Title	電力制約通信のためのConstruction A及びD'格子
Author(s)	周, 帆
Citation	
Issue Date	2021-12
Type	Thesis or Dissertation
Text version	ETD
URL	http://hdl.handle.net/10119/17602
Rights	
Description	Supervisor:KURKOSKI, Brian Michael, 先端科学技術研究科, 博士

Construction A and D' Lattices for Power-Constrained Communications

Fan Zhou

Japan Advanced Institute of Science and Technology

Doctoral Dissertation

Construction A and D' Lattices for Power-Constrained Communications

Fan Zhou

Supervisor: Professor Brian M. Kurkoski

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

December 2021

Abstract

Lattices have the potential to provide reliable and power-efficient data transmission in the next-generation wireless communications. Information theory has provided remarkable insights into lattices and their applications for practical communication systems. The benefits of lattices for communications are: 1) high code rate 2) higher transmit power efficiency than conventional quadrature amplitude modulation constellations and 3) they form an essential component of compute-and-forward relaying, which provides high throughput and high spectral efficiency.

This dissertation addresses the designs and methods of nested lattice codes with good coding properties, a high shaping gain, and low-complexity encoding and decoding. Construction D' lattices based on quasi-cyclic low-density parity-check (QC-LDPC) codes are for coding and thus contribute to reliable data transmission. Construction A lattices based on convolutional codes are used to satisfy the channel power-constraint and provide shaping gain. These constructions have group property and provide high code rates.

Two encoding methods and a decoding algorithm for Construction D' coding lattices that can be used with shaping lattices for power-constrained channels are given. The multistage decoding algorithm uses successive cancellation by employing binary decoders of the component binary codes that form a Construction D' lattice. An indexing method for nested lattice codes is modified to avoid an integer overflow problem at high dimension. Convolutional code generator polynomials for Construction A lattices with the greatest shaping gain are given, as a result of an extensive search. It is shown that rate 1/3 convolutional codes provide a more favorable performance-complexity trade-off than rate 1/2 convolutional codes. For a given dimension, tail-biting convolutional codes have higher shaping gain than that of zero-tailed convolutional codes and truncated convolutional codes. A design for QC-LDPC codes to form Construction D' lattices is presented, where their parity-check matrices can be easily triangularized, thus enabling efficient encoding and indexing when formed a nested lattice code. The resulting QC-LDPC Construction D' lattices are evaluated using four shaping lattices: the E_8 lattice, the BW_{16} lattice, the Leech lattice and the best-found convolutional code lattice, showing a shaping gain of approximately 0.65 dB, 0.86 dB, 1.03 dB and 1.25 dB at dimension 2304.

Keywords: Construction D' lattices, Construction A lattices, nested lattice codes, QC-LDPC codes, shaping gain

Committee in charge:

Professor Brian M. Kurkoski, Chair
Professor Gregory Schwartzman
Professor Kiyofumi Tanaka
Professor Eiichiro Fujisaki
Professor Hideki Yagi

Acknowledgment

I gratefully appreciate Professor Brian M. Kurkoski for his constant advice and help in many ways. His enthusiasm, genuineness, and diligence in research have always been great encouragements to me. I learned from Brian to divide a challenging problem into several practical targets, but think thoroughly about the theoretical aspects. He is also a nice friend. Thank you.

Special thanks go to all team members in the BITS lab for their friendship, and JAIST staff for providing a comfortable research environment.

I would like to express my sincere gratitude to the committee members: Prof. Yagi, Prof. Tanaka, Prof. Fujisaki, and Prof. Schwartzman, for their careful review and critical comments. It is their valuable and insightful suggestions that enhance the quality of this dissertation.

THIS DISSERTATION IS DEDICATED TO

MY CAT DANHUANG
FOR COMING INTO MY LIFE

MOM
FOR BEING MY FIRST TEACHER

DAD
FOR GIVING ME CONFIDENCE IN STUDYING

BAIJIANG
FOR FRIENDSHIP, LOVE, AND ENCOURAGEMENT

AND THE MEMORY OF MY GRANDPA ZHIYOU
WHOSE BELIEF IN ME HAS MADE THIS JOURNEY POSSIBLE

Acronyms

ALT	approximate lower triangular
AWGN	additive white Gaussian noise
BCH	Bose–Chaudhuri–Hocquenghem
BP	belief propagation
CCL	convolutional code lattices
LDLC	low-density lattice codes
LDPC	low-density parity-check
LLR	log-likelihood ratio
MMSE	minimum mean-square error
NSM	normalized second moment
PEG	progressive edge-growth
QAM	quadrature amplitude modulation
QC-LDPC	quasi-cyclic low-density parity-check
SNR	signal-to-noise ratio
SPC	single parity-check
TBCC	tail-biting convolutional codes
VA	Viterbi algorithm
VNR	volume-to-noise ratio
WAVA	warp around Viterbi algorithm
WER	word error rate
ZTCC	zero-tailed convolutional codes

Symbols

$\mathbf{0}$	zero vector or zero matrix
$\mathbf{A}$	matrix with binary entries
a	number of levels for Construction D'
A_2	hexagonal lattice
$\mathbf{b}$	integer/information vector
BW_{16}	16-dimensional Barnes-Wall lattice
$\mathcal{C}$	linear code or binary linear code
$\mathbf{c}$	binary codeword
$\mathfrak{C}$	nested lattice code
D	delay operator
Dec	decoder
E_8	Gosset lattice
E_b	average transmitted power per bit
E_b/N_0	signal-to-noise ratio per bit
E_s	average transmitted power per symbol
$\mathbb{F}_2$	binary field
$\mathbb{F}_q$	finite field of size q
$\mathbf{G}$	generator matrix
γ_s	shaping gain
$\mathbf{G}(D)$	generator matrix in polynomial form
GF(2)	binary field
$\mathbf{G}_\mathrm{s}$	generator matrix of shaping lattice
$\mathbf{H}$	parity-check matrix or check matrix
$\mathbf{H}_\mathrm{c}$	parity-check matrix of coding lattice
$\widetilde{\mathbf{H}}$	parity-check matrix with binary entries

$\mathbf{I}$	identity matrix
k	dimension of a finite field code
K	scale factor
L	hypercube range
Λ	lattice
$\lambda(x)$	degree distribution of variable nodes
Λ_{c}	coding lattice
Λ_{24}	24-dimensional Leech lattice
Λ_{s}	shaping lattice
m	memory order
mod_2	modulo-2 operation
mod^*	"triangle-function"
n	lattice dimension or block length of finite field codes
$\mathbb{R}$	Euclidean space (real space)
r	number of parity-check basis vectors
$\mathbb{R}^n$	n-dimensional Euclidean space
R	code rate
$\rho(x)$	degree distribution of check nodes
σ^2	noise variance
$\mathbf{u}$	binary information vector
$\mathbf{w}$	noise vector
$\mathbf{x}$	lattice point
$\hat{\mathbf{x}}$	estimated lattice point
$\mathbf{y}$	received sequence or arbitrary point
$\mathbf{z}$	integer vector
$\mathbb{Z}^n$	n-dimensional integer lattice

List of Figures

List of Tables

Contents

Chapter 1

Introduction

1.1 A Brief Overview

Lattices have been studied by mathematicians for their properties such as sphere packing, covering and quantization, and serve as powerful tools with applications in information theory, communications and cryptography. The main application is to the channel coding problem, that designs signals for data transmission and storage. For source coding (analog-to-digital conversion or data compression), lattices tell us how the quantization problem is related to the shaping gain of their Voronoi regions. The lattice quantizers can also be employed in the channel coding.

In communications we are interested in how to reliably transmit information through an unreliable channel. The information could be a text message, a piece of audio or some data stored in a computer. An unreliable channel is noisy medium physically passing the information from a point to another point, or saving the information now and retrieving it later, such as Wi-Fi, an optical fiber, a magnetic disk drive, and so on. In 1948, Claude E. Shannon published a seminal paper entitled "A Mathematical Theory of Communication" [1], in which he established the fundamental theorem for point-to-point communications, and addressed that information can be efficiently and reliably transmitted by coding. Let the unit information bits per channel bit denoted by R, called the code rate. Shannon defined the maximum amount of information a channel can carry as the channel capacity C, and showed that if $R < C$ such codes exist to achieve reliable communications. Conversely, if the code rate R is greater than the channel capacity C, it is not possible to have reliable transmissions.

Error-correcting codes can provide reliable communications over unreliable channels, and are so named because they correct the errors that occur during transmission. Error-correcting linear codes are mainly defined in finite fields[1],

[1] A finite field $\mathbb{F}_q$, also known as a Galois field, is a set of integers $\{0, 1, \ldots, q-1\}$ defined

while a lattice is defined as a discrete additive subgroup of the n-dimensional Euclidean space $\mathbb{R}^n$. In channel coding, a code is mainly measured by two properties: error-correction capability and code rate. The error-correction capability provides reliable transmission, and high code rate allows larger amount of data transmission per unit. Lattice codes can provide high code rates because they are constructed by an alphabet of size larger than that of the finite field codes.

1.1.1 Modulation, Power Constraint, and Shaping

The signals containing data are sent to a channel by the transmitter. The norm of a signal is called the signal power, which determines how much power is required for transmission. In practice the transmitter never has an infinite power, and thus the average transmit power shall be constrained. The n elements of a signal lie within a sphere of radius $\sqrt{nP}$ around the origin, where P defines a power constraint. In a noisy channel, the power of the noise is determined by the variance and mean of the distribution. The most important noisy channel to consider is the additive white Gaussian noise (AWGN) channel, where the noise satisfies a Gaussian distribution. The ratio between the signal power and the noise power is called the signal-to-noise ratio (SNR). For high code rate, transmission needs high SNR.

Assume an arbitrary sequence of data. To transmit the sequence, its elements need to be converted into a signal with the form appropriate for transmission in the channel such that transmitted signals satisfy the power constraint and have zero mean. This can be performed by a modulation technique. In modern communication systems, digital modulations are used, where the constellation consists of discrete points.

One of the widely used digital modulations is the quadrature amplitude modulation (QAM), but QAM modulation scheme without probabilistic shaping [2] on signals cannot achieve the AWGN channel capacity at high SNR. This is because they do not produce Gaussian-like (or hypersphere-like) constellations, and such a constellation is essential for approaching the capacity of the channel when SNR is high. Lattices can provide better constellations than QAM by applying lattice geometric shaping. In addition, at low dimensions the QAM modulation scheme applied to finite field codes can also be regarded as a lattice constellation. The effectiveness of constellation shaping techniques is measured by the normalized second moment (equivalently, shaping gain)—it provides power reduction.

under modulo-q addition and modulo-q multiplication, where q is a prime number. The binary field $\mathbb{F}_2$ is the simplest field that is most commonly used as the alphabet of code symbols for error-correction codes.

1.1.2 Finite Field Codes

Error-correcting codes are mainly defined in finite fields, in both theory and practice. An error-correcting code defined over $\mathbb{F}_q$ maps an information sequence with k symbols to a codeword using n symbols. A codebook, the set of all codewords, is generated using an alphabet $\{0, 1, \ldots, q-1\}$. Almost all error-correcting codes used in practice are linear, and a linear code is defined as a k-dimensional subspace of a vector space $\mathbb{F}_q^n$. Then the code is said to have block length n, dimension k, and code rate $R = k/n$, and the number of codewords is $M = q^k$. Under some conditions, e.g., $R < C$, the information can be recovered even from a noisy channel.

Researchers and engineers had found many codes such as low-density parity-check (LDPC) codes, turbo codes, polar codes, and Bose–Chaudhuri–Hocquenghem (BCH) codes, that provide excellent error-correction performance.

1.1.3 A General View of Lattices and Lattice Codes

Lattices are discrete points in $\mathbb{R}^n$. The set of points in $\mathbb{R}^n$ that have the closest distance to a lattice point than to any other lattice points is called the Voronoi region, thus a lattice point is at the center of the Voronoi region. Due to the discreteness and symmetry of lattices, if a Voronoi region is shifted by every lattice point, the union of the shifted Voronoi regions cover the whole space of $\mathbb{R}^n$. More importantly, a lattice is an infinite structure. For practical use, the signal power must be constrained, thus a finite set of points of the lattice must be selected, e.g., the intersection of the lattice and some region. And this can be accomplished by lattice geometric shaping using the zero-centered Voronoi region[2] of some lattice, combining coding with modulation. The lattice performing shaping is called a shaping lattice, which needs to be a subset of the coding lattice, i.e., the lattice used for coding. The resulting intersection as a set of lattice points of the coding lattice that lie in the zero-centered Voronoi region of the shaping lattice is a nested lattice code [3], also known as a Voronoi code/constellation [4,5]. Under this coding scheme, the coding lattice corrects errors while the shaping lattice satisfies the power constraint and provides the shaping gain—the nested lattice code provides a high code rate.

The simplest lattice is the one-dimensional integer lattice consisting of every integer as a lattice point. In the literature, several low-dimensional lattices [6] are

[2]Every lattice has an all-zero lattice point which is the origin, thus its Voronoi region is called the zero-centered Voronoi region.

well-known especially for their good shaping gain of Voronoi regions, e.g., the E_8 lattice, the BW_{16} lattice and the Leech lattice, where the decoding algorithms for the BW_{16} lattice and the Leech lattice are not as efficient as decoding/quantizing the E_8 lattice. LDLC lattices [7], as an analog to LDPC codes, have good coding properties but require a high decoding complexity. Lattices can also be built from linear codes, using well-known methods such as Construction A, D, and D' [6]. Applying these methods is to lift the codebook of linear codes from finite fields to the real space $\mathbb{R}^n$, and the resulting lattices are called Construction A lattices, Construction D lattices, and Construction D' lattices, respectively. Construction D and D' are applied to a family of nested binary linear codes. Construction A is the one-level special case of Construction D suitable for an arbitrary linear code, which can produce lattices good for quantization but is generally tricky for achieving good coding properties unless applied to a nonbinary code. Construction A and D use a generator matrix while D' uses its inverse matrix called a check matrix. There are some applications such as convolutional code lattices based on convolutional codes and Construction A [8, 9], BCH code lattices based on BCH codes and Construction D [10], LDPC code lattices based on LDPC codes and Construction D' [11, 12].

A question is how to develop a pair of lattices: a coding lattice and a shaping lattice, such that the error-correction capability, a high code rate and the shaping gain are obtained using low-complexity encoding and decoding algorithms? Can the two lattices be same or different? If such lattices are found, how can they be efficiently implemented in a practical coding scheme? These questions will be addressed in the remaining part of this dissertation.

1.1.4 A Little Story

Once upon a time in information-theory wonderland, Alice had a conversation with Bob.

Alice: It is well-known that LDPC codes have been widely used in communications applications. You use them when making a phone call, viewing the world using Google Earth or saving your favorite song in solid state drives.

Bob: Then I cannot live without LDPC codes. Why are they used everywhere?

Alice: Because LDPC codes have outstanding error-correction capability and are friendly for hardware implementation. Does your smartphone support 5G? LDPC codes are used for channel coding in 5G.

Bob: Yes, I am using 5G now. It's great. A live show can be smoothly played

without lag. But I think it would be nice if the downloading speed can be improved. The faster the better. The bad thing is that the smartphone needs recharge several times a day!

Alice: Haha! Many people are addicted to internet. The problem you have depends on the devices, but they can also be improved from channel coding. You need a code with high rate. This requires high SNR. More importantly, shaping saves the energy, for both your devices and base stations.

Bob: Shaping is amazing! Can we use LDPC codes at high SNR and have shaping?

Alice: The answer is yes, in the context of lattices.

Bob: I love lattices! But codes are defined in finite fields and lattices are defined in real space. How does a lattice be developed from a code?

Alice: There do exist several methods that lift a linear code to a real space, such as Construction A and Construction D. Construction A uses a linear code. Construction D is applied to a family of nested linear codes.

Bob: ...Oh, which one do you choose for LDPC codes?

Alice: LDPC codes are described by parity-check matrices, but both of the above methods use generator matrices. Um...wait. There is one method, called Construction D' which is a friend of Construction D that can be used for LDPC codes. And you will need another lattice for shaping. Maybe a Construction A lattice is a good option.

1.2 Motivation

The capacity of the AWGN channel cannot be achieved when equiprobable QAM signal constellations are used[3] at high SNR [13], because they incur a $\pi e/6$ (1.53 dB) loss as the dimension $n \to \infty$. This loss can be overcome using spherical constellations that produce Gaussian-like distributions, but decoding an n-sphere is impractical. Constellation-shaping techniques that produce Gaussian-like distributions with reasonable complexity are desirable.

Lattices are a natural fit for wireless communications because they provide reliable transmission using real-valued algebra and higher transmit power efficiency

[3]Probabilistic QAM constellations can provide shaping gain [2], but probabilistic shaping is not directly compatible with compute-and-forward. This dissertation instead considers geometric shaping.

than the conventional QAM constellation at high SNR. They also provide high code rate and that is essential for the next-generation wireless communications. Lattices also form an important component of compute-and-forward relaying [14], which provides high throughput and high spectral efficiency. Nested lattice codes, constructed using a coding lattice Λ_c and a shaping lattice Λ_s, can be used for power-constrained communications.

Erez and Zamir [3] showed that nested lattice codes can achieve the capacity of the AWGN channel, if the coding lattice Λ_c is *channel-good* and the Voronoi region of the shaping lattice Λ_s is hyperspherical, using dithering and minimum mean-square error (MMSE) scaling techniques. In such a coding scheme, a coding lattice Λ_c does the work for coding, that is, error-correction. Thus, a high coding gain is appealing since it measures the error-correction capability. Also, a low-complexity decoding algorithm is desirable for coding. There are some candidates suitable as coding lattices such as the BCH code lattices, LDLC lattices, and LDPC code lattices. The shaping lattice Λ_s contributes the shaping gain, which tells how good the "shape" its Voronoi region is. The higher the shaping gain is, the more power reduction it provides. The theoretical limit of the shaping gain is 1.53 dB when the "shape" is a hypersphere and the lattice dimension approaches infinity. As a lattice quantizer, the shaping lattice Λ_s requires a quantization algorithm costing low complexity. If such an algorithm exists, the shaping gain is said to be efficiently achievable.

In the context of lattice shaping (or quantization), the input source of a lattice quantizer is a lattice point of the coding lattice Λ_c. The quantization error is dependent of the input source distribution, especially at low SNR. This can be improved by adding a sequence uniformly distributed in the Voronoi of Λ_s to the input source, called a dither, such that the quantization error can be seen independent of the source. And the technique is called dithering. A linear scalar estimator with respect to MMSE needs to be included for the coding scheme applied in the power-constrained AWGN channel. For high code rates, dithering is not required [15] and the role of MMSE scaling becomes negligible [3].

Two lattices Λ_c and Λ_s are called self-similar if Λ_s is an integer-scaled version of Λ_c. The design of Λ_c and Λ_s has competing requirements, as Λ_c demands good coding properties and an efficient decoding algorithm while Λ_s needs good shaping gains and low-complexity quantization. The design of Λ_c and Λ_s can be separated, under the principle of the separation of coding gain and shaping gain [13]. *Rectangular encoding* and *indexing* for non-self-similar nested lattice codes were proposed in [16], and conditions on lattice constructions were given.

LDPC codes have been implemented in a wide variety of communications appli-

cations because of their capacity-achievability, efficient encoding, low-complexity decoding, and suitability for hardware implementation. For these reasons, LDPC codes are also suitable for constructing lattices. Lattices based on binary LDPC codes using Construction D' were first introduced in [11]. Recently Branco da Silva and Silva [12] proposed efficient encoding and decoding for Construction D' lattices, particularly for LDPC codes. A codeword and cosets of component linear codes are used to form systematic codewords for Construction D' lattices. This encoding method naturally produces lattice points in a hypercube. However, a hypercube does not provide shaping gain. A shaping lattice Λ_s is needed to do so.

Erez, Litsyn and Zamir showed that there exist Construction A lattices that are asymptotically good regarding both coding and shaping [17]. However, under practical considerations of finite length and computationally feasible quantization, the lattices best for coding may not be the best for shaping. Convolutional code lattices that are built from convolutional codes using Construction A are attractive for shaping, because of their good shaping gain, flexibility for dimension, and efficient quantization using the Viterbi algorithm [18].

The shaping gain of convolutional code lattices can be increased by increasing the memory order of the convolutional code. This comes at the expense of computational complexity. While it is known that there are lattices which are simultaneously good when used for coding and shaping [17], within the family of convolutional codes, and with finite length, the lattices best for coding may not be the best for shaping. Past work also ignored the role of complexity in performing the shaping operation, beyond the simple observation that complexity increases with the memory order. Also, past work [17] considered BPSK and 4-PAM constellations, which does not reveal the precise shaping gain. Thus, in the search for convolutional code lattices with good shaping gain, it is reasonable that a wider variety of convolutional codes should be considered, including the zero-tailed[4] convolutional codes, tail-biting convolutional codes and truncated convolutional codes.

1.3 Related Work

The two principal classes of codes at high SNR are trellis codes and lattices. Forney proposed generalized cross constellations [19] and trellis shaping [20] that can easily achieve a shaping gain of 1 dB, and claimed an asymptotic shaping gain of 1.36 dB for trellis shaping, citing [21], using trellis coded modulation [22].

[4]This is the conventional termination.

Lattices from linear codes have potentials since the decoder for linear codes can be employed to find the nearest lattice point given a point. Well-known methods to build lattices from linear codes are Construction A and D/D' [6, Ch. 5, 8]. Construction D/D' generate lattices from multi-level nested binary linear codes. Binary Construction A lattices are the special case of one-level Construction D lattices. Unlike Construction A and D using generator matrices, Construction D' describes lattices by check matrices and thus is suitable for LDPC codes.

Erez and ten Brink employed trellis shaping, constructing lattices based on zero-tailed convolutional codes and Construction A that were used for vector quantization in a dirty paper coding scheme [8]; four rate 1/2 convolutional codes and their shaping gains were given. Kudryashov and Yurkov found generator polynomials of rate 1/2 convolutional codes that provide the best asymptotic shaping gain with respect to zero-tailed termination, and near-optimum shaping gain at low dimensions with respect to tail-biting termination in [23] and [24], respectively. This dissertation's results extend their work addressing the optimality of shaping gain to a wider range of dimensions and code rates, and additionally consider the shaping gain-complexity tradeoff. Convolutional codes terminated by direct truncation are also included.

Construction D produces lattices with good coding properties and low-complexity decoding by employing the decoder for underlying linear codes. Construction D was used to build turbo code lattices [25], polar code lattices [26] and BCH code lattices [10]. Polar code lattices can achieve the capacity of the AWGN channel using lattice Gaussian shaping [27].

Conway and Sloane proposed shaping on Voronoi constellations [4]. The normalized second moment of a variety of low-dimensional ($n \leqslant 24$) well-known lattices were listed in [28, Table V]. See also [5] [6, p. 61]. These lattices provide excellent shaping gain for their dimension.

Using self-similar nested lattice codes, a shaping gain of 0.4 dB was shown for LDLC[5] lattices [30], and a shaping gain of 0.776 dB was claimed at $n = 60$ for Construction A lattices based on QC-LDPC codes [31]. A shaping gain of 0.65 dB and 0.86 dB was observed using the E_8 lattice and the BW_{16} lattice for shaping LDLC lattices, respectively [32]. Leech lattice has 1.03 dB shaping gain, and was used to shape LDA lattices [33]. Convolutional code lattices to shape low-density lattice codes (LDLC) lattices [9] a shaping gain of 0.87 dB was preserved at $n = 36$. A shaping gain of 0.63 dB was found using the E_8 lattice for shaping BCH-code based Construction D lattices [34].

[5]The work in [29] also constructed nested lattice codes using self-similar LDLC lattices.

1.4 Contributions

This dissertation addresses the following problems:

1. Develop encoding and decoding algorithms for Construction D' such that Construction D' lattices can be used as coding lattices in power-constrained channels.
2. Design QC-LDPC codes for Construction D' such that the resulting QC-LDPC Construction D' lattices are suitable for coding in a practical coding scheme.
3. Search for good lattices that can help Construction D' lattices satisfy the power constraint, and provide additional shaping gain. Convolutional code lattices formed by Construction A and binary convolutional codes are used, with aspects of design, performance and complexity considered.

The outcome of this work is a comprehensive solution of practically implementable lattice coding applications in communications, under the framework of nested lattice codes. Two distinct constructions of lattices lifted from binary linear codes are studied and designed. QC-LDPC Construction D' lattices inherit good coding properties from the widely practically used QC-LDPC codes, while convolutional code lattices provide the prospect of satisfying power constraint for QC-LDPC Construction D' lattices. This dissertation provides a fundamental strategy for Construction D' lattices to be used under power constraint. Moreover, the resulting nested lattice code constructions can provide a good coding gain, a high shaping gain, efficient encoding and low-complexity decoding, which also make them of interest for compute-and-forward.

The main contributions of this dissertation are classified into several categories:

1. Construction D' encoding and decoding algorithms that can be applied to a variety of nested linear codes for power-constrained channels under the lattice framework.
2. Construction D' lattice constructions and the design of the underlying nested QC-LDPC codes, whose structure provides efficiency for lattice encoding and indexing.
3. Extensive Construction A lattice constructions and evaluations with respect to the *best* tradeoff between the shaping gain and the quantization complexity provided by various binary convolutional codes with three trellis termination methods, including zero-tailed termination, tail-biting termination, and

direct truncation.

4. Modification of nested lattice code indexing method to overcome the integer overflow problem in high dimensions when implemented, and the resulting method is applicable to all nested lattice codes if their generator matrix or check matrix can be easily triangularized.

5. Nested lattice code constructions when distinct lattices are considered for shaping Construction D' lattices, which are the E_8 lattice, the BW_{16} lattice, the Leech lattice, and the *best-found* convolutional code lattice.

1.4.1 Construction D' Encoding/Decoding

This dissertation tackles the encoding and decoding problem of Construction D' lattices to be used in power-constrained communications. Accordingly, two encoding methods: *Encoding method A* and *Encoding method B*, and a decoding algorithm of Construction D' lattices suitable for satisfying target channels are proposed. Encoding method A encodes integers with an approximate lower triangular (ALT) check matrix. Encoding method B shows how binary information bits are mapped to a lattice point using the check matrix of the underlying nested linear codes for a Construction D' lattice. Multistage successive cancellation decoding algorithm employing binary decoders is used; the decoder uses re-encoding based on encoding method B; this method is distinct from [12].

A definition of Construction D' using check-matrix perspective is also given, which is equivalent to the conventional congruences definition [6].

1.4.2 QC-LDPC Code Design for Construction D'

This dissertation constructs QC-LDPC codes to form Construction D' lattices (termed QC-LDPC Construction D' lattices), because QC-LDPC codes are widely used in recent wireless communication standards. In this dissertation, parity-check matrices for nested QC-LDPC codes are designed such that they can be easily triangularized and thus efficient encoding and indexing are allowed. An existing Construction D' lattice based on a QC-LDPC code and a single parity-check (SPC) product-like code [35] is not suitable for indexing/shaping because the SPC product-like code parity-check matrix cannot be efficiently triangularized.

Let $\mathbf{H}_0$ and $\mathbf{H}_1$ be the parity-check matrix of binary code $\mathcal{C}_0$ and $\mathcal{C}_1$ respectively. A *subcode condition* $\mathcal{C}_0 \subset \mathcal{C}_1$ must be satisfied to form a 2-level Construction D' lattice, and this is not straightforward. In [12], LDPC code parity-check matrix $\mathbf{H}_0$

was obtained from $\mathbf{H}_1$ by performing check splitting or progressive edge-growth (PEG)-based check splitting. This dissertation presents a QC-LDPC design for code $\mathcal{C}_0$, and the position of non-zero circulants (or blocks) in $\mathbf{H}_0$ is found by binary linear programming. In contrast to [12], $\mathbf{H}_1$ will be constructed from $\mathbf{H}_0$.

This work also shows how to triangularize the parity-check matrices for QC-LDPC codes with designed structure, and how to build a check matrix for a Construction D' lattice.

1.4.3 Extensive Search for Convolutional Code Lattices

A method to obtain triangular generator matrices for Construction A lattices is given, as a modification of the classical method [6, p. 183] (also [36, pp. 32–33]). The method in this dissertation allows forming the generator matrix without swapping code bit positions for convolutional code lattices with underlying rate $1/2, 1/3, \ldots$ binary convolutional codes. This is also valid for construction of convolutional code lattices from both zero-tailed convolutional codes and tail-biting convolutional codes with any code rate. The best-found shaping gain of convolutional code lattices at various dimensions is obtained by an exhaustive search over zero-tailed convolutional codes, tail-biting convolutional codes, and truncated convolutional codes of rate 1/2 and 1/3 with a variety of numbers of states. The asymptotic shaping gains and the tradeoff between the shaping gain and the complexity of quantization are also studied.

1.4.4 Modified Lattice Indexing Method

Encoding and indexing, and construction of nested lattice codes are reviewed. When applied to high-dimensional nested lattice codes, the conventional indexing algorithm [16, Sec. IV-B] may encounter large-valued integers, which causes an integer overflow problem when implemented—it may fail to recover information even in the absence of noise. To solve this problem, this dissertation provides a modified algorithm to bound the values of integers that are used internally, without changing the solution.

1.4.5 Nested Lattice Code Constructions

Several nested lattice code constructions are given. The coding lattices are the designed QC-LDPC Construction D' lattices, while the shaping lattices are distinct,

including the E_8 lattice, the BW_{16} lattice, the Leech lattice, and the convlutional code lattices. These lattices as lattice quantizer can provide high shaping gains, and the full shaping gains are 0.65 dB, 0.86 dB, 1.03 dB, and possibly greater than 1.25 dB respectively.

When a convolutional code lattice performs lattice quantization, the shaping gain as high as 1.25 dB is preserved in a practical nested lattice coding scheme. This is the best-found numerical result in the literature of lattice shaping.

Hypercube shaping is performed for comparison with a nested lattice code using shaping lattices, with respect to the shaping gain, because a hypercubical constellation provides no shaping gain. When generating a hypercubical constellation for Construction D' lattices, a method simpler than the conventional method is described.

1.5 Dissertation Scope and Notation

The rest of this dissertation is organized as follows:

1. Chapter 2 introduces definitions and properties of lattices, as well as several important transformations of lattices which are used in this work. Lattice construction methods from linear codes are given, including Construction A and D'. The well-known lattices in short dimensions are listed, including the E_8, BW_{16} and Leech lattices that will be employed for shaping.
2. Chapter 3 describes the encoding and indexing of nested lattice codes, and addresses a modified indexing method for high dimensions. The nested lattice coding scheme additionally including the indexing is introduced. Hypercube shaping is also given.
3. Chapter 4 proposes encoding and decoding methods for Construction D', as a foundation for Construction D' lattices to be used in power-constrained channels.
4. Chapter 5 addresses QC-LDPC code design for Construction D', and shows how to generate a triangular check matrix for a QC-LDPC Construction D' lattice.
5. Chapter 6 presents extensive Construction A lattice constructions and evaluations using zero-tailed convolutional codes, tail-biting convolutional codes, and truncated convolutional codes. The tradeoff between the lattice shaping gain and quantization complexity is studied.
6. Chapter 7 shows numerical results for QC-LDPC Construction D' lattices (to be designed in Chapter 5) in a power-constrained channel under the framework that will be described in Chapter 3, where the shaping lattices include: the E_8 lattice, the BW_{16} lattice, and the Leech lattice (that will be introduced in Chapter 2), and the best-found convolutional code lattices (as will be provided in Chapter 6), using the decoding algorithm to be proposed in Chapter 4.
7. Chapter 8 addresses conclusions of this work, and discusses about related future topics.

Notation

Generator vectors are written in columns, which is convenient when considering the lattice check matrix that has basis (check) vectors in rows. This is distinct from the convention used for finite field codes. A lattice point $\mathbf{x}$ is given as a column vector, while sometimes written in the form $(x_1, x_2, \ldots, x_n)$ with respect to its elements.

A tilde indicates a vector or matrix which has only 0s and 1s — $\widetilde{\mathbf{x}}$ and $\widetilde{\mathbf{H}}$ are binary while $\mathbf{x}$ and $\mathbf{H}$ are not necessarily so. Operations over the real numbers $\mathbb{R}$ are denoted $+, \cdot$ (the operator $\cdot$ is sometimes omitted) while operations over the binary field $\mathbb{F}_2$ are denoted $\oplus, \odot$. The binary field $\mathbb{F}_2$ is also written as GF(2). The direct sum operation of two lattices is also denoted $\oplus$. The matrix transpose is denoted $(\cdot)^{\mathrm{t}}$. Element-wise rounding to the nearest integer is denoted $\lfloor \cdot \rceil$.

Chapter 2

Lattices and Lattices From Codes

This chapter addresses how a lattice is generated from a basis, described using a generator matrix or a check matrix and decomposed using a sublattice and cosets of the sublattice. This chapter also introduces how to evaluate a lattice with respect to its coding and shaping properties, and how this is connected to its Voronoi region. A Voronoi region of a lattice plays a key role in both lattice decoding and lattice quantization. If two lattices satisfy a sublattice condition, then the intersection of the "super" lattice and the Voronoi region of the sublattice defines a nested lattice code (also known as a Voronoi code). The super lattice is for coding and the sublattice is for shaping.

Several important lattice transformations used in this dissertation are introduced: identical lattices, lattice scaling and the direct sum of lattices. Two lattices are identical if one basis can be obtained by multiplying another basis by a unimodular matrix. Thus the resulting two distinct generator matrix can be used interchangeably to describe the same lattice. Sometimes a triangular generator matrix is desirable since it is convenient for lattice encoding and indexing, as will be seen in Chapter 3. An equivalent lattice is produced if it is scaled by a real nonzero number. The direct sum of two or more identical lattices can provide a high-dimensional lattice without changing the shaping property. This construction is employed when low-dimensional lattices are to be used in higher dimensions.

Two lattice constructions that build lattices from linear codes, Construction A and Construction D', are briefly described, which are used in lattice constructions in this dissertation. Construction A is applied to generate convolutional lattice codes as shaping lattices and the method will be given in Chapter 6. The encoding/decoding of Construction D' will be addressed in Chapter 4.

Several well-known low-dimensional lattices including the E_8 lattice, the BW_{16} lattice and the Leech lattice that will be employed in the shaping lattice constructions, are then introduced. By showing the definitions and several examples it will be seen how these lattices are related to linear codes, as well as the methods for construction.

2.1 Preliminaries

2.1.1 Definition of Lattices

Definition 2.1 A lattice Λ is a discrete additive subgroup of the n-dimensional Euclidean space $\mathbb{R}^n$.

A lattice is a vector subspace, because $\mathbb{R}^n$ is a vector space and Λ is a subgroup—Λ is a set of points (called lattice points) closed under real addition. A lattice possesses a linearity property: if lattice points $\mathbf{x}_1, \mathbf{x}_2 \in \Lambda$, then the sum of two lattice points $\mathbf{x}_1 + \mathbf{x}_2 \in \Lambda$. Let $\mathbf{x} \in \Lambda$, then there is $\mathbf{x} + \mathbf{x} + \cdots + \mathbf{x} \in \Lambda$, thus Λ has infinite size. Assume an integer α, then $\alpha \cdot \mathbf{x} \in \Lambda$. Observe $\mathbf{x} - \mathbf{x} \in \Lambda$, thus the origin (or the zero point) is always a lattice point.

Example 2.1 The set of all integers $\mathbb{Z}$ is a one-dimensional lattice. Consider two lattice points $3 \in \mathbb{Z}$ and $1 \in \mathbb{Z}$, then the sum $3+1 = 4$, $3+3 = 6$ are lattice points. Since the reflected $-3 \in \mathbb{Z}$, the origin 0 is a lattice point. Scaling the lattice point 3 by an integer -5 gives $-15 \in \mathbb{Z}$.

2.1.2 Lattice Basis, Generator Matrix and Check Matrix

If Λ is an n-dimensional lattice, this n-dimensional space can be spanned by a basis which is a set of n linearly independent basis vectors (also called generator vectors) $\mathbf{g}_1, \mathbf{g}_2, \ldots, \mathbf{g}_n$. A column vector

$$\mathbf{g}_i = \begin{bmatrix} g_1 \\ g_2 \\ \vdots \\ g_n \end{bmatrix} \tag{2.1}$$

represents a point in $\mathbb{R}^n$ for $i \in \{1, 2, \ldots, n\}$.

Let integers $b_j \in \mathbb{Z}$ be elements of a column vector $\mathbf{b}$:

$$\mathbf{b} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{bmatrix}, \tag{2.2}$$

for $j \in \{1, 2, \ldots, n\}$. Then the linear combination of the bases defines a lattice

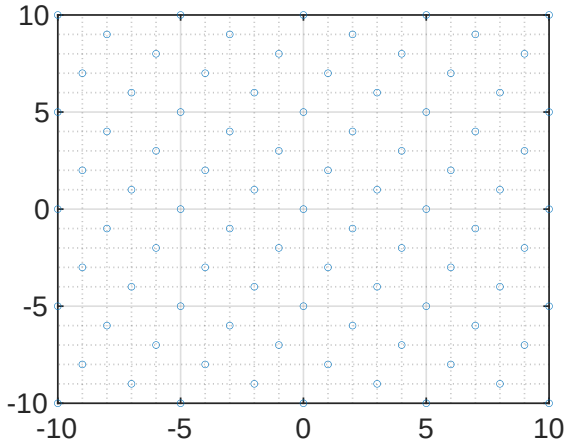


Figure 2.1: Example of a two-dimensional lattice Λ^2.

point $\mathbf{x}$ as

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

which is a vector of n real numbers:

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}. \tag{2.4}$$

Recognize that the points $\mathbf{g}_1, \ldots, \mathbf{g}_n$ are also lattice points.

Example 2.2 Let two linearly independent vectors $\mathbf{g}_1 = [1, 2]^{\mathrm{t}}, \mathbf{g}_2 = [2, -1]^{\mathrm{t}}$ form a basis. Then a two-dimensional lattice Λ^2 is obtained and shown in Figure 2.1.

The basis for a lattice is not unique. The same lattice Λ^2 in Example 2.2 can be generated using bases such as

$$\left\{[-2, 1]^{\mathrm{t}}, [3, 1]^{\mathrm{t}}\right\},\ \left\{[-1, -2]^{\mathrm{t}}, [2, -1]^{\mathrm{t}}\right\},\ \left\{[1, 2]^{\mathrm{t}}, [0, 5]^{\mathrm{t}}\right\}, \ldots \tag{2.5}$$

Can any 2 linearly independent vectors in Λ^2 form a basis for Λ^2? To answer this, let us try the two vectors $[3, 1]^{\mathrm{t}}, [0, 5]^{\mathrm{t}} \in \Lambda^2$. Draw points that are integral linear combinations of these two vectors in Figure 2.2. Observe that $[5, 0]^{\mathrm{t}}$ cannot be produced by any integral linear combinations of $[3, 1]^{\mathrm{t}} \cdot \alpha_1 + [0, 5]^{\mathrm{t}} \cdot \alpha_2$ for $\alpha_1, \alpha_2 \in \mathbb{Z}$, but $[5, 0]^{\mathrm{t}}$ can be expressed as $[1, 2]^{\mathrm{t}} \cdot 1 + [2, -1]^{\mathrm{t}} \cdot 2$ using the basis given in Example 2.2 and thus is a lattice point of Λ^2. And of course there are other lattice points of Λ^2 that cannot be generated by $[3, 1]^{\mathrm{t}}, [0, 5]^{\mathrm{t}}$. Thus the answer to the question is no. How to select a basis is omitted.

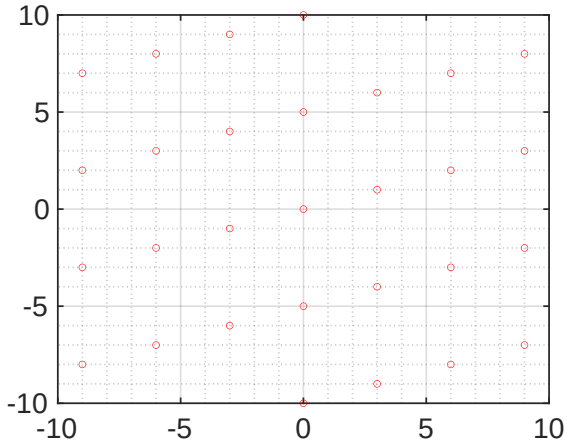


Figure 2.2: Bad basis example that cannot generate Λ^2.

Integer bases span lattices in $\mathbb{Z}^n$, which is straightforward. But a basis need not have all integer values in order to generate a lattice, e.g., a basis with vectors $\{[-1, 0]^{\mathrm{t}}, [-1/2, -\sqrt{3}/2]^{\mathrm{t}}\}$ spans a lattice in Figure 2.8.

Remark 2.1 For a valid basis consisting of n linearly independent basis vectors $\mathbf{g}_1, \mathbf{g}_2, \ldots, \mathbf{g}_n$ in $\mathbb{R}^n$, the subgroup of all linear combinations with integral coefficients of the basis vectors forms a lattice.

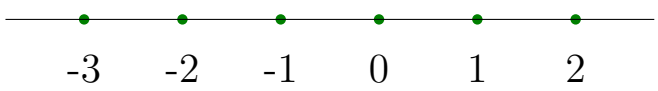
Write n generator vectors in a square matrix of order n, called a *generator matrix*:

$$\mathbf{G} = \begin{bmatrix} | & | & & | \\ \mathbf{g}_1 & \mathbf{g}_2 & \cdots & \mathbf{g}_n \\ | & | & & | \end{bmatrix}. \tag{2.6}$$

Thus (2.3) can also be written by the matrix form as:

$$\mathbf{x} = \mathbf{G} \cdot \mathbf{b}. \tag{2.7}$$

Example 2.3 The integer lattice $\mathbb{Z}^n$ is the simplest lattice. The one-dimensional integer lattice is the set of all integers:



The two-dimensional integer lattice can be drawn and is given in Figure 2.3. A

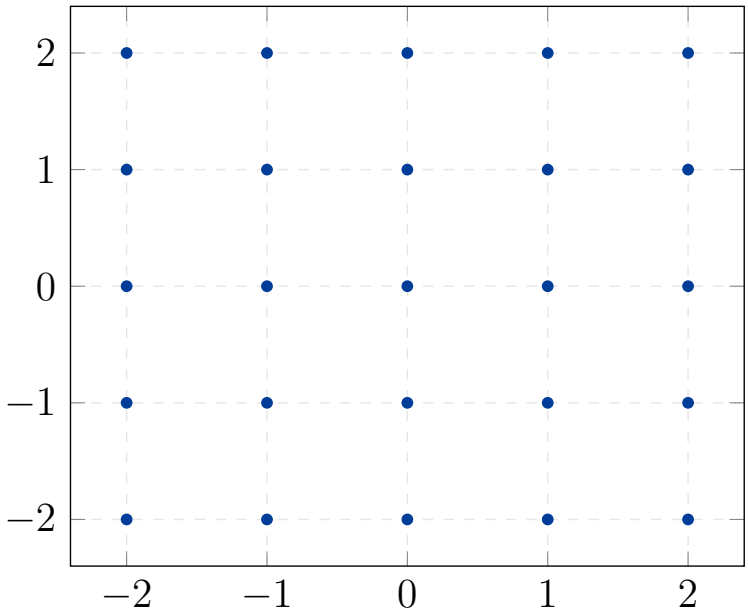


Figure 2.3: Two-dimensional integer lattice $\mathbb{Z}^2$.

natural generator matrix of $\mathbb{Z}^n$ is the identity matrix $\mathbf{I}_n$ for dimension n:

$$\mathbf{I}_n = \begin{bmatrix} 1 & 0 & \cdots & 0 \\ 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \ddots & 0 \\ 0 & 0 & \cdots & 1 \end{bmatrix}. \tag{2.8}$$

A lattice Λ can also be described using the *check matrix* $\mathbf{H} = \mathbf{G}^{-1}$ which is expressed by

$$\mathbf{H} = \begin{bmatrix} — & \mathbf{h}_1 & — \\ — & \mathbf{h}_2 & — \\ & \vdots & \\ — & \mathbf{h}_n & — \end{bmatrix}, \tag{2.9}$$

where each check basis vector $\mathbf{h}_i$ is a row vector of n real-valued elements, for $i = 1, 2, \ldots, n$. Then computing (2.7) is equivalent to solve

$$\mathbf{H} \cdot \mathbf{x} = \mathbf{b}. \tag{2.10}$$

For a lattice with check matrix $\mathbf{H}$, $\mathbf{x}$ is a lattice point if and only if $\mathbf{H} \cdot \mathbf{x}$ is a vector of integers.

2.1.3 Lattice Cosets

A lattice coset is a discrete set of points which is a shift of a lattice by some vector.

Definition 2.2 A lattice coset of Λ with respect to a vector $\mathbf{s}$ is the set:

$$\mathbf{s} + \Lambda = \{\mathbf{s} + \mathbf{x} \mid \mathbf{x} \in \Lambda\}. \tag{2.11}$$

A coset of a lattice is of infinite size, since a lattice is a set of infinite size. The difference vector between every pair of two points of a lattice coset $\mathbf{s} + \Lambda$ is in Λ. In general, a lattice coset is not a lattice as it does not contain the origin, unless $\mathbf{s} \in \Lambda$.

If a lattice Λ' contains all lattice points of another lattice Λ, written as $\Lambda \subset \Lambda'$, then the Λ is a sublattice of Λ'. Since Λ is a subgroup of the group Λ', the two lattices form a quotient group denoted Λ'/Λ. A vector $\mathbf{s} \in \Lambda'$ can be used to determine a coset of Λ relative to Λ' where the relative coset is: $\mathbf{s} + \Lambda$. Since $\Lambda \subset \Lambda'$, each relative coset belongs to Λ'. The set of relative cosets is Λ'/Λ:

$$\{\mathbf{s} + \Lambda \mid \mathbf{s} \in \Lambda'\}. \tag{2.12}$$

The union of Λ'/Λ covers Λ' and then the size of Λ'/Λ is finite. Each coset can be represented by an element, called a coset leader. The coset leaders can be selected such that they form a codebook for a code—nested lattice code. More description will be given in the next chapter. The following example shows another choice of coset leaders.

Example 2.4 In Figure 2.4, a lattice $5\mathbb{Z}^2$ is scaled by 5 from the integer lattice $\mathbb{Z}^2$ given in Figure 2.3, thus its lattice points are also integers. A coset of $5\mathbb{Z}^2$ takes a shift of all points in $5\mathbb{Z}^2$ by a vector, e.g., $(2,2) + 5\mathbb{Z}^2$, shown as red diamonds in Figure 2.5. Since $(2,2) + (2,2) = (4,4)$ is not an element of $(2,2) + 5\mathbb{Z}^2$, $(2,2) + 5\mathbb{Z}^2$ is not a lattice. Consider the lattice Λ^2 given in Example 2.2. Λ^2 can be constructed using the union of the lattice $5\mathbb{Z}^2$ and its cosets with respect to vectors $\mathbf{s}_i \in \{(0,0),(1,2),(2,4),(3,1),(4,3)\}$, described as:

$$\Lambda = \bigcup_{i=0}^{4} (\mathbf{s}_i + 5\mathbb{Z}^2). \tag{2.13}$$

Each coset is represented by a distinct marker in Figure 2.6. Observe that the lattice $5\mathbb{Z}^2$ is a sublattice of Λ^2: $5\mathbb{Z}^2 \subset \Lambda^2$. See Figure 2.7.

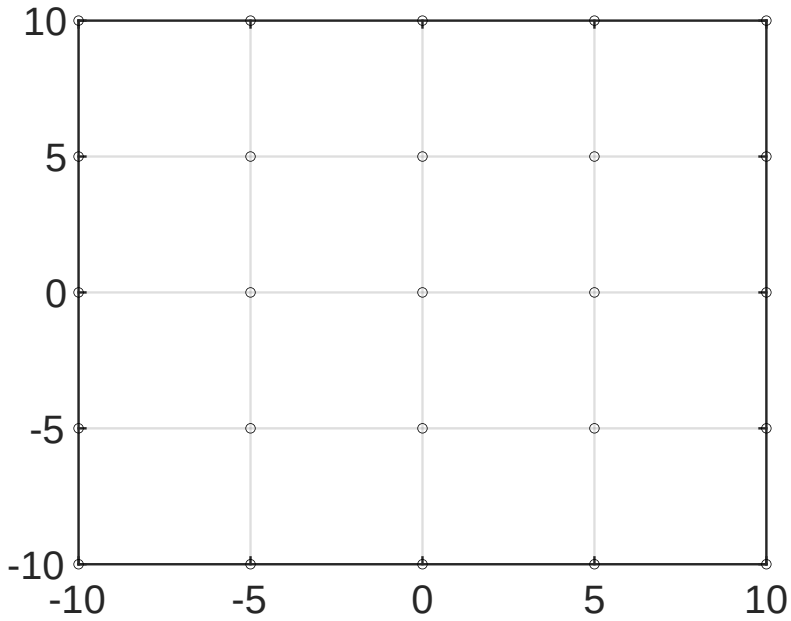


Figure 2.4: Scaled integer lattice $5\mathbb{Z}^2$.

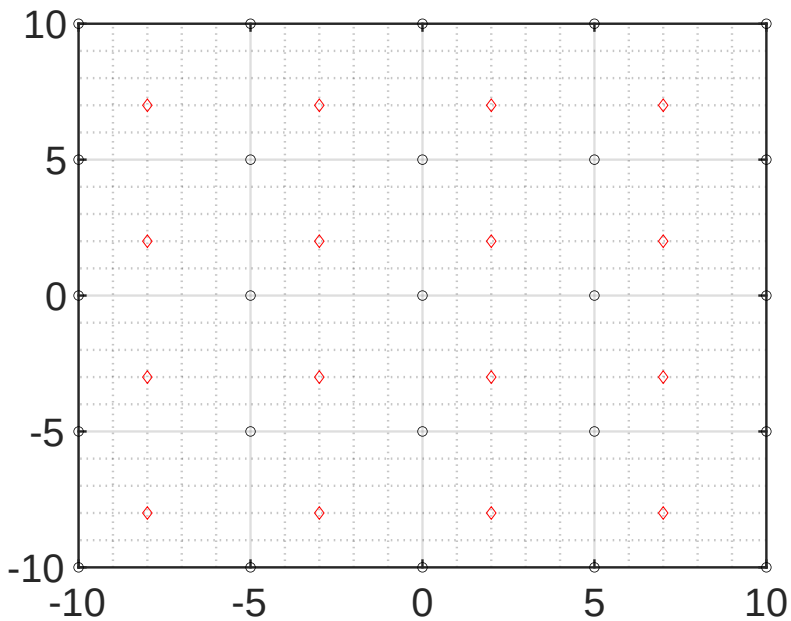


Figure 2.5: Coset of lattice $5\mathbb{Z}^2$ with respect to a vector $(2, 2)$ where $5\mathbb{Z}^2$ is marked as black circles and its coset is marked as red diamonds.

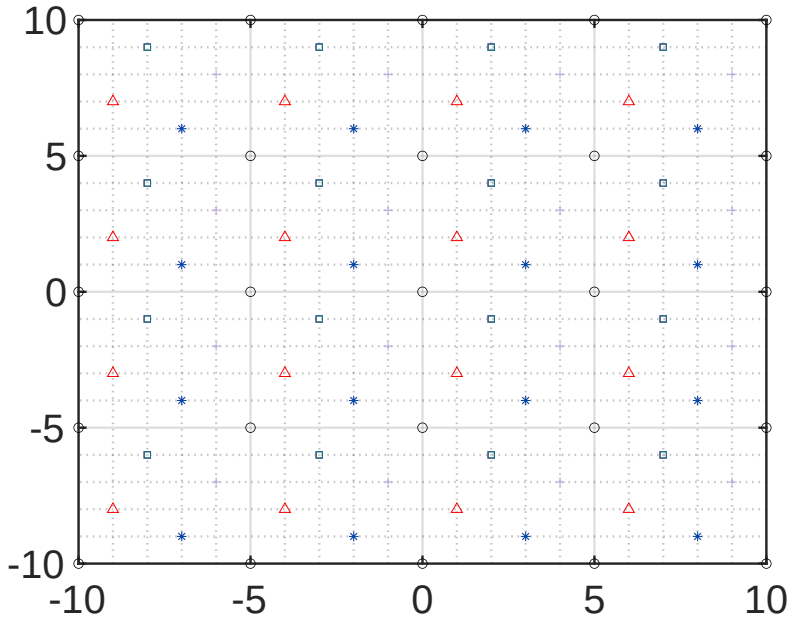


Figure 2.6: Lattice decomposed by $5\mathbb{Z}^2$ (black circles) and 4 cosets of $5\mathbb{Z}^2$ (red triangles, green squares, blue asterisks and purple pluses).

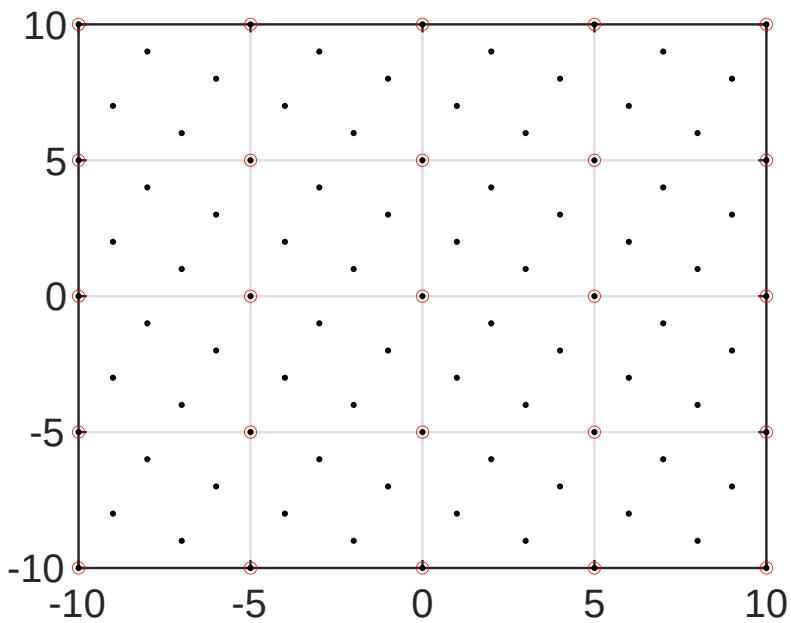


Figure 2.7: Nested lattices example where a lattice is marked as black solid circles and its sublattice is marked as red circles.

2.1.4 Lattice Quantization and Modulo

For a lattice Λ, the nearest-neighbor quantizer (or decoder) finds the closest lattice point $\mathbf{x} = (x_1, x_2, \ldots, x_n) \in \Lambda$ given an arbitrary point $\mathbf{y} = (y_1, y_2, \ldots, y_n) \in \mathbb{R}^n$ which is expressed as:

$$Q(\mathbf{y}) = \arg\min_{\lambda \in \Lambda} \|\mathbf{y} - \lambda\|^2, \tag{2.14}$$

where λ is some lattice point in Λ and $\|\cdot\|^2$ denote the squared Euclidean distance (or norm) that computes

$$\|\mathbf{y} - \lambda\|^2 = \sum_{i=1}^{n} (y_i - \lambda_i)^2. \tag{2.15}$$

Lattice quantization (2.14) is also written by

$$\mathbf{x} = Q_\Lambda(\mathbf{y}), \tag{2.16}$$

that is read as the quantization of $\mathbf{y}$ to find its nearest $\mathbf{x} \in \Lambda$.

The quantization error vector

$$\mathbf{t} = \mathbf{y} - Q_\Lambda(\mathbf{y}) \tag{2.17}$$

is usually written as:

$$\mathbf{t} = \mathbf{y} \bmod \Lambda, \tag{2.18}$$

and the operation is referred to as the lattice modulo operation. It will be shown that $\mathbf{t}$ and $\mathbf{y}$ belong to the same coset of Λ where $\mathbf{t}$ is constrained in a shape called the zero-centered Voronoi region, when a Voronoi codebook is applied. This is applied when encoding nested lattice codes that will be introduced in Chapter 3. The quantization effectiveness is measured by the shaping gain, which will be described in Subsection 2.1.7.

Lattice Coset Decoding

A lattice Λ' may be decoded using lattice coset decoding if Λ' can be represented by a union of cosets. Let $\Lambda \subset \Lambda'$ and decompose Λ' by its sublattice Λ and cosets of Λ as: $\Lambda' = \bigcup_{i=0}^{l-1} (\mathbf{s}_i + \Lambda)$ where $l - 1$ denotes the number of cosets for which $\mathbf{s}_i$ is a coset leader and $\mathbf{s}_0 = (0, \ldots, 0)$. Assume that how to find a nearest lattice point for Λ is known. Given an arbitrary point $\mathbf{y}$, then Λ' can be decoded in the following way. For $i = 0, 1, \ldots, l-1$, $\mathbf{s}_i$ is subtracted from $\mathbf{y}$ as $\mathbf{y} - \mathbf{s}_i$, then decode $\widehat{\mathbf{y}}'_i = \mathbf{s}_i + Q_\Lambda(\mathbf{y} - \mathbf{s}_i)$. Let d_i denote the squared Euclidean distance between $\widehat{\mathbf{y}}'_i$ and $\mathbf{y}$. The closest lattice point in Λ' is $\widehat{\mathbf{y}} = \widehat{\mathbf{y}}'_j$ corresponding to the smallest value d_j for some $j \in \{0, 1, \ldots, l-1\}$. This includes the quantization of a lattice coset, and more on quantization will be introduced in the nest subsection.

2.1.5 Voronoi Region and Its Volume

Definition 2.3 For a lattice Λ, a fundamental region (or fundamental cell) denoted $\mathcal{F} \subset \mathbb{R}^n$ is a shape that, if shifted by each lattice point in Λ, will exactly cover the whole Euclidean space $\mathbb{R}^n$.

The Euclidean space $\mathbb{R}^n$ can be divided into disjoint congruent partitions where each partition is a shift of the fundamental region, by a lattice point in Λ. A fundamental region is a maximal set satisfying that the difference vector between any pair of two points in $\mathcal{F}$ is not a lattice point. Any point in $\mathbb{R}^n$ is in exactly one fundamental region. A coset $\mathbf{s} + \Lambda$ can be represented by the unique intersection point (called a coset representative) of the coset and $\mathcal{F}$: $(\mathbf{s} + \Lambda) \cap \mathcal{F}$. Thus $\mathcal{F}$ is a complete set of coset representatives, and the set of all cosets is called the quotient group $\mathbb{R}^n/\Lambda = \{\mathbf{s} + \Lambda \mid \mathbf{s} \in \mathcal{F}\}$.

The division is not unique thus there are different fundamental regions for a lattice, and the most important one is the Voronoi region.

Definition 2.4 A Voronoi region $\mathcal{V}(\mathbf{x})$ is defined as a set of all points nearest to $\mathbf{x} \in \Lambda$ than to any other lattice point.

Using the nearest-neighbor quantizer (2.16), a Voronoi region $\mathcal{V}(\mathbf{x})$ can be described as:

$$\mathcal{V}(\mathbf{x}) = \{\mathbf{y}' \in \mathbb{R}^n \mid Q(\mathbf{y}') = \mathbf{x}\}, \tag{2.19}$$

which is a convex polytope since Euclidean distance is used. A line that connects $\mathbf{x}$ to one of its neighbors[1] is crossing orthogonally a hyperplane, which determines[2] a face of $\mathcal{V}(\mathbf{x})$.

The Voronoi region does not depend on the generator matrix but relies on lattice points. The zero-centered Voronoi region $\mathcal{V}(\mathbf{0})$ corresponding to the origin $\mathbf{0}$ is simply denoted $\mathcal{V}$. All shifts of $\mathcal{V}$ by lattice points $\mathbf{x} \in \Lambda$ cover the entire space $\mathbb{R}^n$, thus any point in $\mathbb{R}^n$ can be represented as the sum of a lattice point and a point in $\mathcal{V}$ where this point in $\mathcal{V}$ is the shortest vector in its coset, regarded as a coset leader. The set of all coset leaders in $\mathcal{V}$ can be used to define a nested lattice code if each is a lattice point of the "superlattice", as will be introduced in Chapter 3. Note that the lattice modulo operation (2.18) is to find the coset leader for an arbitrary point in $\mathbb{R}^n$.

[1]The neighbors of a lattice point $\mathbf{x}$ are defined as a set of all lattice points in Λ that have the shortest distance to $\mathbf{x}$ than the distance between any other lattice point with $\mathbf{x}$.

[2]There exist some points in $\mathbb{R}^n$ that have the same distance to two or more lattice points. In order to keep the fundamental regions congruent, each point can belong to only one fundamental region. This can be solved by a systematic tie-breaking rule for the quantizer $Q(\mathbf{y})$.

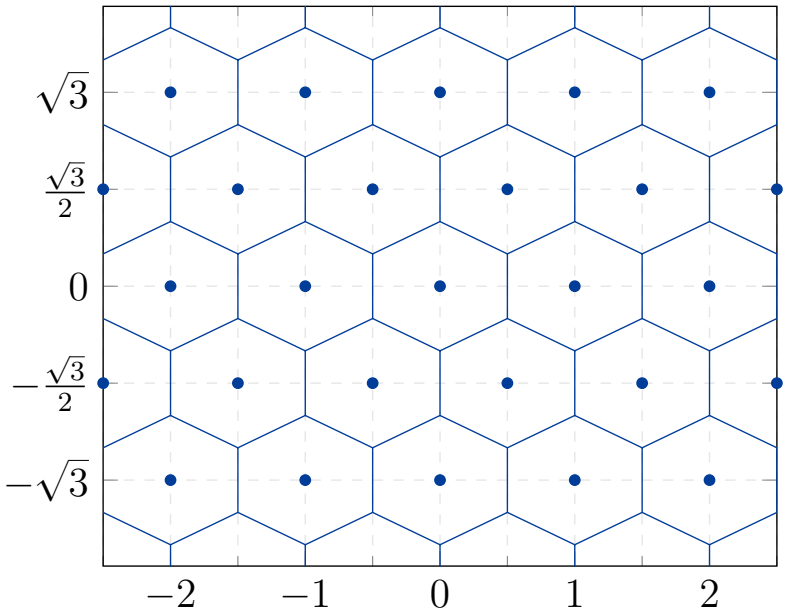


Figure 2.8: Two-dimensional hexagonal lattice A_2 with lattice points • and Voronoi regions ⬡.

Another commonly used fundamental region is the parallelotope region that consists of all points which are linear combinations of the basis vectors $\mathbf{g}_1, \ldots, \mathbf{g}_n$ with coefficients between zero and one. It is convenient to use the parallelotope region to enumerate the Voronoi codebook for a nested lattice code.

For a lattice Λ described by a generator matrix $\mathbf{G}$, its various fundamental regions have the same volume $V(\Lambda)$, given by:

$$V(\Lambda) = \det(\Lambda) = |\det(\mathbf{G})|, \tag{2.20}$$

where $\det(\cdot)$ computes the determinant.

Example 2.5 The 2-dimensional hexagonal lattice can be described by a generator matrix:

$$\begin{bmatrix} \frac{1}{2} & 1 \\ \frac{\sqrt{3}}{2} & 0 \end{bmatrix}, \tag{2.21}$$

which has volume $\frac{\sqrt{3}}{2}$. The lattice points and the Voronoi regions are shown in Figure 2.8. The shape of the Voronoi region is a hexagon, thus the lattice is called the hexagonal lattice, and it is equivalent to A_2 lattice, as will be introduced in Subsection 2.3.1.

2.1.6 Identical Lattices, Lattice Scaling and Direct Sum

2.1.6.1 Identical Lattices

Two lattices Λ_1 and Λ_2 are identical if all the lattice points are the same, called identical lattices. A basis of Λ_1 can be transformed into another basis representing an identical lattice Λ_2. That is to say, the generator matrix of a lattice is not unique, and can be transformed to a distinct generator matrix which describes the same lattice. This is performed by using a *unimodular*[3] *matrix*.

Proposition 2.1. Let an n-by-n matrix $\mathbf{G}$ be a generator matrix for a lattice Λ. If $\mathbf{W}$ is a unimodular matrix then $\mathbf{G} \cdot \mathbf{W}$ is also a generator matrix for Λ. Similarly, let $\mathbf{H} = \mathbf{G}^{-1}$, then $\mathbf{W}' \cdot \mathbf{H}$ is also a check matrix for Λ if $\mathbf{W}'$ is unimodular.

2.1.6.2 Lattice Scaling

An n-dimensional lattice Λ is equivalent to a scaled version of Λ, and the coding gain (2.34) does not change. Let K be a real nonzero number, then $\Lambda' = K\Lambda$ is a lattice scaled from Λ. For an arbitrary lattice point $\mathbf{x} \in \Lambda$, there is $K\mathbf{x} \in K\Lambda$. Let $\mathbf{G}$ be a generator matrix of Λ, then the lattice $K\Lambda$ has a generator matrix $K\mathbf{G}$. The volume of $K\Lambda$ is

$$V(K\Lambda) = \det(K\Lambda) = |\det(K\mathbf{G})| = |K^n \det(\mathbf{G})|. \tag{2.22}$$

Given an arbitrary point $\mathbf{y} \in \mathbb{R}^n$, the quantization using $\Lambda' = K\Lambda$ can be performed by

$$\mathbf{x}' = Q_{\Lambda'}(\mathbf{y}) = Q_{K\Lambda}(\mathbf{y}) = K \cdot Q_{\Lambda}(\mathbf{y}/K), \tag{2.23}$$

where a quantizer (2.16) for Λ is used and thus is straightforward.

Example 2.6 Let a lattice be scaled by $\sqrt{2}$ from the integer lattice $\mathbb{Z}^3$. Then $\sqrt{2}\mathbb{Z}^3$ has a generator matrix

$$\begin{bmatrix} \sqrt{2} & 0 & 0 \\ 0 & \sqrt{2} & 0 \\ 0 & 0 & \sqrt{2} \end{bmatrix}, \tag{2.24}$$

and volume $2\sqrt{2}$.

[3] A unimodular matrix is an n-by-n matrix with integer entries and determinant ± 1. The inverse of a unimodular matrix is also unimodular.

2.1.6.3 Direct Sum

A simple method to build a high-dimensional lattice from low-dimensional lattices is using the direct sum[4].

Let Λ_1 and Λ_2 be two lattices with dimensions n_1 and n_2, generator matrices $\mathbf{G}_1$ and $\mathbf{G}_2$, respectively. Let the lattice points be denoted $\mathbf{x} \in \Lambda_1$ and $\mathbf{y} \in \Lambda_2$. An n-dimensional lattice Λ can be produced using direct sum of Λ_1 and Λ_2, represented by:

$$\Lambda = \Lambda_1 \oplus \Lambda_2 = \left\{ \left(\mathbf{x}, \mathbf{y}\right) \;\middle|\; \mathbf{x} \in \Lambda_1, \mathbf{y} \in \Lambda_2 \right\}, \tag{2.25}$$

where $n = n_1 + n_2$. The generator matrix $\mathbf{G}$ of Λ is a block-diagonal matrix given by

$$\mathbf{G} = \begin{bmatrix} \mathbf{G}_1 & \mathbf{0} \\ \mathbf{0} & \mathbf{G}_2 \end{bmatrix}. \tag{2.26}$$

The volume of $\Lambda = \Lambda_1 \oplus \Lambda_2$ has:

$$\begin{aligned} V(\Lambda) &= \det(\Lambda) && (2.27)\\ &= \det(\Lambda_1 \oplus \Lambda_2) && (2.28)\\ &= |\det(\mathbf{G})| && (2.29)\\ &= |\det(\mathbf{G}_1)| \cdot |\det(\mathbf{G}_2)| && (2.30)\\ &= \det(\Lambda_1) \cdot \det(\Lambda_2). && (2.31) \end{aligned}$$

Example 2.7 Consider two lattices $3\mathbb{Z}^3$ and $2A_2$, where $\mathbb{Z}^3$ has a generator matrix (2.8) and A_2 has a generator matrix (2.21). Then a generator matrix of $3\mathbb{Z}^3 \oplus 2A_2$ is obtained:

$$\begin{bmatrix} 3 & 0 & 0 & 0 & 0 \\ 0 & 3 & 0 & 0 & 0 \\ 0 & 0 & 3 & 0 & 0 \\ 0 & 0 & 0 & 1 & 2 \\ 0 & 0 & 0 & \sqrt{3} & 0 \end{bmatrix}, \tag{2.32}$$

whose volume is $(3^3 \cdot 1) \cdot (2^2 \cdot \frac{\sqrt{3}}{2}) = 54\sqrt{3}$.

[4]The direct sum of two lattices $\Lambda_1 \oplus \Lambda_2$ is the same as their Cartesian product $\Lambda_1 \times \Lambda_2$. Here the notation $\oplus$ is the same as that of the addition operation used in binary field GF(2), but there should be no ambiguity between the two distinct operations.

It is convenient to build a high-dimensional lattice Λ by the direct sum of a sequence of identical low-dimensional lattices Λ' as $\Lambda = \Lambda' \oplus \Lambda' \oplus \cdots \oplus \Lambda'$. The lattice Λ has the same shaping gain (2.42) as the component lattice Λ'. This is often applied to well-known lattices such as the 8-dimensional E_8 lattice, the 16-dimensional BW_{16} lattice and the 24-dimensional Leech lattice, which will be introduced in Section 2.3 and will be employed when building high-dimensional lattices for shaping in Chapter 7. Let an n-dimensional lattice Λ be the direct sum of n/n' copies[5] of an n'-dimensional lattice Λ', then the quantization of Λ can be performed by quantizing each component lattice Λ' for its corresponding elements, that is, an estimated lattice point for Λ is produced by employing the quantizer of Λ' for n/n' times.

2.1.7 Properties for Coding and Shaping

2.1.7.1 Coding Gain and Volume-to-Noise Ratio

Coding Gain

The minimum distance $d_{\min}$ indicates the error-correction capability of a code $\mathcal{C}$. If $\mathcal{C}$ is a block code defined in finite fields, $d_{\min}$ is defined as the minimum of the Hamming distances between all distinct pairs of codewords $\mathbf{x} \in \mathcal{C}$ and $\mathbf{y} \in \mathcal{C}$. The Euclidean distance is used when considering Euclidean-space codes. It is convenient to use the squared minimum distance $d^2_{\min}$.

Due to the linearity of a lattice Λ, the squared minimum distance is given by:

$$d^2_{\min} = \min_{\mathbf{x} \in \Lambda \setminus \mathbf{0}} \|\mathbf{x}\|^2. \tag{2.33}$$

Scaling Λ by K produces a lattice with squared minimum distance of $K^2 d^2_{\min}$, but the average transmitted power is also increased. Thus the coding gain γ_{c} is defined as the normalized squared minimum distance which is independent of lattice scaling, and is expressed as:

$$\gamma_{\mathrm{c}} = \frac{d^2_{\min}}{V(\Lambda)^{2/n}}. \tag{2.34}$$

Volume-to-Noise Ratio

The common notion signal-to-noise ratio (SNR) used for codes is not meaningful for lattices because they are a set of infinite size. The transmitted sequences

[5]Here n/n' must be a positive integer.

thus are included in an infinite constellation. This was modeled by Poltyrev [37], such that a lattice-goodness can be evaluated in a power-unconstrained AWGN channel, where the capacity is called Poltyrev limit (or Poltyrev capacity).

The volume-to-noise ratio (VNR) of a lattice Λ is defined as the ratio between the normalized volume of Λ and the normalized volume of the noise sphere, given by:

$$\text{VNR} = \frac{V(\Lambda)^{2/n}}{2\pi e \sigma^2}, \tag{2.35}$$

which is the distance to the Poltyrev limit. Let the probability of error for decoding an n-dimensional lattice be $P_{\text{e}}(\sigma^2)$ using a minimum-distance decoder. Unless $V(\Lambda)^{2/n} > 2\pi e \sigma^2$, $P_{\text{e}}(\sigma^2)$ cannot be small. Moreover, if $V(\Lambda)^{2/n} \approx 2\pi e \sigma^2$ then $P_{\text{e}}(\sigma^2)$ cannot be small unless n is large [38]. Thus the VNR is useful when measuring error-correction performance of lattices, and is commonly given in decibels as $10 \log_{10} \text{VNR}$. When $\text{VNR} = 1$ ($\text{VNR} = 0$ dB) the capacity is achieved given a noise variance $\sigma^2 = \frac{V(\Lambda)^{2/n}}{2\pi e}$.

2.1.7.2 Normalized Second Moment and Shaping Gain

Consider the lattice quantization (2.16) for a lattice Λ. The quantization error vector (2.18) can be measured by the normalized second moment (NSM).

The NSM of a region $\mathcal{R} \subset \mathbb{R}^n$ is defined:

$$G_n(\mathcal{R}) = \frac{1}{nV(\mathcal{R})^{1+2/n}} \int_{\mathcal{R}} \|\mathbf{x}\|^2 d\mathbf{x}, \tag{2.36}$$

which depends only on the "shape" of the region and does not change if is scaled. The following three kinds of regions are of interest: hypercubes, n-spheres and Voronoi regions.

NSM of Hypercubes

The NSM of an n-dimensional hypercube (also called an n-cube) is the same as the NSM of the integer lattice $\mathbb{Z}^n$, given by

$$G_n(\mathbb{Z}^n) = \frac{1}{12}. \tag{2.37}$$

It is clear that the NSM of an n-hypercube does not change when n is changed, and thus is useful as a baseline when compared to a general lattice quantizer. This will be addressed when introducing the shaping gain in the remaining part of this subsection, and in Section 3.5.

NSM of n-Spheres

The NSM of an n-sphere S_n (or an n-ball[6]) is important because it has the lowest possible value among all regions, and is found:

$$G_n(S_n) = \frac{\Gamma(\frac{n}{2}+1)^{2/n}}{\pi(n+2)}, \tag{2.38}$$

where $\Gamma(\cdot)$ is the Gamma function[7].

At all dimensions, the NSM of all regions cannot be lower than that of the sphere and thus is constrained by the sphere bound (2.38). The asymptotic value can be obtained using the Stirling approximation:

$$\Gamma(\frac{n}{2}+1) = (\frac{n}{2})! \approx (\frac{n}{2e})^{n/2}, \tag{2.39}$$

and is given by:

$$\lim_{n\to\infty} G_n(S_n) = \frac{1}{2\pi e}. \tag{2.40}$$

NSM of Voronoi Regions for Lattices

Monte Carlo integration [39, 40] can be applied to practically find the NSM of a lattice Λ with unknown Voronoi region, and was used by Conway and Sloane to estimate lattice quantizers [6]. Assume that N uniformly distributed samples $\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_N$ are generated in $\mathbb{R}^n$. Then the quantization error vectors $\mathbf{t}_1, \mathbf{t}_2, \ldots, \mathbf{t}_N$ given by (2.18) are uniformly distributed in the zero-centered Voronoi region $\mathcal{V}$ for Λ. The estimated NSM is given by:

$$\widehat{G_n}(\Lambda) = \frac{1}{nNV(\Lambda)^{2/n}} \sum_{i=1}^{N} \|\mathbf{t}_i\|^2. \tag{2.41}$$

The precision of this value increases as N increases.

Shaping Gain

The shaping gain of a region $\mathcal{R}$ measures the improvement in NSM relative to the hypercubes (2.37). It is defined (in decibels) as:

$$\gamma_{\mathrm{s}}(\mathcal{R}) = 10\log_{10}\frac{1/12}{G_n(\mathcal{R})} \text{ dB}. \tag{2.42}$$

[6]In lattice literature, the two terms: sphere and ball, are often used synonymously. The volume of an n-ball concentrates on the surface which is an n-sphere as $n \to \infty$, thus in this dissertation the term n-sphere (or hypersphere) is used for simplicity.

[7]The Gamma function extends the factorial function to non-integer arguments, e.g., $\Gamma(\frac{1}{2}) = \sqrt{\pi}$. If a is a positive integer, then $\Gamma(a+1) = a\Gamma(a) = a!$ and $\Gamma(a+\frac{1}{2}) = (a-\frac{1}{2})\cdot(a-\frac{3}{2})\cdot\cdots\cdot\frac{1}{2}\cdot\sqrt{\pi}$.

The hypercube has 0 dB of shaping gain.

As $n \to \infty$, an n-sphere yields the theoretical limit of the shaping gain, given by:

$$10 \log_{10} \frac{\pi e}{6} = 1.53 \text{ dB}, \tag{2.43}$$

known as the asymptotic shaping gain.

The shaping gain depends on the "shape" of a region $\mathcal{R}$. If $\mathcal{R}$ is scaled, the shaping gain does not change. It measures the reduction in average transmitted power with respect to a hypercubical constellation.

2.2 Lattices From Linear Codes

There exist several methods to build lattices from linear codes, including Construction A, B, C, D/D' and E, that lift code or codes to the Euclidean space $\mathbb{R}^n$. The simplest method is Construction A, which can use a binary code or a non-binary code. The following methods: B, C, D/D', E, are related to Construction A. Construction B requires a binary code and is a special case of Construction D. While Construction A and B can be applied to lattice constructions from a linear code, Construction C, generalized from Construction A and B, can be applied to a nested family of linear codes $\mathcal{C}_0 \subseteq \mathcal{C}_1 \subseteq \cdots \subseteq \mathcal{C}_a$. As Construction C can also be applied to codes that are not nested and to nonlinear codes, it may not always produce lattice packings, but its variant Construction D does. Construction D' converts a set of parity-checks which define nested binary linear codes into congruences for a lattice in the same way that Construction D converts a set of generator vectors for nested binary linear codes into a basis for a lattice. Construction E is a powerful generalization of most of the previous constructions that can be applied recursively.

Among these constructions, Construction A and D' are of greatest interest in this dissertation, as it will be shown that Construction A provides low-complexity quantization as well as a high shaping gain when applied to convolutional codes, and Construction D' can be applied to lattice constructions from a nested family of powerful low-density parity-check (LDPC) codes, providing good coding properties.

2.2.1 Construction A

A lattice can be built from a q-ary[8] linear code with Construction A, called a Construction A lattice, or a modulo-q lattice.

Definition 2.5 Let $\mathcal{C}$ be a q-ary linear code of dimension k and block length n. An n-dimensional Construction A lattice Λ_{A} is generated by:

$$\Lambda_{\mathrm{A}} = \mathcal{C} + q\mathbb{Z}^n. \tag{2.44}$$

Λ_{A} is the set of all integer vectors whose modulo-q reduction are codewords of $\mathcal{C}$, where each is a lattice point $\mathbf{x}$. Then (2.44) can also be expressed as:

$$\Lambda_{\mathrm{A}} = \left\{\mathbf{x} \in \mathbb{Z}^n \;\middle|\; \mathbf{x} \bmod q \in \mathcal{C}\right\}. \tag{2.45}$$

Properties

A Construction A lattice Λ_{A} provides the following properties:

1. Λ_{A} contains and is contained in integer lattices: $q\mathbb{Z}^n \subset \Lambda_{\mathrm{A}} \subset \mathbb{Z}^n$, as will be explained in Example 2.8.
2. The volume of Λ_{A} is

   $$V(\Lambda_{\mathrm{A}}) = q^{n-k}. \tag{2.46}$$

   This is because a quotient group $\Lambda_{\mathrm{A}}/q\mathbb{Z}^n$ is formed, which has size

   $$|\Lambda_{\mathrm{A}}/q\mathbb{Z}^n| = \frac{V(q\mathbb{Z}^n)}{V(\Lambda_{\mathrm{A}})} = \frac{q^n}{V(\Lambda_{\mathrm{A}})} = |\mathcal{C}|, \tag{2.47}$$

   where $|\mathcal{C}| = q^k$ is the number[9] of codewords of $\mathcal{C}$ if generated by a full-rank n-by-k matrix $\mathbf{G}_{\mathcal{C}}$.
3. Λ_{A} is spanned by a basis of $\mathcal{C}$ and a basis of $q\mathbb{Z}^n$, thus the generator matrix is of size n-by-$(n+k)$, written as:

   $$\mathbf{G}'_{\Lambda_{\mathrm{A}}} = \left[\mathbf{G}_{\mathcal{C}} \mid q\mathbf{I}_n,\right] \tag{2.48}$$

   where $\mathbf{G}_{\mathcal{C}}$ consists of k linearly independent basis vectors in column for $\mathcal{C}$, and $q\mathbf{I}_n$ is a possible generator matrix for $q\mathbb{Z}^n$ naturally scaled by q from (2.8).

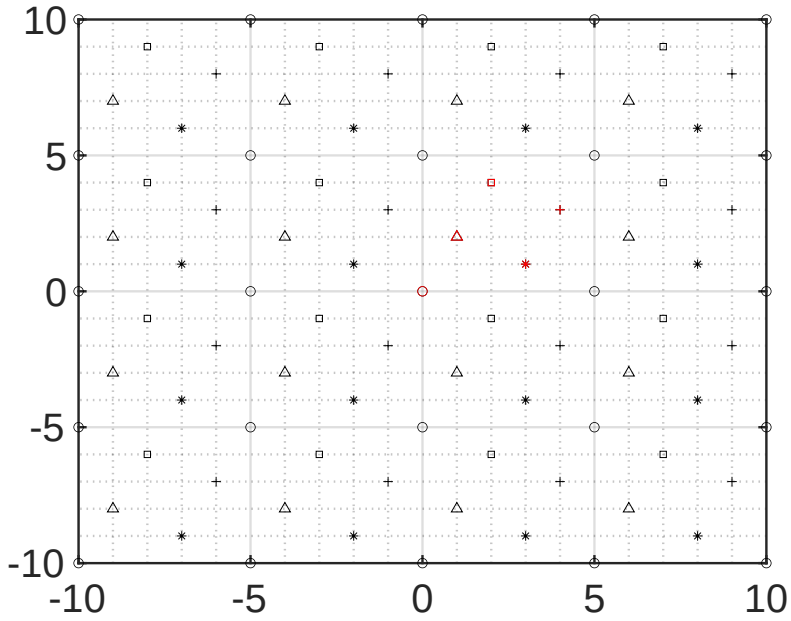


Figure 2.9: Two-dimensional Construction A lattice example.

Example 2.8 The $n = 2$-dimensional lattice given in Example 2.2 is a Construction A lattice Λ_{A}^2, whose basis can also be $\mathbf{g}_1 = [1, 2]^{\mathrm{t}}, \mathbf{g}_2 = [0, 5]^{\mathrm{t}}$. To see this, let $\mathbf{g}_1 = [1, 2]^{\mathrm{t}}$ be a basis for a $q = 5$ quinary linear code $\mathcal{C}$ over GF(5), whose dimension (i.e., information vector length) is $k = 1$ and block length is $n = 2$. The information can be a digit in $\{0, 1, \ldots, 4\}$, thus all $q^k = 5$ codewords[10] of $\mathcal{C}$ are $(0, 0), (1, 2), (2, 4), (3, 1), (4, 3)$, which can be represented by the red markers in Figure 2.9.

Applying Construction A to $\mathcal{C}$ lifts all codewords in $\mathcal{C}$ to $\mathbb{R}^n$ as $\Lambda_{\mathrm{A}}^2 = \mathcal{C} + 5\mathbb{Z}^2$. The resulting lattice Λ_{A}^2 has dimension $n = 2$. Since $(0, 0) \in \mathcal{C}$, $5\mathbb{Z}^2$ is also included in Λ_{A}^2, shown as circles. A shift of $5\mathbb{Z}^2$ by $(1, 2)$ is a coset of $5\mathbb{Z}^2$, and is drawn as triangles. The resulting cosets of $5\mathbb{Z}^2$ with respect to $(2, 4), (3, 1), (4, 3)$ are shown as squares, asterisks and pluses, respectively. The union of these cosets and $5\mathbb{Z}^2$ itself is the Construction A lattice Λ_{A}^2. The codewords of $\mathcal{C}$ can be regarded as coset leaders. See also Example 2.4. Note that the codewords of $\mathcal{C}$ are integer vectors and $5\mathbb{Z}^2$ is a scaled integer lattice; their sum are integer vectors in $\mathbb{Z}^2$. Thus $5\mathbb{Z}^2 \subset \Lambda_{\mathrm{A}}^2 \subset \mathbb{Z}^2$.

[8]Here q is a prime number.

[9]The codebook size of $\mathcal{C}$ is at most q^k, with equality for information vector of length $k > 0$ in $\{0, 1, \ldots, q-1\}^k$ and prime q.

[10]For example, information represented by 4 produces a codeword $\begin{bmatrix} 1 \\ 2 \end{bmatrix} \cdot 4 \bmod 5 = \begin{bmatrix} 4 \\ 3 \end{bmatrix}$.

Triangular Generator Matrix

It is clear that the n-by-$(n+k)$ matrix $\mathbf{G}'_{\Lambda_{\mathrm{A}}}$ (2.48) is not full rank, which can be reduced to a square matrix $\mathbf{G}_{\Lambda_{\mathrm{A}}}$ of order n by eliminating k linearly dependent columns, and $\mathbf{G}_{\Lambda_{\mathrm{A}}}$ is not unique.

Let $\mathbf{G}'_{\mathcal{C}} = [\mathbf{I}_k; |; \mathbf{P}]^{\mathrm{t}}$ be a systematic form of $\mathbf{G}_{\mathcal{C}}$ in (2.48) which always exists as long as $\mathbf{G}_{\mathcal{C}}$ is full rank. Zamir [36, pp. 32-33] gave a systematic approach for finding the generator matrix of a Construction A lattice Λ_{A}, that is an n-by-n lower-triangular matrix expressed as:

$$\mathbf{G}_{\Lambda_{\mathrm{A}}} = \left[\begin{array}{c|c} \mathbf{G}'_{\mathcal{C}} & \begin{matrix} \mathbf{0} \\ q\mathbf{I}_{n-k} \end{matrix} \end{array}\right] = \left[\begin{array}{c|c} \mathbf{I}_k & \mathbf{0} \\ \mathbf{P} & q\mathbf{I}_{n-k} \end{array}\right]. \tag{2.49}$$

For $q = 2$, the generator matrix (2.49) is a transpose of [6, p. 183, eq. (5)] given by Conway and Sloane, whose basis vectors are written in row, provided that $\sqrt{2}$ was omitted as mentioned in the remark.

A systematic matrix $\mathbf{G}'_{\mathcal{C}}$ is not necessary. Another work [41, pp. 42-44] also addressed a method to obtain a triangular generator matrix for Construction A lattices, using the Hermite normal form[11]. The Hermite normal form in column satisfies two conditions: (1) lower-triangular form and (2) the diagonal entry has the maximum value for each row. The Hermite normal form can be computed using the algorithm in [42, pp. 67-68].

Assume two positive integers $n \leqslant n'$. Any n-by-n' integer matrix $\mathbf{A}$ can be reduced to an n-by-n matrix $\mathbf{B}$ in column Hermite normal form, expressed as:

$$\mathbf{A} = \big[\mathbf{B} \mid \mathbf{0}\big] \cdot \mathbf{U}, \tag{2.50}$$

where $\mathbf{U}$ is an n'-by-n' unimodular matrix and the size of the zero matrix $\mathbf{0}$ is n-by-$(n'-n)$. If $\mathbf{A}$ has full row rank n, then $\mathbf{B}$ is also full rank. When a systematic form $\mathbf{G}'_{\mathcal{C}}$ can be found, the connection from Hermite normal form to Zamir's work was shown in [41]. Even when some cases the resulting $\mathbf{B}$ is not in the Hermite normal form, a triangular matrix can be found for Λ_{A}.

This dissertation derives a systematic method to generate a lower-triangular matrix for a Construction A lattice that is based on a binary code as will be discussed in Chapter 6, which is straightforward to be applied to convolutional codes. The resulting lower-triangular matrix also satisfies the column Hermite normal form. The simplest Construction A lattice is using a $q = 2$ binary linear

[11] The Hermite normal form is an analog to reduced echelon form in $\mathbb{Z}^n$.

code. In this work, Construction A is applied to binary convolutional codes to build lattices suitable for quantization.

Quantization

Assume an n-dimensional Construction A lattice Λ_{A} is obtained from a binary linear code $\mathcal{C}$. The quantization of Λ_{A} is straightforward due to the fact that any lattice point $\mathbf{x} \in \Lambda_{\mathrm{A}}$ is congruent modulo 2 to a codeword in $\mathcal{C}$ (2.45). Thus given an arbitrary point $\mathbf{y} \in \mathbb{R}^n$, a lattice point $\widehat{\mathbf{y}} \in \Lambda_{\mathrm{A}}$ which is closest to $\mathbf{y}$ than any other lattice point can be found by employing a decoder for $\mathcal{C}$. This is valid based on the following lemma.

Lemma 2.1 Suppose $\mathbf{y}$ lies in the hypercube $0 \leqslant y_i \leqslant 1$ for $i = 1, 2, \ldots, n$. Then no points of Λ_{A} is closer to $\mathbf{y}$ than the closest codeword $\widehat{\mathbf{y}}$ of $\mathcal{C}$.

Proof Suppose the contrary, and let $\widehat{\mathbf{y}}$ be a closest lattice point to $\mathbf{y}$. By hypothesis some y_i are neither 0 nor 1. By subtracting a suitable vector $2\mathbf{z}$, these coordinates can be changed to 0 or 1, depending on their parity, to produce a point of Λ_{A} that is in $\mathcal{C}$, and is at least as close to $\mathbf{y}$ as $\widehat{\mathbf{y}}$ is, a contradiction. □

The steps of quantizating or decoding Construction A were given in [6, p. 450], and are written in $0, 1$ notation as:

1. Given an input $\mathbf{y}$, reduce all y_i to the range $0 \leqslant y_i < 2$ by subtracting a vector $2\mathbf{z}$: $\mathbf{y}' = \mathbf{y} - 2\mathbf{z}$, for $\mathbf{z} \in \mathbb{Z}^n$.
2. Let S denote the set of i for which $1 \leqslant y'_i < 2$. For $i \in S$, replace y'_i by $2 - y'_i$ resulting in a vector $\mathbf{y}''$.
3. Since $\mathbf{y}''$ is now in the cube $0 \leqslant y''_i \leqslant 1$, then apply the binary decoder Dec to $\mathbf{y}''$ (Lemma 2.1), obtaining an output $\mathbf{c} = \mathrm{Dec}(\mathbf{y}'')$.
4. For $i \in S$, change c_i to $2 - c_i$: $\mathbf{c}'(S) = 2 - \mathbf{c}(S)$. Then $\widehat{\mathbf{y}} = \mathbf{c}' + 2\mathbf{z}$ is the closest point of Λ_{A} to $\mathbf{y}$. Thus $\widehat{\mathbf{y}}$ is the output.

The above procedure[12] is described in Algorithm 2.1. A binary decoder can be employed when quantizing a Construction A lattice due to Lemma 2.1. The quantization method will be described in a simpler way in Algorithm 6.1 in Section 6.3. In this dissertation, binary convolutional codes are considered when building Construction A lattices, where convolutional codes can be decoded using the well-known Viterbi algorithm proposed by Andrew Viterbi in 1967 [18]—it provides optimality, feasibility and efficiency. More discussions will be given

[12]Here $\mathbf{c}'$ need not be binary.

in Chapter 6.

Algorithm 2.1 Quantization of Construction A Lattices [6, p. 450]

Input: real-valued input $\mathbf{y}$

Output: estimated lattice point $\widehat{\mathbf{y}}$

Subtract a vector $2\mathbf{z}$: $\mathbf{y}' = \mathbf{y} \bmod 2$, where $\mathbf{z} \leftarrow (\mathbf{y} - \mathbf{y}')/2$

$S \leftarrow \text{find}(\mathbf{y}' > 1)$

$\mathbf{y}'' = \mathbf{y}'$

for $i \in S$ **do**

  $y''_i = 2 - y'_i$

end for

Call the binary decoder: $\mathbf{c} = \text{Dec}(\mathbf{y}'')$

$\mathbf{c}' = \mathbf{c}$

for $i \in S$ **do**

  $c'_i = 2 - c_i$

end for

$\widehat{\mathbf{y}} = \mathbf{c}' + 2\mathbf{z}$

return the estimated lattice point $\widehat{\mathbf{y}}$

2.2.2 Construction D/D'

Construction D generates a lattice from nested binary linear codes $\mathcal{C}_0 \subseteq \mathcal{C}_1 \subseteq \cdots \subseteq \mathcal{C}_a = \mathbb{F}_2^n$ where integer $a \geqslant 1$. When $a = 1$, Construction D is reduced to Construction A for a binary linear code.

Construction D describes a lattice using a generator matrix, while Construction D' describes a lattice using a check matrix. It is convenient and natural to build lattices using Construction D' from LDPC codes, because they are described by parity-check matrices. The main focus of this dissertation is Construction D', as will be addressed in Chapters 4, 5, and 7.

Table 2.1: Normalized second moment (NSM) and corresponding shaping gain of low-dimensional well-known lattices.

Dimension	Lattice	Name	NSM	Shaping Gain (dB)	Sphere Bound (dB)	Related Code
1	$\mathbb{Z}_1$	integer	0.0833	0	0	
2	A_2	hexagonal	0.0802	0.17	0.20	
3	D_3	checkerboard	0.0787	0.25	0.34	single parity-check code
4	D_4	checkerboard	0.0766	0.37	0.45	single parity-check code
5	D_5		0.0758	0.41	0.54	
6	E_6		0.0743	0.50	0.61	
7	E_7		0.0732	0.57	0.67	$(7, 4)$ Hamming code
8	E_8	Gosset	0.0717	0.65	0.72	$(8, 4)$ extended Hamming code
16	BW_{16}	Barnes-Wall	0.0683	0.86	0.97	Reed-Muller code
24	Λ_{24}	Leech	0.0658	1.03	1.10	extended binary Golay code
∞	-	-	0.0585	1.53	1.53	-

2.3 Well-Known Low-Dimensional Lattices

This section introduces the lattices with low dimensions $1 \leqslant n \leqslant 24$ where each at its dimension either is or nearly is the best appeared in the literature. The integer lattice $\mathbb{Z}^n$, the D_n lattice and the A_n lattice are introduced in Subsection 2.3.1. Then the E_8 lattice, the BW_{16} lattice and the Leech lattice that will be employed to build shaping lattices in this dissertation, are given in Subsection 2.3.2.

Table 2.1 lists a best-known lattice [6] in each dimension, and its related code if it can be constructed from the code. Since searching for potential shaping lattices is of interest, the normalized second moment (2.36) and the shaping gain (2.42) are included, as well as the theoretical highest shaping gain[13] of each dimension provided by an n-sphere (2.38).

2.3.1 $\mathbb{Z}^n, D_n, A_n$ Lattices

Integer Lattice $\mathbb{Z}^n$

The integer lattice (or cubic lattice) $\mathbb{Z}^n$ is in $\mathbb{R}^n$ whose lattice points are n-tuples of integers. A generator matrix for $\mathbb{Z}^n$ is the n-by-n identity matrix $\mathbf{I}_n$ (2.8). At dimension $n = 1$ the integer lattice, denoted $\mathbb{Z}_1$, is important. The case when $n = 2$ as given in Figure 2.3 is also called the square lattice. There are

[13]The 1.53 dB asymptotic shaping gain cannot be achieved by any code at finite dimension.

several applications due to its good properties.

As addressed in the last section, $\mathbb{Z}^n$ can be used to build Construction A, D/D' lattices, where all codewords of a linear code or a family of binary linear codes are lifted to the real space.

Another application of the integer lattice is to produce a hypercubical constellation[14] because the Voronoi region of $\mathbb{Z}^n$ is a hypercube. Thus, instead of transmitting a lattice point with arbitrary large power, the signal power is constrained with respect to the size of a hypercube. However, a hypercubical constellation provides no shaping gain, as explained in Subsection 2.1.7.2.

D_n Lattice

The D_n lattice consists of all integer vectors in $\mathbb{R}^n$ where the sum of n coordinates of $\mathbf{x} = (x_1, x_2, \ldots, x_n)$ is even:

$$D_n = \left\{ \mathbf{x} \in \mathbb{Z}^n \;\middle|\; \sum_{i=1}^{n} x_i \equiv 0 \pmod 2 \right\}. \tag{2.51}$$

The lattice D_2 , as shown in Figure 2.10, looks like a 2-checkerboard and thus is also known as the checkerboard[15] lattice. At dimension $n = 3, 4, 5$ the D_n lattice provides the best normalized second moment. A possible generator matrix for D_n is an n-by-n matrix given by:

$$\begin{bmatrix} -1 & 1 & 0 & \cdots & 0 \\ -1 & -1 & 1 & \cdots & 0 \\ 0 & 0 & -1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & 1 \\ 0 & 0 & 0 & \cdots & -1 \end{bmatrix}. \tag{2.52}$$

The D_n lattice is a Construction A lattice generated by a binary single parity-check (SPC) code. And an alternative generator matrix for D_4 is given by:

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ -1 & 1 & 0 & 0 \\ 0 & -1 & 1 & 0 \\ 0 & 0 & -1 & 2 \end{bmatrix}. \tag{2.53}$$

[14]It is indeed a nested lattice code, but this dissertation only refers to it as hypercube shaping. In this work, a nested lattice code emphasizes that a shaping lattice with shaping gain is used.

[15]The D_n lattice is not similar to the n-checkerboard for $n > 4$.

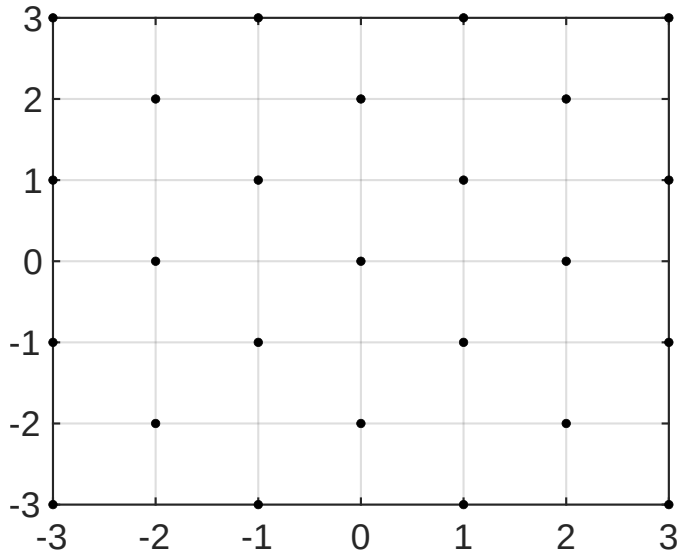


Figure 2.10: Checkerboard lattice D_2.

A_n Lattice

An n-dimensional A_n lattice is a sublattice of $\mathbb{Z}^{n+1}$ that lies in the hyperplane where the sum of the $n+1$ coordinates is zero ($A_n \subset D_{n+1} \subset \mathbb{Z}^{n+1}$), and is defined in $\mathbb{R}^{n+1}$:

$$A_n = \left\{ \mathbf{x} \in \mathbb{Z}^{n+1} \;\middle|\; \sum_{i=1}^{n+1} x_i = 0 \right\}, \tag{2.54}$$

where $\mathbf{x} = (x_1, x_2, \ldots, x_{n+1})$. An $(n+1)$-by-n generator matrix for A_n is:

$$\begin{bmatrix} 1 & 0 & 0 & \cdots & 0 \\ -1 & 1 & 0 & \cdots & 0 \\ 0 & -1 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & 1 \\ 0 & 0 & 0 & \cdots & -1 \end{bmatrix}. \tag{2.55}$$

The A_2 lattice provides the best shaping gain at this dimension, and is equivalent to the hexagonal lattice given in Example 2.5. Thus (2.21) is also a generator matrix for A_2.

2.3.2 E_8 Lattice

The E_8 lattice is also known as the Gosset lattice, that is defined as:

$$E_8 = \left\{ \mathbf{x} \in \mathbb{Z}^8 \;\middle|\; \sum_{i=1}^{n} x_i \equiv 0 \pmod 2 \text{ or } \sum_{i=1}^{n} \left(x_i + \frac{1}{2} \right) \equiv 0 \pmod 2 \right\}, \tag{2.56}$$

where $\mathbf{x} = (x_1, \ldots, x_8)$. A generator matrix is given by:

$$(\frac{1}{2}) \cdot \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 2 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & -2 & 2 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & -2 & 2 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & -2 & 2 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & -2 & 2 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & -2 & 2 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & -2 & 4 \end{bmatrix}. \tag{2.57}$$

The E_8 lattice is the most important lattice at $n = 8$. There are also the 6-dimensional E_6 lattice and 7-dimensional E_7 lattice that are well-known at their dimensions. Recall that a lattice point of E_8 is denoted $\mathbf{x} = (x_1, \ldots, x_8) \in E_8$, then E_6 and E_7 are defined in $\mathbb{R}^8$ as:

$$E_6 = \{\mathbf{x} \in E_8 \mid x_1 = x_2 = x_3\}, \qquad \text{and} \tag{2.58}$$

$$E_7 = \{\mathbf{x} \in E_8 \mid x_1 = x_2\}, \tag{2.59}$$

respectively.

The E_8 lattice can be lifted from an extended Hamming code using Construction A. Let $\mathbf{G}^8$ be a generator matrix of an extended binary Hamming code with

block length $n = 8$ and dimension $k = 4$:

$$\mathbf{G}^8 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{bmatrix}. \tag{2.60}$$

Then a lattice can be obtained by applying Construction A, using (2.49) the following matrix

$$\mathbf{G}_{\Lambda_{\mathrm{A}}^8} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 2 & 0 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 2 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 2 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 & 0 & 2 \end{bmatrix} \tag{2.61}$$

is an alternative generator matrix for the E_8 lattice.

The E_8 lattice has a shaping gain of 0.65 dB, and can be efficiently quantized using the algorithm in [43]. In this dissertation, the direct sum of multiple scaled copies of the E_8 lattice will be used to construct a high-dimensional lattice for shaping a Construction D' lattice, as will be given in Example 4.5 and in Chapter 7.

2.3.3 BW_{16} Lattice

The Barnes-Wall lattice BW_n, introduced by [44], is defined in dimensions $n = 2^2, 2^4, \ldots$ and can be generated from the Reed-Muller codes using Construction B, C, or D [6, pp. 129-131]. For $n = 16$, if using Construction B, a generator matrix

of the BW_{16} lattice is given by:

$$(\frac{1}{\sqrt{2}}) \cdot \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 0 & 2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 1 & 1 & 0 & 0 & 2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 2 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 2 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 2 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 2 & 0 & 0 & 0 \\ 1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 2 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 2 & 0 \\ 1 & 0 & 0 & 0 & 1 & 2 & 2 & 2 & 2 & 2 & 2 & 2 & 2 & 2 & 2 & 4 \end{bmatrix}, \tag{2.62}$$

where the first five columns form a basis of a first-order Reed-Muller code.

When $n = 16$, the Barnes-Wall lattice BW_n can be decoded using [45]. A maximum-likelihood decoding algorithm can be efficiently performed for small n [46], but is impractical as n increases. For decoding BW_n with large n, bounded-distance decoder [47] and the recently proposed recursive bounded-distance decoder [48, 49] are suitable.

The BW_{16} lattice provides a shaping gain of 0.86 dB. Similar to the E_8 lattice, the BW_{16} lattice will also be employed to build a high-dimensional shaping lattice in this dissertation.

2.3.4 Leech Lattice

At dimension $n = 24$, the most famous lattice is the Leech lattice Λ_{24} introduced by J. Leech in 1964 [50]. See also [51]. There have been developed a variety of constructions for Λ_{24}, and one of the constructions is related to the extended binary Golay code $\mathcal{C}$, defined by:

$$\Lambda_{24} = \left\{ \mathbf{x} \in \mathbb{Z}^{24} \;\middle|\; x_i \equiv a \pmod 2,\ \frac{(x_i - a)}{2} \in \mathcal{C},\ \sum_{i=1}^{24} x_i \equiv 4a \pmod 8 \right\}. \tag{2.63}$$

A generator matrix of the Leech lattice is given by:

$$
(\frac{1}{2\sqrt{2}}) \cdot \begin{bmatrix}
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 0 & 2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 2 & 2 & 2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 0 & 0 & 0 & 2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 2 & 0 & 0 & 0 & 2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 0 & 2 & 0 & 0 & 0 & 2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 2 & 2 & 2 & 2 & 2 & 2 & 4 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 0 & 2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 2 & 2 & 2 & 0 & 0 & 0 & 0 & 2 & 2 & 2 & 4 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 0 & 0 & 0 & 2 & 0 & 0 & 0 & 2 & 0 & 0 & 0 & 2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 2 & 0 & 0 & 0 & 2 & 0 & 0 & 0 & 2 & 0 & 0 & 2 & 4 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 0 & 2 & 0 & 0 & 0 & 2 & 0 & 0 & 0 & 2 & 0 & 2 & 0 & 4 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 2 & 2 & 2 & 2 & 2 & 2 & 0 & 2 & 2 & 2 & 0 & 2 & 0 & 0 & 4 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 0 & 0 & 0 & 0 & 0 & 2 & 0 & 2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 0 & 0 & 0 & 2 & 0 & 0 & 0 & 0 & 2 & 0 & 0 & 0 & 0 & 0 & 0 & 2 & 4 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 0 & 0 & 0 & 0 & 2 & 0 & 0 & 0 & 0 & 2 & 0 & 0 & 0 & 0 & 0 & 2 & 0 & 4 & 0 & 0 & 0 & 0 & 0 \\
1 & 0 & 0 & 2 & 2 & 2 & 2 & 0 & 2 & 2 & 2 & 0 & 0 & 0 & 0 & 0 & 2 & 0 & 0 & 4 & 0 & 0 & 0 & 0 \\
1 & 0 & 0 & 2 & 0 & 2 & 0 & 0 & 2 & 0 & 0 & 0 & 2 & 0 & 0 & 0 & 2 & 0 & 0 & 0 & 4 & 0 & 0 & 0 \\
1 & 0 & 0 & 2 & 0 & 0 & 2 & 0 & 0 & 2 & 0 & 0 & 2 & 0 & 0 & 0 & 2 & 0 & 0 & 0 & 0 & 4 & 0 & 0 \\
1 & 0 & 0 & 2 & 2 & 0 & 0 & 0 & 0 & 0 & 2 & 0 & 2 & 0 & 0 & 0 & 2 & 0 & 0 & 0 & 0 & 0 & 4 & 0 \\
-3 & 0 & 0 & 0 & 2 & 2 & 2 & 4 & 2 & 2 & 2 & 4 & 2 & 4 & 4 & 4 & 2 & 4 & 4 & 4 & 4 & 4 & 4 & 8
\end{bmatrix}. \tag{2.64}
$$

There exist several algorithms for quantizing the Leech lattice [5, 45, 52–54]. More efficient algorithms were proposed in [55, 56]. Sphere decoding algorithms given in [57, 58] can also be used for quantization.

Due to its good shaping gain as high as 1.03 dB, the Leech lattice is also suitable for shaping, and will be used for shaping high-dimensional coding lattices in Chapter 7.

2.4 Concluding Remarks

This chapter addressed preliminaries, such as the definition and description of lattices, the Voronoi region of lattices, and other important properties of lattices. They all together form a basis for understanding the following chapters.

Lattice basis, lattice cosets, nested lattices, identical lattices, lattice scaling and direct sum of lattices are corresponding to the design and construction of lattices and lattice codes, as will be described in Chapters 3–7.

Lattices based on Construction A and Construction D' are the main focus of this dissertation. The encoding and decoding of Construction D' will be addressed in Chapter 4. Lattice constructions of using Construction A and Construction D' will be given in Chapters 4–6.

Lattice quantization associated with the normalized second moment as well as the shaping gain will be used for finding convolutional code lattices with best trade off between the shaping gain and the quantization complexity. This part will be presented in Chapter 6.

The E_8 lattice, the $BW16$ lattice and the Leech lattice will contribute the shaping gain by forming nested lattice codes with Construction D' lattices, as will be shown in Chapter 7.

Chapter 3

Nested Lattice Codes

A coding lattice Λ_c and a shaping lattice Λ_s are needed to form a nested lattice code[1]. A nested lattice code is the intersection of Λ_c and the zero-centered Voronoi region of Λ_s, and is also called a Voronoi code/constellation [4, 5]. The shaping lattice Λ_s is used as a lattice quantizer [17] for the coding lattice Λ_c whose average transmit power is thus reduced, and the power reduction is measured by the normalized second moment (NSM) of the Voronoi region of Λ_s. The shaping gain tells how good the NSM of a region is, comparing with the NSM of a hypercube—the Voronoi region of an integer lattice $\mathbb{Z}^n$. Thus hypercubical constellations (or $\mathbb{Z}^n$) have 0 dB of shaping gain.

A nested lattice code $\mathfrak{C}$ can be constructed if and only if Λ_s is a sublattice of Λ_c. This is introduced in Section 3.1. If the shaping lattice Λ_s is an integer-scaled version of the coding lattice Λ_c, then Λ_s and Λ_c form a self-similar nested lattice code. As pointed out in Chapter 2, lattice scaling does not change the error-correction capability if the noise is suitably normalized, and it also scales the Voronoi region but does not change the shaping gain. A self-similar $\mathfrak{C}$ in general does not necessarily provide both good coding and shaping properties because Λ_c and Λ_s have competitive design requirements [30, 31]. A pair of distinct lattices Λ_c and Λ_s forming a nested lattice code $\mathfrak{C}$ is desirable, where Λ_c has good coding properties and Λ_s has good shaping properties. When Λ_s is a sublattice of Λ_c but is not an integer-scaled version of Λ_c, they form a non self-similar nested lattice code. This was used in past work: shaping low-density lattice codes (LDLC) using the E_8 lattice and the BW_{16} lattice [32], convolutional code lattices [9], and shaping LDA lattices using the Leech lattice [33]. These results show that Λ_c and Λ_s can be designed to provide both good coding properties and efficiently achievable shaping gains. Thus in this dissertation, non self-similar nested lattice codes are used.

Encoding maps information to lattice codewords, and indexing is the inverse operation. Conway and Sloane [4] studied encoding and indexing which can only

[1]Some authors refer to a coding lattice as a fine lattice, and call a shaping lattice a coarse lattice.

be applied to self-similar codes. The methods suitable for more general nested lattice codes were proposed in [16, 59] and used in [32]. If the check matrix of Λ_c and the generator matrix of Λ_s are triangularizable, the indexing of nested lattice codes can be straightforwardly performed. In this chapter, the matrices are assumed lower triangular for encoding and indexing. The encoding method is briefly described, followed by the main contribution of this chapter, that is, an indexing algorithm modified from [16, Sec. IV-B]. This modified method overcomes the integer overflow problem for high-dimensional lattices since bounding values for integers are found.

After that, the coding scheme used in this dissertation is addressed, which additionally considers the indexing for nested lattice codes compared with [3], approaching to practical applications. The shaping gain provided by a nested lattice code with respect to the Λ_s can be observed when compared with hypercube shaping, that instead uses a hypercubical constellation. Section 3.5 presents a method to generate hypercubical constellations for Construction D' lattices that is simpler than the conventional method [30].

3.1 Nested Lattice Codes

A nested lattice code $\mathfrak{C}$ is constructed using two lattices: a coding lattice Λ_c and a shaping lattice Λ_s, that satisfy:

$$\Lambda_s \subseteq \Lambda_c, \tag{3.1}$$

which is referred to as the *sublattice condition* [36, p. 179]. Let $\mathcal{V}$ be the zero-centered Voronoi region of the shaping lattice Λ_s, then a nested lattice code is defined by:

$$\mathfrak{C} = \Lambda_c \cap \mathcal{V}. \tag{3.2}$$

A shift of Λ_s by a lattice point $\lambda \in \Lambda_c$ defines a relative coset (2.12) that contains all lattice points of Λ_c relative to Λ_s with respect to a vector $\lambda \in \Lambda_c$, expressed as: $\lambda + \Lambda_s$. The relative cosets form a quotient group Λ_c/Λ_s, and each distinct relative coset can be represented by a unique coset leader in $\mathcal{V}$. Thus $\mathfrak{C}$ is the set of coset leaders of Λ_c/Λ_s. The union of Λ_c/Λ_s is:

$$\Lambda_c = \bigcup_{\lambda' \in \mathfrak{C}} (\lambda' + \Lambda_s), \tag{3.3}$$

which can also be decomposed into the union of shifts of lattice codebook $\mathfrak{C}$ by Λ_s expressed as:

$$\Lambda_c = \bigcup_{\lambda'' \in \Lambda_s} (\lambda'' + \mathfrak{C}). \tag{3.4}$$

A good property of nested lattice codes is that by shaping Λ_c with the Voronoi region $\mathcal{V}$ of Λ_s the resulting codewords meet power constraint, and satisfy a distribution with zero mean.

Let $\mathbf{G}_c$ and $\mathbf{G}_s$ be a generator matrix of a coding lattice Λ_c and a shaping lattice Λ_s respectively. The check matrix of Λ_c is

$$\mathbf{H}_c = \mathbf{G}_c^{-1}. \tag{3.5}$$

Lemma 3.1 [16, Lemma 1] $\Lambda_s \subseteq \Lambda_c$ if and only if $\mathbf{H}_c\mathbf{G}_s$ is a matrix of integers.

Proof Let $\mathbf{G}_s\mathbf{b} \in \Lambda_s$. The point $\mathbf{G}_s\mathbf{b}$ is a point in Λ_c if and only if $\mathbf{H}_c\mathbf{G}_s\mathbf{b}$ is a vector of integers. For an arbitrary $\mathbf{b} \in \mathbb{Z}^n$, this is true if and only if $\mathbf{H}_c\mathbf{G}_s$ is a matrix of integers. □

The codebook size of $\mathfrak{C}$ is given by

$$\begin{aligned} |\mathfrak{C}| &= |\Lambda_c/\Lambda_s| && (3.6)\\ &= \frac{|1/V(\Lambda_c)|}{|1/V(\Lambda_s)|} && (3.7)\\ &= \frac{|V(\Lambda_s)|}{|V(\Lambda_c)|} && (3.8)\\ &= \frac{\left|\det(\mathbf{G}_s)\right|}{\left|\det(\mathbf{G}_c)\right|}. && (3.9) \end{aligned}$$

Thus the rate of a nested lattice code $\mathfrak{C}$ is defined:

$$\begin{aligned} R &= \frac{1}{n}\log_2 \left|\mathfrak{C}\right| && (3.10)\\ &= \frac{1}{n}\log_2 \frac{\left|\det(\mathbf{G}_s)\right|}{\left|\det(\mathbf{G}_c)\right|}. && (3.11) \end{aligned}$$

3.2 Encoding

The mapping from integers to a lattice codeword in $\mathfrak{C}$ is called encoding [16]. Assume that $\mathbf{H}_c$ and $\mathbf{G}_s$ are lower triangular. Let $h_{i,i}$ and $g_{i,i}$ be diagonal elements

of $\mathbf{H}_\mathrm{c}$ and $\mathbf{G}_\mathrm{s}$ for $i = 1, 2, \ldots, n$. It follows that

$$M_i = h_{i,i} g_{i,i} \qquad \text{is a positive integer.} \tag{3.12}$$

Let information be represented by a vector of integers $\mathbf{b}$ where

$$b_i \in \{0, 1, \ldots, M_i - 1\} \tag{3.13}$$

and position i encodes $\log_2 M_i$ bits. Encoding is bijectively mapping a vector of integers $\mathbf{b}$ to a lattice codeword $\mathbf{x}' \in \mathfrak{C}$, where the number of codewords (3.9) can also be written as

$$|\mathfrak{C}| = \prod_{i=1}^{n} M_i. \tag{3.14}$$

The lattice codeword is given by:

$$\mathbf{x}' = \mathbf{x} \bmod \Lambda_\mathrm{s}, \tag{3.15}$$

where $\mathbf{x} \in \Lambda_\mathrm{c}$ can be found by solving[2] $\mathbf{H}_\mathrm{c}\mathbf{x} = \mathbf{b}$. Here $\mathbf{H}_\mathrm{c}$ need not be lower triangular but needs to be triangularizable using a unimodular transformation.

The lattice shaping operation is performed using (3.15). If a codebook (or a constellation) is nonhypercubical, the shaping gain (2.42) can be obtained.

Note that dithering is omitted when encoding and indexing are discussed for simplicity, and will be described in Section 3.4.

3.3 Indexing

The inverse of encoding is called indexing that maps a lattice codeword $\mathbf{x}' \in \mathfrak{C}$ to the vector of integers $\mathbf{b}$ used by the encoder. Note that $\mathbf{x}'$ and $\mathbf{x} = \mathbf{G}_\mathrm{c}\mathbf{b}$ are in the same coset, so when $\mathbf{x} \neq \mathbf{x}'$, in general, using $\mathbf{H}_\mathrm{c}\mathbf{x}'$ cannot recover $\mathbf{b}$ and thus an indexing method is necessary. This can be done by a systematic procedure as suggested in [16, Sec. IV-B]. The modulo-Λ_s expression (3.15) can also be written as

$$\mathbf{x}' = \mathbf{G}_\mathrm{c}\mathbf{b} - Q_{\Lambda_\mathrm{s}}(\mathbf{G}_\mathrm{c}\mathbf{b}), \tag{3.16}$$

[2] As addressed in Subsection 2.1.2, solving $\mathbf{H}_\mathrm{c}\mathbf{x} = \mathbf{b}$ is equivalent to compute $\mathbf{x} = \mathbf{G}_\mathrm{c}\mathbf{b}$ for $\mathbf{H}_\mathrm{c} = \mathbf{G}_\mathrm{c}^{-1}$, which is useful for Construction D' lattices as will be introduced in the next chapter.

where Q_{Λ_s} is a lattice quantizer (2.16) that finds the nearest lattice point in Λ_s given a point. Let

$$\mathbf{b}' = \mathbf{H}_\mathrm{c}\mathbf{x}'. \tag{3.17}$$

Multiply $\mathbf{H}_\mathrm{c}$ on the left of both sides of (3.16) so that

$$\mathbf{b}' = \mathbf{b} - \mathbf{H}_\mathrm{c} Q_{\Lambda_\mathrm{s}}(\mathbf{G}_\mathrm{c}\mathbf{b}). \tag{3.18}$$

The indexing can be performed by finding $\mathbf{t} \in \mathbb{Z}^n$ that satisfies

$$Q_{\Lambda_\mathrm{s}}(\mathbf{G}_\mathrm{c}\mathbf{b}) = \mathbf{G}_\mathrm{s}\mathbf{t} \tag{3.19}$$

such that

$$\mathbf{b}' = \mathbf{b} - \mathbf{H}_\mathrm{c}\mathbf{G}_\mathrm{s}\mathbf{t}, \tag{3.20}$$

where $\mathbf{G}_\mathrm{s}\mathbf{t}$ is the nearest lattice point in Λ_s of the lattice point $\mathbf{x} = \mathbf{G}_\mathrm{c}\mathbf{b} \in \Lambda_\mathrm{c}$.

The triangular structure of $\mathbf{H}_\mathrm{c}\mathbf{G}_\mathrm{s}$ is used which is expressed as

$$\mathbf{H}_\mathrm{c}\mathbf{G}_\mathrm{s} = \begin{bmatrix} \theta_{1,1} & 0 & \cdots & 0 \\ \theta_{2,1} & \theta_{2,2} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ \theta_{n,1} & \theta_{n,2} & \cdots & \theta_{n,n} \end{bmatrix}, \tag{3.21}$$

where $\theta_{i,i} = M_i$ as given in (3.12). The indexing algorithm was described in [16, Sec. IV-B] and is quoted[3] here.

[3]The notation was changed accommodate to the expression used in this chapter. Also, the notation $\mathbf{t}$ and t used in this section is distinct from that of (2.18) and Section 3.5.

The first line of (3.20) is

$$b_1' = b_1 - M_1 t_1, \tag{3.22}$$

which has solution b_1 and t_1 given by

$$b_1 = b_1' \bmod M_1, \qquad \text{and} \tag{3.23}$$

$$t_1 = \frac{b_1 - b_1'}{M_1}. \tag{3.24}$$

For following lines $i = 2, \ldots, n$:

$$b_i' = b_i - \Big(\sum_{j=1}^{i-1} \theta_{i,j} t_j\Big) - M_i t_i \tag{3.25}$$

has solution

$$b_i = \Big(b_i' + \sum_{j=1}^{i-1} \theta_{i,j} t_j\Big) \bmod M_i, \qquad \text{and} \tag{3.26}$$

$$t_i = \frac{b_i - b_i' - \sum_{j=1}^{i-1} \theta_{i,j} t_j}{M_i}. \tag{3.27}$$

Consider high-dimensional nested lattice codes. As the integers b_i, t_i are found sequentially (3.25), the values for t_i (3.27) can become extremely large[4] which leads to an integer overflow problem in practical implementations, depending on the elements of $\mathbf{H}_\mathrm{c}\mathbf{G}_\mathrm{s}$ and especially when $\mathbf{G}_\mathrm{s}$ has large scaling. An integer overflow problem will cause a failure when recovering the information even in the absence of noise, and thus needs to be avoided.

3.3.1 Indexing of High-Dimensional Nested Lattice Codes

Now a modified method suitable for indexing high-dimensional nested lattice codes is proposed. This is to obtain the same solution of $\mathbf{b} \in \mathbb{Z}^n$ given a lattice codeword $\mathbf{x}' \in \mathfrak{C}$ without computing the sum of large-valued integers used internally. The

[4]For example, when performing the indexing for the code construction of using convolutional code lattice scaled by 20 which is based on polynomials $(73, 57, 41)$ as will be shown in Subsection 7.2.2, it might involve the value of t_i as large as 10^{30} at dimension $n = 1152$ and 10^{300} at dimension $n = 10008$. Even though $\mathbf{t}$ is only used internally, in MATLAB implementation, the latter case will cause an integer overflow problem.

principle is to use the linearity. Since a lattice is linear, a lattice point can be represented by the sum of two lattice points. A lattice is generated by the integer combinations of its generator basis vectors and thus its corresponding integer vector can also be written as the sum of two vectors of integers.

Instead of using (3.20) an integer vector $\mathbf{s} \in \mathbb{Z}^n$ is introduced such that

$$\mathbf{b}' = \mathbf{b} + \mathbf{H}_{\mathrm{c}}\mathbf{G}_{\mathrm{s}}\mathbf{s} - \mathbf{H}_{\mathrm{c}}\mathbf{G}_{\mathrm{s}}\mathbf{e}, \tag{3.28}$$

where

$$\mathbf{e} = \mathbf{t} + \mathbf{s} \tag{3.29}$$

will be shown to be a vector of bounded-valued integers. The solution $\mathbf{b}$ can be found without explicitly computing $\mathbf{t}$ and $\mathbf{s}$, thus the integer overflow problem can be avoided. How to obtain a value of $\mathbf{s}$ without changing the solution of $\mathbf{b}$ will be given in Appendix A and the value might not be unique. However it needs not be explicitly calculated as will be shown in the remaining of this subsection.

To find $\mathbf{b}$ and $\mathbf{e}$, these equations are solved sequentially first for $i = 1$, then $i = 2, \ldots, n$. The first line of (3.28) is

$$b_1' = b_1 + M_1 s_1 - M_1 e_1. \tag{3.30}$$

Then for $i = 2, \ldots, n$:

$$b_i' = b_i + \sum_{j=1}^{i-1} \theta_{i,j} s_j + M_i s_i - \sum_{j=1}^{i-1} \theta_{i,j} e_j - M_i e_i. \tag{3.31}$$

Firstly, the solution of b_i is found as follows. To obtain e_i, write

$$q_i = s_i + \sum_{j=1}^{i-1} \frac{\theta_{i,j}}{M_i} s_j \tag{3.32}$$

but s_i need not be computed. Then q_i should be chosen such that e_i is bounded and after e_i is obtained as indexing proceeds, the value is used for $i+1, \ldots, n$. The solution $\mathbf{b}$ of (3.28) is the same as that of (3.20) by choosing q_i such that

$$\frac{q_i}{\operatorname{lcm}(M_{i+1}, \ldots, M_n)} \qquad \text{is an integer.} \tag{3.33}$$

The algorithm is given as follows. The solution of (3.30) is b_1 and e_1 given by

$$b_1 = b_1' \bmod M_1, \qquad \text{and} \tag{3.34}$$

$$e_1 = \frac{b_1 - b_1'}{M_1} \bmod \operatorname{lcm}(M_2, M_3, \ldots, M_n). \tag{3.35}$$

Then for $i = 2, \ldots, n$, (3.31) has solution b_i and e_i given by

$$b_i = b'_i + \sum_{j=1}^{i-1} \theta_{i,j} e_j \bmod M_i, \qquad \text{and} \tag{3.36}$$

$$e_i = \frac{b_i - b'_i - \sum_{j=1}^{i-1} \theta_{i,j} e_j}{M_i} \bmod \operatorname{lcm}(M_{i+1}, \ldots, M_n) \tag{3.37}$$

where the integer $0 \leqslant e_i < \operatorname{lcm}(M_{i+1}, \ldots, M_n)$ is thus bounded—this is practical.

Triangular $\mathbf{H}_\mathrm{c}$ and $\mathbf{G}_\mathrm{s}$ allow efficient encoding and indexing, where $\mathbf{H}_\mathrm{c}$ and $\mathbf{G}_\mathrm{s}$ can be obtained from triangularizable full-rank check matrix and generator matrix of Λ_c and Λ_s respectively. The author has not yet found a straightforward method to index nested lattice codes using non-triangular matrices.

3.4 Coding Scheme

Erez and Zamir [3] proposed a coding scheme using nested lattice codes with dithering and minimum mean-square error (MMSE) scaling techniques that can achieve the capacity of the power-constrained AWGN channel, which is transformed into a modulo-lattice additive noise channel. This dissertation uses a similar coding scheme, but additionally include the indexing. Since this work considers primarily high rate codes in the high-SNR domain, the MMSE scaling is close to 1. As proven by di Pietro, Zémor, and Boutros [15], dithering is not mandatory because lattice points of Λ_c at high code rate fill well in the Voronoi region of Λ_s.

Let the dither $\mathbf{U}$ be uniformly distributed in Voronoi region of Λ_s, which is independent of the lattice point $\mathbf{x}$ of Λ_c. Instead of using (3.15), a vector

$$\mathbf{x}'' = \mathbf{x} - \mathbf{U} \bmod \Lambda_\mathrm{s} \tag{3.38}$$

is sent to the AWGN channel. The average transmitted power per symbol

$$E_\mathrm{s} = \frac{1}{n}\mathrm{E}[\|\mathbf{x}''\|^2] = \frac{1}{n}\mathrm{E}\left[\|\mathbf{U}\|^2\right], \tag{3.39}$$

can also be represented by

$$E_\mathrm{s} = \mathrm{NSM} \cdot V^{2/n}(\Lambda_\mathrm{s}), \tag{3.40}$$

where NSM is the normalized second moment[5] and $V(\Lambda_\mathrm{s})$ is the volume of Λ_s. The MMSE scaling coefficient α is defined

$$\alpha = \frac{E_\mathrm{s}}{E_\mathrm{s} + \sigma^2}, \tag{3.41}$$

[5]See equations (2.36) and (2.41).

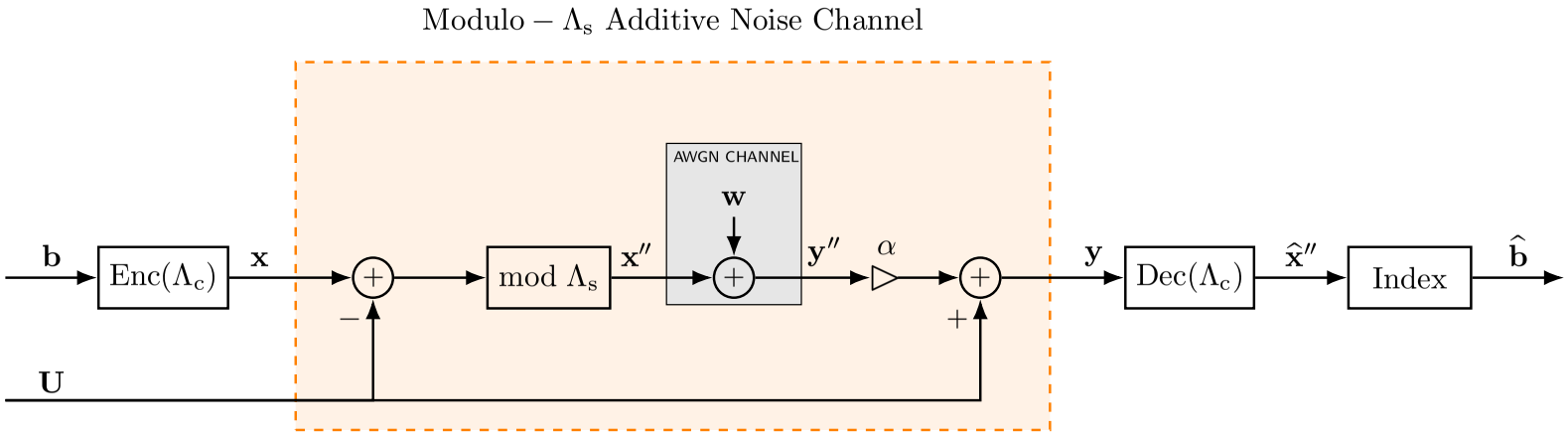


Figure 3.1: Block diagram of nested lattice codes with a dither variable $\mathbf{U}$ uniformly distributed over the Voronoi region of Λ_s and a "Wiener coefficient" α was chosen for MMSE.

where $0 \leqslant \alpha \leqslant 1$.

The signal-to-noise ratio (SNR) is defined as

$$\mathrm{SNR} = \frac{E_\mathrm{s}}{\sigma^2}. \tag{3.42}$$

Thus α can also be expressed

$$\alpha = \frac{\mathrm{SNR}}{1 + \mathrm{SNR}}. \tag{3.43}$$

Given a received sequence

$$\mathbf{y}'' = \mathbf{x}'' + \mathbf{w}, \tag{3.44}$$

where $\mathbf{w}$ is noise, the input to the decoder is computed

$$\mathbf{y} = \alpha \mathbf{y}'' + \mathbf{U}. \tag{3.45}$$

See [3, 33].

The average transmitted power per bit can be computed

$$E_\mathrm{b} = \frac{E_\mathrm{s}}{R}. \tag{3.46}$$

This dissertation measures the decoding error rate of nested lattice codes as a function of SNR per bit, expressed as

$$\frac{E_\mathrm{b}}{N_0} = \frac{E_\mathrm{b}}{2\sigma^2} \tag{3.47}$$

$$= \frac{\mathrm{SNR}}{2R}, \tag{3.48}$$

where N_0 is the noise power spectral density, and E_b/N_0 is given in decibels as: $10 \log_{10}(E_\mathrm{b}/N_0)$ dB. To observe the shaping gains, it is convenient to define the Shannon limit in terms of E_b/N_0 as

$$\frac{E_\mathrm{b}}{N_0}_{\text{ Shannon limit}} = 10 \log_{10} \frac{2^{2R} - 1}{2R} \tag{3.49}$$

given in decibels.

3.5 Hypercube Shaping

As addressed in Section 3.2, a nonhypercubical constellation provides the shaping gain. For comparison, hypercube shaping is introduced. This was used for LDLC lattices [30], the E_8 lattice [60], and for Construction A lattices based on LDPC codes [31]. Lattice points of Λ_c are transformed into lattice points in a hypercube[6]

$$\mathcal{B} = \{0, 1, \ldots, L-1\}^n, \tag{3.50}$$

for an even integer L. This allows lattices to be evaluated in a power-constrained channel, but no shaping gain is obtained. How to generate a hypercubical constellation is briefly described and then simplified especially for Construction D' lattices.

Given a vector of integers $\mathbf{b}^\dagger$, in general a lattice point $\mathbf{x}^\dagger = \mathbf{G}_c\mathbf{b}^\dagger$ is not in $\mathcal{B}$. Note that $\mathbf{G}_c$ is not necessary because $\mathbf{x}^\dagger = \mathbf{G}_c\mathbf{b}^\dagger$ is equivalent to solve $\mathbf{H}_c\mathbf{x}^\dagger = \mathbf{b}^\dagger$. For $i = 1, 2, \ldots, n$, hypercube shaping [30, Subsec. III-A] finds integers $b_i = b_i^\dagger - L_i t_i$ such that a lattice point

$$\mathbf{x} = \mathbf{G}_c\mathbf{b} \in \mathcal{B}, \tag{3.51}$$

for an integer

$$L_i = L h_{i,i}, \tag{3.52}$$

where an integer $b_i^\dagger \in \{0, 1, \ldots, L_i - 1\}$ and $t_i \in \mathbb{Z}^n$. This can be performed as follows.

For $i = 1, \ldots, n$, the integer t_i is found sequentially:

$$t_i = \left\lfloor \frac{1}{L_i}\Big(b_i^\dagger - \sum_{j=1}^{i-1} h_{i,j}x_j\Big) \right\rfloor, \tag{3.53}$$

where $h_{i,j}$ is an entry of $\mathbf{H}_c$ at row i, column j. The solution to (3.53) is

$$b_i = b_i^\dagger - L_i t_i, \qquad \text{and} \tag{3.54}$$

$$x_i = b_i - \sum_{j=1}^{i-1} h_{i,j}x_j. \tag{3.55}$$

[6]The intersection of Λ_c and $\mathcal{B}$ can form a nested lattice code with a hypercubical constellation.

In practice, a vector $\mathbf{x} - \frac{L-1}{2}$ that is uniformly distributed in a hypercube

$$\left\{ -\frac{L-1}{2}, -\frac{L-3}{2}, \ldots, -\frac{1}{2}, \frac{1}{2}, \ldots, \frac{L-3}{2}, \frac{L-1}{2} \right\}^n, \tag{3.56}$$

is transmitted to the channel instead. This is to reduce the average transmitted power, which is thus given by:

$$E'_{\mathrm{s}} = \frac{1}{L} \sum_{j=0}^{L-1} \Big(-\frac{L-1}{2} + j \Big)^2. \tag{3.57}$$

The code rate is defined

$$R' = \frac{1}{n} \log_2 \frac{L^n}{|\det(\mathbf{G}_{\mathrm{c}})|}. \tag{3.58}$$

3.5.1 Simplified Method Performing Hypercube Shaping for Construction D' Lattices

Consider an a-level Construction D' lattice Λ_{c}. Let an integer L be a multiple of 2^a. The procedure given above can be simplified. Let Λ_{c} be described by a lower-triangular check matrix $\mathbf{H}_{\mathrm{c}}$ with diagonal elements $h_{i,i}$ for $i = 1, \ldots, n$, and let $L\mathbf{I}_n$ be a generator matrix of the "shaping lattice" $L\mathbb{Z}^n$ where $\mathbf{I}_n$ is an identity matrix of size n. Choose L such that the product of $\mathbf{H}_{\mathrm{c}}$ and $L\mathbf{I}_n$ is a matrix of integers. The information vector consists of integers in $\{0, 1, \ldots, Lh_{i,i} - 1\}$. Performing modulo-$L$ on a lattice point $\mathbf{x}^\dagger$ of Λ_{c} is the "shaping" operation reducing $\mathbf{x}^\dagger$ into a hypercube $\{0, 1, \ldots, L-1\}^n$, written as

$$\mathbf{x} = \mathbf{x}^\dagger \bmod L, \tag{3.59}$$

which does not require sequential computation. Recover the integers from an estimated lattice point $\widehat{\mathbf{x}}$ is straightforward. Let $\widehat{\mathbf{b}} = \lfloor \mathbf{H}_{\mathrm{c}} \cdot \widehat{\mathbf{x}} \rceil$, then

$$\widehat{b}_i^\dagger = \widehat{b}_i \bmod Lh_{i,i} \tag{3.60}$$

is computed, for $i = 1, 2, \ldots, n$.

The work in [12] can also produce a hypercubical constellation for Construction D', but this dissertation performs hypercube shaping with respect to the decoding algorithm that will be described in Chapter 4.

For an a-level Construction D' lattice with hypercube shaping, it is natural to use 2^a-PAM signalling. For the shaped lattice codes in this dissertation, the lattice

points $\mathbf{x}$ are integers due the use of Construction D' as will be introduced in the next chapter; however greater than 2^a modulation levels are required. Construction D' lattices with hypercube shaping can also use greater than 2^a modulation levels, but no shaping gain is provided.

3.6 Concluding Remarks

Nested lattice codes can provide both good coding properties and high shaping gain, if the component lattices are chosen with desirable properties and using a coding scheme described in [3]. And this nested lattice coding scheme was extended in this dissertation such that indexing operation was included. If the underlying lattices have high dimensions, an integer overflow problem might occur when the existing indexing algorithm is implemented—this can be avoided using the modified algorithm proposed in this chapter. Hypercube shaping was also introduced for comparisons, as will be shown in Chapter 7.

Chapter 4

Construction D' Lattices

Construction D' lattices are built from nested binary linear codes. The definition of nested binary linear codes is reviewed. Then a definition of Construction D' using a check-matrix perspective which is equivalent to the congruences definition is presented. After that, how to form lattices from nested binary codes using Construction D' is shown. Lastly two equivalent encoding methods and a decoding algorithm for Construction D' lattices to be used in power-constrained channels are proposed. In this chapter the check matrix instead of the generator matrix is used, because it provides the benefit for lattices designed using LDPC codes, which are conveniently described by parity-check matrices.

4.1 Lattices Based on Construction D'

Lattices can be constructed using Construction D' and nested binary linear codes.

4.1.1 Nested Linear Codes

Definition 4.1 Let row vectors $\mathbf{h}_1, \mathbf{h}_2, \ldots, \mathbf{h}_n$ be a basis for $\mathbb{F}_2^n$. For level $a \geqslant 1$, $\mathcal{C}_0 \subseteq \mathcal{C}_1 \subseteq \cdots \subseteq \mathcal{C}_a = \mathbb{F}_2^n$ are nested linear codes if $\mathbf{h}_{k_i+1}, \ldots, \mathbf{h}_n$ are $r_i = n - k_i$ parity-checks for $\mathcal{C}_i$, where k_i denotes the dimension of code $\mathcal{C}_i$ whose rate is $R_i = k_i/n$. That is, a codeword $\widetilde{\mathbf{x}} \in \mathcal{C}_i$ if and only if:

$$\mathbf{h}_j \odot \widetilde{\mathbf{x}} = 0, \tag{4.1}$$

for $j = k_i + 1, k_i + 2, \ldots, n$ and $i = 0, 1, \ldots, a - 1$.

The n-by-n matrix of row vectors is denoted

$$\widetilde{\mathbf{H}} = \begin{bmatrix} — & \mathbf{h}_1 & — \\ — & \mathbf{h}_2 & — \\ & \vdots & \\ — & \mathbf{h}_n & — \end{bmatrix} \tag{4.2}$$

The matrix $\widetilde{\mathbf{H}}_0$ is the parity-check matrix for $\mathcal{C}_0$, and consists of r_0 rows, from $\mathbf{h}_{k_0+1}$ to $\mathbf{h}_n$. The matrix $\widetilde{\mathbf{H}}_1$ is the parity-check matrix for $\mathcal{C}_1$, and consists of r_1 rows, from $\mathbf{h}_{k_1+1}$ to $\mathbf{h}_n$, and so on. This illustrates that the parity-check matrix for $\mathcal{C}_0$ contains the check matrices for the supercodes $\mathcal{C}_1, \ldots, \mathcal{C}_{a-1}$. The basis vectors $\mathbf{h}_1$ to $\mathbf{h}_{k_0}$ do not contribute to the error-correction capability of the code, but are selected so that $\widetilde{\mathbf{H}}$ is a unimodular [1] matrix.

4.1.2 Definition of Construction D'

Construction D' converts a set of parity-checks defining nested linear codes into congruences for a lattice [6, p. 235]. A vector $\mathbf{x}$ satisfies a congruence $\mathbf{h} = [h_1, \ldots, h_n]$ with respect to a modulo value q if:

$$\mathbf{h} \cdot \mathbf{x} \equiv 0 \pmod{q}. \tag{4.3}$$

A congruence can be expressed in an equivalent way. Let $\mathbf{h}' = \mathbf{h}/q$. Then $\mathbf{x}$ satisfies this congruence if and only if:

$$\mathbf{h}' \cdot \mathbf{x} \qquad \text{is an integer.} \tag{4.4}$$

Any $\mathbf{x}$ satisfying (4.3) will also satisfy (4.4).

Two equivalent definitions of Construction D' are given. The conventional definition of Construction D' uses congruences of parity-checks of nested binary codes.

Definition 4.2 [Construction D' (congruences)] [6, p. 235] Let $\mathcal{C}_0 \subseteq \mathcal{C}_1 \subseteq \cdots \subseteq \mathcal{C}_a = \mathbb{F}_2^n$ be nested binary linear codes. Let the dimension of $\mathcal{C}_i$ be k_i. Let $\mathbf{h}_1, \mathbf{h}_2, \ldots, \mathbf{h}_n$ be a basis for $\mathbb{F}_2^n$ such that $\mathcal{C}_i$ is defined by $n - k_i$ parity-check vectors $\mathbf{h}_{k_i+1}, \ldots, \mathbf{h}_n$. Then the Construction D' lattice is the set of all vectors $\mathbf{x} \in \mathbb{Z}^n$ satisfying the congruences:

$$\mathbf{h}_j \cdot \mathbf{x} \equiv 0 \pmod{2^{i+1}}, \tag{4.5}$$

[1] A unimodular matrix is a square integer matrix with determinant $+1$ or -1.

for all $i \in \{0, \ldots, a-1\}$ and $k_i + 1 \leqslant j \leqslant n$.

Instead of congruences, the following definition uses the check matrix which is defined as $\mathbf{H} = \mathbf{G}^{-1}$ where $\mathbf{G}$ is a generator matrix. This is the definition used in low-density lattice codes [7], and is distinct from the definition of [11]. Note also that the check matrix of a Construction D' lattice is related to, but distinct from, the parity-check matrices of the corresponding binary codes.

Definition 4.3 [Construction D' (check matrix)] Let a unimodular matrix $\widetilde{\mathbf{H}}$ be the parity-check matrix of nested linear codes $\mathcal{C}_0 \subset \mathcal{C}_1 \subset \cdots \subset \mathcal{C}_a = \mathbb{F}_2^n$. The dimension of $\mathcal{C}_i$ is k_i for $i = 0, 1, \ldots, a$, and it has $k_i < k_{i+1}$. Let $\mathbf{D}$ be a diagonal matrix with entries:

$$d_{j,j} = 2^{-i}, \tag{4.6}$$

for $k_{i-1} < j \leqslant k_i$ where $k_{-1} = 0$ and $k_a = n$. Then the Construction D' lattice is the set of all vectors $\mathbf{x}$ satisfying:

$$\mathbf{H} \cdot \mathbf{x} \qquad \text{are integers,} \tag{4.7}$$

where

$$\mathbf{H} = \mathbf{D} \cdot \widetilde{\mathbf{H}} \tag{4.8}$$

is the lattice check matrix.

The following proposition shows that the two definitions are equivalent.

Proposition 4.1. Let $\mathbf{h}_1, \ldots, \mathbf{h}_n$ in Definition 4.2 be the rows of $\widetilde{\mathbf{H}}$ in Definition 4.3. Then the lattice given by Definition 4.2 is identical to the lattice of Definition 4.3.

Proof It should be clear that because the congruences in (4.5) can be expressed as (4.4), then relevant rows of check matrix $\mathbf{H}$ are an alternative form of the respective congruences. However, the definition of check matrix $\mathbf{H}$ in this chapter does not include Definition 4.2's restriction to $\mathbf{x} \in \mathbb{Z}^n$. To achieve this, it is required that $\widetilde{\mathbf{H}}$ is unimodular, so that the Construction D' lattice in Definition 4.3 satisfies $\Lambda \subset \mathbb{Z}^n$. To see this, $\mathbf{G} = \mathbf{H}^{-1} = \widetilde{\mathbf{H}}^{-1} \cdot \mathbf{D}^{-1}$. Since $\widetilde{\mathbf{H}}$ is unimodular, $\widetilde{\mathbf{H}}^{-1}$ is an integer matrix. $\mathbf{D}^{-1}$ also is a matrix of integers. Thus $\mathbf{G}$ is an integer matrix and $\Lambda \subset \mathbb{Z}^n$. □

As a matter of design, after $\widetilde{\mathbf{H}}_0$ to $\widetilde{\mathbf{H}}_{a-1}$ are fixed, the upper rows of $\widetilde{\mathbf{H}}$ should be chosen such that $\widetilde{\mathbf{H}}$ is unimodular; it is also convenient to choose these upper rows so that $\widetilde{\mathbf{H}}$ is approximate lower triangular (ALT) form.

Volume of Construction D' Lattices

The volume of an n-dimensional Construction D' lattice Λ is given

$$V(\Lambda) = 2^{an-\sum_{i=0}^{a-1} k_i}, \qquad \text{or} \tag{4.9}$$

$$= 2^{\sum_{i=0}^{a-1} r_i}. \tag{4.10}$$

Since the code rate of $\mathcal{C}_i$ is $R_i = k_i/n$, the volume can also be expressed as

$$V(\Lambda) = 2^{an-n\sum_{i=0}^{a-1} R_i}. \tag{4.11}$$

It will sometimes be convenient to write

$$V(\Lambda)^{2/n} = 4^{a-\sum_{i=0}^{a-1} R_i}. \tag{4.12}$$

Thus the VNR is simply computed using (2.35).

Example 4.1 Let nested binary codes $\mathcal{C}_0 \subset \mathcal{C}_1$ be described by parity-check matrices $\widetilde{\mathbf{H}}_0, \widetilde{\mathbf{H}}_1$, given by:

$$\widetilde{\mathbf{H}}_0 = \begin{bmatrix} 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 0 \end{bmatrix}, \qquad \text{and} \tag{4.13}$$

$$\widetilde{\mathbf{H}}_1 = \begin{bmatrix} 1 & 0 & 1 & 0 \end{bmatrix}, \tag{4.14}$$

where the parity check of $\mathcal{C}_1$ is also a parity check for $\mathcal{C}_0$. The subcode $\mathcal{C}_0$ has dimension $k_0 = 1$ and rate $R_0 = 1/4$. The supercode $\mathcal{C}_1$ has dimension $k_1 = 3$ and rate $R_1 = 3/4$.

Add a top row $[1, 0, 0, 0]$ to $\widetilde{\mathbf{H}}_0$ such that the resulting matrix

$$\widetilde{\mathbf{H}} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 0 \end{bmatrix} \tag{4.15}$$

is unimodular. Let a diagonal matrix $\mathbf{D}$ be

$$\mathbf{D} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1/2 & 0 & 0 \\ 0 & 0 & 1/2 & 0 \\ 0 & 0 & 0 & 1/4 \end{bmatrix}. \tag{4.16}$$

Then the check matrix of an $a = 2$-level Construction D' lattice $\Lambda^4_{D'}$ is obtained using Definition 4.3:

$$\mathbf{H} = \mathbf{D} \cdot \widetilde{\mathbf{H}} \tag{4.17}$$

$$= \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1/2 & 0 & 0 \\ 0 & 0 & 1/2 & 0 \\ 0 & 0 & 0 & 1/4 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 0 \end{bmatrix} \tag{4.18}$$

$$= \begin{bmatrix} 1 & 0 & 0 & 0 \\ 1/2 & 0 & 0 & 1/2 \\ 0 & 1/2 & 1/2 & 1/2 \\ 1/4 & 0 & 1/4 & 0 \end{bmatrix}, \tag{4.19}$$

which is a real-valued square matrix. The volume of the $n = 4$-dimensional lattice $\Lambda^4_{D'}$ is obtained using (4.11): $V(\Lambda^4_{D'}) = 16$.

Whether a vector is a lattice point of $\Lambda^4_{D'}$ can be verified using (4.7). Given two vectors

$$\mathbf{x}_1 = \begin{bmatrix} 1 \\ -36 \\ 35 \\ 3 \end{bmatrix}, \qquad \text{and} \qquad \mathbf{x}_2 = \begin{bmatrix} 1 \\ 1 \\ 2 \\ 6 \end{bmatrix}. \tag{4.20}$$

Observe that $\mathbf{x}_1 \in \Lambda^4_{D'}$ but $\mathbf{x}_2 \notin \Lambda^4_{D'}$. This is because

$$\mathbf{H} \cdot \mathbf{x}_1 = \begin{bmatrix} 1 \\ 2 \\ 1 \\ 9 \end{bmatrix} \qquad \text{is a vector of integers,} \tag{4.21}$$

but

$$\mathbf{H} \cdot \mathbf{x}_2 = \begin{bmatrix} 1 \\ 7/2 \\ 9/2 \\ 3/4 \end{bmatrix} \qquad \text{contains non-integer entries.} \tag{4.22}$$

If taking the inverse of $\mathbf{H}$ the generator matrix of $\Lambda^4_{D'}$ is written as:

$$\mathbf{G} = \mathbf{H}^{-1} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 2 & -2 & 2 & -4 \\ -1 & 0 & 0 & 4 \\ -1 & 2 & 0 & 0 \end{bmatrix}, \tag{4.23}$$

and any integer vector $\mathbf{b} \in \mathbb{Z}^4$ produces $\mathbf{G} \cdot \mathbf{b} \neq \mathbf{x}_2$.

Also, the congruences perspective in Definition 4.2 can be used for verification. Write the parity-check basis vectors of $\widetilde{\mathbf{H}}$ as $\mathbf{h}_1 = [1, 0, 0, 0]$, $\mathbf{h}_2 = [1, 0, 0, 1]$, $\mathbf{h}_3 = [0, 1, 1, 1]$, and $\mathbf{h}_4 = [1, 0, 1, 0]$. When $i = 0$, for $\mathbf{x}_1$ there are $\mathbf{h}_2 \cdot \mathbf{x}_1 \bmod 2 = 0$, $\mathbf{h}_3 \cdot \mathbf{x}_1 \bmod 2 = 0$, and $\mathbf{h}_4 \cdot \mathbf{x}_1 \bmod 2 = 0$, but for $\mathbf{x}_2$ there are $\mathbf{h}_2 \cdot \mathbf{x}_2 \bmod 2 = 1$, $\mathbf{h}_3 \cdot \mathbf{x}_2 \bmod 2 = 1$, and $\mathbf{h}_4 \cdot \mathbf{x}_2 \bmod 2 = 1$. Similarly when $i = 1$, there is $\mathbf{h}_4 \cdot \mathbf{x}_1 \bmod 4 = 0$ but $\mathbf{h}_4 \cdot \mathbf{x}_2 \bmod 4 = 3$. Observe that $\mathbf{x}_1$ satisfies (4.5) but $\mathbf{x}_2$ does not, thus $\mathbf{x}_1$ is a lattice point in $\Lambda^4_{D'}$ but $\mathbf{x}_2$ isn't.

4.2 Encoding

Two equivalent encoding methods are given. Encoding method A finds a lattice point $\mathbf{x}$ given $\mathbf{b} \in \mathbb{Z}^n$ using its check matrix $\mathbf{H}$ in the ALT form. Encoding method B describes explicitly how information bits $\mathbf{u}_i$ of the component binary linear code $\mathcal{C}_i$ are mapped to a vector of integers $\mathbf{b}$ and a lattice point. The two encoding methods can be applied to produce nonhypercubical constellations, which is distinct from the encoding in [12].

4.2.1 Encoding Method A

Near linear-time encoding of LDPC codes can be accomplished using parity-check matrix in the ALT form [61]. This subsection draws inspiration from this idea, to implement encoding of Construction D' lattice Λ with a similar procedure. The steps are distinct from [61] because check matrix $\mathbf{H}$ of Λ is a real-valued square matrix.

A vector of integers $\mathbf{b}$ is provided and its corresponding lattice point $\mathbf{x}$ is found by solving:

$$\mathbf{H} \cdot \mathbf{x} = \mathbf{b}. \tag{4.24}$$

If $\mathbf{H}$ is not too big, then $\mathbf{x}$ can be found by matrix inversion:

$$\mathbf{x} = \mathbf{H}^{-1} \cdot \mathbf{b}. \tag{4.25}$$

If $\mathbf{H}$ is large but is sparse and in the ALT form, as may be expected for Construction D' lattices based on LDPC codes, then the following procedure can be used.

Suppose that $\mathbf{H}$ is in the ALT form, that is, it is partially lower triangular. Specifically, $\mathbf{H}$ can be written as:

$$\mathbf{H} = \begin{bmatrix} \mathbf{B} & \mathbf{A} \\ \mathbf{X} & \mathbf{C} \end{bmatrix}, \tag{4.26}$$

where $\mathbf{A}$ is an s-by-s lower-triangular matrix with non-zero elements on the diagonal; $\mathbf{X}$ is a g-by-g square matrix. The "gap" is g—the smaller the gap, the easier the encoding. Let

$$\Delta = (\mathbf{X} - \mathbf{C}\mathbf{A}^{-1}\mathbf{B})^{-1}. \tag{4.27}$$

The blockwise inverse [62] of $\mathbf{H}$ is:

$$\mathbf{H}^{-1} = \begin{bmatrix} -\Delta\mathbf{C}\mathbf{A}^{-1} & \Delta \\ \mathbf{A}^{-1} + \mathbf{A}^{-1}\mathbf{B}\Delta\mathbf{C}\mathbf{A}^{-1} & -\mathbf{A}^{-1}\mathbf{B}\Delta \end{bmatrix}. \tag{4.28}$$

Using the block structure, $\mathbf{H} \cdot \mathbf{x} = \mathbf{b}$ can be written as:

$$\begin{bmatrix} \mathbf{B} & \mathbf{A} \\ \mathbf{X} & \mathbf{C} \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ \vdots \\ x_g \\ x_{g+1} \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} b_1 \\ \vdots \\ b_g \\ b_{g+1} \\ \vdots \\ b_n \end{bmatrix}. \tag{4.29}$$

To perform encoding, first $x_1, \ldots, x_g$ are found using (4.28):

$$\begin{bmatrix} x_1 \\ \vdots \\ x_g \end{bmatrix} = \begin{bmatrix} -\Delta\mathbf{C}\mathbf{A}^{-1} & \Delta \end{bmatrix} \cdot \mathbf{b}. \tag{4.30}$$

Then, coordinates $x_{g+1}, \ldots, x_n$ are found sequentially by back-substitution, using the lower triangular structure of $\mathbf{H}$ which has entry $h_{j,w}$ in row j, column w. For $w = g+1, \ldots, n$:

$$x_w = \frac{1}{h_{j,w}} \left(b_j - \sum_{l=1}^{w-1} h_{j,l} x_l \right) \tag{4.31}$$

where $j = w - g$.

This method is efficient when g is small and $\mathbf{H}$ is sparse. It uses pre-computation and storage of the g-by-n matrix in (4.30). The sum in (4.31) is

performed over the few non-zero terms in sparse $\mathbf{H}$. If the check matrix $\mathbf{H}$ is purely triangular, then encoding is simply performed by back-substitution.

Example 4.2 Consider a 10-dimensional Construction D' lattice $\Lambda_{D'}^{10}$ generated by nested binary codes $\mathcal{C}_0 \subset \mathcal{C}_1$ with parity-check matrix $\widetilde{\mathbf{H}}_0$ and $\widetilde{\mathbf{H}}_1$, respectively. Let $\Lambda_{D'}^{10}$ be described by a check matrix $\mathbf{H}$ in the ALT form, expressed as:

$$\mathbf{H} = \left[\begin{array}{cc:cccccccc}
1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1/2 & 1/2 & 0 & 1/2 & 0 & 0 & 0 & 0 \\
1/2 & 0 & 0 & 1/2 & 0 & 1/2 & 1/2 & 0 & 0 & 0 \\
1/2 & 0 & 1/2 & 1/2 & 0 & 0 & 0 & 1/2 & 0 & 0 \\
0 & 0 & 0 & 1/2 & 1/2 & 0 & 0 & 1/2 & 1/2 & 0 \\
0 & 0 & 0 & 1/4 & 1/4 & 0 & 1/4 & 1/4 & 0 & 1/4 \\
\hdashline
1/4 & 1/4 & 1/4 & 0 & 0 & 1/4 & 0 & 0 & 0 & 1/4 \\
1/4 & 0 & 1/4 & 1/4 & 0 & 1/4 & 1/4 & 0 & 1/4 & 0
\end{array}\right]
\begin{array}{l}
\left.\vphantom{\begin{array}{c}1\\1\\1\\1\\1\\1\\1\\1\end{array}}\right\} \tfrac{1}{2}\widetilde{\mathbf{H}}_0 \\
\left.\vphantom{\begin{array}{c}1\\1\end{array}}\right\} \tfrac{1}{4}\widetilde{\mathbf{H}}_1
\end{array}, \tag{4.32}$$

where the block partition follows (4.26). Assume an integer vector $\mathbf{b} = [1, 2, 0, 2, 4, 0, 2, 0, 2, 1]^{\mathrm{t}}$. Using (4.30) the first two positions of the lattice point $\mathbf{x}$ are computed: $x_1 = -11, x_2 = 52$. Then applying (4.31) a lattice point $\mathbf{x} = [-11, 52, 12, -10, 21, 2, 27, 9, -16, -47]^{\mathrm{t}}$ is obtained.

4.2.2 Encoding Method B

Encoding can also be performed by mapping the message sequence consisting of information vectors $\mathbf{u}_i \in \mathbb{F}_2^{k_i}$ of the component binary codes $\mathcal{C}_i$ for $i = 0, 1, \ldots, a-1$ and an integer vector $\mathbf{z} \in \mathbb{Z}^n$ to a lattice point $\mathbf{x}$. In addition, how $\mathbf{u}_i, \mathbf{z}$ of method B correspond to integers $\mathbf{b}$ of method A with respect to a lattice point $\mathbf{x}$ is explicitly shown, to establish the equivalence of method A and method B.

For clarity, consider $a = 3$. The integer vector $\mathbf{b}$ is related to $\mathbf{u}_0, \mathbf{u}_1, \mathbf{u}_2$ and $\mathbf{z}$ as:

$$b_j = u_{0_j} + 2u_{1_j} + 4u_{2_j} + 8z_j, \qquad \text{for } 1 \leqslant j \leqslant k_0 \tag{4.33}$$
$$b_j = u_{1_j} + 2u_{2_j} + 4z_j, \qquad \text{for } k_0 < j \leqslant k_1 \tag{4.34}$$
$$b_j = u_{2_j} + 2z_j, \qquad \text{for } k_1 < j \leqslant k_2 \tag{4.35}$$
$$b_j = z_j, \qquad \text{for } k_2 < j \leqslant n \tag{4.36}$$

Let $\mathbf{u}'_i$ be the zero-padded version of $\mathbf{u}_i$ for $i = 0, 1, \ldots, a-1$, to have n components:

$$\mathbf{u}'_i = [u_{i_1}, u_{i_2}, \ldots, u_{i_{k_i}}, \underbrace{0, \ldots, 0}_{n-k_i}]^{\mathrm{t}}. \tag{4.37}$$

Given a diagonal matrix $\mathbf{D}$ with entries $d_{j,j} = 2^{-i}$ for $k_{i-1} < j \leqslant k_i$ where $k_{-1} = 0$, and 2^{-a} for the remaining diagonal entries. Then, the integer vector $\mathbf{b}$ is written as:

$$\mathbf{b} = \mathbf{D} \cdot (\mathbf{u}'_0 + 2\mathbf{u}'_1 + 4\mathbf{u}'_2 + 8\mathbf{z}), \tag{4.38}$$

where $\mathbf{D}$ is given Definition 4.3.

For Construction D', the lattice point $\mathbf{x}$ may be decomposed as:

$$\mathbf{x} = \sum_{i=0}^{a} 2^i \mathbf{x}_i, \tag{4.39}$$

with components $\mathbf{x}_i$ depending on $\mathbf{u}_i$ expressed below; $\mathbf{x}_i$ are not necessarily binary.

Now how information bits are related to a lattice point is described, to show that recovering integers from a lattice point is possible. Using (4.8) and (4.38)–(4.39) there are

$$\mathbf{H} \cdot \mathbf{x} = \mathbf{b} \tag{4.40}$$

$$\widetilde{\mathbf{H}} \cdot \mathbf{x} = \mathbf{D}^{-1} \cdot \mathbf{b} \tag{4.41}$$

$$\widetilde{\mathbf{H}} \cdot (\mathbf{x}_0 + 2\mathbf{x}_1 + \cdots + 2^a \mathbf{x}_a) = \mathbf{u}'_0 + 2\mathbf{u}'_1 + \cdots + 2^a \mathbf{z} \tag{4.42}$$

and the lattice components $\mathbf{x}_i \in \mathbb{Z}^n$ satisfy:

$$\widetilde{\mathbf{H}} \cdot \mathbf{x}_i = \mathbf{u}'_i, \qquad \text{for } i = 0, \ldots, a-1, \qquad \text{and} \tag{4.43}$$

$$\widetilde{\mathbf{H}} \cdot \mathbf{x}_a = \mathbf{z}. \tag{4.44}$$

Note that encoding performed using (4.43)–(4.44) is equivalent to encoding method A.

4.3 Decoding

Re-encoding using the generator matrix is typically needed for multistage decoding of Construction D lattices [63] (see also [10]). To produce hypercubical constellations with Construction D', multistage decoding may compute cosets instead of

re-encoding [12]. For Construction D', this section extends [63] and performs re-encoding using the check matrix, and describes a multistage successive cancellation decoding algorithm for Construction D' such that nonhypercubical constellations are allowed. In particular, this decoding algorithm is suitable for Construction D' coding lattices to be used with shaping lattices, likewise employing a binary decoder Dec_i of $\mathcal{C}_i$, but re-encoding is distinct because it corresponds to encoding method B. The encoding and decoding scheme is shown in Fig. 4.2, where encoding method B is to demonstrate the validity of the decoding algorithm.

4.3.1 Lattice Component and Re-encoding

Before proposing the decoding algorithm, there is a need to illustrate why multistage decoding is applicable to Construction D'. Assume a lattice point expressed in decomposition $\mathbf{x} = \mathbf{x}_0 + 2\mathbf{x}_1 + 2^2\mathbf{x}_2 + \cdots + 2^a\mathbf{x}_a$ was transmitted without noise. A sequence of operations can be performed recursively for $i = 0, \ldots, a-1$ to produce a binary codeword $\widetilde{\mathbf{x}}_i \in \mathcal{C}_i$ which will be used in Proposition 4.2. First $\widetilde{\mathbf{x}}_0$ is obtained by applying a modulo-2 operation to $\mathbf{x}$, such that the contribution of $2\mathbf{x}_1 + 2^2\mathbf{x}_2 \cdots + 2^a\mathbf{x}_a$ is removed. Assume that $\mathbf{x}_0$ can be found using $\widetilde{\mathbf{x}}_0$, and call this operation re-encoding denoted Re-enc_0. Then $\mathbf{x}_0$ is subtracted from $\mathbf{x}$ and divided by two, producing $\mathbf{x}_1 + 2\mathbf{x}_2 + \cdots + 2^{a-1}\mathbf{x}_a$ to which a modulo-2 operation is applied such that $\widetilde{\mathbf{x}}_1$ is obtained. This iterative procedure can be described as:

$$\widetilde{\mathbf{x}}_i = \mathbf{x}_i + 2\mathbf{x}_{i+1} + \cdots + 2^{a-i}\mathbf{x}_a \bmod 2, \tag{4.45}$$

$$\mathbf{x}_i = \text{Re-enc}_i(\widetilde{\mathbf{x}}_i), \tag{4.46}$$

$$\mathbf{x}_{i+1} + 2\mathbf{x}_{i+2} + \cdots + 2^{a-(i+1)}\mathbf{x}_a = \frac{(\mathbf{x}_i + 2\mathbf{x}_{i+1} + \cdots + 2^{a-i}\mathbf{x}_a) - \mathbf{x}_i}{2}, \tag{4.47}$$

for $i = 0, 1, \ldots, a-1$, and proceeds until $\mathbf{x}_a = (\mathbf{x} - \mathbf{x}_0 - 2\mathbf{x}_1 - \cdots - 2^{a-1}\mathbf{x}_{a-1})/2^a$ is obtained. The above description provides a foundation such that in each level, if a lattice component $\mathbf{x}_i$ can be found using re-encoding from the sequence reproduced by a separate binary decoder, then a lattice point can be recovered. The re-encoding is required when decoding Construction D' lattices. This will be performed with respect to encoding method B given in Subsection 4.2.2, because it shows how a lattice component is mapped from binary information bits, which can be obtained from a binary codeword produced by each level's decoder. More details will be given in Subsection 4.3.3.

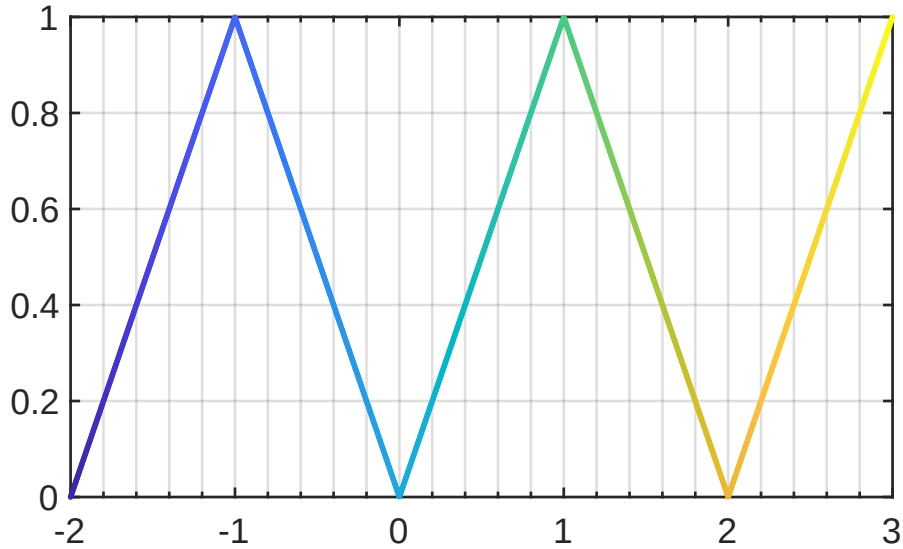


Figure 4.1: Mapping from a real number $y \in \mathbb{R}$ (horizontal axis) to a real number $y' \in [0, 1]$ (vertical axis) using the "triangle-function" $y' = \text{mod}^*(y)$.

4.3.2 Mod-2 AWGN Channel in Multistage Decoding

In practice the channel noise shall not be ignored. For multistage successive cancellation decoding, if the previous level $i-1$ produces the correct estimates, the decoder at level i outputs an estimate relying on an independent error probability, thus each level $i \in \{0, 1, \ldots, a-1\}$ can be seen as coding over an independent channel. For Construction D' lattices in zero-mean AWGN channel with noise variance σ^2, each level can be regarded as a mod-2 AWGN channel with input $\mathbf{x}_i$ and output $\mathbf{y}'_i$ given by:

$$\mathbf{y}'_i = (\mathbf{x}_i + \mathbf{w}_i) \bmod 2, \tag{4.48}$$

where the noise $\mathbf{w}_i$ satisfies a Gaussian distribution with zero mean and noise variance $\sigma^2/4^i$. This also leads the probability of error design rule to select a family of nested linear codes such that their corresponding contributions (the probability of error for decoding the code over the mod-2 channel) to the lattice probability of error are approximately equal. This design rule was used in [12].

It is evident from the findings in [10, 12, 63] that multistage decoding has benefits from binary decoders. Likewise, this dissertation proposes a decoding algorithm for Construction D' employing binary decoders. This is valid based upon Proposition 4.2 as will be addressed in the next subsection. There is a need to convert a received sequence to an appropriate input for the binary decoder. Let the received sequence of level i be denoted $\mathbf{y}_i \in \mathbb{R}^n$. Assume a binary decoder Dec_i

of $\mathcal{C}_i$ is used to find the binary codeword closest to $\mathbf{y}_i$, where Dec_i assumes a binary $0, 1$ codeword[2] was transmitted over an AWGN channel. Then a mapping from $\mathbf{y}_i \in \mathbb{R}^n$ to $\mathbf{y}'_i \in [0, 1]^n$ is required. Note that performing a modulo-2 operation is essential to remove the contributions from the following levels $i + 1, \ldots, a$, but modulo operation should correctly preserve distances to code symbols 0 and 1 as well. This can be accomplished by:

$$\mathrm{mod}^* (\mathbf{y}_i) = \Big| \mathrm{mod}_2 (\mathbf{y}_i + 1) - 1 \Big|, \tag{4.49}$$

called the "triangle-function", where mod_2 indicates a modulo-2 operation. This function is also suitable for Construction A[3] and Construction D [10].

The mapping using (4.49) is shown in Figure 4.1. An example is given accordingly to show that performing (4.49) can correctly preserve distances.

Example 4.3 Assume a channel message -0.05. The modulo-2 operation produces $-0.05 \bmod 2 = 1.95$ and the binary decoder recognizes that 1.95 is closer to 1 instead of 0. If the "triangle-function" in (4.49) is performed, then $|(-0.05 + 1 \bmod 2) - 1| = 0.05$ is the input to the decoder, and thus correctly produces 0.

4.3.3 Decoding Algorithm

Proposition 4.2. For Construction D', the lattice component $\mathbf{x}_i$ is congruent modulo 2 to a codeword $\widetilde{\mathbf{x}}_i \in \mathcal{C}_i$, for $i = 0, \ldots, a - 1$.

Proof The lattice component $\mathbf{x}_i$ satisfies $\widetilde{\mathbf{H}} \cdot \mathbf{x}_i = \mathbf{u}'_i$ and the codeword satisfies $\widetilde{\mathbf{H}}_i \odot \widetilde{\mathbf{x}}_i = \mathbf{0}$. Recall the last $n - k_i$ positions of $\mathbf{u}'_i$ are 0s. Row l of $\widetilde{\mathbf{H}}_i$ is equal to row $l + k_i$ of $\widetilde{\mathbf{H}}$, call this row $\mathbf{h}_l$. By definition, $\mathbf{h}_l \cdot \mathbf{x}_i = 0$ and $\mathbf{h}_l \odot \widetilde{\mathbf{x}}_i = 0$ for $l = 1, 2, \ldots, n - k_i$. Thus, $\mathbf{x}_i \bmod 2 = \widetilde{\mathbf{x}}_i$ and the proposition holds. □

Consider a lattice point $\mathbf{x}$ transmitted over a channel and the received sequence is

$$\mathbf{y}_0 = \mathbf{x} + \mathbf{w}, \tag{4.50}$$

[2]This is distinct from the more conventional BPSK signaling for binary codes, where the decoders assume $\{+1, -1\}$ were transmitted.

[3]The quantization of Construction A lattices as will be given in Section 6.3 is a variant of Algorithm 2.1, but is valid due to Lemma 2.1. When Construction A is applied, re-encoding is not needed, because a lattice point is decomposed by a binary codeword and a vector of integers.

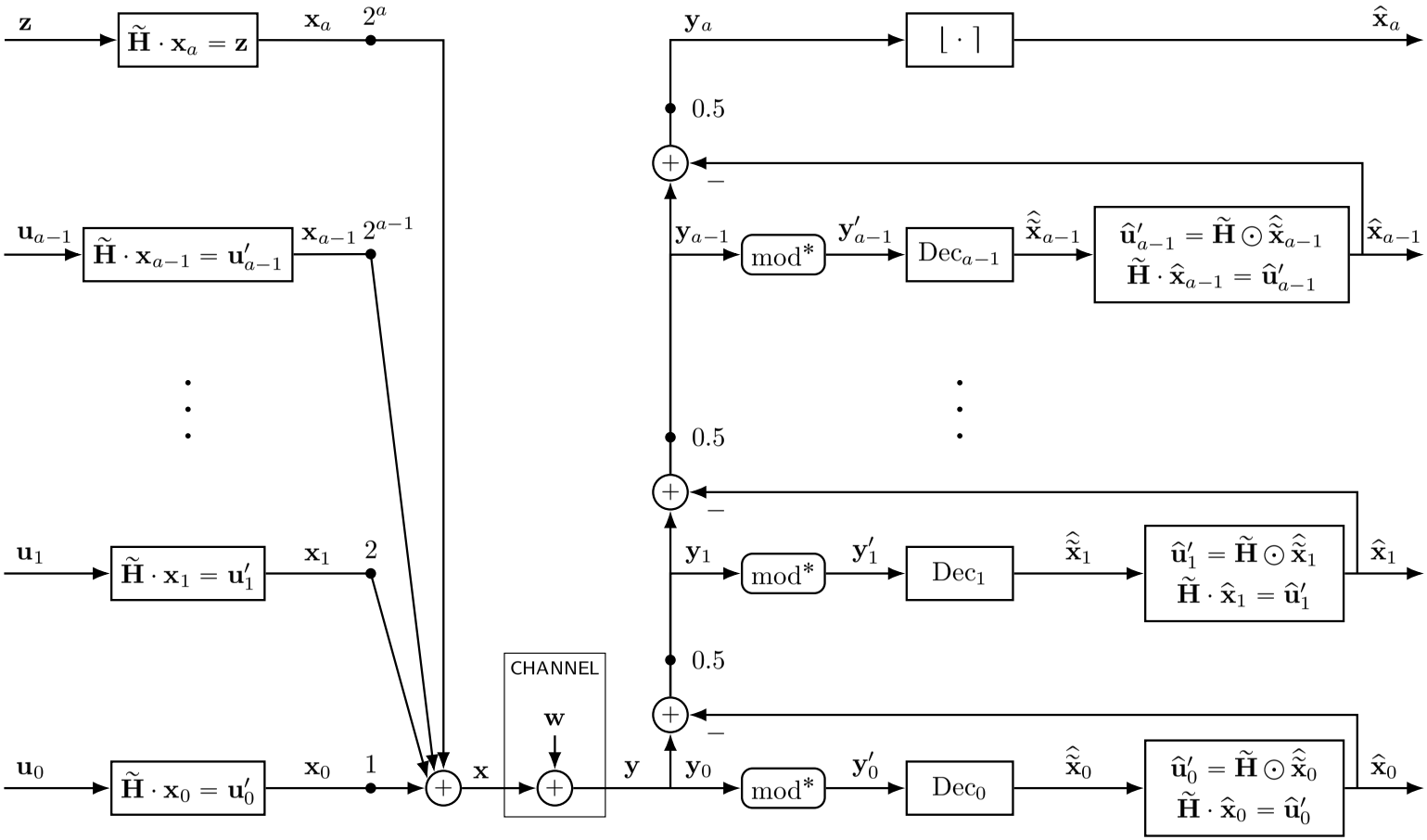


Figure 4.2: Block diagram of proposed encoding and decoding Construction D' lattices. mod* denotes the "triangle-function" $\mathrm{mod}^*(\mathbf{y}_i) = |\mathrm{mod}_2(\mathbf{y}_i + 1) - 1|$ where mod_2 indicates a modulo-2 operation.

Algorithm 4.1 Decoding Construction D' Lattices

Input: noisy input $\mathbf{y}$

Output: estimated lattice point $\widehat{\mathbf{x}}$

$\mathbf{y}_0 = \mathbf{y}$

$\mathbf{y}'_0 = |\mathrm{mod}_2\,(\mathbf{y}_0 + 1) - 1|$

$\widehat{\widetilde{\mathbf{x}}}_0 = \mathrm{Dec}_0(\mathbf{y}'_0)$

$\widehat{\mathbf{u}}'_0 = \widetilde{\mathbf{H}} \odot \widehat{\widetilde{\mathbf{x}}}_0$

solve $\widetilde{\mathbf{H}} \cdot \widehat{\mathbf{x}}_0 = \widehat{\mathbf{u}}'_0$

for $i = 1, 2, \ldots, a-1$ **do**

  $\mathbf{y}_i = (\mathbf{y}_{i-1} - \widehat{\mathbf{x}}_{i-1})/2$

  $\mathbf{y}'_i = |\mathrm{mod}_2\,(\mathbf{y}_i + 1) - 1|$

  $\widehat{\widetilde{\mathbf{x}}}_i = \mathrm{Dec}_i(\mathbf{y}'_i)$

  $\widehat{\mathbf{u}}'_i = \widetilde{\mathbf{H}} \odot \widehat{\widetilde{\mathbf{x}}}_i$

  solve $\widetilde{\mathbf{H}} \cdot \widehat{\mathbf{x}}_i = \widehat{\mathbf{u}}'_i$

end for

$\mathbf{y}_a = (\mathbf{y}_{a-1} - \widehat{\mathbf{x}}_{a-1})/2$

$\widehat{\mathbf{x}}_a = \lfloor \mathbf{y}_a \rceil$

$\widehat{\mathbf{x}} = \widehat{\mathbf{x}}_0 + 2\widehat{\mathbf{x}}_1 + \cdots + 2^{a-1}\widehat{\mathbf{x}}_{a-1} + 2^a\widehat{\mathbf{x}}_a$

where $\mathbf{w}$ is noise. Decoding proceeds recursively for $i = 0, 1, \ldots, a-1$. The decoding result at level $i-1$ is used before beginning decoding at level i. Each level receives $\mathbf{y}_i \in \mathbb{R}^n$ as input, which is mapped to a vector $\mathbf{y}'_i$ using (4.49) with each element $y'_j \in [0, 1]$ for $j = 1, 2, \ldots, n$. For binary decoders using log-likelihood ratio (LLR) as input, the bit LLR value

$$\mathrm{LLR} = \log \frac{\Pr(\widetilde{x}_j = 0 | y'_j)}{\Pr(\widetilde{x}_j = 1 | y'_j)} \tag{4.51}$$

may be estimated as:

$$\mathrm{LLR} = \frac{1 - 2y'_j}{2\sigma^2}. \tag{4.52}$$

The decoder Dec_i produces a binary codeword $\widehat{\widetilde{\mathbf{x}}}_i$ closest to $\mathbf{y}'_i$, which is an estimate of $\widetilde{\mathbf{x}}_i$, expressed by:

$$\widehat{\widetilde{\mathbf{x}}}_i = \mathrm{Dec}_i(\mathbf{y}'_i). \tag{4.53}$$

It is necessary to find $\widehat{\mathbf{x}}_i$. If $\widetilde{\mathbf{x}}_i$ does not contain an embedded $\widehat{\mathbf{u}}'_i$, first find

$$\widehat{\mathbf{u}}'_i = \widetilde{\mathbf{H}} \odot \widehat{\widetilde{\mathbf{x}}}_i. \tag{4.54}$$

Then re-encoding is performed to find $\widehat{\mathbf{x}}_i$, that is, (4.43). This estimated component $\widehat{\mathbf{x}}_i$ is subtracted from the input, and this is divided over reals by 2:

$$\mathbf{y}_{i+1} = (\mathbf{y}_i - \widehat{\mathbf{x}}_i)/2, \tag{4.55}$$

to form $\mathbf{y}_{i+1}$, which is passed as input to the next level. This process continues recursively, until $\mathbf{y}_a$ is obtained. The integers are estimated as

$$\widehat{\mathbf{x}}_a = \lfloor \mathbf{y}_a \rceil. \tag{4.56}$$

The estimated lattice point is written as

$$\widehat{\mathbf{x}} = \widehat{\mathbf{x}}_0 + 2\widehat{\mathbf{x}}_1 + \cdots + 2^a \widehat{\mathbf{x}}_a. \tag{4.57}$$

This successive cancellation decoding is described in Algorithm Algorithm 4.1.

Re-encoding (4.43) is necessary because it guarantees that an estimated lattice component is congruent modulo-2 to a codeword of the binary code at each level.

4.4 Alternative Encoding and Decoding

Furthermore, a Construction D' lattice point can also be generated without the need to use the zero-padded $\mathbf{u}_i$ (4.37), but is written as

$$\mathbf{x} = 2^a \mathbf{z} + \sum_{i=0}^{a-1} 2^i \mathbf{x}_i, \tag{4.58}$$

and the lattice components $\mathbf{x}_i$ should be in a systematic form:

$$\mathbf{x}_i = [u_{i_1}, u_{i_2}, \ldots, u_{i_{k_i}}, x_{i_{k_i+1}}, \ldots, x_{i_n}]^{\mathrm{t}}, \tag{4.59}$$

where $x_{i_{k_i+1}}, \ldots, x_{i_n}$ are found to satisfy $\widetilde{\mathbf{H}}_i \cdot \mathbf{x}_i = \mathbf{0}$. Note that $\mathbf{x}_i$ are not necessarily binary.

It can be shown that (4.58) is a lattice point. Write

$$\mathbf{H} \cdot \mathbf{x} = \mathbf{D} \cdot (\widetilde{\mathbf{H}} \cdot \mathbf{x}_0 + \cdots + 2^{a-1}\widetilde{\mathbf{H}} \cdot \mathbf{x}_{a-1} + 2^a \mathbf{z}). \tag{4.60}$$

Recognize that the vector $\widetilde{\mathbf{H}} \cdot \mathbf{x}_i$ is an integer in rows 1 to k_i and is 0 in rows $k_i + 1$ to n. The product $2^i \mathbf{D} \cdot \widetilde{\mathbf{H}} \cdot \mathbf{x}_i$ is also an integer vector. Thus, $\mathbf{H} \cdot \mathbf{x}$ is an integer vector. So the decomposition of $\mathbf{x}$ is a lattice point.

Therefore decoding Construction D' lattices can also be performed using (4.59) for re-encoding; this is distinct from Algorithm 4.1. The block diagram is given in Figure 4.3.

4.5 Shaping Construction D' Lattices

In this section, two examples will be given to show how shaping is performed as described in Section 3.2 for Construction D' lattice points using four distinct shaping lattices: the E_8, BW_{16}, Leech, and convolutional code lattices. The coding lattice Λ_{c} is described by a check matrix $\mathbf{H}_{\mathrm{c}}$ while the shaping lattice Λ_{s} is described by a generator matrix $\mathbf{G}_{\mathrm{s}}$.

When constructing a nested lattice code using an n-dimensional Construction D' lattice for coding and the E_8 lattice, the BW_{16} lattice, or the 24-dimensional Leech lattice for shaping, the dimension n must be a multiple of 8, 16 or 24, respectively, such that the direct sum of these low-dimensional lattices produces an n-dimensional shaping lattice. See Subsection 2.1.6.3. A convolutional code lattice can be more flexible on dimension as a shaping lattice. More discussions on their constructions will be addressed in Chapter 6. Also, the shaping lattice is often scaled to satisfy Lemma 3.1.

Example 4.4 Let the shaping lattice $\Lambda_{\mathrm{s}} = 4\Lambda_{\mathrm{A}}^{10}$ be described by a generator matrix $\mathbf{G}_{\mathrm{s}}$, which is the scaled-by-4 version of the matrix $\mathbf{G}_{\Lambda_{\mathrm{A}}^{10}}$ that will be given in Example 6.6 in Section 6.2. Let $\mathbf{H}_{\mathrm{c}}$ be the triangularized version of the check

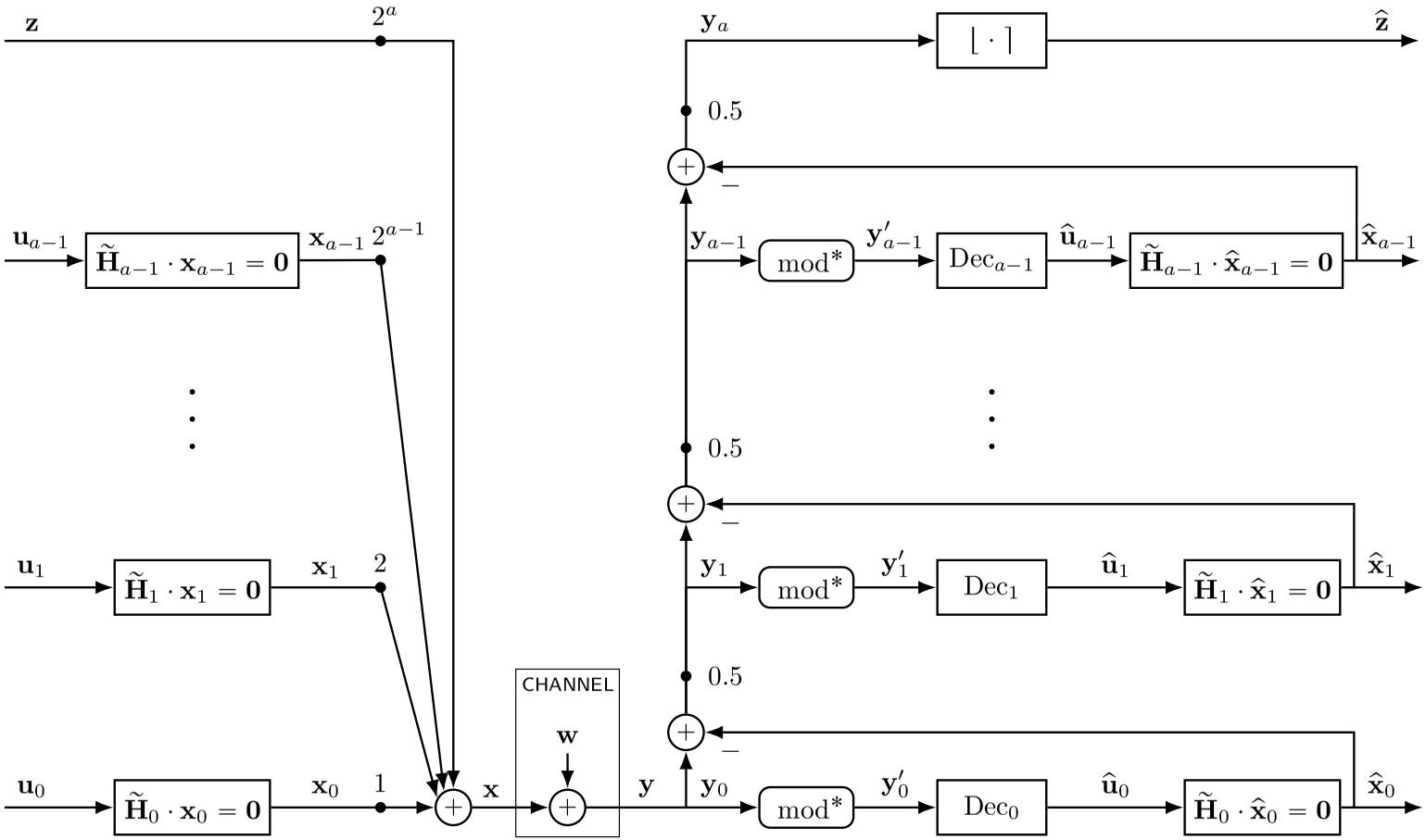


Figure 4.3: Block diagram of decoding Construction D' lattices where re-encoding corresponds to the alternative encoding method.

matrix $\mathbf{H}$ in equation (4.32) with $\mathbf{W} \cdot \mathbf{H} = \mathbf{H}_{\mathrm{c}}$ for a unimodular matrix $\mathbf{W}$:

$$\mathbf{W} = \begin{bmatrix} -1 & -1 & 1 & 0 & -2 & 2 & -2 & 0 & 0 & 4 \\ 4 & 4 & -3 & -4 & 10 & -6 & 8 & -4 & 4 & -16 \\ 2 & 1 & -1 & 0 & 2 & -2 & 2 & 0 & 0 & -4 \\ -3 & -2 & 2 & 2 & -6 & 4 & -5 & 2 & -2 & 10 \\ -1 & -1 & 1 & 2 & -3 & 1 & -2 & 2 & -2 & 4 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix}. \tag{4.61}$$

The lower-triangular matrix

$$\mathbf{H}_{\mathrm{c}} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & -1/2 & 0 & 1/2 & 0 & 0 & 0 & 0 & 0 & 0 \\ -1/2 & -1/2 & 0 & 1/2 & 1/2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1/2 & 1/2 & 0 & 1/2 & 0 & 0 & 0 & 0 \\ 1/2 & 0 & 0 & 1/2 & 0 & 1/2 & 1/2 & 0 & 0 & 0 \\ -1/4 & -1/4 & -1/4 & 1/4 & 1/4 & -1/4 & 1/4 & 1/4 & 0 & 0 \\ 1/4 & 0 & 1/4 & 1/4 & 0 & 1/4 & 1/4 & 0 & 1/4 & 0 \\ 1/4 & 1/4 & 1/4 & 0 & 0 & 1/4 & 0 & 0 & 0 & 1/4 \end{bmatrix} \tag{4.62}$$

is used when encoding and indexing. The diagonal elements M_i of $\mathbf{H}_{\mathrm{c}}\mathbf{G}_{\mathrm{s}}$ for $i = 1, 2, \ldots, 10$ are:

$$M_i \in \{4, 8, 4, 4, 2, 4, 4, 2, 2, 2\}, \tag{4.63}$$

which gives the range of information integers. See Section 3.2. Then the code rate (3.11) is $R = \frac{1}{10} \log_2 \prod_{i=1}^{10} M_i = 1.7$ bits per dimension. Assume the information vector[4] is: $\mathbf{b} = [2, 4, 1, 2, 0, 0, 2, 1, 0, 0]^{\mathrm{t}}$. By solving $\mathbf{H}_{\mathrm{c}}\mathbf{x} = \mathbf{b}$ using back-substitution a lattice point $\mathbf{x} = [2, 4, 1, 8, -2, -9, 3, -7, -5, 2]^{\mathrm{t}}$ is generated. The shaping operation (3.15) using $4\Lambda_{\mathrm{A}}^{10}$ gives a lattice codeword:

$$\mathbf{x}' = [-2, 0, -3, 0, -2, -1, -1, 1, -1, -2]^{\mathrm{t}}.$$

[4]The corresponding information bits are $\mathbf{u}_0 = [0, 0, 1]^{\mathrm{t}}$ and $\mathbf{u}_1 = [1, 0, 0, 0, 0, 0, 0]^{\mathrm{t}}$ for the underlying binary codes $\mathcal{C}_0$ and $\mathcal{C}_1$ of Λ_{c}, respectively. The remaining information bit positions in $\mathbf{b}$ may be selected using integers $\mathbf{z}$ similar to (4.33)–(4.36). Under correct decoding, these $\mathbf{u}_0, \mathbf{u}_1$ and $\mathbf{z}$ are produced by each level of the decoder. Note that the matrices used for encoding and the decoder's re-encoding should agree.

In this example, $4\Lambda_{\mathrm{A}}^{10}$ has a shaping gain of 0.58 dB which is obtainable because any lattice codeword $\mathbf{x}'$ lies in the zero-centered Voronoi region of $4\Lambda_{\mathrm{A}}^{10}$—this produces a nonhypercubical constellation.

Example 4.5 Let Λ_{c} be an $n = 48$-dimensional Construction D' lattice generated by $a = 3$-level nested binary codes. Let the check matrix $\mathbf{H}_{\mathrm{c}}$ of Λ_{c} be given in Table 4.1, which is lower-triangular and thus is convenient when encoding and indexing.

Since the dimension of the E_8 lattice, the BW_{16} lattice and the Leech lattice is $n' = 8, 16, 24$, it is needed to have $n/n' = 6, 3, 2$ copies of generator matrix (2.57), (2.62) and (2.64), respectively. For each distinct lattice, to satisfy Lemma 3.1, the resulting lattice obtained by the direct sum as given in Subsection 2.1.6.3 must be scaled by at least 16, $8\sqrt{2}$, and $16\sqrt{2}$, respectively. Using these least scale factors, the constructed shaping lattice has generator matrix $\mathbf{G}_{\mathrm{s}}^{E_8}$, $\mathbf{G}_{\mathrm{s}}^{BW_{16}}$, and $\mathbf{G}_{\mathrm{s}}^{\mathrm{Leech}}$, shown in Tables 4.2–4.4, respectively.

Consider a convolutional code lattice (CCL) for shaping Λ_{c}, where the underlying binary zero-tailed convolutional code has rate 1/3 and block length 48. The generator matrix $\mathbf{G}_{\mathrm{s}}^{\mathrm{CCL}}$ scaled by 8 is given in Table 4.5, which satisfies Lemma 3.1. The 48 diagonal elements of $\mathbf{H}_{\mathrm{c}}\mathbf{G}_{\mathrm{s}}^{\mathrm{CCL}}$ are:

{8,16,16,4,8,8,4,8,8,4,8,8,4,8,8,4,8,8,4,8,8,4,8,8,4,8,8,4,8,4,2,4,4,4,4,4,4,4,4,4,4,2,2,2,2,2,2,2},

which are used when performing the encoding and indexing operations.

Table 4.1: Check matrix of a 48-dimensional Construction D' lattice.

1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1/2	1/2	1/2	1/2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	1/2	1/2	1/2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	1/2	1/2	0	1/2	1/2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1/2	0	1/2	0	1/2	1/2	1/2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1/2	0	0	0	1/2	1/2	1/2	1/2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	1/2	0	1/2	1/2	0	1/2	1/2	1/2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	1/2	0	1/2	0	0	1/2	1/2	0	1/2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1/2	1/2	0	0	0	0	1/2	0	1/2	1/2	1/2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	1/2	0	0	0	1/2	1/2	1/2	0	0	1/2	1/2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	1/2	1/2	0	1/2	1/2	1/2	0	1/2	1/2	0	1/2	1/2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	1/2	0	1/2	1/2	0	0	0	1/2	0	0	1/2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	1/2	0	1/2	1/2	1/2	0	1/2	1/2	0	0	1/2	0	1/2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	1/2	0	1/2	0	0	1/2	1/2	0	1/2	0	0	1/2	1/2	0	1/2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	1/2	1/2	1/2	0	0	0	1/2	1/2	0	0	1/2	0	1/2	1/2	0	1/2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	1/2	1/2	1/2	0	0	0	0	0	1/2	1/2	1/2	1/2	1/2	0	0	1/2	1/2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1/2	0	0	0	0	1/2	1/2	1/2	1/2	1/2	0	1/2	0	0	0	0	0	1/2	1/2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	1/2	0	0	1/2	0	0	1/2	1/2	1/2	1/2	0	0	0	1/2	1/2	1/2	1/2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	1/2	0	1/2	0	1/2	1/2	1/2	0	0	1/2	0	1/2	0	0	1/2	1/2	0	0	1/2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1/2	0	1/2	1/2	0	1/2	0	1/2	0	1/2	0	1/2	1/2	0	0	1/2	1/2	0	1/2	0	0	1/2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1/2	0	0	0	0	0	1/2	1/2	1/2	0	0	0	1/2	0	0	1/2	0	0	1/2	1/2	0	1/2	1/2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	1/2	1/2	1/2	0	1/2	0	1/2	0	0	0	1/2	0	0	0	1/2	0	1/2	1/2	0	0	0	1/2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1/2	1/2	1/2	1/2	1/2	0	1/2	1/2	0	1/2	1/2	1/2	1/2	1/2	1/2	1/2	1/2	0	0	1/2	0	1/2	0	0	1/2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1/2	1/2	0	0	1/2	1/2	1/2	0	1/2	1/2	0	1/2	1/2	1/2	0	1/2	1/2	0	1/2	1/2	0	1/2	0	1/2	1/2	1/2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	1/2	1/2	0	1/2	0	0	0	0	1/2	1/2	1/2	0	0	0	0	1/2	1/2	0	1/2	0	0	1/2	1/2	1/2	1/2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1/2	1/2	0	0	0	1/2	0	0	1/2	1/2	1/2	0	1/2	0	0	0	1/2	0	0	1/2	0	0	0	1/2	1/2	1/2	0	1/2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1/2	1/2	1/2	1/2	0	1/2	0	1/2	1/2	1/2	0	0	1/2	1/2	1/2	1/2	1/2	1/2	1/2	1/2	1/2	0	0	0	0	1/2	0	1/2	1/2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	1/4	1/4	1/4	0	1/4	0	1/4	0	1/4	0	1/4	1/4	0	1/4	0	0	1/4	0	0	1/4	0	1/4	1/4	0	1/4	0	1/4	1/4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	1/4	1/4	1/4	0	1/4	0	0	1/4	0	0	1/4	0	1/4	0	0	0	1/4	1/4	0	1/4	1/4	1/4	1/4	1/4	1/4	1/4	0	1/4	1/4	1/4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	1/4	1/4	1/4	1/4	1/4	1/4	0	1/4	0	0	1/4	0	0	0	0	1/4	1/4	1/4	1/4	0	1/4	1/4	0	1/4	0	0	0	0	0	1/4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	1/4	1/4	1/4	1/4	1/4	0	1/4	0	0	1/4	0	0	0	0	0	0	0	0	0	0	1/4	0	1/4	1/4	1/4	1/4	1/4	1/4	0	1/4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1/4	1/4	0	1/4	1/4	0	0	0	0	1/4	0	1/4	0	1/4	0	0	1/4	1/4	1/4	1/4	0	1/4	1/4	0	0	0	0	1/4	0	0	0	1/4	1/4	1/4	0	0	0	0	0	0	0	0	0	0	0	0	0
0	1/4	1/4	0	1/4	1/4	0	1/4	0	1/4	1/4	1/4	1/4	0	1/4	0	0	1/4	1/4	1/4	0	0	1/4	0	0	0	0	0	1/4	0	1/4	1/4	0	1/4	1/4	0	0	0	0	0	0	0	0	0	0	0	0
1/4	1/4	0	0	0	0	0	0	0	0	0	1/4	0	1/4	0	0	1/4	1/4	1/4	0	0	0	1/4	0	0	1/4	0	1/4	0	0	1/4	1/4	1/4	1/4	0	1/4	0	0	0	0	0	0	0	0	0	0	0
0	0	1/4	0	0	1/4	1/4	1/4	0	0	1/4	1/4	1/4	0	0	1/4	1/4	0	0	0	0	1/4	1/4	1/4	0	0	1/4	0	0	0	0	0	1/4	0	1/4	1/4	1/4	0	0	0	0	0	0	0	0	0	0
1/4	1/4	0	0	1/4	0	1/4	1/4	1/4	1/4	1/4	0	0	1/4	0	1/4	0	1/4	0	1/4	0	1/4	0	1/4	1/4	0	1/4	1/4	0	0	1/4	0	1/4	1/4	0	0	1/4	1/4	0	0	0	0	0	0	0	0	0
0	1/4	0	0	0	1/4	0	1/4	1/4	0	0	0	0	0	0	0	0	1/4	0	1/4	0	0	1/4	1/4	0	0	1/4	0	0	1/4	1/4	0	1/4	1/4	1/4	0	0	1/4	0	0	0	0	0	0	0	0	0
1/4	0	0	0	1/4	1/4	1/4	1/4	0	1/4	1/4	1/4	0	0	0	1/4	1/4	0	1/4	1/4	1/4	1/4	1/4	0	0	1/4	0	0	1/4	0	1/4	0	1/4	0	0	0	0	1/4	0	1/4	0	0	0	0	0	0	0
1/4	0	0	0	1/4	1/4	1/4	1/4	0	1/4	0	1/4	1/4	0	1/4	0	0	0	0	0	1/4	1/4	1/4	0	0	0	0	1/4	0	0	1/4	0	0	0	1/4	1/4	1/4	0	1/4	0	1/4	0	0	0	0	0	0
0	1/8	0	0	0	0	1/8	1/8	1/8	0	1/8	0	1/8	1/8	0	0	0	0	1/8	1/8	1/8	0	0	0	0	1/8	1/8	1/8	1/8	1/8	0	1/8	0	1/8	0	0	1/8	1/8	0	0	1/8	0	0	0	0	0	0
1/8	1/8	1/8	1/8	0	0	0	1/8	0	0	0	1/8	0	0	1/8	0	0	0	0	0	0	0	0	0	1/8	1/8	1/8	0	1/8	0	1/8	0	0	0	0	0	0	0	1/8	1/8	0	0	1/8	0	0	0	0
1/8	0	0	0	0	1/8	1/8	1/8	0	1/8	0	1/8	0	1/8	1/8	0	1/8	0	0	0	1/8	0	0	0	1/8	1/8	1/8	0	1/8	0	1/8	1/8	1/8	0	0	0	1/8	0	0	1/8	1/8	1/8	0	1/8	0	0	0
1/8	0	0	1/8	0	1/8	0	1/8	1/8	1/8	1/8	0	1/8	1/8	1/8	1/8	1/8	0	0	0	0	1/8	0	0	1/8	1/8	1/8	1/8	1/8	1/8	0	1/8	1/8	1/8	1/8	0	0	0	1/8	1/8	1/8	0	1/8	1/8	1/8	0	0	0
0	1/8	1/8	0	1/8	1/8	1/8	1/8	0	0	1/8	1/8	1/8	1/8	1/8	0	0	1/8	1/8	1/8	0	0	0	1/8	0	1/8	1/8	1/8	0	1/8	1/8	1/8	0	0	1/8	1/8	0	1/8	1/8	1/8	0	1/8	0	1/8	1/8	1/8	0	0
0	0	0	1/8	0	1/8	0	0	0	0	0	1/8	0	1/8	1/8	0	1/8	0	0	0	1/8	0	0	0	1/8	0	1/8	1/8	1/8	0	1/8	1/8	1/8	0	0	0	0	1/8	0	0	1/8	1/8	1/8	0	1/8	1/8	1/8	0
0	0	0	0	1/8	0	0	0	1/8	1/8	1/8	0	0	0	1/8	1/8	0	0	1/8	1/8	0	1/8	1/8	0	1/8	0	1/8	0	1/8	0	1/8	1/8	1/8	0	0	1/8	1/8	0	1/8	1/8	1/8	0	1/8	1/8	1/8	0	1/8	1/8

Table 4.2: Generator matrix $\mathbf{G}_{\mathrm{s}}^{E_8}$ of a 48-dimensional lattice built from $16E_8$.

8	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
8	16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
8	-16	16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
8	0	-16	16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
8	0	0	-16	16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
8	0	0	0	-16	16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
8	0	0	0	0	-16	16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
8	0	0	0	0	0	-16	32	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	8	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	8	16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	8	-16	16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	8	0	-16	16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	8	0	0	-16	16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	8	0	0	0	-16	16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	8	0	0	0	0	-16	16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	8	0	0	0	0	0	-16	32	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	8	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	8	16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	8	-16	16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	8	0	-16	16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	8	0	0	-16	16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	8	0	0	0	-16	16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	8	0	0	0	0	-16	16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	8	0	0	0	0	0	-16	32	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	8	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	8	16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	8	-16	16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	8	0	-16	16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	8	0	0	-16	16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	8	0	0	0	-16	16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	8	0	0	0	0	-16	16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	8	0	0	0	0	0	-16	32	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	8	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	8	16	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	8	-16	16	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	8	0	-16	16	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	8	0	0	-16	16	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	8	0	0	0	-16	16	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	8	0	0	0	0	-16	16	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	8	0	0	0	0	0	-16	32	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	8	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	8	16	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	8	-16	16	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	8	0	-16	16	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	8	0	0	-16	16	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	8	0	0	0	-16	16
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	8	0	0	0	0	-16
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	8	0	0	0	0	0

Table 4.3: Generator matrix $\mathbf{G}_{\mathrm{s}}^{BW_{16}}$ of a 48-dimensional lattice built from $8\sqrt{2}BW_{16}$.

8	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
8	8	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
8	0	8	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
8	0	0	8	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
8	8	0	0	8	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
8	8	8	0	0	16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
8	0	8	8	0	0	16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
8	8	0	8	8	0	0	16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
8	0	8	0	8	0	0	0	16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
8	8	0	8	0	0	0	0	0	16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
8	8	8	0	8	0	0	0	0	0	16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
8	8	8	8	0	0	0	0	0	0	0	16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
8	8	8	8	8	0	0	0	0	0	0	0	16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
8	0	8	8	8	0	0	0	0	0	0	0	0	16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
8	0	0	8	8	0	0	0	0	0	0	0	0	0	16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
8	0	0	0	8	16	16	16	16	16	16	16	16	16	16	32	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	8	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	8	8	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	8	0	8	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	8	0	0	8	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	8	8	0	0	8	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	8	8	8	0	0	16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	8	0	8	8	0	0	16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	8	0	0	8	8	0	0	16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	8	8	8	8	0	0	0	0	16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	8	8	8	0	8	0	0	0	0	16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	8	8	8	8	0	0	0	0	0	0	16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	8	8	8	8	8	0	0	0	0	0	0	16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	8	0	0	8	8	0	0	0	0	0	0	0	16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	8	8	8	8	8	8	0	0	0	0	0	0	0	16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	8	0	8	0	8	0	0	0	0	0	0	0	0	0	16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	8	8	0	8	8	0	0	0	0	0	0	0	0	0	0	16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	8	0	8	0	8	0	0	0	0	0	0	0	0	0	0	0	16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	8	8	0	8	8	0	0	0	0	0	0	0	0	0	0	0	0	16	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	8	0	8	8	8	0	0	0	0	0	0	0	0	0	0	0	0	0	16	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	8	8	8	8	8	0	0	0	0	0	0	0	0	0	0	0	0	0	0	16	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	8	0	8	8	8	8	0	0	0	0	0	0	0	0	0	0	0	0	0	0	16	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	8	8	8	8	8	8	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	16	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	8	0	0	8	8	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	16	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	8	8	8	8	8	8	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	16	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	8	0	0	8	8	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	16	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	8	8	8	8	8	8	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	16	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	8	0	0	8	8	8	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	16	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	8	0	0	8	8	8	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	16	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	8	0	0	0	8	16	16	16	16	16	16	16	16	16	16	16	16	16	16	16	16	16	16	16	16	16	16	16	16	16	16	32

Table 4.4: Generator matrix $\mathbf{G}_{\mathrm{s}}^{\mathrm{Leech}}$ of a 48-dimensional lattice built from $16\sqrt{2}\Lambda_{24}$.

8	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
8	16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
8	0	16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
8	16	16	16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
8	0	0	0	16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
8	16	0	0	0	16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
8	0	16	0	0	0	16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
8	16	16	16	16	16	16	32	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
8	0	0	0	0	0	0	0	16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
8	16	0	0	0	0	0	0	0	16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
8	0	16	0	0	0	0	0	0	0	16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
8	16	16	16	0	0	0	0	16	16	16	32	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
8	0	0	0	16	0	0	0	16	0	0	0	16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
8	16	0	0	0	16	0	0	0	16	0	0	16	32	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
8	0	16	0	0	0	16	0	0	0	16	0	16	0	32	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
8	16	16	16	16	16	16	0	16	16	16	0	16	0	0	32	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
8	0	0	0	0	0	16	0	16	0	0	0	0	0	0	0	16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
8	0	0	0	16	0	0	0	0	16	0	0	0	0	0	0	16	32	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
8	0	0	0	0	16	0	0	0	0	16	0	0	0	0	0	16	0	32	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
8	0	0	16	16	16	16	0	16	16	16	0	0	0	0	0	16	0	0	32	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
8	0	0	16	0	16	0	0	16	0	0	0	16	0	0	0	16	0	0	0	32	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
8	0	0	16	0	0	16	0	0	16	0	0	16	0	0	0	16	0	0	0	0	32	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
8	0	0	16	16	0	0	0	0	0	16	0	16	0	0	0	16	0	0	0	0	0	32	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
-24	0	0	0	16	16	16	32	16	16	16	32	16	32	32	32	16	32	32	32	32	32	32	64	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	8	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	8	16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	8	0	16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	8	16	16	16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	8	0	0	0	16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	8	16	0	0	0	16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	8	0	16	0	0	0	16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	8	16	16	16	16	16	16	32	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	8	0	0	0	0	0	0	0	16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	8	16	0	0	0	0	0	0	0	16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	8	0	16	0	0	0	16	0	0	0	16	0	0	0	16	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	8	16	0	0	0	16	0	0	0	16	0	0	0	16	32	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	8	0	16	0	0	0	16	0	0	0	16	0	16	0	32	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	8	16	16	16	16	16	16	0	16	16	16	0	16	0	0	32	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	8	0	0	0	0	0	16	0	16	0	0	0	0	0	0	0	16	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	8	0	0	0	16	0	0	0	0	16	0	0	0	0	0	16	32	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	8	0	0	0	0	16	0	0	0	0	16	0	0	0	16	0	0	32	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	8	0	0	16	16	16	16	0	16	16	16	0	0	0	0	0	16	0	0	32	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	8	0	0	16	0	16	0	0	16	0	0	0	16	0	0	0	16	0	0	0	32	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	8	0	0	16	0	16	0	16	0	16	0	0	16	0	0	0	16	0	0	0	0	32	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	8	0	0	16	16	0	0	0	0	0	16	0	16	0	0	0	16	0	0	0	0	0	32	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	-24	0	0	0	16	16	16	32	16	16	16	32	16	32	32	32	16	32	32	32	32	32	32	64		

Table 4.5: Generator matrix $\mathbf{G}_{\mathrm{s}}^{\mathrm{CCL}}$ of a 48-dimensional convolutional code lattice.

8	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
8	16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
8	0	16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
8	0	0	8	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	8	16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	8	0	16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
8	0	0	8	0	0	8	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
8	0	0	0	0	0	8	16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	8	0	16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	8	0	0	8	0	0	8	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
8	0	0	8	0	0	0	0	0	8	16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	8	0	16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
8	0	0	0	0	0	8	0	0	8	0	0	8	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
8	0	0	8	0	0	8	0	0	0	0	0	8	16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	8	0	16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
8	0	0	8	0	0	0	0	0	8	0	0	8	0	0	8	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
8	0	0	8	0	0	8	0	0	8	0	0	0	0	0	8	16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
8	0	0	0	0	0	0	0	0	0	0	0	0	0	0	8	0	16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	8	0	0	8	0	0	0	0	0	8	0	0	8	0	0	8	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	8	0	0	8	0	0	8	0	0	8	0	0	0	0	0	8	16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	8	0	0	0	0	0	0	0	0	0	0	0	0	0	0	8	0	16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	8	0	0	8	0	0	0	0	0	8	0	0	8	0	0	8	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	8	0	0	8	0	0	8	0	0	8	0	0	0	0	0	8	16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	8	0	0	0	0	0	0	0	0	0	0	0	0	0	0	8	0	16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	8	0	0	8	0	0	0	0	0	8	0	0	8	0	0	8	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	8	0	0	8	0	0	8	0	0	8	0	0	0	0	0	8	16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	8	0	0	0	0	0	0	0	0	0	0	0	0	0	0	8	0	16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	8	0	0	8	0	0	0	0	0	8	0	0	8	0	0	8	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	8	0	0	8	0	0	8	0	0	8	0	0	0	0	0	8	16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	8	0	0	0	0	0	0	0	0	0	0	0	0	0	0	8	0	16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	8	0	0	8	0	0	0	0	0	8	0	0	8	0	0	8	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	8	0	0	8	0	0	8	0	0	8	0	0	0	0	0	8	16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	8	0	0	0	0	0	0	0	0	0	0	0	0	0	0	8	0	16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	8	0	0	8	0	0	0	0	0	8	0	0	8	0	0	16	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	8	0	0	8	0	0	8	0	0	8	0	0	0	0	0	0	16	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	8	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	16	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	8	0	0	8	0	0	0	0	0	8	0	0	0	0	0	16	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	8	0	0	8	0	0	8	0	0	8	0	0	0	0	0	0	16	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	8	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	16	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	8	0	0	8	0	0	0	0	0	0	0	0	0	0	0	16	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	8	0	0	8	0	0	8	0	0	0	0	0	0	0	0	0	16	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	8	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	16	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	8	0	0	8	0	0	0	0	0	0	0	0	0	0	0	16	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	8	0	0	8	0	0	0	0	0	0	0	0	0	0	0	0	16	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	8	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	16	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	8	0	0	0	0	0	0	0	0	0	0	0	0	0	0	16	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	8	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	16	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	8	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	16

4.6 Concluding Remarks

This chapter gave a definition of Construction D' using parity-check matrix which is equivalent to the classical definition of Construction D', and showed how to build a Construction D' lattice using nested binary linear codes. Two encoding methods for Construction D' lattices were proposed, where encoding method A is related to the well-known encoding algorithm commonly implemented for LDPC codes [61], and encoding method B showed explicitly the relation between information bits and integers which correspond to a lattice point. Decoding Construction D' includes a re-encoding step with respect to encoding method B that transforms a binary codeword into a lattice component. An alternative encoding method and its corresponding decoding algorithm were also provided. How to perform a shaping operation for a Construction D' lattice was given by an example. Moreover, the encoding and decoding methods addressed in this chapter are suitable to obtain shaping gains for power-constrained channels.

Chapter 5

Design of LDPC Codes

This chapter considers two-level Construction D' lattices, that are based on binary quasi-cyclic low-density parity-check (QC-LDPC) codes. Nested QC-LDPC codes are designed for forming high-dimensional Construction D' lattices.

One approach of lattice construction is to employ QC-LDPC codes and single parity-check product codes [35]. The first level code parity-check matrix consists of a top matrix that is modified from a QC-LDPC code used in a wireless standard [64, Table I] and bottom rows which contribute to parity checks for the product code. The second level code parity-check matrix is constructed using row operations on a submatrix for the previous level's matrix. For this design, it is not clear how to obtain a triangular matrix for a Construction D' lattice, however, it showed some benefit of using QC-LDPC codes. Thus this chapter addresses a design of Construction D' lattices using only QC-LDPC codes, where the second level code parity-check matrix $\mathbf{H}_1$ can be generated using row operations on a submatrix of the first level code parity-check matrix $\mathbf{H}_0$.

A *subcode condition* $\mathcal{C}_0 \subset \mathcal{C}_1$ must be satisfied to form a 2-level Construction D' lattice where $\mathcal{C}_0$ is the first level code and $\mathcal{C}_1$ denotes the second level code, and this is not straightforward. Branco da Silva and Silva also addressed the design of multilevel Construction D' lattices based on LDPC codes [12]. In their work, $\mathbf{H}_0$ was obtained from $\mathbf{H}_1$ by performing check splitting or progressive edge-growth (PEG)-based check splitting. In contrast to [12], this chapter designs the first level code parity-check matrix $\mathbf{H}_0$ such that the second level code parity-check matrix $\mathbf{H}_1$ may be constructed using row operations, where $\mathbf{H}_0$ and $\mathbf{H}_1$ can be easily triangularized and thus efficient encoding and indexing is possible. With this design, a straightforward method to find a triangular check matrix for Construction D' lattices is also given. QC-LDPC codes are designed using binary linear programming to guarantee that the necessary supercode can be constructed, as well as to satisfy the column and the row weight distribution.

5.1 Prototype Matrix of QC-LDPC Codes

A QC-LDPC code can be described using a prototype matrix, which is an M-by-N matrix with integer entries:

$$\begin{bmatrix} p_{1,1} & p_{1,2} & \cdots & p_{1,N} \\ p_{2,1} & p_{2,2} & \cdots & p_{2,N} \\ \vdots & \vdots & \ddots & \vdots \\ p_{M,1} & p_{M,2} & \cdots & p_{M,N} \end{bmatrix}, \tag{5.1}$$

where Z is an integer greater than 1, and $-1 \leqslant p_{i,j} < Z$ for $i = 1, 2, \ldots, M$ and $j = 1, 2, \ldots, N$. See [65].

The parity-check matrix $\mathbf{H}_0$ of a QC-LDPC code $\mathcal{C}_0$ can be expressed by

$$\mathbf{H}_0 = \begin{bmatrix} \mathbf{P}^{p_{1,1}} & \mathbf{P}^{p_{1,2}} & \ldots & \mathbf{P}^{p_{1,N}} \\ \mathbf{P}^{p_{2,1}} & \mathbf{P}^{p_{2,2}} & \ldots & \mathbf{P}^{p_{2,N}} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{P}^{p_{M,1}} & \mathbf{P}^{p_{M,2}} & \ldots & \mathbf{P}^{p_{M,N}} \end{bmatrix}, \tag{5.2}$$

where $\mathbf{P}^{p_{i,j}}$ is a Z-by-Z right-shift cyclic-permutation matrix[1] and the power $p_{i,j}$ is an element of (5.1), for $i = 1, 2, \ldots, M$ and $j = 1, 2, \ldots, N$. When $p_{i,j} = -1$, instead use a zero matrix and $\mathbf{P}^0$ is the identity matrix $\mathbf{I}_Z$. Thus Z is called circulant size and $\mathcal{C}_0$ has block length $n = ZN$.

Example 5.1 Let the circulant size be $Z = 6$ and write a 2-by-4 prototype matrix as:

$$\begin{bmatrix} 5 & 2 & 0 & -1 \\ 2 & 0 & 2 & 1 \end{bmatrix}. \tag{5.3}$$

Then a parity-check matrix of size 12-by-24 can be lifted from the prototype matrix

[1]Each matrix $\mathbf{P}^{p_{i,j}}$ is a submatrix of the parity-check matrix, which is referred to as a block or a circulant. A single circulant has row weight 1 and column weight 1. A double circulant has row weight 2 and column weight 2. A block row means Z rows in the parity-check matrix that are corresponding to a row in the prototype matrix.

in (5.3), given by:

$$
\left|\begin{array}{cccccc|cccccc|cccccc|cccccc}
0&0&0&0&0&1&0&0&1&0&0&0&1&0&0&0&0&0&0&0&0&0&0&0\\
1&0&0&0&0&0&0&0&0&1&0&0&0&1&0&0&0&0&0&0&0&0&0&0\\
0&1&0&0&0&0&0&0&0&0&1&0&0&0&1&0&0&0&0&0&0&0&0&0\\
0&0&1&0&0&0&0&0&0&0&0&1&0&0&0&1&0&0&0&0&0&0&0&0\\
0&0&0&1&0&0&1&0&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0\\
0&0&0&0&1&0&0&1&0&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0\\
\hline
0&0&1&0&0&0&1&0&0&0&0&0&0&0&1&0&0&0&0&1&0&0&0&0\\
0&0&0&1&0&0&0&1&0&0&0&0&0&0&0&1&0&0&0&0&1&0&0&0\\
0&0&0&0&1&0&0&0&1&0&0&0&0&0&0&0&1&0&0&0&0&1&0&0\\
0&0&0&0&0&1&0&0&0&1&0&0&0&0&0&0&0&1&0&0&0&0&1&0\\
1&0&0&0&0&0&0&0&0&0&1&0&1&0&0&0&0&0&0&0&0&0&0&1\\
0&1&0&0&0&0&0&0&0&0&0&1&0&1&0&0&0&0&1&0&0&0&0&0
\end{array}\right|
$$

5.2 Construction D' Lattices Formed by QC-LDPC Codes

This dissertation proposes 2-level Construction D' lattices based on nested binary QC-LDPC codes $\mathcal{C}_0 \subset \mathcal{C}_1$. The first level component code $\mathcal{C}_0$ has an M-by-N prototype matrix while the second level component code $\mathcal{C}_1$ has a 2-by-N prototype matrix. The code $\mathcal{C}_0$ has a design rate $1 - M/N$. And $\mathcal{C}_1$ is a high-rate code—a column weight 2, row weight N parity-check matrix is sufficient; column weight 2 was also used in [35]. The code $\mathcal{C}_0$ is a subcode of $\mathcal{C}_1$ thus the parity-check matrix of $\mathcal{C}_1$ is a matrix obtained from linear combinations of a $\mathcal{C}_0$ parity-check submatrix. Binary linear codes $\mathcal{C}_0$ and $\mathcal{C}_1$, and their parity-check matrices $\mathbf{H}_0$ and $\mathbf{H}_1$ are nested.

The parity-check matrix $\mathbf{H}_1$ does not provide good row and column distributions if the rows were taken from $\mathbf{H}_0$, thus it is needed to find $\mathbf{H}_1$ using linear combinations of rows in $\mathbf{H}_0$. Let the set $\mathcal{A}_q$ consist of row numbers $k \in \{1, \ldots, M\}$ where each element k corresponds to the k-th block row of $\mathbf{H}_0$, such that their block-wise sum[2] is a single block row of weight N and column weight 1, for $q = 1, 2$. In addition, the two sets $\mathcal{A}_1$ and $\mathcal{A}_2$ are disjoint.

[2]For example, the block-wise sum of the two parity-check matrices corresponding to prototype matrices $[0, 0, -1]$ and $[2, 0, 0]$ of circulant size 3 is $[0/2^*, -1, 0]$ where $*$ denotes a double circulant which corresponds to a parity-check submatrix written as $\left[\begin{smallmatrix} 1&0&1\\ 1&1&0\\ 0&1&1 \end{smallmatrix}\right]$.

The parity-check matrix $\mathbf{H}_1$ can be expressed as

$$\mathbf{H}_1 = \begin{bmatrix} \mathbf{H}'_1 \\ \mathbf{H}'_2 \end{bmatrix}, \tag{5.4}$$

where $\mathbf{H}'_1$ and $\mathbf{H}'_2$ are the sum of block rows $\mathcal{A}_1$ and $\mathcal{A}_2$, respectively:

$$\mathbf{H}'_q = \bigoplus_{k \in \mathcal{A}_q} \left[\mathbf{P}^{p_{k,1}} \ \mathbf{P}^{p_{k,2}} \ \cdots \ \mathbf{P}^{p_{k,N}} \right], \tag{5.5}$$

for $q = 1, 2$. Accordingly, $\mathbf{H}_1$ is a QC-LDPC parity-check matrix with column weight 2.

5.3 Binary Linear Programming for Prototype Matrix Construction

To form a 2-level Construction D' lattice using QC-LDPC codes, the two component binary codes are needed to satisfy the properties given in the previous section. A part of the design is to find the position of non-zero circulants.

To achieve this, this section[3] designs a matrix, given several constraints: the subcode condition, row and column weight degree constraints, and the matrix should be in the ALT form to enable efficient encoding. Binary linear programming can be used to satisfy these constraints to provide a binary matrix describing non-zero circulants' position when designing a prototype matrix [67].

Set up the programming problem by writing the M-by-N matrix as

$$\mathbf{A} = \begin{bmatrix} a_{1,1} & a_{1,2} & \cdots & a_{1,N} \\ \vdots & \vdots & \ddots & \vdots \\ a_{M,1} & a_{M,2} & \cdots & a_{M,N} \end{bmatrix}, \tag{5.6}$$

where $a_{i,j}$ is a binary variable for $i \in \{1, 2, \ldots, M\}$ and $j \in \{1, 2, \ldots, N\}$, and $a_{i,j} = 1$ indicates a non-zero circulant for (5.1).

The row weights of $\mathbf{A}$ are

$$\mathbf{r} = \{r_1, r_2, \cdots, r_M\}, \tag{5.7}$$

[3]Part of the work in Section 5.3 is the output of a collaborative research with co-authors in [66].

and the column weights are

$$\mathbf{c} = \{c_1, c_2, \cdots, c_N\}. \tag{5.8}$$

There are M row constraints: row i has weight r_i, expressed as

$$a_{i,1} + \cdots + a_{i,N} = r_i, \tag{5.9}$$

and N column constraints: column j has weight c_j, written as

$$a_{1,j} + \cdots + a_{M,j} = c_j. \tag{5.10}$$

For one of the subcode constraints, the rows from $\mathcal{A}_q$ sum to a block row (5.5) of weight N, thus a constraint for $q = 1, 2$ is added for (5.6):

$$\sum_{k \in \mathcal{A}_q} \sum_{j=1}^{N} a_{k,j} = N. \tag{5.11}$$

A constraint for the ALT form to force all-ones along the offset-by-one diagonal is expressed as:

$$\sum_{i=1}^{M-1} a_{i,N-M+1+i} = M - 1. \tag{5.12}$$

And another constraint to force all-zeros above the offset diagonal can be added in addition. The detail is omitted here.

The goal is to find $a_{i,j}$ that satisfies the above conditions, which can be expressed using the following binary linear program:

$$\min \sum_i \sum_j a_{i,j}, \tag{5.13}$$

subject to

$$\mathbf{K} \cdot \mathbf{a} = \begin{bmatrix} \mathbf{r} \\ \mathbf{c} \\ N \\ N \\ M-1 \\ 0 \end{bmatrix}, \tag{5.14}$$

where $\mathbf{K}$ is a constraint matrix that includes all the constraints described in (5.9)–(5.12) and $\mathbf{a}$ is the vectorized version of $\mathbf{A}$ (5.6). Because this is a binary linear programming problem, for the set $\mathcal{A}_q$, only one position will contain a 1 and the remaining $|\mathcal{A}_q| - 1$ positions will contain 0, in any column. The implementation of this optimization problem is easily solved using standard optimization packages.

5.4 Easily Triangularizable QC-LDPC Code Design for Construction D'

5.4.1 Design Requirements

Now a specific design of binary QC-LDPC codes $\mathcal{C}_0$ and $\mathcal{C}_1$ for 2-level Construction D' lattices is given. Let $\mathbf{H}_0$ and $\mathbf{H}_1$ be a parity-check matrix of the first level component code $\mathcal{C}_0$ and the second level component code $\mathcal{C}_1$, respectively. The parity-check matrices $\mathbf{H}_0$ and $\mathbf{H}_1$ are designed to satisfy the following properties:

1. $\mathcal{C}_0 \subset \mathcal{C}_1$;
2. $\mathbf{H}_0$ and $\mathbf{H}_1$ are of full rank;
3. $\mathbf{H}_0$ and $\mathbf{H}_1$ can be easily triangularized;
4. $\mathbf{H}_0$ and $\mathbf{H}_1$ have girth[4] as high as possible.

Property 1 allows $\mathcal{C}_0$ and $\mathcal{C}_1$ to form a Construction D' lattice Λ. It is convenient to generate a triangular check matrix for a Construction D' lattice if $\mathbf{H}_0$ and $\mathbf{H}_1$ have properties 2 and 3. Property 4 is designed subject to the decoding performance.

LDPC codes can be decoded using the belief propagation algorithm [68], which performs iterative message-passing on a Tanner graph [69]. The cycles in a Tanner graph strongly affect the decoding performance, thus eliminating short cycles is important. When the code has long block length the computational complexity to detect short cycles in a binary parity-check matrix is large. However, detecting cycles using a prototype matrix instead significantly reduces the running time. This dissertation follows the conditions given in [70, 71] when designing prototype matrices without short cycles.

The binary linear programming constraints (5.14) given in the previous section were described for a more general design, where the resulting QC-LDPC codes use offset-by-one diagonal and all-zeros above the diagonal. To satisfy the design requirements listed above, the last two constraints of $\mathbf{K}$ in (5.14) needs to be

[4]The girth is equal to the length of the shortest cycle that exist in the Tanner graph of the code.

changed for property 3. Firstly, (5.12) needs to be modified as

$$\sum_{i=1}^{M} a_{i,N-M+i} = M, \tag{5.15}$$

such that no-offset diagonal is used. Then, a non-zero entry is placed above the diagonal. Thus using the new constraint matrix denoted $\mathbf{K}'$, the goal can be expressed using (5.13) subject to

$$\mathbf{K}' \cdot \mathbf{a} = \begin{bmatrix} \mathbf{r} \\ \mathbf{c} \\ N \\ N \\ M \\ 1 \end{bmatrix}. \tag{5.16}$$

5.4.2 Resulting Design

To meet the design requirements, first use the binary linear programming to find a binary matrix $\mathbf{A}$ (5.6) with $M = 12$ rows and $N = 24$ columns, using degree distribution polynomials[5] of variable nodes and check nodes:

$$\lambda(x) = \frac{1}{3}x^2 + \frac{5}{12}x^3 + \frac{1}{8}x^4 + \frac{1}{8}x^6, \qquad \text{and} \tag{5.17}$$

$$\rho(x) = \frac{2}{3}x^6 + \frac{1}{3}x^7, \tag{5.18}$$

respectively, where $\lambda_d x^d$ and $\rho_d x^d$ means λ_d and ρ_d are the fraction of nodes with degree[6] d. This structure is a modified version of [64, Table I]. The prototype matrix of $\mathbf{H}_1$ is designed using two sets:

$$\mathcal{A}_1 = \{5, 7, 9, 11\}, \qquad \text{and} \tag{5.19}$$

$$\mathcal{A}_2 = \{6, 8, 10, 12\}, \tag{5.20}$$

as explained in Section 5.2, which has degree distribution polynomials

$$\lambda'(x) = x^2, \qquad \text{and} \tag{5.21}$$

$$\rho'(x) = x^{24}. \tag{5.22}$$

[5]Degree distribution can describe the row and column weight.

[6]In this subsection, x used in a degree distribution polynomial is distinct from the notation in other chapters.

Table 5.1: Prototype matrix of $\mathbf{H}_0$ with $Z = 96$ and $n = 2304$ where $*$ denotes a double circulant

-1	-1	53	-1	15	56	-1	-1	55	35	-1	8	0	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	26	-1	-1	51	-1	59	14	-1	16	-1	0	0	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
18	-1	3	-1	-1	82	42	-1	33	-1	-1	-1	-1	0	0	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	30	73	53	-1	49	-1	-1	8	-1	-1	-1	-1	-1	0	0	-1	-1	-1	-1	-1	-1	-1	-1
-1	67	-1	15	84	-1	-1	-1	-1	-1	-1	-1	-1	3	-1	82	0	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	71	83	34	-1	-1	-1	-1	-1	0	-1	-1	25	0	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	8	27	87	-1	-1	-1	0	-1	-1	-1	-1	59	0	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	91	-1	62	52	-1	-1	-1	0	-1	-1	6	0	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	11	5	17	-1	-1	0	-1	-1	-1	-1	12	0	-1	-1	-1
-1	-1	2	43	53	-1	-1	-1	-1	-1	-1	-1	-1	-1	73	-1	-1	-1	-1	-1	34	0	-1	-1
54	-1	26	-1	-1	12	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	9	0	0
52	91	-1	-1	-1	-1	-1	-1	-1	38	-1	-1	13	-1	-1	-1	-1	-1	-1	-1	-1	-1	66/71*	0

Table 5.2: Prototype matrix of $\mathbf{H}_1$ with $Z = 96$ and $n = 2304$ where $*$ denotes a double circulant

54	67	26	15	84	12	8	27	87	11	5	17	0	3	0	82	0	59	0	12	0	9	0	0
52	91	2	43	53	71	83	34	91	38	62	52	13	0	73	0	25	0	6	0	34	0	66/71*	0

Using a circulant size $Z = 96$, the prototype matrix of $\mathbf{H}_0$ can be generated by assigning:

$$\begin{cases} p_{i,j} = -1, & \text{for } a_{i,j} = 0 \\ -1 < p_{i,j} < Z, & \text{for } a_{i,j} = 1 \end{cases} \tag{5.23}$$

such that the lifted parity-check matrices $\mathbf{H}_0$ (5.2) and $\mathbf{H}_1$ (5.4) are free of four-cycles and six-cycles. The designed prototype matrices are shown in Tables 5.1–5.2.

Note that offset diagonal was not used and a double circulant $p^*_{12,23}$ was assigned such that $\mathbf{H}_0$ and $\mathbf{H}_1$ can be easily triangularized. This provides efficient encoding and indexing [16]. The power for adding double circulant was chosen without introducing short cycles. The designed QC-LDPC codes $\mathcal{C}_0$ and $\mathcal{C}_1$ are of block length $n = 2304$, with code rate 1/2 and 11/12, respectively. The design rates are chosen similar to [35]. Thus the resulting design of prototype matrix for $\mathbf{H}_0$ and $\mathbf{H}_1$ satisfies the following degree distributions[7]:

$$\lambda(x) = \frac{7}{24}x^2 + \frac{11}{24}x^3 + \frac{1}{8}x^4 + \frac{1}{8}x^6, \tag{5.24}$$

$$\rho(x) = \frac{7}{12}x^6 + \frac{5}{12}x^7, \qquad \text{and} \tag{5.25}$$

$$\lambda'(x) = \frac{23}{24}x^2 + \frac{1}{24}x^3, \tag{5.26}$$

$$\rho'(x) = \frac{1}{2}x^{24} + \frac{1}{2}x^{25}, \tag{5.27}$$

respectively. The code rates and degree distributions were relevant to previous work showing a good coding property, but can be further optimized by a density evolution algorithm [72].

The check matrix $\mathbf{H}$ of a Construction D' lattice can be constructed from $\mathbf{H}_0$ and $\mathbf{H}_1$. The QC-LDPC code prototype matrices for $n = 5016, 10008$ will be given in Appendix C. The corresponding parity-check matrices are of girth 8.

5.4.3 Triangular Matrix of Construction D' Lattices

A lower-triangular check matrix $\mathbf{H}$ for a 2-level Construction D' lattice is used for encoding. This can be constructed if the parity-check matrices $\mathbf{H}_0$ and $\mathbf{H}_1$ for nested binary codes $\mathcal{C}_0 \subset \mathcal{C}_1$ are triangularizable. Transform $\mathbf{H}_0$ and $\mathbf{H}_1$ into lower-triangular form by performing block row operations in the binary field, resulting in $\widetilde{\mathbf{H}}_0$ and $\widetilde{\mathbf{H}}_1$ respectively. The triangular matrix $\widetilde{\mathbf{H}}_0$ must contain the basis vectors

[7] This is changed from (5.17), (5.18), (5.21), and (5.22) due to the added double circulant.

of $\widetilde{\mathbf{H}}_1$ such that they both satisfy Definition 4.1. Then a lower-triangular check matrix[8] $\mathbf{H}$ is built using Definition 4.3 in Chapter 4.

The design of parity-check matrices $\mathbf{H}_0$ and $\mathbf{H}_1$ for QC-LDPC codes given in the previous subsection allows a straightforward method to generate the lower-triangular check matrix $\widetilde{\mathbf{H}}_0$. Let $p^{(1)}_{12,23} = \mathfrak{a}$ and $p^{(2)}_{12,23} = \mathfrak{b}$ be selected[9] such that

$$\mathbf{Q} = \mathbf{I}_Z + \mathbf{P}^{\mathfrak{a}} + \mathbf{P}^{\mathfrak{b}} \tag{5.28}$$

is a triple circulant and full rank. The lower-triangular $\widetilde{\mathbf{H}}_1$ can be obtained as follows. Let $\mathbf{V}$ be the block-wise sum of the two block rows of $\mathbf{H}_1$ over GF(2). The twenty-third block column of $\mathbf{V}$ is a square matrix $\mathbf{Q}$ (5.28). Using only row operations over GF(2), $\mathbf{Q}$ can be transformed to triangular form $\mathbf{T}$. Find a binary matrix $\mathbf{W}$ such that $\mathbf{W} \odot \mathbf{Q} = \mathbf{T}$. Replace the first block row of $\mathbf{H}_1$ by $\mathbf{W} \odot \mathbf{V}$ then the resulting matrix is lower-triangular and denoted $\widetilde{\mathbf{H}}_1$. After that, $\widetilde{\mathbf{H}}_0$ is built by replacing the last two block rows of $\mathbf{H}_0$ by $\widetilde{\mathbf{H}}_1$.

5.5 Concluding Remarks

This chapter gave a design method for binary QC-LDPC codes to form two-level Construction D' lattices. The position of non-zero circulants of the QC-LDPC code prototype matrix was found by binary linear programming. The powers of the prototype circulants were selected such that the resulting parity-check matrices are free of short cycles. As a matter of design, the parity-check matrix of the second level code was constructed by linear combinations of a first-level code submatrix. Moreover, using two nested binary QC-LDPC codes with the structure proposed in this chapter, a triangular check matrix of a QC-LDPC Construction D' lattice can be obtained, and the generating method is also addressed. The triangular structure contributes to nested lattice code indexing as shown in Chapter 3. QC-LDPC Construction D' lattices of various dimensions as well as their component binary QC-LDPC codes will be evaluated and the numerical results will be given in Chapter 7.

[8] Although $\mathbf{H}$ obtained in this way introduces double circulants that might result in short cycles, this $\mathbf{H}$ is only used for encoding and indexing as described in Chapter 3. When decoding a Construction D' lattice as addressed in Section 4.3, nontriangular matrices $\mathbf{H}_0$ and $\mathbf{H}_1$ are used by the binary decoders for $\mathcal{C}_0$ and $\mathcal{C}_1$, respectively.

[9] For example, the element 66/71* in Table 5.1 means a double circulant of size $Z = 96$, which has $\mathfrak{a} = 66$ and $\mathfrak{b} = 71$. Thus $\mathbf{Q}$ is a triple circulant and has full rank.

Chapter 6

Convolutional Code Lattices

This chapter focuses on the design of convolutional code lattices which are Construction A lattices using binary convolutional codes. The zero-centered Voronoi region of a convolutional code lattice can be used when constructing a nested lattice code, which is indeed a quantizer. The effectiveness of an n-dimensional lattice quantizer is measured by the shaping gain with respect to the normalized second moment of the integer lattice $\mathbb{Z}^n$. The shaping gain measures the power reduction, and the theoretic limit is 1.53 dB given by an n-sphere as $n \to \infty$ [13].

The shaping gains of convolutional code lattices were studied in [8, 9, 23, 24, 73]. Convolutional code lattices have high shaping gain, flexibility of lattice dimension, and low-complexity quantization using the well-known Viterbi algorithm. For these reasons, convolutional code lattices are suitable as shaping lattices. Both shaping gain and the complexity of quantization are of interest.

First, binary convolutional codes are reviewed. Then Section 6.2 gives a method to obtain triangular generator matrices for Construction A lattices that is modified from [6, 9, 36, 73]. This is applied to build convolutional code lattices from zero-tailed (i.e., conventional termination) convolutional codes, tail-biting convolutional codes, and truncated convolutional codes. Extensive numerical results of an exhaustive search finding the generator polynomials of the convolutional code which provides the best-found shaping gain are given. The tradeoff between shaping gain and quantization complexity of convolutional code lattices is also studied.

6.1 Convolutional Codes

Convolutional codes are a class of error-correcting codes introduced by Peter Elias in 1955 [74], and have been widely used for radio, digital video, mobile communications and satellite communications. They often form concatenations for constructing turbo codes [75]. Convolutional codes can be maximum-likelihood

decoded by the Viterbi algorithm [18], thus the decoder is optimal.

6.1.1 Description of Binary Convolutional Codes

A convolutional code encoder continuously produces code bit stream given an incoming information bit stream, thus the code rate R is the reciprocal of the number of code bits assigned to each data bit. This dissertation considers only rate $R = 1/2, 1/3, \ldots$ binary convolutional codes. Let $f = 1/R$. The encoder uses a shift register that contains m binary memory cells delaying the input bit, and m is called *memory order*; hence the encoder has 2^m states[1] and f filters with an impulse response $\mathbf{g}^{(i)}$ corresponding to a modulo-2 adder for $i = 0, 1, \ldots, f-1$. The feedforward encoder outputs f sequences $\mathbf{c}^{(i)}$ where each is the convolution under $GF(2)$ of the binary input $\mathbf{u}$ and the impulse response $\mathbf{g}^{(i)}$ in the time domain, expressed as:

$$\mathbf{c}^{(i)} = \mathbf{g}^{(i)} \circledast \mathbf{u}. \tag{6.1}$$

The codeword of a convolutional code is obtained by multiplexing the bits of $\mathbf{c}^{(0)}, \mathbf{c}^{(1)}, \ldots, \mathbf{c}^{(f-1)}$. Let D denote the delay operator. Let the information vector

$$\mathbf{u} = \begin{bmatrix} u_0 \\ u_1 \\ u_2 \\ \vdots \end{bmatrix}, \tag{6.2}$$

and output sequence

$$\mathbf{c}^{(i)} = \begin{bmatrix} c_0^{(i)} \\ c_1^{(i)} \\ c_2^{(i)} \\ \vdots \end{bmatrix}, \tag{6.3}$$

be the coefficients of polynomials

$$u(D) = u_0 D^{k-1} + u_1 D^{k-2} + \cdots + u_{k-2} D + u_{k-1}, \qquad \text{and} \tag{6.4}$$

$$c^{(i)}(D) = c_0^{(i)} D^{n-1} + c_1^{(i)} D^{n-2} + \cdots + c_{n-2}^{(i)} D + c_{n-1}^{(i)}, \tag{6.5}$$

[1]The state of an encoder is the content in memory cells. The total number of states is 2^m. For example, when $m = 2$ the possible states are $\{00, 01, 10, 11\}$.

respectively, where k is the length of information bits and n is the length of codeword, regarding a finite stream consideration. Then (6.1) can also be written in the transform domain as:

$$c^{(i)}(D) = g^{(i)}(D) \odot u(D), \tag{6.6}$$

where the polynomial in descending order[2]

$$g^{(i)}(D) = g_0^{(i)} D^m + g_1^{(i)} D^{m-1} + \cdots + g_{m-1}^{(i)} D + g_m^{(i)} \tag{6.7}$$

is called the *generator polynomial* and its coefficients form the binary vector

$$\mathbf{g}^{(i)} = \begin{bmatrix} g_0^{(i)} \\ g_1^{(i)} \\ \vdots \\ g_m^{(i)} \end{bmatrix} \tag{6.8}$$

in (6.1). For convenience, $\mathbf{g}^{(i)}$ can be represented by a number in the octal form and the upmost bit $g_0^{(i)}$ is the most significant bit. For example, $D^4 + D^2 + D$ corresponding to a binary sequence $\{1, 0, 1, 1, 0\}$ can be expressed by an octal number 26.

The encoding operation can also be expressed as:

$$\begin{bmatrix} c^{(0)}(D) \\ c^{(1)}(D) \\ \vdots \\ c^{(f-1)}(D)] \end{bmatrix} = \begin{bmatrix} g^{(0)}(D) \\ g^{(1)}(D) \\ \vdots \\ g^{(f-1)}(D) \end{bmatrix} \odot u(D), \tag{6.9}$$

$$\mathbf{c}(D) = \mathbf{G}(D) \odot u(D), \tag{6.10}$$

where the code polynomial generator matrix $\mathbf{G}(D)$ is equivalent to:

$$\mathbf{G}(D) = \mathbf{G}_0 D^m + \mathbf{G}_1 D^{m-1} + \cdots + \mathbf{G}_{m-1} D + \mathbf{G}_m, \tag{6.11}$$

where

$$\mathbf{G}_j = \begin{bmatrix} g_j^{(0)} \\ g_j^{(1)} \\ \vdots \\ g_j^{(f-1)} \end{bmatrix}, \tag{6.12}$$

[2]This convention is distinct from the representation in [76, Ch. 12], but follows the implementation in MATLAB Communications Toolbox [77, pp. 2-640–2-644].

for $j = 0, 1, \ldots, m$. Thus a convolutional code *generator matrix*[3] can also be written as:

$$\mathbf{G} = \begin{bmatrix} \mathbf{G}_0 & & & \\ \mathbf{G}_1 & \mathbf{G}_0 & & \\ \mathbf{G}_2 & \mathbf{G}_1 & \mathbf{G}_0 & \\ \vdots & \vdots & \vdots & \ddots \\ \mathbf{G}_m & \mathbf{G}_{m-1} & \mathbf{G}_{m-2} & \\ & \mathbf{G}_m & \mathbf{G}_{m-1} & \ddots \\ & & \mathbf{G}_m & \ddots \\ & & & \ddots \end{bmatrix}. \tag{6.13}$$

And a codeword can be simply obtained:

$$\mathbf{c} = \mathbf{G} \odot \mathbf{u}. \tag{6.14}$$

Example 6.1 Let a rate 1/2 convolutional code with memory order 3 be described with a polynomial generator matrix:

$$\mathbf{G}(D) = \begin{bmatrix} g^{(0)}(D) \\ g^{(1)}(D) \end{bmatrix} = \begin{bmatrix} D^3 + D^2 + D + 1 \\ D^3 + D + 1 \end{bmatrix}, \tag{6.15}$$

where $\mathbf{g}^{(0)} = [1\ 1\ 1\ 1]$ and $\mathbf{g}^{(1)} = [1\ 0\ 1\ 1]$ can be written in the octal form as $(17, 13)$. A generator matrix is:

$$\mathbf{G} = \begin{bmatrix} 1 & & & & \\ 1 & & & & \\ 1 & 1 & & & \\ 0 & 1 & & & \\ 1 & 1 & 1 & & \\ 1 & 0 & 1 & & \\ 1 & 1 & 1 & 1 & \\ 1 & 1 & 0 & 1 & \\ & 1 & 1 & 1 & \ddots \\ & 1 & 1 & 0 & \ddots \\ & & \vdots & \vdots & \ddots \end{bmatrix}. \tag{6.16}$$

The code can be realized using the nonsystematic feedforward encoder given in Figure 6.1. All possible states are listed in Table 6.1.

[3]For convenience, the generator bases are given in column convention. The empty entries are zeros.

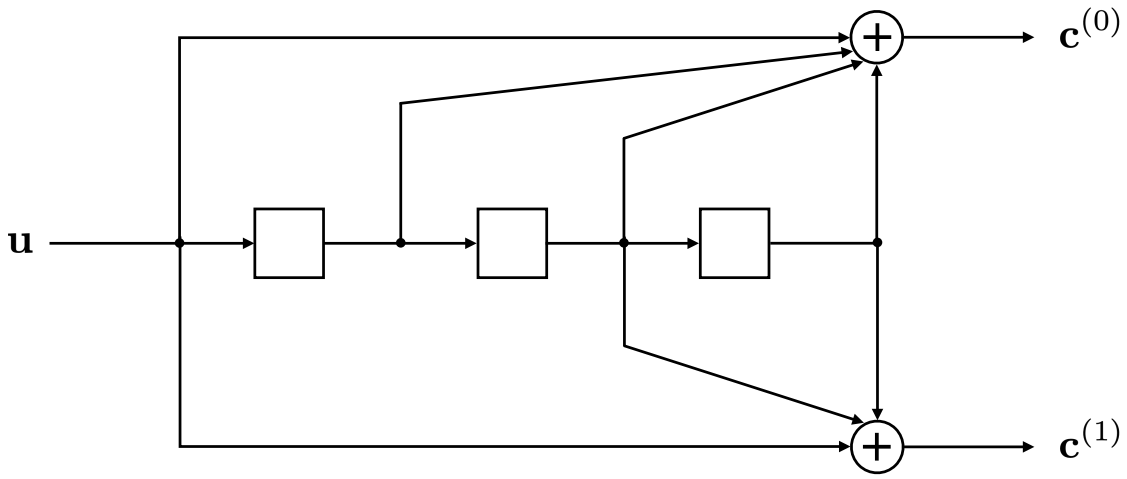


Figure 6.1: Rate 1/2 convolutional code nonsystematic feedforward encoder for code generator polynomials $(17, 13)$.

Table 6.1: All possible states of the encoder given in Figure 6.1.

S_0	0	0	0
S_1	0	0	1
S_2	0	1	0
S_3	0	1	1
S_4	1	0	0
S_5	1	0	1
S_6	1	1	0
S_7	1	1	1

6.1.2 Termination of Convolutional Codes

A convolutional code encoder produces output streams of infinite length when the information streams have infinite length. If the information stream is separated into sequences of finite length, a convolutional code can be regarded as a block code. Due to the existence of the memory delay the encoder needs to be terminated for each information sequence.

There are three termination methods: zero-tailed termination, tail-biting termination, and direct truncation. The corresponding convolutional codes are referred to as zero-tailed convolutional codes, tail-biting convolutional codes, and truncated convolutional codes, respectively.

6.1.2.1 Zero-Tailed Convolutional Codes

The conventional termination method is to input m zeros to the memory cells such that the encoder starts and ends in all-zero state, thus is called zero-tailed termination (or zero termination). This termination method reduces the code rate because the added m zeros representing no information. The decoding using the Viterbi algorithm is optimal and low-complexity, thus is widely implemented in communication systems.

Example 6.2 Let an information sequence of length $k = 3$ be $\mathbf{u} = [1\ 0\ 1]^{\mathrm{t}}$. The generator matrix (6.16) in Example 6.1 can be terminated as a matrix of $n = 2(k + m) = 12$ rows and $k = 3$ columns, written as:

$$\mathbf{G} = \begin{bmatrix} 1 & & \\ 1 & & \\ 1 & 1 & \\ 0 & 1 & \\ 1 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 0 \\ & 1 & 1 \\ & 1 & 1 \\ & & 1 \\ & & 1 \end{bmatrix}. \tag{6.17}$$

Then the codeword is obtained (6.14):

$$\mathbf{c} = \mathbf{G} \odot \mathbf{u} = [1\ 1\ 1\ 0\ 0\ 0\ 0\ 1\ 1\ 1\ 1\ 1]^{\mathrm{t}}, \tag{6.18}$$

where the last $2m = 6$ bits are due to the zero tail. Notice that the actual code rate now reduces to $k/n = 1/4$. Observe that when k is much larger than the memory order m, the rate loss is negligible. Also, due to a consideration of decoding complexity, using a small value for m is desirable for practical use. More discussions will be given in Section 6.5.

Using the binary polynomial representation (6.10), the encoding operation for information $u(D) = D^2 + 1$ can also be expressed as:

$$\mathbf{c}(D) = \begin{bmatrix} D^3 + D^2 + D + 1 \\ D^3 + D + 1 \end{bmatrix} \odot [D^2 + 1] \tag{6.19}$$

$$= \begin{bmatrix} D^5 + D^4 + D + 1 \\ D^5 + D^2 + D + 1 \end{bmatrix}, \tag{6.20}$$

corresponding to the multiplexed binary sequence (6.18).

6.1.2.2 Tail-Biting Convolutional Codes

If the memory cells were set as the m last bits of the information sequence before encoding, then the encoder starts and ends in the same state, thus is called tail-biting termination. This termination method does not cause rate loss, but the decoding algorithm requires higher complexity than that of the Viterbi algorithm, as will be addressed in Section 6.5.

Example 6.3 Let a sequence of information with length $k = 5$ be $\mathbf{u} = [1\ 0\ 1\ 1\ 0]^{\mathrm{t}}$. Before encoding, set the memory cells with the last $m = 3$ bits of $\mathbf{u}$: $\{1, 1, 0\}$—the encoder starts with the state S_3. Thus the generator matrix (6.16) in Example 6.1 can be terminated as a matrix of $n = 2k = 10$ rows and $k = 5$ columns, written as:

$$\mathbf{G} = \begin{bmatrix} 1 & & 1 & 1 & 1 \\ 1 & & 1 & 1 & 0 \\ 1 & 1 & & 1 & 1 \\ 0 & 1 & & 1 & 1 \\ 1 & 1 & 1 & & 1 \\ 1 & 0 & 1 & & 1 \\ 1 & 1 & 1 & 1 & \\ 1 & 1 & 0 & 1 & \\ & 1 & 1 & 1 & 1 \\ & 1 & 1 & 0 & 1 \end{bmatrix}. \tag{6.21}$$

Then the codeword is obtained (6.14):

$$\mathbf{c} = \mathbf{G} \odot \mathbf{u} = [1\ 1\ 0\ 1\ 0\ 0\ 1\ 0\ 0\ 1]^{\mathrm{t}}. \tag{6.22}$$

The codeword length is 10 and thus the code design rate 1/2 does not change.

6.1.2.3 Truncated Convolutional Codes

Truncated termination means direct truncatiion. The decoding complexity is about the same as zero-tailed termination. This method does not reduce the code rate.

Example 6.4 Consider the convolutional code in Example 6.1 terminated by direct truncation. Let a sequence of information with length $k = 3$ be $\mathbf{u} = [1\ 0\ 1]^{\mathrm{t}}$. Assume the encoder was in all-zero state S_0, that is, each memory cell contains a 0. Then the encoder produces a codeword: $[1\ 1\ 1\ 0\ 0\ 0]^{\mathrm{t}}$, which is the same as first 6 bits of (6.22). When the encoder starts with a different state, the corresponding codeword is also changed. For instance, if the initial state of the encoder is S_4, the encoder produces a codeword: $[0\ 1\ 0\ 1\ 1\ 1]^{\mathrm{t}}$.

6.2 Triangular Matrix of Construction A Lattices

Construction A applied to a binary code is Construction D reduced to one level. Triangular matrices provide efficient encoding and indexing, thus there is a need to obtain a triangular generator matrix $\mathbf{G}_{\Lambda_\mathrm{A}}$ for a Construction A lattice Λ_A. As addressed in Subsection 2.2.1, the well-known methods in [6, p. 183] and [36, pp. 32–33] require a systematic generator matrix for the code. The method given below does not require a systematic code generator matrix; while convolutional codes do have a systematic form it requires swapping bit positions (or coordinate permutation). Also, this section's method produces matrices already in the Hermite normal form as defined in [41, pp. 42–44] for forming Construction A generator matrices.

Let $\mathbf{G}' = [\mathbf{g}'_1, \mathbf{g}'_2, \ldots, \mathbf{g}'_k]$ be an n-by-k full-rank generator matrix with basis vectors in columns for a binary code $\mathcal{C}$. Perform column operations on $\mathbf{G}'$ to find $\mathbf{G} = [\mathbf{g}_1, \mathbf{g}_2, \ldots, \mathbf{g}_k]$ where $\mathbf{G}$ has the property that for each column $i = 1, \ldots, k$, there are only zeros to the right of the first one in column i. The canonical form for rate $1/2, 1/3, \ldots$ zero-tailed convolutional codes already satisfy this condition. Let $\mathbf{I}_n$ be an n-by-n identity matrix. The lower-triangular generator matrix $\mathbf{G}_{\Lambda_\mathrm{A}}$ of a Construction A lattice Λ_A can be obtained by replacing k columns in $2\mathbf{I}_n$ using the columns in $\mathbf{G}$. If $\mathbf{g}_i$ has its first one in position j, then replace column j of $2\mathbf{I}_n$ with $\mathbf{g}_i$, for all i. As a Construction A lattice, the determinant is $\det(\Lambda_\mathrm{A}) = |\det(\mathbf{G}_{\Lambda_\mathrm{A}})| = 2^{n-k}$.

Example 6.5 Consider a full-rank generator matrix $\mathbf{G}'$ of an arbitrary binary linear code. Replace the second column of $\mathbf{G}'$ by the sum of the first two columns to obtain $\mathbf{G}$:

$$\mathbf{G}' = \begin{bmatrix} 1 & 1 & 0 \\ 1 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix} \quad \Longrightarrow \quad \mathbf{G} = \begin{bmatrix} 1 & 0 & 0 \\ 1 & 0 & 0 \\ 1 & 1 & 0 \\ 0 & 1 & 0 \\ 1 & 1 & 0 \\ 1 & 0 & 1 \end{bmatrix}, \tag{6.23}$$

where $\mathbf{G}$ has the form described above. Apply Construction A to form a lattice

Λ_{A}^{6} by adding 3 columns to $\mathbf{G}$, then the generator matrix $\mathbf{G}_{\Lambda_{\mathrm{A}}^{6}}$ is given by:

$$\mathbf{G}_{\Lambda_{\mathrm{A}}^{6}} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 2 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 2 & 0 & 0 \\ 1 & 0 & 1 & 0 & 2 & 0 \\ 1 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}. \tag{6.24}$$

The code in the example is not a convolutional code, but was chosen to illustrate the construction of a lower-triangular generator matrix for Construction A.

Example 6.6 Consider a generator matrix $\mathbf{G}'$ of a nonsystematic feedforward zero-tailed convolutional code with generator polynomials represented as octal numbers $(7, 5)$, where the information sequence has length 3. Then apply Construction A to form a lattice $\Lambda_{\mathrm{A}}^{10}$ by replacing the 3 columns in $2\mathbf{I}_{10}$ using the columns in $\mathbf{G}'$, resulting in a lower-triangular generator matrix $\mathbf{G}_{\Lambda_{\mathrm{A}}^{10}}$. This is expressed as:

$$\mathbf{G}' = \begin{bmatrix} 1 & 0 & 0 \\ 1 & 0 & 0 \\ 1 & 1 & 0 \\ 0 & 1 & 0 \\ 1 & 1 & 1 \\ 1 & 0 & 1 \\ 0 & 1 & 1 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 1 \end{bmatrix} \implies \mathbf{G}_{\Lambda_{\mathrm{A}}^{10}} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 2 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 2 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 2 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 2 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 2 \end{bmatrix}. \tag{6.25}$$

6.3 Quantization of Construction A Lattices

Given an arbitrary vector $\mathbf{y}$, the quantization (2.16) is to find its nearest lattice point. The quantization of Construction A lattices given in Algorithm 2.1 can be simply described in Algorithm 6.1 where the mapping from the received sequence to the input of binary encoder is performed by equation (4.49). This is the case when Construction D [10] is reduced to one level. The block diagram of encoding and decoding Construction A lattices is shown in Figure 6.2. The effectiveness

Algorithm 6.1 Quantization of Construction A Lattices (Reduced From Quantization of Construction D Lattices [10])

Input: noisy input $\mathbf{y}$

Output: estimated lattice point $\widehat{\mathbf{x}}$

$\mathbf{y}' = |\mathrm{mod}_2\,(\mathbf{y}+1) - 1|$

$\widehat{\mathbf{c}} = \mathrm{Dec}(\mathbf{y}')$

$\widehat{\mathbf{z}} = \lfloor \frac{\mathbf{y}-\widehat{\mathbf{c}}}{2} \rceil$

$\widehat{\mathbf{x}} = \widehat{\mathbf{c}} + 2\widehat{\mathbf{z}}$

of quantization described by normalized second moment (NSM) can be estimated using Monte Carlo simulations, computed by (2.41). For convenience, shaping gain (2.42) with respect to NSM is used for observation.

Example 6.7 Assume a vector

$$\mathbf{y} = [-16.82, 2.06, 0.15, -41.27, -18.57, -48.61, -58.04, -1.87, -24.99, 26.37]^{\mathrm{t}}$$

is quantized using the convolutional code lattice in Example 6.6. An estimated lattice point $\widehat{\mathbf{x}} \in \Lambda_{\mathrm{A}}^{10}$ nearest to $\mathbf{y}$ can be found as follows. First perform the mapping[4] (4.49) from $\mathbf{y}$ to $\mathbf{y}' = [0.82, 0.06, 0.15, 0.73, 0.57, 0.61, 0.04, 0.13, 0.99, 0.37]^{\mathrm{t}}$. Using $\mathbf{y}'$, the convolutional decoder produces the output $\widehat{\mathbf{c}} = [1, 1, 0, 1, 0, 1, 0, 0, 1, 1]^{\mathrm{t}}$. Then a lattice point[5] can be obtained:

$$\widehat{\mathbf{x}} = [-17, 3, 0, -41, -18, -49, -58, -2, -25, 27]^{\mathrm{t}}. \tag{6.26}$$

6.4 Best-Found Convolutional Code Lattices

In this section, rate 1/2 and 1/3 binary convolutional codes of block length n, dimension k, and memory order m with non-systematic feed-forward encoders are used to build n-dimensional Construction A lattices Λ_{A}.

[4]Convolutional code decoder assumes a binary $0, 1$ codeword of length 10 was transmitted.

[5]The corresponding integer vector is $\widehat{\mathbf{z}} = [-9, 1, 0, -21, -9, -25, -29, -1, -13, 13]^{\mathrm{t}}$ which is possibly distinct from the integer vector $\mathbf{z}$ internally used in Algorithm 2.1 because the mapping methods used in the two quantization algorithms are different. But the estimated lattice point as well as the binary codeword produced by Algorithm 6.1 agree with those of Algorithm 2.1.

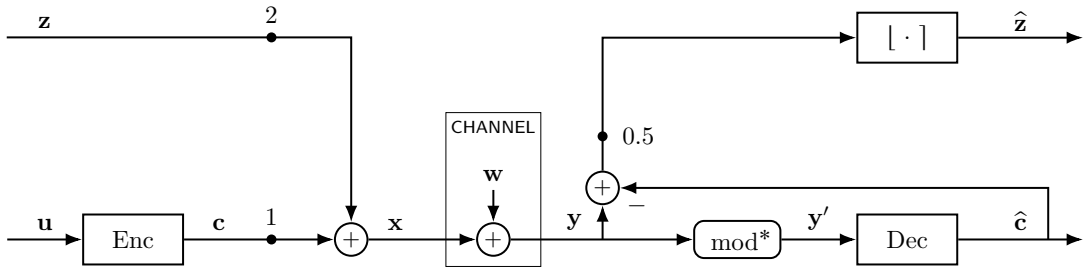


Figure 6.2: Block diagram of Construction A lattices corresponding to Algorithm 6.1.

The generator matrix of zero-tailed convolutional codes has the desired form described in Section 6.2, and thus is straightforward to find a lower-triangular generator matrix for convolutional code lattices. Let R_{ZTCC} be the code rate[6] of a zero-tailed convolutional code. The information length is $k = nR_{\mathrm{ZTCC}} - m$.

Tail-biting convolutional codes have excellent coding performance at short-to-medium block length, thus are suitable to form Construction A shaping lattices for low-to-moderate dimension. The information length is $k = nR_{\mathrm{TBCC}}$, where the code rate is R_{TBCC}.

Truncated convolutional codes are also considered. Let R_{DTCC} be the code rate. The information length is $k = nR_{\mathrm{DTCC}}$.

A convolutional code lattice may be scaled by $K = 2^2, 2^3, 2^4, \ldots$ to be used with a Construction D/D' coding lattice to form a nested lattice code, so as to satisfy Lemma 3.1.

Generator polynomials which give good coding properties for convolutional codes are well-known [76, Ch. 12]. See also [78, Ch. 4], [79, Table I–II]. However, it is not clear if these generator polynomials are the best choice for building shaping lattices. Thus, an exhaustive search of generator polynomials was performed for rate 1/2 and 1/3 nonsystematic feedforward binary convolutional codes. For each one, the shaping gain (2.42) of the resulting lattice was found by Monte Carlo integration [39, 40] using at least 10^7 samples.

[6] As explained in Subsection 6.1.2, the actual code rate of zero-tailed convolutional codes is lower than the design rate R_{ZTCC}, depending on the value of m. In this dissertation, the design rate R_{ZTCC} is used when referring to a zero-tailed code without losing the generality.

6.4.1 Exhaustive Search Procedure

The exhaustive search was performed using the combinations of polynomial (6.7). For rate 1/2 codes, the polynomial generator matrix is

$$\mathbf{G}(D) = \begin{bmatrix} g^{(0)}(D) \\ g^{(1)}(D) \end{bmatrix}, \tag{6.27}$$

where the polynomials are:

$$g^{(0)}(D) = g_0^{(0)} D^m + g_1^{(0)} D^{m-1} + \cdots + g_{m-1}^{(0)} D + g_m^{(0)} \tag{6.28}$$

$$g^{(1)}(D) = g_0^{(1)} D^m + g_1^{(1)} D^{m-1} + \cdots + g_{m-1}^{(1)} D + g_m^{(1)}. \tag{6.29}$$

For rate 1/3 codes, the polynomial generator matrix is

$$\mathbf{G}(D) = \begin{bmatrix} g^{(0)}(D) \\ g^{(1)}(D) \\ g^{(2)}(D) \end{bmatrix}, \tag{6.30}$$

which additionally includes:

$$g^{(2)}(D) = g_0^{(2)} D^m + g_1^{(2)} D^{m-1} + \cdots + g_{m-1}^{(2)} D + g_m^{(2)}. \tag{6.31}$$

Each coefficient is either 0 or 1 since only binary codes are considered.

The size of search space on convolutional code polynomial generator matrix can be reduced. Only non-systematic codes are considered, that is, $g^{(i)}(D) \neq 1$. The pairs with all zeros for the first or the last coefficient of polynomial are excluded. The pair $[g^{(0)}(D)\ g^{(1)}(D)]^{\mathrm{t}}$ is regarded as equivalent as the pair $[g^{(1)}(D)\ g^{(0)}(D)]^{\mathrm{t}}$, and thus only pairs with descending maximum degrees are included.

Given a convolutional code, by applying Construction A as shown in Subsection 2.2.1 and Section 6.2, a convolutional code lattice is generated and then evaluated using the quantization method given in Algorithm 2.1. For each lattice, the Monte Carlo integration simulated the shaping gain (2.42) using at least 10^7 samples. The simulated lattice dimension n (i.e., codeword length of the underlying convolutional code) are 18, 24, 30, 36, 72, 144, 288, 576, 1152, and 2304—these values are chosen such that both rate 1/2 and 1/3 codes with the same memory order m can produce convolutional code lattices with the same dimension, for a fair comparison. Convolutional codes using the three termination methods were evaluated in the exhaustive search. Rate 1/2 codes used memory order $m = 2, 3, 4, 5, 6, 7$ and rate 1/3 codes used $m = 2, 3, 4, 5$. And these values were chosen for two reasons. The complexity of decoding a convolutional code

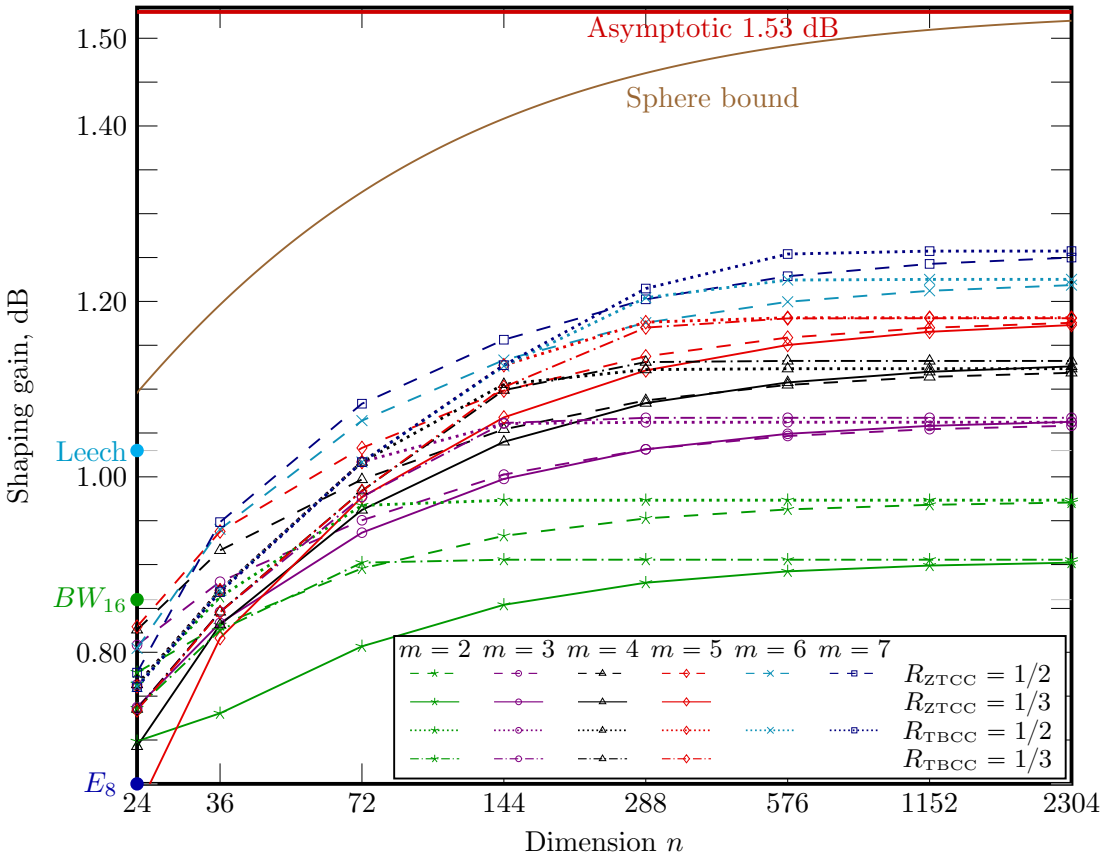


Figure 6.3: Best-found shaping gain of convolutional code lattices formed by zero-tailed convolutional codes (ZTCCs) and tail-biting convolutional codes (TBCCs) for rate 1/2 and 1/3 with various memory orders m. The 0.65 dB, 0.86 dB and 1.03 dB shaping gains of the E_8 lattice, the BW_{16} lattice and the Leech lattice are also shown for comparison.

exponentially increases as m increases. And the size of exhaustive search space for rate 1/3 codes is much larger than rate 1/2 codes when m is not small.

For the same code parameters except for generator polynomials, the corresponding convolutional code lattice with the highest shaping gain is recognized as the best-found lattice. The generator polynomials are given in the octal form.

Table 6.2: Recommended convolutional code generator polynomials (represented in octal numbers corresponding to the encoder implementation in a descending order) for a range of dimension n based on best-found convolutional code lattices for shaping, and asymptotic shaping gain γ_{s}

Convolutional code	m	$18 \leqslant n \leqslant 24$	$24 < n < 72$	$72 \leqslant n \leqslant 144$	$n > 144$	asymptotic γ_{s} (dB)	note
Rate 1/2, zero-tailed	2	$7,5$	$7,5$	$7,5$	$7,5$	0.9734	-
	3	$17,13$	$17,11$	$17,13$	$17,13$	1.0622	-
	4	$35,23$	$33,25$	$31,23$	$31,23$	1.1233	$\mathscr{C}_4$
	5	$67,51$	$77,55$	$75,57$	$75,57$	1.1814	$\mathscr{C}_5$
	6	$175,133$	$175,133$	$165,127$	$165,127$	1.2251	-
	7	$365,327$	$331,257$	$357,251$	$357,251$	1.2574	$\mathscr{C}_6$
Rate 1/3, zero-tailed	2	$7,7,5$	$7,7,5$	$7,6,5$	$7,6,5$	0.9055	$\mathscr{C}_2$
	3	$17,15,13$	$17,15,13$	$17,15,13$	$17,15,13$	1.0673	-
	4	$37,33,25$	$37,33,25$	$37,33,25$	$37,33,25$	1.1321	$\mathscr{C}_3$
	5	$71,65,57$	$71,65,57$	$73,57,41$	$73,57,41$	1.1808	$\mathscr{C}_1$
Rate 1/2, tail-biting	2	$7,6$	$7,5$	$7,5$	$7,5$	0.9734	-
	3	$16,3$	$15,6$	$17,13$	$17,13$	1.0622	-
	4	$30,7$	$30,13$	$36,15$	$31,23$	1.1233	-
	5	$70,3$	$60,13$	$74,13$	$75,57$	1.1814	-
	6	$140,7$	$140,13$	$130,17$	$165,127$	1.2251	-
	7	$340,3$	$320,3$	$320,17$	$357,251$	1.2574	-
Rate 1/3, tail-biting	2	$7,6,4$	$7,6,5$	$7,6,5$	$7,6,5$	0.9055	-
	3	$16,10,3$	$13,10,7$	$17,15,13$	$17,15,13$	1.0673	-
	4	$30,10,7$	$26,10,7$	$36,26,23$	$37,33,25$	1.1321	-
	5	$40,34,3$	$70,13,10$	$74,64,31$	$73,57,41$	1.1808	-

6.4.2 Exhaustive Search Result

For rate 1/2 convolutional codes, it is worthwhile to mention that the generator polynomials for zero-tailed codes this dissertation found[7] for asymptotic shaping gain match those provided in [23], except for $m = 5$, where this dissertation found $(75, 57)$ provides 0.01 dB higher asymptotic shaping gain than $(61, 57)$. The shaping gain of tail-biting codes with short block length were also studied in [24], which is higher than that of the Leech lattice.

The greatest shaping gain the exhaustive search found for various m and n is shown in Tables B.1–B.6 in Appendix B. Part of the results is drawn in Figure 6.3. In general, convolutional code lattices based on tail-biting convolutional codes have higher shaping gains than zero-tailed convolutional codes, for a given dimension. Truncated convolutional codes provide shaping gains lower than that of tail-biting codes, but are higher than that of zero-tailed codes. The generator polynomials of best-found convolutional codes vary depending on the dimension. At each searched dimension, there might be several codes providing comparable shaping gains and the precision depends on the number of Monte Carlo samples. For a range of dimensions, generator polynomials for a code with a shaping gain which is either the best-found shaping gain or within 0.01 dB to the best-found shaping gain is provided in Table 6.2, with exceptions as follows. An improvement for around 0.03–0.08 dB shaping gain can be obtained using generator polynomials $(77, 76, 73)$ at $n = 18$ and $(331, 257)$ at $n = 24$ for zero-tailed convolutional codes, and using generator polynomials $(31, 27)$, $(73, 25)$, $(144, 57)$, $(250, 67)$, $(37, 33, 25)$ and $(75, 45, 26)$ at $n = 144$ for tail-biting convolutional codes instead. The *asymptotic shaping gain* obtained at $n = 2^{20}$ and $n = 2^{20} + 2$ for rate 1/2 and 1/3 convolutional codes respectively is also provided. It is observed that at $n \leqslant 1152$ the shaping gain of convolutional code lattices using tail-biting convolutional codes can achieve the asymptotic shaping gain.

The maximal free distance d_{free} determines the goodness for coding of the convolutional code used with Viterbi decoding. But this may not necessarily be the case with shaping. Consider nonsystematic zero-tailed convolutional codes. The maximal d_{free} for optimal coding codes are $5, 6, 7, 8, 10, 10$ with memory order $m = 2, 3, 4, 5, 6, 7$ for rate 1/2 codes, and are $8, 10, 12, 13$ with $m = 2, 3, 4, 5$ for rate 1/3 codes [78, Ch. 4]. Regarding the best-found codes for shaping given in Table 6.2, their d_{free} may not be maximal. For instance, when $m = 2$ the code generated by $(7, 6, 5)$ has $d_{\text{free}} = 7 < 8$; when $m = 4$ the code generated

[7] This dissertation found these rate 1/2 code polynomials independently. The shaping gains shown in [23, 24] are slightly higher, but by no more than 0.0066 dB; this work has no particular explanation for this discrepancy.

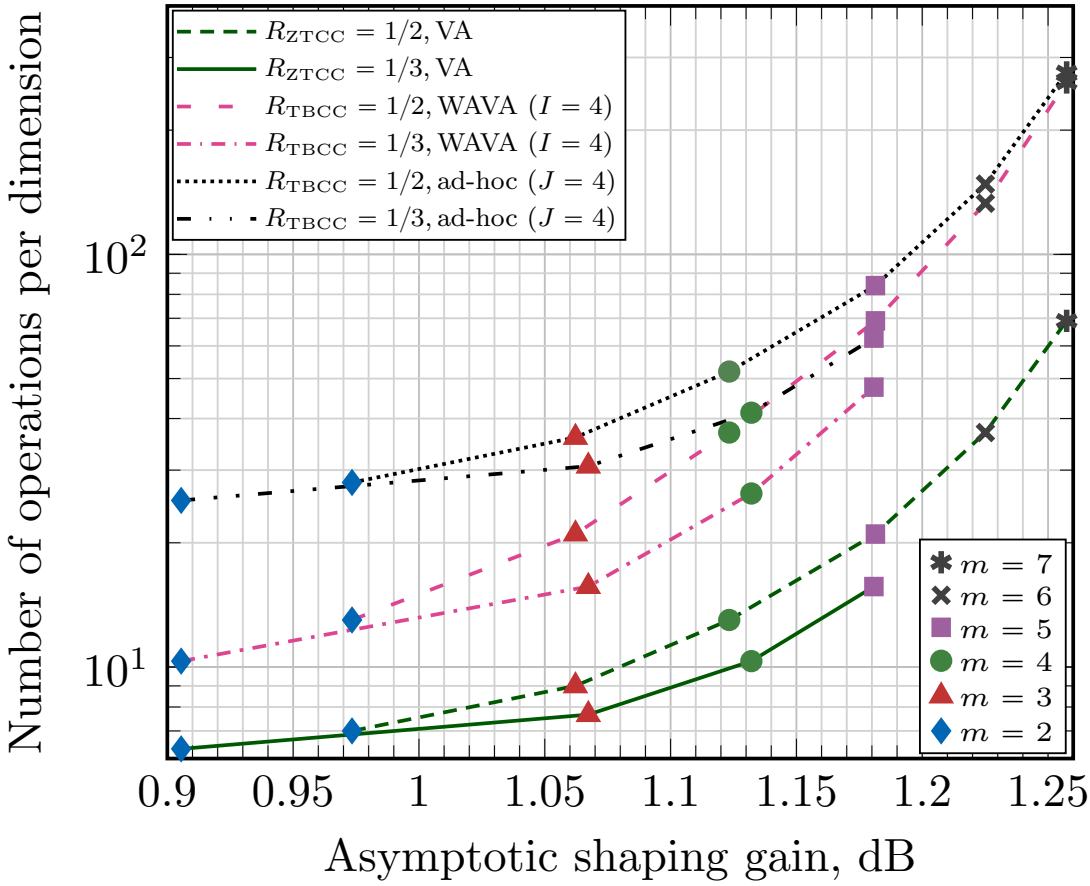


Figure 6.4: Performance-complexity tradeoff of convolutional code lattices formed by ZTCCs and TBCCs with various memory orders m where the decoding employs the Viterbi algorithm (VA) for ZTCCs, the wrap-around Viterbi algorithm (WAVA) ($I = 4$ iterations) and the ad-hoc method ($J = 4$ repeated times) for TBCCs.

by $(31, 23)$ has $d_{\text{free}} = 6 < 7$; when $m = 5$ the code generated by $(73, 57, 41)$ has $d_{\text{free}} = 12 < 13$; and when $m = 6$ the code generated by $(165, 127)$ has $d_{\text{free}} = 8 < 10$.

6.5 Complexity of Quantization

This section studies the tradeoff between shaping gain and quantization complexity for convolutional code lattices, when the Viterbi algorithm (VA) is used. Con-

struction A lattice quantization given in Algorithm 2.1 requires 5 operations per dimension [6, p. 450] to lift the binary codeword to a lattice point and the inverse. The Viterbi decoder uses 2^m comparisons at each trellis stage where the total number of trellis stages is nR_{ZTCC} for zero-tailed codes. It is assumed that n is much larger than m so that the contribution of termination and initialization to complexity can be ignored. Thus the normalized time complexity is $5 + 2^m R_{\text{ZTCC}}$. The decoding complexity of truncated codes is similar to that of zero-tailed codes, thus the detail is omitted.

The complexity of quantizing convolutional code lattices based on tail-biting convolutional codes was also analyzed, using the warp around Viterbi algorithm (WAVA) [80] with a maximum of I iterations and an ad-hoc suboptimal scheme [81] that decodes repeated-J-times sequence using the Viterbi algorithm with zero termination, requiring $5+2^m R_{\text{TBCC}} I$ and $(5+2^m R_{\text{TBCC}})J$ operations per dimension respectively. The results given in the previous section were obtained using $J \geqslant 16$ and $nJ \geqslant 1152$ for the ad-hoc decoding.

The normalized quantization complexity (or number of operations per dimension) is shown in Figure 6.4 as a function of asymptotic shaping gain. Rate 1/3 convolutional codes outperform rate 1/2 convolutional codes for $m = 3$ and $m = 4$ in terms of shaping gain and quantization complexity, and convolutional code lattices based on rate 1/2 convolutional codes have the best shaping gain for a fixed memory order $m = 2$ and $m = 5$. Decoding tail-biting convolutional codes requires higher complexity than that of zero-tailed convolutional codes and truncated convolutional codes. In summary, using rate 1/3 convolutional codes produces a more favorable performance-complexity trade-off.

6.6 Concluding Remarks

Convolutional code lattices provide many desirable properties for shaping such as high shaping gain and low-complexity quantization/decoding algorithm. Unlike using the direct sum of the E_8 lattice, the BW_{16} lattice and the Leech lattice for shaping, convolutional code lattices have more flexibility in lattice dimension for matching the coding lattice dimension. They were also considered for self-similar nested lattice codes in [82], but are only used as shaping lattices in this dissertation.

High shaping gains are appealing. However, there exists a tradeoff between the shaping gain and the quantization complexity. In general, tail-biting convolutional codes can provide higher shaping gains than that of zero-tailed convolutional codes and truncated convolutional codes, but require higher complexity when

decoding. Convolutional code lattices based on the three termination methods have comparable shaping gains increasing the lattice dimension n. The numerical results indicate how to select a shaping lattice. At dimension $\leqslant 24$, use the well-known lattices given in Section 2.3. At low-to-moderate dimension, use convolutional code lattices that are based on truncated codes. At high dimension, use convolutional code lattices that are based on zero-tailed convolutional codes.

Convolutional code lattices based on zero-tailed convolutional codes will be chosen for building nested lattice codes as will be shown in the next chapter, since the coding lattice will have high dimensions. For a fixed termination method, rate 1/3 convolutional codes provide a better performance-complexity tradeoff than rate 1/2 convolutional codes. When memory order m is small, tail-biting convolutional codes approach the asymptotic shaping gain even at medium dimension. As m is increased, the dimension to achieve the shaping gain is also increased, but is as small as 1152 for $m = 7$.

Chapter 7

Evaluation of QC-LDPC Construction D' Lattices for the Power-Constrained Channel

Three quasi-cyclic low-density parity-check (QC-LDPC) Construction D' lattices with dimensions $2304, 5016, 10008$ designed in Chapter 5 were simulated in both power-unconstrained and power-constrained additive white Gaussian noise (AWGN) channels using the Construction D' decoding algorithm proposed in Section 4.3. It is necessary to evaluate the error-rate performance using Monte Carlo simulations. The multistage successive cancellation decoding was performed by employing the belief propagation decoding algorithm for decoding LDPC codes, and reencoding follows encoding method B proposed in Subsection 4.2.2. Each two-level QC-LDPC Construction D' lattice was formed by two binary QC-LDPC codes $\mathcal{C}_0$ and $\mathcal{C}_1$. The prototype matrices of the component QC-LDPC codes with block length $2304, 5016, 10008$ used for simulations are shown in Tables 5.1–5.2, Tables C.1–C.2, and Tables C.3–C.4, respectively.

In the power-unconstrained AWGN channel, the volume-to-noise ratio (VNR) performance is of interest. The component QC-LDPC codes were also simulated in mod-2 AWGN channel, since each component binary linear code of a Construction D/D' lattice can be regarded as independent evaluation in a mod-2 AWGN channel, if the previous level was decoded correctly.

The main goal is to evaluate the Construction D' lattices in power-constrained AWGN channels. This was done by constructing a variety of nested lattice codes using QC-LDPC Construction D' lattices for coding, and using distinct lattices for shaping—they are the E_8 lattice, the BW_{16} lattice and the Leech lattice, as well as the best-found convolutional code lattices presented in Chapter 6. The nested lattice coding scheme and the encoding/indexing follows Chapter 3. Various codes were compared with hypercube shaping, using several code rates.

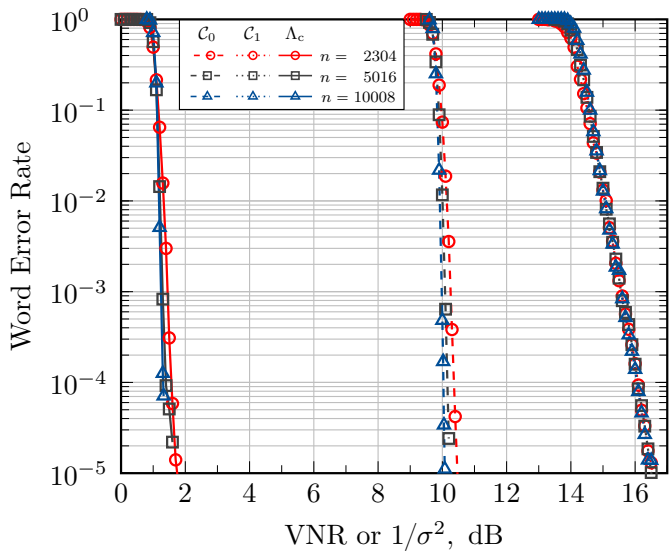


Figure 7.1: Simulation results over mod-2 AWGN channel: word error rate of QC-LDPC Construction D' lattices versus VNR and word error rate of the underlying component codes versus $1/\sigma^2$

A word error occurs when any element of a lattice codeword and its estimate disagree $\mathbf{x} \neq \hat{\mathbf{x}}$. Same definition applies to the binary codeword and its estimate in mod-2 AWGN channel. The word error rate (WER) is estimated as the number of word errors divided by the total number of simulated words (lattice codewords or binary codewords). The simulation results give the WER as a function of the volume-to-noise ratio (VNR), or E_b/N_0 for power-unconstrained AWGN channel and power-constrained AWGN channel, respectively.

7.1 Power-Unconstrained AWGN Channel

Simulations for QC-LDPC Construction D' lattices in a power-unconstrained AWGN channel were performed in the coding scheme given in Figure 4.2. Simulations for the proposed Construction D' lattices and underlying QC-LDPC codes C_0 and C_1 were performed in the mod-2 AWGN channel. The decoding using Algorithm 4.1 applies belief propagation decoding for LDPC codes with maximum 100 iterations. The WER is shown in Figure 7.1 as a function of VNR or signal-to-noise ratio SNR $= E_\mathrm{s}/\sigma^2 = 1/\sigma^2$ given in decibels. The codes C_0 and C_1 were designed with code rates similar to that of the codes in [35].

Observe that in Figure 7.2 as the dimension n increases, the proposed QC-

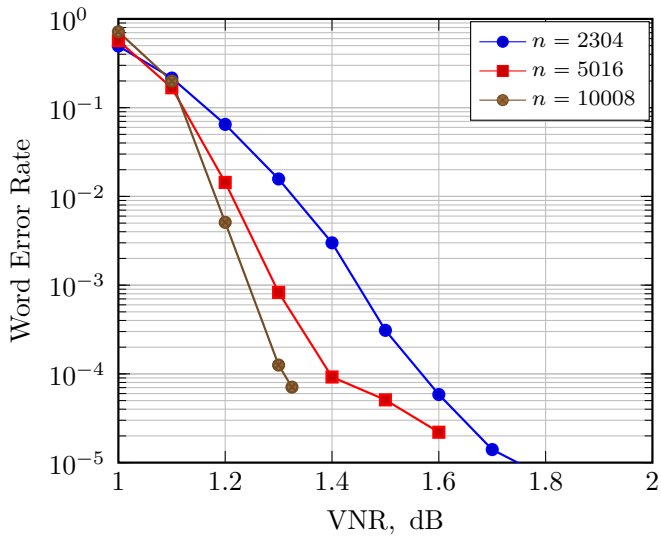


Figure 7.2: VNR performance of proposed QC-LDPC Construction D' lattices in various dimensions.

LDPC Construction D' lattice[1] has a better VNR performance at some WER. But the 5016-dimensional lattice has an error floor at WER around 10^{-4} while the 2304-dimensional lattice does not.

7.2 Power-Constrained AWGN Channel

Construction D' lattices Λ_c of dimension $n = 2304, 5016, 10008$ formed by QC-LDPC codes were evaluated in the power-constrained AWGN channel. At the decoder, the re-encoding implicitly assumes that method B of Subsection 4.2.2 is being used, which is equivalent to method A of Subsection 4.2.1, even for triangular Construction D' matrices of Chapter 5. The belief propagation (BP) decoder of LDPC codes ran maximum 50 iterations. The well-known low-dimensional E_8, BW_{16} and Leech lattices were used for shaping a 2304-dimensional coding lattice, respectively. The channel model follows Figure 3.1 where the encoding and indexing are performed as shown in Sections 3.2–3.3. For comparison hypercube shaping as described in Section 3.5 was performed where lattice points were transformed into a hypercube $\mathcal{B} = \{0, 1, \ldots, L-1\}^n$ for an even[2] integer L, where code rate R' is computed by equation (3.58). .

[1]Note that the underlying QC-LDPC codes were not optimized.

[2]When the simplified hypercube shaping method in Subsection 3.5.1 is performed for an a-level Construction D' lattice, it is needed to let L be a multiple of 2^a.

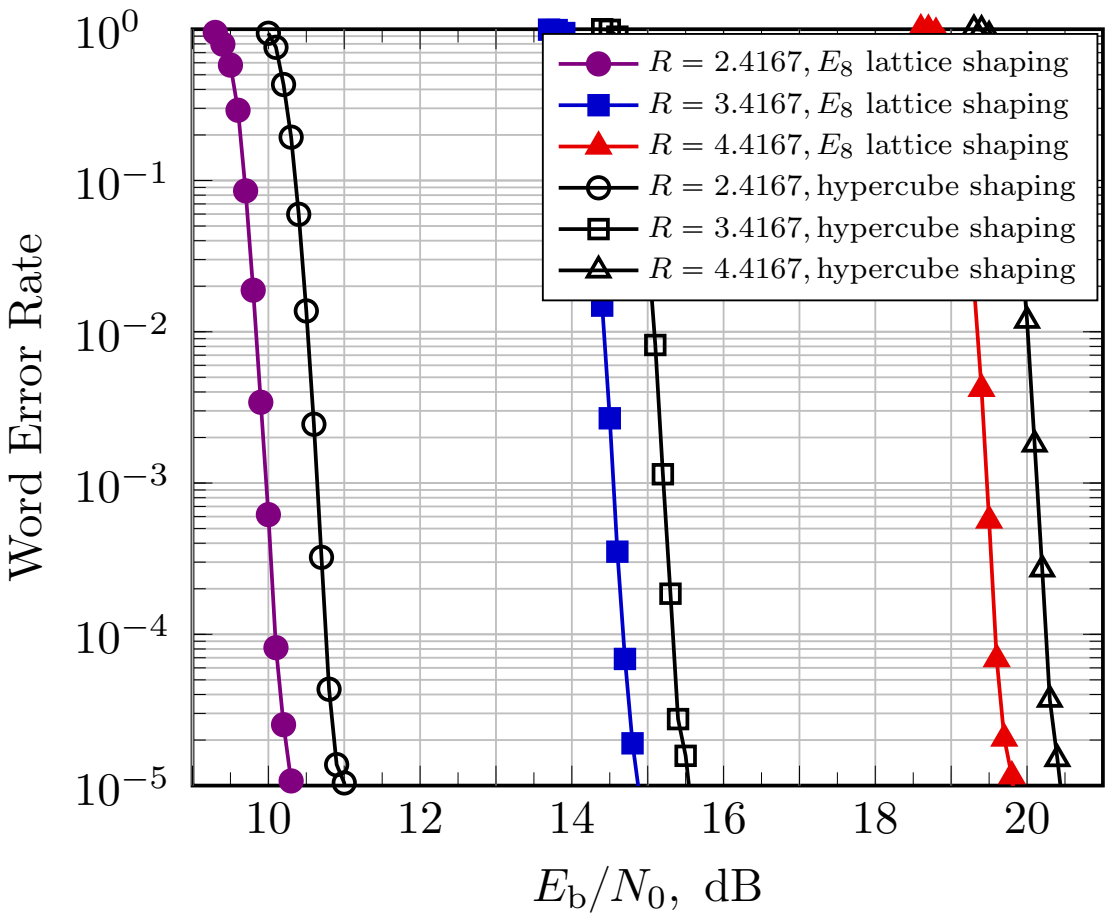


Figure 7.3: Word error rate of shaping a 2304-dimensional Construction D' lattice (formed by QC-LDPC codes) using E_8 lattice shaping and hypercube shaping at a variety of code rates.

Convolutional code lattices were also used as shaping lattices. A variety of convolutional codes were chosen based on the best-found generator polynomials and quantization complexity analysis in Chapter 6, for shaping $n = 2304, 5016, 10008$-dimensional Construction D' lattices separately. The numerical results are given as follows.

7.2.1 E_8, BW_{16} and Leech Lattice Shaping

Well-known low-dimensional lattices were used for shaping high-dimensional lattices because they can provide good shaping gains and their decoding is well-

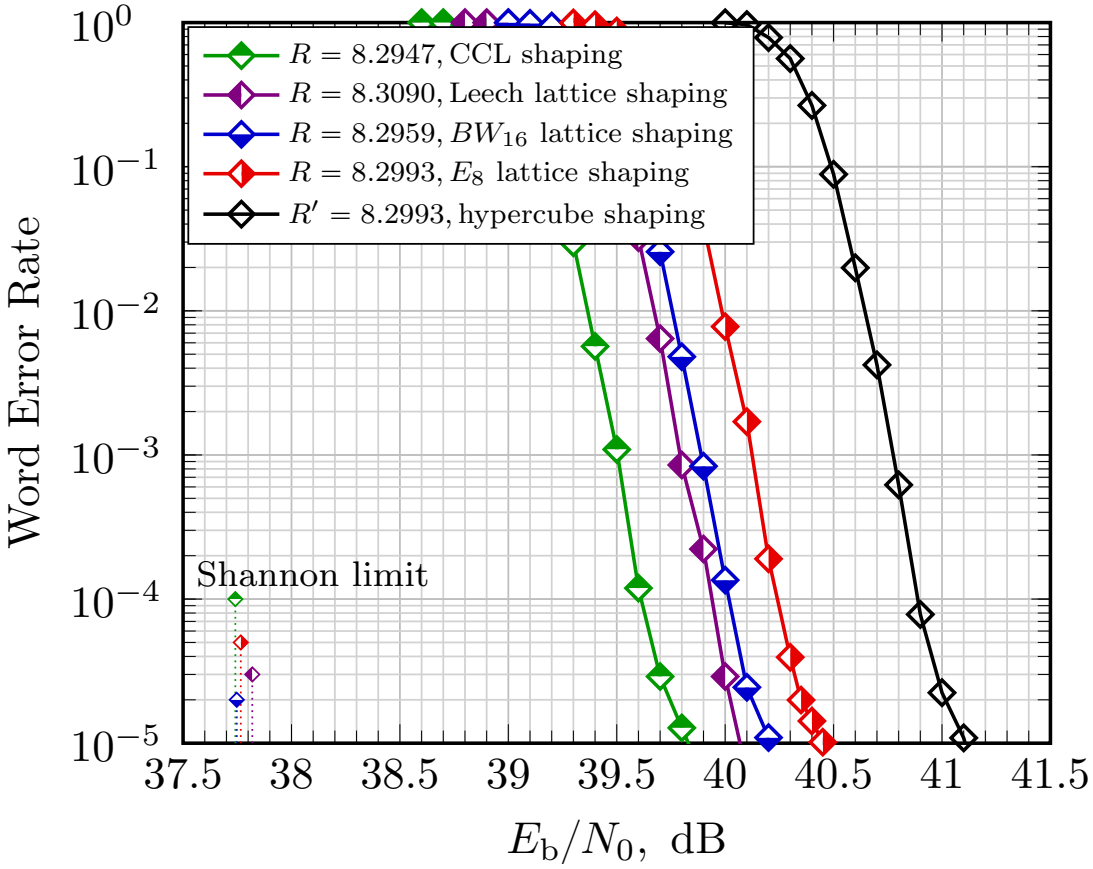


Figure 7.4: Word error rate as a function of E_b/N_0 using a variety of lattices for shaping a 2304-dimensional Construction D' lattice, where the CCL is formed by a zero-tailed convolutional code $\mathscr{C}_6$ with 128 states.

studied. The E_8 lattice, the BW_{16} lattice and the Leech lattice have optimal quantization algorithms [43, 45, 56]. The authors in [32] used the E_8 and BW_{16} lattices for shaping LDLC lattices. At $n = 24$ the Leech lattice has a shaping gain of 1.03 dB, which was used for shaping LDA lattices [33]. Following [32, 33], this dissertation built shaping lattices using direct sum of scaled copies of the E_8, BW_{16}, and Leech lattices by a scale factor K. Let $\mathbf{H}_c$ be the check matrix of an n-dimensional Construction D' coding lattice, and $\mathbf{G}$ be the generator matrix of an n'-dimensional lattice where n/n' is an integer. The factor K is chosen such that $\mathbf{H}_c\mathbf{G}_s \in \mathbb{Z}^n$ where $\mathbf{G}_s$ is a block diagonal matrix of size n/n' with each block $K\mathbf{G}$. Rectangular encoding and its inverse indexing can be efficiently implemented due to the lower-triangular structure in matrix $\mathbf{H}_c$ and $\mathbf{G}_s$. By choosing various K nested lattice codes can be generated with a variety of code rates R.

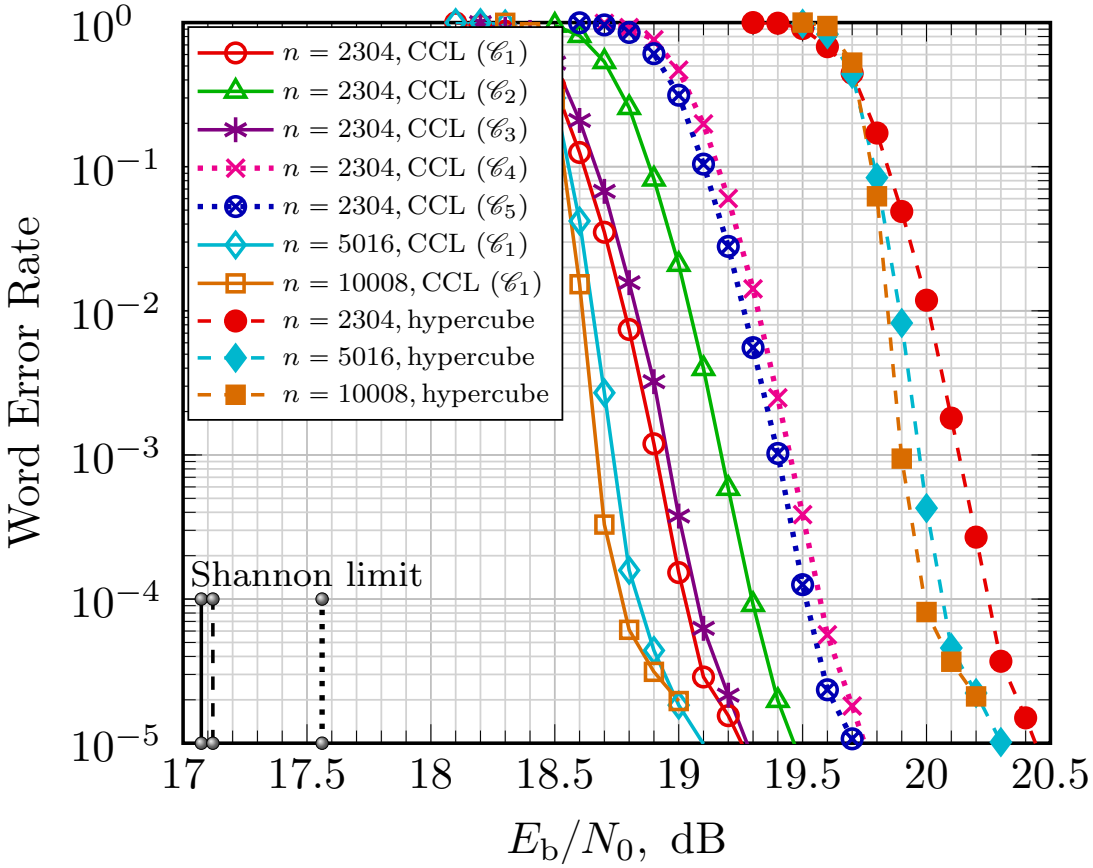


Figure 7.5: Word error rate as a function of E_b/N_0 using various convolutional code lattices (CCLs) based on $\mathscr{C}_1$-$\mathscr{C}_5$ with generator polynomials in Table 6.2 for shaping n-dimensional Construction D' lattices where the code rate is listed in Table 7.1.

For shaping the 2304-dimensional Construction D' lattice, the same code rate for both the E_8 lattice shaping and hypercube shaping can be easily achieved. The word error rate using $K_{E_8} = L = 8, 16, 32$ is shown in Fig. 7.3 as a function of E_b/N_0 given in decibels, suggesting a shaping gain of 0.65 dB. Let $K_{BW_{16}} = 280\sqrt{2}$ and $K_{\text{Leech}} = 168\sqrt{8}$, then BW_{16} and Leech lattice shaping produce code rate approximately 8.2959 and 8.3090, respectively, close to $R = R' = 8.2993$ of choosing $K_{E_8} = L = 472$. The word error rate is given in Figure 7.4 as a function of E_b/N_0. If the code rate differences are taken into account, a 0.65 dB, 0.86 dB and 1.03 dB shaping gain is preserved respectively, as the full shaping gain of the E_8, BW_{16} and Leech lattices.

7.2.2 Convolutional Code Lattices for Shaping Construction D' Lattices

This dissertation considers high-dimensional Construction D' lattices, thus zero-tailed convolutional codes are suitable for constructing convolutional code lattices for shaping. At $n \geqslant 2304$, using zero-tailed convolutional codes provides comparable shaping gain and requires lower quantization complexity than that of tail-biting convolutional codes. A variety of convolutional code lattices based on rate $1/2, 1/3$ zero-tailed convolutional codes selected from Table 6.2 were also used for shaping the proposed QC-LDPC Construction D' Lattices, where the smallest possible scale factor $K = 4$ to satisfy Lemma 3.1 can produce a code rate approximately 2.084 and 1.917 respectively.

Considering that lattices are ideal at high code rate, it is needed to chose $K > 4$ for evaluation. The nested lattice code parameters in simulations are listed in Table 7.1, including the code rates, close to that of hypercube shaping for a fair comparison. The numerical results in terms of word error rate as a function of E_b/N_0 are shown in Figure 7.5. Convolutional code lattice shaping using a rate $1/3$ convolutional code with $m = 5$ was performed for $n = 2304, 5016, 10008$, showing an improvement on the error-correction performance and the shaping gain as n increases. At $n = 10008$, the distance to the capacity of the AWGN channel, that is, the gap between the orange solid curve to the black solid line, is approximately 1.9 dB, considering the tiny code rate difference. For a fixed dimension $n = 2304$, it is shown that a higher shaping gain is achieved by increasing the memory order m. The numerical results of using rate $1/2$ zero-tailed convolutional codes are also provided, where the code rate was chosen as close as possible to hypercube shaping. The resulting shaping gains are approximate to the estimated shaping gains listed in Table 7.1 if the code rate differences are taken into account.

The improvement of the shaping gain provided by a convolutional code lattice compared with that of the E_8, BW_{16} and Leech lattices was also investigated. The simulation results of using these distinct lattices for shaping an $n = 2304$-dimensional QC-LDPC Construction D' lattice are plotted in Figure 7.4. The shaping gain of 1.25 dB was preserved—this is the best-found shaping gain achieved by lattice shaping in the power-constrained channel, to the best of the author's knowledge. For the four shaping lattices: convolutional code lattice, the E_8 lattice, the BW_{16} lattice and the Leech lattice, using a smallest possible scale factor $4, 4, 4\sqrt{2}, 4\sqrt{8}$ respectively for shaping the proposed 2304-dimensional Construction D' lattice, the integers solutions $e_i \in [0, \beta)$ (3.37) are bounded by $\beta = 8, 16, 16, 32$. The values of integers are bounded by $\beta = 944, 944, 1120, 1344$ for the results in Figure 7.4. Regarding the distance to the Shannon limit, while

Table 7.1: Code rate R of nested lattice codes using various convolutional code $\mathscr{C}$ with memory order m, where the convolutional code lattice is scaled by a factor K. The estimated shaping gain γ_{s} is given in decibels. Hypercube side length L is chosen to achieve $R' \approx R$

Dimension	Convolutional code lattice shaping					Hypercube shaping	
n	m	$\mathscr{C}$	γ_{s} (dB)	K	R	L	R'
2304	5	$\mathscr{C}_1$	1.1731	20	4.4074	32	4.4167
5016	5	$\mathscr{C}_1$	1.1772	20	4.4063	32	4.4167
10008	5	$\mathscr{C}_1$	1.1790	20	4.4058	32	4.4167
2304	2	$\mathscr{C}_2$	0.9022	20	4.4061	32	4.4167
2304	4	$\mathscr{C}_3$	1.1259	20	4.4070	32	4.4167
2304	4	$\mathscr{C}_4$	1.1186	24	4.5034	32	4.4167
2304	5	$\mathscr{C}_5$	1.1756	24	4.5038	32	4.4167
2304	7	$\mathscr{C}_6$	1.2500	332	8.2947	472	8.2993

the LDA lattice construction [33] has better performance, it requires nonbinary LDPC codes, whereas the construction in this dissertation uses lower-complexity binary LDPC codes. The LDLC construction [32] has similar performance, but higher decoding complexity than binary LDPC codes.

7.3 Concluding Remarks

The encoding/decoding methods proposed in Chapter 4 were carefully evaluated by simulating the QC-LDPC Construction D' lattices designed in Chapter 5. In the power-constrained AWGN channel, the shaping gains of the E_8 lattice, the BW_{16} lattice and the Leech lattice were preserved when shaping a 2304-dimensional QC-LDPC Construction D' lattice, which are 0.65 dB, 0.86 dB, and 1.03 dB, respectively.

Also, a variety of convolutional code lattices were selected for shaping the QC-LDPC Construction D' lattices with dimension $n = 2304, 5016, 10008$ and their shaping gains were preserved. When the Construction A lattice was constructed from a rate 1/2 binary convolutional code with memory order 7, a shaping gain as high as 1.25 dB out of 1.53 dB was preserved at $n = 2304$. This is the highest shaping gain appeared in the literature of nested lattice coding scheme. And the low-complexity quantization was provided when a rate 1/3 binary convolutional code with memory order 5 was employed, on little penalty of shaping gain, but its 1.17 dB shaping gain is still higher than the 1.03 dB of the Leech lattice.

Chapter 8

Conclusion

This dissertation provided a comprehensive practical lattice coding scheme for power-constrained communications, including a Construction D' lattice construction with good coding properties by employing QC-LDPC codes, a Construction A lattice construction with efficiently achievable shaping gain, two encoding methods and a decoding algorithm for Construction D', as well as a modified indexing method for nested lattice codes. There are still several interesting problems to think about.

Construction of Construction D' Lattices

This work had a methodology for selecting LDPC code design rate, but the rates and degree distributions can be optimized by a density evolution algorithm. Construction D/D' lattices can be designed based upon the probability of error rule, the capacity rule and the minimum distance rule. In [12] the LDPC codes were selected using the probability of error rule. The minimum distance rule was applied to the code selection in [35].

Single parity-check product codes were shown to be a good choice as a second level code, because of their simplicity and good performance. However, the current construction method requires adding a stair case, which cannot be easily triangularizable.

Lattices With Short Dimensions

This dissertation provided a solution for lattices with high dimensions—QC-LDPC Construction D' lattices for coding and the E_8 lattice, the BW_{16} lattice, the Leech lattice or convolutional code lattices for shaping. There exist codes better than LDPC codes at short block length, and can be used to construct BCH code lattices and polar code lattices. One approach is to apply convolutional code lattices for shaping low-to-moderate-dimensional BCH and polar code lattices. Once a matrix construction is found, it is simple to apply the methods provided in this dissertation to these lattices. It was found that convolutional code lattices based on tail-biting convolutional codes can provide a high shaping gain in short

dimensions. Truncated convolutional codes may be chosen under the consideration of quantization complexity.

Encoding/Indexing Using Non-Triangular Matrices

It is convenient to have lattice matrices written in triangular form when performing the indexing operation for a nested lattice code. In general, it is not straightforward to design a triangular matrix for an arbitrary lattice. Lattice generator/check matrix can be transformed into a triangular matrix using a unimodular matrix. However, it is not clear how to find such a unimodular matrix in a systematic way, or if it exists for an arbitrary lattice. One interesting direction is to develop a systematic encoding/indexing method using non-triangular matrices.

Lattices for Compute-and-Forward

Nested lattice code constructions addressed in this dissertation provide a good group structure that might be suitable for compute-and-forward. And more considerations need to be encountered for practical wireless communications. A future research objective is to find nested lattice codes additionally possessing a ring isomorphism which is necessary for compute-and-forward.

Appendix A

Solutions of Congruences

To recover $\mathbf{b}$ without the effect of adding $\mathbf{s}$ (3.28), s_i is chosen as a solution of the system of linear congruences:

$$\theta_{v,i} s_i \equiv 0 \pmod{M_v}, \tag{A.1}$$

where $v = i+1, \ldots, n$. The solution of (A.1) can be found as follows. If $M_v = 1$, an arbitrary integer s_i is a solution. Now consider $M_v > 1$. Let c_v be the greatest common divisor of $\theta_{v,i}$ and M_v:

$$c_v = \gcd(\theta_{v,i}, M_v). \tag{A.2}$$

For fixed v, since zero is divisible by c_v, i.e., $c_v|0$, a linear congruence (A.1) has solutions and are given by the solutions of the equivalent linear congruence:

$$\frac{\theta_{v,i}}{c_v} s_i \equiv 0 \pmod{\frac{M_v}{c_v}}, \tag{A.3}$$

which implies that integers $\frac{\theta_{v,i}}{c_v}$ and $\frac{M_v}{c_v}$ are coprime. Thus the solutions of (A.3) are given by

$$s_i \equiv 0 \pmod{\frac{M_v}{c_v}} \tag{A.4}$$

Solving (A.1) for $v = i+1, \ldots, n$ is equivalent to solve the system of linear congruences (A.4). This has solutions according to the Chinese remainder theorem, since for every pair of congruences within the system $\gcd(\frac{M_l}{c_l}, \frac{M_w}{g_w})|0$ holds, where $l, w \in \{i+1, \ldots, n\}$ and $l \neq w$. The least positive solution is given

$$s'_i = \operatorname{lcm}\left(\frac{M_{i+1}}{c_{i+1}}, \frac{M_{i+2}}{c_{i+2}}, \cdots, \frac{M_n}{c_n}\right), \tag{A.5}$$

and thus $s_i = \beta s'_i$ is also a solution for any integer β.

Appendix B

Best-Found Shaping Gains of Convolutional Code Lattices

Table B.1: Best-found shaping gain and corresponding generator polynomials of convolutional code lattices based on rate 1/2 zero-tailed convolutional codes.

	Dimension n	18	24	30	36	72	144	288	576	1152	2304	2^{20}
$m = 2$	Generator polynomials	$(7,5)$	$(7,5)$	$(7,5)$	$(7,5)$	$(7,5)$	$(7,5)$	$(7,5)$	$(7,5)$	$(7,5)$	$(7,5)$	$(7,5)$
	Shaping gain (dB)	0.7305	0.7771	0.8083	0.8309	0.8956	0.9328	0.9526	0.9629	0.9681	0.9707	0.9734
$m = 3$	Generator polynomials	$(17,13)$	$(17,11)$	$(17,11)$	$(17,11)$	$(17,13)$	$(17,13)$	$(17,13)$	$(17,13)$	$(17,13)$	$(17,13)$	$(17,13)$
	Shaping gain (dB)	0.7395	0.8083	0.8524	0.8804	0.9505	1.0026	1.0314	1.0466	1.0543	1.0583	1.0622
$m = 4$	Generator polynomials	$(35,23)$	$(33,25)$	$(33,25)$	$(33,25)$	$(31,23)$	$(31,23)$	$(31,23)$	$(31,23)$	$(31,23)$	$(31,23)$	$(31,23)$
	Shaping gain (dB)	0.7314	0.8256	0.8813	0.9162	0.9970	1.0542	1.0871	1.1048	1.1139	1.1186	1.1233
$m = 5$	Generator polynomials	$(67,51)$	$(67,51)$	$(77,55)$	$(77,55)$	$(75,57)$	$(75,57)$	$(75,57)$	$(75,57)$	$(75,57)$	$(75,57)$	$(75,57)$
	Shaping gain (dB)	0.7061	0.8292	0.8948	0.9377	1.0330	1.0983	1.1374	1.1588	1.1670	1.1756	1.1814
$m = 6$	Generator polynomials	$(145,137)$	$(175,133)$	$(175,133)$	$(145,137)$	$(165,127)$	$(165,127)$	$(165,127)$	$(165,127)$	$(165,127)$	$(165,127)$	$(165,127)$
	Shaping gain (dB)	0.6297	0.8044	0.8901	0.9402	1.0641	1.1332	1.1757	1.1996	1.2121	1.2186	1.2251
$m = 7$	Generator polynomials	$(365,327)$	$(331,257)$	$(331,257)$	$(331,257)$	$(357,251)$	$(357,251)$	$(357,251)$	$(357,251)$	$(357,251)$	$(357,251)$	$(357,251)$
	Shaping gain (dB)	0.5467	0.7767	0.8884	0.9484	1.0834	1.1563	1.2024	1.2287	1.2428	1.2500	1.2574

Table B.2: Best-found shaping gain and corresponding generator polynomials of convolutional code lattices based on rate 1/2 tail-biting convolutional codes.

	Dimension n	18	24	30	36	72	144	288	576	1152	2304	2^{20}
$m = 2$	Generator polynomials	$(7,6)$	$(7,6)$	$(7,6)$	$(7,5)$	$(7,5)$	$(7,5)$	$(7,5)$	$(7,5)$	$(7,5)$	$(7,5)$	$(7,5)$
	Shaping gain (dB)	0.6893	0.7636	0.7857	0.8631	0.9674	0.9733	0.9734	0.9734	0.9734	0.9734	0.9734
$m = 3$	Generator polynomials	$(16,3)$	$(16,3)$	$(15,6)$	$(15,6)$	$(17,13)$	$(17,13)$	$(17,13)$	$(17,13)$	$(17,13)$	$(17,13)$	$(17,13)$
	Shaping gain (dB)	0.6818	0.7633	0.8340	0.8706	1.0171	1.0608	1.0622	1.0622	1.0622	1.0622	1.0622
$m = 4$	Generator polynomials	$(25,20)$	$(30,7)$	$(30,13)$	$(30,13)$	$(36,15)$	$(31,27)$	$(31,23)$	$(31,23)$	$(31,23)$	$(31,23)$	$(31,23)$
	Shaping gain (dB)	0.6856	0.7632	0.8341	0.8709	1.0168	1.1055	1.1223	1.1233	1.1233	1.1233	1.1233
$m = 5$	Generator polynomials	$(40,25)$	$(70,3)$	$(60,13)$	$(60,13)$	$(74,13)$	$(73,25)$	$(75,57)$	$(75,57)$	$(75,57)$	$(75,57)$	$(75,57)$
	Shaping gain (dB)	0.6825	0.7611	0.8333	0.8707	1.0169	1.1273	1.1763	1.1814	1.1814	1.1814	1.1814
$m = 6$	Generator polynomials	$(100,7)$	$(140,7)$	$(140,13)$	$(140,13)$	$(130,17)$	$(144,57)$	$(161,133)$	$(165,127)$	$(165,127)$	$(165,127)$	$(165,127)$
	Shaping gain (dB)	0.6800	0.7611	0.8325	0.8703	1.0169	1.1264	1.2042	1.2244	1.2251	1.2251	1.2251
$m = 7$	Generator polynomials	$(340,3)$	$(300,7)$	$(320,3)$	$(320,3)$	$(320,17)$	$(250,67)$	$(362,233)$	$(357,251)$	$(357,251)$	$(357,251)$	$(357,251)$
	Shaping gain (dB)	0.6884	0.7602	0.8296	0.8687	1.0168	1.1274	1.2147	1.2540	1.2573	1.2574	1.2574

Table B.3: Best-found shaping gain and corresponding generator polynomials of convolutional code lattices based on rate 1/2 truncated convolutional codes.

	Dimension n	18	24	30	36	72	144	288	576	1152	2304	2^{20}
$m = 2$	Generator polynomials	$(7,5)$	$(7,5)$	$(7,5)$	$(7,5)$	$(7,5)$	$(7,5)$	$(7,5)$	$(7,5)$	$(7,5)$	$(7,5)$	$(7,5)$
	Shaping gain (dB)	0.6938	0.7603	0.8021	0.8301	0.9011	0.9372	0.9552	0.9643	0.9688	0.9711	0.9734
$m = 3$	Generator polynomials	$(17,11)$	$(17,13)$	$(17,13)$	$(17,13)$	$(17,13)$	$(17,13)$	$(17,13)$	$(17,13)$	$(17,13)$	$(17,13)$	$(17,13)$
	Shaping gain (dB)	0.6937	0.7736	0.8277	0.8657	0.9626	1.0121	1.0371	1.0496	1.0560	1.0591	1.0622
$m = 4$	Generator polynomials	$(33,25)$	$(37,26)$	$(37,26)$	$(37,26)$	$(31,27)$	$(31,23)$	$(31,23)$	$(31,23)$	$(31,23)$	$(31,23)$	$(31,23)$
	Shaping gain (dB)	0.6941	0.7753	0.8319	0.8727	0.9906	1.0555	1.0894	1.1063	1.1148	1.1190	1.1233
$m = 5$	Generator polynomials	$(76,47)$	$(61,56)$	$(75,46)$	$(73,52)$	$(75,46)$	$(67,43)$	$(75,57)$	$(75,57)$	$(75,57)$	$(75,57)$	$(75,57)$
	Shaping gain (dB)	0.6949	0.7770	0.8385	0.8856	1.0225	1.1002	1.1398	1.1606	1.1710	1.1762	1.1814
$m = 6$	Generator polynomials	$(163,122)$	$(175,130)$	$(175,130)$	$(164,127)$	$(173,135)$	$(165,127)$	$(165,127)$	$(165,127)$	$(165,127)$	$(165,127)$	$(165,127)$
	Shaping gain (dB)	0.6971	0.7790	0.8400	0.8873	1.0381	1.1293	1.1770	1.2010	1.2130	1.2191	1.2251

Table B.4: Best-found shaping gain and corresponding generator polynomials of convolutional code lattices based on rate 1/3 zero-tailed convolutional codes.

	Dimension n	18	24	30	36	72	144	288	576	1152	2304	$2^{20}+2$
$m=2$	Generator polynomials	$(7,7,5)$	$(7,7,5)$	$(7,7,5)$	$(7,7,5)$	$(7,6,5)$	$(7,6,5)$	$(7,6,5)$	$(7,6,5)$	$(7,6,5)$	$(7,6,5)$	$(7,6,5)$
	Shaping gain (dB)	0.6598	0.6986	0.7180	0.7305	0.8067	0.8541	0.8793	0.8923	0.8989	0.9022	0.9055
$m=3$	Generator polynomials	$(17,15,13)$	$(17,15,13)$	$(17,15,13)$	$(17,15,13)$	$(17,15,13)$	$(17,15,13)$	$(17,15,13)$	$(17,15,13)$	$(17,15,13)$	$(17,15,13)$	$(17,15,13)$
	Shaping gain (dB)	0.6297	0.7374	0.7958	0.8334	0.9364	0.9976	1.0314	1.0491	1.0581	1.0627	1.0673
$m=4$	Generator polynomials	$(37,33,25)$	$(37,33,25)$	$(37,33,25)$	$(37,33,25)$	$(37,33,25)$	$(37,33,25)$	$(37,33,25)$	$(37,33,25)$	$(37,33,25)$	$(37,33,25)$	$(37,33,25)$
	Shaping gain (dB)	0.5469	0.6928	0.7776	0.8315	0.9623	1.0399	1.0841	1.1076	1.1198	1.1259	1.1321
$m=5$	Generator polynomials	$(77,76,73)$	$(71,65,57)$	$(71,65,57)$	$(71,65,57)$	$(65,53,47)$	$(73,57,41)$	$(73,57,41)$	$(73,57,41)$	$(73,57,41)$	$(73,57,41)$	$(73,57,41)$
	Shaping gain (dB)	0.3920	0.6197	0.7466	0.8162	0.9769	1.0677	1.1213	1.1503	1.1654	1.1731	1.1808

Table B.5: Best-found shaping gain and corresponding generator polynomials of convolutional code lattices based on rate 1/3 tail-biting convolutional codes.

	Dimension n	18	24	30	36	72	144	288	576	1152	2304	$2^{20}+2$
$m=2$	Generator polynomials	$(7,6,4)$	$(7,6,2)$	$(7,5,4)$	$(7,6,5)$	$(7,6,5)$	$(7,6,5)$	$(7,6,5)$	$(7,5,3)$	$(7,5,3)$	$(7,6,5)$	$(7,6,5)$
	Shaping gain (dB)	0.6654	0.7361	0.7636	0.8252	0.9022	0.9055	0.9055	0.9055	0.9055	0.9055	0.9055
$m=3$	Generator polynomials	$(16,10,3)$	$(16,10,3)$	$(13,10,7)$	$(13,10,7)$	$(17,15,13)$	$(17,15,13)$	$(17,15,13)$	$(17,15,13)$	$(17,15,13)$	$(17,15,13)$	$(17,15,13)$
	Shaping gain (dB)	0.6621	0.7356	0.7907	0.8462	0.9771	1.0614	1.0673	1.0673	1.0673	1.0673	1.0673
$m=4$	Generator polynomials	$(34,20,3)$	$(30,10,7)$	$(34,13,10)$	$(26,10,7)$	$(36,26,23)$	$(37,33,25)$	$(37,33,25)$	$(37,33,25)$	$(37,33,25)$	$(37,33,25)$	$(37,33,25)$
	Shaping gain (dB)	0.6605	0.7352	0.7899	0.8460	0.9841	1.0988	1.1308	1.1321	1.1321	1.1321	1.1321
$m=5$	Generator polynomials	$(40,34,3)$	$(40,34,3)$	$(70,13,10)$	$(70,13,10)$	$(74,64,31)$	$(75,45,26)$	$(75,67,41)$	$(73,57,41)$	$(75,67,41)$	$(73,57,41)$	$(73,57,41)$
	Shaping gain (dB)	0.6646	0.7337	0.7883	0.8456	0.9839	1.1023	1.1702	1.1806	1.1808	1.1808	1.1808

Table B.6: Best-found shaping gain and corresponding generator polynomials of convolutional code lattices based on rate 1/3 truncated convolutional codes.

	Dimension n	18	24	30	36	72	144	288	576	1152	2304	$2^{20}+2$
$m=2$	Generator polynomials	$(7,6,5)$	$(7,6,5)$	$(7,6,5)$	$(7,6,5)$	$(7,6,5)$	$(7,6,5)$	$(7,6,5)$	$(7,6,5)$	$(7,6,5)$	$(7,6,5)$	$(7,6,5)$
	Shaping gain (dB)	0.6678	0.7246	0.7597	0.7837	0.8442	0.8747	0.8901	0.8978	0.9017	0.9036	0.9055
$m=3$	Generator polynomials	$(17,15,13)$	$(17,15,13)$	$(17,15,13)$	$(17,15,13)$	$(17,15,13)$	$(17,15,13)$	$(17,15,13)$	$(17,15,13)$	$(17,15,13)$	$(17,15,13)$	$(17,15,13)$
	Shaping gain (dB)	0.6781	0.7558	0.8107	0.8508	0.9571	1.0117	1.0395	1.0534	1.0603	1.0638	1.0673
$m=4$	Generator polynomials	$(37,32,21)$	$(35,27,21)$	$(33,25,23)$	$(33,25,23)$	$(37,33,25)$	$(37,33,25)$	$(35,27,21)$	$(33,25,23)$	$(33,25,23)$	$(37,33,25)$	$(37,33,25)$
	Shaping gain (dB)	0.6801	0.7558	0.8122	0.8572	0.9874	1.0592	1.0955	1.1138	1.1230	1.1275	1.1321

Appendix C

QC-LDPC Prototype Matrices

The prototype matrices for QC-LDPC codes with block length $n = 5016, 10008, 50016$ are provided, which are also available at https://github.com/fanzhou-code/qcldpc.

Table C.1: Prototype matrix of $\mathbf{H}_0$ with $Z = 209$ and $n = 5016$ where $*$ denotes a double circulant

-1	-1	190	-1	93	113	-1	-1	76	188	-1	171	0	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	44	-1	-1	160	-1	199	155	-1	63	-1	0	0	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
133	-1	72	-1	-1	50	138	-1	148	-1	-1	-1	-1	0	0	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	78	120	121	-1	123	-1	-1	147	-1	-1	-1	-1	-1	0	0	-1	-1	-1	-1	-1	-1	-1	-1
-1	98	-1	183	19	-1	-1	-1	-1	-1	-1	-1	-1	50	-1	115	0	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	97	73	63	-1	-1	-1	-1	-1	0	-1	-1	157	0	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	54	34	3	-1	-1	-1	0	-1	-1	-1	-1	176	0	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	79	-1	27	205	-1	-1	-1	0	-1	-1	36	0	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	67	82	180	-1	-1	0	-1	-1	-1	-1	188	0	-1	-1	-1
-1	-1	102	14	118	-1	-1	-1	-1	-1	-1	-1	-1	-1	67	-1	-1	-1	-1	-1	23	0	-1	-1
77	-1	56	-1	-1	180	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	156	0	0
86	106	-1	-1	-1	-1	-1	-1	-1	125	-1	-1	71	-1	-1	-1	-1	-1	-1	-1	-1	-1	152/15*	0

Table C.2: Prototype matrix of $\mathbf{H}_1$ with $Z = 209$ and $n = 5016$ where $*$ denotes a double circulant

77	98	56	183	19	180	54	34	3	67	82	180	0	50	0	115	0	176	0	188	0	156	0	0
86	106	102	14	118	97	73	63	79	125	27	205	71	0	67	0	157	0	36	0	23	0	152/15*	0

Table C.3: Prototype matrix of $\mathbf{H}_0$ with $Z = 417$ and $n = 10008$ where $*$ denotes a double circulant

-1	-1	95	-1	56	413	-1	-1	172	147	-1	97	0	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	139	-1	-1	121	-1	369	51	-1	28	-1	0	0	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
43	-1	190	-1	-1	295	22	-1	239	-1	-1	-1	-1	0	0	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	211	308	377	-1	224	-1	-1	395	-1	-1	-1	-1	-1	0	0	-1	-1	-1	-1	-1	-1	-1	-1
-1	320	-1	402	258	-1	-1	-1	-1	-1	-1	-1	-1	156	-1	314	0	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	82	78	350	-1	-1	-1	-1	-1	0	-1	-1	260	0	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	20	51	108	-1	-1	-1	0	-1	-1	-1	-1	165	0	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	412	-1	157	23	-1	-1	-1	0	-1	-1	151	0	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	88	70	376	-1	-1	0	-1	-1	-1	-1	38	0	-1	-1	-1
-1	-1	213	262	160	-1	-1	-1	-1	-1	-1	-1	-1	-1	347	-1	-1	-1	-1	-1	143	0	-1	-1
123	-1	160	-1	-1	288	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	229	0	0
23	119	-1	-1	-1	-1	-1	-1	-1	260	-1	-1	331	-1	-1	-1	-1	-1	-1	-1	-1	-1	193/339*	0

Table C.4: Prototype matrix of $\mathbf{H}_1$ with $Z = 417$ and $n = 10008$ where $*$ denotes a double circulant

123	320	160	402	258	288	20	51	108	88	70	376	0	156	0	314	0	165	0	38	0	229	0	0
23	119	213	262	160	82	78	350	412	260	157	23	331	0	347	0	260	0	151	0	143	0	193/339*	0

Table C.5: Prototype matrix of $\mathbf{H}_0$ with $Z = 2084$ and $n = 50016$ where $*$ denotes a double circulant

-1	-1	9	-1	2022	496	-1	-1	619	1690	-1	1997	0	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	1699	-1	-1	1704	-1	1033	1164	-1	2080	-1	0	0	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
7	-1	1331	-1	-1	846	1595	-1	142	-1	-1	-1	-1	0	0	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	1065	935	1717	-1	972	-1	-1	145	-1	-1	-1	-1	-1	0	0	-1	-1	-1	-1	-1	-1	-1	-1
-1	804	-1	1776	1753	-1	-1	-1	-1	-1	-1	-1	-1	1865	-1	6	0	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	1983	1198	347	-1	-1	-1	-1	-1	0	-1	-1	1334	0	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	1908	680	349	-1	-1	-1	0	-1	-1	-1	-1	1674	0	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	1974	-1	2057	1106	-1	-1	-1	0	-1	-1	512	0	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	1481	314	156	-1	-1	0	-1	-1	-1	-1	135	0	-1	-1	-1
-1	-1	510	974	165	-1	-1	-1	-1	-1	-1	-1	-1	-1	1739	-1	-1	-1	-1	-1	549	0	-1	-1
320	-1	1674	-1	-1	2011	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	215	0	0
1114	648	-1	-1	-1	-1	-1	-1	-1	2022	-1	-1	651	-1	-1	-1	-1	-1	-1	-1	-1	-1	1009/795*	0

Table C.6: Prototype matrix of $\mathbf{H}_1$ with $Z = 2084$ and $n = 50016$ where $*$ denotes a double circulant

320	804	1674	1776	1753	2011	1908	680	349	1481	314	156	0	1865	0	6	0	1674	0	135	0	215	0	0
1114	648	510	974	165	1983	1198	347	1974	2022	2057	1106	651	0	1739	0	1334	0	512	0	549	0	1009/795*	0

References

[1] C. E. Shannon, "A mathematical theory of communication," *The Bell System Technical Journal*, vol. 27, pp. 379–423, 623–656, July / October 1948.

[2] G. Böcherer, F. Steiner, and P. Schulte, "Bandwidth efficient and rate-matched low-density parity-check coded modulation," *IEEE Transactions on Communications*, vol. 63, no. 12, pp. 4651–4665, December 2015.

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

[4] J. H. Conway and N. J. A. Sloane, "A fast encoding method for lattice codes and quantizers," *IEEE Transactions on Information Theory*, vol. 29, no. 6, pp. 820–824, November 1983.

[5] G. D. Forney, Jr., "Multidimensional constellations—Part II: Voronoi constellations," *IEEE Journal on Selected Areas in Communications*, vol. 7, no. 6, pp. 941–958, August 1989.

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

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

[8] U. Erez and S. ten Brink, "A close-to-capacity dirty paper coding scheme," *IEEE Transactions on Information Theory*, vol. 51, no. 10, pp. 3417–3432, October 2005.

[9] F. Zhou and B. M. Kurkoski, "Shaping LDLC lattices using convolutional code lattices," *IEEE Communications Letters*, vol. 21, no. 4, pp. 730–733, April 2017.

[10] T. Matsumine, B. M. Kurkoski, and H. Ochiai, "Construction D lattice decoding and its application to BCH code lattices," in *Proceedings IEEE Global Telecommunications Conference*, Abu Dhabi, United Arab Emirates, December 2018, pp. 1–6.

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

[12] P. R. Branco da Silva and D. Silva, "Multilevel LDPC lattices with efficient encoding and decoding and a generalization of Construction D'," *IEEE Transactions on Information Theory*, vol. 65, no. 5, pp. 3246–3260, May 2019.

[13] G. D. Forney, Jr. and G. Ungerboeck, "Modulation and coding for linear Gaussian channels," *IEEE Transactions on Information Theory*, vol. 44, no. 6, pp. 2384–2415, 1998.

[14] 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, October 2011.

[15] N. di Pietro, G. Zémor, and J. J. Boutros, "LDA lattices without dithering achieve capacity on the Gaussian channel," *IEEE Transactions on Information Theory*, vol. 64, no. 3, pp. 1561–1594, March 2018.

[16] B. M. Kurkoski, "Encoding and indexing of lattice codes," *IEEE Transactions on Information Theory*, vol. 64, no. 9, pp. 6320–6332, September 2018.

[17] U. Erez, S. Litsyn, and R. Zamir, "Lattices which are good for (almost) everything," *IEEE Transactions on Information Theory*, vol. 51, no. 10, pp. 3401–3416, October 2005.

[18] A. Viterbi, "Error bounds for convolutional codes and an asymptotically optimum decoding algorithm," *IEEE Transactions on Information Theory*, vol. 13, no. 2, pp. 260–269, April 1967.

[19] G. D. Forney, Jr. and L.-F. Wei, "Multidimensional constellations—Part I: Introduction, figures of merrit, and generalized cross constellations," *IEEE Journal on Selected Areas in Communications*, vol. 7, no. 6, pp. 877–891, August 1989.

[20] G. D. Forney, Jr., "Trellis shaping," *IEEE Transactions on Information Theory*, vol. 38, no. 2, pp. 281–300, 1992.

[21] M. W. Marcellin and T. R. Fischer, "Trellis coded quantization of memoryless and Gauss-Markov sources," *IEEE Transactions on Communications*, vol. 38, no. 1, pp. 82–93, January 1990.

[22] G. Ungerboeck, "Channel coding with multilevel/phase signals," *IEEE Transactions on Information Theory*, vol. 28, no. 1, pp. 55–67, January 1982.

[23] B. Kudryashov and K. Yurkov, "Linear code-based vector quantization for independent random variables," 2008, arXiv:0805.2379 [cs.IT].

[24] B. D. Kudryashov and K. V. Yurkov, "Near-optimum low-complexity lattice quantization," in *Proceedings of IEEE International Symposium on Information Theory*, Austin, TX, USA, June 2010, pp. 1032–1036.

[25] A. Sakzad, M. Sadeghi, and D. Panario, "Construction of turbo lattices," in *Proceedings 48th Annual Allerton Conference on Communication, Control, and Computing*, Monticello, IL, USA, September 2010, pp. 14–21.

[26] Y. Yan and C. Ling, "A construction of lattices from polar codes," in *Proceedings of the IEEE Information Theory Workshop*, Lausanne, Switzerland, 2012, pp. 124–128.

[27] L. Liu, Y. Yan, C. Ling, and X. Wu, "Construction of capacity-achieving lattice codes: Polar lattices," *IEEE Transactions on Communications*, vol. 67, no. 2, pp. 915–928, February 2019.

[28] J. H. Conway and N. Sloane, "Voronoi regions of lattices, second moments of polytopes, and quantization," *IEEE Transactions on Information Theory*, vol. 28, no. 2, pp. 211–226, Mar 1982.

[29] B. Kurkoski, J. Dauwels, and H.-A. Loeliger, "Power-constrained communications using LDLC lattices," in *Proceedings of IEEE International Symposium on Information Theory*, Seoul, South Korea, June–July 2009, pp. 739–743.

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

[31] H. Khodaiemehr, M.-R. Sadeghi, and A. Sakzad, "Practical encoder and decoder for power constrained QC LDPC-lattice codes," *IEEE Transactions on Communications*, vol. 65, no. 2, pp. 486–500, February 2017.

[32] N. S. Ferdinand, B. M. Kurkoski, M. Nokleby, and B. Aazhang, "Low-dimensional shaping for high-dimensional lattice codes," *IEEE Transactions on Wireless Communications*, vol. 15, no. 11, pp. 7405–7418, November 2016.

[33] N. di Pietro and J. J. Boutros, "Leech constellations of Construction-A lattices," *IEEE Transactions on Communications*, vol. 65, no. 11, pp. 4622–4631, November 2017.

[34] H. Buglia and R. R. Lopes, "Voronoi shaping for lattices with efficient encoding," *IEEE Communications Letters*, vol. 25, no. 5, pp. 1439–1442, May 2021.

[35] S. Chen, B. M. Kurkoski, and E. Rosnes, "Construction D′ lattices from quasi-cyclic low-density parity-check codes," in *Proceedings International Symposium on Turbo Codes & Iterative Information Processing*, Hong Kong, China, December 2018, pp. 1–5.

[36] R. Zamir, *Lattice Coding for Signals and Networks.* Cambridge, UK: Cambridge, 2014.

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

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

[39] J. M. Hammersley and D. C. Handscomb, *Monte Carlo Methods.* New York: John Wiley, 1964, ISBN 978-94-009-5819-7.

[40] J. H. Halton, "A retrospective and prospective survey of the Monte Carlo method," *SIAM Review*, vol. 12, no. 1, pp. 1–63, 1970.

[41] S. I. R. Costa, F. Oggier, A. Campello, J.-C. Belfiore, and E. Viterbo, *Lattices Applied to Coding for Reliable and Secure Communications*, ser. SpringerBriefs in Mathematics. Cham, Switzerland: Springer International Publishing, 2017.

[42] H. Cohen, *A course in computational algebraic number theory*, 3rd ed., ser. Graduate texts in mathematics. Berlin ; New York: Springer, 1996, no. 138.

[43] J. Conway and N. Sloane, "Fast quantizing and decoding and algorithms for lattice quantizers and codes," *IEEE Transactions on Information Theory*, vol. 28, no. 2, pp. 227–232, March 1982.

[44] E. S. Barnes and G. E. Wall, "Some extreme forms defined in terms of Abelian groups," *Journal of the Australian Mathematical Society*, vol. 1, no. 1, pp. 47–63, August 1959.

[45] J. Conway and N. Sloane, "On the Voronoi regions of certain lattices," *SIAM Journal on Discrete Mathematics*, vol. 5, no. 3, pp. 294–305, 1984.

[46] G. D. Forney, Jr., "Coset codes—Part II: Binary lattices and related codes," *IEEE Transactions on Information Theory*, vol. 34, no. 5, pp. 1152–1187, 1988.

[47] D. Micciancio and A. Nicolosi, "Efficient bounded distance decoders for Barnes-Wall lattices," in *2008 IEEE International Symposium on Information Theory*. Toronto, ON, Canada: IEEE, July 2008, pp. 2484–2488.

[48] V. Corlay, J. J. Boutros, P. Ciblat, and L. Brunel, "On the decoding of Barnes-Wall lattices," in *2020 IEEE International Symposium on Information Theory (ISIT)*. Los Angeles, CA, USA: IEEE, June 2020, pp. 519–524.

[49] V. Corlay, "Decoding algorithms for lattices," Ph.D. dissertation, Institut Polytechnique de Paris, 2020.

[50] J. Leech, "Some sphere packings in higher space," *Canadian Journal of Mathematics*, vol. 16, pp. 657–682, 1964.

[51] ——, "Notes on sphere packings," *Canadian Journal of Mathematics*, vol. 19, pp. 251–267, 1967.

[52] J. Conway and N. Sloane, "Soft decoding techniques for codes and lattices, including the Golay code and the Leech lattice," *IEEE Transactions on Information Theory*, vol. 32, no. 1, pp. 41–50, Jan 1986.

[53] G. Lang and F. Longstaff, "A Leech lattice modem," *IEEE Journal on Selected Areas in Communications*, vol. 7, no. 6, pp. 968–973, August 1989.

[54] Y. Be'ery, B. Shahar, and J. Snyders, "Fast decoding of the Leech lattice," *IEEE Journal on Selected Areas in Communications*, vol. 7, no. 6, pp. 959–967, Aug 1989.

[55] A. Vardy and Y. Be'ery, "Maximum likelihood decoding of the Leech lattice," *IEEE Transactions on Information Theory*, vol. 39, no. 4, pp. 1435–1444, July 1993.

[56] E. Viterbo and J. Bouros, "A universal lattice code decoder for fading channels," *IEEE Transactions on Information Theory*, vol. 45, no. 5, pp. 1639–1642, July 1999.

[57] 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, August 2005.

[58] H. Vikalo and B. Hassibi, "On the sphere-decoding algorithm II. Generalizations, second-order statistics, and applications to communications," *IEEE Transactions on Signal Processing*, vol. 53, no. 8, pp. 2819–2834, August 2005.

[59] B. M. Kurkoski, "On the encoding and indexing of lattice codes," in *Proceedings of the 39th Symposium on Information Theory and Its Applications*. Takayama, Gifu, Japan: IEICE, December 2016, pp. 378–383.

[60] ——, "Rewriting codes for flash memories based upon lattices, and an example using the E8 lattice," in *Proceedings IEEE Global Telecommunications Conference*. Miami, USA: IEEE, December 2010, pp. 1923–1927.

[61] T. Richardson and R. Urbanke, "Efficient encoding of low-density parity-check codes," *IEEE Transactions on Information Theory*, vol. 47, no. 2, pp. 638–656, February 2001.

[62] D. V. Ouellette, "Schur complements and statistics," *Linear Algebra and its Applications*, vol. 36, pp. 187–295, 1981.

[63] A. Vem, Y.-C. Huang, K. R. Narayanan, and H. D. Pfister, "Multilevel lattices based on spatially-coupled LDPC codes with applications," in *Proceedings of IEEE International Symposium on Information Theory*, Honolulu, HI, USA, June 2014, pp. 2336–2340.

[64] E. Rosnes, Ø. Ytrehus, M. A. Ambroze, and M. Tomlinson, "Addendum to "An efficient algorithm to find all small-size stopping sets of low-density parity-check matrices"," *IEEE Transactions on Information Theory*, vol. 58, no. 1, pp. 164–171, January 2012.

[65] M. Fossorier, "Quasi-Cyclic low-density parity-check code from circulant permutation matrices," *IEEE Transactions on Information Theory*, vol. 50, no. 8, pp. 1788–1793, August 2004.

[66] F. Zhou, A. Fitri, K. Anwar, and B. M. Kurkoski, "Encoding and decoding Construction D' lattices for power-constrained communications," in *Proceedings of IEEE International Symposium on Information Theory*. Melbourne, Australia: IEEE, July 2021, pp. 1005–1010.

[67] W. Sułek, "Protograph based low-density parity-check codes design with mixed integer linear programming," *IEEE Access*, vol. 7, pp. 1424–1438, 2019.

[68] J. Pearl, *Probabilistic Reasoning in Intelligent Systems.* San Mateo, CA, USA: Morgan Kaufmann, 1988.

[69] R. M. Tanner, "A recursive approach to low complexity codes," *IEEE Transactions on Information Theory*, no. 5, pp. 533–547, September 1981.

[70] K. Lally, "Explicit construction of type-II QC LDPC codes with girth at least 6," in *Proceedings of IEEE International Symposium on Information Theory*, Nice, June 2007, pp. 2371–2375.

[71] G. Zhang, Y. Hu, Y. Fang, and J. Wang, "Constructions of type-II QC-LDPC codes with girth eight from Sidon sequence," *IEEE Transactions on Communications*, vol. 67, no. 6, pp. 3865–3878, June 2019.

[72] T. J. Richardson, M. A. Shokrollahi, and R. L. Urbanke, "Design of capacity-approaching irregular low-density parity-check codes," *IEEE Transactions on Information Theory*, vol. 47, no. 1, pp. 619–637, February 2001.

[73] F. Zhou and B. M. Kurkoski, "Shaping gain of lattices based on convolutional codes and Construction A," in *International Symposium on Information Theory and its Applications*, Singapore, October 2018, pp. 183–187.

[74] P. Elias, "Coding for noisy channels," in *IRE Conv. Rec.*, ser. Pt. 4, 1955, pp. 37–46.

[75] C. Berrou, A. Glavieux, and P. Thitimajshima, "Near Shannon limit error-correcting coding and decoding: Turbo-codes. 1," in *Proceedings IEEE International Conference on Communications*, vol. 2. Geneva, Switzerland: IEEE, May 1993, pp. 1064–1070.

[76] S. Lin and D. J. Costello, *Error Control Coding*, 2nd ed. Upper Saddle River, NJ, USA: Prentice-Hall, Inc., 2004.

[77] *Communications Toolbox Reference.* Natick, MA, USA: The MathWorks Inc., September 2021, MATLAB version 9.11.0 (R2021b).

[78] J. P. Odenwalder, "Optimal decoding of convolutional codes," Ph.D. dissertation, School of Engineering and Applied Science, University of California, Los Angeles, 1970.

[79] K. Larsen, "Short convolutional codes with maximal free distance for rates 1/2, 1/3, and 1/4 (Corresp.)," *IEEE Transactions on Information Theory*, vol. 19, no. 3, pp. 371–372, May 1973.

[80] R. Shao, Shu Lin, and M. Fossorier, "Two decoding algorithms for tailbiting codes," *IEEE Transactions on Communications*, vol. 51, no. 10, pp. 1658–1665, October 2003.

[81] Y.-P. E. Wang and R. Ramesh, "To bite or not to bite-a study of tail bits versus tail-biting," in *Personal, Indoor, and Mobile Radio Communications*, vol. 2, Taipei, October 1996, pp. 317–321.

[82] M. M. Molu, K. Cumanan, M. Bashar, and A. Burr, "On convolutional lattice codes and lattice decoding using trellis structure," *IEEE Access*, vol. 4, pp. 9702–9715, 2016.

Publications

[1] **F. Zhou** and B. M. Kurkoski, "Construction D' lattices for power-constrained communications," submitted to *IEEE Transactions on Communications*, available at arXiv:2103.08263 [cs.IT]. Under review—minor revision.

[2] **F. Zhou**, A. Fitri, K. Anwar, and B. M. Kurkoski, "Encoding and decoding Construction D' lattices for power-constrained communications," in *Proceedings of the 2021 IEEE International Symposium on Information Theory*, Melbourne, Australia, July 2021, pp. 1005–1010.

[3] **F. Zhou** and B. M. Kurkoski, "Shaping gain of lattices based on convolutional codes and Construction A," in *Proceedings of the 2018 International Symposium on Information Theory and its Applications*, Singapore, October 2018, pp. 183–187.

[4] **F. Zhou** and B. M. Kurkoski, "On low-dimensional convolutional code lattices which are good for shaping," *Croucher Summer Course in Information Theory*, Hong Kong, China, July 2017.

[5] **F. Zhou** and B. M. Kurkoski, "Shaping LDLC lattices using convolutional code lattices," *IEEE Communications Letters*, vol. 21, no. 4, pp. 730–733, April 2017.