}{1+e^{\frac{2(m_{c,Re}h_{Re}-m_{c,Im}h_{Im})-1}{2|h|^2(v_c+v_a)}}}\right)^2, \tag{B.24}$$

where $\mathbf{b}_0 = 0$ and , $\mathbf{b}_1 = -1$. Then, (B.24) is substituted into (B.13), where $|\mathbf{m}_i - \mathbf{m}_j|^2 \leq 1/|h|^2$. Form the upper bound from (B.11) by replacing the $2(k-1)$ non-zero terms with the greatest term, so the upper bound of two Gaussian approximation is:

$$D(Y(\mathbf{z})R(\mathbf{z})||Y(\mathbf{z})\widetilde{R}(\mathbf{z})) \leq \frac{(k-1)v_a}{|h|^2(v_c+v_a)(1+e^{\frac{2(m_{c,Re}h_{Re}-m_{c,Im}h_{Im})-1}{2|h|^2(v_c+v_a)}})^2}. \quad \text{(B.25)}$$

Bibliography

[1] R. Zamir, *Lattice Coding for Signals and Networks: A Structured Coding Approach to Quantization, Modulation, and Multiuser Information Theory.* Cambridge University Press, 2014.

[2] R. Urbanke and B. Rimoldi, "Lattice codes can achieve capacity on the AWGN channel," *IEEE Transactions on Information Theory*, vol. 44, no. 1, pp. 273–278, Jan 1998.

[3] U. Erez and R. Zamir, "Achieving 1/2 log (1+SNR) on the AWGN channel with lattice encoding and decoding," *IEEE Transactions on Information Theory*, vol. 50, no. 10, pp. 2293–2314, Oct 2004.

[4] Y. Tian, D. Wu, C. Yang, and A. F. Molisch, "Asymmetric two-way relay with doubly nested lattice codes," *IEEE Transactions on Wireless Communications*, vol. 11, no. 2, pp. 694–702, February 2012.

[5] Y. Huang, N. E. Tunali, and K. R. Narayanan, "A compute-and-forward scheme for Gaussian bi-directional relaying with inter-symbol interference," *IEEE Transactions on Communications*, vol. 61, no. 3, pp. 1011–1019, March 2013.

[6] D. Fang, Y. Huang, Z. Ding, G. Geraci, S. Shieh, and H. Claussen, "Lattice partition multiple access: A new method of downlink non-orthogon multiuser transmissions," in *2016 IEEE Global Communications Conference (GLOBECOM)*, Dec 2016, pp. 1–6.

[7] A. Hindy and A. Nosratinia, "Lattice coding and decoding for multiple-antenna ergodic fading channels," *IEEE Transactions on Communications*, vol. 65, no. 5, pp. 1873–1885, May 2017.

[8] B. Kurkoski and J. Dauwels, "Reduced-memory decoding of low-density lattice codes," *IEEE Communications Letters*, vol. 14, no. 7, pp. 659–661, July 2010.

[9] L. Liu, Y. Yan, C. Ling, and X. Wu, "Construction of capacity-achieving lattice codes: Polar lattices," *IEEE Transactions on Communications*, vol. 67, no. 2, pp. 915–928, 2018.

[10] G. Falciasecca, "Marconi's early experiments in wireless telegraphy, 1895," *IEEE Antennas and Propagation Magazine*, vol. 52, no. 6, pp. 220–221, 2010.

[11] T. Farley, "Mobile telephone history," *Telektronikk*, vol. 101, no. 3/4, p. 22, 2005.

[12] F. Hillebrand, "The creation of standards for global mobile communication: Gsm and umts standardization from 1982 to 2000," *IEEE Wireless Communications*, vol. 20, no. 5, pp. 24–33, 2013.

[13] S. Ahmadi, *5G NR: Architecture, technology, implementation, and operation of 3GPP new radio standards.* Academic Press, 2019.

[14] Y. Wu, S. Singh, T. Taleb, A. Roy, H. S. Dhillon, M. R. Kanagarathinam, and A. De, *6G mobile wireless networks.* Springer, 2021.

[15] M. El Soussi, A. Zaidi, and L. Vandendorpe, "Compute-and-forward on a multi-user multi-relay channel," *IEEE Wireless Communications Letters*, vol. 3, no. 6, pp. 589–592, 2014.

[16] Q. Huang and A. Burr, "Compute-and-forward in cell-free massive mimo: Great performance with low backhaul load," in *2017 IEEE International Conference on Communications Workshops (ICC Workshops).* IEEE, 2017, pp. 601–606.

[17] B. Nazer and M. Gastpar, “Compute-and-forward: Harnessing interference through structured codes,” *IEEE Transactions on Information Theory*, vol. 57, no. 10, pp. 6463–6486, 2011.

[18] N. E. Tunali, Y.-C. Huang, J. J. Boutros, and K. R. Narayanan, “Lattices over eisenstein integers for compute-and-forward,” *IEEE Transactions on Information Theory*, vol. 61, no. 10, pp. 5306–5321, 2015.

[19] N. Sommer, M. Feder, and O. Shalvi, “Low-density lattice codes,” *IEEE Transactions on Information Theory*, vol. 54, no. 4, pp. 1561–1585, April 2008.

[20] Y. Yona and M. Feder, “Complex low density lattice codes,” in *2010 IEEE International Symposium on Information Theory.*

[21] J. Zhu and M. Gastpar, “Gaussian multiple access via compute-and-forward,” *IEEE Transactions on Information Theory*, vol. 63, no. 5, pp. 2678–2695, May 2017.

[22] F. R. Kschischang, B. J. Frey, and H. . Loeliger, “Factor graphs and the sum-product algorithm,” *IEEE Transactions on Information Theory*, vol. 47, no. 2, pp. 498–519, Feb 2001.

[23] B. Kurkoski and J. Dauwels, “Message-passing decoding of lattices using Gaussian mixtures,” in *2008 IEEE International Symposium on Information Theory*, July 2008, pp. 2489–2493.

[24] Y. Yona and M. Feder, “Efficient parametric decoder of low density lattice codes,” in *2009 IEEE International Symposium on Information Theory*, June 2009, pp. 744–748.

[25] R. A. Parrao Hernandez and B. M. Kurkoski, “The three/two Gaussian parametric LDLC lattice decoding algorithm and its analysis,” *IEEE Transactions on Communications*, vol. 64, no. 9, pp. 3624–3633, Sep. 2016.

[26] S. Liu, Y. Hong, E. Viterbo, A. Marelli, and R. Micheloni, “Efficient decoding of low density lattice codes,” *IEEE Wireless Communications Letters*, vol. 8,

no. 4, pp. 1195–1199, Aug 2019.

[27] X. Wang and W. H. Mow, “Efficient LDLC decoder design from the lattice viewpoint.” IEEE, 2019.

[28] J. Zhang and M. P. Fossorier, “Shuffled iterative decoding,” *IEEE Transactions on Communications*, vol. 53, no. 2, pp. 209–213, 2005.

[29] J. H. Conway and N. J. A. Sloane, *Sphere packings, lattices and groups*. Springer Science & Business Media, 2013, vol. 290.

[30] Q. T. Sun, J. Yuan, T. Huang, and K. W. Shum, “Lattice network codes based on Eisenstein integers,” *IEEE transactions on communications*, vol. 61, no. 7, pp. 2713–2725, 2013.

[31] G. Poltyrev, “On coding without restrictions for the awgn channel,” *IEEE Transactions on Information Theory*, vol. 40, no. 2, pp. 409–417, 1994.

[32] G. Poltyrev, “On coding without restrictions for the AWGN channel,” *IEEE Transactions on Information Theory*, vol. 40, no. 2, pp. 409–417, March 1994.

[33] J. Pearl, “Reverend bayes on inference engines: A distributed hierarchical approach,” in *Probabilistic and Causal Inference: The Works of Judea Pearl*, 2022, pp. 129–138.

[34] Y. Wang, A. Burr, and D. Fang, “Complex low density lattice codes to physical layer network coding,” in *2015 IEEE International Conference on Communications (ICC)*, June 2015, pp. 2060–2065.

[35] E. Eleftheriou and S. Olcer, “Low-density parity-check codes for multilevel modulation,” in *Proceedings IEEE International Symposium on Information Theory,*. IEEE, 2002, p. 442.

[36] S. Myung, K. Yang, and J. Kim, “Quasi-cyclic ldpc codes for fast encoding,” *IEEE Transactions on Information Theory*, vol. 51, no. 8, pp. 2894–2901, 2005.

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

[38] M. Sipser and D. A. Spielman, "Expander codes," *IEEE transactions on Information Theory*, vol. 42, no. 6, pp. 1710–1722, 1996.

[39] D. J. MacKay, "Good error-correcting codes based on very sparse matrices," *IEEE transactions on Information Theory*, vol. 45, no. 2, pp. 399–431, 1999.

[40] R. A. P. Hernandez and B. M. Kurkoski, "Low complexity construction of low density lattice codes based on array codes," in *2014 International Symposium on Information Theory and its Applications.* IEEE, 2014, pp. 264–268.

[41] C. E. Rasmussen and C. K. I. Williams, *Gaussian processes for machine learning.* The MIT Press, 2005.

[42] C. Nafornita, Y. Berthoumieu, I. Nafornita, and A. Isar, "Kullback-Leibler distance between complex generalized Gaussian distributions," in *2012 Proceedings of the 20th European Signal Processing Conference (EUSIPCO)*, Aug 2012, pp. 1850–1854.

[43] J. R. Hershey and P. A. Olsen, "Approximating the Kullback-Leibler divergence between Gaussian mixture models," in *2007 IEEE International Conference on Acoustics, Speech and Signal Processing - ICASSP '07*, vol. 4, 2007, pp. IV–317–IV–320.

[44] J. J. Boutros, A. Alloum, and G. I. Shamir, "Enhanced channel decoding via em source-channel estimation," in *Proc. 2nd International Symposium Commun., Control, Signal Processing*, 2006.

[45] E. Sharon and A. Bazarsky, "Dynamic memory error model estimation for read and ecc adaptations," in *Proc. Non-Volatile Memories Workshop*, 2017. [Online]. Available: http://nvmw.eng.ucsd.edu/2017/program

[46] Z. Mei, K. Cai, and X. He, "Deep learning-aided dynamic read thresholds design for multi-level-cell flash memories," *CoRR*, vol. abs/1907.03938, 2019. [Online]. Available: http://arxiv.org/abs/1907.03938

[47] C. M. Bishop, *Pattern recognition and machine learning.* Springer Science+ Business Media, 2006.

[48] Xiao-Yu Hu, E. Eleftheriou, and D. M. Arnold, “Regular and irregular progressive edge-growth tanner graphs,” *IEEE Transactions on Information Theory*, vol. 51, no. 1, pp. 386–398, Jan 2005.

Publications

1. W. Wiriya and B. M. Kurkoski, “Reliability-based decoding of low-density lattice codes using Gaussian and Eisenstein integers,” 2023. Submitted to IEEE Access.

2. W. Wiriya and B. M. Kurkoski, “Reliability-based decoding of complex low-density lattice codes,” in Proceedings of the IEEE Information Theory Workshop, (Kanazawa, Japan), October 2021.

3. W., Wiriya, and B. M. Kurkoski. ”EM algorithm for DMC channel estimation in NAND flash memory.” In Proc. Non-Volatile Memories Workshop. 2020.

4. W. Wiriya and B. M. Kurkoski, “Reliability-based parametric LDLC decoding,” in 2018 International Symposium on Information Theory and Its Applications, (Singapore), pp. 188-192, October 2018

5. W. Wiriya and B. M. Kurkoski, “Shuffled belief propagation parametric LDLC decoding,” in Proceedings of the 2018 IEICE Society Conference, (Kanazawa, Japan), September 2018. Presented on 12 September 2018.