Doctoral Dissertation

Construction and Decoding of Low Density Lattice Codes

Ricardo Antonio PARRAO HERNANDEZ

Supervisor: Associate Professor Brian M. KURKOSKI

School of Information Science
Japan Advanced Institute of Science and Technology

December, 2016

Reviewed by
Professor Tadashi Matsumoto
Professor Le Minh Nguyen
Professor Emanuele Viterbo
Professor Tadashi Wadayama

Abstract

Modern information and communication systems are based on the reliable and efficient transmission of information. For practical applications the coding scheme used by the transmission system not only needs to have good coding characteristics, but also needs to be efficiently implemented.

Different efficient coding schemes exist for q-ary fields, but in the real world communications the noise model is usually not in a q-ary field, instead the exist in the real domain. A coding scheme that can exploit the real algebra of the channel is a more natural approach for data transmission.

Lattice codes have potential to become an efficient and practical coding scheme for the AWGN channel and particularly for multi-terminal Gaussian networks because the encoder and the channel use the same real algebra.

Recently, a variety of lattices called low density lattices codes (LDLC) have been studied because they can be seen as a Euclidean space code analogue to low density parity check codes (LDPC). Has been reported that LDLC lattices can attain 0.6dB to the unconstraint capacity for dimension $100,000$. In addition, they can be decoded efficiently using iterative decoders. Previous constructions for LDLC lattices, such as the latin square design, are based on high-complexity computer search algorithms to eliminate 4-cycles.

On the other hand, finite fields codes based on array codes have been widely studied, these codes have a deterministic (no pseudorandom) and low computational complexity construction. In addition, a triangular-structured parity check matrix based on array codes can be easily constructed, which add benefits for encoding.

In the iterative LDLC decoder the messages consist of infinite Gaussian mixtures, and for any implementation, the Gaussian mixtures must be approximated. Different authors had introduced various ways to overcome the

Gaussian mixtures approximation, but these methods are not a good approximation and/or have a high computational complexity.

The focus of this dissertation is to describe an efficient construction and iterative decoding algorithm for LDLC lattices. In this dissertation there are two main contributions:

1. The first main contribution is the design of LDLC lattices based on array codes. The proposed lattices are called "array LDLC lattices". The inverse generator matrices for array LDLC lattices can be defined by four parameters, And has the following properties: a 4-cycle free matrix to improve the performance of the belief propagation (BP) decoding, triangular structure to aid encoding and shaping operations, sparseness for low storage and has a deterministic construction, i.e no pseudorandom construction.

   The benefit of the structure of the array LDLC lattices is that the generator matrix can be obtained by doing block matrix inversion. And the generator matrix can be use to derivate a upper bound for the minimum distance. By numerical results the derivate upper bound is a good approximation for most of the array LDLC lattices. In addition having a triangular structure some elements less protected than others. A method to balance the protection of the elements is given. These methods also can be use as a guide for LDLC lattice design.

   Finally, for all cases considered, the array LDLC lattices have a better performance than the latin square construction.

2. The second main contribution is a new parametric LDLC lattice decoding algorithm, the new decoding algorithm is called the "three/two Gaussian parametric decoder", the proposed decoding algorithm approximate the infinite mixture of Gaussian with a finite number of Gaussians either two or three. The major advantage of the proposed LDLC decoding algorithm is a favorable performance-complexity tradeoff as compared to previous parametric decoding algorithms. Another advantage of the proposed algorithm is that it is nearly parameter-free; the only parameter selection of interest is the number of Gaussians in

the approximation, two or three Gaussians. This is in contrast to other LDLC decoders that have algorithmic parameters.

Strengths of the algorithm include its simplicity and suitability for analysis. Analysis is performed by evaluating the Kullback-Leibler divergence between the true messages and the three/two Gaussian approximation. The approximation using three or two Gaussians is more accurate than previously proposed approximations.

Also, noise thresholds for the three/two Gaussian parametric decoding algorithm are presented, the proposed decoder reduces the noise thresholds 0.05dB compared to previous parametric decoders. The numerical results show that for $n = 1,000$ the two-Gaussian approximation is the same as the best known decoding algorithm. But when the dimension is $n = 10,000$, a three-Gaussian approximation is needed. Finally the results presented are use as a guide on how to choose different parameters for LDLC lattice design.

The array LDLC lattices and three/two Gaussian parametric decoding algorithm are a step forward to a more practical algorithms for LDLC lattices.

Keywords: Lattice codes, LDLC lattices, deterministic construction, parametric decoder,

Acknowledgments

Firstly, I would like to express my sincere gratitude to my advisor associate professor Brian M. Kurkoski for the continuous support of my Ph.D study and related research, for his patience, motivation, and immense knowledge.

Besides my advisor. I would like to thank the rest of my thesis committee: Prof. Tadashi Matsumoto, Prof. Le Minh Nguyen, Prof. Emanuele Viterbo and Prof. Tadashi Wadayama for their insightful comments and encouragement, which incented me to widen my research from various perspectives.

I thank my fellow labmates and friends: Jessica Sanchez, Javier Cuadros, Erick Garcia, Fan Zhou, Xiaobo Zhou, Meng Cheng, Penshun Lu, Kun Wu, Sekiya Ryota, Valtteri Tervo, Shen Qian, Weiwei Jiang, Jiajie Xue, Ha Doan, Warangrat Wiriya, Alan Zhang, Thanh Nguyen, Shoh Kato and Tung Nguyen for the stimulating discussions, for the sleepless nights we were working together before deadlines, and for all the funny moments we have had in the last years.

I like to thank the administrative staff Aya Inoue and Tomoko Taniguchi for their help.

Last but not the least, I want to thank my family for all the love an support throughout writing this dissertation and my life in general.

Acronyms

AWGN Additive white Gaussian noise. 5

BP Belief propagation. 2, 31, 36

GMR Gaussian mixture reduction. 9

KL Kullback-Leiber. 76

LDLC Low density lattice codes. 2, 31

LDPC Low density parity check. 2, 31

LLL Lenstra-Lenstra-Lovasz. 17

MM Moment matching. 44, 70

SER Symbol error rate. 62

SNR Signal-to-noise ratio. 5

VNR Volume-to-noise ratio. 5

List of Symbols

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Modern information and communication systems are based on the reliable and efficient transmission of information. Channels encountered in practical applications are usually disturbed regardless whether they are correspond to information transmission over noisy and time-variant wired/wireless channels, or information stored over physical media (e.g. flash memories and optical discs) that might be damaged by scratches. Knowing these disturbances, appropriate coding schemes need to be employed, such that errors during the transmission can be corrected.

Practical applications for channel coding include space and satellite communications, mobile communications, digital video and audio broadcasting, data transmission and data storage. For practical applications the coding scheme not only needs to have good coding characteristics but also needs to be efficiently implementable, e.g. hardware within integrated circuits.

For data transmission, there exist efficient binary coding schemes, such as turbo codes [4] and low density parity check codes (LDPC) [5]. But in real life the noise and the transmission medium are not binary, instead the noise and the medium are functions in the real domain $\mathbb{R}^n$, e.g. Fig 1.1 shows an example of a wireless transmission. Lattice codes are codes over the real numbers which possess great potential to become an efficient, practical and reliable communication scheme for the AWGN channel. Shannon showed that codes with very long random Gaussian-distributed codewords can approach the AWGN capacity [6], and now it is known that lattice codes can also achieve the AWGN capacity [7] [8] [9]. Lattices are especially appealing

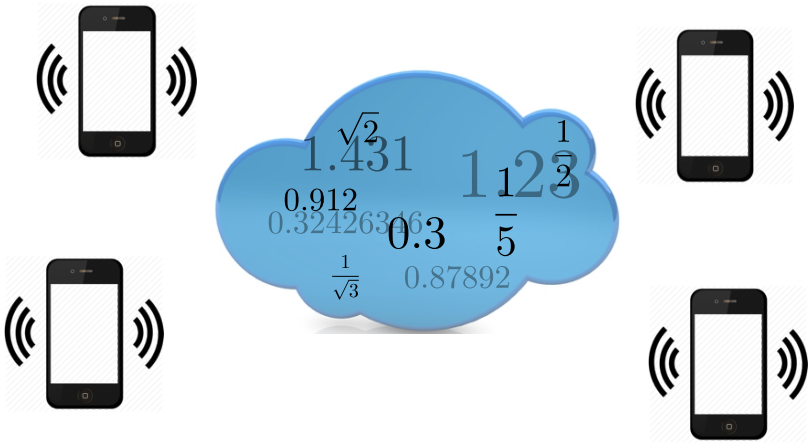

Figure 1.1: In real communications the channel and the medium consist of functions in the real domain.

for multi-terminal Gaussian networks, where the encoder and the channel use the same real algebra.

Recently, a variety of lattices called low density lattice codes (LDLC), introduced by Sommer, Feder and Shalvi [1], have been studied. LDLC lattices are high-dimensional lattices defined by a sparse inverse generator matrix. The construction and decoding of LDLCs resemble low density parity check (LDPC) codes, that is, using a belief propagation (BP) decoding algorithm on a sparse graph. It was reported that the LDLC belief propagation decoder attains a symbol error rate of 10^{-5} at 0.6 dB from the unconstrained AWGN channel capacity. Already, relaying and physical layer network coding schemes that use LDLCs have been described [10][11] [12][13]. But how to construct and efficiently decode LDLCs are an open problem.

1.1 A brief history of lattices

In the seventeenth century the astronomer Johannes Kepler conjectures that the face-centered cubic lattice forms the best sphere packing in three dimensions. And Gauss showed that there is no other lattice better than the face-centers cubic lattice in three dimension.

The first step on the concept of lattice was taken by George Boole when he tried to formalize propositional logic in the style of algebra. In [14] the list of laws which are satisfied by various algebras was presented, e.g. calculus of logic. The rules that Boole set concerned three binary operations on a structure, the first two satisfied associative, commutative and distributive laws, the third one corresponded to the creation of complements. Richard Dedekind was interested in algebraic number theory, and Dedekind investigated properties of structures called "dual groups.

Lattice structures started to be studied in other areas of mathematics at the end of 1920s. Karl Menger presented the set of axioms characterizing projective geometries [15] which are in fact complemented modular lattices.

Merits in the early developments of lattice theory belong to Garrett Birhoff who also approached it from the side of algebra and united its various applications [16]. G. Birkhoff also introduced the English word "lattice, but was inspired by the image of some Hasse diagrams presenting lattices. The development did not take place only in lattice theory proper or in its connection with algebra, but also in the field of geometry, topology, logic, probability and functional analysis. the first summarising works on lattices started to appear, Birkhoff in [17] presented the notions of lattice theory.

The mathematician Hermann Minkowski used lattices to related n-dimensional geometry with number theory. The Minkowski-Hlawka theorem play the role of Shannon's random coding technique for providing the existence of "good" lattice codes.

The relation between error-correcting codes, sphere packing and lattices was studied by Leech and Sloane [18], and Conway and Sloane [19] in 1970's and 1980's. In a series of works through the 1980's and 1990's, Forney [20][21] [22]established tools to characterise and evaluate lattice codes, towards their

implementation in digital communications.

In the 802.11 and LTE standards the set of possible coded signals corresponds to a finite segment from some high dimensional lattices. The ITU-T standard V.34 for voice band telephone channel modems at 33.6 kbits per second uses a four-dimensional constellation selected form D_4 lattice. Lattice codebooks are also used for data compression; one recommendation for the ITU-T 729.1 speech-coding standardised the Gosset lattice E_8 as the codebook for code-excited linear prediction(CELP)[23].

In 1996 Ajtai shows a remarkable worst-case to average-case reduction for lattice problems, yielding to lattice-based cryptography [24]. A crypto systems based on lattices is the NTRU algorithm [25].

Erez and Zamir showed in [8] that lattice codes can achieve the AWGN channel capacity in 2004. In 2006 Sommer, Erez and Shalvi introduced low density lattice codes (LDLC)[26], that can achieve 0.6dB from the lattice capacity. In [1] the construction and a decoding algorithm was given in detail. LDLC lattices resemble LDPC codes.

In 2006 Nazer and Gatspar introduced a scheme for the physical layer network, called compute-and-forward[10], that allows intermediate nodes to both recover linear combinations of codewords and eliminate noise. This reduces a network into a set of reliable linear equations. The compute-and-forward strategy can be done efficiently if the code is a lattice code. This technique makes lattice codes very attractive for the physical layer network problem.

1.2 System model

In Fig 1.2 the basic structure of the communication model is shown. The transmitter performs two operations, source coding and channel coding, and the receiver performs the inverse of the transmitter operations. The channel code is used such that the receiver is able to detect and/or correct errors.

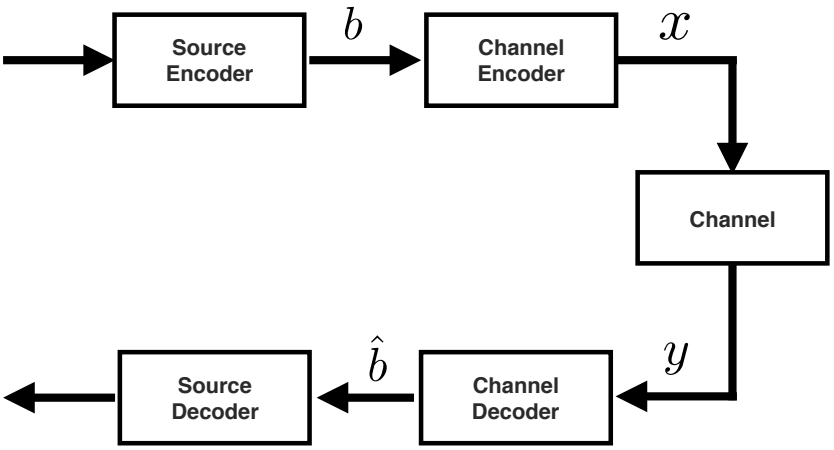

Figure 1.2: A basic communication model.

1.2.1 AWGN channel

This work considers the additive white Gaussian noise (AWGN) channel model [27]. In this model the received signal $\mathbf{y}$ is a sum of the transmitted signal $\mathbf{x}$ and white Gaussian noise $\mathbf{z}$, for $z_i \sim \mathcal{N}(0, \sigma^2)$, as:

$$\mathbf{y} = \mathbf{x} + \mathbf{z}, \tag{1.1}$$

where the noise $\mathbf{z}$ is assumed to be independent of the signal $\mathbf{x}$. Without further conditions, the capacity of these channel may be infinite. This dissertation considers the AWGN channel without restrictions [28]. These consideration leads to the volume-to-noise ratio (VNR) [29], analogous to the signal-to-noise ratio (SNR). The VNR is the ratio of the normalized volume of the lattice and the normalized volume of a noise sphere.

$$VNR = \frac{\text{Volume of the lattice}}{\text{Volume of noise sphere}} \tag{1.2}$$

$$VNR = \frac{V(\Lambda)^{2/n}}{2\pi e\sigma^2}. \tag{1.3}$$

where $V(\Lambda)$ is the volume of the lattice and $2\pi e\sigma^2$ is the volume of the n-dimensional noise sphere of squared radius $n\sigma^2$. This dissertation considers lattice capacity when

$$\sigma^2 = \frac{V(\Lambda)^{2/n}}{2\pi e}. \tag{1.4}$$

A lattice that achieves lattice capacity, also achieves the channel capacity of the power constrained AWGN channel, with properly shaping region [8].

Characteristics like fading, interference and other channel properties are not considered in this model. However, this channel model is an accurate mathematical model for evaluating system performances.

1.3 Problems to solve

Lattices have been studied for many years as was described in previous section. Nevertheless there are still open problems to be solved, especially for high dimensional lattices. This dissertation is focused on the study of LDLC lattices, and specifically its main focus is on two problems. The first one is how to construct LDLC lattices, and the second one is how to find the closest lattice point to a given point for LDLC lattices i.e. LDLC lattice decoding.

1.3.1 LDLC construction

The construction process of the LDLC lattices inverse generator matrices in [1] requires high processing time and high amount of memory. The construction algorithm given by Sommer et. al. [1] generates random permutations

in order to search for and eliminate 4-cycles in the matrix and finally assign the values of the non-zero elements in a latin square manner. These permutation matrices need to be stored and as the dimension increases, these operations require high computational complexity. The characteristic of these latin square LDLC lattice inverse generator matrices is that they are 4-cycle free and sparseness.

Having a triangular structure is a desired property for the LDLC lattice inverse generator matrices, in order to aid shaping and encoding operations. In [30] a method to construct triangular inverse generator matrices for LDLC lattices was given. First a latin square LDLC lattice inverse generator matrix was generated as was described in [1], and then by row and column permutations the triangular structure of the inverse generator matrix was generated. This row and column permutations operations are computationally demanding as the dimension increases.

The first problem to solve in this dissertation is how to construct LDLC lattice inverse generator matrices with a low computational complexity algorithm. In addition the LDLC inverse generator matrices need to have a triangular structure, free of 4-cycles and present sparseness.

1.3.2 LDLC decoding algorithm

In the idealized LDLC belief propagation decoder, the messages passed between check and variable nodes are continuous functions. In any implementation, these continuous functions must be approximated. In the original implementation [1], these messages were approximated by a discretely quantized function. The amount of quantization, typically 1024 bins, is impracticably large.

A computationally efficient approach is to represent the messages as a mixture of Gaussian functions. For the AWGN channel, the messages are precisely represented using a mixture containing an infinite number of Gaussians, which is clearly not practical.

Various authors introduced parametric decoder algorithms in order to approximate the infinite mixture of Gaussians with a finite number of Gaus-

sians. These parametric algorithms use tables to find the dominant Gaussians in the mixture [31], or use a single Gaussian approximation for combining pair of Gaussians for reducing the number of Gaussians in the mixture [2][32]. But those operations are not suitable for implementation and/or present a weak approximation.

The second problem this dissertation solves is to construct a parametric decoding algorithm for LDLC lattices, the decoding algorithm need to use a good approximation for the infinite Gaussian mixtures and have low computational complexity.Having a low computational complexity construction and decoding algorithms are a step forward to a more practical algorithms for LDLC lattices.

1.4 Contributions

In this section a description of the contributions of the dissertation is given. The contributions are divided into two sections. The first section corresponds to the contributions for the proposed construction algorithm. The second section summarizes the contributions for the proposed decoding algorithm.

1.4.1 Proposed LDLC lattice construction: "Array LDLC"

The new method to construct LDLC lattices uses the construction of the binary array LDPC codes [33] as base for design the LDLC inverse generator matrix. To be in congruence with the base of the construction, it is called "Array LDLC lattice".

The array LDLC lattice inverse generator matrices have a deterministic construction, i.e. no pseudo-random construction, are 4-cycle free in order to improve the performance of the BP decoding, triangular structure, for aid encoding and shaping operation, and sparse for low storage. The benefits of the array LDLC lattice inverse generator matrices are summarized in Table 1.1.

Table 1.1: Benefits of the array LDLC lattice inverse generator matrices.

Properties	Benefits
Deterministic construction	Low computational complexity construction
4-cycle free	Improve BP-decoding
Sparseness	Low storage
Triangular structure	Simplify encoding and shaping operations

In addition to the new construction, the following contributions are given:

1. Upper bound for the minimum distance.

2. A method to increase the protection to the less protected elements.

3. Guideline for choosing the balance factor.

1.4.2 Proposed LDLC decoder: "Three/Two Gaussian Parametric Decoding Algorithm"

The major advantage of the new LDLC decoding algorithm is a favorable performance-complexity tradeoff as compared to previous parametric decoding algorithms such as the Gaussian mixture reduction (GMR) algorithm [2] and the table search algorithm [31].

The new decoding algorithm is nearly parameter free; the only parameter selection of interest is the number of Gaussians M used in the approximation which is either 2 or 3, accordingly it is called the "three/two Gaussian

Table 1.2: Benefits of the three/two Gaussian parametric decoding algorithm.

Properties	Benefits
Almost parameter-free	one degree of freedom
Message between nodes are single Gaussians	Low storage
Infinite Gaussian mixture are approximated with 2 or 3 Gaussians	Good approximation

parametric decoding algorithm".

Having a nearly parameter free algorithm is an advantage compared to other LDLC decoders that have algorithmic parameters, e.g. threshold parameters and list size of the GMR algorithm. The benefits of the three/two Gaussian parametric decoding algorithm are summarized in Table 1.2.

For small and medium-dimension lattices of dimension $n = 1,000$ or less, the complexity-performance tradeoff is particularly favorable using $M = 2$ Gaussians in the approximation. It is shown that performance is practically indistinguishable from the GMR algorithm [2], but complexity is significantly lower. For higher-dimensional lattices of dimension $n = 10,000$, the two-Gaussian approximation has some performance loss, which is recovered by increasing to an $M = 3$ Gaussian approximation. To maintain a low computational complexity the number of Gaussians used in the approximation should be either $M = 2$ or $M = 3$. Selecting a greater number of Gaussians $M \geq 4$ does not give any significant advantage in performance.

In addition, the noise thresholds for the Three/Two Gaussian parametric decoding algorithm are shown, the result of the noise thresholds can be used as a reference for LDLC lattice designing.

1.5 Organization

Chapter 2 gives the definition and background on lattices, not only those which are used in these dissertation, but general definitions as well, for readers which are interested in the study of lattices. These definitions include the description of some small dimensional lattices, such as A_2, D_3, D_4, E_8 and the Barnes-Wall lattice.

In Chapter 3, the description of low density lattices codes, the conventional construction algorithm and the previous decoding algorithm are described. In Chapter 4 description of LDPC array codes are presented and the new LDLC lattice construction called "array LDLC lattices" are introduced. The proposed LDLC decoding algorithm called "three/two Gaussian parametric decoding algorithm" and its analysis are presented in Chapter 5, and finally the conclusion and future work is presented in Chapter 6.

Chapter 2

Lattices

In this chapter basic definitions for the study of lattices are presented. These basic definitions include the well known A_2, D_3, D_4, E_8 and Barnes-Wall lattices, basic constructions based on binary codes, notions on the minimum distance and the volume-to-noise ratio. This chapter not only includes the lattice knowledge used in this dissertation, but also definitions that are useful for those who want to begin the study of lattices.

2.1 Definition of lattices

A lattice Λ is a periodic linear arrangement of points in the n-dimensional Euclidean space $\mathbb{R}^n$. It can be seen as the set of all integer combinations of basis vectors. A matrix $\mathbf{G}$ whose columns are the basis vectors $\mathbf{v}$, is called the generator matrix of the lattice.

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

A lattice point $\mathbf{x}$ is defined as:

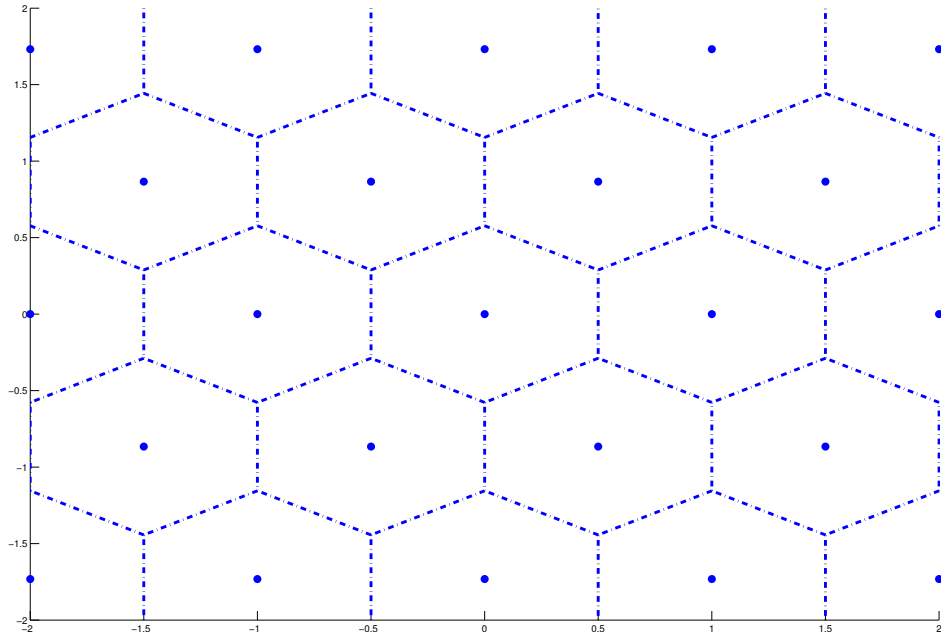

Figure 2.1: The two dimensional hexagonal lattice (A_2 lattice), the blue points are the lattice point, in addition the Voronoi region of the lattice point is also illustrated. Each lattice point of the A_2 lattice has 6 neighbors.

$$\mathbf{x} = \mathbf{G}\mathbf{b}, \tag{2.2}$$

where $\mathbf{x} \in \mathbb{R}^n$ and $\mathbf{b} \in \mathbb{Z}^n$ are column vectors.

In Fig 2.1 can be seen the graphical representation of the 2-dimensional A_2 hexagonal lattice, which can be constructed with

$$\mathbf{G} = \begin{bmatrix} \frac{\sqrt{3}}{2} & 0 \\ \frac{1}{2} & 1 \end{bmatrix}. \tag{2.3}$$

The blue dots represent lattice points and also the Voronoi region is il-

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

$$V(\Lambda) = |\det(\mathbf{G})|. \tag{2.4}$$

Other interesting property of the lattice is the **kissing number**, which is the number of closest neighbors a lattice point has. The **density** of a lattice is the proportion of the space that is occupied by a sphere inside the fundamental region (spheres centered on each lattice point, with radius such spheres just touch). The n-dimensional sphere in $\mathbb{R}^n$ with center $\mathbf{u} = (u_1, \ldots, u_n)$ and radius ρ, consists of all points $\mathbf{x} = (x_1, \ldots, x_n)$ that satisfy:

$$(x_1 - u_1)^2 + \cdots + (x_n - u_n)^2 = \rho^2 \tag{2.5}$$

The density of a lattice is defined by:

$$\frac{\text{volume of one sphere}}{\text{volume of fundamental region}}. \tag{2.6}$$

These two properties are used to analyze the packing properties of the lattice. The packing problem is similar to the quantization problem [34].

Also for lattices, given any vector $\mathbf{y} \in \mathbb{R}^n$, how to estimate the closest lattice point in Λ is a non-trivial problem. For small dimensions low complexity algorithms are known, but as the dimension increases the algorithms to estimate the closest lattice point are not simple. Recall that an advantage of lattice codes is that decoding and quantization are combined as a "single entity"; lattice codes can directly maps digital information into a vector in $\mathbb{R}^n$ and vice versa.

2.1.1 Definition of Nested lattices

A lattice code is all lattice points to lie inside of a constrained region, usually this region is a Voronoi region of a scaled lattice.

A pair of n-dimensional lattices Λ_1 and Λ_2, are called nested if

$$\Lambda_2 \subseteq \Lambda_1, \tag{2.7}$$

Λ_2 is a sublattice of Λ_1 and it is denoted as Λ_1/Λ_2. The generator matrices of Λ_1 and Λ_2 satisfy

$$\mathbf{G}_2 = \mathbf{G}_1 \cdot \mathbf{J}, \tag{2.8}$$

where $\mathbf{J}$ is the $n \times n$ nesting matrix and $\det(\mathbf{J}) \geq 1$, with equality if the two lattices are identical. For nested lattices Λ_1 is called the fine lattice and Λ_2 is called the coarse lattice. The volumes satisfy

$$V(\Lambda_2) = \det\left(\mathbf{J} \cdot V(\Lambda_1)\right) \tag{2.9}$$

In Fig 2.2 a two dimensional nested lattice code $A_2/4A_2$ can be seen, where the blue dots represent the A_2 lattice, the fine lattice, and the red dots represent the coarse lattice $4A_2$. Each coarse lattice contains sixteen points of the fine lattice, and can be represented using four bits.

2.1.2 Minimum distance of a lattice

The squared norm τ^2 of a lattice point $\mathbf{x} \in \Lambda$ is

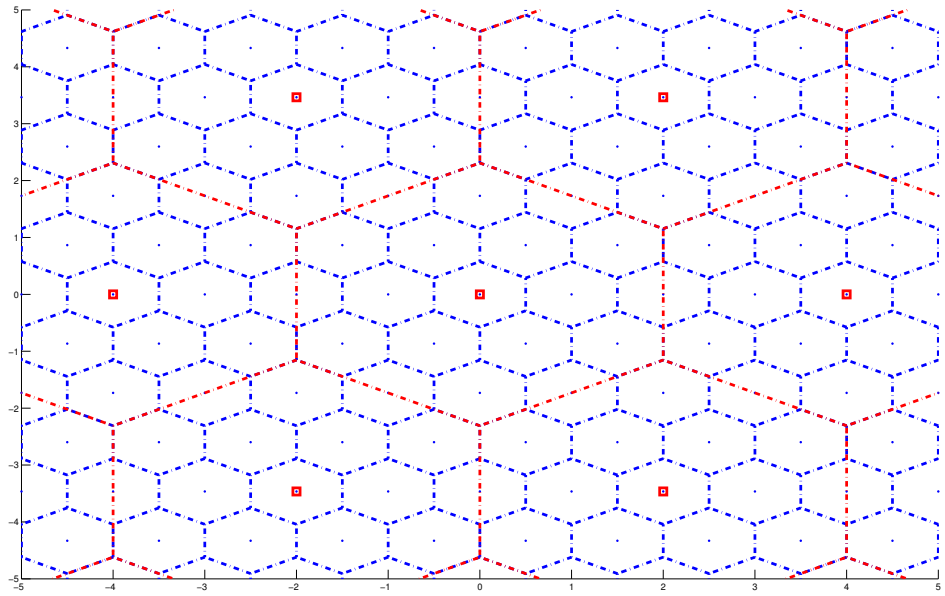

Figure 2.2: A nested lattice representation in two dimension the coarse lattice is $4A_2$ (in red) and the fine lattice is the A_2 lattice (in blue).

$$\tau_{\mathbf{x}}^2 = ||\mathbf{x}||^2 = \sum_{i=1}^{n} x_i^2. \tag{2.10}$$

The squared minimum distance or squared minimal norm of the lattice $d_{min}^2(\Lambda)$ is the minimum square norm among two distinct lattice points, and it is defined as:

$$d_{min}^2(\Lambda) = \min\{||\mathbf{x}-\mathbf{y}||^2 : \mathbf{x}, \mathbf{y} \in \Lambda, \mathbf{x} \neq \mathbf{y}\} \tag{2.11}$$

$$d_{min}^2(\Lambda) = \min_{\mathbf{x}\in\Lambda}\{||\mathbf{x}||^2 : \mathbf{x} \neq 0\}, \tag{2.12}$$

and $d_{min}^2(\Lambda) > 0$.

The squared minimum distance of a lattice is upper bounded by the norm of all basis vectors $\mathbf{v}_i$, for $i = 1, 2, \ldots, n$, in the generator matrix shown in

equation (2.3):

$$d^2_{min}(\Lambda) \leq \min(||\mathbf{v}_1||^2, ||\mathbf{v}_2||^2, \ldots, ||\mathbf{v}_n||^2) \tag{2.13}$$

Finding the minimum distance of lattice is not a trivial problem since it requires an intensive search among all lattice points, and as the dimension increases the number of point also increases.

There exist different techniques to estimate the squared minimum distance of a lattice. One technique is by searching using the sphere decoding algorithm [35], but depending on the parameters can lead to search over great number of points or not find any point(for more information see Section 2.4.3). Other technique is by using the Lenstra-Lenstra-Lovasz (LLL) lattice basis reduction algorithm [36].

The LLL-reduction algorithm is a polynomial time lattice reduction algorithm invented by Arjen Lenstra, Hendrik Lenstra and Laszlo Lovasz in 1982, that changes the basis of $\mathbf{G}$ into a shortest basis $\mathbf{G}'$ such that the lattice remains the same.The LLL-reduction algorithm has the following operations:

1. Swapping 2 vectors of the basis. This swapping only changes the order of vector in the basis, so the lattice is not affected.
2. Replacing $\mathbf{v}_i$ by $-\mathbf{v}_i$.
3. Adding to a vector $\mathbf{v}_i$ a linear combination of the other vector of the basis.

In aAppendix B.1 the pseudocode for the LLL-reduction algorithm is described. Since the squared minimum distance d^2_{min} of the lattice is upper bounded by the square norms of the basis vectors in the generator matrix, the LLL-reduction algorithm gives us a better knowledge of the squared minimum distance.

2.2 Root lattices

The root lattices are the n-dimensional lattices $A_n (n \geq 1)$, $D_n (n \geq 2)$ and $E_8 (n = 8)$. These lattices are the densest known sphere packing and coverings in dimensions $n \leq 8$, for dimension $n = 16$ the best known lattice is the Barnes-Wall lattice. They can be used as the basis for efficient block quantizers for uniformly distributed inputs, and to construct codes for a band-limited channel with Gaussian noise.

2.2.1 A_n lattice

The A_n lattice is defined as:

$$A_n = \{(x_0, x_1, \ldots, x_n) \in \mathbb{Z}^{n+1} : x_0 + x_1 + \cdots + x_n = 0\} \tag{2.14}$$

which uses $n + 1$ coordinates to define an n-dimensional lattice. A_n lies in the hyperplane $\sum x_i = 0$ in $\mathbb{R}^n$.

In two dimensions the densest lattice is the A_2 lattice also called the hexagonal lattice, since the Voronoi cells are hexagons. A possible generator matrix for this lattice is:

$$\mathbf{G} = \begin{bmatrix} \frac{\sqrt{3}}{2} & 0 \\ -\frac{1}{2} & 1 \end{bmatrix}. \tag{2.15}$$

2.2.2 D_n lattice

Th D_n lattice is defined as:

$$D_n = \{(x_1, x_2, \ldots, x_n) \in \mathbb{Z}^n : x_1 + x_2 + \cdots + x_n \text{ is even}\}, \tag{2.16}$$

D_n is obtained by coloring the points of $\mathbb{Z}^n$ alternately black and white, like a checkerboard, and taking the black points as lattice points. Also the D_n lattice is known as the "checkerboard lattice".

In three dimensions the densest lattice is the face-centered cubic lattice A_3 and D_3, and has the shape of the pyramid of oranges. A possible generator matrix for this lattice is:

$$\mathbf{G} = \begin{bmatrix} -1 & 1 & 0 \\ -1 & -1 & 1 \\ 0 & 0 & -1 \end{bmatrix}. \tag{2.17}$$

In four dimensions the most useful lattice is the D_4 lattice. A possible generator matrix for this lattice is:

$$\mathbf{G} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ -1 & 1 & 0 & 0 \\ 0 & -1 & 1 & 0 \\ 0 & 0 & -1 & 2 \end{bmatrix}. \tag{2.18}$$

2.2.3 E_8 lattice

In eight dimensions the most useful lattice is the E_8 lattice. The E_8 lattice consists of the points

$$(x_1, \ldots, x_8) : \text{ all } x_i \in \mathbb{Z} \text{ or all } x_i \in \mathbb{Z} + \frac{1}{2}, \sum x_i \equiv 0 \mod 2, \tag{2.19}$$

A possible generator matrix is:

$$\mathbf{G} = \begin{bmatrix} 2 & -1 & 0 & 0 & 0 & 0 & 0 & \frac{1}{2} \\ 0 & 1 & -1 & 0 & 0 & 0 & 0 & \frac{1}{2} \\ 0 & 0 & 1 & -1 & 0 & 0 & 0 & \frac{1}{2} \\ 0 & 0 & 0 & 1 & -1 & 0 & 0 & \frac{1}{2} \\ 0 & 0 & 0 & 0 & 1 & -1 & 0 & \frac{1}{2} \\ 0 & 0 & 0 & 0 & 0 & 1 & -1 & \frac{1}{2} \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & \frac{1}{2} \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{2} \end{bmatrix}. \tag{2.20}$$

2.2.4 Barnes-Wall lattice

In dimension $n = 16$ the best known lattice is the Barnes-Wall (BW_{16}) lattice, introduced by Barnes and Wall in [37]. This lattice can be generated by applying construction B to the first-order Reed-Muller code of length 16. A possible generator matrix is shown in Fig. 2.3.

In Table 2.1 a comparison between A_2, D_3, D_4, E_8 and BW_{16} is shown, where the kissing number, density, packing radius and minimal norm are given. These values are from Conway and Sloane [19].

2.3 Construction based on linear binary codes

There are lattice constructions that are based on linear binary codes. An (n, k, d) linear binary code C maps k information bits into binary codewords $\mathbf{c}$ of length n. The number of codewords is given by $M = 2^k$, and these codewords are subject to the constraint

$$\mathbf{G} = \frac{1}{\sqrt{2}} \begin{bmatrix} 4 & 2 & 2 & 2 & 2 & 2 & 2 & 2 & 2 & 2 & 2 & 1 & 0 & 0 & 0 & 1 \\ 0 & 2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 1 \\ 0 & 0 & 2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 2 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 2 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 2 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 2 & 0 & 0 & 1 & 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 2 & 0 & 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 2 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 \end{bmatrix}$$

Figure 2.3: Generator matrix for the Barnes-Wall lattice.

Table 2.1: Comparison between different root lattices

	Determinant	Kissing number	Density	Packing radius	Minimal norm
A_2	$\frac{3}{4}$	6	$\frac{\pi}{\sqrt{12}}$	$\frac{1}{2}$	1
D_3	4	12	$\frac{\pi}{\sqrt{18}}$	$\frac{1}{\sqrt{2}}$	2
D_4	4	24	$\frac{\pi^2}{16}$	$\frac{1}{\sqrt{2}}$	2
E_8	1	240	$\frac{\pi^4}{384}$	$\frac{1}{\sqrt{2}}$	2
BW_{16}	254	4320	$\frac{\pi^8}{16 \cdot 8!}$	1	4

$$\mathbf{H} \cdot \mathbf{c}_m = 0 \tag{2.21}$$

for $m = 1, \ldots, M$, where all the additions and multiplications are modulo 2, and $\mathbf{H}$ is the parity check matrix of the code C. The minimum distance of the code C is denoted by d, i.e., the minimum Hamming distance weight

over all non-zero codewords.

2.3.1 Construction A

Construction A is method to construct a lattice by lifting a linear binary code C to the Euclidean space.

The set of all integer vectors whose modulo 2 reductions belongs to C forms a lattice. In other words construction A is defined as:

$$\Lambda_c = \{\mathbf{x} \in \mathbb{Z}^n : \mathbf{x} \bmod 2 \in C\} \text{ or} \tag{2.22}$$

$$\Lambda_c = 2\mathbb{Z}^n + C, \tag{2.23}$$

where $\mathbf{x}$ is a lattice point if and only if $\mathbf{x}$ is congruent modulo 2 to a codeword of C, and the minimum distance of the lattice constructed using construction A, Λ_c, is given by $d_{min}(\Lambda_c) = \min\{\sqrt{d}, 2\}$.

This construction is not only applicable for binary alphabet, but it can be extended to q-ary linear code C and is applied as:

$$\Lambda_c = q\mathbb{Z}^n + C. \tag{2.24}$$

where q is a prime number. And has the following properties:

1. $\det(\Lambda_c) = q^{n-k}$.
2. Λ_c contains a cubic lattice: $q\mathbb{Z}^n \subset \Lambda_c$.
3. If the generator matrix of C has a systematic representation $\mathbf{G}_C = [\mathbf{I}_k|\mathbf{P}^t]^t$, where $\mathbf{I}_k$ is an $k \times k$ identity matrix and $\mathbf{P}$ is a $(n-k) \times k$

matrix. A possible generator matrix for the lattice constructed using construction A is:

$$\mathbf{G}_{\Lambda_c} = \begin{bmatrix} \mathbf{I}_k & | & \mathbf{0} \\ \mathbf{P} & | & q\mathbf{I}_{n-k} \end{bmatrix}. \tag{2.25}$$

The advantage of construction A is that there is a series of similarities between properties of the codes and that of the resulting lattices, e.g. linking the dual of the code and the dual of the lattice, or the weight enumerator of the code and the theta series of the lattice [19]. The well known lattice E_8 can be constructed by applying construction A to the (8,4,4) extended Hamming code.

2.3.2 Construction D and D'

Construction D and D' use a nested family of binary codes to produce a lattice Λ_D or $\Lambda_{D'}$ [38][39]. The difference between construction D and D' is that, construction D works with a set of generator matrices and construction D' works with the set of parity-check matrices. In this dissertation only construction D' is described, and it is as follows:

Let $\alpha = 1$ or 2, and let $C_0 \subseteq C_1 \subseteq \cdots \subseteq C_a$ be a family of binary linear codes,where has parameters $[n, k_i, d_i]$ and the minimum distance satisfies $d_i \geq \alpha 4^i$ for $i = 0, \ldots, a$. Let $\mathbf{h}_1, \ldots, \mathbf{h}_n$ be linear independent binary vectors such that for $i = 1, \ldots, a$, the code C_i is defined by the $r_i = n - k_i$ parity-check vectors $\mathbf{h}_1, \ldots, \mathbf{h}_{r_i}$, and let $r_{-1} = 0$. Consider the vector $\mathbf{h}_j$ as integral vector in $\mathbb{R}^n$. The new lattice $\Lambda_{D'}$ consists of those $\mathbf{x} \in \mathbb{Z}^n$ that satisfy

$$\mathbf{h}_j \cdot \mathbf{x} \equiv 0 (\mathrm{mod} 2^{i+1}) \tag{2.26}$$

for all $i = 0, \ldots, a$ and $r_{a-i-1} + 1 \leq j \leq r_{a-1}$. Then minimum distance $d_{\Lambda_{D'}}$ satisfies

$$\min\{d_a, 4d_{a-1}, 4^2 d_{a-2}, \ldots, 4^a d_0, 4^{a+1}\} \leq d^2_{\Lambda'_D} \leq 4^{a+1}. \tag{2.27}$$

The kissing number γ satisfies[40]:

$$\gamma \leq 2n + \sum_{1 \leq i \leq a} 2^{4^i} A_{d_i} \tag{2.28}$$

where A_{d_i} is the number of codewords in C_i with minimum weight d_i.

Construction D and D' have been used to construct lattices based on low density parity check (LDPC) codes and these lattices are called LDPC lattices [41], or those based on turbo codes and are called turbo lattices [40], and more recently those which are based on polar codes, are called polar lattices [42]. There exist other lattice constructions, in the following chapter a construction for low density lattice codes is introduced, which is the main lattice code of this work.

2.4 Finding the closest lattice point

An important problem is finding the closest lattice point $\hat{\mathbf{x}} \in \Lambda$ to a given point $\mathbf{y} \in \mathbb{R}^n$ such that the Euclidean distance between $\mathbf{y}$ and $\hat{\mathbf{x}}$ is minimized. If the lattice is used as a code for the AWGN channel, then finding the closest lattice point corresponds to maximum likelihood decoding [43]. If the lattice is used for quantization then the closest lattice point problem corresponds to the minimum distortion point.

In 1981 Boas showed that finding the closest lattice point problem is NP-hard [44], and no polynomial-time algorithm is known with a performance

ratio better than exponential in terms of the lattice dimension. Different authors had proposed algorithms that want to solve this problem, and for some specific lattices there exist low complexity algorithms which are attractive for practical applications. In this section some of the well known decoding algorithms are shown for different lattices.

2.4.1 Root lattice decoding

In [43] low complexity algorithms for finding the closest point for different root lattices to an arbitrary point $\mathbf{y} \in \mathbb{R}^n$ are given. Let

$$f(\mathbf{y}) = \text{closest integer to } \mathbf{y}, \tag{2.29}$$

in case of a tie, choose the integer with the smallest absolute value. And let $g(\mathbf{y})$, be the same as $f(\mathbf{y})$ except that the worst component of $\mathbf{y}$, the one furtherest apart from an integer, is rounded the wrong way.

To find the closest point of D_n to $\mathbf{y}$: Given $\mathbf{y} \in \mathbb{R}^n$, the closest point of D_n, is whichever of $f(\mathbf{y})$ and $g(\mathbf{y})$ has an even sum of components.

And to find the closest point of $E8$ given $\mathbf{y} = (y_1, \ldots, y_8) \in \mathbb{R}^8$ is as follows:

1. Compute $f(\mathbf{y})$ and $g(\mathbf{y})$.
2. Select whichever has an even sum of components; call it $\hat{\mathbf{x}}_0$.
3. Compute $f(\mathbf{y} - \frac{\mathbf{1}}{\mathbf{2}})$ and $g(\mathbf{y} - \frac{\mathbf{1}}{\mathbf{2}})$, where $\frac{\mathbf{1}}{\mathbf{2}} = (\frac{1}{2}, \frac{1}{2}, \frac{1}{2}, \frac{1}{2}, \frac{1}{2}, \frac{1}{2}, \frac{1}{2}, \frac{1}{2})$
4. Select whichever has an even sum of components; add $\frac{1}{2}$ and call the result $\hat{\mathbf{x}}_1$.
5. The result, is whichever the closest point to $\mathbf{y}$ between $\hat{\mathbf{x}}_0$ and $\hat{\mathbf{x}}_1$ is.

2.4.2 Decoding Algorithm for Construction A lattices

Finding the closest lattice point from a lattice constructed with construction A [19, Chap 20.5], given point $\mathbf{y} \in \mathbb{R}^n$ is described as follows:

1. Reduce all y_i to the range $-1 \leq y_i < 3$ by subtracting a vector $4\mathbf{z}$, where $\mathbf{z} \in \mathbb{Z}^n$.

2. Denote the set of i for which $1 < y_i < 3$ as S.

3. For $i \in S$, replace y_i with $2 - y_i$.

4. Since $\mathbf{y}$ is now in the cube $-1 \leq y_i \leq 1$, apply the decoder for C to $\mathbf{y}$ to obtain $\mathbf{c}$.

5. For $i \in S$ change c_i to $2 - c_i$.

6. Then $\hat{\mathbf{x}} = \mathbf{c} + 4\mathbf{z}$ is the closest lattice point to $\mathbf{y}$.

2.4.3 Sphere decoding

The sphere decoding algorithm [35][45] solves the problem

$$\min_{\hat{\mathbf{x}} \in \Lambda} ||\mathbf{y} - \hat{\mathbf{x}}|| = \min_{\mathbf{w} \in \mathbf{y} - \Lambda} ||\mathbf{w}||, \tag{2.30}$$

where $\mathbf{y} = \mathbf{G}\zeta$ and $\mathbf{w} = \mathbf{G}\xi$ for $\zeta, \xi \in \mathbb{R}^n$. Then we have $\mathbf{w} = \sum_{i=1}^{n} \xi_i \mathbf{v}_i$ where $\xi_i = \zeta_i - b_i$, for $\mathbf{b} \in \mathbb{Z}^n$.

It is needed to find the shortest vector in the translate lattice $\mathbf{y} - \Lambda$. By constructing a ball of radius r centered at $\mathbf{y}$ and testing all lattice points inside the ball.

In Fig 2.4 a visual representation of the sphere decoding is shown. Clearly for the sphere decoding, if r is too large, there exist too many points inside the ball and the search is exponential in size. And if r is too small, there are no points inside the ball. There is no specific way on how to choose the value of r. One approach is that r be the covering radius of the lattice. The covering radius is the smallest distance from a lattice point and a deep hole, which is the furthest point in $\mathbb{R}^n$ form a lattice point. Another candidate is the minimum distance of the lattice, which is upper bounded by the norms of the basis vector in the generator matrix. But knowing the covering radius

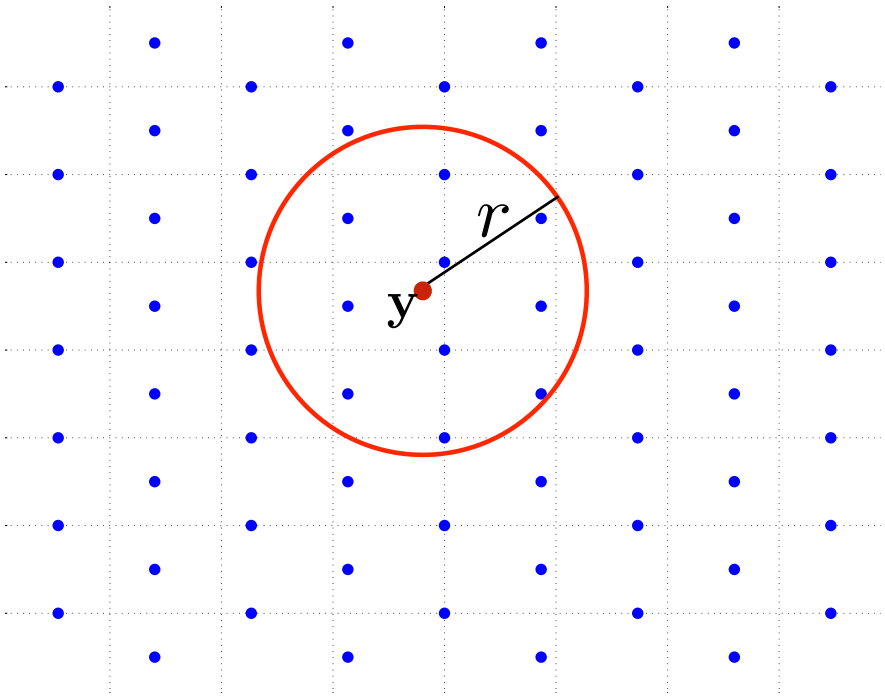

Figure 2.4: Sphere decoding algorithm, search the closest lattice point to $\mathbf{y}$ that are inside of a ball of radius r.

and the minimum distance of a lattice is not a trivial problem as the dimension increases.

For a more practical use of the sphere decoding algorithm the radius r could be adaptively adjusted according to the noise level. If there is no lattice point inside the ball the radius must be increased. If the distance of a detected lattice point is smaller than the radius, the radius r can be decreased. The sphere decoding algorithm has exponential complexity in the dimension of the lattice, which makes it unattractive for high dimensional lattices.

2.5 Volume-to-Noise Ratio

A useful measurement for comparison of the performance of the decoding of lattice codes is the volume-to-noise ratio (VNR)[29], which is analogous to

signal-to-noise ratio. The VNR is the ratio of the normalized volume of Λ and the normalized volume of a noise sphere. The VNR is defined as:

$$VNR = \frac{V(\Lambda)^{2/n}}{2\pi e\sigma^2}. \tag{2.31}$$

where $V(\Lambda)^{2/n}$ is the normalized volume of Λ per two dimensions. And $2\pi e\sigma^2$ is the volume of the normalized noise sphere of square radius $n\sigma^2$ for n large.

The VNR relies on the following ideas:

- For a given $V(\Lambda)$ the best possible shape $R(\Lambda)$ is a n-dimensional sphere.
- For having the probability of error $P_e(\Lambda, \sigma^2)$ to be small, the $R(\Lambda)$ must be larger than a noise sphere of radius $n\sigma^2$.

The volume $V_s(n, r^2)$ of an n-dimensional sphere of square radius nr^2, is given by

$$V_s(n, r^2) = \frac{(2\pi k r^2)^k}{k!} \to (2\pi e r^2)^k \tag{2.32}$$

for n even and $k = \frac{n}{2}$. The limit for $n \to \infty$ uses Stirling's approximations $k! \to (\frac{k}{e})^k$.

By the law of large numbers, as $n \to \infty$, the probability $P_e s(n, r^2, \sigma^2)$ that a Gaussian noise with variance σ^2 per dimension falls outside a sphere of square radius nr^2 goes to 0 if $\sigma^2 < r^2$, and goes to 1 if $\sigma^2 > r^2$.

Theorem 1: For large n , the probability of error $P_e(\Lambda, \sigma^2)$ of a minimum-distance decoder for a n-dimensional lattice Λ on an AWGN channel with noise variance σ^2 per dimension cannot be small unless $V(\Lambda)^{2/n} > 2\pi e\sigma^2$.

Moreover, if $V(\Lambda)^{2/n} \approx 2\pi e\sigma^2$, then n must be small unless n is large [29, Thoeriem 1].

In this dissertation the probability of error as a function of VNR was used as evaluation criteria, with has the following benefits:

- if a lattice Λ is scaled by $\beta > 0$ while σ^2 is scaled by β^2, then the volume scales by β^n, i.e $V(\beta\Lambda) = \beta^n V(\Lambda)$. And the probability of error does not change.
- The probability of error $P_e(\Lambda, \sigma^2)$ cannot be small unless $VNR > 1$.

The lattice capacity is when $VNR = 1$, this is only possible if:

$$\sigma^2 = \frac{V(\Lambda)^{2/n}}{2\pi e} \tag{2.33}$$

Remark In this dissertation the VNR is used, since only the unconstrained lattice transmission is considered. And it is considered a symbol error when one element in the decoded lattice point $\hat{x}_i$ is different to the transmitted lattice point x_i, i.e. $\hat{x}_i \neq x_i$. In this dissertation most of the results presented are in terms of the symbol error rate as a function of the VNR.

In Fig 2.5 the symbol error rate in terms of the VNR for different lattices is shown. It has been shown that as the dimension goes to infinity the lattice codes can achieve capacity with zero error probability[28][8].

2.6 Conlcusion

In this chapter various characteristics of lattices were described, like the kissing number, density and minimum distance of a lattice, also the well known small dimensional lattice were given, like the A_2, D_4, and E_8. For small dimensional lattices the decoding algorithm was also described. In addition

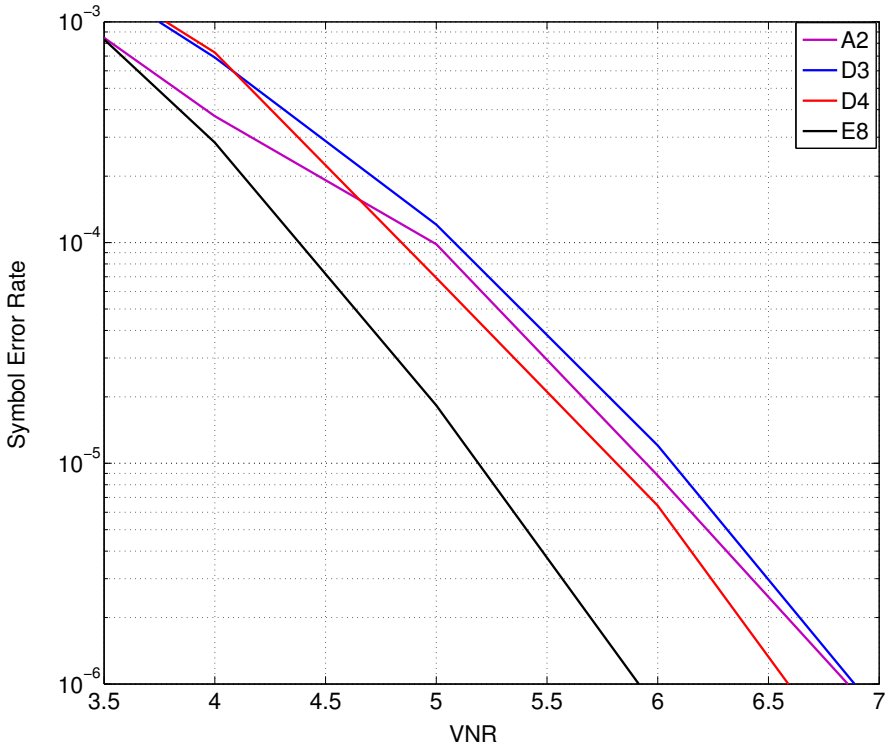

Figure 2.5: Probability of decoding error in terms of the volume-to-noise ratio (VNR) for different lattices.

construction based on linear binary codes and its decoding algorithm were described. Finally this chapter gave the definition of the volume-to-noise ratio (VNR) which is a way to normalize the noise power, and it is used to evaluate the performance of the lattice, since this dissertation considers only the unconstrained lattice transmission.

Chapter 3

Low Density Lattice Codes

In this chapter low density lattice code (LDLC) are introduced. LDLCs are high-dimensional lattices defined by a sparse inverse generator matrix. The decoding of LDLCs resembles low density parity check codes (LDPC), that is, using a belief propagation (BP) decoding algorithm on a sparse graph.

3.1 Definition

A low density lattice code (LDLC), introduced by Sommer et al. [1], is an n-dimensional lattice code defined by a non-singular generator matrix $\mathbf{G}$ satisfying the property that the inverse generator matrix $\mathbf{H}= \mathbf{G}^{-1}$ is sparse and $|\det(\mathbf{G})|=1$.

The row/column degree is the number of nonzero elements in the row/column of the inverse generator matrix $\mathbf{H}$. The LDLC is called regular if all row degrees and column degrees of the inverse generator matrix are equal to the same value d.

For LDLC lattices the values of the non-zero elements of the inverse generator matrix $\mathbf{H}$ are taken from the generator sequence $\mathbf{h}= \{h_1, \ldots, h_d\}$, such that $h_1 \geq \cdots \geq h_d$. Note that the inverse generator for binary low density parity check codes (LDPC) are only defined by the locations of the non-zero elements. But for LDLC lattices these non-zero values need to be chosen.

In order to guarantee the exponential converge of the iterative decoding the values of the generator sequence need to satisfy:

$$\alpha \triangleq \frac{\sum_{i=2}^{d} h_i^2}{h_1^2} \leq 1, \tag{3.1}$$

having α too small will results in a faster convergence, but numerical results show that α should be close to 1. But how to choose the best generator sequence or the best value of α is a hard problem. In this dissertation a contribution on how to choose the generator sequence is given.

A regular LDLC is called "latin square" if every row and column of the inverse generator matrix has the same d non-zero elements, except for a possible change of random sign.

As an example we have a latin square LDLC with dimension $n = 6$, degree $d = 3$ and generator sequence $\{1, 0.8, 0.5\}$, a possible inverse generator matrix is:

$$\mathbf{H} = \begin{bmatrix} 0 & -0.8 & 0 & -0.5 & 1 & 0 \\ 0.8 & 0 & 0 & 1 & 0 & -0.5 \\ 0 & 0.5 & 1 & 0 & 0.8 & 0 \\ 0 & 0 & -0.5 & -0.8 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0.5 & 0.8 \\ 0.5 & -1 & -0.8 & 0 & 0 & 0 \end{bmatrix}. \tag{3.2}$$

This inverse generator matrix 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.

3.2 Construction

In [1] an algorithm to construct latin square LDLC lattice inverse generator matrix $\mathbf{H}$ was introduced. This algorithm generates d random permutations

of length n. Then, a searching operation is performed to identify cycles of degree 2 and 4 in order to eliminate them, the importance of the inverse generator matrix not having 2-cycles and 4-cycles is to improve the performance of the BP decoding. This process gives the positions of the non-zero elements. These non zero elements are taken from the generator sequence $\mathbf{h}$ and are assigned in a latin square manner, i.e. where all columns and rows contain all elements in the generator sequence just with a change in order and random sign. In Fig 3.1 the block diagram of the construction algorithm presented in [1] is shown.

The algorithm presented in [1] has the following limitations:

1. All the position of the matrices needs to be stored while searching for cycles (high use of memory).
2. Searching for cycles is computationaly demanding as the dimension increases.

A construction that can reduce the computational complexity for LDLC lattice inverse generator matrices is desired, in order to lead to more practical applications and will be shown in Chapt 4.

3.2.1 Triangular Form

A triangular form for the inverse generator matrix is a convenient structure for encoding and shaping operations.

A triangular inverse generator matrix for LDLC lattices was introduced in [30]. The triangular inverse generator $\mathbf{H_T}$ assumes that all elements along the diagonal are equal to the largest element in the generator sequence with "+" sign. This can be done by permuting rows and columns in the inverse generator matrix $\mathbf{H}$, and by multiplying whole rows by -1 when is necessary. Then by a similar procedure by which the latin squared LDLC lattice was constructed, the column degree of the rightmost column of $\mathbf{H}$ will start from 1 and gradually increase until d. In the same manner, the row degree of the top row will be 1, and it will gradually increase until d. The triangular

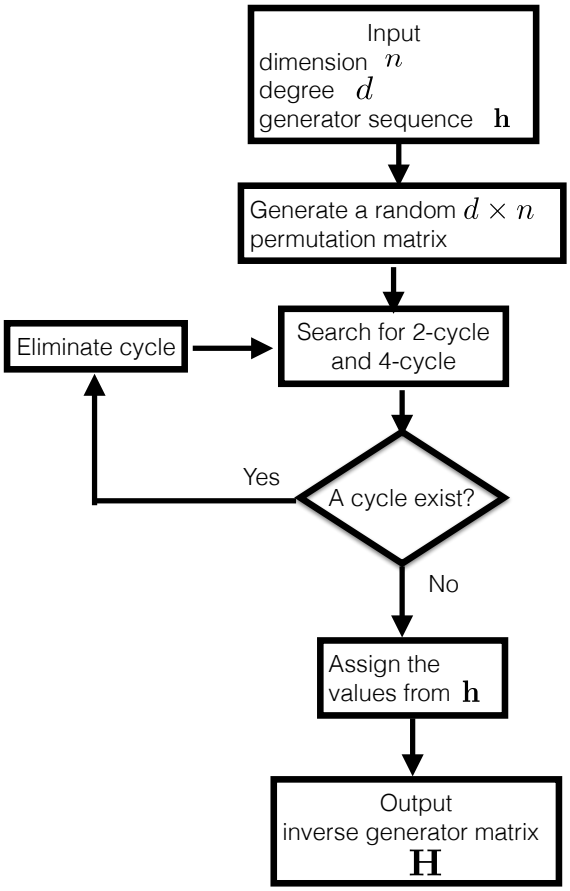

Figure 3.1: Block diagram for constructing latin square LDLC lattice inverse generator matrices [1].

inverse generator $\mathbf{H_T}$ is not regular.

For example the triangular LDLC lattice inverse generator of dimension $n = 8$, degree $d = 3$ and generator sequence $h = \{1, 0.5, 0.7\}$ is as follows:

$$\mathbf{H_T} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0.7 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -0.7 & 1 & 0 & 0 & 0 & 0 \\ -0.5 & 0 & 0 & 0.7 & 1 & 0 & 0 & 0 \\ 0 & -0.7 & 0 & 0.5 & 0 & 1 & 0 & 0 \\ 0 & -0.5 & 0 & 0 & 0.7 & 0 & 1 & 0 \\ 0 & 0 & -0.5 & 0 & 0 & 0.7 & 0 & 1 \end{bmatrix}. \tag{3.3}$$

Having a triangular structure is convenient for encoding, but the components where the $\mathbf{H_T}$ column have a low degree are less protected against error due to noise. This can be solved by giving to the less protected elements a smaller amount of information.

A possible finite constellation for the $\mathbf{H_T}$ of the previous example is as follows: the first two integers can assume one of two possible integer values (1 bit of information, e.g $\{0, 4\}$). The next two integers can assume one of four integer values (2 bits of information, e.g $\{0, 2, 4, 6\}$), where all the other integers can assume one of eight possible values and contain 3 bits of information $\{0, 1, 2, 3, 4, 5, 6, 7\}$.

If the constellation size of all integers is eight, the information rate is 3 bits/integer. But having a triangular form makes it impractical to have this rate. Instead the information rate is $(2 \cdot 1 + 2 \cdot 2 + 4 \cdot 3)/8 = 2.25$bits/integer. There is tradeoff between the information/rate of the code and the performance in terms of the VNR.

3.3 Decoding LDLC lattices

In this section the description of the decoding algorithm for LDLC lattices is described. The decoding algorithm is based on the belief propagation algorithm.

3.3.1 Belief Propagation

The belief propagation (BP) is a technique invented in 1982 by Pearl [46] to solve inference problems. Inference problems can be found in different fields, such as artificial intelligence, computer vision, statistical physics and digital communications.

The BP algorithm has given excellent results for decoding error-correcting codes. And has been used to decode turbo codes [4] and LDPC codes [5]. In fact the decoding of error correcting codes is an inference problem, the receiver tries to infer the message that was transmitted and has been corrupted with noise.

In general the the BP decoding algorithm sends messages between two types of nodes, one called variable nodes and another called check nodes. And after enough iterations these message are likely to converge. A graphical representation which the variable node and the check nodes are illustrated is called Tanner graph or bipartite graph [47].

3.3.2 Decoding Algorithm

Similar to LDPC codes, the bipartite graph is defined with variable nodes corresponding to a single element of the lattice codeword $\mathbf{x} = \mathbf{Gb}$ and check nodes corresponding to a check equation of the form $\sum_k h_k x_{i_k}$ is an *integer*, where i_k are the positions of the non-zero elements at the corresponding row of the inverse generator $\mathbf{H}$, and the *integer* is unknown. In Fig 3.2 a factor graph is shown for the inverse generator:

$$\mathbf{H} = \begin{bmatrix} 1 & 0 & 0 & \frac{1}{\sqrt{2}} & 0 & 0 \\ 0 & 1 & 0 & 0 & \frac{1}{\sqrt{2}} & 0 \\ 0 & 0 & 1 & 0 & 0 & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & 0 & 0 & 0 & 1 & 0 \\ 0 & \frac{1}{\sqrt{2}} & 0 & 0 & 0 & 1 \\ 0 & 0 & \frac{1}{\sqrt{2}} & 1 & 0 & 0 \end{bmatrix}, \tag{3.4}$$

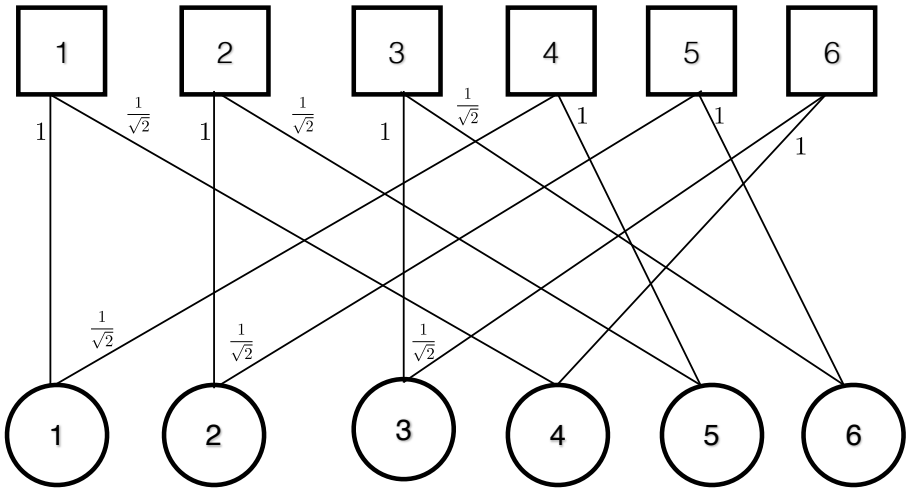

Figure 3.2: Example for the factor graph for LDLC lattice with $n = 6$ and generator sequence $\mathbf{h} = \{1, \frac{1}{\sqrt{2}}\}$.

with $n = 6$ and generator sequence $\mathbf{h} = \{1, \frac{1}{\sqrt{2}}\}$. An edge connects variable node i and check node j if and only if $\mathbf{H}_{i,j} \neq 0$. This properties of the LDLC lattice can be exploited by decoding it by iterative decoding algorithm such as belief propagation.

Sommer et al. introduced an iterative decoder for LDLC lattices in [1]. In the case of LDLC lattice decoder the messages between variable and check nodes are real functions, for the AWGN channel these functions are Gaussian mixtures, instead of scalar values as in binary LDPC codes.

Let the transmitted lattice point be $\mathbf{x}= \mathbf{Gb}$ where $\mathbf{G}$ is the generator of the n-dimensional LDLC lattice with $\mathbf{G} = \mathbf{H}^{-1}$, and $\mathbf{b} \in \mathbb{Z}^n$. And let $\mathbf{y}$ be the received signal given by:

$$\mathbf{y} = \mathbf{x} + \mathbf{z}, \tag{3.5}$$

where $\mathbf{z} \sim \mathcal{N}(w; 0, \sigma^2)$ is the additive white Gaussian noise with 0 mean and variance σ^2. The Gaussian function is defined as:

$$\mathcal{N}(w; \mu, \sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(w-\mu)^2}{2\sigma^2}} \tag{3.6}$$

.

The LDLC decoder algorithm presented in [1] is as follows:

- *Initialization*: each variable node k send to all its check nodes a Gaussian message $f(w)$

$$f_k(w) = Y_k(w) = \mathcal{N}(w; y_k, \sigma^2), \tag{3.7}$$

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

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

  1. Convolution step: all messages except $f_j(w)$, are convolved after expanding each message by its generator sequence h:

$$\tilde{p}_j(w) = f_1\Big(\frac{w}{h_1}\Big) \otimes \cdots \otimes f_{j-1}\Big(\frac{w}{h_{j-1}}\Big) \otimes f_{j+1}\Big(\frac{w}{h_{j+1}}\Big) \otimes \ldots f_d\Big(\frac{w}{h_d}\Big). \tag{3.8}$$

     for $j = 1, 2 \ldots d$, where $\otimes$ denotes convolution.

  2. Stretching step: The messages $\tilde{p}_j(w)$ are stretched by $-h_j$ to:

$$p_j(w) = \tilde{p}(-h_j w). \tag{3.9}$$

3. Periodic extension: Each message $p_j(w)$ is extended to a periodic function with period $1/|h_j|$:

$$R_j(w) = \sum_{i=-\infty}^{\infty} p_j(w - \frac{i}{h_j}). \tag{3.10}$$

- *Variable-to-check message*: The variable-to-check messages $f_j(w)$ is calculated in two steps.

  1. The product step: where a product of all incoming messages is made, except for the message j:

$$\tilde{f}_j(w) = Y_j(w) \prod_{l=1, l\neq j}^{d} R_l(w). \tag{3.11}$$

  2. The normalization step: where the message $\tilde{f}_j(w)$ is normalized as:

$$f_j(w) = \frac{\tilde{f}_j(w)}{\int_{-\infty}^{\infty} \tilde{f}_j(w)dw}. \tag{3.12}$$

These steps are repeated until the maximum number of iterations is reached.

- *Final decision*: The estimated integer information vector $\hat{\mathbf{b}}$ is calculated. At the last iteration r, the product of all incoming message is performed as:

$$\tilde{f}_{final}^{(r)}(w) = Y_j(w) \prod_{l=1}^{d} R_l(w), \tag{3.13}$$

and the estimated lattice point $\hat{\mathbf{x}}$ and estimated information $\hat{\mathbf{b}}$ are:

$$\hat{\mathbf{x}} = \underset{\mathbf{w}}{\operatorname{argmax}} \tilde{f}^{(r)}_{final}(w), \text{ and} \tag{3.14}$$

$$\hat{\mathbf{b}} = \lfloor \mathbf{H}\hat{\mathbf{x}} \rceil \tag{3.15}$$

respectively, the $\lfloor \rceil$ denotes the rounding to the closest integer operation.

The complexity of the quantized BP decoding algorithm [1] is $\mathcal{O}(n \cdot t \cdot d \cdot \frac{L}{\Delta})$ where Δ is the probability density function resolution and L is the range length, and is dominated by a discrete Fourier transform of size $1/\Delta$ occurred during the convolution step. Also, as the message gets narrow in every iteration, quantization errors could occur, i.e. a zero output instead of an impulse. In addition, the storage required for the messages between variable and check nodes is d/Δ per node. During the implementation w is restricted to some range of integers which increases the storage needed.

Sommer et al. shows the performance of the quantized version for different lattice dimension, they show that for dimension $n = 100,000$ LDLCs can attain a symbol error rate of 10^{-5} at VNR of 0.6dB. The generator sequence for this result is $\{\frac{1}{2.31}, \frac{1}{3.17}, \frac{1}{5.11}, \frac{1}{7.33}, \frac{1}{11.71}, \frac{1}{13.11}, \frac{1}{17.55}\}$ which has $\alpha = 0.9220$. The results given in [1] are shown in Table 3.1.

3.3.3 Decoder Convergence

In this section a review of the proof of the variance convergence for the LDLC lattice decoder [1] is presented. Analysis of the convergence of the variance is simpler than analyzing the convergence on the means, because eq (3.10) and (3.12) are mixtures of infinite Gaussians.

In the latin square LDLC construction there are d edge types, corresponding to the d coefficients. For any iteration l, all messages have the

Table 3.1: Results presented in [1] for different LDLC lattice dimension. $VNR = 0$ dB is the capacity.

Dimension n	Degree d	Generator sequence $\mathbf{h}$	VNR (dB)	SER
100	5	$\{\frac{1}{2.31}, \frac{1}{3.17}, \frac{1}{5.11}, \frac{1}{7.33}, \frac{1}{11.71}\}$	3.7	10^{-5}
1,000	7	$\{\frac{1}{2.31}, \frac{1}{3.17}, \frac{1}{5.11}, \frac{1}{7.33}, \frac{1}{11.71}, \frac{1}{13.11}, \frac{1}{17.55}\}$	1.5	10^{-5}
10,000	7	$\{\frac{1}{2.31}, \frac{1}{3.17}, \frac{1}{5.11}, \frac{1}{7.33}, \frac{1}{11.71}, \frac{1}{13.11}, \frac{1}{17.55}\}$	0.8	10^{-5}
100,000	7	$\{\frac{1}{2.31}, \frac{1}{3.17}, \frac{1}{5.11}, \frac{1}{7.33}, \frac{1}{11.71}, \frac{1}{13.11}, \frac{1}{17.55}\}$	0.6	10^{-5}

same mixture variance [1, Lemma 3]. Let $v_i^{(l)}$ be the mixture variance from the input of a check node along the edge for h_i, for $i = 1, \ldots, d$, the mixture variance of the outgoing message is u_i.

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

$$\frac{1}{v_2^{l+1}} = \frac{1}{u_1} + \frac{1}{u_3} + \frac{1}{u_4} + \frac{1}{\sigma^2} \tag{3.16}$$

$$\leq \frac{1}{u_1}, \tag{3.17}$$

since $u_i \geq 0$.

For the check node, the mixture variance is:

$$u_1 = \frac{h_2^2 v_2 + h_3^2 v_3 + h_4^4 v_4}{h_1^2}, \tag{3.18}$$

we can write

$$u_1 \leq \frac{h_2^2 + h_3^2 + h_4^4}{h_1^2} \cdot v_2 = \alpha v_2 \tag{3.19}$$

since it is assumed that $v_1 \geq v_2 \geq v_3 \geq v_4$. Combining (3.19) and (3.17)

$$\frac{1}{v_2^{(l+1)}} \geq \frac{1}{u_1} \geq \frac{1}{\alpha v_2^{(l)}} \tag{3.20}$$

$$v_2^{(l+1)} \leq \alpha v_2 \tag{3.21}$$

since $v_2^{(1)} = \sigma^2$, the sequence of $v_2^{(l)}$ are exponentially approaching 0 in the iteration number l. In order to guarantee the exponential convergence of the decoding α needs to satisfy $\alpha < 1$.

Fig. 3.3 shows the path of the messages that participate in this check. Choosing a small value of α results in a faster convergence, but should not be too small since errors occurs, probably due to the lattice itself is bad or due to the fast convergence . In the following chapters an analysis of the noise thresholds, for the proposed decoder, is shown. The result (section 5.5.1) gives an insight of which values for α "close to, but strictly less than, 1" are suitable for a good performance (contribution of this dissertation).

3.4 Parametric Decoders

In order to implement decoders in an efficient way, in terms of computational complexity, various authors have proposed parametric LDLC lattice decod-

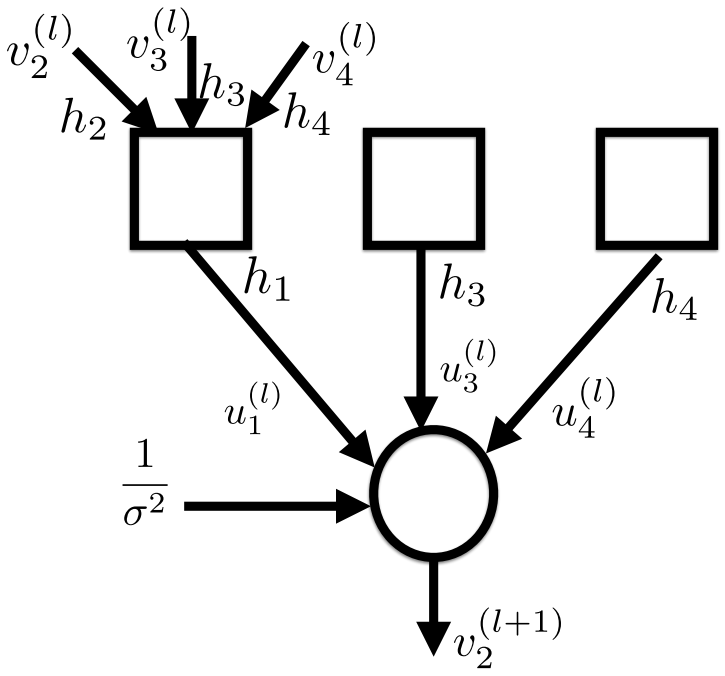

Figure 3.3: Message propagation for variance converge analysis at variance v_2^{l+1}

.

ing algorithms which employs finite Gaussian mixtures to approximate the messages consisting on an infinite number of Gaussians.

An LDLC lattice decoder using a Gaussian mixture reduction algorithm was introduced in [2], this algorithm approximates a large number of Gaussians with a smaller number of Gaussians. All possible pairs of Gaussians on a list are searched and the closest pairs, in terms of the Kullback-Leiber divergence, are replaced with a single Gaussian. This reduction process is done at every multiplication that takes place at the variable node. Further, using single Gaussians as the messages between the variable and check nodes leads to reduced memory requirements with a minor performance penalty [32]. This parametric algorithm uses two parameters, the number of Gaussians in the mixture M and a threshold θ in order to say if two Gaussians can be approximated with a smaller number of Gaussian. The smaller the value of θ, the better the approximation performed, but increases the number of

Gaussians in the mixture. The complexity of this algorithm is $O(n \cdot d \cdot t \cdot M^4)$.

Yona and Feder [31] presented a parametric decoder, where the Gaussian mixture approximation is made by taking the dominating Gaussian in the mixture. First, the Gaussian in the mixture are stored in a table, and in order to obtain the dominant mixture a searching process is needed, the Gaussians mixture tables are sorted in terms of the mixture coefficients. But these relatively complicated operations, whether the Gaussian mixture reduction or sorting, must be performed at every multiplication at the variable node, on each iteration. Such operations may not be suitable if LDLC decoders are to be implemented in hardware. The complexity of this algorithm is $O(n \cdot d \cdot t \cdot K \cdot M^3)$, where t is the number of iterations, M is the size of the search tables, and K is the finite number of periods taken in the periodic expansion.

In [48] a single-Gaussian moment matching (MM) approximation was used internally at the variable node for every incoming message, with this approximation internally at the variable node maintains a single Gaussian message that is easily represented with its mean and variance, which reduces the storage memory required. This approximation is not an accurate approximation since at every multiplication some part of information is lost. This loss can be high, depending on the values of the means and variances. In parametric LDLC lattice decoding algorithms, having an accurate approximation for the message is a key element in order to have a good performance.

In addition, in [48] density evolution noise thresholds were presented. The noise threshold give an insight of the lowest VNR for the parametric decoder asymptotically large dimensional lattice converges. The noise thresholds were calculated with a data pool of 10^5, over 50 iterations and different degrees d were studied. The single Gaussian moment matching decoder is computationally efficient, it can reduce the memory needed for implementation and has a noise threshold of 0.1dB to the best know LDLC lattice using the quantized decoder at dimension $100,000$. But in the work presented in [48] finite dimension results were not given.

3.5 Conclusion

Low density lattice codes (LDLC), introduced by Sommer, Feder and Shalvi [1], are high-dimensional lattices defined by a sparse inverse generator matrix. The construction and decoding of LDLCs resemble low density parity check (LDPC) codes, that is, using a belief propagation (BP) decoding algorithm on a sparse graph. It was reported that the LDLC belief propagation decoder attains a symbol error rate of 10^{-5} at VNR= 0.6 dB.

In order to implement a LDLC decoder the message needs to be quantized, and various authors had presented different parametric decoders, where the message are only represented by the means and variance of the Gaussian mixtures, which is a key feature to the proposed decoder in this work.

Already, relaying and physical layer network coding schemes that use LDLCs have been described [49] [12][13].

Chapter 4

Array LDLC Lattices

In this chapter a proposed LDLC inverse generator matrix construction called "Array LDLC lattices" is described. This construction uses principles of array codes in order to have an efficiently encoded and deterministic construction. The proposed inverse generator matrix is sparse, 4-cycle free and is defined by four parameters.

4.1 Array Codes

Low density parity check codes (LDPC) array codes [50] refer to a general class of algebraic error correcting codes defined on arrays, for detecting and correcting burst of errors.

LDPC array codes use a deterministic construction rather than random construction. If we see array codes as binary codes, their parity check matrices are sparse, and can be decoded in an iterative manner [51]. Also having an structured construction can be used to guarantee distance properties and/or reduce the complexity of the implementation.

4.1.1 Construction

An algebraic construction for array codes, which is analogous to Reed-Salomon codes is presented in [52], has symbols that lie in rings rather than in Galois fields. A construction for LDPC binary code is introduced in [33], these

LDPC codes are called "LDPC array codes".

The LDPC array code parity check matrix is defined by three parameters: a prime number p and two integers k and j such that $j \leq k \leq p$, where k and j 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. Then the parity check matrix of an LDPC array code is:

$$\mathbf{H} = \begin{bmatrix} (\mathbf{P}^0)^0 & (\mathbf{P}^1)^0 & \cdots & (\mathbf{P}^{k-1})^0 \\ (\mathbf{P}^0)^1 & (\mathbf{P}^1)^1 & \cdots & (\mathbf{P}^{k-1})^1 \\ (\mathbf{P}^0)^2 & (\mathbf{P}^1)^2 & \cdots & (\mathbf{P}^{k-1})^2 \\ \vdots & \vdots & \ddots & \vdots \\ (\mathbf{P}^0)^{j-1} & (\mathbf{P}^1)^{j-1} & \cdots & (\mathbf{P}^{k-1})^{j-1} \end{bmatrix}, \tag{4.1}$$

$$\mathbf{H} = \begin{bmatrix} \mathbf{I} & \mathbf{I} & \cdots & \mathbf{I} \\ \mathbf{I} & (\mathbf{P}^1)^1 & \cdots & (\mathbf{P}^{k-1})^1 \\ \mathbf{I} & (\mathbf{P}^1)^2 & \cdots & (\mathbf{P}^{k-1})^2 \\ \vdots & \vdots & \ddots & \vdots \\ (\mathbf{I} & (\mathbf{P}^1)^{j-1} & \cdots & (\mathbf{P}^{k-1})^{j-1} \end{bmatrix}, \tag{4.2}$$

where $\mathbf{P}$ is a $p \times p$ permutation matrix and $\mathbf{I}$ is the $p \times p$ identity matrix. The matrix $\mathbf{P}^k$ represents a k left cyclic shift. For example, for $p = 5$, the matrix $\mathbf{P}$ is:

$$\mathbf{P} = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 \end{bmatrix}, \tag{4.3}$$

and

$$\mathbf{P}^3 = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \end{bmatrix}, \tag{4.4}$$

represents three left shift by $\mathbf{P}$. By construction, $\mathbf{H}$ is 4-cycle free, since there are not two rows where a 1 is overlapping in two columns.

4.1.2 Minimum Distance

The minimum distance d of a binary code C is the least Hamming distance between any two distinct codewords $\mathbf{u}$ and $\mathbf{u}'$, which can be defined as:

$$d = \min\{d_H(\mathbf{u}, \mathbf{u}') | u, u' \in C, \mathbf{u} \neq \mathbf{u}'\} \tag{4.5}$$

where $d_H(\mathbf{u}, \mathbf{u}')$ denotes the Hamming distance between $\mathbf{u}$ and $\mathbf{u}'$.

In [3] the minimum distance $d(p, k)$ of the array LDPC codes for the prime number p and row/column weight $k = j$, were studied. The results are presented on Table 4.1.

4.1.3 Triangular Form

Having a parity check matrix in triangular form is desirable. Gaussian elimination could be used to obtain the desirable triangular form, but this operation results in increasing the processing complexity as the dimension increases, making this method unattractive.

Elefteriou and Olcer [53] define a modified array code by cyclically shifting the rows of the $\mathbf{H}$ matrix in a block-wise manner. The number of cyclic shifts for each block row is such that the $jp \times kp$ leftmost subblock contains

p	d(p,7)	d(p,6)	d(p,5)	d(p,4)
7	14	12	12	8
11	20	16	10	10
13	20	14	12	10
17	18-24	16	12	10
19	18 or 20	18	12	10
23	18-22	18-20	12	10
29	18-24	18-20	12	10
31	18-24	18-20	12	10
43	18-24	18-20	12	10
47	18-24	18-20	12	10
53	18-24	18-20	12	10
59	18-24	18-20	12	10

Table 4.1: Minimum distance results presented in [3] for array LDPC codes for different values of p and $k = j$.

the identity matrix $\mathbf{I}$ along the diagonal. We define this new matrix $\mathbf{H_s}$ as:

$$\mathbf{H_s} = \begin{bmatrix} \mathbf{I} & \mathbf{I} & \mathbf{I} & \cdots & \mathbf{I} & \mathbf{I} & \cdots & \mathbf{I} \\ \mathbf{P}^{k-1} & \mathbf{I} & \mathbf{P} & \cdots & \mathbf{P}^{j-2} & \mathbf{P}^{j-1} & \cdots & \mathbf{P}^{k-2} \\ \mathbf{P}^{2(k-2)} & \mathbf{P}^{2(k-1)} & \mathbf{I} & \cdots & \mathbf{P}^{2(j-3)} & \mathbf{P}^{2(j-2)} & \cdots & \mathbf{P}^{2(k-3)} \\ \vdots & \vdots & \vdots & & \vdots & \vdots & & \vdots \\ \mathbf{P}^{(j-1)(k-j+1)} & \mathbf{P}^{(j-1)(k-j+2)} & & \cdots & \mathbf{I} & \mathbf{P}^{j-1} & \cdots & \mathbf{P}^{(j-1)(k-1)} \end{bmatrix}. \tag{4.6}$$

The matrix $\mathbf{H_s}$ is 4-cycle free and has the same column and row weight as $\mathbf{H}$.

To make this matrix in a triangular form, the lower triangular elements of $\mathbf{H_s}$ are set to zero, obtaining:

$$\mathbf{H_p} = \begin{bmatrix} \mathbf{I}\,\mathbf{I}\,\mathbf{I}\cdots & \mathbf{I} & \mathbf{I} & \cdots & \mathbf{I} \\ \mathbf{0}\,\mathbf{I}\,\mathbf{P}\cdots & \mathbf{P}^{j-2} & \mathbf{P}^{j-1} & \cdots & \mathbf{P}^{k-2} \\ \mathbf{0}\,\mathbf{0}\,\mathbf{I}\cdots & \mathbf{P}^{2(j-3)} & \mathbf{P}^{2(j-2)} & \cdots & \mathbf{P}^{2(k-3)} \\ \vdots\,\vdots\,\vdots & \vdots & \vdots & & \vdots \\ \mathbf{0}\,\mathbf{0}\,\mathbf{0}\cdots & \mathbf{I} & \mathbf{P}^{j-1} & \cdots & \mathbf{P}^{(j-1)(k-1)} \end{bmatrix}, \tag{4.7}$$

where $\mathbf{0}$ is the $p \times p$ null matrix.

The triangular form of the LDPC array code has no 4-cycles. The triangular form structure is useful for an easy encoding.

4.2 Array LDLC Lattice

In this section one of the main contributions of this work is described, that is a structured construction for LDLC lattices. Having a structured construction is desired since it can be easily implemented in hardware.

The proposed method to construct LDLC lattices takes as a reference the construction of the array LDPC codes. To be in congruence with the base of the construction, it is called "Array LDLC lattice".

4.2.1 Desired Conditions for the parity check matrix H

In order to construct the parity check matrix $\mathbf{H}$ the following conditions are desired:

1. Sparseness, the row and column weight is less or equal to d.

2. 4-cycle free, to improve the performance of the belief propagation decoding algorithm.

3. Triangular structure, In order to apply shaping operation for practical applications.

4. For every row with degree d_j the condition,

$$\alpha \triangleq \frac{\sum_{i=2}^{d_j} h_i^2}{h_1^2} < 1, \tag{4.8}$$

is satisfied. Recall that $\alpha < 1$ guarantees an exponential rate for convergence on the message variances as was shown in Section 3.3.3.

4.2.2 Proposed Construction

Since the LDLC parity check matrix is square, the parameters j, k are the same and denote the maximum row and column degree, that is $d = j = k$. Generating the modified array code as in equation (4.7) gives a sparse, 4-cycle free and triangular binary matrix.

The non-zero elements are modified in two ways:

1. Elements on the main have the greatest element in the generator sequence h_1.

2. Off-diagonal elements are modified to be elements of the generator sequence $h_2 \geq \cdots \geq h_{d-1} > 0$ with random sign, as with [1].

The array LDLC lattice inverse generator matrix $\mathbf{H}$ is:

$$\mathbf{H} = \begin{pmatrix} h_1\mathbf{I} & h_2\mathbf{I} & h_3\mathbf{I} & h_4\mathbf{I} & \dots & h_d\mathbf{I} \\ & h_1\mathbf{I} & h_2\mathbf{P} & h_3\mathbf{P}^2 & \dots & h_{d-1}\mathbf{P}^{d-1} \\ & & h_1\mathbf{I} & h_2\mathbf{P}^2 & \vdots & h_{d-2}\mathbf{P}^{d-2} \\ & 0 & & h_1\mathbf{I} & \dots & h_{d-3}\mathbf{P}^{d-3} \\ & & & & \ddots & \vdots \\ & & & & & h_1\mathbf{I} \end{pmatrix} \tag{4.9}$$

4.2.3 Reliability for low degree message

Having a lower triangular structure, it is evident that the rows and columns do not all contain d non-zero elements. This implies that the codewords components whose column degree is low are less protected. For example, a column with only one non-zero element is uncoded, since it only participates in a single check equation.

There are two techniques that can increase the protection of the less protected elements. the first one is to let the information integer that are less protected should have a smaller amount of information, for example belong to a smaller constellation, as was observed by Sommer et al.[30]. But there is not an efficient way on how to choose this constellation.

The second technique is by increase the power of the less protected elements, this is done by scaling those elements by some factor. In the proposed construction, the elements are scaled by a factor $\frac{1}{c_i}$ for $i = 1, 2 \dots, d$, where $c_1 > c_2 > \cdots > c_d$, the vectors $\mathbf{c} = \{c_1, \dots, c_d\}$ are called "balance factors". In Fig. 4.1 the idea of this principle is illustrated.

Lemma 4.2.1 *In order to keep a constrain in the power the balance factors need to satisfy that:*

$$\prod_{i=1}^{d} c_i = 1, \tag{4.10}$$

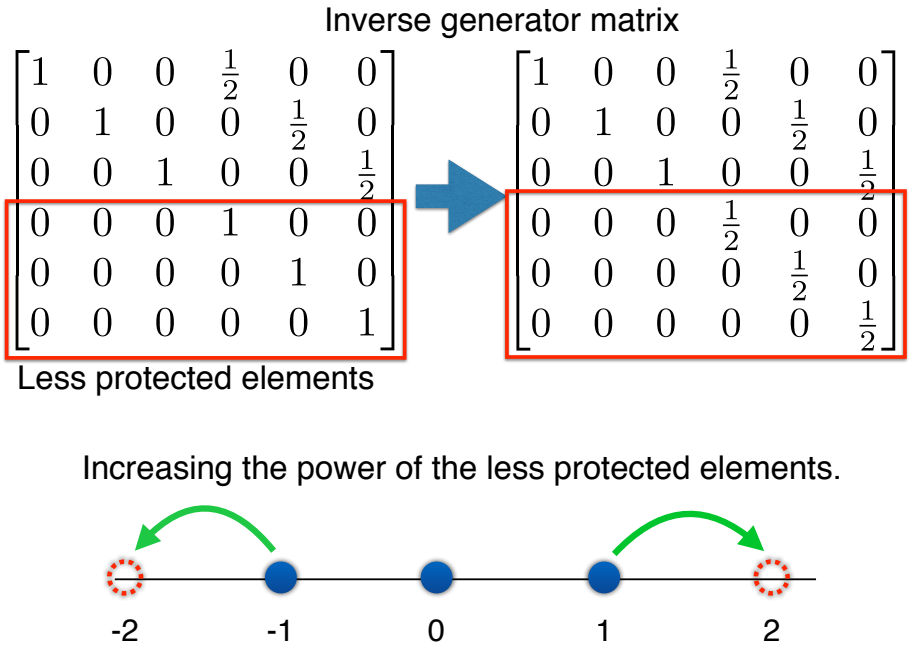

Figure 4.1: Increasing the power of less protected elements (red square) also increases the protection of those elements.

so that $|\det(\mathbf{H})| = |h_1^n|$ *is not modified.*

Lemma 4.2.2 *The balance factors multiplying each block in the inverse generator matrix* $\mathbf{H}$ *do not affect the value of* α_j *[54], then* α_j *is given by:*

$$\alpha_j = \frac{\sum_{i=2}^{d_j} \frac{h_i^2}{c_j^2}}{\frac{h_1^2}{c_j^2}}, \tag{4.11}$$

for block $j = 1, 2, \ldots, d$.

Finally the inverse generator matrix for array LDLC lattices $\mathbf{H}$ is:

$$\mathbf{H} = \begin{pmatrix} \frac{h_1}{c_1}\mathbf{I} & \frac{h_2}{c_1}\mathbf{I} & \frac{h_3}{c_1}\mathbf{I} & \frac{h_4}{c_1}\mathbf{I} & \dots & \frac{h_d}{c_1}\mathbf{I} \\ & \frac{h_1}{c_2}\mathbf{I} & \frac{h_2}{c_2}\mathbf{P}^1 & \frac{h_3}{c_2}\mathbf{P}^2 & \dots & \frac{h_{d-1}}{c_2}\mathbf{P}^{d-1} \\ & & \frac{h_1}{c_3}\mathbf{I} & \frac{h_1}{c_3}\mathbf{P}^2 & \vdots & \frac{h_{d-2}}{c_3}P^{d-2} \\ & 0 & & \frac{h_1}{c_4}\mathbf{I} & \dots & \frac{h_{d-3}}{c_4}\mathbf{P}^{d-3} \\ & & & & \ddots & \vdots \\ & & & & & \frac{h_1}{c_d}\mathbf{I} \end{pmatrix}. \tag{4.12}$$

An example in Fig. 4.2 is presented, where a LDLC based on array codes parity check matrix is given with $d = 4$, $p = 5$,the generator sequence $\mathbf{h} = \{1, \frac{1}{2}, \frac{1}{2}, \frac{1}{2}, \frac{1}{2}\}$., and the balance factors are $\mathbf{c} = \{1, 1, 2, 4\}$.

Since it has been proved that there no exist a length 4 cycles in array codes. The array LDLC lattices inverse generator matrix is also free of 4-cycles.

4.2.4 Array LDLC lattice construction algorithm

In this section the algorithm for constructing the inverse generator for array LDLC lattices is given. The construction algorithm generates the inverse generator matrix which is Which is lower triangular, sparse and 4-cycles free. The construction algorithm is as follows:

Input:

- prime number p.
- maximum number of nonzero elements d.
- generator sequence $\mathbf{h}$.
- balance factors $\mathbf{c}$.

Output :

- A $dp \times dp$ parity check matrix $\mathbf{H}$.

$$\left[\begin{array}{cccccccccccccccccccc}
1 & 0 & 0 & 0 & 0 & 0 & -\frac{1}{2} & 0 & 0 & 0 & 0 & 0 & \frac{1}{2} & 0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{2} \\
0 & 1 & 0 & 0 & 0 & 0 & 0 & \frac{1}{2} & 0 & 0 & 0 & 0 & 0 & -\frac{1}{2} & \frac{1}{2} & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & \frac{1}{2} & 0 & 0 & 0 & -\frac{1}{2} & 0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{2} & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & -\frac{1}{2} & 0 & 0 & 0 & \frac{1}{2} & 0 & 0 & 0 & 0 & 0 & 0 & -\frac{1}{2} & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & \frac{1}{2} & 0 & 0 & 0 & 0 & 0 & 0 & -\frac{1}{2} & 0 & \frac{1}{2} \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & \frac{1}{2} & 0 & -\frac{1}{2} & 0 & 0 & 0 & 0 & \frac{1}{2} & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & -\frac{1}{2} & 0 & \frac{1}{2} & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & \frac{1}{2} & 0 & 0 & 0 & 0 & 0 & -\frac{1}{2} & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & \frac{1}{2} & 0 & 0 & 0 & 0 & 0 & \frac{1}{2} & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & -\frac{1}{2} & 0 & 0 & 0 & 0 & 0 & -\frac{1}{2} & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{2} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{2} & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{2} & 0 & 0 & 0 & -\frac{1}{2} & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{2} & 0 & 0 & 0 & -\frac{1}{2} & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{2} & 0 & 0 & 0 & \frac{1}{2} & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{2} & 0 & 0 & 0 & -\frac{1}{2} & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{4} & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{4} & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{4} & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{4} & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{4}
\end{array}\right]$$

Figure 4.2: Parity check matrix of LDLC lattice based on array codes, constructed with $d = 4$, $p = 5$ and the generator sequence $h = \{1, \frac{1}{2}, \frac{1}{2}, \frac{1}{2}, \frac{1}{2}, \frac{1}{2}\}$. An the elements with low row degree were multiplying by $c_1 = 4$, $c_2 = 2$ and $c_{3,\ldots,n} = 1$.

1. Construct the $dp \times dp$ modified array code as equation (4.7), using $j = k = d$.

2. Change the non-zero elements of $\mathbf{H}$ with elements in the generator sequence $\mathbf{h}$, where each block i in block row l contains h_i, with random sign.

3. For each block row i in the inverse generator $\mathbf{H}$, multiply it by $\frac{1}{c_i}$.

In Fig. 4.3 the block diagram of the construction algorithm is shown. The advantage of the construction algorithm is it low memory requirements since there is no need to storage additional elements, and it is a deterministic

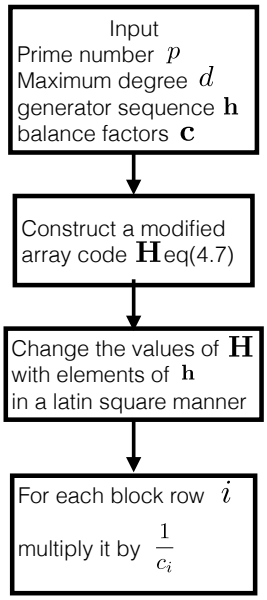

Figure 4.3: Block diagram for the construction of the inverse generator matrix for the array LDLC lattices

construction, i.e. no pseudorandom construction.

4.3 Minimum distance of the array LDLC lattices

By using the generator matrix $\mathbf{G} = \mathbf{H}^{-1}$ we can give some bound in the minimum distance of the array LDLC lattice.

An advantage of the construction of the inverse generator for array LDLC lattices is that it is possible to compute the generator matrix by using block matrix inversion (see appendix A.1).

Lemma 4.3.1 *The square minimum distance d^2_{min} is upper bounded by the*

square norms τ^2 of the basis vectors in the generator matrix $\mathbf{G}$.

$$d^2_{min} \leq \min(\tau^2_1, \tau^2_2, \ldots, \tau^2_n). \tag{4.13}$$

Since the generator matrix gives an upper bound for the square minimum distance, it is desire that $\tau^2_1 = \tau^2_2 = \cdots = \tau^2_n$. But these in practice is hard to achieve, it is not guarantee that all basis vectors in the generator matrix has the minimum norm.

Lets derivate the square minimum distance upper bound for some finite dimensions array LDLC lattices, in order to show the process of the derivation. Consider $d = 3$ and let the generator sequence be $\mathbf{h} = \{1, h_a, h_a\}$, where $h_a < 1$, to simplify the derivation. The array LDLC lattice inverse generator with these characteristics is:

$$\mathbf{H} = \begin{pmatrix} \frac{1}{c_1}\mathbf{I} & \frac{h_a}{c_1}\mathbf{I} & \frac{h_a}{c_1}\mathbf{I} \\ \mathbf{0} & \frac{1}{c_2}\mathbf{I} & \frac{h_a}{c_1}\mathbf{P} \\ \mathbf{0} & \mathbf{0} & \frac{1}{c_3}\mathbf{I} \end{pmatrix}, \tag{4.14}$$

by using block matrix inversion [55] (see appendix A.1), the generator matrix $\mathbf{G} = \mathbf{H}^{-1}$ is:

$$\mathbf{G} = \begin{pmatrix} c_1\mathbf{I} & -h_a c_2\mathbf{I} & (h_a c_3\mathbf{I} - h_2^2 c_3\mathbf{P}) \\ \mathbf{0} & c_2\mathbf{I} & -h_a c_3\mathbf{P} \\ \mathbf{0} & \mathbf{0} & c_3\mathbf{I} \end{pmatrix}, \tag{4.15}$$

Let $\tau^2_1, \tau^2_2, \tau^2_3$ be the square norms of the block vectors in the generator matrix $\mathbf{G}$, and assuming that $\tau^2_1 = \tau^2_2 = \tau^2_3$ (a desire condition in the generator matrix). The square norms are:

$$\tau_1^2 = c_1^2, \tag{4.16}$$

$$\tau_3^2 = c_2^2(h_a^2 + 1) \tag{4.17}$$

and

$$\tau_3^2 = c_3^2(h_a^4 + 2h_a^2 + 1). \tag{4.18}$$

In addition, by Lemma 4.2.2 the balance factors needs to satisfy that :

$$c_1 \cdot c_2 \cdot c_3 = 1, \tag{4.19}$$

in order to have $\det(\mathbf{G}) = 1$. Now with the system of the equation the solutions are:

$$d_{min}^2 \leq c_1^2 = \sqrt[3]{(h_a^2 + 1)(h_a^4 + 2h_a + 1)}. \tag{4.20}$$

$$c_2^2 = \frac{c_1^2}{(h_a^2 + 1)}. \tag{4.21}$$

$$c_3^2 = \frac{c_1^2}{(h_a^4 + 2h_a + 1)}. \tag{4.22}$$

Theorem 4.3.2 *The square minimum distance upper bound for array LDLC lattices is independent of the size of the block, i.e is independent of the prime number p. The square minimum distance upper bound only depends on the generator sequence* $\mathbf{h}$ *and the balance factors* $\mathbf{c}$.

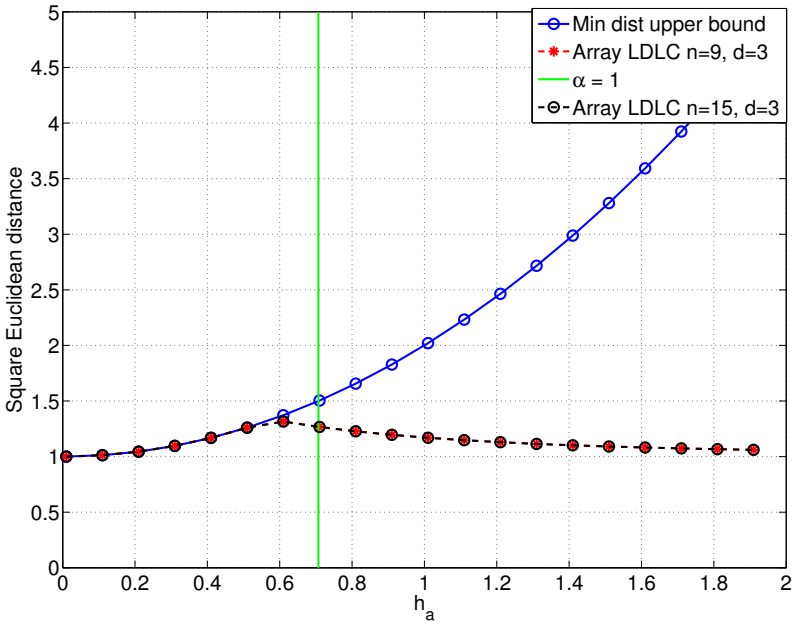

Figure 4.4: Comparison of the minimum distance for the array LDLC lattices, for $p = 3$ and $d = 3$ with generator sequence $\mathbf{h} = \{1, h_a, h_a\}$.

Fig. 4.4 the minimum distance upper bound for $d = 3$ is shown. A comparison for the minimum distance in terms of h_a for some finite length array LDLC lattices, constructed using $p = 3$ and $p = 5$, is performed. The exact minimum distance for the finite length array LDLC lattices is obtained by performing sphere decoding (see section 2.4.3). In Fig. 4.4 can be seen that the minimum distance upper bound goes to infinity as the value of h_a increases, but for lattices the minimum distance can not go to infinity, because there exist a lattice point at some finite minimum distance. In addition it is shown which values satisfy $\alpha \leq 1$ (a necessary condition for designing LDLC lattices).

For most of the case when $\alpha \leq 1$ the derived square minimum distance upper bound is a very accurate approximation of the true square minimum distance. The biggest square minimum distance for the array LDLC lattices with degree $d = 3$ is $d^2_{min} = 1.3151$, and is when the generator sequence $\mathbf{h} = \{1, 0.61, 0.61\}$, balance factors $\mathbf{c} = \{1.7114, 1, 0.8537\}$ and $\alpha = 0.7442$.

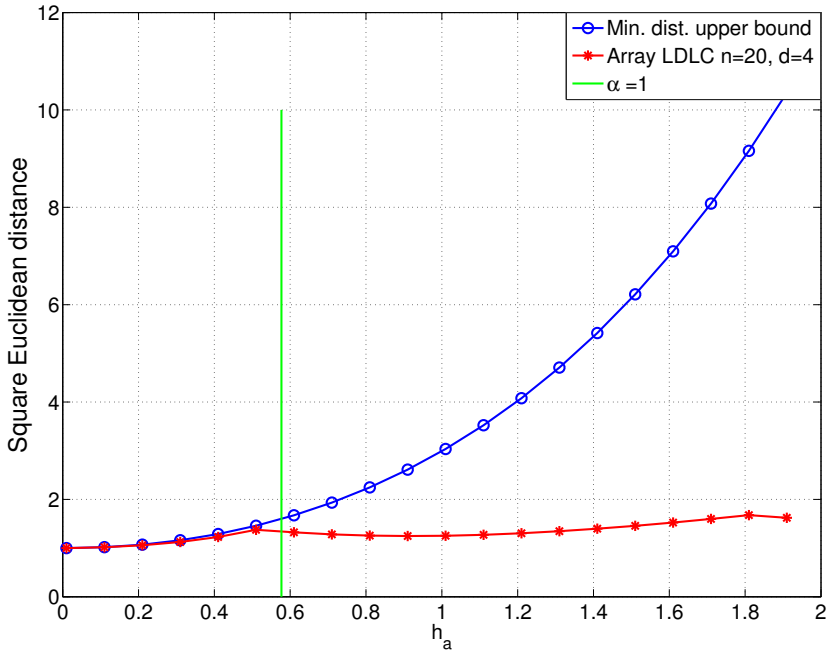

Figure 4.5: Comparison of the minimum distance for the array LDLC lattices, for $p = 5$ and $d = 4$, with generator sequence $\mathbf{h} = \{1, h_a, h_a, h_a\}$.

Now consider the array LDLC lattices with the following characteristics: the degree be $d = 4$ and generator sequence $\mathbf{h} = \{1, h_a, h_a, h_a\}$, by block matrix inversion the generator matrix $\mathbf{G}$ is given by:

$$\mathbf{G} = \begin{pmatrix} c_1\mathbf{I} & -h_a c_2\mathbf{I} & (h_a c_3\mathbf{I} - h_a^2 c_3\mathbf{P}) & (h_a^2 c_4\mathbf{P}^2 - h_a^2 c_4\mathbf{P} + (h_a^2 c_4 + h_a c_4)\mathbf{I}) \\ \mathbf{0} & c_2\mathbf{I} & h_a c_3\mathbf{P} & h_a^2 c_4\mathbf{P}^3 + h_a c_4\mathbf{I} \\ \mathbf{0} & \mathbf{0} & c_3\mathbf{I} & h_a c_4\mathbf{P} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & c_4\mathbf{I} \end{pmatrix} \tag{4.23}$$

,

and by performing the same procedure as for $d = 3$ the square minimum distance upper bound is given by:

Table 4.2: Comparison of the exact minimum distance for different array LDLC lattices, when $\alpha \leq 1$.

d	p	$\mathbf{c}$	$\mathbf{h}$	d^2_{min}	α
3	3	$\{1.7114, 1, 0.8537\}$	$\{1, 0.61, 0.61\}$	1.3151	0.7442
3	5	$\{1.7114, 1, 0.8537\}$	$\{1, 0.61, 0.61\}$	1.3151	0.7442
4	5	$\{1.2080, 1.0761, 0.9586, 0.8025\}$	$\{1, 1, 0.51, 0.51, 0.51\}$	1.3375	0.7803

$$d^2_{min} = \sqrt{(h_a^2+1)(h_a^4+h_a^2+1)(h_a^6+2h_a^4+2h_a^2+(h_a^2+h_a)+1)}. \quad (4.24)$$

The comparison of the true square minimum distance and the proposed upper bound for the triangular array LDLC lattice of dimension $n = 20$, this imply that $d = 4$ and $p = 5$ is shown in Fig. 4.5. The biggest value for the square minimum distance is $d^2_{min} = 1.6778$, and can be get it when the generator sequence is $\mathbf{h} = \{1, 1.81, 1.81, 1.81\}$, balance factors $\mathbf{c} = \{3.0206, 1.4636, 0.7078, 0.3189\}$ and $\alpha = 3.8283$. Even if the condition on α is not satisfy is still a lattice (not a LDLC lattice). For the case when α is satisfied the biggest square minimum distance is $d^2_{min} = 1.3375$ and is when the generator sequence is $\mathbf{h} = \{1, 0.51, 0.51, 0.51\}$, balance factors $\mathbf{c} = \{1.2080, 1.0761, 0.9586, 0.8025\}$ and $\alpha = 0.7803$.

The same analysis can be done for $d > 4$, but he generator matrix become larger and does not yield insight, and the true minimum distance become harder to compute. In Fig. 4.6 a comparison various minimum distance upper bound in terms of the degree d is shown. The minimum distance upper bound increases in terms of the degree distribution d and not in terms of the prime number p.

Table 4.2 shows the array LDLC lattices with the biggest square minimum distance, when $\alpha \leq 1$, for different finite dimensions. When $\alpha > 1$ it is clearly that the generator sequence is cumbersome as d gets bigger.

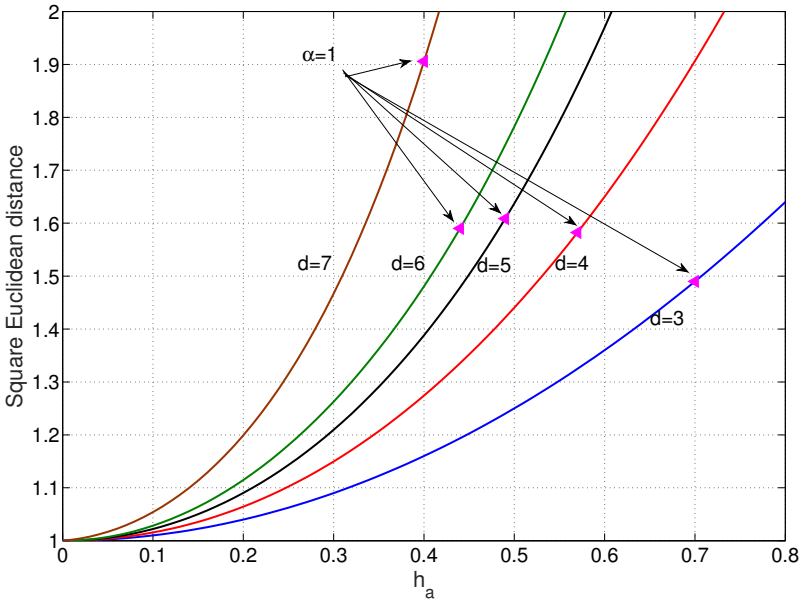

Figure 4.6: Minimum distance upper bound for triangular array LDLC lattices for various degree d.

4.4 Numerical results

Various array LDLC lattices based were simulated for the AWGN channel. This dissertation evaluate the performance of the array LDLC lattices using the symbol error rate (SER) in terms of volume-to-noise ratio (VNR), it is consider a symbol error when one element of the estimated lattice point $\hat{\mathbf{x}}$ is not equal to the transmitted lattice point $\mathbf{x}$, i.e $\hat{x}_i \neq x_i$.

The array LDLC lattice inverse generator matrices $\mathbf{H}$ was generated using the parameters shown in Table 4.2. And two cases were simulated, the first one is when the inverse generator matrix has a triangular structure and the second case is when the inverse generator is a full matrix. For the full matrix case the balance factors are not needed, the balance factors are $\mathbf{c} = 1$. For all inverse generator matrices are normalize to have $|\det(\mathbf{H})| = 1$ for a fair comparison.

Figure 4.7 shows the comparison between array LDLC lattice inverse gen-

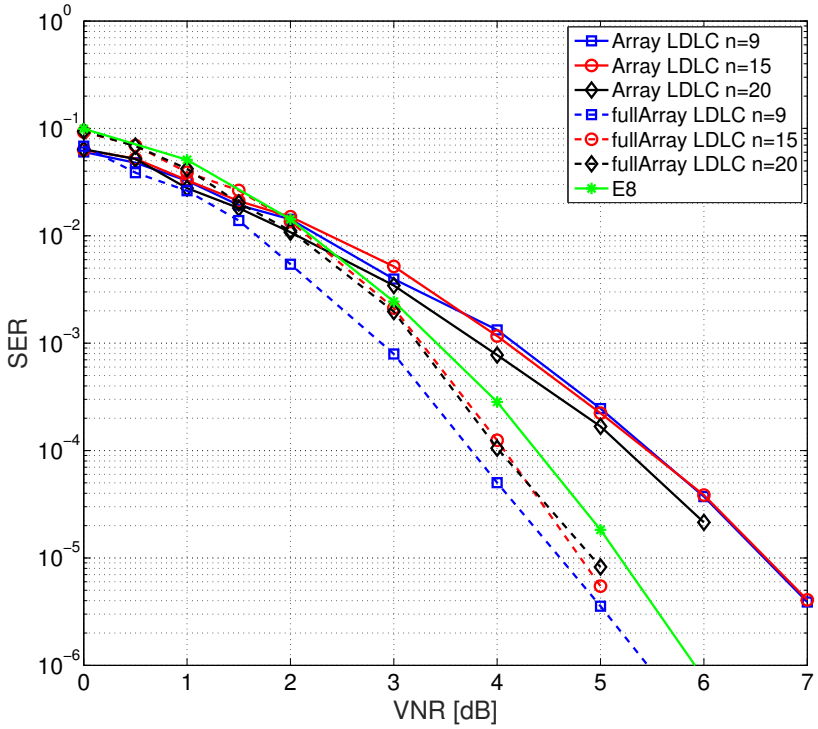

Figure 4.7: Comparison between the triangular inverse generator matrices and the full inverse generator matrices for small dimensional array LDLC lattices.

erator be a full matrix and be triangular. For all the simulated array LDLC lattices the full matrix case present a better performance than the triangular version, this is due to the protection of each element, it is clear that having a full array LDLC lattice, all elements are equally protected. In addition a comparison with the well know E_8 is perform, where the full array LDLC lattice presents a gain of 0.6dB in terms of the VNR compare to the E_8 lattice.

In addition a comparison with the triangular LDLC lattice inverse generator given in [30] was done. The construction parameters are shown in Table 4.4 and the constellation size is shown in Table 4.3. For the array LDLC lattices the balance factors are $c = \{4, 2, 1, 1, 1, 1, 1\}$ (for $n = 91$, $n = 49$) and $c = \{4, 2, 1, 1\}$ (for $n = 20$) were used. The inverse generator matrices were

Table 4.3: Row/column degree and constellation size

Degree	constelation size
1	2
2	2
3	4
4	4
5	8
6	8
7	8

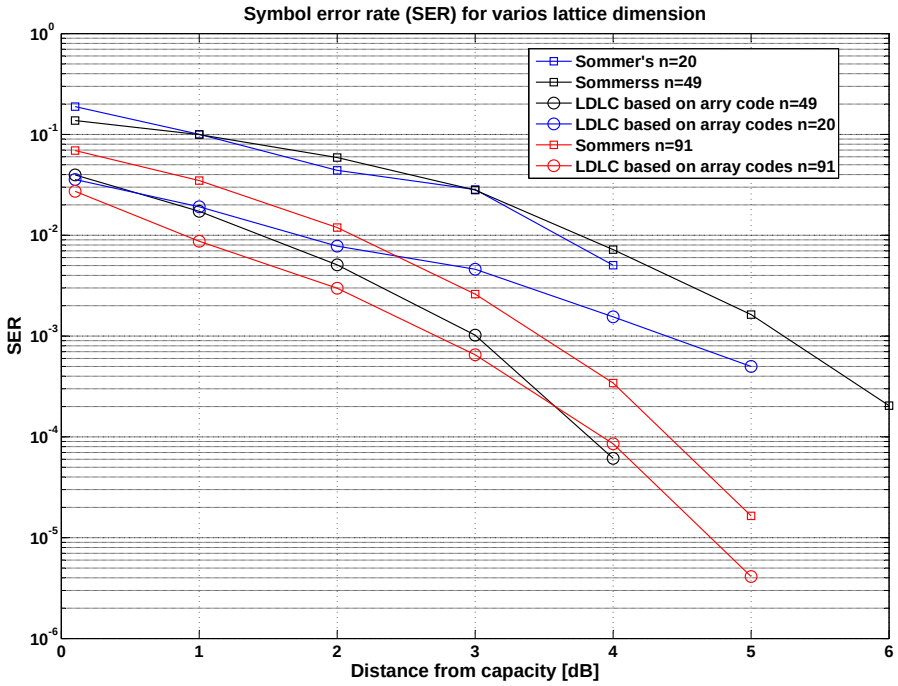

Figure 4.8: Simulation results for various lattice dimensions

further normalized to get $\det(\mathbf{H}) = 1$, as the definition of LDLC lattices, for a fair comparison.

Figure 4.8 shows the comparison between the array LDLC lattices and Sommer's construction [30]. For the array LDLC lattices presents a gain of

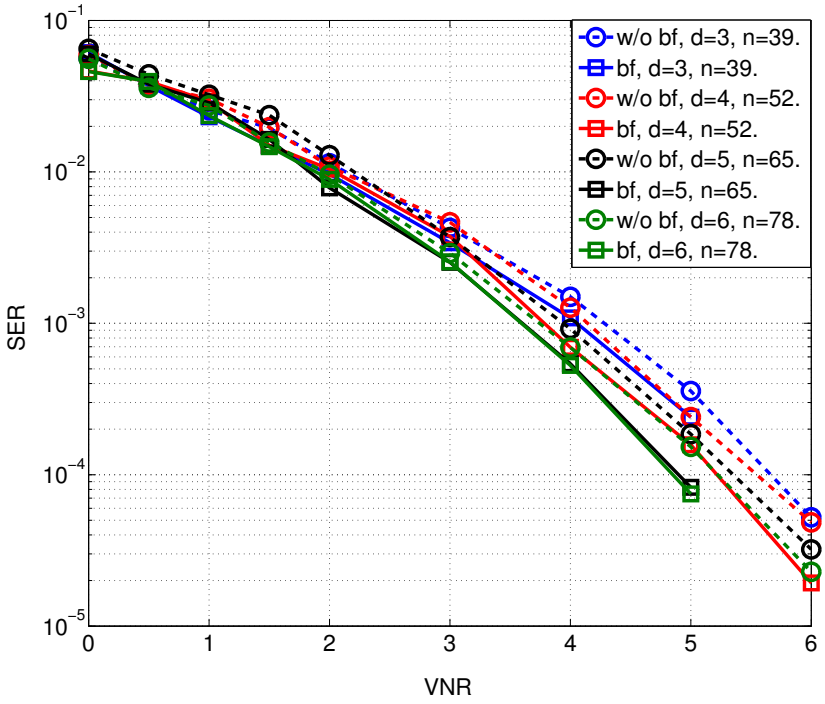

Figure 4.9: Comparison when the balance factors (bf) are used and when are not, for various lattice dimension with parameter: degree $d = 3, 4, 5, 6$, generator sequence of the form $\mathbf{h} = \{1, h_a, h_a, \ldots, h_a\}$, with $h_a = 0.4$, and prime number $p = 13$

0.3 dB close to lattice capacity for dimension $n = 91$.

Finally Figure 4.9 shows the symbol error rate (SER) comparison for triangular array LDLC lattices, when the balance factors are used and when are not used them. These triangular array LDLC lattices was constructed be setting the prime number $p = 13$ and using different degree distribution $d = 3, 4, 5, 6$ for triangular array LDLC lattice dimension $n = 39, 52, 65, 78$ respectively, the generator sequence is of the form $\mathbf{h} = \{1, h_a, \ldots, h_a\}$. The performance of the lattice was evaluated using the three/two Gaussian parametric decoder algorithm (See chapter 5). The use of the balance factor improve the performance about 0.4dB in terms of the VNR for dimension $n = 78$. Table 4.4 shows the values for balance factors $\mathbf{c}$ and the minimum

Table 4.4: Balance factors **c** and minimum distance upper bound for different triangular LDLC lattices.

d	**c**	d^2_{min}
3	$\{1.0770, 1, 0.9285\}$	1.16
4	$\{1.1288, 1.0481, 0.9731, 0.8686\}$	1.2742
5	$\{1.1782, 1.0939, 1.0157, 0.9446, 0.8087\}$	1.3881
6	$\{1.2168, 1.1298, 1.0490, 0.9903, 0.8905, 0.7864\}$	1.4896

distance upper bound of the simulated triangular array LDLC lattices.

Increasing the dimension n by increasing the prime number p does not give good performance, since the number of low degree row/column increases, but increasing the dimension of the lattice by its degree distribution d causes an improved on the performance.

4.5 Conclusion

LDLC lattices provide close to optimal, practical lattice codes. However methods to construct these codes present a high computational complexity. In this dissertation a new method to construct a $dp \times dp$ LDLC inverse generator matrices is shown. The new construction is based on LDPC array codes, accordingly with the base the new construction is called "array LDLC lattices". The array LDLC lattices takes the characteristics of LDPC array codes, that are deterministic construction (not pseudorandom) to have low computational complexity, sparseness in order to require low storage, 4-cycle free to eliminate tedious computations under the BP decoding and triangular structure to simplify encoding and shaping operations.

The inverse generators matrices for array LDLC lattices are defined by four parameters: the maximum degree d, a prime number p, the generator sequence **h** (non-zero elements in the matrix) and the balance factors **c**.

The structure that the inverse generator and the generator matrices of the array LDLC lattices is suitable for derivate an upper bound on the square minimum distance. The dissertation shows how to derivate the square mini-

mum distance upper bound. The derived upper bound is a close approximation for the true square minimum distance for $\alpha \leq 1$, this results can be use as a guide on how to choose the values for the generator sequence $\mathbf{h}$.

Chapter 5

Three/Two Gaussian Parametric Decoder

In this chapter, a proposed LDLC decoding algorithm for low density lattice codes (LDLC) is described. This decoding algorithm has only one parameter, that is the number of Gaussians needed for the approximation, two or three Gaussians, and correspondingly is called the "Three/Two Gaussian Parametric Decoder". In addition,variable node operation in the three/two Gaussian parametric decoder is suitable for analysis, performed by evaluating the Kullback-Leiber divergence. The three/two decoding algorithm has similar performance to previous decoders with a lower computational complexity.

5.1 Introduction

On Section 3.3.2 a description on the full complexity quantize decoding algorithm for LDLC lattices was described. The quantized decoding passes infinite Gaussian mixtures among the variable nodes and check nodes. And the operations that takes pace in the variable node and in the check node are made over continuous functions. This algorithm is not suitable for implementation due to the high computational complexity, like the memory required and processing time.

Different authors has proposed parametric decoding algorithms for LDLC lattices, such that the Gaussian mixture reduction (GMR) algorithm [2] and

the table search algorithm [31] (see Section 3.4). These algorithms presents a week approximation or tedious operations.

Having a parametric algorithm implies that the Gaussians functions are only represented by its mean and variance. Let define a Gaussian function be:

$$\mathcal{N}(w;\mu,v) = \frac{1}{\sqrt{2\pi v^2}} e^{\frac{-(w-\mu)^2}{2v^2}} \tag{5.1}$$

with mean μ and variance v. First lets described some operations over Gaussian mixtures.

5.2 Operations on Gaussian Mixtures

This section describes the product of Gaussian mixtures and the moment matching approximation.

Let $f(w)$ be a mixture of N Gaussians,

$$f(w) = \sum_{i=1}^{N} c_i \mathcal{N}(w; m_i, v_i), \tag{5.2}$$

with mean m_i, variance v_i and mixing coefficients $c_i > 0$ and $\sum_{i=1}^{N} c_i = 1$ for $i = 1, 2 \ldots, N$.

5.2.1 Product over Gaussian mixtures

The product of two Gaussian mixtures $f(w) = \sum_{i=1}^{N} f_i(w)$ and $g(w) = \sum_{j=1}^{M} g_j(w)$ is $f(w) \cdot g(w)$. The product of two components $f_i(w) = c_1 \mathcal{N}(w; m_1, v_1)$ and $g_j(w) = c_2 \mathcal{N}(w; m_2, v_2)$ is a single Gaussian $s(w) = c\mathcal{N}(w; m, v)$ with mean m, variance v and mixing coefficient c given by:

$$\frac{1}{v} = \frac{1}{v_1} + \frac{1}{v_2}, \tag{5.3}$$

$$\frac{m}{v} = \frac{m_1}{v_1} + \frac{m_2}{v_2} \tag{5.4}$$

and

$$c = \frac{c_1 c_2}{\sqrt{2\pi(v_1 + v_2)}} e^{-\frac{(m_1 - m_2)^2}{2v_1 + 2v_2}}. \tag{5.5}$$

The Gaussian product $f(w) \cdot g(w)$ is the mixture of the $N \cdot M$ products obtained using the pair-wise operation above.

5.2.2 Moment Matching Approxiamtion

The single Gaussian moment matching (MM) approximation, is the single-Gaussian approximation of a Gaussian mixture $f(w)$, given by (5.2), with a single Gaussian $q(w) = \mathcal{N}(w; m, v)$ which minimizes the Kullback-Leiber (KL) divergence between $f(w)$ and $q(w)$ [56, appdx. A]. The moment-matching approximation (MM) finds the single Gaussian $q(w)$ which has the same mean m and variance v as $f(w)$.

The mean m is given by:

$$m = \sum_{i=1}^{N} c_i m_i, \tag{5.6}$$

and variance v is given by:

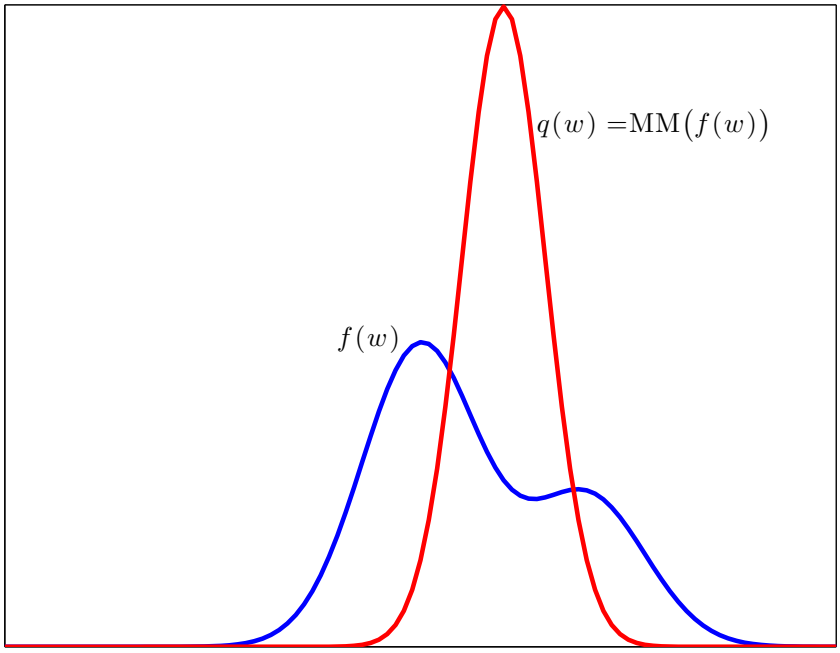

Figure 5.1: The single Gaussian moment matching (MM) approximation, red line, for the Gaussian mixture $f(w)$ in blue.

$$v = \sum_{i=1}^{N} c_i m_i^2 - \big(\sum_{i=1}^{N} c_i m_i\big)^2. \tag{5.7}$$

This operation is denoted as:

$$q(w) = \text{MM}\big(f(w)\big). \tag{5.8}$$

In Figure 5.1 the MM approximation is shown, the blue line is the Gaussian mixture $f(w)$ and in red line the MM approximation of $f(w)$.

5.3 Three/Two Gaussian approximation

In this section an approximation of the product of a single Gaussian and a Gaussian mixture is described, which is key for understanding the behavior of the three/two Gaussian parametric decoding algorithm. Analysis is performed by evaluating the Kullback-Leiber divergence.

Having a good approximation at the variable node is a key step for accurate performance in the parametric LDLC decoder. The approximation in the tails of the Gaussian function is very important. A poor approximation in the Gaussian messages causes errors to accumulate as the LDLC iterative decoding progresses.

Instead of calculating the periodic expansion over all integers, it is convenient to use a reduced number of integers. Since the periodic expansion takes place at the variable node, and due to multiplication with the channel message, the resultant periodic Gaussians that are far from the channel message have near-zero mixing coefficients, and can safely be ignored.

Consider the multiplication of the following to Gaussians. The single Gaussian $Y(w)$ represents the channel message and has mean m_a and variance v_a with $v_a = \sigma^2$. The Gaussian mixture $R(w)$ represents the check-to-variable node messages and is a periodic mixture of Gaussians with period $\frac{1}{|h|}$ and parameters m_c and v_c, and is given by:

$$R(w) = \sum_{i=-\infty}^{\infty} \mathcal{N}(w; m_c + \frac{i}{h}, v_c). \tag{5.9}$$

And let an approximation $\widetilde{R}(w)$ be:

$$\widetilde{R}(w) = \sum_{i \in \mathcal{B}} \mathcal{N}(w; m_c + \frac{i}{h}, v_c). \tag{5.10}$$

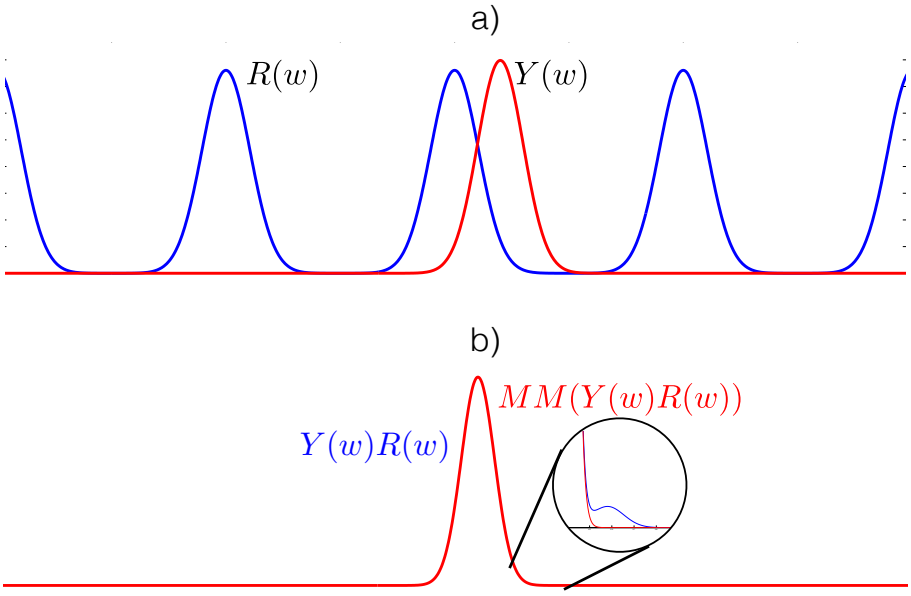

Figure 5.2: Multiplication of a Gaussian mixture $R(w)$ and a single Gaussian $Y(w)$. The true product $Y(w)R(w)$ and the single Gaussian moment matching (MM) approximation $\mathrm{MM}\big(Y(w)R(w)\big)$. This operation take place at the variable node.

that is the summation in (5.9) restricted to some finite integer set $\mathcal{B}$.

The idea is to approximate an infinite Gaussian mixture $Y(w)R(w)$ with $Y(w)\widetilde{R}(w)$, which consists of a finite number of Gaussians. In Fig. 5.2-a $Y(\mathbf{w})$, $R(\mathbf{w})$ are illustrated. In Fig. 5.2-b the true product $Y(w)R(w)$, the single Gaussian moment matching approximation $\mathrm{MM}\big(Y(w)R(w)\big)$ are shown. The true product and the MM approximation has a dissimilarity in its tails, and this dissimilarity can causes errors to accumulate as decoder progresses, i.e making a bad decision on the selection of the Gaussian.

5.3.1 Gaussian Neighbors Selection

Here two cases of $|\mathcal{B}| = 3$ and $|\mathcal{B}| = 2$ Gaussians neighbors near m_a are considered. Let the two-Gaussian set be $\mathcal{B} = \{b_1, b_2\}$ with $b_2 = b_1 + 1$, and the three-Gaussian set be $\mathcal{B} = \{b_0, b_1, b_2\}$, with $b_0 = b_1 - 1$ and $b_2 = b_1 + 1$.

For the two-Gaussian set, two integers are selected, one less than, and one greater than a non-integer estimate. Find b_1 such that:

$$\frac{b_1}{h} + m_c < m_a < \frac{b_1 + 1}{h} + m_c, \tag{5.11}$$

for $h > 0$. That is,

$$b_1 = \lfloor -h(m_c - m_a) \rfloor. \tag{5.12}$$

And in the three-Gaussian set the nearest Gaussian and its two nearest neighbors are chosen. That is:

$$b_0 = b_1 - 1, \tag{5.13}$$

$$b_1 = \lceil h(m_c - m_a) \rfloor, \text{ and} \tag{5.14}$$

$$b_2 = b_1 + 1. \tag{5.15}$$

The resulting mixture is:

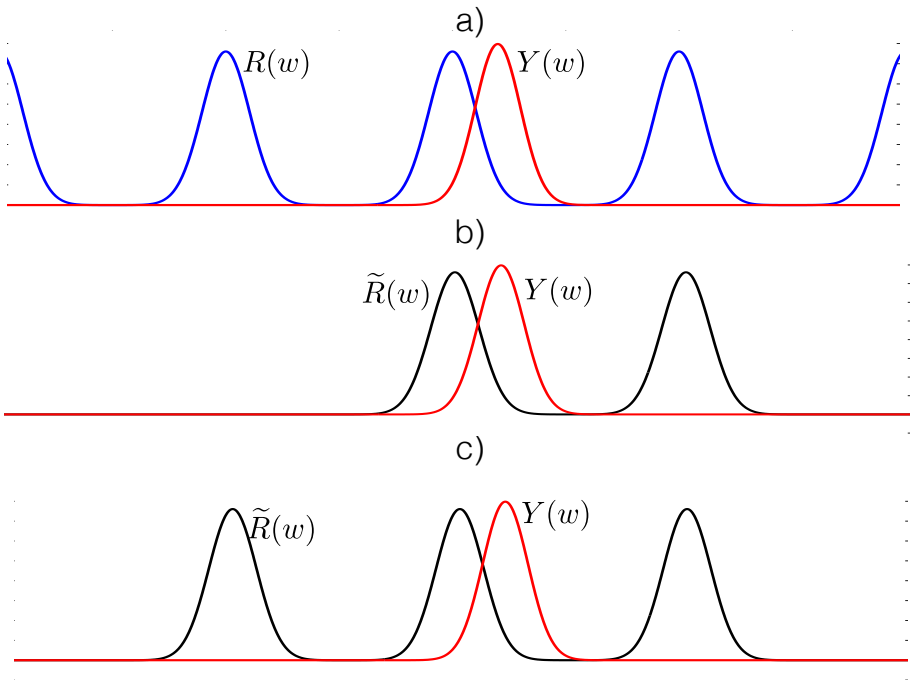

Figure 5.3: Proposed approximation for the infinite Gaussian mixture $R(w)$, by selecting Gaussians that are close the single Gaussian $Y(w)$. b) The two-Gaussian approximation. c) The three-Gaussian approximation.

$$\widetilde{R}(w) = \mathcal{N}(w; \frac{b_1}{h} + m_c, v_c) + \mathcal{N}(w; \frac{b_2}{h} + m_c, v_c), \tag{5.16}$$

for the two-Gaussian set. And for three-Gaussian set it is:

$$\widetilde{R}(w) = \mathcal{N}(w; \frac{b_0}{h} + m_c, v_c) + \mathcal{N}(w; \frac{b_1}{h} + m_c, v_c) + \mathcal{N}(w; \frac{b_2}{h} + m_c, v_c), \tag{5.17}$$

where $\widetilde{R}(w)$ is the approximation of $R(w)$.

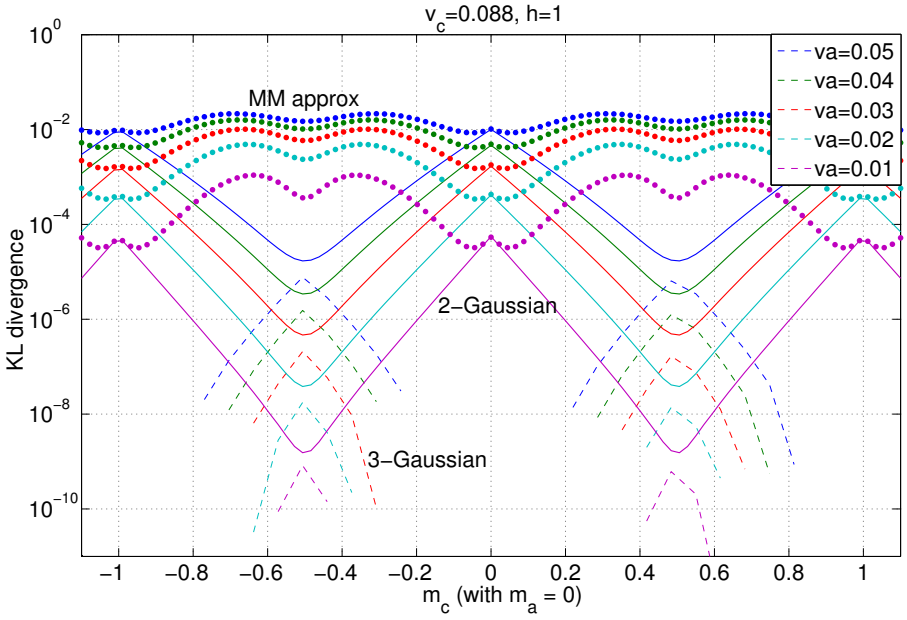

Figure 5.4: KL divergence for the dominant message ($h = 1$), for single Gaussian approximation (dot line), two Gaussian approximation (solid line) and three Gaussian approximation (dash line). For $v_c = 0.088$ correspond to an early iteration.

Figure 5.3 shows the idea of how to selecting Gaussians in $R(w)$ which are closest to the single Gaussian $Y(w)$. Figure 5.3-b shows the selection for the two-Gaussian set and Figure 5.3-c shows the selection for the three-Gaussian case.

5.3.2 Kullback-Leiber divergence

In this section the analysis using the Kullback-Leiber (KL) divergence is shown. By selecting a small number of Gaussians it is desired to minimize the KL divergence between $Y(w)R(w)$ and the approximation $Y(w)\tilde{R}(w)$. The KL divergence is given by

$$\int_{-\infty}^{\infty} Y(w)R(w) \log \frac{Y(w)R(w)}{Y(w)\widetilde{R}(w)} dw. \tag{5.18}$$

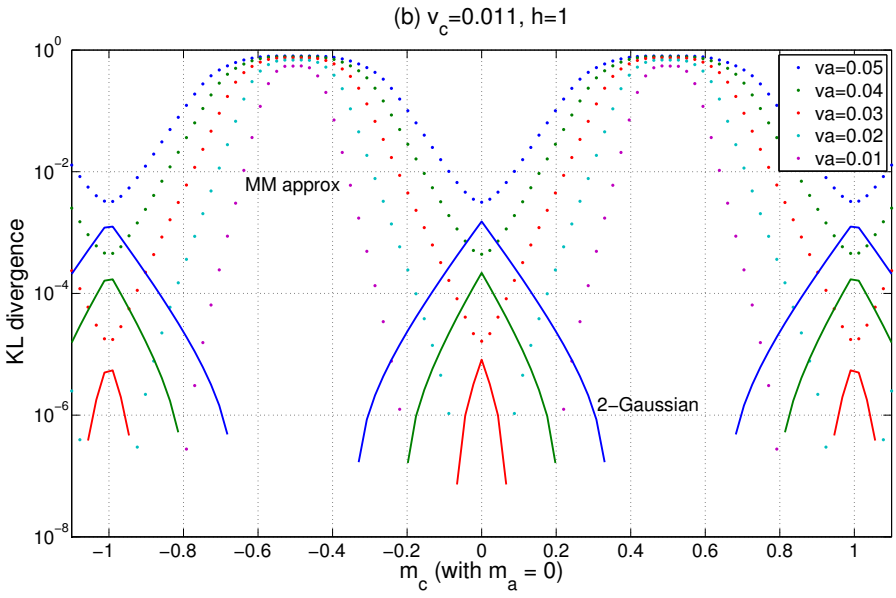

Figure 5.5: KL divergence for the dominant message ($h = 1$), for single Gaussian approximation (dot line), two Gaussian approximation (solid line). For $v_c = 0.011$ correspond to an intermediate iteration, and the single Gaussian is not accurate.

When $Y(w)R(w)$ and $Y(w)\widetilde{R}(w)$ are a mixture of Gaussians, by selecting the mean and variance close to each other will minimize the divergence. While the KL divergence between the two Gaussian mixtures is not analytically tractable in general, various approximations for the KL divergence for general Gaussians mixture models were proposed [57]. However, these approximations are not suitable for the Gaussian mixtures which occur during the message passing decoding of lattices, in the sense they do not give insight to the problem. Instead the KL divergence is evaluated numerically, this has the advantage of giving the exact value.

Figures 5.4–5.6 show the KL divergence for the single-Gaussian moment matching approximation (dashed-line), three-Gaussian approximation (solid-line) and two-Gaussian approximation (dotted-line), using typically observed

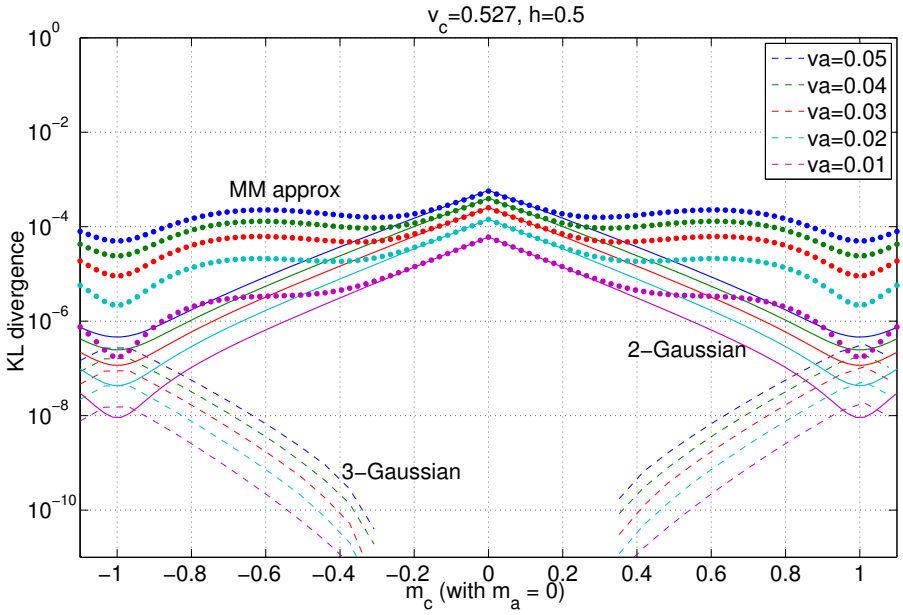

Figure 5.6: KL divergence for the non-dominant message $h_i = 0.5$ (early iteration), for single Gaussian approximation (dot line), two Gaussian approximation (solid line) and three Gaussian approximation (dash line)

values for v_a and v_c under LDLC decoding. All values for m_c are presented, but not all are equally likely because m_c is not uniformly distributed. The Kullback-Leiber divergence only depends on $m_c - m_a$, so by setting $m_a = 0$, the analysis depends only on m_c, v_c and v_a. The worse case is when $h = 1$.

Fig. 5.4 shows $v_c = 0.088$, corresponding to an early decoding iteration. Here, even the MM approximation has a KL divergence of less than 10^{-2}. Empirically it has been observed that a KL divergence of greater than 10^{-2} is a poor approximation for the proposed LDLC decoding algorithm. But KL divergence of less than 10^{-3} at least gives visually similar Gaussian functions.

Fig. 5.5 shows $v_c = 0.011$, corresponding to intermediate iterations of LDLC decoding, where the MM approxiamtion presents worse KL divergence. The KL divergence for two-Gaussian approximation is always less than 10^{-2} and the three-Gaussian approximation is even better. This suggests that the two-Gaussian approximation may be sufficient. The simulation

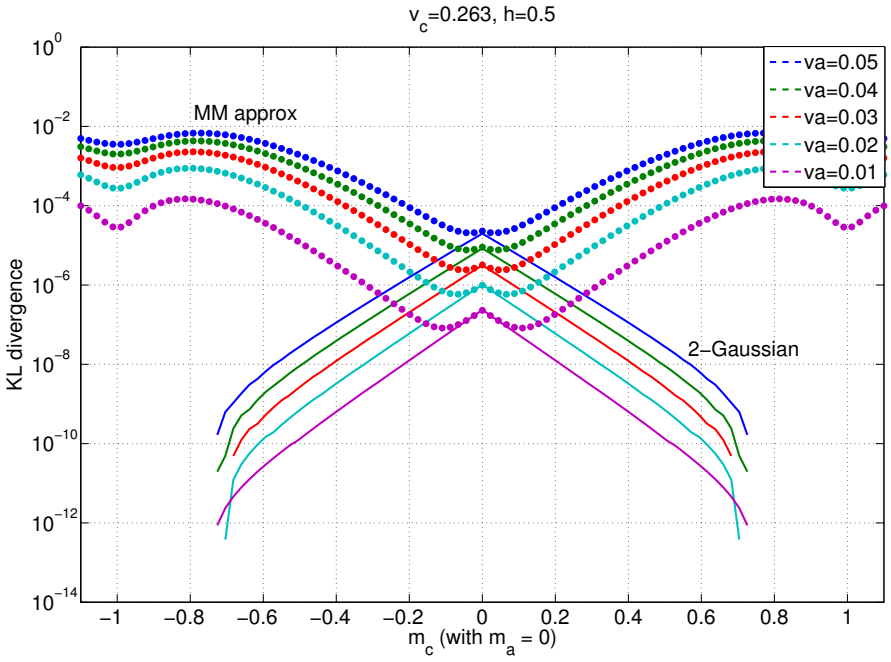

Figure 5.7: KL divergence for the non-dominant message $h_i = 0.5$ (middle iteration), for single Gaussian approximation (dash line), two Gaussian approximation (dot line) and three Gaussian approximation (solid line)

results will show that this is often true, but when the dimension is very large, e.g. $n = 10,000$, the three-Gaussian approximation is more reliable.

In Fig. 5.6 the MM approximation presents a good approximation for an early iteration for the non-dominant edge ($h_i < 1$). In Fig. 5.7 an intermediate iteration for the non-dominant edge, and Fig. 5.8 shows the KL divergence for the late iteration.

The message on the edge with the highest value in the generator sequence ($h_i = 1$; dominant edge) gives more reliable information during the message passing. For this reason the dominant edge required more accurate approximation.

Clearly, the accuracy of the approximation increases as the number of Gaussians increases. In order to have a low complexity parametric decoding

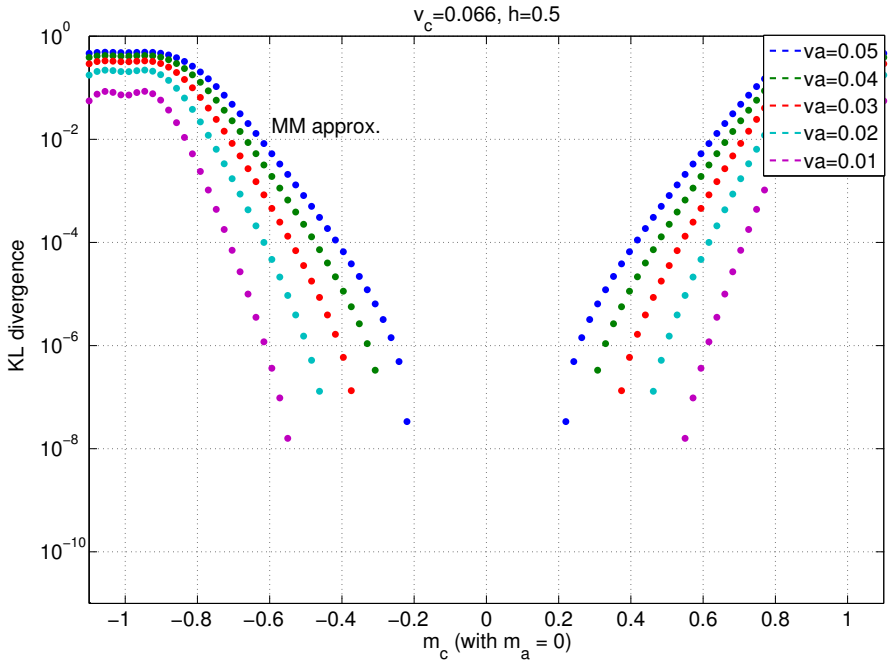

Figure 5.8: KL divergence for the non-dominant message $h_i = 0.5$ (late iteration), for single Gaussian approximation (dash line), the two Gaussian case and three Gaussian case are below 10^{-15}.

algorithm a single Gaussian approximation is desired. However, using the MM approximation is not accurate.

As is shown in Fig. 5.2, the MM approximation has difference in the tails of the Gaussian mixture which contribute significantly in the KL divergence. For this reason the MM approximation is not suitable for a good approximation.

Note that the approximation given in this section does not minimize the KL divergence, but is used because it is efficient for a decoder to implement. By maintaining the dominant Gaussians in the mixture, the approximation is a good one, as we have shown in this section.

5.4 Three/Two Gaussian Parametric Decoder

A parametric LDLC decoding algorithm is presented here. Internally at the variable node, messages are represented by mixtures of Gaussians, but externally only single Gaussians are used. There are two types of approximations used at the variable node: (1) the 3/2 Gaussian approximation from Section 5.3.1, and (2) the MM approximation used before variable node output; previous work [32] has shown that a single Gaussian message from the variable node to the check node is sufficient.

The variable-to-check message along edge k is a single Gaussian denoted $f_k(w)$. The check-to-variable message along edge k is a single Gaussian denoted $R_k(w)$. Single Gaussians are represented by its mean and variance. Thus, storage of variable-to-check messages requires $2 \cdot n \cdot d$ elements, and likewise for the check-to-variable messages. Internally at the variable node, messages are represented by mixtures of multiple Gaussians.

5.4.1 Description

For the AWGN channel, let the received message be

$$y_k(w) = \mathcal{N}(w; y_k, \sigma^2). \tag{5.19}$$

for $k = 1, 2, \ldots, n$ and σ^2 is the noise variance. The check node and variable node operations are:

- *Check Node*: The incoming messages are d single Gaussians $f_i(w) = \mathcal{N}(w; m_i, v_i)$. The output message $\tilde{p}_i(w)$ at the convolution step is a single Gaussian with mean $\tilde{m}_i$ and variance $\tilde{v}_i$ given by:

$$\widetilde{m}_i = -\frac{1}{h_i} \sum_{j=1}^{d\backslash i} h_j m_j \tag{5.20}$$

$$\widetilde{v}_i = \frac{1}{h_i^2} \sum_{j=1}^{d\backslash i} h_j^2 v_j. \tag{5.21}$$

The computation of $\widetilde{\mathbf{m}}$ and $\widetilde{\mathbf{v}}$ can be performed using a forward-backward recursion.

- *Variable node*: The messages coming from the check node are a single Gaussian $\mathcal{N}(w; \tilde{m}_i, \widetilde{v}_i)$, for $i = 1, 2, \ldots, d$. Then the expansion step (periodic with period $1/|h_i|$ if $\mathcal{B} \in \mathbb{Z}$) is approximated by:

$$\widetilde{R}_i(w) = \sum_{b \in \mathcal{B}} \mathcal{N}(w; m_i(b), \tilde{v}_i) \tag{5.22}$$

where the mean $m_i(b)$ of each Gaussian, for $b \in \mathcal{B}$, is given by:

$$m_i(b) = \widetilde{m}_i + \frac{b}{h_i}, \tag{5.23}$$

and the set $\mathcal{B}$ represents a subset of the integers, e.g. two or three. Rather than searching over all integers $\mathcal{B} \in \mathbb{Z}$, instead select two or three integers close to the channel message, as described in Section 5.3.1. The message $f_i(w)$ sent back to the check node is a single Gaussian approximated by:

$$p_i(w) = y_k(w) \prod_{j=1}^{d\backslash i} \widetilde{R}_j(w), \tag{5.24}$$

$$f_i(w) = \mathrm{MM}\Big(p_i(w)\Big), \tag{5.25}$$

where $y_k(w) = \mathcal{N}(w; y_k, \sigma^2)$ is the channel message, and $\widetilde{R}_j(w)$ is the approximation of the periodic expansion. To maintain low complexity, just a single Gaussian is used in the messages between variable and check nodes. The single Gaussian message to send to the check node is calculated by the moment matching algorithm.

A forward-backward recursion can be used to reduce the complexity to calculate message multiplications, as shown in the next section.

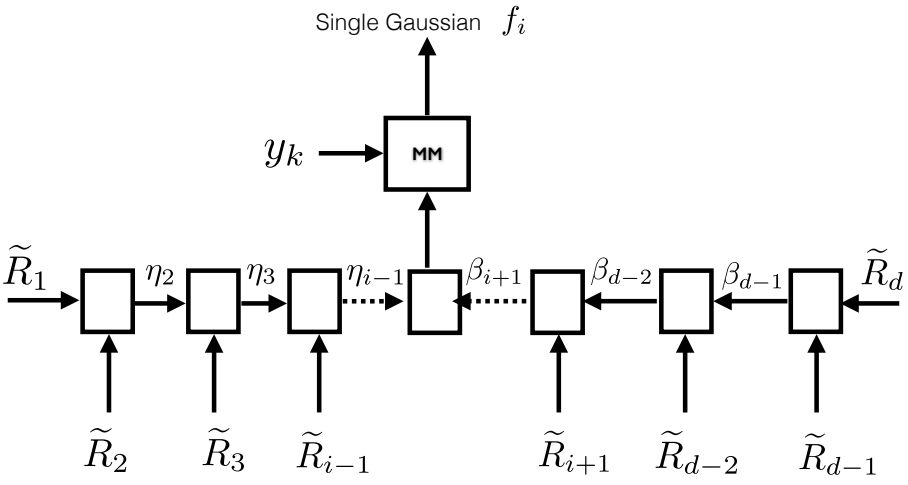

Figure 5.9: Forward-backward recursion at the variable node

5.4.2 Forward-backward recursion

Computing the output at the variable node $f_i(w)$, for $i = 1, 2, \ldots, d$ can be implemented by a forward-backward recursion. This recursion is distinct from previously described forward-backward approaches [2] in how the channel value y is handled — in the three/two Gaussian parametric decoding algorithm the channel message is multiplied last (although the channel message y_k is used to select the periodic Gaussians).

The forward-backward recursion is done as follows:

1. The forward recursion defined as:

$$\eta_i(w) = \eta_{i-1}(w) \cdot \widetilde{R}_i(w), \tag{5.26}$$

for $i = 2, 3, \ldots, d$, with $\eta_1(w)$ initialized as equal to $\widetilde{R}_1(w)$.

2. The backward recursion $\beta_i(w)$ is computed, for $i = d-1, d-2, \ldots, 1$, as:

$$\beta_{i-1}(w) = \beta_i(w) \cdot \widetilde{R}_{i-1}(w), \tag{5.27}$$

with $\beta_d(w)$ initialized as the approximation $\widetilde{R}_d(w)$.

3. Then combining the forward and backward recursion, we obtain:

$$\tilde{f}_i(w) = \eta_{i-1}(w) \cdot \beta_{i+1}(w). \tag{5.28}$$

4. Finally, the single Gaussian output of the variable node k is calculated using the moment matching approximation:

$$f_i(w) = \mathrm{MM}\Big(y_k(w) \cdot \tilde{f}_i(w)\Big). \tag{5.29}$$

The forward-backward process is illustrated in Fig. 5.9.

5.4.3 Complexity

The complexity of the three/two Gaussian parametric decoding algorithm is dominated by the forward and backward algorithm which is $\mathcal{O}(n \cdot t \cdot 2^{d-1})$ if messages are approximated by two Gaussians, and $\mathcal{O}(n \cdot t \cdot 3^{d-1})$ if three Gaussians are selected, in general the complexity is $\mathcal{O}(n \cdot t \cdot M^{d-1})$, where M is the number of Gaussians used in the approximation, t is the number of iterations, n is the lattice dimension and d is the degree of the LDLC inverse generator matrix. In practice, it would appear that $M \geq 4$, does not ssignificantly increase the performance.

For comparison, the complexity of the quantized BP decoding algorithm [1] is $\mathcal{O}(n \cdot t \cdot d \cdot \frac{L}{\Delta})$ where Δ is the probability density function resolution and L is the range length, and is dominated by a discrete Fourier transform. The complexity for [2] is $\mathcal{O}(n \cdot d \cdot t \cdot K^2 \cdot M^4)$, and is dominated by a moment matching algorithm. And for [31] the complexity is $\mathcal{O}(n \cdot d \cdot t \cdot K \cdot M^3)$, and is dominated by sorting and searching in tables, where K is the number of replications and M the number of Gaussian used in the mixtures. The common values which present similar performance are $n = 100$, $d = 5$, $L = 4$, $\Delta = \frac{1}{256}$, $M = 6$ and $K = 3$.

The parametric decoder presented in [31] requires storing a $n \cdot d$ list of M Gaussians.

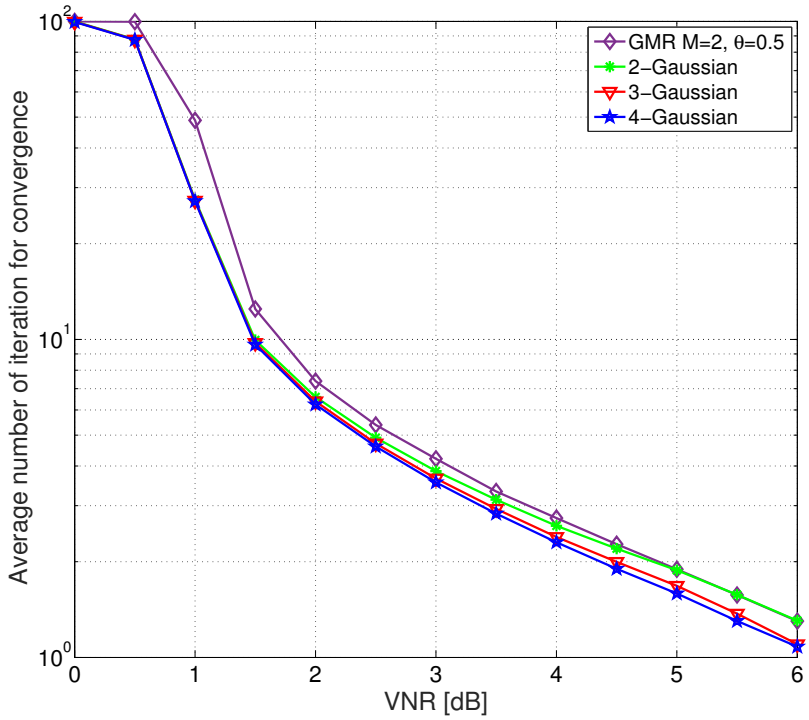

Figure 5.10: Average number of iterations required for decoder convergence in terms of the VNR, for LDLC dimension $n = 1000$ and degree $d = 7$.

In Fig. 5.10, the average number of iterations required for decoder convergence is shown. We took a sample of 1000 converged codewords (non-converging cases are ignored) and evaluated the mean of the number of iterations needed. The average number of iteration reduces as VNR increases. The use of three Gaussians does not reduce the average number of iterations. The GMR decoder for $M = 2$ and $\theta = 0.01$ required more iterations on average to converge. The three/two Gaussian parametric decoding algorithm looks about the same.

In addition, we also compare the computation time of the three/two Gaus-

sian parametric decoding algorithm with that of the GMR algorithm. The performance of the GMR decoder presented in [2] depends on two parameters, the Gaussian quadratic loss threshold θ and the maximum number of Gaussians M. A small value of θ presents a better approximation (and thus better performance) but the computational complexity increases. In Fig. 5.11 a computation time comparison for one iteration between the GMR decoder and the three/two Gaussian parametric decoding algorithm is shown. We simulated 1000 codewords for dimension $n = 1000$, at $VNR = 2$dB, and found the average time for one iteration. The proposed decoding algorithm presents a lower computation time for the values of θ when the GMR presents a good approximation [2]. The three/two Gaussian parametric decoding algorithm is independent of the value of θ. In fact, the only parameter is whether there are two or three Gaussians.

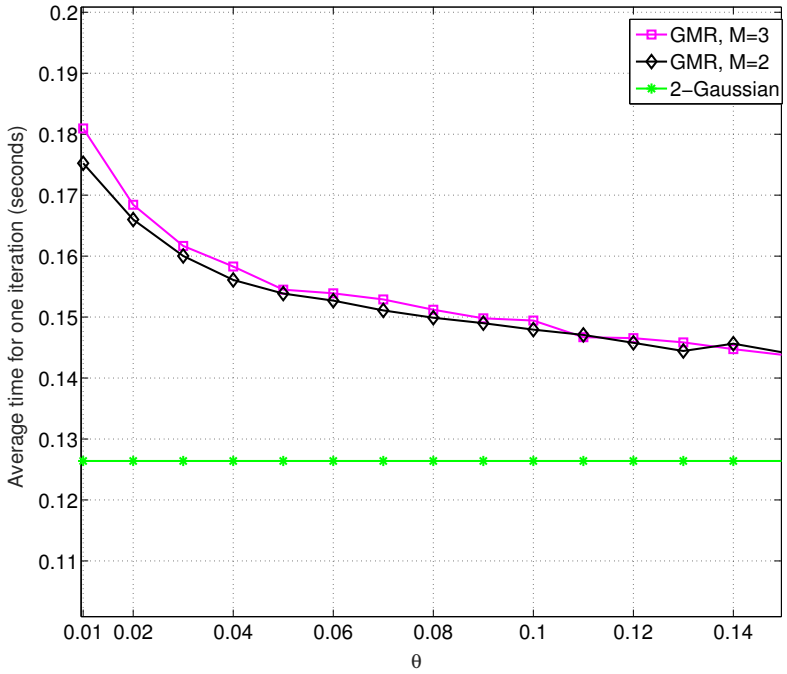

Figure 5.11: Time comparison between GMR algorithm [2], with $M = 2$ and $M = 3$, and the three/two Gaussian parametric decoding algorithm, for lattice dimension $n = 1000$ and $VNR = 2$.

The storage needed for the three/two Gaussian parametric decoding algorithm is $2 \cdot n \cdot d$ for the M-Gaussian approximation, since the message passed between check and variable nodes are single Gaussians, and these messages are parameterized by two values, the mean and the variance. Internally in the variable node the temporary storage needed is 2^{d-1} and 3^{d-1}, for the two-Gaussian and three-Gaussian approximation respectively. The use of a larger number of Gaussians as an approximation presents additional increment in the complexity, where the storage required in the variable node is 4^{d-1} or more.

5.4.4 Pseudocode of the Three/Two Gaussian Parametric Decoding Algorithm

In this section the pseudocode of the three/two Gaussian parametric decoding algorithm is given. And is as follows:

Input:

- The received message $\mathbf{y} = \mathbf{Gb} + \mathbf{z}$.
- the channel variance σ^2.
- The inverse generator $\mathbf{H}$.
- The maximum number of iterations $iter\_max$.

Output:

- The estimated information $\hat{\mathbf{b}}$.

Initialization:

1. At variable node k, for $k = 1, 2, \ldots, n$, send to all connected check nodes the message y_k and σ^2 from the channel.

Check node:

2. At the check node, every single Gaussian message $\widetilde{p}_i(w)$ to send it back to the variable node, for $i = 1, 2, \ldots, d$, is computed as in equation (5.20) and equation (5.21).

Variable node:

3. At the variable node k the ith message, for $i = 1, 2, \ldots, d$, calculate the message to send back to the check node. By selecting 2 or 3 Gaussians as describe in section 5.3.1.

4. The selecting mixtures are multiplied except the message i, to calculate

$$\prod_{j=1}^{d\backslash i} \widetilde{R}_j(w). \tag{5.30}$$

5. Then $p_i(w)$is calculated by multiplying the channel message $y_k(w)$ as in equation (5.24).

6. A moment matching is performed to send back to the check node a single Gaussian message $f_i(w)$.

7. Steps 2-6 are repeated until the maximum number of iterations $iter\_max$ is reached.

Final Decision:

8. The final estimate $\hat{x}_k$ is made by combining all messages at the variable node, and $\hat{x}_k$ is the mean of

$$\mathrm{MM}\Big(y_k(w) \prod_{i=1}^{d} \widetilde{R}_i(w)\Big). \tag{5.31}$$

9. Finally the received message is estimated by

$$\hat{\mathbf{b}} = \mathbf{H}\hat{\mathbf{x}}. \tag{5.32}$$

The benefits of the three/two Gaussian parametric decoding algorithm are that is almost parameter free, the only degree of freedom is the number of Gaussians M. The infinite Gaussian mixtures are approximated only with two or three Gaussians which are a good approximation. The messages between variable nodes and check nodes are single Gaussians and are only defined by two variables, the mean and variance, which required a low storage.

5.5 Numerical Results

In this section two types of numerical results for the proposed three/two Gaussian parametric decoding algorithm are shown:

- The noise thresholds for different LDLC lattices.
- The symbol error rate (SER) for dimension $n = \{100, 1000, 10000\}$ in terms of the volume-to-noise ratio (VNR).

5.5.1 Noise Thresholds

Noise thresholds are used to evaluate the performance of the three/two Gaussian parametric decoding algorithm. The noise threshold is the lowest VNR for which three/two Gaussian parametric decoding of asymptotically large-dimensional lattice converges. Performing exact density evolution would require the joint distribution for the mean and variance of the messages sent between the variable node and check node, which are the parameters used for the messages in the decoding algorithm. The evaluation of exact density evolution for two variables is computationally demanding. Instead we perform Monte Carlo density evolution which has been used for non-binary low density parity check codes [58].

We consider a lattice construction as described in [1], with generator sequence $\mathbf{h} = \{1, w, \ldots, w\}$, where w is given by:

$$w = \sqrt{\frac{\alpha}{d-1}}, \tag{5.33}$$

and α is defined as:

$$\alpha = \frac{\sum_{i=2}^{d} h_i^2}{h_1^2}, \tag{5.34}$$

and $\alpha \leq 1$ is a necessary condition for exponential convergence of the belief-propagation decoder [1].

The evaluation of Monte Carlo density evolution is similar to the one given in [48], and is as follows. The data pool consist of two types of elements: ones with label 1 denoted by $P_{(1)}$ and others with label w denoted by $P_{(w)}$. $P_{(1)}$ consists in $N_{(1)} = 10^6$ messages and $P_{(w)}$ consists of $N_{(w)} = (d-1) \cdot 10^6$ messages, i.e.

$$P_{(1)} = \{(m_1, v_1), \ldots, (m_{N_{(1)}}, v_{N_{(1)}})\} \tag{5.35}$$

$$P_{(w)} = \{(m_1, v_1), \ldots, (m_{N_{(w)}, N_{(w)}})\}, \tag{5.36}$$

where m and v denote the mean and variance respectively.

The messages (m_l, v_l), for all $l = 1, \ldots, N_{(1)}$ ($l = 1, \ldots, N_{(w)}$ for the w label message), are initialized as follows. The noise variance σ^2 is assigned to v_l, and m_l is initialized with the received symbol generated from $\mathcal{N}(0, \sigma^2)$, since the all zero codeword (lattice point) is assumed.

At each half iteration the variable/check node input consists of one element of $P_{(1)}$ and $d-1$ elements from $P_{(w)}$. The check and variable nodes are computed as shown in Section 5.4.1 and stored in an output pool. The

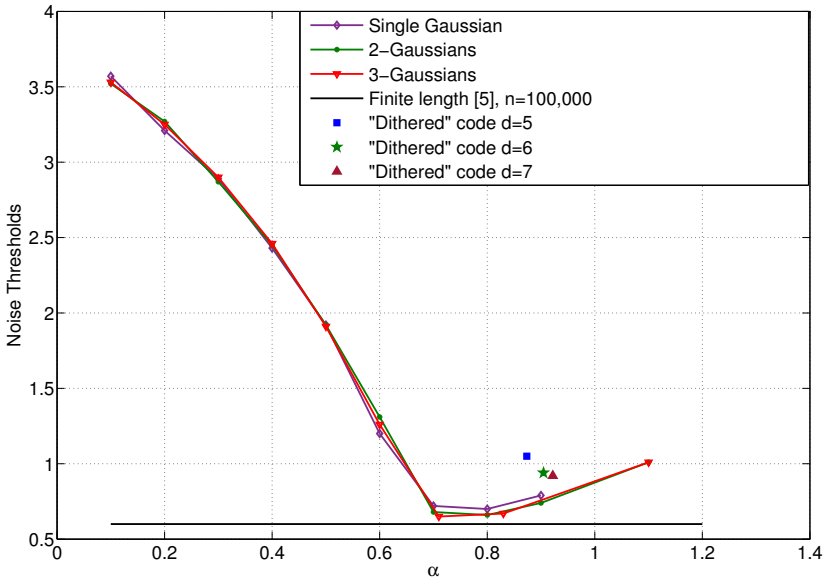

Figure 5.12: Noise thresholds, measured in distance from capacity, for three/two Gaussian decoder and the single Gaussian decoder, for various LDLC lattices with parameters $d = 7$ and α.

output pool becomes the input pool for the next half iteration. The mean of the variable-to-check node messages for the w labeled edge was used to check for convergence. When the mean of all $v_i \in P_{(w)}$ samples fell below to 0.001, within 50 iterations, then convergence was declared.

The noise thresholds, obtained using the three/two parametric decoding algorithm and the single-Gaussian decoder [48], are shown in Fig. 5.12 for several values of α and $d = 7$. In addition, the noise thresholds for the sequence $\{\frac{1}{2.31}, \frac{1}{3.17}, \frac{1}{5.11}, \frac{1}{7.33}, \frac{1}{11.71}, \frac{1}{13.11}, \frac{1}{17.55}\}$ normalized with $\frac{1}{2.31}$ to obtain $\mathbf{h} = \{1, 0.73, 0.45, 0.32, 0.20.18, 0.13\}$ for $d = 5$, $d = 6$ and $d = 7$ as proposed in [1] are shown; this LDLC is denoted as the "dithered" code.

In Fig. 5.13 a closer look at the noise thresholds, shows that the use of three Gaussians to approximate the messages does not significantly reduce the noise threshold compared with approximating with two Gaussians. The noise thresholds for the three/two Gaussian parametric decoding algorithm are reduced by 0.05 dB compared with the noise thresholds for the

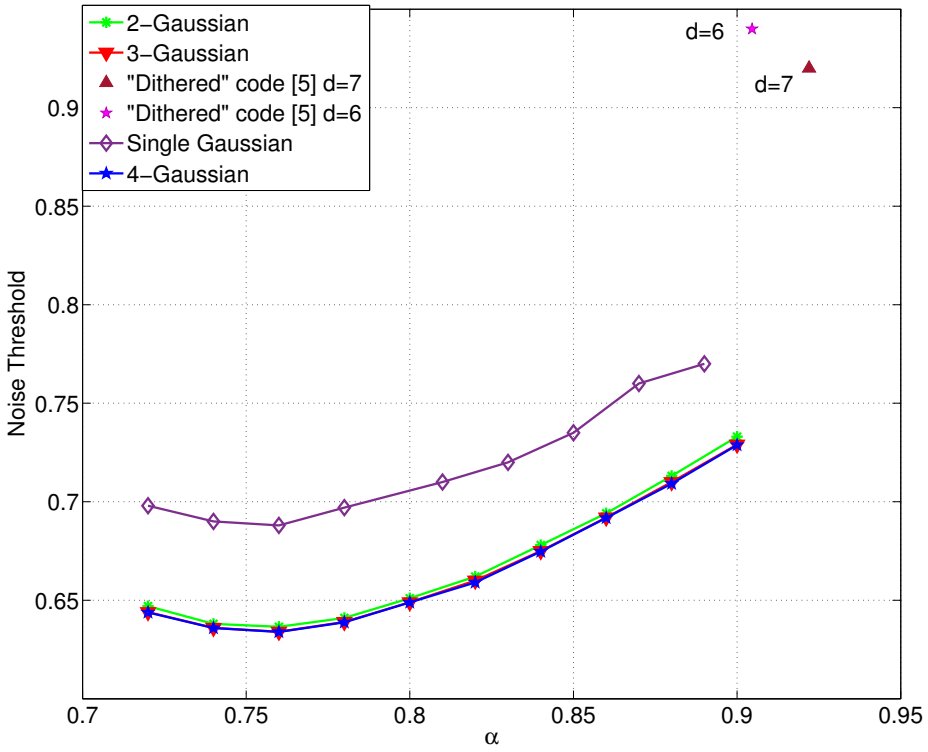

Figure 5.13: Noise thresholds details, measured in distance from capacity, for the Three/Two Gaussian parametric decoder and the single Gaussian decoder, for various LDLC lattices with parameters $d = 7$ and α.

single-Gaussian decoder in [48]. Interestingly, the noise thresholds for the considered lattice construction are slightly better than the dithered code.

By evaluating the noise threshold it is confirm that for having a good performance $\alpha \leq 1$. Also an approximation of the best value of α is derived, where $0.75 < \alpha < 0.77$. In addition Figure 5.12 and Figure 5.13 can be use as guide for LDLC lattice design, by selecting the values of the generator sequence $\mathbf{h}$ in terms of α.

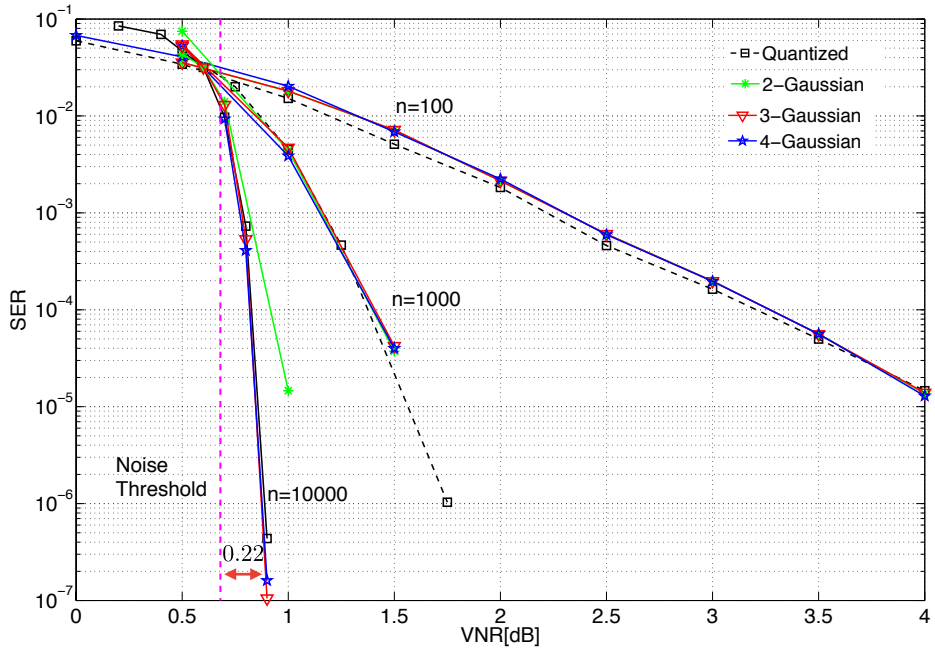

Figure 5.14: Comparison in terms of the number of Gaussians M used in the approximation. The comparison is made in terms SER vs VNR. For LDLC lattices with dimension $n = 100$, $n = 1000$, $n = 10000$.

5.5.2 Finite-length results

To evaluate the three/two Gaussian parametric decoding algorithm the all zeros codeword was simulated over the AWGN channel. The inverse generator matrix was generated as in [1], with the generator sequence $\mathbf{h} = \{1, \frac{1}{\sqrt{d}}, \ldots, \frac{1}{\sqrt{d}}\}$, with this choice of generator sequence we have $\alpha = \frac{d-1}{d}$. The inverse generator matrices $\mathbf{H}$ were further normalized in order to have $\sqrt[n]{|\det(\mathbf{H})|} = 1$ for fair comparison.

Different lattice dimensions $n = 100$, $n = 1000$ and $n = 10000$ were simulated, and the inverse generator H has degree $d = 3$ for dimension $n = 100$, and $d = 7$ for dimension $n = 1000$ and $n = 10000$. The symbol error rate (SER; a symbol error is $\hat{b}_i \neq b_k i$, for $i = 1, 2, \ldots, n$) as a function of volume-to-noise ratio (VNR). It is consider that lattice capacity is when $VNR = 0$dB.

In the three/two Gaussian parametric decoding algorithm two cases are investigated. The first one is how the performance is affected based on the number of Gaussians M selected. The second case is how the three/two decoding algorithm compared with the quantize decoding algorithm [1] and the GMR decoding algorithm [2].

In Figure 5.14 the comparison between the number of Gaussians M selected as the approximation is shown. It is shown that for dimension $n \leq 1,000$ selecting two-Gaussian $M = 2$ are enough for achieve the performance of the quantize algorithm. But when $n = 10,000$ three-Gaussians are enough. In addition a comparison with four-Gaussian was simulated, increasing to $M = 4$ did not give any visible improvement. The constructed LDLC for $d = 7$ has $\alpha = 0.8571$ and its noise threshold is 0.68dB. The gap to the noise threshold for the three/two Gaussian parametric decoding algorithm is 0.22dB at SER of 10^{-7} when the approximation is done with three-Gaussians or four-Gaussians for lattice dimension $n = 10,000$.

In Figure 5.15 the comparison between the quantized decoding algorithm quantize decoding algorithm [1], the GMR decoding algorithm [2] and the three/two Gaussian parametric decoding algorithm is shown. In the quantize algorithm the probability density function resolution is $\delta = \frac{1}{256}$ and each probability density function was represented with a vector of $L = 1024$. For the GMR decoding algorithm the number of Gaussians in the mixture are set to $M = 3$ and $M = 10$, and the threshold $\theta = 0.1$ and $\theta = 0.01$ respectively. Finally for the three/two Gaussian parametric decoding algorithm two cases are simulated when the approximation is done with two-Gaussians and three-Gaussians.

The three/two Gaussian parametric decoding algorithm perform nearly as well as the quantize algorithm. The GMR decoding algorithm present an error floor this is due to the number of Gaussians using in mixture. The three/two parametric decoding algorithm performs as well as the GMR decoding algorithm but with a significant reduction on computational complexity. In addition the three/two Gaussian parametric decoding algorithm does not present an error floor at least at SER of 10^7 at dimension $n = 10,000$, which is a good property for practical applications.

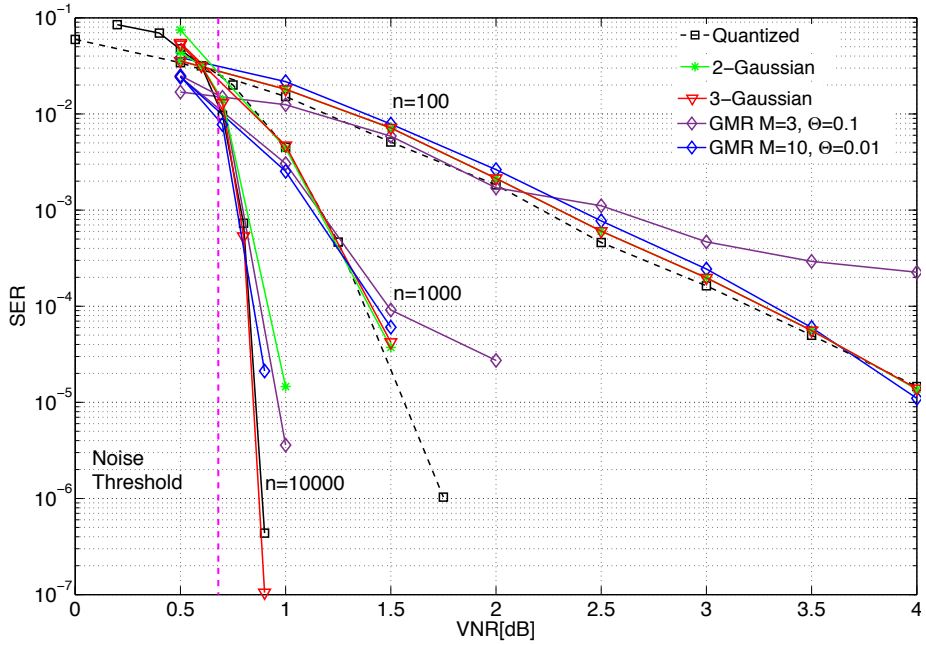

Figure 5.15: Comparison in terms of the number of Gaussians M used in the approximation. The comparison is made in terms SER vs VNR. For LDLC lattices with dimension $n = 100$, $n = 1000$, $n = 10000$.

5.6 Conclusion

In this chapter the three/two Gaussian parametric decoding algorithm for low density lattice codes (LDLC) was presented.

The three/two Gaussian parametric decoding algorithm approximates the Gaussian mixture with only two or three Gaussians which are close to the channel value. These selections are more accurate approximations than the single Gaussian approximation, in terms of the Kullback-Leiber divergence.

The advantages of the three/two Gaussian parametric decoding algorithm are:

1. Reduces the noise thresholds compared to single Gaussian decoders.

2. Has lower complexity, not only in terms in computation time per iteration but also in the average number of iterations to converge. Convergence in terms of volume-to-noise ratio for LDLC is presented for the first time in the literature. The noise threshold analysis and the average number of iteration presented in this work are a guideline for LDLC lattice design.

3. For dimension $n = 1,000$ the approximated with two Gaussian performs nearly similar compare to the best-known decoding algorithm, and for dimension $n = 10,000$ three Gaussian approximation is needed.

4. Has only one parameter that is the number of Gaussians (three or two) needed for approximating the messages.

These characteristics of the three/two Gaussian parametric decoder algorithm makes it attractive for different applications, such as the physical layer network that uses LDLC lattices [10] and/or crypto-systems which use LDLC lattices [59].

In addition it has been verify that $\alpha \leq 1$ for a good performance, and estimation of the best value of $0.75 < \alpha < 0.77$ was given. This results can be use as guide for LDLC lattice design.

Chapter 6

Conclusion

Modern information and communication systems are based on the reliable and efficient transmission of information. In wireless communication the noise is a real function, a coding scheme that can exploits the real algebra of the channel seems to be a more natural approach for data transmission.

On the other hand lattice codes have potential to become an efficient and practical coding scheme for the AWGN channel, since lattice codes are codes over the real numbers.

Recently a variety of lattices called low density lattices codes (LDLC) have been studied. The LDLC lattices can be decoded in an iterative manner, and have been reported that LDLC lattices attains 0.6dB from the unconstrained capacity at dimension $n = 100,000$.

In the LDLC belief propagation decoder the messages passed between nodes are continuous functions, and for implementation these functions need to be approximated, previous decoding algorithm are not suitable for implementation due to not accurate approximation and/or high computational complexity. In addition construction for LDLC present a high computational complexity, since its required to search and store a big number of data as the dimension increase.

In this dissertation an efficient method to construct LDLC lattices based on array codes and a parametric decoder for LDLC lattices were presented.

The proposed construction method generated a $dp \times dp$ LDLC inverse generator matrix, and is based on array codes, this construction is called "array LDLC lattices". The inverse generator of array LDLC lattices is defined by 4 parameters: the maximum degree d, a prime number p, the balance factors $\mathbf{c}$ and the value of the non zero elements (generator sequence) $\mathbf{h}$. The inverse generator matrix for array LDLC lattices is sparse, triangular and 4-cycle free, and the proposed construction has a low computational complexity. In addition a upper bound for the minimum distance was given.

The 4-cycle free property, by construction, eliminates tedious computations, and having a triangular structure is an important property that aids to perform shaping and encoding operations.

The proposed parametric decoding algorithm approximates the Gaussian mixture with only two or three Gaussians which are close to the channel value. These selections are more accurate approximations than the single Gaussian approximation, in terms of the Kullback-Leiber divergence. The proposed decoding algorithm is nearly parameter-free; the only parameter selection of interest is the number of Gaussians in the approximation, two or three Gaussians.

The numerical results show that for $n = 1,000$ the two-Gaussian approximation is the same as the full-complexity decoder. But when the dimension is $n = 10,000$, a three-Gaussian approximation is needed. Selecting a bigger number of Gaussians e.g $M = 4$, did not give any visual advantage in performance in terms of the symbol error rate.

In addition, this dissertation provided the noise thresholds, which can be use for lattice design, as a guide to select the generator sequence. The proposed construction and decoding algorithm are a step forward to a more practical algorithms for LDLC lattices.

Based on the achievements of this dissertation, several directions as the future work can be provided.

- Implementation of high dimensional LDLC lattices based on array codes, this implies the parameter p is large.

- In the triangular array LDLC lattice, investigate how to choose the most suitable values of generator sequence $\mathbf{h}$ and the balance factor $\mathbf{c}$ in order to increase the reliability of those symbols which are less protected. These parameters are related with the convergence under iterative decoder and the performance in terms of symbol error rate.

- In the array LDLC lattices increase the degree distribution in order to have less elements with a low degree. One possibility is by adding elements of the generator sequence randomly, but keeping the 4-cycle free property.

- Applying shaping operations to the array LDLC lattices. This imply to have a finite constellation.

- Applying the proposed decoder algorithm to fading channels, not only in the point-to-point communication scheme, but also in the multi terminal network scheme. For multi terminal networks, we can exploit the benefits of lattice codes under a compute-and-forward relaying scheme [10].

- Extend the proposed LDLC decoding algorithm to the complex LDLC case. Other authors have preliminary study complex LDLC lattices [60].

- The Three/Two Gaussian parametric decoder can be occupied not only in the communications field but also in other fields where LDLC are occupied, e.g crypto-systems which use LDLC lattices [59].

Appendix A

Matrix operations

In this section we describe some matrix operation used in this work.

A.1 Block matrix inversion

A block matrix or partitioned matrix is a matrix which has been partitioned in a collection of smaller matrices or sub-matrices.

Lets define the matrix $\mathbf{H}$ be of the form:

$$\mathbf{H} = \begin{bmatrix} \mathbf{A} & \mathbf{B} \\ \mathbf{C} & \mathbf{D} \end{bmatrix} \tag{A.1}$$

.

The inverse $\mathbf{H^{-1}}$ is given by:

$$\mathbf{H^{-1}} = \begin{bmatrix} \mathbf{A}^{-1} + \mathbf{A}^{-1}\mathbf{B}(\mathbf{D} - \mathbf{C}\mathbf{A}^{-1}\mathbf{B})^{-1}\mathbf{C}\mathbf{A}^{-1} & -\mathbf{A}^{-1}\mathbf{B}(\mathbf{D} - \mathbf{C}\mathbf{A}^{-1}\mathbf{B})^{-1} \\ -(\mathbf{D} - \mathbf{C}\mathbf{A}^{-1}\mathbf{B})^{-1}\mathbf{C}\mathbf{A}^{-1} & (\mathbf{D} - \mathbf{C}\mathbf{A}^{-1}\mathbf{B})^{-1} \end{bmatrix}, \tag{A.2}$$

assuming that all sub-matrix inverse exist.

When the block matrix is in triangular form (e.g the array LDLC lattices) like:

$$\mathbf{H} = \begin{bmatrix} \mathbf{A} & \mathbf{B} \\ \mathbf{0} & \mathbf{D} \end{bmatrix}, \tag{A.3}$$

the inverse is given by:

$$\mathbf{H}^{-1} = \begin{bmatrix} \mathbf{A}^{-1} & \mathbf{A}^{-1}\mathbf{B}\mathbf{D}^{-1} \\ \mathbf{0} & \mathbf{D}^{-1} \end{bmatrix}. \tag{A.4}$$

A.2 QR factorization

The QR factorization is a decomposition of a matrix $\mathbf{A}$ in to a product of matrices $\mathbf{A} = \mathbf{QR}$, where $\mathbf{Q}$ is an orthogonal matrix and $\mathbf{R}$ is an upper triangular matrix. The QR factorization can be calculated using the Gram-Schmit process [55]

The QR factorization for the $n \times m$ matrix $\mathbf{A}$ has the following properties:

- if the $rank(\mathbf{A}) = m$, that factors $\mathbf{Q}$ and $\mathbf{R}$ are uniquely determined and the main diagonal of $\mathbf{R}$ are all positive.
- if $\mathbf{A}$ is a square matrix ($n = m$), then the matrix $\mathbf{Q}$ is unitary.

Appendix B

Pseudocode

In this appendix different pseudocode for different algorithms are described.

B.1 LLL-reduction algorithm

Algorithm 1: Subroutine: reduce

input : i such that $(i \leq m)$
while $j \leftarrow (i-1) > 0$ **do**
    $\mathbf{v}_i = \mathbf{v}_i - round(\mathbf{u}_i[j])\mathbf{v}_j$;
    $\mathbf{u}_i = \mathbf{u}_i - round(\mathbf{u}_i[j])\mathbf{u}_j$;
    $j = j - 1$

Algorithm 2: LLL-reduction algorithm

input : Basis $\{ \mathbf{v}_1, \mathbf{v}_2, \cdots, \mathbf{v}_n \}$
output: Reduce basis $\{ \mathbf{v}_1^*, \mathbf{v}_2^*, \cdots, \mathbf{v}_n^* \}$
for $i \leftarrow 1$ **to** n **do**
    $\mathbf{u}_i = 0$;
    $\mathbf{u}_i[i] = 1$;
    $\mathbf{v}_i^* = \mathbf{v}_i$;
    for $j \leftarrow 1$ **to** $i - 1$ **do**
        $\mathbf{u}_i[j] = (\mathbf{v}_i \cdot \mathbf{v}_j)/(\mathbf{v}_j^* \cdot \mathbf{v}_j^*)$;
        $\mathbf{v}_i^* = \mathbf{v}_i^* - \mathbf{u}_i[j]\mathbf{v}_j^*$;
    reduce(i)
while $i \leftarrow 1 < n$ **do**
    if $||\mathbf{v}_i^*||^2 \leq c||\mathbf{v}_{i+1}^*||^2$ **then**
        $i = i + 1$;
    else
        $\mathbf{v}_{i+1}^* = \mathbf{v}_{i+1}^* + \mathbf{u}_{i+1}[i]\mathbf{v}_i^*$;
        $\mathbf{u}_i[i] = (\mathbf{v}_i \cdot \mathbf{v}_{i+1}^*)/(\mathbf{v}_{i+1}^* \cdot \mathbf{v}_{i+1}^*)$;
        $\mathbf{u}_i[i+1] = 1$;
        $\mathbf{u}_{i;1}[i] = 1$;
        $\mathbf{u}_{i+1}[i+1] = 0$;
        $\mathbf{v}_i^* = \mathbf{v}_i^* - \mathbf{u}_i[i]\mathbf{v}_{i+1}^*$;
        $swap(\mathbf{u}_i, \mathbf{u}_{i+1})$ $swap(\mathbf{v}_i^*, \mathbf{v}_{i+1}^*)$ $swap(\mathbf{v}_i, \mathbf{v}_{i+1})$;
        for $k \leftarrow i + 2$ **to** n **do**
            $\mathbf{u}_k[i] = (\mathbf{v}_k \cdot \mathbf{v}_i)/(\mathbf{v}_i^* \cdot \mathbf{v}_i^*)$;
            $\mathbf{u}_k[i+1] = (\mathbf{v}_k \cdot \mathbf{v}_{i+1})/(\mathbf{v}_{i+1}^* \cdot \mathbf{v}_{i+1}^*)$;
        if $|\mathbf{u}_{i+1}[i]| > \frac{1}{2}$ **then**
            reduce(i+1)
        $i = \max(i - 1, 1)$

B.2 LDLC Construction

Algorithm 3: latin square LDLC lattice construction algorithm

input : Dimension n, degree d and generator sequence $\mathbf{h}$
output: Latin square LDLC inverse generator matrix $\mathbf{H}$
Initialization:
choose d random permutation on $\{1, 2, \ldots, n\}$;
Arrange the permutation in an $d \times n$ matrix $\mathbf{P}$;
$c \leftarrow 1$ `// Column index`;
$loopless\_col \leftarrow 0$ `// Num. of consecutive cols. without loops`
while $loopless\_col < n$ **do**
    $chn\_perm \leftarrow 0$;
    `// Search for 2-cycles and 4-cylces`;
    if *exist* $i \neq j$ *such that* $P_{i,c} = P_{j,c}$ **then**
        $chn\_perm \leftarrow i$;
    else
        if *exist* $c_0 \neq c$ *such that* $P_{:,c} = P_{:,c}$ *have two or more common elements* **then**
            $chn\_perm \leftarrow$ row index which have the first common element appear in column c of $\mathbf{P}$;
    if $chn\_perm \neq 0$ **then**
        Choose a random integer $1 \leq i \leq n$;
        Swap location c and i in row chn_perm;
        $loopless\_col \leftarrow 0$;
    else
        $loopless\_col \leftarrow loopless\_col + 1$;
    $c \leftarrow c + 1$;
    if $c > n$ **then**
        $c \leftarrow 1$;
`// Constructing the inverse generator` $\mathbf{H}$;
Initialize $\mathbf{H}$ as an $n \times n$ zero matrix;
for $i \leftarrow 1 : n$ **do**
    for $j \leftarrow 1 : d$ **do**
        $H_{P_{j,i},i} \leftarrow h_j \cdot random\_sign$

Bibliography

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

[2] B. Kurkoski and J. Dauwels, "Message-passing decoding of lattices using Gaussian mixtures," in *Info.Theory, 2008. ISIT 2008. IEEE International Symposium on*, July 2008, pp. 2489–2493.

[3] E. Rosnes, M. A. Ambroze, and M. Tomlinson, "On the minimum/stopping distance of array low-density parity-check codes," *IEEE Transactions on Information Theory*, vol. 60, no. 9, pp. 5204–5214, Sept 2014.

[4] C. Berrou, A. Glavieux, and P. Thitimajshima, "Near shannon limit error-correcting coding and decoding: Turbo-codes. 1," in *Communications, 1993. ICC '93 Geneva. Technical Program, Conference Record, IEEE International Conference on*, vol. 2, May 1993, pp. 1064–1070 vol.2.

[5] R. G. Gallager, *Low-Density Parity-Check Codes*. Cambridge, MA, USA: The M.I.T. Press, 1963.

[6] C. E. Shannon, "Probability of error for optimal codes in a Gaussian channel," *Bell System Technical Journal, The*, vol. 38, pp. 611–656, 1959.

[7] R. de Buda, "The upper error bound of a new near-optimal code," *Info. Theory, IEEE Trans. on*, vol. 21, pp. 441–445, Jul. 1975.

[8] U. Erez and R. Zamir, "Achieving $\frac{1}{2}\log(1+\text{SNR})$ on the AWGN channel with lattice encoding and decoding," *Info. Theory, IEEE Trans. on*, vol. 50, no. 10, pp. 2293–2314, Oct. 2004.

[9] T. Linder, C. Schlegel, and K. Zeger, "Corrected proof of de Buda's theorem," *Info. Theory, IEEE Trans. on*, vol. 39, no. 5, pp. 1735–1737, Sep 1993.

[10] 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, Oct 2011.

[11] B. Chen, D. Jayakody, and M. Flanagan, "Low-density lattice coded relaying with joint iterative decoding," *Communications, IEEE Transactions on*, vol. 63, no. 12, pp. 4824–4837, Dec 2015.

[12] Y. Wang and A. Burr, "Physical-layer network coding via low density lattice codes," in *Networks and Communications (EuCNC), 2014 European Conference on*, June 2014, pp. 1–5.

[13] N. Ferdinand, M. Nokleby, and B. Aazhang, "Low density lattice codes for the relay channel," in *Communications (ICC), 2013 IEEE International Conference on*, June 2013, pp. 3035–3040.

[14] G. Boole, *The Mathematical Analysis of Logic*. Cambridge University Press, 1847, cambridge Books Online. [Online]. Available: http://dx.doi.org/10.1017/CBO9780511701337

[15] K. Menger, "New foundations of projective and affine geometry," *Annals of Mathematics*, vol. 37, no. 2, pp. 456–482, 1936. [Online]. Available: http://www.jstor.org/stable/1968458

[16] G. Birkhoff, "On the combination of subalgebras," *Mathematical Proceedings of the Cambridge Philosophical Society*, vol. 29, pp. 441–464, 10 1933.

[17] ——, *Lattice Theory*, ser. American Mathematical Society colloquium publications. American Mathematical Society, 1940, no. v. 25, pt. 2.

[18] J. Leech and N. J. A. Sloane, *Sphere Packing and Error-Correcting Codes*. New York, NY: Springer New York, 1988, pp. 136–156.

[19] J. H. Conway and N. J. A. Sloane, *Sphere Packings, Lattices and Groups*, 3rd ed. New York, NY, USA: Springer-Verlag, 1999, iSBN 0-387-98585-9.

[20] G. D. Forney, Jr., "Multidimensional constellations—part II: Voronoi constellations," vol. 7, no. 6, pp. 941–958, Aug. 1989.

[21] G. D. Forney, "Coset codes. II. binary lattices and related codes," *IEEE Transactions on Information Theory*, vol. 34, no. 5, pp. 1152–1187, Sep 1988.

[22] ——, "Trellis shaping," *IEEE Transactions on Information Theory*, vol. 38, no. 2, pp. 281–300, March 1992.

[23] J. D. Gibson, T. Berger, T. Lookabaugh, D. Lindbergh, and R. L. Baker, *Digital Compression for Multimedia: Principles and Standards*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1998.

[24] M. Ajtai, "The shortest vector problem in l2 is np-hard for randomized reductions (extended abstract)," in *Proceedings of the Thirtieth Annual ACM Symposium on Theory of Computing*, ser. STOC '98. New York, NY, USA: ACM, 1998, pp. 10–19. [Online]. Available: http://doi.acm.org/10.1145/276698.276705

[25] J. Hoffstein, J. Pipher, and J. H. Silverman, *NTRU: A ring-based public key cryptosystem*. Berlin, Heidelberg: Springer Berlin Heidelberg, 1998, pp. 267–288. [Online]. Available: http://dx.doi.org/10.1007/BFb0054868

[26] N. Sommer, M. Feder, and O. Shalvi, "Low density lattice codes," in *Info. Theory, 2006 IEEE International Symposium on*. Seattle, WA, USA: IEEE, Jul. 2006.

[27] T. M. Cover and J. A. Thomas, *Elements of Information Theory (Wiley Series in Telecommunications and Signal Processing)*. Wiley-Interscience, 2006.

[28] G. Poltyrev, "On coding without restrictions for the AWGN channel," *Info. Theory, IEEE Trans. on*, vol. 40, no. 2, pp. 409–417, Mar. 1994.

[29] G. Forney, M. Trott, and S.-Y. Chung, "Sphere-bound-achieving coset codes and multilevel coset codes," *Information Theory, IEEE Transactions on*, vol. 46, no. 3, pp. 820–850, May 2000.

[30] N. Sommer, M. Feder, and O. Shalvi, "Shaping methods for low-denisty lattice codes," in *Proc. Info. Theory Workshop, 2009*, Oct. 2009, pp. 238–242.

[31] Y. Yona and M. Feder, "Efficient parametric decoder of low density lattice codes," in *Information Theory, 2009. ISIT 2009. IEEE International Symposium on*. Seoul, Korea: IEEE, Jun.-Jul. 2009, pp. 744–748.

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

[33] J. L. Fan, "Array codes as low-density parity check codes," in *Proc. Intern. Symp. on Turbo Codes*, 2000.

[34] R. Zamir, *Lattice Coding for Signals and Networks*. Cambridge University Press, 2014, iSBN 9780521766982.

[35] E. Viterbo and E. Biglieri, "A universal decoding algorithm for lattice codes," *in 14 Colloq. GRETSI (Juan-les-Pins, France)*, pp. 611–614, Sep 1993.

[36] A. K. Lenstra, H. W. Lenstra, and L. Lovász, "Factoring polynomials with rational coefficients," *Mathematische Annalen*, vol. 261, no. 4, pp. 515–534, 1982. [Online]. Available: http://dx.doi.org/10.1007/BF01457454

[37] E. S. Barnes and G. E. Wall, "Some extreme forms defined in terms of abelian groups," *Journal of the Australian Mathematical Society*, vol. 1, pp. 47–63, 8 1959.

[38] E. S. Barnes and N. J. A. Sloane, "New lattice packings of spheres," *Canad. J. Math*, pp. 117–130, 1983.

[39] A. Bos, J. H. Conway, and N. J. A. Sloane, "Further lattice packings in high dimensions," vol. 29, pp. 171–180, 12 1982.

[40] A. Sakzad, M.-R. Sadeghi, and D. Panario, "Construction of turbo lattices," in *Communication, Control, and Computing (Allerton), 2010 48th Annual Allerton Conference on*, Sept 2010, pp. 14–21.

[41] M.-R. Sadeghi, A. Banihashemi, and D. Panario, "Low-density parity-check lattices: Construction and decoding analysis," *Info. Theory, IEEE Trans. on*, vol. 52, no. 10, pp. 4481–4495, Oct 2006.

[42] Y. Yan and C. Ling, "A construction of lattices from polar codes," in *Information Theory Workshop (ITW), 2012 IEEE*, Sept 2012, pp. 124–128.

[43] J. Conway and N. Sloane, "Fast quantizing and decoding and algorithms for lattice quantizers and codes," vol. 28, no. 2, pp. 227–232, Mar. 1982.

[44] P. van Emde-Boas, *Another NP-complete partition problem and the complexity of computing short vectors in a lattice*, ser. Report. Department of Mathematics. University of Amsterdam. Department, Univ., 1981.

[45] B. Hassibi and H. Vikalo, "On the sphere-decoding algorithm i. expected complexity," *IEEE Transactions on Signal Processing*, vol. 53, no. 8, pp. 2806–2818, Aug 2005.

[46] J. Pearl, "Reverend bayes on inference engines: a distributed hierarchical approach," in *in Proceedings of the National Conference on Artificial Intelligence*, 1982, pp. 133–136.

[47] R. M. Tanner, "A recursive approach to low complexity codes," no. 5, pp. 533–547, 1981.

[48] B. Kurkoski, K. Yamaguchi, and K. Kobayashi, "Single-Gaussian messages and noise thresholds for decoding low-density lattice codes," in *Information Theory, 2009. ISIT 2009. IEEE International Symposium on.* Seoul, Korea: IEEE, Jun.–Jul. 2009, pp. 734–738.

[49] B. Chen, D. Jayakody, and M. Flanagan, "Cooperative relaying with low-density lattice coding and joint iterative decoding," in *Turbo Codes and Iterative Information Processing (ISTC), 2014 8th International Symposium on*, Aug 2014, pp. 254–258.

[50] P. F. M. Blaum and H. van Tilborg, *Hanbook of Coding Theory*. Elesevier, Amsterdam, 1998.

[51] J. Fan, "Constrained coding and soft iterative decoding," ser. The Springer International Series in Engineering and Computer Science, vol. 627. Springer US, 2001, pp. 195–203.

[52] M. Blaum and R. Roth, "New array codes for multiple phased burst correction," *Information Theory, IEEE Transactions on*, vol. 39, no. 1, pp. 66–77, Jan 1993.

[53] E. Eleftheriou and S. Olcer, "Low-density parity-check codes for digital subscriber lines," in *Communications, 2002. ICC 2002. IEEE International Conference on*, vol. 3, 2002, pp. 1752–1757 vol.3.

[54] B. M. Kurkoski and R. A. P. Hernandez, "Message variance convergence condition for generalizations of ldlc lattices," in *Information Theory Workshop (ITW), 2014 IEEE*, Nov 2014, pp. 20–24.

[55] R. A. Horn and C. R. Johnson, Eds., *Matrix Analysis*. New York, NY, USA: Cambridge University Press, 1986.

[56] C. E. Rasmussen and C. K. I. Williams, *Gaussian Processes for Machine Learning*. The MIT Press, 2005.

[57] J. R. Hershey and P. A. Olsen, "Approximating the kullback leibler divergence between gaussian mixture models," in *Acoustics, Speech and Signal Processing, 2007. ICASSP 2007. IEEE International Conference on*, vol. 4, April 2007, pp. IV–317–IV–320.

[58] M. C. Davey, "Error-correction using low-density parity-check codes," Ph.D. dissertation, University of Cambridge, 1999.

[59] R. Hooshmand, T. Eghlidos, and M. R. Aref, "Improving GGH public key scheme using low density lattice codes," *CoRR*, vol. abs/1503.03292, 2015. [Online]. Available: http://arxiv.org/abs/1503.03292

[60] Y. Yona and M. Feder, "Complex low density lattice codes," in *2010 IEEE International Symposium on Information Theory*, June 2010, pp. 1027–1031.

Publications

[1] **R. A. Parrao Hernandez** and B. M. Kurkoski, "The three/two Gaussian parametric LDLC lattice decoding and its analysis," *IEEE Transactions on Communications*, vol. 64, no. 9, pp. 36243633, Sept 2016.

[2] **R. A. Parrao Hernandez** and B. M. Kurkoski, "Design of triangular array LDLC lattices based on minimum distance bounds," submitted to *IEEE Communications Letter*. Submitted October 2016.

[3] **R. A. Parrao Hernandez** and B. M. Kurkoski, "The three/two Gaussian parametric LDLC lattice decoding algorithm and its analysis," *in Proceedings of the IEEE Information Theory Workshop*, (Jeju Island, Korea), October 2015.

[4] T. X. Nguyen, **R. A. Parrao Hernandez**, and B. M. Kurkoski, "Robust content-based image hash functions using nested lattice codes," *in International Workshop on Digital Forensics and Watermarking (IWDW 2015)*, (Tokyo, Japan), pp 406-407, October 2015.

[5] B. M. Kurkoski and **R. A. Parrao Hernandez**, "Message variance convergence condition for generalizations of LDLC lattices", *Information Theory Workshop (ITW), 2014 IEEE*, pp. 20-24, November 2014.

[6] **R. A. Parrao Hernandez** and B. M. Kurkoski, "Low Complexity Construction of Low Density Lattice Codes Based on Array Codes," *IEEE International Symposium on Information Theory and Its Applications (ISITA 2014)*, pp. 264-268,Melbourne, Australia, October, 2014

[7] **R. A. Parrao Hernandez**, Brian M. Kurkoski, "Construction Array LDLC Lattices"; *Proceedings of The 37th Symposium on Information*

Theory and Its Applications (SITA 2014), pp. 415-420, Toyama, Japan, December 2014.

[8] **R. A. Parrao Hernandez**, Brian M. Kurkoski, "Construction of Low Density Lattice Codes Based on Array Codes," *Proceedings of The 36th Symposium on Information Theory and Its Applications (SITA 2013)*, pp. 232-237, Ito, Japan, November, 2013.

[9] **R. A. Parrao Hernandez**, Brian M. Kurkoski, "Proportional Labelling Schemes for Multi-level Nested Lattice Codes" *Presentation at Workshop on Coding for Flash Memories*, Fukui, Japan, May, 2013