EXTENDED LP BOUND FOR LCD CODES AND NEW BINARY AND TERNARY LCD CODES

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF APPLIED MATHEMATICS
OF
MIDDLE EAST TECHNICAL UNIVERSITY

BY

EMRE KARABAKLA

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR
THE DEGREE OF MASTER OF SCIENCE
IN
CRYPTOGRAPHY

JULY 2025

Approval of the thesis:

EXTENDED LP BOUND FOR LCD CODES AND NEW BINARY AND TERNARY LCD CODES

submitted by **EMRE KARABAKLA** in partial fulfillment of the requirements for the degree of **Master of Science in Cryptography Department, Middle East Technical University** by,

Prof. Dr. A. Sevtap Kestel
Dean, Graduate School of **Applied Mathematics** _____

Assoc. Prof. Dr. Oğuz Yayla
Head of Department, **Cryptography** _____

Assist. Prof. Dr. BUKET ÖZKAYA
Supervisor, **Cryptography** _____

Examining Committee Members:

Assoc. Prof. Dr. Oğuz Yayla
Cryptography, Middle East Technical University _____

Assist. Prof. Dr. BUKET ÖZKAYA
Cryptography, Middle East Technical University _____

Prof. Dr. Ferruh Özbudak
Faculty of Engineering and Natural Sciences, Sabancı University _____

Prof. Dr. Patrick Solé
I2M, Aix-Marseille University _____

Dr. Markus Grassl
ICTQT, University of Gdańsk _____

Date: _____

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

Name, Last Name: EMRE KARABAKLA

Signature :

ABSTRACT

EXTENDED LP BOUND FOR LCD CODES AND NEW BINARY AND TERNARY LCD CODES

Karabakla, Emre
M.S., Department of Cryptography
Supervisor : Assist. Prof. Dr. BUKET ÖZKAYA

July 2025, 63 pages

The linear-programming (LP) methodology proposed by Dougherty et al. [9], originally formulated for binary LCD codes, is generalized herein to arbitrary q-ary settings. A unified LP bound is derived that subsumes and strengthens existing binary and ternary limits, yielding strictly tighter theoretical constraints. Within this framework, refined LP-bound tables for binary LCD codes are presented—augmenting and improving upon previously known entries—and, for the first time, analogous tables for ternary LCD codes are compiled. Several canonical construction results are lifted from the binary and ternary cases to arbitrary q, thereby producing novel LCD codes with enhanced parameters. Finally, algebraic analysis of cyclic and quasi-cyclic structures elucidates new criteria and techniques for the construction of LCD codes, offering insights into their construction.

Keywords: Linear programming bound, linear complementary dual code, cyclic code, quasi-cyclic code.

ÖZ

ÜÇLÜ LCD KODLARI İÇİN DOĞRUSAL PROGRAMLAMA SINIRLARI

Karabakla, Emre
Yüksek Lisans, Kriptografi Bölümü
Tez Yöneticisi : Dr. Öğr. Üyesi BUKET ÖZKAYA

Temmuz 2025, 63 sayfa

Dougherty ve diğ. [9] tarafından ikili LCD kodları için geliştirilen yöntemi genişleterek, öncelikle q-ary LCD kodlarına uygulanabilir, genelleştirilmiş ve geliştirilmiş bir doğrusal programlama (LP) sınırı öneriyoruz. Bu sayede, çalışma kapsamı ikili kodların ötesine taşınmaktadır. Bu birleştirilmiş çerçeve, hem ikili hem de üçlü LCD kodları için daha sıkı ve daha hassas teorik sınırların elde edilmesini sağlamaktadır. Bu çalışmanın bir parçası olarak, mevcut literatürü geliştirip genişleterek ikili LCD kodları için genişletilmiş LP sınır tabloları derliyoruz. Ayrıca, araştırmada önemli bir boşluğu doldurarak üçlü LCD kodlarına özel ilk LP sınır tablolarını sunuyoruz. Sınırların belirlenmesinin ötesinde, LCD kodları için önceden bilinen çeşitli sonuçları genelleştirerek, daha önce bildirilenlerden daha iyi parametrelere sahip yeni ikili ve üçlü LCD kodlarının inşasına olanak sağlıyoruz. Son olarak, dairesel ve yarı-dairesel LCD kodlarının yapısal özelliklerini cebirsel teknikler aracılığıyla inceleyerek, bu kodların inşası ve potansiyel uygulamaları hakkında yeni bakış açıları sunuyoruz.

Anahtar Kelimeler: Doğrusal programlama sınırı, doğrusal tamamlayıcı ikili kod, çevrimsel kod, yarı-çevrimsel kod.

To my beloved wife.

ACKNOWLEDGMENTS

I would like to express my deepest appreciation to my thesis supervisor, Assoc. Prof. Dr. Buket Özkaya, for her patient guidance, invaluable support, and insightful advice throughout the development and preparation of this thesis. Her encouragement, dedication, and willingness to share her expertise have been truly inspiring and instrumental in my journey. I am also profoundly grateful to my dear wife for her unwavering support, understanding, and patience, which have given me strength and motivation at every step of this process.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF ABBREVIATIONS

LCD	Linear Complementary Dual
LP	Linear Programming
ILP	Integer Linear Programming
MDS	Maximum Distance Separable

CHAPTER 1

INTRODUCTION

Linear complementary dual (LCD) codes have become a cornerstone of Boolean masking strategies designed to thwart side-channel attacks, including fault injection and hardware Trojan insertion. First introduced by Massey in 1992 for binary adder channels [21], these codes have experienced a resurgence due to their demonstrable benefits in improving side-channel resistance. Moreover, Carlet et al. [8]. have shown that for $q > 3$ every linear code is equivalent to an LCD code, which highlights the special significance of dedicated bounds in the cases $q = 2$ and $q = 3$.

The seminal work of Dougherty et al. [9] established a linear-programming (LP) bound for binary LCD codes of arbitrary length. We have extended these bounds to binary codes of length up to $n = 60$ and introduced LP bounds for ternary codes of length up to $n = 50$. Explicit tables for binary LCD codes of length $n \leq 30$ with bounds through $n \leq 40$ were compiled by Bouyuklieva [7], and later augmented to $n \leq 50$ by Wang et al. [25]. Meanwhile, exact minimum-distance results for binary $[n, k]$ LCD codes with $k \in \{1, 2, 3, 4, 5, n, n-1, \ldots, n-6\}$ and for ternary codes with $k \in \{1, 2, 3, n, n-1, \ldots, n-5\}$ have been systematically derived, as summarized in Araya et al. [3]. In what follows, codes whose parameters are already determined in these works are omitted from our searches.

This thesis advances the theory and construction of LCD codes in three principal aspects. We generalize the LP methodology of Dougherty et al. [9] to arbitrary q-ary codes, thereby obtaining a unified LP bound that both subsumes the known binary and ternary limits and tightens them. Within this framework, we compile updated LP-bound tables for binary LCD codes up to $n = 60$ and, for the first time, for

ternary LCD codes up to $n = 50$. Complementing these theoretical advances, we introduce novel algebraic constructions based on extension, puncturing, and concatenation. These constructions preserve the complementary-dual property while yielding binary and ternary LCD codes with improved parameters. Finally, by exploiting the factorization of $x^n - 1$ into self-reciprocal factors, we develop systematic algorithms for the construction of cyclic and quasi-cyclic LCD codes, revealing new structural insights and criteria.

This thesis is organized as follows, Chapter 2 reviews the necessary preliminaries, including key definitions and notation. Chapter 3 presents the extended LP bound for q-ary LCD codes, details its integer-linear-programming formulation, and reports the resulting bound tables. In Chapter 4 we describe our new algebraic-construction methods, analyze their minimum-weight properties, and summarize the construction results. Chapter 5 is devoted to cyclic and quasi-cyclic LCD codes, where we outline algorithmic procedures grounded in the factorization of $x^n - 1$.

CHAPTER 2

PRELIMINARIES

Throughout the thesis, q denotes a prime power and $\mathbb{F}_q$ denotes the finite field of size q. If not defined differently, $\mathrm{char}(\mathbb{F}_q) = p$, where p is the prime number satisfying $q = p^m$ for $m \in \mathbb{Z}^+$. A linear code C of length n is a (vector) subspace of $\mathbb{F}_q^n$. The Hamming weight of a codeword $c \in C$ is denoted by wt(c), which is the number of nonzero coordinates in c. The minimum distance $d(C)$ of C is defined as the minimum weight among its nonzero codewords, that is,

$$d(C) := \min\{\text{wt(c)} : c \in C \setminus \{0\}\}.$$

With this notation, we say that C is an $[n, k, d]_q$ code over $\mathbb{F}_q$ if $C \subseteq \mathbb{F}_q^n$, $\dim_{\mathbb{F}_q}(C) = k$ and $d(C) = d$. The (Euclidean) dual code $C^\perp$ of C is the orthogonal to C in $\mathbb{F}_q^n$, i.e.,

$$C^\perp = \{d \in \mathbb{F}_q^n : \langle c, d\rangle = 0\},$$

where $\langle\ ,\rangle$ denotes the usual Euclidean inner product $\langle c, d\rangle = c_0d_0 + c_1d_1 + \cdots + c_{n-1}d_{n-1}$ in $\mathbb{F}_q^n$. Clearly, $C^\perp$ is an $[n, n-k, d^\perp]_q$ code over $\mathbb{F}_q$ such that $d^\perp = d(C^\perp)$. A $k \times n$ matrix over $\mathbb{F}_q$ is called the generator matrix of C if its rows form a basis of C. An $(n-k) \times n$ matrix H is said to be a parity-check matrix of C if we have $GH^\top = 0$ and the rows of H form a basis of $C^\perp$ if H has full rank. In fact, the minimum distance d of C can be characterized by the columns of H as follows.

Theorem 2.1. *[15, Corollary 1.4.14] A linear code C has minimum distance d if and only if any $d-1$ columns of its parity check matrix H are linearly independent but H has a set of d linearly dependent columns.*

The following lower and upper bounds on the size of a given linear code will be used

in the next section.

Theorem 2.2.

1. *[15, Theorem 1.12.1] (Hamming Bound/Sphere Packing Bound) For an $[n,k,d]_q$ linear code C over $\mathbb{F}_q$, we have*

$$|C| \leq \frac{q^n}{\sum_{i=0}^{\lfloor \frac{d-1}{2} \rfloor} (q-1)^i \binom{n}{i}}$$

2. *[15, Theorem 2.2.1](Plotkin Bound) For an $[n,k,d]_q$ linear code C over $\mathbb{F}_q$, such that $rn < d$ where $r = 1 - q^{-1}$ we have*

$$|C| \leq \left\lfloor \frac{d}{d - rn} \right\rfloor$$

3. *[15, Theorem 2.4.1](Singleton Bound) For an $[n,k,d]_q$ linear code C over $\mathbb{F}_q$, we have*

$$|C| \leq q^{n-d+1}$$

   If $|C| = q^{n-d+1}$, then C is said to be maximum distance separable (MDS).

4. *[15, Theorem 2.5.3](Elias Bound) For an $[n,k,d]_q$ linear code C over $\mathbb{F}_q$. Let $r = 1 - q^{-1}$. Suppose that $w \leq rn$ and $w^2 - 2rnw + rnd > 0$, we have*

$$|C| \leq \frac{rnd}{w^2 - 2rnw + rnd} \cdot \frac{q^n}{\sum_{i=0}^{w} \binom{n}{i} (q-1)^i}$$

5. *[15, Theorem 2.8.1](Gilbert Lower Bound/Sphere Covering Bound) For an $[n,k,d]_q$ linear code C over $\mathbb{F}_q$, we have*

$$|C| \geq \frac{q^n}{\sum_{i=0}^{d-1} (q-1)^i \binom{n}{i}}.$$

6. *[15, Theorem 2.9.3](Varshamov Lower Bound) For an $[n,k,d]_q$ linear code C over $\mathbb{F}_q$, we have $|C| \geq q^{n-M}$, where*

$$M = \left\lceil \log_q \left(1 + \sum_{i=0}^{d-2} (q-1)^i \binom{n-1}{i} \right) \right\rceil.$$

A linear code C of length n over $\mathbb{F}_q$ is called linear complementary dual (LCD) if $C \cap C^{\perp} = \{0\}$. Since $(C^{\perp})^{\perp} = C$, $C^{\perp}$ is also LCD and we have $\mathbb{F}_q^n = C \oplus C^{\perp}$. The following characterization of LCD codes is due to Massey [21].

Theorem 2.3. *[21, Proposition 1] Let G and H be a generator matrix and a parity-check matrix of the given code $C \subseteq \mathbb{F}_q^n$, respectively. Then the following properties are equivalent:*

1. *C is LCD;*
2. *$C^{\perp}$ is LCD;*
3. *$G\,G^{\top}$ is nonsingular;*
4. *$H\,H^{\top}$ is nonsingular.*

For an $\mathbb{F}_Q$–linear code $C \subseteq \mathbb{F}_Q^{\ell}$, where Q is an even power of a prime p, we denote by $\overline{C}$ the *conjugate* code

$$\overline{C} = \big\{\overline{\mathbf{c}} \,:\, \mathbf{c} \in C\big\},$$

where the overbar on each coordinate c_i denotes the usual field involution (i.e. raising to the power $\sqrt{Q}$). Given $x, y \in \mathbb{F}_Q^n$, we define their Hermitian inner product over $\mathbb{F}_Q$ by

$$\langle x, y\rangle_h = x_0\overline{y}_0 + x_1\overline{y}_1 + \cdots + x_{n-1}\overline{y}_{n-1} = x_0 y_0^{\sqrt{Q}} + x_1 y_1^{\sqrt{Q}} + \cdots + x_{n-1} y_{n-1}^{\sqrt{Q}}.$$

Using this notation, the Hermitian dual $C^{\perp_h}$ of a given code C is defined by $C^{\perp_h} = \{d \in \mathbb{F}_q^n : \langle c, d\rangle_h = 0\}$. It is straightforward to observe that

$$C^{\perp_h} = (\overline{C})^{\perp_e},$$

where $C^{\perp_h}$ is the *Hermitian dual* of C and $C^{\perp_e}$ is the *Euclidean dual* of C.

Definition 2.4. If C is an $[n, k]$ linear code over $\mathbb{F}_q$ with a generating matrix G, then its conjugate $\overline{C}$ is also an $[n, k]$ linear code over $\mathbb{F}_q$ with generating matrix $\overline{G}$. Moreover, C is *Hermitian LCD* if and only if

$$C \cap C^{\perp_h} = \{0\}.$$

In other words,

$$\mathbb{F}_q^{\ell} = C \oplus C^{\perp_h}.$$

The Hermitian version of Theorem 2.3 is stated and proven in [12].

Proposition 2.5. *[12, Proposition 3.5] A linear code C is a Hermitian LCD code if and only if $G\,\overline{G}^{\top}$ is nonsingular.*

Now we will provide the necessary background for the linear programming bound.

Definition 2.6. Let C be a q-ary $[n, k, d]$ linear code and let $C^{\perp}$ be its dual code with parameters $[n, n-k, d^{\perp}]$. The weight distribution of C is the sequence $\{A_i\}_{i=0}^{n}$, where

$$A_i = \Big|\{\, c \in C : \mathrm{wt}(c) = i\}\Big|.$$

Similarly, the weight distribution of $C^{\perp}$ is the sequence $\{B_i\}_{i=0}^{n}$, where

$$B_i = \Big|\{\, c' \in C^{\perp} : \mathrm{wt}(c') = i\}\Big|.$$

Remark 2.7. Since the zero codeword is the only codeword of weight zero, we have

$$A_0 = 1 \quad \text{and} \quad B_0 = 1.$$

Moreover, because the minimum distance of C is d, there are no nonzero codewords of weight less than d; that is,

$$A_i = 0 \quad \text{for } 1 \le i < d.$$

Similarly, for the dual code $C^{\perp}$ with minimum distance $d^{\perp}$, it follows that

$$B_j = 0 \quad \text{for } 1 \le j < d^{\perp}.$$

Definition 2.8. Given two positive integers n and q, the Krawtchouk polynomial of degree j is defined as

$$K_j^{n,q}(x) := \sum_{k=0}^{j} (-1)^k (q-1)^{j-k} \binom{x}{k} \binom{n-x}{j-k} \quad \text{for } 0 \le j \le n.$$

MacWilliams used Krawtchouk polynomials to relate the weight distribution $\{A_i\}_{i=0}^{n}$ of a given binary code and the weight distribution $\{B_i\}_{i=0}^{n}$ of its dual code. We will prove a p-ary version of the following identity in Lemma 3.3.

$$B_j = \frac{1}{|C|} \sum_{i=0}^{n} A_i K_j^{n,q}(i) \tag{2.1}$$

Theorem 2.9. *[15, Theorem 2.6.4](Delsarte's Linear Programming Bound) Given an* $[n, k, d]_q$ *code* C *over* $\mathbb{F}_q$ *with weight distribution* $\{A_i\}_{i=0}^n$*, the linear programming bound for the size of* C *is given as*

$$|C| \leq \max\left\{\sum_{i=0}^{n} A_i\right\}$$

subject to the following conditions:

- $A_0 = 1$,
- $A_i = 0$, *for* $0 < i < d$,
- $A_i \geq 0$, *for* $d \leq i \leq n$,
- $\sum_{i=0}^{n} A_i K_j^{n,q}(i) \geq 0$ *for* $0 \leq j \leq n$ *by* (2.1).

CHAPTER 3

EXTENDED LINEAR PROGRAMMING BOUND

In order to state and prove the LP bound for q-ary LCD codes, we need the well-known MacWilliams identities in the q-ary case. Since we mainly focus on binary and ternary LCD codes, we will derive the MacWilliams identities in the p-ary case for completeness.

Lemma 3.1. *Let C be linear code of length n and dimension k over $\mathbb{F}_p$. Let u, $v \in \mathbb{F}_p$. Let ξ be a p^{th} root of unity. Then,*

$$\sum_{u \in C} \xi^{\langle u,v \rangle} = \begin{cases} p^k, & v \in C^{\perp}, \\ 0, & v \notin C^{\perp}. \end{cases} \tag{3.1}$$

Proof. If $v \in C^{\perp}$, then we have $\langle u, v \rangle = 0$. This implies $\sum_{u \in C} \xi^{\langle u,v \rangle} = \sum_{u \in C} 1 = p^k$.

If $v \notin C^{\perp}$, then there exists a nonzero codeword $u_0 \in C$ such that $\langle u_0, v \rangle = a \neq 0$ and therefore $\xi^{\langle u_0,v \rangle} = \xi^a$. For an arbitrary codeword $u \in C$, if $\langle u, v \rangle = 0$, then we have $\langle (u + u_0), v \rangle = a$. Otherwise, if $\langle (u + u_0), v \rangle = 0$, then we set $w := u + u_0$ and we have $\langle (w + u_0), v \rangle = a$. Let $C = C_1 + \text{span}_{\mathbb{F}_p}(u_0)$. We can rewrite Lemma 3.1 as

$$\sum_{u \in C} \xi^{\langle u,v \rangle} = \sum_{u_1 \in C_1} \sum_{\alpha \in \mathbb{F}_p} \xi^{\langle u_1 + \alpha u_0, v \rangle} = \sum_{u_1 \in C_1} \sum_{\alpha \in \mathbb{F}_p} \xi^{a\alpha} = \sum_{u_1 \in C_1} \frac{(\xi^a)^p - 1}{\xi^a - 1} = 0. \tag{3.2}$$

□

Let $f : \mathbb{F}_q \to \mathbb{Z}_q$ any bijection with $f(0) = 0 \in \mathbb{Z}_q$. If $\mathbf{c} = (c_0, c_1, \ldots, c_{n-1}) \in \mathbb{F}_q^n$

and $\mathbf{v} := (f(c_0), f(c_1), \ldots f(c_{n-1})) \in \mathbb{Z}_q^n$, then we have $wt(\mathbf{c}) = wt(\mathbf{v})$. Also, one can extend the Euclidean inner product $\langle , \rangle$ to n tuples over $\mathbb{Z}_q$.

Lemma 3.2. *[15, Lemma 2.6.2] Let ξ be a q^{th} root of unity and let $u' \in \mathbb{Z}_q^n$ such that $wt(u') = i$. Then,*

$$\sum_{\substack{v' \in \mathbb{Z}_q^n \\ wt(v')=j}} \xi^{\langle u', v' \rangle} = K_j^{n,q}(i). \tag{3.3}$$

Now we are ready to state and prove the MacWilliams identities for a p-ary linear code C and its dual.

Lemma 3.3 (MacWilliams)**.** *Let f be the identity map when $q = p$, so that $u = u'$ and $v = v'$. Given an $[n, k]$ code C over $\mathbb{F}_p$, let A_i and B_i be the weight distributions of C and $C^\perp$, respectively. Then, for any $j \in \{0, 1, \ldots, n\}$, we have*

$$B_j = \frac{1}{p^k} \sum_{i=0}^{n} A_i K_j^{n,p}(i). \tag{3.4}$$

Proof. For $j \in \{0, 1, \ldots, n\}$, we set $V_j := \{v \in \mathbb{F}_p^n : wt(v) = j\}$. Then, by Lemma 3.1, we have

$$\sum_{v \in V_j} \sum_{u \in C} \xi^{\langle u, v \rangle} = \sum_{v \in V_j \cap C^\perp} p^k = p^k \cdot B_j. \tag{3.5}$$

On the other hand, Lemma 3.2 implies

$$\sum_{u \in C} \sum_{v \in V_j} \xi^{\langle u, v \rangle} = \sum_{\substack{u \in C \\ wt(u)=i}} K_j^{n,p}(i) = \sum_{i=0}^{n} A_i K_j^{n,p}(i). \tag{3.6}$$

Since the left-hand sides of eqn. (3.5) and (3.6) are identical, we obtain

$$\begin{aligned} p^k \cdot B_j &= \sum_{i=0}^{n} A_i K_j^{n,p}(i) \\ \implies B_j &= \frac{1}{p^k} \sum_{i=0}^{n} A_i K_j^{n,p}(i). \end{aligned}$$

□

Note that, the generating function of the Krawtchouk polynomial of degree j is related with the given formula

$$\sum_{j=0}^{n} K_j^{n,q}(i) z^j = (1 - z)^i (1 + (q - 1)z)^{n-i}.$$

The general MacWilliams identities in the q-ary case are given in terms of this generating function in Theorem 1 in [20].

The following result extends the LP bound in [9], which was given for binary LCD codes.

Theorem 3.4 (Extended LP)**.** *Let $A_{LCD}(q;n,d)$ denote the maximal size of a q-ary LCD code of length n and minimum distance d. If $\{A_0, A_1, \ldots, A_n\}$ is the weight distribution of C, then*

$$A_{LCD}(q;n,d) \leq \max\left\{\sum_{i=0}^{n} A_i\right\},$$

where the weight distribution of C is subject to the following conditions:

i) $A_0 = 1$, $A_1 = \cdots = A_{d-1} = 0$ *and* $A_i \geq 0$, *for* $d \leq i \leq n$.

ii) $\sum_{i=1}^{n} A_i K_j^{n,q}(i) \geq -(q-1)^j \binom{n}{j}$, *for* $0 \leq j \leq n$.

iii) $q^{k_0} A_j \leq \sum_{i=1}^{n} A_i \left[(q-1)^j \binom{n}{j} - K_j^{n,q}(i)\right]$, *for any fixed* $k_0 \leq k$, *where* $q^k = \sum_{i=0}^{n} A_i$.

Proof. The first item is straightforward. From eqn. (3.4), we obtain $\frac{1}{q^k}\sum_{i=0}^{n} A_i K_j^{n,q}(i) \geq 0$, since each $B_j \geq 0$, for $j \in \{0,1,\ldots,n\}$. This implies

$$\sum_{i=0}^{n} A_i K_j^{n,q}(i) \geq 0. \tag{3.7}$$

By substituting $i = 0$ in Definition 2.8, we get

$$K_j^{n,q}(0) = (q-1)^j \binom{n}{j}. \tag{3.8}$$

We now combine eqn. (3.7) with eqn. (3.8) to obtain item ii) as follows.

$$\begin{aligned}
\sum_{i=0}^{n} A_i K_j^{n,q}(i) = A_0 K_j^{n,q}(0) + \sum_{i=1}^{n} A_i K_j^{n,q}(i) &\geq 0 \\
\implies \sum_{i=1}^{n} A_i K_j^{n,q}(i) &\geq -A_0 K_j^{n,q}(0) \\
\implies \sum_{i=1}^{n} A_i K_j^{n,q}(i) &\geq -(q-1)^j \binom{n}{j}.
\end{aligned}$$

To prove item iii), we will use the following fact that LCD codes must satisfy:

$$A_j + B_j \leq (q-1)^j \binom{n}{j}, \tag{3.9}$$

for any $j \in \{0, 1, \ldots, n\}$. By substituting eqn. (3.4) in eqn. (3.9) above, we get

$$\begin{aligned}
A_j + \frac{1}{q^k} \sum_{i=0}^{n} A_i K_j^{n,q}(i) &\leq (q-1)^j \binom{n}{j} \\
q^k A_j + \sum_{i=0}^{n} A_i K_j^{n,q}(i) &\leq q^k (q-1)^j \binom{n}{j} \\
q^k A_j &\leq \sum_{i=0}^{n} A_i (q-1)^j \binom{n}{j} - \sum_{i=0}^{n} A_i K_j^{n,q}(i) \\
q^k A_j &\leq \sum_{i=0}^{n} A_i \left[(q-1)^j \binom{n}{j} - K_j^{n,q}(i) \right] \\
q^{k_0} A_j &\leq \sum_{i=1}^{n} A_i \left[(q-1)^j \binom{n}{j} - K_j^{n,q}(i) \right],
\end{aligned}$$

where the last inequality follows from $K_j^{n,q}(0) = (q-1)^j \binom{n}{j}$. Moreover, the fixed value k_0 can be chosen from a known lower bound on the dimension k of C such as Varshamov lower bound given in Theorem 2.2 item 6 or Gilbert lower bound given in Theorem 2.2 item 5.

□

The following theorem improves Theorem 3.4 by introducing extra bounds on A_i and B_j.

Theorem 3.5 (Improved LCD LP)**.** *Let $A_{LCD}(q; n, d)$ denote the maximal size of a q-ary LCD code C of length n and minimum distance d and let $d_0^{\perp}$ denote the possible minimum distance distance of its dual code. If $\{A_0, A_1, \ldots, A_n\}$ is the weight distribution of C, then*

$$A_{LCD}(q; n, d) \leq \max \left\{ \sum_{i=0}^{n} A_i \right\},$$

where the weight distribution of C is subject to the following conditions:

$$\textit{(i)} \quad A_0 = 1, \quad A_1 = \cdots = A_{d-1} = 0, \quad A_i \geq 0 \quad \textit{for } d \leq i \leq n,$$

$$\textit{(ii)} \quad \sum_{i=d}^{n} A_i \, K_j^{n,q}(i) = -(q-1)^j \binom{n}{j} \quad \textit{for } 1 \leq j < d_0^{\perp},$$

$$\textit{(iii)} \quad \sum_{i=d}^{n} A_i \, K_j^{n,q}(i) \geq -(q-1)^j \binom{n}{j} \quad \textit{for } d_0^{\perp} \leq j \leq n,$$

$$\textit{(iv)} \quad A_j \leq (q-1)^j \binom{n}{j} \quad \textit{for } d \leq j < d_0^{\perp} \textit{ when } \; d_0^{\perp} > d,$$

$$\textit{(v)} \quad q^{k_0} A_j \leq \sum_{i=d}^{n} A_i \Big[(q-1)^j \binom{n}{j} - K_j^{n,q}(i) \Big] \quad \textit{for any fixed } k_0 \leq k \textit{ and } \max(d_0^{\perp}, d) \leq j \leq n.$$

Proof. The proof comes from the fact that $B_j = 0$ where $j \in \{0, 1 \cdots, d_0^{\perp}\}$ and using eqn. (3.4) since $d_0^{\perp} \leq d^{\perp}$. To prove item ii)

$$\begin{aligned}
\frac{1}{q^k} \sum_{i=0}^{n} A_i K_j^{n,q}(i) &= 0 \\
\sum_{i=1}^{n} A_i K_j^{n,q}(i) &= K_j^{n,q}(0) \\
\sum_{i=1}^{n} A_i K_j^{n,q}(i) &= -(q-1)^j \binom{n}{j}, \quad \text{from 3.8} \\
\sum_{i=d}^{n} A_i K_j^{n,q}(i) &= -(q-1)^j \binom{n}{j}, \quad \text{since } A_i = 0 \text{ when } i < d
\end{aligned}$$

For item iii) and v), it is also trivial that we set the summation's lower bound of item ii) and iii) in Theorem 3.4 to d since $A_i = 0$ when $i < d$.
For item iv), replace B_j in eqn. (3.9) with 0 when $j < d_0^{\perp}$

□

Remark 3.6. Note that, $d_0^{\perp}$ can be found using a lower bound as follows,

- Find k_0 using a lower bound for the parameters n and d.
- Estimate an upper bound for $C^{\perp}$ as $k_{up}^{\perp} = n - k_0$, it is logical since $d_0^{\perp}$ and $k_{up}^{\perp}$ are inversely related.
- Use the Gilbert Lower Bound to derive a lower bound on the minimum distance of the dual code $d_0^{\perp}$ as $\sum_{i=0}^{d_0^{\perp}-1} (q-1)^i \binom{n}{i} = \frac{q^n}{q^{k_{up}^{\perp}}} = q^{k_0}$.

In the LP program, we have tested Theorem 3.5. Although it gives a slightly better tightened LP bound it increases the LP program run time significantly, so we didn't use it. However, Theorem 3.5 might yield better result larger code lengths.

3.1 Linear Programming Bound Program

Since our linear programming formulation operates on linear codes, each A_i for $i \in \{0, \ldots, n\}$ must take on integer values. Consequently, the problem becomes an instance of integer linear programming (ILP). To solve the ILP, we employ Gurobi Optimization's Mixed Integer Solver modules, with the implementation written in Python. Moreover, if a value is already known or if the computed upper and lower bounds coincide, the corresponding instance is skipped. For selecting the constant k_0, we utilize the Varshamov Lower Bound together with the Gilbert Lower Bound.

Table 3.1: ILP configuration parameters used in Gurobi Solver.

Initial Settings
Model = ILP with Maximization
Variable Definition = integer, lower bound as 0
Upper bound of $A_i = 0$ for $i \in \{1, \ldots, d-1\}$
Upper bound of $A_i = (q-1)^i \binom{n}{i}$ for $i \in \{d, \ldots, n\}$
Integer Feasibility Tolerance = 1×10^{-6}
Feasibility Tolerance = 10^{-3}
Optimality Tolerance = 10^{-3}
MIP Gap = 10^{-1}
Quad = 1
ObjScale = -0.75
NumericFocus = 3

To compute the extended LP bound for LCD codes, we initially set the Gurobi parameters as detailed in Table 3.1. In practice, however, these initial settings do not always yield a feasible or optimal solution. Therefore, we implemented an iterative trial procedure in which solver parameters are systematically relaxed or modified whenever the model fails to meet feasibility conditions or when the computed bound does not satisfy our criteria.

Our ILP solver is executed within a loop that performs up to 10 trials, each with a 4-hours timeout. After each trial, the model is solved and its feasibility is assessed by

checking the computed bound against known upper and lower bounds. If the model is feasible and produces a valid bound, the solution is accepted; otherwise, the following parameter adjustments are made:

- **Trial 0:** Increase the `MIPFocus` parameter to 2.
- **Trial 1:** Set `MIPFocus` to 1 and adjust `ObjScale` to -0.5.
- **Trial 2:** Relax the integrality feasibility tolerance (`IntFeasTol`) to 1×10^{-4}, adjust `ObjScale` to -0.4, set `MIPFocus` back to 2, and increase `NumericFocus` to 2.
- **Trial 3:** Further relax the integrality tolerance to 1×10^{-2} and relax both the feasibility (`FeasibilityTol`) and optimality (`OptimalityTol`) tolerances to 1×10^{-2}; simultaneously, adjust `ObjScale` to -0.25 and reduce `NumericFocus` to 1.
- **Trial 4:** Increase the integrality tolerance to 1×10^{-1}, set `ObjScale` to -0.1, and switch `MIPFocus` to 1.
- **Trial 5:** Maintain `IntFeasTol` $= 1 \times 10^{-1}$, adjust `ObjScale` to 0, and keep `MIPFocus` at 1.
- **Trial 6:** Retain the integrality tolerance at 1×10^{-1} and `ObjScale` at -0.1, but change `MIPFocus` to 2.
- **Trial 7:** Continue with `IntFeasTol` $= 1 \times 10^{-1}$ and `ObjScale` at -0.1 while increasing `MIPFocus` to 3.
- **Trial 8:** Reduce the integrality tolerance to 1×10^{-2} while keeping `ObjScale` at -0.1.
- **Trial 9:** Finally, relax the integrality tolerance further to 1×10^{-3} and reset `ObjScale` to -0.25.

These parameter adjustments are designed to balance the numerical precision required for feasibility with the computational efficiency of the solver. Relaxing the integrality and feasibility tolerances helps overcome numerical difficulties arising from a wide

range of coefficient values, while adjustments to the objective scaling and MIP focus parameters guide the branch-and-bound process more effectively toward feasible regions.

By iteratively relaxing and fine-tuning these settings, our approach increases the likelihood of obtaining a feasible and tight bound for the LCD codes under investigation. This trial-based strategy is essential for addressing the inherent complexity of the ILP formulation and has proven effective in our computational experiments.

3.2 LP Tables

In the table, an asterisk ($*$) indicates skipped values, and an upward arrow ($\uparrow$) marks cases where the LP program was unable to determine a tighter upper bound than known upper bounds. For the algebraic upper bound calculation, the minimum of the Elias, sphere packing, Plotkin, and Singleton upper bounds is used, from the SageMath's library [23].

Table 3.2: Binary LCD Upper Bounds on Dimension for $2 \leq n \leq 30$ and $1 \leq d \leq 30$

$n\backslash d$	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30
2	2^*	0^*																												
3	3^*	2^*	1^*																											
4	4^*	2^*	1^*	0^*																										
5	5^*	4^*	2^*	1^*	1^*																									
6	6^*	4^*	3	2	1^*	0^*																								
7	7^*	6^*	4	2^*	1^*	1^*	1^*																							
8	8^*	6^*	4	3	2^*	1^*	1^*	0^*																						
9	9^*	8^*	5^*	3^*	2	2^*	1^*	1^*	1^*																					
10	10^*	8^*	6	5	3^*	2^*	1^*	1^*	1^*	0^*																				
11	11^*	10^*	7^*	5^*	3^*	2^*	2	1^*	1^*	1^*	1^*																			
12	12^*	10^*	8^*	5^*	5	3^*	2^*	2	1^*	1^*	1^*	0^*																		
13	13^*	12^*	9^*	7	5^*	3^*	3	2^*	1^*	1^*	1^*	1^*	1^*																	
14	14^*	12^*	10^*	9	5^*	6	3^*	3	2^*	1^*	1^*	1^*	1^*	0^*																
15	15^*	14^*	11^*	10	8	5^*	3^*	4	2	2^*	1^*	1^*	1^*	1^*	1^*																
16	16^*	14^*	11^*	10^*	8^*	6^*	5	3^*	2	2^*	1^*	1^*	1^*	1^*	1^*	0^*														
17	17^*	16^*	12^*	10^*	9^*	8^*	5^*	4^*	3^*	2^*	2	1^*	1^*	1^*	1^*	1^*	1^*														
18	18^*	16^*	13^*	12^*	9^*	8^*	6^*	4^*	3^*	3	2^*	2	1^*	1^*	1^*	1^*	1^*	0^*												
19	19^*	18^*	14^*	12^*	10^*	9^*	7^*	6^*	4^*	3^*	2	2^*	1^*	1^*	1^*	1^*	1^*	1^*	1^*												
20	20^*	18^*	15^*	14^*	11^*	10^*	7^*	6^*	5^*	4^*	3	2	2^*	1^*	1^*	1^*	1^*	1^*	1^*	0^*										
21	21^*	20^*	16^*	14^*	12^*	10^*	8^*	7^*	5^*	4^*	3^*	3	2	2^*	1^*	1^*	1^*	1^*	1^*	1^*	1^*									
22	22^*	20^*	17^*	16^*	13^*	12^*	9^*	8^*	6^*	5^*	3^*	3	2	2^*	1^*	1^*	1^*	1^*	1^*	1^*	1^*	0^*								
23	23^*	22^*	18^*	16^*	13^*	12^*	10^*	8^*	7^*	6^*	4^*	3^*	2	2^*	2	1^*	1^*	1^*	1^*	1^*	1^*	1^*	1^*							
24	24^*	22^*	19^*	18^*	14^*	13^*	11^*	10^*	7^*	6^*	5^*	4^*	3^*	2	2^*	2	1^*	1^*	1^*	1^*	1^*	1^*	1^*	1^*	0^*					
25	25^*	24^*	20^*	18^*	15^*	14^*	12^*	10^*	8^*	7^*	5^*	4^*	3^*	3	2	2^*	1^*	1^*	1^*	1^*	1^*	1^*	1^*	1^*	1^*	1^*				
26	26^*	24^*	21^*	20^*	16^*	14^*	13^*	12^*	9^*	8^*	6^*	5^*	4	3^*	2	2	2^*	1^*	1^*	1^*	1^*	1^*	1^*	1^*	1^*	1^*	0^*			
27	27^*	26^*	22	20^*	17^*	16^*	13^*	12^*	10^*	8^*	7^*	6^*	4^*	3^*	3	2	2	2^*	1^*	1^*	1^*	1^*	1^*	1^*	1^*	1^*	1^*	1^*		
28	28^*	26^*	23	21^*	18^*	16^*	14^*	13^*	12	10^*	7^*	6^*	5^*	4^*	3^*	3	2	2^*	1^*	1^*	1^*	1^*	1^*	1^*	1^*	1^*	1^*	1^*	0^*	
29	29^*	28^*	24	22^*	19^*	18^*	16	14^*	13	10^*	8^*	7^*	5^*	4^*	3^*	3	2	2^*	2	1^*	1^*	1^*	1^*	1^*	1^*	1^*	1^*	1^*	1^*	
30	30^*	28^*	25	23^*	20^*	18^*	17	14^*	14	10^*	9^*	8^*	6^*	5^*	5	3^*	2	2	2^*	2	1^*	1^*	1^*	1^*	1^*	1^*	1^*	1^*	1^*	0^*

Table 3.3: Binary LCD Upper Bounds on Dimension for $31 \leq n \leq 60$ and $1 \leq d \leq 30$

$n\backslash d$	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30
31	31*	30*	26	24*	21	20*	18	14*	15	10*	9*	8*	7*	6*	4*	5	3*	2	2	2*	1*	1*	1*	1*	1*	1*	1*	1*	1*	1*
32	32*	30*	26*	24*	22	20*	19	18	16	11*	10*	9*	7*	6*	5*	4*	3*	3	2	2	2*	1*	1*	1*	1*	1*	1*	1*	1*	1*
33	33*	32*	27*	26*	23	22*	20	19	17	12*	12	10*	8*	7*	5*	4*	4	3*	2	2	2	2*	1*	1*	1*	1*	1*	1*	1*	1*
34	34*	32*	28*	26*	24	22*	21	20	17	17	12*	10*	10	8*	6*	5*	4*	3*	3	2	2	2*	1*	1*	1*	1*	1*	1*	1*	1*
35	35*	34*	29*	28*	25	24	21	21	18	14*	15	12*	11	8*	7*	6*	6	4*	3*	3	2	2*	2	1*	1*	1*	1*	1*	1*	1*
36	36*	34*	30*	28*	26	23*	22	21	19	18	16	12*	11	8*	7*	6*	5*	4*	3*	3	2	2	2*	2	1*	1*	1*	1*	1*	1*
37	37*	36*	31*	30*	27	26	23	22	20	16*	17	16	13	12	8*	7*	5*	4*	4	3*	2	2	2	2*	1*	1*	1*	1*	1*	1*
38	38*	36*	32*	30*	28	27	24	23	21	20	17	17	14	10*	11	8*	6*	5*	4*	4	3*	2	2	2	2*	1*	1*	1*	1*	1*
39	39*	38*	33*	32*	29	28	25	24	22	21	18	17	15	14	11	8*	7*	6*	6	4*	3*	3	2	2	2	2*	1*	1*	1*	1*
40	40*	38*	34*	32*	30	29	26	25	23	22	19	18	15	14	11	8*	7*	6*	5*	4*	3	3*	2	2	2	2*	1*	1*	1*	1*
41	41*	40*	35*	34*	31	28*	27	23*	23	20*	20	19	16	15	13	9*	8*	7*	6*	4*	4	3*	3	2	2	2*	2	1*	1*	1*
42	42*	40*	36*	34*	32	29*	28	24*	24	20*	21	20	17	16	14	10*	9*	8*	7	6*	4*	4	3*	3	2	2	2*	2	1*	1*
43	43*	42*	37*	36*	32	30*	29	28	25	21*	22	21	18	14*	15	13	11	8*	7*	6*	6	4*	3*	3	2	2	2	2*	1*	1*
44	44*	42*	38*	36*	33	30*	30	26*	26	25	23	22	19	18	15	15	12	9*	9	6*	5*	4*	4	3*	2	2	2	2	2*	1*
45	45*	44*	39*	38*	34	32*	30	30	27	26	24	23	20	16*	16	14	13	12	10	7*	5*	4*	4	4	3*	2	2	2	2	2*
46	46*	44*	40*	38*	35	32*	31	28*	28	27	25	24	21	20	17	16	14	13	11	7*	6*	5*	4*	4	3*	3	2	2	2	2*
47	47*	46*	41*	40*	36	34*	32	31	29	28	25	25	22	21	18	17	15	14	11	11	8	6*	6	4*	3	3*	2	2	2	2*
48	48*	46*	42*	40*	37	34*	33	30*	30	29	26	25	23	22	19	18	15	15	11	10	8	6*	5*	4*	4	3*	3	2	2	2
49	49*	48*	43*	42*	38	36*	34	30*	30	30	27	26	23	23	20	19	16	15	13	12	10	9	6*	7	4*	4	3*	3	2	2
50	50*	48*	44*	42*	39	36*	35	31*	31	30	28	27	24	23	21	20	17	16	14	13	10	10	8	6*	5	4*	3*	3	2	2
51	51*	50*	45*	44*	40	39	36	35	32	31	29	28	25	24	22	21	18	17	15	14	11	10	8	8	5*	4*	3	3*	2	2
52	52*	50*	46*	45	41	40	36	36	33	32	30	29	26	25	23	22	19	18	16	15	12	11	9	8	5*	4*	4	3	3*	2
53	53*	52*	47*	46*	41	41	38	37	34	33	31	30	27	23	23	23	20	19	16	16	13	12	9	9	7	5*	4*	4	3*	3
54	54*	52*	48*	47	43	42	39	38	35	34	31	31	28	27	24	23	20	20	17	16	13	13	10	10	8	5*	5	4*	3	3*
55	55*	54*	49*	48*	43	43	40	39	36	35	32	31	29	28	25	24	21	20	18	17	14	13	11	10	8	8	5*	4*	3	3*
56	56*	54*	50*	49	45	44	41	40	37	36	33	32	30	29	26	25	22	21	19	18	15	14	12	11	9	8	5*	4*	4	3
57	57*	56*	51*	50*	46	45	41	40	38	37	34	33	31	30	27	26	23	22	20	19	16	15	13	12	9	9	7	5*	4*	4
58	58*	56*	52*	51	47	46	43	41	39	38	35	34	32	31	28	27	24	23	20	20	17	16	13	13	10	9	8	5*	5	4*
59	59*	58*	53*	52	48	47	43	42	39	39	36	35	32	32	29	28	25	24	21	20	18	17	14	13	11	10	8	8	5*	4*
60	60*	58*	54*	51	49	48	44	43	40	39	37	36	33	32	29	29	26	25	22	21	19	18	15	14	12	11	9	8	5*	4*

Table 3.4: Binary LCD Upper Bounds on Dimension for $31 \leq n \leq 60$ and $31 \leq d \leq 60$

$n\backslash d$	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	51	52	53	54	55	56	57	58	59	60
31	1^*																													
32	1^*	0^*																												
33	1^*	1^*	1^*																											
34	1^*	1^*	1^*	0^*																										
35	1^*	1^*	1^*	1^*	1^*																									
36	1^*	1^*	1^*	1^*	1^*	0^*																								
37	1^*	1^*	1^*	1^*	1^*	1^*	1^*																							
38	1^*	1^*	1^*	1^*	1^*	1^*	1^*	0^*																						
39	1^*	1^*	1^*	1^*	1^*	1^*	1^*	1^*	1^*																					
40	1^*	1^*	1^*	1^*	1^*	1^*	1^*	1^*	1^*	0^*																				
41	1^*	1^*	1^*	1^*	1^*	1^*	1^*	1^*	1^*	1^*	1^*																			
42	1^*	1^*	1^*	1^*	1^*	1^*	1^*	1^*	1^*	1^*	1^*	0^*																		
43	1^*	1^*	1^*	1^*	1^*	1^*	1^*	1^*	1^*	1^*	1^*	1^*	1^*																	
44	1^*	1^*	1^*	1^*	1^*	1^*	1^*	1^*	1^*	1^*	1^*	1^*	1^*	0^*																
45	1^*	1^*	1^*	1^*	1^*	1^*	1^*	1^*	1^*	1^*	1^*	1^*	1^*	1^*	1^*															
46	1^*	1^*	1^*	1^*	1^*	1^*	1^*	1^*	1^*	1^*	1^*	1^*	1^*	1^*	1^*	0^*														
47	2	1^*	1^*	1^*	1^*	1^*	1^*	1^*	1^*	1^*	1^*	1^*	1^*	1^*	1^*	1^*	1^*													
48	2^*	2	1^*	1^*	1^*	1^*	1^*	1^*	1^*	1^*	1^*	1^*	1^*	1^*	1^*	1^*	1^*	0^*												
49	2	2^*	1^*	1^*	1^*	1^*	1^*	1^*	1^*	1^*	1^*	1^*	1^*	1^*	1^*	1^*	1^*	1^*	1^*											
50	2	2	2^*	1^*	1^*	1^*	1^*	1^*	1^*	1^*	1^*	1^*	1^*	1^*	1^*	1^*	1^*	1^*	1^*	0^*										
51	2	2	2	2^*	1^*	1^*	1^*	1^*	1^*	1^*	1^*	1^*	1^*	1^*	1^*	1^*	1^*	1^*	1^*	1^*	1^*									
52	2	2	2	2^*	1^*	1^*	1^*	1^*	1^*	1^*	1^*	1^*	1^*	1^*	1^*	1^*	1^*	1^*	1^*	1^*	1^*	0^*								
53	2	2	2	2^*	2	1^*	1^*	1^*	1^*	1^*	1^*	1^*	1^*	1^*	1^*	1^*	1^*	1^*	1^*	1^*	1^*	1^*	1^*							
54	2	2	2	2	2^*	2	1^*	1^*	1^*	1^*	1^*	1^*	1^*	1^*	1^*	1^*	1^*	1^*	1^*	1^*	1^*	1^*	1^*	0^*						
55	3	2	2	2	2	2^*	1^*	1^*	1^*	1^*	1^*	1^*	1^*	1^*	1^*	1^*	1^*	1^*	1^*	1^*	1^*	1^*	1^*	1^*	1^*					
56	3^*	3	2	2	2	2	2^*	1^*	1^*	1^*	1^*	1^*	1^*	1^*	1^*	1^*	1^*	1^*	1^*	1^*	1^*	1^*	1^*	1^*	1^*	0^*				
57	3^*	3	2	2	2	2	2	2^*	1^*	1^*	1^*	1^*	1^*	1^*	1^*	1^*	1^*	1^*	1^*	1^*	1^*	1^*	1^*	1^*	1^*	1^*	1^*			
58	3	3^*	2	2	2	2	2	2^*	1^*	1^*	1^*	1^*	1^*	1^*	1^*	1^*	1^*	1^*	1^*	1^*	1^*	1^*	1^*	1^*	1^*	1^*	1^*	0^*		
59	4	3	3^*	2	2	2	2	2^*	2	1^*	1^*	1^*	1^*	1^*	1^*	1^*	1^*	1^*	1^*	1^*	1^*	1^*	1^*	1^*	1^*	1^*	1^*	1^*	1^*	
60	4	4	3^*	3	2	2	2	2	2^*	2	1^*	1^*	1^*	1^*	1^*	1^*	1^*	1^*	1^*	1^*	1^*	1^*	1^*	1^*	1^*	1^*	1^*	1^*	1^*	0^*

Table 3.5: Ternary LCD Upper Bounds on Dimension for $2 \leq n \leq 25$ and $1 \leq d \leq 25$

$n\backslash d$	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
2	2^*	1^*																							
3	3^*	1^*	0^*																						
4	4^*	3^*	2	1^*																					
5	5^*	4^*	2^*	1^*	1^*																				
6	6^*	4^*	3^*	2^*	1^*	0^*																			
7	7^*	6^*	4^*	3^*	2^*	1^*	1^*																		
8	8^*	7^*	5^*	4^*	2^*	2	1^*	1^*																	
9	9^*	7^*	6^*	4^*	3^*	2^*	1^*	1^*	0^*																
10	10^*	9^*	7^*	5^*	4^*	3^*	2^*	1^*	1^*	1^*															
11	11^*	10^*	7^*	6^*	5^*	3^*	2^*	2	1^*	1^*	1^*														
12	12^*	10^*	8^*	7^*	6^*	6	3^*	2^*	2	1^*	1^*	0^*													
13	13^*	12^*	9^*	8^*	7^*	6	5	3^*	2^*	1^*	1^*	1^*	1^*												
14	14^*	13^*	10^*	9^*	8^*	7	6	3^*	3	2^*	1^*	1^*	1^*	1^*											
15	15^*	13^*	11^*	10^*	9	8	7	5	3^*	2^*	2	1^*	1^*	1^*	0^*										
16	16^*	15^*	12^*	11^*	10	9	7	6	5	3^*	2^*	2	1^*	1^*	1^*	1^*									
17	17^*	16^*	13^*	12^*	11	10	8	7	6	5	3^*	2^*	1^*	1^*	1^*	1^*	1^*								
18	18^*	16^*	14^*	13^*	12	11	9	8	7	6	3^*	3	2^*	1^*	1^*	1^*	1^*	0^*							
19	19^*	18^*	15^*	14^*	12	12	10	9	8	7	5	3^*	2^*	2	1^*	1^*	1^*	1^*	1^*						
20	20^*	19^*	16^*	14^*	13	12	10^*	10	8	7	6	5	3^*	2^*	2	1^*	1^*	1^*	1^*	1^*					
21	21^*	19^*	17^*	15^*	14	13	12	11	9	8	7	6	3^*	3	2^*	1^*	1^*	1^*	1^*	1^*	0^*				
22	22^*	21^*	18^*	16^*	15	14	13	12	10	9	8	7	5	3^*	3	2^*	1^*	1^*	1^*	1^*	1^*	1^*			
23	23^*	22^*	19^*	17^*	16	15	14	13	11	10	9	7	6	5	3^*	2^*	2	1^*	1^*	1^*	1^*	1^*	1^*		
24	24^*	22^*	20^*	18^*	17	16	15	14	12	11	10	8	7	6	3^*	3	2^*	2	1^*	1^*	1^*	1^*	1^*	0^*	
25	25^*	24^*	21^*	19^*	18	17	15	15	13	12	10	9	8	7	5	3^*	3	2^*	1^*	1^*	1^*	1^*	1^*	1^*	1^*

Table 3.6: Ternary LCD Upper Bounds on Dimension for $26 \leq n \leq 50$ and $1 \leq d \leq 25$

$n\backslash d$	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
26	26^*	25^*	22^*	20^*	19	18	16	15	14	13	11	10	9	7	6	5	3^*	3	2^*	1^*	1^*	1^*	1^*	1^*	1^*
27	27^*	25^*	23^*	21^*	20	19	17	16	15	14	12	11	10	8	7	6	3^*	4	2^*	2	1^*	1^*	1^*	1^*	1^*
28	28^*	27^*	24^*	22^*	21	20	18	17	16	15	13	12	10	9	8	7	5	3^*	3	2^*	2	1^*	1^*	1^*	1^*
29	29^*	28^*	25^*	23^*	22	21	19	18	17	16	14	13	11	10	9	8	6	5	3^*	3	2^*	1^*	1^*	1^*	1^*
30	30^*	28^*	26^*	24^*	23	22	20	19	18	17	15	14	12	11	10	8	7	6	5	3^*	2	2^*	1^*	1^*	1^*
31	31^*	30^*	27^*	25^*	24	23	21	20	18	17	16	15	13	12	11	9	8	7	5	3^*	3	2^*	2	1^*	1^*
32	32^*	31^*	28^*	26^*	25	24	22	21	19	18	17	16	14	13	12	10	9	8	6	5	3^*	3	2^*	2	1^*
33	33^*	31^*	29^*	27^*	25	25	23	22	20	19	18	17	15	14	12	11	10	8	7	6	5	3^*	2	2^*	1^*
34	34^*	33^*	30^*	28^*	26	25	24	23	21	20	19	18	16	15	13	12	11	9	8	7	5	3^*	3	2	2^*
35	35^*	34^*	31^*	29^*	27	26	24	24	22	21	19	18	17	16	14	13	11	10	9	8	6	5	3^*	3	2^*
36	36^*	34^*	32^*	30^*	28	27	25	24	23	22	20	19	18	17	15	14	12	11	10	8	7	6	5	3^*	2
37	37^*	36^*	32^*	31^*	29	28	26	25	24	23	21	20	19	18	16	15	13	12	11	9	8	7	5	3^*	3
38	38^*	37^*	33^*	32^*	30	29	27	26	25	24	22	21	19	18	17	16	14	13	11	10	9	8	6	5	3^*
39	39^*	37^*	34^*	33^*	31	30	28	27	26	25	23	22	20	19	18	17	15	14	12	11	10	8	7	6	5
40	40^*	39^*	35^*	34^*	32	31	29	28	26	26	24	23	21	20	19	17	16	15	13	12	11	9	8	7	5
41	41^*	40^*	36^*	35^*	33	32	30	29	27	26	25	24	22	21	20	18	17	16	14	13	11	10	9	8	6
42	42^*	40^*	37^*	36^*	34	33	31	30	28	27	26	25	23	22	20	19	18	17	15	14	12	11	9	8	7
43	43^*	42^*	38^*	37^*	35	34	32	31	29	28	27	26	24	23	21	20	19	17	16	14	13	12	10	9	8
44	44^*	43^*	39^*	38^*	36	35	$33^\uparrow$	33	30	29	28	27	25	24	22	21	20	18	17	16	14	13	11	10	9
45	45^*	43^*	40^*	39^*	$37^\uparrow$	$37^\uparrow$	$34^\uparrow$	$34^\uparrow$	$31^\uparrow$	$31^\uparrow$	29	28	26	25	23	22	20	19	18	17	15	14	12	11	9
46	46^*	45^*	41^*	40^*	$38^\uparrow$	$38^\uparrow$	$35^\uparrow$	$34^\uparrow$	$32^\uparrow$	$32^\uparrow$	29	29	27	26	24	23	21	20	19	17	16	14	13	12	10
47	47^*	46^*	42^*	41^*	$39^\uparrow$	$39^\uparrow$	$36^\uparrow$	$35^\uparrow$	$33^\uparrow$	$33^\uparrow$	$30^\uparrow$	$30^\uparrow$	28	27	25	24	22	21	20	18	17	15	14	12	11
48	48^*	46^*	43^*	42^*	$40^\uparrow$	$39^\uparrow$	$37^\uparrow$	$36^\uparrow$	$34^\uparrow$	$34^\uparrow$	$31^\uparrow$	$31^\uparrow$	$29^\uparrow$	28	26	25	23	22	20	19	18	16	15	13	12
49	49^*	48^*	44^*	43^*	$41^\uparrow$	$40^\uparrow$	$38^\uparrow$	$37^\uparrow$	$35^\uparrow$	$34^\uparrow$	$32^\uparrow$	$32^\uparrow$	$30^\uparrow$	$29^\uparrow$	27	26	24	23	21	20	19	17	16	14	13
50	50^*	49^*	45^*	44^*	$42^\uparrow$	$41^\uparrow$	$39^\uparrow$	$38^\uparrow$	$36^\uparrow$	$35^\uparrow$	$33^\uparrow$	$33^\uparrow$	$31^\uparrow$	$30^\uparrow$	28	27	25	24	22	21	20	18	17	15	14

Table 3.7: Ternary LCD Upper Bounds on Dimension for $26 \leq n \leq 50$ and $26 \leq d \leq 50$

$n\backslash d$	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50
26	1^*																								
27	1^*	0^*																							
28	1^*	1^*	1^*																						
29	1^*	1^*	1^*	1^*																					
30	1^*	1^*	1^*	1^*	0^*																				
31	1^*	1^*	1^*	1^*	1^*	1^*																			
32	1^*	1^*	1^*	1^*	1^*	1^*	1^*																		
33	1^*	1^*	1^*	1^*	1^*	1^*	1^*	0^*																	
34	1^*	1^*	1^*	1^*	1^*	1^*	1^*	1^*	1^*																
35	2	1^*	1^*	1^*	1^*	1^*	1^*	1^*	1^*	1^*															
36	2^*	2	1^*	1^*	1^*	1^*	1^*	1^*	1^*	1^*	0^*														
37	2	2^*	1^*	1^*	1^*	1^*	1^*	1^*	1^*	1^*	1^*	1^*													
38	3	2	2^*	1^*	1^*	1^*	1^*	1^*	1^*	1^*	1^*	1^*	1^*												
39	3^*	3	2^*	2	1^*	1^*	1^*	1^*	1^*	1^*	1^*	1^*	1^*	0^*											
40	3^*	4	2	2^*	2	1^*	1^*	1^*	1^*	1^*	1^*	1^*	1^*	1^*	1^*										
41	5	3^*	3	2	2^*	1^*	1^*	1^*	1^*	1^*	1^*	1^*	1^*	1^*	1^*	1^*									
42	6	5	3^*	3	2	2^*	1^*	1^*	1^*	1^*	1^*	1^*	1^*	1^*	1^*	1^*	0^*								
43	7	6	5	3^*	2	2^*	2	1^*	1^*	1^*	1^*	1^*	1^*	1^*	1^*	1^*	1^*	1^*							
44	7	6	5	3^*	3	2	2^*	2	1^*	1^*	1^*	1^*	1^*	1^*	1^*	1^*	1^*	1^*	1^*						
45	8	7	6	5	3^*	3	2	2^*	1^*	1^*	1^*	1^*	1^*	1^*	1^*	1^*	1^*	1^*	1^*	0^*					
46	9	8	6	6	5	3^*	2	2	2^*	1^*	1^*	1^*	1^*	1^*	1^*	1^*	1^*	1^*	1^*	1^*	1^*				
47	10	9	7	6	5	3^*	3	2	2^*	2	1^*	1^*	1^*	1^*	1^*	1^*	1^*	1^*	1^*	1^*	1^*	1^*			
48	11	9	8	7	6	5	3^*	3	2	2^*	2	1^*	1^*	1^*	1^*	1^*	1^*	1^*	1^*	1^*	1^*	1^*	0^*		
49	12	10	9	8	6	6	5	3^*	2	2	2^*	1^*	1^*	1^*	1^*	1^*	1^*	1^*	1^*	1^*	1^*	1^*	1^*	1^*	
50	12	11	10	9	7	6	5	3^*	3	2	2	2^*	1^*	1^*	1^*	1^*	1^*	1^*	1^*	1^*	1^*	1^*	1^*	1^*	1^*

CHAPTER 4

LINEAR ALGEBRAIC CONSTRUCTIONS OF LCD CODES

In this chapter, we focus on the construction of new LCD codes from the known ones. For this, we use and generalize some known results in the literature. First, we need to fix some notation. Notice that if C is an LCD code over $\mathbb{F}_q$ with $d^\perp = 1$, then by Theorem 2.1 we have $G = (\mathbf{0}|G')$ for some zero matrix $\mathbf{0}$ of suitable size (say, $k \times r$ with $r \geq 1$). In this case, if C is LCD and C' is the code of length $n-r$ and dimension k with generator matrix G', then C' is also LCD. Conversely, if C' is LCD, then C is also LCD. Excluding these cases, we rely on Theorems 2.1 and 2.3 and assume that $d \geq 2$ and $d^\perp \geq 2$ in the following result. What remains is to recall that if C_1 and C^1 denote the shortened and the punctured codes of C on the first coordinate, respectively, then we have $(C_1)^\perp = (C^\perp)^1$ and $(C^1)^\perp = (C^\perp)_1$ (see [15, Theorem 1.5.57]). We are now ready to generalize the result given in [7, Lemma 2] for binary codes to q-ary setup.

Lemma 4.1. *If C is a q-ary LCD code of length n with $d \geq 2$ and $d^\perp \geq 2$, then exactly one of the codes C_1 and C^1 is also LCD.*

Proof. Since $\mathbb{F}_q^n = C \oplus C^\perp$, we have $(1, 0, \ldots, 0) = (0, v) + (1, -v)$, where $v \in \mathbb{F}_q^{n-1}$ such that $(0, v) \in C$ or $(0, v) \in C^\perp$. Let $B_1 = (C^\perp)_1 = (C^1)^\perp$ and $B^1 = (C^\perp)^1 = (C_1)^\perp$. There are two possibilities:

- $v \in C_1$, which implies $(0, v) \in C$ and $(1, -v) \in C^\perp$.
- $v \in C^1$, which implies $(0, v) \in C^\perp$ and $(1, -v) \in C$.

1. If $v \in C_1$, then $(1, -v) \in C^\perp$ implies $-v \in B^1$ and therefore $v \in B^1 = (C_1)^\perp$.

Hence, C_1 is not LCD. Suppose that C^1 is not LCD either. Let $w \in C^1 \cap B_1$. Then, $(0, w) \in C^\perp$ and $(1, -w) \in C$. Note that if $v = w$, then $(0, w) \in C \cap C^\perp$, which forces $v = w = 0$, but this contradicts $d \geq 2$ as $(1, -w) \in C$. Hence, $v \neq w$ and

$$(1, 0, \ldots, 0) = \underbrace{(0, v)}_{\in C} + \underbrace{(1, -v)}_{\in C^\perp} = \underbrace{(0, w)}_{\in C^\perp} + \underbrace{(1, -w)}_{\in C}$$

yields two different representations of $(1, 0, \ldots, 0)$, which is a contradiction. Therefore, C^1 should be LCD.

2. If $v \in B_1$, then $(1, -v) \in C$ implies $-v \in C^1$ and therefore $v \in C^1$. Hence, C^1 is not LCD. Suppose that C_1 is not LCD either. Let $w \in C_1 \cap B^1$. Then, $(0, w) \in C$ and $(1, -w) \in C^\perp$. Note that if $v = w$, then $(0, w) \in C \cap C^\perp$, which forces $v = w = 0$, but this contradicts $d^\perp \geq 2$ as $(1, -w) \in C^\perp$. Hence, $v \neq w$ and

$$(1, 0, \ldots, 0) = \underbrace{(0, v)}_{\in C^\perp} + \underbrace{(1, -v)}_{\in C} = \underbrace{(0, w)}_{\in C} + \underbrace{(1, -w)}_{\in C^\perp}$$

yields two different representations of $(1, 0, \ldots, 0)$, which is a contradiction. Therefore, C_1 should be LCD.

□

The proposition below is proven for binary LCD codes in [7, Proposition 4]. We generalize it to q-ary case as follows.

Proposition 4.2. *Let the C be q-ary $[n, k, d]_q$ code with $k \times n$ generator matrix G, where $q = p^m$ for some prime p and $n \in \mathbb{Z}^+$. We define the $k \times (p + n)$ matrix $G' = \begin{pmatrix} v^\top & v^\top & \cdots & v^\top & G \end{pmatrix}$, where $v \in \mathbb{F}_q^k$. Let C' be the q-ary code generated by the rows of G'. Then C is LCD if and only if C' is LCD.*

Proof. For $x, y \in C$, consider $x' = (\underbrace{a, a, \cdots, a}_{p \text{ times}}, x)$ and $y' = (\underbrace{b, b, \cdots, b}_{p \text{ times}}, y) \in C'$, for some $a, b \in \mathbb{F}_q$.
Then we have $\langle x', y' \rangle = p \cdot (a \cdot b) + \langle x, y \rangle = \langle x, y \rangle \Rightarrow G' \cdot G'^T = G \cdot G^\top$. □

In the next two theorems, we provide extensions of Theorem 3.1 given in [13] and Theorems 3.5 and 3.10 given in [22].

Theorem 4.3 (Extended Construction Method 1)**.** *Let C be an arbitrary LCD $[n, k]$ code with generator matrix G over $\mathbb{F}_q$ with rows $r_1, r_2, \cdots, r_k \in \mathbb{F}_q^n$. Let $x = (x_1, x_2, \cdots, x_n) \in \mathbb{F}_q^n \setminus \{0\}$ and let $r > 1$ be a positive integer such that there exists $a = (a_1, a_2, \cdots, a_r), b = (b_1, b_2, \cdots, b_r) \in \mathbb{F}_q^r$ satisfying $\langle a, a\rangle + \langle x, x\rangle \neq 0$, $\langle b, b\rangle = 0$ and $\langle b, a\rangle + 1 = 0$ when $x \notin C^{\perp}$. Let $C(x, a, b)$ be the a code with the following generator matrix:*

$$G(x,a,b) = \begin{pmatrix} a_1 & a_2 & \cdots & a_r & x_1 & x_2 & \cdots & x_n \\ b_1\langle x, r_1\rangle & b_2\langle x, r_1\rangle & \cdots & b_r\langle x, r_1\rangle & & & & \\ \vdots & & \vdots & \vdots & & G & & \\ b_1\langle x, r_k\rangle & b_2\langle x, r_k\rangle & \cdots & b_r\langle x, r_k\rangle & & & & \end{pmatrix} = \begin{pmatrix} a & x \\ B & G \end{pmatrix}$$

Then $C(x, a, b)$ is an LCD [n+r, k+1] code over $\mathbb{F}_q$.

Proof. We will show that $G(x, a, b) \cdot G(x, a, b)^{\top}$ is nonsingular. We have

$$\begin{aligned} G(x,a,b) \cdot G(x,a,b)^{\top} &= \begin{pmatrix} a & x \\ B & G \end{pmatrix} \cdot \begin{pmatrix} a^{\top} & B^{\top} \\ x^{\top} & G^{\top} \end{pmatrix} \\ &= \begin{pmatrix} \langle a, a\rangle + \langle x, x\rangle & a \cdot B^{\top} + x \cdot G^{\top} \\ B \cdot a^{\top} + G \cdot x^{\top} & B \cdot B^{\top} + G \cdot G^{\top} \end{pmatrix}, \end{aligned}$$

where

$$B = \begin{pmatrix} b_1\langle x, r_1\rangle & b_2\langle x, r_1\rangle & \cdots & b_r\langle x, r_1\rangle \\ \vdots & & \ddots & \vdots \\ b_1\langle x, r_k\rangle & b_2\langle x, r_k\rangle & \cdots & b_r\langle x, r_k\rangle \end{pmatrix}.$$

First, we will show that $a \cdot B^\top + x \cdot G^\top = B \cdot a^\top + G \cdot x^\top = 0$. We consider

$$\begin{aligned} a \cdot B^\top &= \begin{pmatrix} a_1 & a_2 & \cdots & a_r \end{pmatrix} \cdot \begin{pmatrix} b_1\langle x, r_1\rangle & b_1\langle x, r_2\rangle & \cdots & b_1\langle x, r_k\rangle \\ \vdots & & \ddots & \vdots \\ b_r\langle x, r_1\rangle & b_r\langle x, r_2\rangle & \cdots & b_r\langle x, r_k\rangle \end{pmatrix} \\ &= \begin{pmatrix} \langle x, r_1\rangle \cdot \langle b, a\rangle & \langle x, r_2\rangle \cdot \langle b, a\rangle & \cdots & \langle x, r_k\rangle \cdot \langle b, a\rangle \end{pmatrix} \\ &= \langle b, a\rangle \cdot \begin{pmatrix} \langle x, r_1\rangle & \langle x, r_2\rangle & \cdots & \langle x, r_k\rangle \end{pmatrix}. \end{aligned}$$

On the other hand,

$$x \cdot G^\top = \begin{pmatrix} x_1 & x_2 & \cdots & x_n \end{pmatrix} \cdot \begin{pmatrix} r_{11} & r_{21} & \cdots & r_{k1} \\ r_{12} & r_{22} & \cdots & r_{k2} \\ \vdots & \vdots & \ddots & \vdots \\ r_{1n} & r_{2n} & \cdots & r_{kn} \end{pmatrix} = \begin{pmatrix} \langle x, r_1\rangle & \langle x, r_2\rangle & \cdots & \langle x, r_k\rangle \end{pmatrix}.$$

Hence, we have

$$a \cdot B^\top + x \cdot G^\top = (B \cdot a^\top + G \cdot x^\top)^\top = (\langle b, a\rangle + 1) \cdot \begin{pmatrix} \langle x, r_1\rangle & \langle x, r_2\rangle & \cdots & \langle x, r_k\rangle \end{pmatrix} = 0,$$

since either we have $x \in C^\perp$ and each $\langle x, r_i\rangle = 0$, or $x \in C \setminus \{0\}$ and $\langle b, a\rangle + 1 = 0$.

Next, we consider $B \cdot B^\top + G \cdot G^\top$, where $G \cdot G^\top$ is known to be nonsingular as C is LCD. Using the condition $\langle b, b\rangle = 0$, we obtain

$$\begin{aligned} B \cdot B^\top &= \begin{pmatrix} b_1\langle x, r_1\rangle & b_2\langle x, r_1\rangle & \cdots & b_r\langle x, r_1\rangle \\ \vdots & & \ddots & \vdots \\ b_1\langle x, r_k\rangle & b_2\langle x, r_k\rangle & \cdots & b_r\langle x, r_k\rangle \end{pmatrix} \cdot \begin{pmatrix} b_1\langle x, r_1\rangle & b_1\langle x, r_2\rangle & \cdots & b_1\langle x, r_k\rangle \\ \vdots & & \ddots & \vdots \\ b_r\langle x, r_1\rangle & b_r\langle x, r_2\rangle & \cdots & b_r\langle x, r_k\rangle \end{pmatrix} \\ &= \begin{pmatrix} b_1\langle x, r_1\rangle \cdot b_1\langle x, r_1\rangle + b_2\langle x, r_1\rangle \cdot b_2\langle x, r_1\rangle + \cdots + b_r\langle x, r_1\rangle \cdot b_r\langle x, r_1\rangle & \cdots \\ \vdots & \\ b_1\langle x, r_k\rangle \cdot b_1\langle x, r_k\rangle + b_2\langle x, r_k\rangle \cdot b_2\langle x, r_k\rangle + \cdots + b_r\langle x, r_k\rangle \cdot b_r\langle x, r_k\rangle & \cdots \end{pmatrix} \\ &= \begin{pmatrix} \langle b, b\rangle\langle x, r_1\rangle\langle x, r_1\rangle & \langle b, b\rangle\langle x, r_1\rangle\langle x, r_2\rangle & \cdots \langle b, b\rangle\langle x, r_1\rangle\langle x, r_k\rangle \\ \vdots & \\ \langle b, b\rangle\langle x, r_k\rangle\langle x, r_1\rangle & \langle b, b\rangle\langle x, r_k\rangle\langle x, r_2\rangle & \cdots \langle b, b\rangle\langle x, r_k\rangle\langle x, r_k\rangle \end{pmatrix} = 0. \end{aligned}$$

Therefore, we get

$$G(x,a,b)\cdot G(x,a,b)^{\top} = \begin{pmatrix} \langle a,a\rangle + \langle x,x\rangle & 0 \\ 0 & G\cdot G^{\top} \end{pmatrix}.$$

Since $G\cdot G^{\top}$ is nonsingular and $\langle a,a\rangle + \langle x,x\rangle \neq 0$, the matrix $G(x,a,b)\cdot G(x,a,b)^{\top}$ is nonsingular, and hence $C(x,a,b)$ is an LCD code. □

Remark 4.4. Observe that, when $x \notin C^{\perp}$ and x is nonzero, the condition $\langle a,b\rangle + 1 = 0$ implies that the vector b cannot be the zero vector even though $\langle b,b\rangle = 0$.

Moreover, for any prime p, the case $r = 1$ is impossible. To see this, assume that $r = 1$. Then, $\langle b,b\rangle = b^2$ implies that $b = 0$. However, this contradicts the condition $\langle a,b\rangle + 1 = 0$.

In the particular case when $\mathrm{char}(\mathbb{F}_q) = p = 3$, we must have $r > 2$. If we assume that $r = 2$, then having $\langle b,b\rangle = 0 = b_1^2 + b_2^2$ again forces $b = (0,0)$ and leads to the same contradiction

The following corollary is a special case when we take $r = p$.

Corollary 4.5. *Let C be a q-ary LCD $[n,k]$ code with generator matrix G. Let $x \in \mathbb{F}_q^n$ so that*

(i) $\langle x,x\rangle + 1 \neq 0$ and we set

$$G_1(x) = \begin{pmatrix} 1 & 0 & \cdots & 0 & x_1 & \cdots & x_n \\ -\langle x,r_1\rangle & -\langle x,r_1\rangle & \cdots & -\langle x,r_1\rangle & & & \\ & & \vdots & & & G & \\ \underbrace{-\langle x,r_k\rangle}_{p \text{ times}} & -\langle x,r_k\rangle & \cdots & -\langle x,r_k\rangle & & & \end{pmatrix},$$

(ii) or $\langle x,x\rangle + 3 \neq 0$ and we set

$$G_2(x) = \begin{pmatrix} 1 & 1 & \cdots & 2 & x_1 & \cdots & x_n \\ -\langle x,r_1\rangle & -\langle x,r_1\rangle & \cdots & -\langle x,r_1\rangle & & & \\ & & \vdots & & & G & \\ \underbrace{-\langle x,r_k\rangle}_{p \text{ times}} & -\langle x,r_k\rangle & \cdots & -\langle x,r_k\rangle & & & \end{pmatrix}.$$

Then, both $G_1(x)$ and $G_2(x)$ generate LCD $[n+p, k+1]$ codes.

Proof. Proof is given using Theorem 4.3.

(i) If we take $a = (1, 0, \cdots, 0)$, $b = (-1, -1, \cdots, -1)$,
then the conditions become

1. $\langle a, a\rangle + \langle x, x\rangle = \langle x, x\rangle + 1 \neq 0$.
2. $\langle b, b\rangle = \sum_{i=1}^{p} (-1)^2 = p = 0$.
3. $\langle b, a\rangle + 1 = -1 + 1 = 0$.

(ii) If we take $a = (1, 1, \cdots, 2)$ and $b = (-1, -1, \cdots, -1)$, then we get

$$\begin{aligned}
\langle a, a\rangle &= \sum_{i=1}^{p-1} 1 + 2^2 = (p-1) + 4 = 3, \\
\langle b, a\rangle &= \langle (-1, -1, \cdots, -1), (1, 1, \cdots, 2)\rangle \\
&= (\sum_{1}^{p-1} -1) + p - 2 = (-1)\cdot(p-1) + (-1)\cdot 2 \\
&= (-1)\cdot(p - 1 + 2) = -1.
\end{aligned}$$

Hence,

1. $\langle a, a\rangle + \langle x, x\rangle = \langle x, x\rangle + 3 \neq 0$.
2. $\langle b, b\rangle = \sum_{i=1}^{p} (-1)^2 = p = 0$.
3. $\langle b, a\rangle + 1 = -1 + 1 = 0$.

□

Remark 4.6. Corollary 4.5 above generalizes the binary and ternary construction methods originally presented in [13, Theorem 3.1] to the q-ary setup.

If we consider the case $B = 0$ and $x \in C^{\perp}$ in the above construction, we can further extend this case containing [22, Theorems 3.5 and 3.10] as follows.

Theorem 4.7 (Extended Construction Method 2)**.** *Let C be an arbitrary q-ary LCD [n, k] code with the generator matrix G. For $1 \leq m < (n-k)$ and $r > 1$, let $X = \left(x_1^\top,\ x_2^\top\ \cdots\ x_m^\top\right)^\top$, where $x_i \in C^\perp$, for all $1 \leq i \leq m$. Let A be an $m \times r$ matrix over $\mathbb{F}_q$ such that $AA^\top + XX^\top$ is nonsingular. If we consider*

$$G(A, X) = \begin{pmatrix} A & X \\ \mathbf{0} & G \end{pmatrix},$$

then the [n+r, k+m] code $C(A, X)$ generated by $G(A, X)$ is also LCD.

Proof. We have

$$\begin{aligned} G(A,X) \cdot G(A,X)^\top &= \begin{pmatrix} A & X \\ \mathbf{0} & G \end{pmatrix} \cdot \begin{pmatrix} A^\top & \mathbf{0} \\ X^\top & G^\top \end{pmatrix} \\ &= \begin{pmatrix} AA^\top + XX^\top & XG^\top \\ GX^\top & GG^\top \end{pmatrix} = \begin{pmatrix} AA^\top + XX^\top & \mathbf{0} \\ \mathbf{0} & GG^\top \end{pmatrix}. \end{aligned}$$

□

Remark 4.8. Let us focus on the particular case when $m = r = 1$, that is, $a \in \mathbb{F}_q^*$ and $x \in C^\perp$ such that

$$G(a, x) = \begin{pmatrix} a & x \\ 0 & G \end{pmatrix}.$$

Notice that, $G(a, x)$ can be simplified by extracting factor a, i.e.

$$G(a,x) = a \cdot \begin{pmatrix} 1 & a^{-1} \cdot x \\ 0 & a^{-1} \cdot G \end{pmatrix} = a \cdot \begin{pmatrix} 1 & x' \\ 0 & G' \end{pmatrix} = a \cdot G'(1, x').$$

Clearly, $x' \in C^\perp$ and $G' = a^{-1} \cdot G$ also generates C. Since $G'G'^\top$ is nonsingular and $1 + \langle x', x' \rangle = a^2 + \langle x, x \rangle \neq 0$, we obtain an LCD [n+1, k+1] code over $\mathbb{F}_q$, which yields the q-ary generalization of [22, Theorem 3.5].

Furthermore, if we consider the particular case when $m = 1$ and $r = 0$, then we get

$$G(0, x) = \begin{pmatrix} x \\ G \end{pmatrix},$$

where $x \in C^\perp$. Obviously,

$$G(0,x) \cdot G(0,x)^\top = \begin{pmatrix} x \\ G \end{pmatrix} \cdot \begin{pmatrix} x^\top & G^\top \end{pmatrix} = \begin{pmatrix} \langle x, x \rangle & 0 \\ 0 & G \cdot G^\top \end{pmatrix}$$

together with the constraint $\langle x, x\rangle \neq 0$ gives an LCD [n, k+1] over $\mathbb{F}_q$, which yields the q-ary generalization of [22, Theorem 3.10].

4.1 Minimum Weights of the New Codes

Assume we have an arbitrary $[n, k, d]_q$ code C with generator matrix G. We define a new code $C(X, A, B)$ with parameters $[n + r,\ k + m, d']$ generated by

$$G'(X, A, B) = \begin{pmatrix} A & X \\ B & G \end{pmatrix}.$$

Here, the matrices are given by

$$A \in \mathbb{F}_q^{m\times r}, \quad X \in \mathbb{F}_q^{m\times n}, \quad B \in \mathbb{F}_q^{k\times r}.$$

Any codeword $c' \in C(X, A, B)$ can be expressed as

$$c' = (u \,|\, v) \cdot G(X, A, B),$$

where $(u \,|\, v) \in \mathbb{F}_q^{m+k}$ such that $u \in \mathbb{F}_q^m, v \in \mathbb{F}_q^k$. In other words,

$$c' = (u \,|\, v) \begin{pmatrix} A & X \\ B & G \end{pmatrix} = \Big(uA + vB \,\Big|\, uX + vG\Big).$$

Thus, the Hamming weight of c' is

$$\text{wt}(c') = \text{wt}(uA + vB) + \text{wt}(uX + vG).$$

Special Case 1: $B = 0$ and $X = 0$ (Direct Sum):
If the matrices B and X are omitted, then the new code depends solely on A and G and the Hamming weight of c' becomes

$$\text{wt}(c') = \text{wt}(uA) + \text{wt}(vG).$$

If we set $\mathcal{A}$ as the code generated by the matrix A, then clearly we have $d(C(0, A, 0)) = \min\{d(\mathcal{A}), d(C)\}$.

Special Case 2: $A = 0$ and $X = 0$:

If the matrices A and X are omitted, then the new code depends solely on B and G and Hamming weight of c' becomes

$$\text{wt}(c') = \text{wt}(vB) + \text{wt}(vG).$$

Then, the minimum distance of $C(0, 0, B)$ satisfies

$$d(C(0,0,B)) = \begin{cases} d, & \text{if } \exists v \in \mathbb{F}_q^k : vB = 0, \ \& \ vG \neq 0 \\ > d, & \text{otherwise.} \end{cases}$$

Here, the case $vB \neq 0 \ \ \& \ \ vG = 0$ is impossible, since there is no v other than zero vector that makes $vG = 0$. Hence, we have $d(C(0, 0, B)) \geq d(C)$.

Special Case 3: $A = 0$ and $B = 0$:

When both A and B are omitted, the generator matrix reduces to the matrix $\begin{pmatrix} X \\ G \end{pmatrix}$ and the Hamming weight of c' is given by

$$\text{wt}(c') = \text{wt}(uX + vG).$$

Since C becomes a subcode of $C(X, 0, 0)$, we can conclude that $d(C(X, 0, 0)) \leq d(C)$.

4.2 LCD Code Generation Algorithm

Since the methods in chapter 4 controls only the code length and dimension (not the minimum weight), the designed algorithm generates codes that maximize the minimum weight for given length and dimension constraints. Codes involve trade-offs among parameters n, k, and d, categorized as follows:

- Methods that decreases n and may decrease k but increase d.
  - Lemma 4.1: Given [n, k, d], generates [$n' < n$, $k' \leq k$, $d' \geq d$]. Note that d may increase due to shortening.
- Methods that increase n while keeping k constant, potentially also increasing d.

  - Proposition 4.2: Given $[n, k, d]$, generates $[n' > n, k, d' \geq d]$. The value of d' may increase or remain the same, as detailed in Special Case 2 of Section 4.1.

- Methods that increase both n and k but may decrease d.

  - Theorem 4.3: Given $[n, k, d]$, generates $[n' > n, k' > k, d']$. The value of d' may increase or remain the same, as explained in Section 4.1.

  - Theorem 4.7: Given $[n, k, d]$, generates $[n' > n, k' > k, d']$. Here, the value of d' may decrease or remain the same, as discussed under Special Case 3 in Section 4.1.

Codes already known for given parameters n and k are skipped. The algorithm uses cyclic and quasi-cyclic codes from Chapter 5 and LCD codes filtered from database of best dimension linear codes Magma V2.28-19's [6].

The algorithm operates in two stages:

1. Direct enhancement of a given code targeting specific parameters n and k, maximizing minimum weight d.

2. A breadth-first approach allowing initial parameter deviations. Algorithm 1 is applied recursively to deviated codes, eventually reapplying to original target parameters. Codes not meeting targets and unexplored are queued for further processing. Processing continues until achieving the optimal minimum weight or reaching iteration limits, guided by known results and LP bounds from Chapter 3.

Remark 4.9. Each method, when applied, strives to eagerly maximize the minimum distance whenever possible. For instance, in the case of Lemma 4.1, the program randomly selects certain positions and retains the code with the highest minimum distance among the generated candidate codes. The terminology used in the algorithm, such as "search" or "best possible," is chosen to reflect this intent.

Algorithm 1 Pseudo-code for code improvement algorithm

procedure $get\_improved\_code(target\_n, target\_k, target\_d, code\_record)$
    $p \leftarrow$ characteristic of $code\_record$
    $current\_n, current\_k, current\_d \leftarrow code\_record$
    $diff\_n \leftarrow target\_n - current\_n$
    $diff\_k \leftarrow target\_k - current\_k$
    $should\_improve\_n \leftarrow diff\_n > 0$
    $should\_improve\_k \leftarrow diff\_k > 0$
    $should\_improve\_d \leftarrow target\_d > current\_d$

    if $should\_improve\_d$ **or** $diff\_n \leq 0$ **or** $diff\_k \leq 0$ **then**
        $new\_code \leftarrow$ Search best possible code Lemma 4.1 (Try different positions)
        if $new\_code.d \geq target\_d$ **or** $new\_code.d > current\_d$ **then**
            return new_code, **False**
        end if
    end if

    if $diff\_n \leq 0$ **or** $diff\_k \leq 0$ **then**
        return $code\_record$, **True** ▷ No further improvement possible
    end if

    if $should\_improve\_k$ **then**
        if $diff\_n \geq p$ **then**
            return Apply Theorem 4.3 to generate best possible code, **False**
        else
            return Apply Theorem 4.7 to generate best possible code, **False**
        end if
    end if

    if $should\_improve\_n$ **then**
        return Search the best possible code using Proposition 4.2, **False**
    end if

    return $code\_record$, **True** ▷ No further improvement possible
end procedure

Algorithm 2 Breadth-first Search Algorithm for Code Generation

```
Initialize queue with initial_code_pool
visited ← initial_code_pool
iterations ← 0
best_for_target ← None
batch_size ← 12                          ▷ Select a value for processor count
deviation ← 2                            ▷ Select higher value to explore more
queue.extend(Apply deviation methods to initial_code_pool)
while queue is not empty and iterations < max_iterations do
    current_batch ← Pop up to batch_size codes from queue
    improved_codes ← Run Algorithm 1 in parallel on current_batch
    for each improved_code, dead_end in improved_codes do
        iterations ← iterations + 1
        if improved_code matches target n and k then
            if best_for_target is None or improved_code.d ≥ best_for_target.d
then
                best_for_target ← improved_code
            end if
            if best_for_target.d == absolute_max_possible_d then
                return best_for_target
            end if
        end if
        if dead_end or improved_code already in visited then
            continue
        end if
        Add improved_code to visited
        Add improved_code to queue
    end for
end while
return best_for_target
```

Algorithm 3 Determine Maximum Possible Minimum Weight d

procedure $get\_maximum\_possible\_d(q, n, k, n\_d\_lp_{results})$
    $d_{known} \leftarrow$ largest known minimum distance from previous results for parameters (q, n, k)
    if d_{known} exists **then**
        return d_{known}
    end if

    $d_{singleton} \leftarrow n - k + 1$
    $d_{lp} \leftarrow$ largest known minimum distance where $k_{lp} \leq k$ from $n\_d\_lp_{results}$
    if $\exists d_{lp}$ **then**
        return $\min(d_{singleton}, d_{lp})$
    end if
    return $get\_maximum\_possible\_d(q, n, k-1, n\_d\_lp_{results})$
end procedure

4.3 Generated LCD Codes Results

Since the complete output is extensive, we present one representative example each for binary and ternary codes. The remaining code results are available in the `outputs` folder of our GitHub repository [17]. The generated codes are systematically stored in *JSON* files using the naming convention $q\_n\_k\_d_{max}$, where each element of the key represents specific properties of the code $C \subseteq \mathbb{F}_q^n$: q is the cardinality of the finite field $\mathbb{F}_q$, **n** is the length, **k** is the dimension of the code C, and d_{max} is an upper bound for the minimum distance, as calculated using *Algorithm 3*.

Each JSON file includes comprehensive details of the generated codes:

- `"code"`: A textual summary of the current code's parameters.
- `"GenMatrix"`: The generator matrix explicitly represented as arrays, enabling direct construction of the codewords.
- `"min_distance"`: The computed or known minimum distance, indicating the error-correcting capability.
- `"const_method_params"`: The specific theoretical construction method used, identified by theorem or lemma from referenced literature.

- "`gen_objects`": Parameters or vectors involved in the code construction, clearly illustrating how the current code was derived.

- "`parent_code`": A nested JSON object linking to the parent code, establishing a hierarchical and traceable derivation from initial codes. If this field is `null`, the code is the root code.

This structured representation facilitates straightforward verification, reproduction, and future extensions of the generated code results. Note that, when applying Lemma 4.1 positions start from 0.

Example 4.10. Binary $[46, 30, 5]$ LCD code generated from $[41, 21, 9]$

1. **[41, 21, 9]**:

$$\begin{pmatrix}
1\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,1\,0\,1\,1\,1\,1\,1\,0\,0\,1\,1\,1\,0\,0\,1\,1\,1\,1\,1\,0 \\
0\,1\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,1\,0\,1\,1\,1\,1\,1\,0\,0\,1\,1\,1\,0\,0\,1\,1\,1\,1\,1 \\
0\,0\,1\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,1\,0\,0\,1\,0\,0\,0\,1\,1\,1\,1\,0\,1\,1\,1\,1\,0\,0\,0\,1 \\
0\,0\,0\,1\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,1\,1\,1\,1\,0\,1\,1\,0\,1\,0\,0\,0\,0\,1\,0\,0\,0\,1\,1\,0 \\
0\,0\,0\,0\,1\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,1\,1\,1\,1\,0\,1\,1\,0\,1\,0\,0\,0\,0\,1\,0\,0\,0\,1\,1 \\
0\,0\,0\,0\,0\,1\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,1\,0\,0\,0\,0\,0\,1\,1\,1\,1\,0\,1\,0\,0\,1\,0\,1\,1\,1\,1 \\
0\,0\,0\,0\,0\,0\,1\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,1\,1\,1\,1\,1\,1\,1\,1\,1\,0\,0\,1\,1\,0\,1\,0\,1\,0\,0\,1 \\
0\,0\,0\,0\,0\,0\,0\,1\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,1\,1\,0\,0\,0\,0\,0\,1\,1\,0\,1\,1\,1\,1\,1\,0\,1\,0\,1\,0 \\
0\,0\,0\,0\,0\,0\,0\,0\,1\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,1\,1\,0\,0\,0\,0\,0\,1\,1\,0\,1\,1\,1\,1\,1\,0\,1\,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\,0\,1\,1\,1\,0\,0\,0\,0\,1\,1\,1\,0\,0\,0\,1\,0\,0 \\
0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,1\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,1\,0\,0\,0\,1\,1\,1\,0\,0\,0\,0\,1\,1\,1\,0\,0\,0\,1\,0 \\
0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,1\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,1\,0\,0\,0\,1\,1\,1\,0\,0\,0\,0\,1\,1\,1\,0\,0\,0\,1 \\
0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,1\,0\,0\,0\,0\,0\,0\,0\,0\,1\,0\,1\,0\,1\,1\,1\,1\,1\,0\,1\,1\,0\,0\,0\,0\,0\,1\,1\,0 \\
0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,1\,0\,0\,0\,0\,0\,0\,0\,0\,1\,0\,1\,0\,1\,1\,1\,1\,1\,0\,1\,1\,0\,0\,0\,0\,0\,1\,1 \\
0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,1\,0\,0\,0\,0\,0\,0\,1\,0\,0\,1\,0\,1\,0\,1\,1\,0\,0\,1\,1\,1\,1\,1\,1\,1\,1\,1 \\
0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,1\,0\,0\,0\,0\,0\,1\,1\,1\,1\,0\,1\,0\,0\,1\,0\,1\,1\,1\,1\,0\,0\,0\,0\,0\,1 \\
0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,1\,0\,0\,0\,0\,1\,1\,0\,0\,0\,1\,0\,0\,0\,0\,1\,0\,1\,1\,0\,1\,1\,1\,1\,0 \\
0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,1\,0\,0\,0\,0\,1\,1\,0\,0\,0\,1\,0\,0\,0\,0\,1\,0\,1\,1\,0\,1\,1\,1\,1 \\
0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,1\,0\,0\,1\,0\,0\,0\,1\,1\,1\,1\,0\,1\,1\,1\,1\,0\,0\,0\,1\,0\,0\,1 \\
0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,1\,0\,1\,1\,1\,1\,1\,0\,0\,1\,1\,1\,0\,0\,1\,1\,1\,1\,1\,0\,1\,0 \\
0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,1\,0\,1\,1\,1\,1\,1\,0\,0\,1\,1\,1\,0\,0\,1\,1\,1\,1\,1\,0\,1
\end{pmatrix}$$

2. Apply Theorem 4.3 to **[41, 21, 9]** with parameters: $r = 3$, $a = (0\ 1\ 1)$, $b = (1\ 0\ 1)$ and $x = (0\,1\,0\,0\,0\,0\,0\,0\,1\,0\,0\,1\,0\,1\,0\,1\,1\,0\,0\,1\,0\,1\,1\,1\,1\,0\,0\,1\,1\,0\,0\,1\,0\,0\,0\,1\,0\,1\,0\,0\,1)$

[44, 22, 8]:

$$\begin{pmatrix}
0\,1\,1\,0\,1\,0\,0\,0\,0\,0\,0\,1\,0\,0\,1\,0\,1\,0\,1\,1\,0\,0\,1\,0\,1\,1\,1\,1\,0\,0\,1\,1\,0\,0\,1\,0\,0\,0\,1\,0\,1\,0\,0\,1\\
1\,0\,1\,1\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,1\,0\,1\,1\,1\,1\,1\,0\,0\,1\,1\,1\,0\,0\,1\,1\,1\,1\,1\,1\,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\,1\,0\,1\,1\,1\,1\,1\,0\,0\,1\,1\,1\,0\,0\,1\,1\,1\,1\,1\\
0\,0\,0\,0\,0\,1\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,1\,0\,0\,1\,0\,0\,0\,1\,1\,1\,1\,0\,1\,1\,1\,1\,0\,0\,0\,1\\
1\,0\,1\,0\,0\,0\,1\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,1\,1\,1\,1\,0\,1\,1\,0\,1\,0\,0\,0\,0\,1\,0\,0\,0\,1\,1\,0\\
1\,0\,1\,0\,0\,0\,0\,1\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,1\,1\,1\,1\,0\,1\,1\,0\,1\,0\,0\,0\,0\,1\,0\,0\,0\,1\,1\\
0\,0\,0\,0\,0\,0\,0\,0\,1\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,1\,0\,0\,0\,0\,0\,1\,1\,1\,1\,0\,1\,0\,0\,1\,0\,1\,1\,1\,1\\
1\,0\,1\,0\,0\,0\,0\,0\,1\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,1\,1\,1\,1\,1\,1\,1\,1\,1\,0\,0\,1\,1\,0\,1\,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\,0\,0\,1\,1\,0\,0\,0\,0\,0\,1\,1\,0\,1\,1\,1\,1\,1\,0\,1\,0\,1\,0\\
1\,0\,1\,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\,1\,1\,0\,1\,1\,1\,1\,1\,0\,1\,0\,1\\
0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,1\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,1\,0\,0\,0\,1\,1\,1\,0\,0\,0\,0\,1\,1\,1\,0\,0\,0\,1\,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\,1\,0\,0\,0\,1\,1\,1\,0\,0\,0\,0\,1\,1\,1\,0\,0\,0\,1\,0\\
0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,1\,0\,0\,0\,0\,0\,0\,0\,0\,0\,1\,0\,0\,0\,1\,1\,1\,0\,0\,0\,0\,1\,1\,1\,0\,0\,0\,0\,1\\
1\,0\,1\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,1\,0\,0\,0\,0\,0\,0\,0\,0\,1\,0\,1\,0\,1\,1\,1\,1\,1\,0\,1\,1\,0\,0\,0\,0\,0\,1\,1\,0\\
0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,1\,0\,0\,0\,0\,0\,0\,0\,1\,0\,1\,0\,1\,1\,1\,1\,1\,0\,1\,1\,0\,0\,0\,0\,0\,1\,1\,1\\
0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,1\,0\,0\,0\,0\,0\,0\,1\,0\,0\,1\,0\,1\,1\,0\,0\,1\,1\,1\,1\,1\,1\,1\,1\,1\,1\,1\\
1\,0\,1\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,1\,0\,0\,0\,0\,1\,1\,1\,0\,1\,0\,0\,1\,0\,1\,1\,1\,1\,0\,0\,0\,0\,0\,0\,1\\
1\,0\,1\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,1\,0\,0\,0\,0\,1\,1\,0\,0\,0\,1\,0\,0\,0\,0\,1\,0\,1\,1\,0\,1\,1\,1\,1\,1\,0\\
0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,1\,0\,0\,0\,0\,1\,1\,0\,0\,0\,1\,0\,0\,0\,0\,1\,0\,1\,1\,0\,1\,0\,1\,1\,1\,1\\
0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,1\,0\,0\,1\,0\,0\,0\,1\,1\,1\,1\,0\,1\,1\,1\,1\,0\,0\,0\,1\,0\,0\,1\\
0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,1\,0\,1\,1\,1\,1\,1\,0\,0\,1\,1\,1\,0\,0\,1\,1\,1\,1\,1\,1\,0\,1\,0\\
1\,0\,1\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,1\,0\,1\,1\,1\,1\,1\,0\,0\,1\,1\,1\,0\,0\,1\,1\,1\,1\,1\,1\,1\,0\,1
\end{pmatrix}$$

3. Apply Theorem 4.3 to **[44, 22, 8]** with parameters: $r = 2$, $a = (0\ 1)$, $b = (1\ 1)$ and $x = (0\,0\,1\,0\,1\,0\,1\,1\,0\,1\,0\,0\,0\,1\,1\,0\,0\,0\,0\,1\,1\,1\,0\,0\,1\,0\,0\,0\,0\,1\,0\,1\,1\,0\,1\,0\,1\,0\,0\,1\,0\,0\,1\,0)$,

[46, 23, 8]: $\begin{pmatrix}
0\,1\,0\,0\,1\,0\,1\,0\,1\,1\,0\,1\,0\,0\,0\,1\,1\,0\,0\,0\,0\,1\,1\,1\,0\,0\,1\,0\,0\,0\,0\,1\,0\,1\,1\,0\,1\,0\,1\,0\,0\,1\,0\,0\,1\,0\\
1\,1\,0\,1\,1\,0\,1\,0\,0\,0\,0\,0\,0\,1\,0\,0\,1\,0\,1\,0\,1\,1\,0\,0\,1\,0\,1\,1\,1\,1\,0\,0\,1\,1\,0\,0\,1\,0\,0\,0\,1\,0\,1\,0\,0\,1\\
0\,0\,1\,0\,1\,1\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,1\,0\,1\,1\,1\,1\,1\,0\,0\,1\,1\,1\,0\,0\,1\,1\,1\,1\,1\,0\\
1\,1\,0\,0\,0\,0\,1\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,1\,0\,1\,1\,1\,1\,1\,0\,0\,1\,1\,1\,0\,0\,1\,1\,1\,1\,1\\
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\,1\,0\,0\,1\,0\,0\,0\,1\,1\,1\,1\,0\,1\,1\,1\,1\,0\,0\,0\,1\\
0\,0\,1\,0\,1\,0\,0\,0\,1\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,1\,1\,1\,1\,0\,1\,1\,0\,1\,0\,0\,0\,0\,1\,0\,0\,0\,1\,1\,0\\
0\,0\,1\,0\,1\,0\,0\,0\,0\,1\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,1\,1\,1\,1\,0\,1\,1\,0\,1\,0\,0\,0\,0\,1\,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\,0\,0\,0\,0\,0\,0\,0\,1\,0\,0\,0\,0\,0\,1\,1\,1\,1\,0\,1\,0\,0\,1\,1\,1\,1\,0\,1\\
1\,1\,1\,0\,1\,0\,0\,0\,0\,0\,0\,1\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,1\,1\,1\,1\,1\,1\,1\,1\,1\,0\,0\,1\,1\,0\,1\,0\,1\,0\,0\,1\\
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\,1\,1\,0\,0\,0\,0\,0\,1\,1\,0\,1\,1\,1\,1\,1\,1\,0\,1\,0\,1\\
0\,0\,1\,0\,1\,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\,1\,1\,0\,1\,1\,1\,1\,1\,0\,1\,0\,1\\
1\,1\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,1\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,1\,0\,0\,0\,1\,1\,1\,0\,0\,0\,0\,1\,1\,1\,0\,0\,0\,1\,0\,0\\
1\,1\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,1\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,1\,0\,0\,0\,1\,1\,1\,0\,0\,0\,0\,1\,1\,1\,0\,0\,0\,0\,1\,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\,1\,0\,0\,0\,0\,1\,1\,1\,0\,0\,0\,0\,1\,1\,1\,0\,0\,0\,0\,1\\
1\,1\,1\,0\,1\,0\,0\,0\,0\,0\,0\,0\,0\,0\,1\,0\,0\,0\,0\,0\,0\,0\,0\,1\,0\,1\,0\,1\,1\,1\,1\,1\,0\,1\,1\,0\,0\,0\,0\,0\,0\,1\,1\,0\\
1\,1\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,1\,0\,0\,0\,0\,0\,0\,0\,0\,1\,0\,1\,0\,1\,1\,1\,1\,1\,0\,1\,1\,0\,0\,0\,0\,0\,0\,1\,1\\
1\,1\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,1\,0\,0\,0\,0\,0\,0\,0\,1\,0\,0\,1\,0\,1\,0\,1\,1\,0\,0\,1\,1\,1\,1\,1\,1\,1\,1\\
0\,0\,1\,0\,1\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,1\,0\,0\,0\,0\,0\,1\,1\,1\,1\,0\,1\,0\,0\,1\,0\,1\,1\,1\,1\,0\,0\,0\,0\,0\,1\\
0\,0\,1\,0\,1\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,1\,0\,0\,0\,0\,1\,1\,0\,0\,0\,1\,0\,0\,0\,0\,1\,0\,1\,1\,0\,1\,1\,1\,1\,1\,0\\
0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,1\,0\,0\,0\,0\,1\,1\,0\,0\,0\,1\,0\,0\,0\,0\,1\,0\,1\,1\,0\,1\,1\,1\,0\,1\,1\\
0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,1\,0\,0\,1\,0\,0\,0\,1\,1\,1\,1\,0\,1\,1\,1\,1\,0\,0\,0\,1\,0\,0\,1\,0\,0\,1\\
0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,1\,0\,1\,1\,1\,1\,1\,0\,0\,1\,1\,1\,0\,0\,1\,1\,1\,1\,1\,1\,1\,0\,1\,0\\
1\,1\,1\,0\,1\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,1\,0\,1\,1\,1\,1\,1\,0\,0\,1\,1\,1\,0\,0\,1\,1\,1\,1\,1\,1\,0\,1
\end{pmatrix}$

4. Apply Theorem 4.7 to **[46, 23, 8]** with parameters: $r = 1$ $m = 8$,
   $A = (0\,1\,1\,0\,1\,0\,0\,1)^T$
   $X = \begin{pmatrix}
   1\,1\,0\,0\,1\,1\,1\,1\,1\,0\,0\,1\,0\,0\,1\,0\,1\,1\,1\,1\,0\,0\,0\,0\,0\,1\,1\,0\,1\,0\,1\,1\,1\,1\,0\,1\,0\,1\,0\,1\,0\,0\,0\,0\,1\,0\,1\\
   1\,1\,1\,0\,1\,1\,1\,0\,1\,1\,0\,1\,0\,1\,0\,0\,1\,1\,0\,0\,0\,1\,1\,0\,1\,1\,1\,1\,0\,1\,1\,0\,0\,0\,0\,1\,1\,0\,1\,0\,1\,1\,0\,1\,1\,1\\
   1\,0\,1\,0\,1\,1\,0\,1\,1\,0\,0\,0\,0\,1\,1\,0\,0\,1\,1\,1\,0\,1\,1\,0\,0\,0\,0\,0\,1\,0\,1\,1\,1\,0\,1\,0\,1\,0\,0\,1\,0\,1\,1\,0\,0\,1\\
   1\,1\,0\,0\,0\,0\,1\,0\,0\,1\,1\,1\,1\,0\,0\,1\,1\,0\,0\,0\,1\,1\,0\,0\,0\,1\,1\,1\,1\,0\,0\,1\,1\,1\,0\,0\,1\,0\,1\,1\,0\,1\,0\,1\,1\,0\\
   0\,1\,0\,1\,0\,0\,0\,1\,0\,0\,1\,0\,0\,1\,1\,0\,0\,1\,1\,0\,1\,1\,1\,1\,0\,0\,0\,0\,1\,1\,0\,1\,0\,0\,1\,1\,1\,1\,1\,1\,1\,1\,1\,0\,1\,1\\
   0\,0\,1\,1\,0\,1\,1\,0\,1\,1\,0\,1\,0\,0\,1\,0\,0\,1\,0\,0\,1\,1\,1\,0\,0\,1\,0\,0\,1\,0\,1\,0\,0\,0\,1\,1\,1\,0\,1\,0\,0\,0\,0\,1\,1\,0\\
   1\,0\,1\,1\,1\,1\,0\,1\,1\,0\,0\,0\,1\,0\,0\,0\,0\,1\,1\,1\,1\,0\,1\,0\,1\,1\,0\,1\,0\,1\,0\,1\,0\,0\,1\,1\,1\,0\,1\,1\,0\,0\,0\,0\,1\,1\,0\\
   0\,1\,1\,1\,1\,0\,1\,1\,1\,0\,0\,0\,0\,0\,1\,1\,0\,0\,0\,0\,0\,1\,1\,0\,1\,1\,0\,0\,1\,0\,1\,0\,1\,0\,1\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,1
   \end{pmatrix}$,

[47, 31, 4]:

$$\begin{pmatrix}
0\,1\,1\,0\,0\,1\,1\,1\,1\,1\,0\,0\,1\,0\,0\,1\,0\,1\,1\,1\,0\,0\,0\,0\,0\,1\,1\,0\,1\,0\,1\,1\,1\,1\,0\,1\,0\,1\,0\,1\,0\,0\,0\,0\,1\,0\,1\\
1\,1\,1\,1\,0\,1\,1\,1\,0\,1\,1\,0\,1\,0\,1\,0\,0\,1\,1\,0\,0\,0\,1\,1\,0\,1\,1\,1\,1\,0\,1\,1\,0\,0\,0\,0\,0\,1\,1\,0\,1\,0\,1\,1\,0\,1\,1\,1\\
1\,1\,0\,1\,0\,1\,1\,0\,1\,1\,0\,0\,0\,0\,1\,1\,0\,0\,1\,1\,1\,0\,1\,1\,0\,0\,0\,0\,0\,1\,0\,1\,1\,1\,0\,1\,0\,1\,0\,0\,1\,0\,1\,1\,0\,0\,1\\
0\,1\,1\,0\,0\,0\,0\,1\,0\,0\,1\,1\,1\,1\,0\,0\,1\,1\,0\,0\,0\,1\,1\,0\,0\,0\,1\,1\,1\,1\,0\,0\,1\,1\,1\,0\,0\,1\,0\,1\,1\,0\,1\,0\,1\,1\,0\\
1\,0\,1\,0\,1\,0\,0\,0\,1\,0\,0\,1\,0\,0\,1\,1\,0\,0\,1\,1\,0\,1\,1\,1\,1\,0\,0\,0\,0\,1\,1\,0\,1\,0\,0\,1\,1\,1\,1\,1\,1\,1\,1\,1\,0\,1\,1\\
0\,0\,0\,1\,1\,0\,1\,1\,0\,1\,1\,0\,1\,0\,0\,1\,0\,0\,1\,0\,0\,1\,1\,1\,0\,0\,1\,0\,0\,1\,0\,1\,0\,0\,0\,1\,1\,1\,0\,1\,0\,0\,0\,0\,1\,1\,0\\
0\,1\,0\,1\,1\,1\,1\,0\,1\,1\,0\,0\,0\,1\,0\,0\,0\,0\,1\,1\,1\,1\,0\,1\,0\,1\,1\,0\,1\,0\,1\,0\,1\,0\,0\,1\,1\,1\,0\,0\,0\,0\,1\,1\,0\,1\,0\\
1\,0\,1\,1\,1\,1\,0\,1\,1\,1\,0\,0\,0\,0\,0\,1\,1\,0\,0\,0\,0\,0\,0\,1\,1\,0\,1\,1\,0\,0\,1\,0\,1\,0\,1\,0\,1\,0\,0\,0\,0\,0\,0\,0\,0\,0\,1\\
0\,0\,1\,0\,0\,1\,0\,1\,0\,1\,1\,0\,1\,0\,0\,0\,1\,1\,0\,0\,0\,0\,1\,1\,1\,0\,0\,1\,0\,0\,0\,0\,1\,0\,1\,1\,0\,1\,0\,1\,0\,0\,1\,0\,0\,1\,0\\
0\,1\,1\,0\,1\,1\,0\,1\,0\,0\,0\,0\,0\,0\,1\,0\,0\,1\,0\,1\,0\,1\,1\,0\,0\,1\,0\,1\,1\,1\,1\,0\,0\,1\,1\,0\,0\,1\,0\,0\,0\,1\,0\,1\,0\,0\,1\\
0\,0\,0\,1\,0\,1\,1\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,1\,0\,1\,1\,1\,1\,1\,0\,0\,1\,1\,1\,0\,0\,1\,1\,1\,1\,1\,0\\
0\,1\,1\,0\,0\,0\,0\,1\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,1\,0\,1\,1\,1\,1\,1\,0\,0\,1\,1\,1\,0\,0\,1\,1\,1\,1\,1\,1\\
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\,1\,0\,0\,1\,0\,0\,0\,1\,1\,1\,1\,0\,1\,1\,1\,1\,0\,0\,0\,1\\
0\,0\,0\,1\,0\,1\,0\,0\,0\,1\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,1\,1\,1\,1\,0\,1\,1\,0\,1\,0\,0\,0\,0\,1\,0\,0\,0\,1\,1\,0\\
0\,0\,0\,1\,0\,1\,0\,0\,0\,0\,1\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,1\,1\,1\,0\,1\,1\,0\,1\,0\,0\,0\,0\,1\,0\,0\,0\,1\,1\,0\,0\,1\,1\\
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\,1\,0\,0\,0\,1\,1\,1\,0\,1\,0\,0\,1\,0\,1\,1\,1\,1\,1\,0\,1\\
0\,1\,1\,1\,0\,1\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,1\,0\,1\,1\,1\,1\,1\,0\,0\,1\,1\,1\,0\,0\,1\,1\,1\,1\,1\,0\\
0\,1\,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\,1\,0\,1\,1\,1\,1\,1\,0\,1\,1\,0\,0\,0\,0\,0\,1\,1\\
0\,1\,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\,1\,0\,0\,1\,0\,1\,0\,1\,1\,0\,0\,1\,1\,1\,1\,1\,1\,1\,1\\
0\,0\,0\,1\,0\,1\,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\,1\,1\,1\,0\,1\,0\,0\,1\,0\,1\,1\,1\,1\,0\,0\,0\,0\,1\\
0\,0\,0\,1\,0\,1\,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\,1\,1\,0\,0\,0\,1\,0\,0\,0\,0\,1\,0\,1\,1\,0\,1\,1\,1\,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\,1\,0\,0\,0\,0\,1\,1\,0\,0\,0\,1\,0\,0\,0\,0\,1\,0\,1\,1\,1\,0\,1\,1\,1\,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\,1\,0\,0\,1\,0\,0\,0\,1\,1\,1\,0\,1\,1\,1\,1\,0\,0\,0\,1\,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\,1\,0\,1\,1\,1\,1\,1\,0\,0\,1\,1\,1\,0\,0\,1\,1\,1\,1\,1\,1\,0\,1\,0\\
0\,1\,1\,1\,0\,1\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,1\,0\,1\,1\,1\,1\,1\,0\,0\,1\,1\,1\,0\,0\,1\,1\,1\,1\,1\,1\,0\,1
\end{pmatrix}$$

5. Apply Lemma 4.1 **[47, 31, 8]**, shortening at position 29,

[46, 30, 5]:

$$\begin{pmatrix}
1000000000000000000000000000000010000111010011010\\
0100000000000000000000000000000000000000011010110011\\
0010000000000000000000010000001001100010100100 1\\
0001000000000000000000010000001001111011111001 11\\
0000100000000000000000000000000000000011000111 1001\\
0000010000000000000000000000010010000001001001\\
0000001000000000000000010000000000010100011110 00\\
0000000100000000000000010000000000100000011010 10\\
0000000010000000000000000000001000011100010001 0\\
0000000001000000000000010000000000000011010111 1000\\
0000000000100000000000010000000000011000001010 1\\
0000000000010000000000000000000010110111111100\\
0000000000001000000000000000010000111011000101\\
0000000000000100000000000000000111000111010011\\
0000000000000010000000100000010100111010111\\
0000000000000001000000100000010001000100110110\\
0000000000000000100001000000100001000101110 11\\
0000000000000000010000000000010001001100100 01\\
0000000000000000001000000000000111111101110 10\\
0000000000000000001001000000000010111011010\\
0000000000000000000101000000000010101001 01\\
0000000000000000000011000000001000001111111 1\\
0000000000000000000010000100111011010111 11\\
0000000000000000000000100000110000100000 01\\
0000000000000000000000010000011110110110 001\\
0000000000000000000000001001001010011100 010\\
0000000000000000000000000100001000011010 011\\
0000000000000000000000000010001100111101 0000\\
0000000000000000000000000001010101110101 000\\
0000000000000000000000000000111110001110 011
\end{pmatrix}$$

Example 4.11. Ternary $[38, 28, 5]$ LCD code generated from $[36, 28, 5]$

1. **[36, 28, 5]**:

$$\begin{pmatrix}
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&1&2&0&2&2&2&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&1&2&0&2&2&2\\
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&1&0&2&2&2&1&1&2\\
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&1&1&2&0&2&0&0&1\\
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&2&1&2&1&0&1&1&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&2&1&2&1&0&1&1&1\\
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&2&0&0&0&2&0&1&1\\
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&2&2&1&2&0&1&1&1\\
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&2&2&0&0&2&2&2&1\\
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&2&2&0&2&0&1&0&2\\
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&1&2&1&1&2&1&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&1&2&1&1&2&1&1&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&1&2&1&1&2&1&2&1\\
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&2&0&1&0&2&0&2&2\\
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&1&2&2&2&0&0&2&2\\
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&1&1&1&0&2&1&2&2\\
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&1&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&0&0&0&0&0&0&0&0&0&0&0&0&0&1&1&0&1&2&1&2&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&1&1&0&1&2&1&2&2\\
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&1&0&0&2&0&2&1&1\\
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&2&1&1&2&2&2&0&1\\
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&2&2&2&0&2&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&1&0&0&0&0&0&0&0&0&2&2&2&0&2&1&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&1&0&0&0&0&0&0&0&2&2&2&0&2&1&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&1&0&0&0&2&0&1&1&2&1&1&2&1&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&0&0&1&2&2&2&1&0&0&1&1&1&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&1&0&2&1&0&1&2&0&1&0&1&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&1&0&2&1&0&1&2&0&1&1&1&1
\end{pmatrix}$$

2. Apply Lemma 4.1 to **[36, 28, 5]**, shortening at position 29

[35, 27, 5]:

$$\begin{pmatrix}
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&1&0&0&0&0&1&2&1\\
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&1&2&0&2&2&2\\
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&1&2&2&2&1&1&2\\
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&1&1&0&0&0&2&0&2\\
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&1&2&0&1&1&0&1&1\\
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&2&0&0&2&0&1&1&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&2&0&0&2&0&1&1&1\\
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&2&2&0&2&2&2&1&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&2&2&2&0&1&0&2&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&2&2&2&2&2&0&1&1\\
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&2&1&0&1&1&2&0&2\\
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&1&0&0&1&2&1&2&1\\
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&1&2&1&1&2&1&2&1\\
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&2&1&0&2&0&2&2&2\\
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&1&1&2&0&0&2&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&1&1&1&2&1&2&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&1&1&1&0&0&2&1&\\
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&1&0&2&0&2&1&1&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&1&2&2&2&0&1&0&2\\
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&1&0&2&2&1&0&1&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&0&0&0&0&0&0&0&0&0&0&1&0&2&0&1&2&0&2\\
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&1&0&2&0&1&2&2&1\\
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&1&0&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&1&0&0&0&0&0&0&0&0&0&0&1&0&2&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&0&0&0&0&1&0&0&0&0&0&0&0&0&0&1&0&2&0&1&2&1&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&1&0&0&0&0&0&0&0&0&1&0&2&0&1&2&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&0&0&0&0&0&0&0&1&0&2&0&1&2&1&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&1&0&0&0&0&0&0&1&0&2&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&0&0&0&0&0&0&0&0&1&0&0&0&0&0&1&0&2&0&1&2&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&1&0&0&0&0&1&0&2&0&1&2&2&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&1&0&0&0&1&0&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&1&0&0&1&0&2&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&0&0&0&0&0&0&0&0&0&0&0&0&1&0&1&0&2&0&1&2&1&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&1&1&0&2&0&1&2&2&0
\end{pmatrix}$$

3. Apply Theorem 4.3 to **[35, 27, 5]** with parameters $a = (2\ 1\ 0)$, $b = (2\ 1\ 2)$ and $x = (2\ 2\ 2\ 1\ 1\ 0\ 2\ 0\ 2\ 2\ 2\ 2\ 0\ 1\ 2\ 0\ 2\ 2\ 1\ 1\ 0\ 2\ 0\ 2\ 1\ 1\ 0\ 2\ 1\ 1\ 1\ 1\ 0\ 1\ 1)$

[38, 28, 5]:

$$\begin{pmatrix}
2\,1\,0\,2\,2\,2\,1\,1\,0\,2\,0\,2\,2\,2\,2\,0\,1\,2\,0\,2\,2\,1\,1\,0\,2\,0\,2\,1\,1\,0\,2\,1\,1\,1\,1\,0\,1\,1\\
2\,1\,2\,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\,1\,0\,0\,0\,0\,1\,2\,1\\
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\,1\,2\,0\,2\,2\,2\\
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\,1\,2\,2\,2\,1\,1\,2\\
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\,1\,1\,0\,0\,0\,2\,0\,2\\
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\,1\,2\,0\,1\,1\,0\,1\,1\\
2\,1\,2\,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\,2\,0\,0\,2\,0\,1\,1\,0\\
1\,2\,1\,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\,2\,0\,0\,2\,0\,1\,1\\
1\,2\,1\,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\,2\,2\,0\,2\,2\,2\,1\,0\\
2\,1\,2\,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\,2\,2\,2\,0\,1\,0\,2\,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\,2\,2\,2\,2\,2\,2\,0\,1\\
1\,2\,1\,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\,2\,1\,0\,1\,1\,2\,0\,2\\
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\,1\,0\,0\,1\,2\,1\,1\,1\\
2\,1\,2\,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\,1\,2\,1\,1\,2\,1\\
2\,1\,2\,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\,2\,1\,0\,2\,0\,2\,2\\
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\,2\,1\,1\,2\,2\,1\,2\,1\\
2\,1\,2\,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\,1\,1\,2\,0\,0\,0\,2\,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\,1\,1\,1\,2\,1\,2\,0\,0\\
2\,1\,2\,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\,1\,1\,1\,1\,0\,0\,2\,1\\
1\,2\,1\,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\,1\,0\,2\,0\,2\,1\,1\,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\,1\,0\,2\,0\,2\,1\,1\\
2\,1\,2\,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\,1\,2\,2\,2\,0\,1\,0\,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\,0\,0\,0\,0\,0\,0\,0\,0\,0\,2\,2\,1\,0\,1\,2\,0\,2\\
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\,2\,0\,1\,2\,2\,0\,1\,2\\
1\,2\,1\,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\,2\,2\,2\,0\,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\,0\,0\,0\,2\,1\,1\,2\,1\,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\,1\,0\,2\,1\,1\,2\,0\,1\,0\,0\\
1\,2\,1\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,1\,1\,2\,1\,1\,0\,2\,1\,1
\end{pmatrix}$$

4.3.1 Binary Generated LCD Codes

In the following, we list the best parameters of the binary LCD codes that we have found. Note that a code marked with * indicates that the maximum possible minimum weight has been achieved.

[26, 15, 5*], [28, 17, 4], [29, 11, 8], [30, 11, 8], [30, 12, 9*], [30, 15, 6], [30, 19, 4]
[31, 11, 10], [31, 12, 10], [31, 13, 9*], [31, 15, 7*], [31, 16, 6], [31, 21, 5], [32, 12, 10]
[32, 13, 8], [32, 14, 7], [32, 15, 7], [32, 16, 6], [32, 17, 6], [32, 21, 4], [32, 22, 5]
[32, 25, 3*], [33, 11, 10], [33, 13, 9*], [33, 14, 7], [33, 15, 7], [33, 16, 6], [33, 17, 6]
[33, 18, 6], [33, 21, 4], [33, 23, 5], [34, 9, 13*], [34, 11, 10], [34, 13, 9], [34, 14, 8]
[34, 15, 7], [34, 16, 7], [34, 17, 6], [34, 18, 6], [34, 19, 6], [34, 23, 4], [35, 9, 12]
[35, 10, 11], [35, 13, 10], [35, 15, 8], [35, 16, 9*], [35, 17, 7], [35, 18, 6], [35, 19, 6]
[35, 20, 5], [35, 21, 5], [35, 22, 6*], [35, 23, 5], [35, 24, 4], [36, 9, 13*], [36, 10, 11]
[36, 11, 10], [36, 13, 10], [36, 14, 10], [36, 15, 8], [36, 16, 10*], [36, 17, 7], [36, 18, 6]
[36, 19, 6], [36, 20, 5], [36, 21, 5], [36, 22, 5], [36, 24, 6], [36, 25, 4], [37, 9, 13]
[37, 10, 12], [37, 11, 11], [37, 12, 10], [37, 13, 9], [37, 14, 10], [37, 15, 9], [37, 17, 7]
[37, 18, 7], [37, 19, 6], [37, 20, 6], [37, 21, 6], [37, 22, 5], [37, 23, 5], [37, 24, 5]
[37, 25, 5], [37, 26, 4], [38, 9, 13], [38, 11, 11], [38, 12, 10], [38, 13, 10], [38, 14, 10]
[38, 15, 9], [38, 16, 10], [38, 17, 8], [38, 18, 9], [38, 19, 7], [38, 20, 6], [38, 21, 6]
[38, 22, 5], [38, 23, 6], [38, 24, 6*], [38, 25, 5], [38, 26, 4], [38, 27, 4], [39, 9, 13]
[39, 10, 13], [39, 11, 13], [39, 12, 11], [39, 13, 10], [39, 14, 10], [39, 15, 10], [39, 16, 10]
[39, 17, 10], [39, 18, 9], [39, 19, 9], [39, 20, 7], [39, 21, 6], [39, 22, 6], [39, 23, 6]
[39, 24, 6], [39, 25, 5], [39, 26, 5], [39, 27, 4], [39, 28, 4], [40, 9, 13], [40, 10, 13]
[40, 11, 12], [40, 12, 13], [40, 13, 11], [40, 14, 10], [40, 15, 10], [40, 16, 10], [40, 17, 10]
[40, 18, 10], [40, 19, 9], [40, 20, 9], [40, 21, 7], [40, 22, 6], [40, 23, 5], [40, 24, 6]
[40, 25, 6], [40, 26, 5], [40, 27, 4], [40, 28, 4], [40, 29, 4], [41, 10, 12], [41, 11, 12]
[41, 12, 11], [41, 13, 13], [41, 14, 10], [41, 15, 9], [41, 16, 9], [41, 17, 9], [41, 18, 10]
[41, 19, 9], [41, 21, 9*], [41, 22, 6], [41, 24, 5], [41, 25, 6], [41, 26, 5], [41, 29, 4]
[41, 30, 3], [42, 11, 13], [42, 12, 12], [42, 13, 12], [42, 14, 13], [42, 15, 10], [42, 16, 9]
[42, 17, 10], [42, 18, 10], [42, 19, 9], [42, 21, 9*], [42, 22, 7], [42, 23, 6], [42, 25, 6]
[42, 26, 6], [42, 27, 5], [42, 30, 3], [42, 31, 4], [43, 9, 14], [43, 10, 13], [43, 11, 13]
[43, 12, 12], [43, 13, 11], [43, 15, 13*], [43, 16, 9], [43, 17, 9], [43, 18, 9], [43, 19, 9]
[43, 20, 10], [43, 22, 7], [43, 23, 7], [43, 24, 6], [43, 25, 6], [43, 26, 6], [43, 27, 5]

[43, 28, 5], [43, 31, 4], [43, 32, 3], [44, 7, 17], [44, 8, 17], [44, 10, 14], [44, 11, 13]
[44, 12, 12], [44, 13, 12], [44, 14, 13], [44, 15, 11], [44, 16, 13], [44, 17, 9], [44, 18, 9]
[44, 19, 9], [44, 20, 9], [44, 21, 9], [44, 22, 8], [44, 23, 7], [44, 24, 6], [44, 25, 6]
[44, 27, 5], [44, 28, 5], [44, 29, 4], [44, 31, 4], [44, 32, 5*], [44, 33, 3], [45, 8, 17]
[45, 9, 15], [45, 10, 14], [45, 11, 13], [45, 12, 12], [45, 13, 12], [45, 14, 14], [45, 15, 13]
[45, 17, 10], [45, 18, 9], [45, 19, 8], [45, 20, 8], [45, 21, 9], [45, 22, 9], [45, 23, 7]
[45, 24, 6], [45, 25, 6], [45, 26, 6], [45, 27, 5], [45, 28, 5], [45, 29, 5], [45, 30, 5]
[45, 33, 3], [45, 34, 3], [46, 8, 18], [46, 9, 17], [46, 10, 15], [46, 11, 14], [46, 12, 13]
[46, 13, 12], [46, 14, 12], [46, 15, 13], [46, 16, 14], [46, 17, 11], [46, 18, 10], [46, 19, 10]
[46, 20, 10], [46, 21, 9], [46, 22, 9], [46, 23, 8], [46, 24, 7], [46, 25, 6], [46, 26, 6]
[46, 27, 6], [46, 29, 5], [46, 30, 5], [46, 31, 4], [46, 33, 4], [46, 34, 5*], [46, 35, 3]
[47, 7, 17], [47, 8, 16], [47, 9, 16], [47, 10, 15], [47, 11, 13], [47, 12, 13], [47, 13, 12]
[47, 14, 11], [47, 15, 13], [47, 16, 14], [47, 17, 11], [47, 18, 10], [47, 19, 9], [47, 20, 9]
[47, 21, 10], [47, 22, 9], [47, 23, 8], [47, 24, 8], [47, 25, 7], [47, 26, 6], [47, 27, 6]
[47, 28, 5], [47, 29, 5], [47, 30, 5], [47, 31, 5], [47, 32, 4], [47, 35, 5*], [47, 36, 3]
[48, 7, 18], [48, 8, 17], [48, 9, 17], [48, 10, 16], [48, 11, 15], [48, 12, 13], [48, 13, 12]
[48, 14, 14], [48, 15, 13], [48, 16, 14], [48, 17, 11], [48, 18, 10], [48, 19, 10], [48, 20, 9]
[48, 21, 10], [48, 22, 9], [48, 23, 9], [48, 24, 8], [48, 25, 7], [48, 26, 7], [48, 27, 6]
[48, 28, 6], [48, 29, 6], [48, 31, 5], [48, 32, 5], [48, 33, 4], [48, 35, 3], [48, 36, 5*]
[48, 37, 3], [49, 7, 18], [49, 8, 18], [49, 9, 18], [49, 10, 16], [49, 11, 16], [49, 12, 15]
[49, 13, 17], [49, 14, 13], [49, 15, 12], [49, 16, 14], [49, 17, 13], [49, 18, 11], [49, 19, 10]
[49, 20, 10], [49, 21, 9], [49, 22, 9], [49, 23, 9], [49, 24, 8], [49, 25, 8], [49, 26, 7]
[49, 27, 7], [49, 28, 6], [49, 29, 6], [49, 31, 5], [49, 32, 5], [49, 33, 5], [49, 34, 4]
[49, 37, 5*], [49, 38, 3], [50, 7, 19], [50, 8, 18], [50, 9, 18], [50, 10, 17], [50, 11, 16]
[50, 12, 15], [50, 13, 14], [50, 14, 17], [50, 15, 12], [50, 16, 12], [50, 17, 13], [50, 18, 12]
[50, 19, 10], [50, 20, 10], [50, 21, 9], [50, 22, 9], [50, 23, 9], [50, 24, 9], [50, 25, 8]
[50, 26, 7], [50, 27, 7], [50, 28, 7], [50, 29, 6], [50, 30, 6], [50, 32, 5], [50, 33, 4]
[50, 34, 4], [50, 35, 4], [50, 37, 5*], [50, 38, 5*], [51, 6, 21], [51, 7, 19], [51, 8, 19]

[51, 9, 17], [51, 10, 17], [51, 11, 15], [51, 12, 15], [51, 13, 17], [51, 14, 18], [51, 15, 12]
[51, 16, 13], [51, 17, 14], [51, 18, 12], [51, 19, 11], [51, 20, 10], [51, 21, 10], [51, 22, 10]
[51, 23, 9], [51, 24, 8], [51, 25, 8], [51, 26, 8], [51, 27, 7], [51, 28, 7], [51, 29, 7]
[51, 31, 5], [51, 32, 5], [51, 33, 5], [51, 34, 5], [51, 35, 4], [51, 36, 5], [51, 37, 5]
[51, 38, 5], [51, 39, 5*], [51, 40, 3], [51, 41, 3], [51, 42, 3], [51, 43, 3], [52, 6, 21]
[52, 7, 20], [52, 8, 19], [52, 9, 18], [52, 10, 17], [52, 11, 17], [52, 12, 15], [52, 13, 15]
[52, 14, 15], [52, 15, 13], [52, 16, 13], [52, 17, 14], [52, 18, 12], [52, 19, 12], [52, 20, 11]
[52, 21, 10], [52, 22, 10], [52, 23, 9], [52, 24, 9], [52, 25, 9], [52, 26, 8], [52, 27, 8]
[52, 28, 7], [52, 32, 5], [52, 33, 5], [52, 34, 5], [52, 35, 4], [52, 36, 5], [52, 37, 5]
[52, 38, 5], [52, 39, 5], [52, 41, 3], [52, 42, 3], [52, 43, 3], [52, 44, 3], [53, 6, 22]
[53, 7, 21], [53, 8, 20], [53, 9, 18], [53, 10, 18], [53, 11, 17], [53, 12, 16], [53, 13, 15]
[53, 14, 17], [53, 15, 14], [53, 16, 13], [53, 17, 14], [53, 18, 13], [53, 19, 13], [53, 20, 11]
[53, 21, 10], [53, 22, 10], [53, 23, 9], [53, 24, 9], [53, 25, 9], [53, 26, 8], [53, 27, 8]
[53, 30, 9], [53, 31, 7], [53, 32, 6], [53, 33, 6], [53, 34, 5], [53, 35, 5], [53, 36, 4]
[53, 37, 6], [53, 38, 5], [53, 39, 5], [53, 40, 4], [53, 41, 5], [53, 42, 3], [53, 43, 3]
[53, 44, 3], [53, 45, 3], [54, 6, 23], [54, 7, 20], [54, 8, 20], [54, 9, 19], [54, 10, 18]
[54, 11, 17], [54, 12, 16], [54, 14, 18], [54, 15, 16], [54, 16, 14], [54, 17, 13], [54, 18, 12]
[54, 19, 12], [54, 20, 11], [54, 21, 10], [54, 22, 10], [54, 23, 10], [54, 24, 9], [54, 25, 9]
[54, 26, 9], [54, 27, 8], [54, 30, 10], [54, 31, 9], [54, 32, 6], [54, 33, 6], [54, 34, 6]
[54, 35, 5], [54, 36, 5], [54, 37, 6], [54, 38, 5], [54, 39, 5], [54, 40, 4], [54, 41, 4]
[54, 42, 3], [54, 43, 3], [54, 44, 3], [54, 45, 3], [54, 46, 2], [55, 6, 23], [55, 7, 21]
[55, 8, 20], [55, 9, 19], [55, 10, 18], [55, 11, 17], [55, 12, 17], [55, 14, 18], [55, 15, 15]
[55, 16, 14], [55, 17, 13], [55, 18, 12], [55, 19, 12], [55, 20, 11], [55, 21, 11], [55, 22, 11]
[55, 23, 10], [55, 24, 9], [55, 25, 9], [55, 26, 9], [55, 27, 8], [55, 31, 7], [55, 32, 6]
[55, 33, 6], [55, 34, 6], [55, 35, 6], [55, 36, 5], [55, 37, 6], [55, 38, 6], [55, 39, 5]
[55, 40, 5], [55, 41, 4], [55, 42, 4], [55, 44, 3], [55, 45, 3], [55, 46, 3], [55, 47, 3]
[56, 6, 23], [56, 7, 21], [56, 8, 21], [56, 9, 19], [56, 10, 18], [56, 11, 18], [56, 12, 16]
[56, 14, 18], [56, 15, 15], [56, 16, 15], [56, 17, 14], [56, 18, 12], [56, 20, 12], [56, 21, 11]

[56, 22, 11], [56, 23, 11], [56, 24, 10], [56, 25, 9], [56, 26, 9], [56, 27, 8], [56, 30, 10]
[56, 31, 9], [56, 32, 7], [56, 33, 7], [56, 34, 6], [56, 35, 6], [56, 36, 5], [56, 37, 6]
[56, 38, 6], [56, 39, 5], [56, 40, 5], [56, 41, 5], [56, 42, 5], [56, 43, 4], [56, 44, 3]
[56, 45, 3], [56, 46, 3], [56, 47, 3], [56, 48, 2], [57, 6, 23], [57, 7, 21], [57, 8, 20]
[57, 9, 20], [57, 10, 18], [57, 11, 18], [57, 12, 18], [57, 15, 17], [57, 16, 15], [57, 17, 14]
[57, 18, 13], [57, 19, 12], [57, 20, 12], [57, 21, 11], [57, 22, 11], [57, 23, 10], [57, 24, 10]
[57, 25, 10], [57, 26, 9], [57, 27, 9], [57, 28, 8], [57, 30, 10], [57, 31, 9], [57, 32, 9]
[57, 33, 7], [57, 34, 7], [57, 35, 6], [57, 36, 6], [57, 37, 6], [57, 38, 6], [57, 39, 6]
[57, 40, 5], [57, 41, 5], [57, 42, 4], [57, 43, 5], [57, 44, 4], [57, 45, 3], [57, 46, 3]
[57, 47, 3], [57, 48, 3], [57, 49, 2], [58, 6, 24], [58, 7, 21], [58, 8, 19], [58, 9, 20]
[58, 10, 18], [58, 11, 18], [58, 12, 17], [58, 15, 17], [58, 16, 14], [58, 17, 15], [58, 18, 13]
[58, 19, 13], [58, 20, 12], [58, 21, 12], [58, 22, 11], [58, 23, 11], [58, 24, 10], [58, 25, 9]
[58, 26, 9], [58, 27, 9], [58, 28, 9], [58, 29, 8], [58, 31, 9], [58, 32, 9], [58, 33, 7]
[58, 34, 7], [58, 35, 6], [58, 36, 6], [58, 37, 6], [58, 38, 6], [58, 39, 6], [58, 40, 5]
[58, 41, 5], [58, 42, 5], [58, 43, 5], [58, 44, 4], [58, 45, 4], [58, 46, 3], [58, 47, 3]
[58, 48, 3], [58, 49, 3], [58, 50, 3], [59, 6, 25], [59, 7, 22], [59, 8, 21], [59, 9, 20]
[59, 10, 19], [59, 11, 18], [59, 12, 18], [59, 15, 18], [59, 16, 15], [59, 17, 15], [59, 18, 14]
[59, 19, 14], [59, 20, 13], [59, 21, 12], [59, 22, 12], [59, 23, 12], [59, 24, 11], [59, 25, 10]
[59, 26, 10], [59, 27, 9], [59, 28, 9], [59, 29, 8], [59, 30, 8], [59, 31, 10], [59, 32, 9]
[59, 33, 8], [59, 34, 8], [59, 35, 7], [59, 36, 6], [59, 37, 6], [59, 38, 6], [59, 39, 6]
[59, 40, 5], [59, 41, 5], [59, 42, 5], [59, 43, 5], [59, 44, 5], [59, 45, 4], [59, 46, 3]
[59, 47, 3], [59, 48, 3], [59, 49, 3], [59, 50, 3], [59, 51, 2], [60, 6, 25], [60, 7, 22]
[60, 8, 21], [60, 9, 20], [60, 10, 20], [60, 11, 18], [60, 12, 18], [60, 15, 17], [60, 16, 17]
[60, 17, 16], [60, 18, 15], [60, 19, 13], [60, 20, 13], [60, 21, 13], [60, 22, 12], [60, 23, 12]
[60, 24, 11], [60, 25, 10], [60, 26, 10], [60, 27, 9], [60, 28, 9], [60, 29, 9], [60, 30, 7]
[60, 31, 10], [60, 32, 10], [60, 33, 9], [60, 34, 8], [60, 35, 7], [60, 36, 6], [60, 37, 6]
[60, 38, 6], [60, 39, 6], [60, 40, 5], [60, 41, 5], [60, 42, 4], [60, 43, 5], [60, 44, 5]
[60, 45, 5], [60, 46, 4], [60, 47, 4], [60, 48, 3], [60, 49, 3], [60, 50, 3], [60, 52, 2]

4.3.2 Ternary Generated LCD Codes

In the following, we list the best parameters of the ternary LCD codes that we have found. Note that a code marked with * indicates that the maximum possible minimum weight has been achieved.

[21, 4, 12*], [21, 5, 11], [21, 5, 10], [21, 6, 11], [21, 6, 9], [21, 7, 10], [21, 7, 8]
[21, 8, 8], [21, 8, 7], [21, 9, 7], [21, 10, 7], [21, 11, 4], [21, 12, 5], [21, 13, 4]
[21, 14, 5], [22, 4, 12], [22, 5, 11], [22, 6, 11], [22, 7, 9], [22, 8, 8], [22, 9, 8]
[22, 10, 7], [22, 11, 6], [22, 12, 6], [22, 13, 5], [22, 14, 5*], [22, 15, 4*], [23, 4, 13]
[23, 5, 12], [23, 6, 11], [23, 7, 10], [23, 8, 9], [23, 9, 8], [23, 10, 7], [23, 11, 7]
[23, 12, 6], [23, 13, 6], [23, 14, 5], [23, 15, 5*], [23, 16, 4*], [24, 4, 13], [24, 5, 12]
[24, 6, 11], [24, 7, 10], [24, 8, 9], [24, 9, 8], [24, 10, 8], [24, 11, 7], [24, 12, 7]
[24, 13, 6], [24, 14, 5], [24, 15, 5], [24, 16, 5*], [24, 17, 4*], [25, 4, 14], [25, 5, 13]
[25, 6, 12], [25, 7, 11], [25, 8, 10], [25, 9, 9], [25, 10, 8], [25, 11, 8], [25, 12, 7]
[25, 13, 6], [25, 14, 6], [25, 15, 6*], [25, 16, 5], [25, 17, 4], [25, 18, 4*], [26, 4, 15]
[26, 5, 14], [26, 6, 12], [26, 7, 11], [26, 8, 10], [26, 9, 9], [26, 10, 9], [26, 11, 8]
[26, 12, 9], [26, 13, 8], [26, 14, 6], [26, 15, 6], [26, 16, 5], [26, 17, 5], [26, 18, 4]
[26, 19, 4*], [27, 4, 16*], [27, 5, 14], [27, 6, 13], [27, 7, 12], [27, 8, 11], [27, 9, 10]
[27, 10, 9], [27, 11, 8], [27, 12, 8], [27, 13, 8], [27, 14, 7], [27, 15, 6], [27, 16, 6]
[27, 17, 5], [27, 18, 5], [27, 19, 4], [27, 20, 4*], [28, 4, 16], [28, 5, 16*], [28, 6, 14]
[28, 7, 13], [28, 8, 13], [28, 9, 11], [28, 10, 10], [28, 11, 9], [28, 12, 10], [28, 13, 7]
[28, 14, 8], [28, 15, 8], [28, 16, 6], [28, 17, 5], [28, 18, 5], [28, 19, 4], [28, 20, 4]
[28, 21, 4*], [29, 4, 17], [29, 5, 16], [29, 6, 14], [29, 7, 13], [29, 8, 12], [29, 9, 13]
[29, 10, 10], [29, 11, 9], [29, 12, 10], [29, 13, 8], [29, 14, 8], [29, 15, 7], [29, 16, 7]
[29, 17, 6], [29, 18, 5], [29, 19, 5], [29, 20, 4], [29, 21, 4], [29, 22, 4*], [30, 4, 19*]
[30, 5, 16], [30, 6, 14], [30, 7, 13], [30, 8, 13], [30, 9, 12], [30, 10, 11], [30, 11, 10]
[30, 12, 9], [30, 13, 9], [30, 14, 8], [30, 15, 7], [30, 16, 7], [30, 17, 6], [30, 18, 6]
[30, 19, 5], [30, 20, 5], [30, 21, 4], [30, 22, 5*], [30, 23, 4*], [31, 4, 19*], [31, 5, 17]
[31, 6, 16], [31, 7, 14], [31, 8, 13], [31, 9, 13], [31, 10, 12], [31, 11, 11], [31, 12, 10]
[31, 13, 10], [31, 14, 9], [31, 15, 8], [31, 16, 7], [31, 17, 7], [31, 18, 6], [31, 19, 6]

[31, 20, 5], [31, 21, 5], [31, 22, 4], [31, 23, 4], [31, 24, 4^*], [32, 4, 19], [32, 5, 17]
[32, 6, 16], [32, 7, 15], [32, 8, 13], [32, 9, 13], [32, 10, 13], [32, 11, 11], [32, 12, 10]
[32, 13, 10], [32, 14, 10], [32, 15, 8]
[32, 16, 8], [32, 17, 7], [32, 18, 7], [32, 19, 6], [32, 20, 5], [32, 21, 5], [32, 22, 5]
[32, 23, 5], [32, 24, 5^*], [32, 25, 4^*], [33, 4, 19], [33, 5, 18], [33, 6, 16], [33, 7, 15]
[33, 8, 14], [33, 9, 13], [33, 10, 14], [33, 11, 12], [33, 12, 11], [33, 13, 10], [33, 14, 10]
[33, 15, 9], [33, 16, 8], [33, 17, 8], [33, 18, 7], [33, 19, 7], [33, 20, 6], [33, 21, 5]
[33, 22, 5], [33, 23, 5], [33, 24, 5], [33, 25, 4^*], [33, 26, 4^*], [34, 4, 20], [34, 5, 18]
[34, 6, 17], [34, 7, 16], [34, 8, 15], [34, 9, 14], [34, 10, 13], [34, 11, 12], [34, 12, 11]
[34, 13, 10], [34, 14, 10], [34, 15, 9], [34, 16, 9], [34, 17, 8], [34, 18, 8], [34, 19, 7]
[34, 20, 7], [34, 21, 6], [34, 23, 5], [34, 24, 5], [34, 25, 5^*], [34, 26, 4^*], [34, 27, 4^*]
[35, 4, 22^*], [35, 5, 19], [35, 6, 18], [35, 7, 19], [35, 8, 15], [35, 9, 14], [35, 10, 13]
[35, 11, 13], [35, 12, 12], [35, 13, 11], [35, 14, 10], [35, 15, 10], [35, 16, 9], [35, 17, 8]
[35, 18, 8], [35, 19, 8], [35, 20, 7], [35, 21, 6], [35, 22, 7], [35, 23, 6], [35, 24, 5]
[35, 25, 5], [35, 26, 4], [35, 27, 4^*], [35, 28, 4^*], [36, 4, 21], [36, 5, 20], [36, 6, 18]
[36, 7, 19], [36, 8, 16], [36, 9, 15], [36, 10, 14], [36, 11, 13], [36, 12, 13], [36, 13, 11]
[36, 14, 11], [36, 15, 10], [36, 16, 9], [36, 17, 9], [36, 18, 8], [36, 19, 8], [36, 20, 7]
[36, 21, 7], [36, 22, 7], [36, 23, 7], [36, 24, 5], [36, 25, 5], [36, 26, 4], [36, 27, 5^*]
[36, 28, 4^*], [36, 29, 4^*], [37, 4, 22], [37, 5, 20], [37, 6, 18], [37, 7, 17], [37, 8, 17]
[37, 9, 15], [37, 10, 14], [37, 11, 14], [37, 12, 13], [37, 13, 12], [37, 14, 11], [37, 15, 10]
[37, 16, 10], [37, 17, 9], [37, 18, 11], [37, 19, 10], [37, 20, 8], [37, 21, 7], [37, 22, 6]
[37, 24, 7], [37, 25, 5], [37, 26, 6^*], [37, 27, 5], [37, 28, 5^*], [37, 30, 4^*], [38, 4, 23]
[38, 5, 22], [38, 6, 19], [38, 7, 19], [38, 8, 17], [38, 9, 16], [38, 10, 15], [38, 11, 14]
[38, 12, 13], [38, 13, 12], [38, 14, 11], [38, 15, 11], [38, 16, 10], [38, 17, 9], [38, 18, 10]
[38, 19, 10], [38, 20, 8], [38, 21, 7], [38, 22, 7], [38, 23, 8], [38, 24, 7], [38, 25, 7]
[38, 26, 5], [38, 27, 6^*], [38, 28, 5], [38, 29, 5^*], [38, 30, 4^*], [38, 31, 4^*], [39, 4, 23]
[39, 5, 21], [39, 6, 22], [39, 7, 19], [39, 8, 17], [39, 9, 16], [39, 10, 15], [39, 11, 15]
[39, 12, 14], [39, 13, 13], [39, 14, 12], [39, 15, 11], [39, 16, 11], [39, 17, 10], [39, 18, 9]
[39, 19, 10], [39, 20, 9], [39, 21, 8], [39, 22, 7], [39, 23, 7], [39, 24, 8], [39, 25, 7]
[39, 26, 7], [39, 27, 5], [39, 28, 6^*], [39, 29, 5], [39, 30, 5^*], [39, 31, 4^*], [39, 32, 4^*]
[40, 4, 24], [40, 5, 22]

[40, 6, 22], [40, 7, 22], [40, 8, 18], [40, 9, 17], [40, 10, 16], [40, 11, 15], [40, 12, 14]
[40, 13, 13], [40, 14, 12], [40, 15, 12], [40, 16, 11], [40, 17, 10], [40, 18, 11], [40, 19, 10]
[40, 20, 10], [40, 21, 9], [40, 22, 8], [40, 23, 7], [40, 24, 7], [40, 25, 8], [40, 26, 7]
[40, 27, 7], [40, 28, 5], [40, 29, 6*], [40, 31, 5*], [40, 32, 4*], [40, 33, 4*], [41, 4, 25]
[41, 5, 22], [41, 6, 21], [41, 7, 22], [41, 8, 22], [41, 9, 17], [41, 10, 16], [41, 11, 16]
[41, 12, 14], [41, 13, 14], [41, 14, 13], [41, 15, 12], [41, 16, 12], [41, 17, 11], [41, 18, 10]
[41, 19, 11], [41, 20, 10], [41, 21, 9], [41, 22, 8], [41, 23, 8], [41, 24, 7], [41, 25, 8]
[41, 26, 7], [41, 27, 7], [41, 28, 7], [41, 29, 5], [41, 30, 6*], [41, 31, 5], [41, 32, 5*]
[41, 33, 4*], [41, 34, 3], [42, 4, 25], [42, 5, 23], [42, 6, 21], [42, 7, 20], [42, 8, 19]
[42, 9, 18], [42, 10, 17], [42, 11, 16], [42, 12, 15], [42, 13, 14], [42, 14, 14], [42, 15, 13]
[42, 16, 12], [42, 17, 11], [42, 18, 11], [42, 19, 11], [42, 20, 10], [42, 21, 10], [42, 22, 9]
[42, 23, 8], [42, 24, 7], [42, 25, 7], [42, 26, 8], [42, 27, 7], [42, 28, 7], [42, 29, 6]
[42, 30, 6], [42, 31, 6*], [42, 32, 5], [42, 33, 5*], [42, 34, 4*], [42, 35, 3], [43, 4, 25]
[43, 5, 24], [43, 6, 22], [43, 7, 22], [43, 8, 20], [43, 9, 19], [43, 10, 17], [43, 11, 16]
[43, 12, 16], [43, 13, 15], [43, 14, 15], [43, 15, 13], [43, 16, 13], [43, 17, 12], [43, 18, 11]
[43, 19, 11], [43, 20, 11], [43, 21, 10], [43, 22, 9], [43, 23, 9], [43, 24, 8], [43, 25, 7]
[43, 26, 8], [43, 27, 7], [43, 28, 7], [43, 29, 7], [43, 30, 6], [43, 31, 6], [43, 32, 6*]
[43, 33, 5], [43, 34, 5*], [43, 35, 4*], [43, 36, 3], [44, 4, 26], [44, 5, 24], [44, 6, 23]
[44, 7, 22], [44, 8, 22], [44, 9, 19], [44, 10, 18], [44, 11, 17], [44, 12, 17], [44, 13, 16]
[44, 14, 14], [44, 15, 14], [44, 16, 13], [44, 17, 12], [44, 18, 12], [44, 19, 11], [44, 20, 11]
[44, 21, 10], [44, 22, 10], [44, 23, 10], [44, 24, 10], [44, 25, 8], [44, 26, 8], [44, 27, 8]
[44, 28, 7], [44, 29, 6], [44, 30, 7], [44, 31, 6], [44, 32, 6], [44, 33, 6*], [44, 34, 5]
[44, 35, 5*], [44, 36, 4*], [44, 37, 3], [45, 4, 27], [45, 5, 25], [45, 6, 23], [45, 7, 22]
[45, 8, 22], [45, 9, 20], [45, 10, 19], [45, 11, 17], [45, 12, 16], [45, 13, 15], [45, 14, 15]
[45, 15, 14], [45, 16, 13], [45, 17, 13], [45, 18, 12], [45, 19, 12], [45, 20, 11], [45, 21, 11]
[45, 22, 10], [45, 23, 9], [45, 24, 9], [45, 25, 10], [45, 26, 8], [45, 27, 8], [45, 28, 7]
[45, 29, 7], [45, 30, 7], [45, 31, 7], [45, 32, 6], [45, 33, 6], [45, 34, 6*], [45, 35, 5]
[45, 36, 5], [45, 37, 4*]

[45, 38, 4*], [46, 4, 28], [46, 5, 25], [46, 6, 24], [46, 7, 22], [46, 8, 22], [46, 9, 20]
[46, 10, 19], [46, 11, 18], [46, 12, 17], [46, 13, 16], [46, 14, 15], [46, 15, 15], [46, 16, 14]
[46, 17, 13], [46, 18, 13], [46, 19, 12], [46, 20, 11], [46, 21, 11], [46, 22, 10], [46, 23, 10]
[46, 24, 9], [46, 25, 8], [46, 26, 10], [46, 27, 8], [46, 28, 8], [46, 29, 7], [46, 30, 8]
[46, 31, 7], [46, 32, 7], [46, 33, 6], [46, 34, 5], [46, 35, 6*], [46, 36, 5], [46, 37, 5]
[46, 38, 3], [46, 39, 4*], [47, 4, 28], [47, 5, 28], [47, 6, 25], [47, 7, 23], [47, 8, 22]
[47, 9, 21], [47, 10, 19], [47, 11, 19], [47, 12, 17], [47, 13, 17], [47, 14, 16], [47, 15, 15]
[47, 16, 14], [47, 17, 14], [47, 18, 12], [47, 19, 13], [47, 20, 12], [47, 21, 11], [47, 22, 11]
[47, 23, 10], [47, 24, 10], [47, 25, 9], [47, 26, 9], [47, 27, 10], [47, 28, 8], [47, 29, 7]
[47, 30, 7], [47, 31, 8], [47, 32, 7], [47, 33, 7], [47, 34, 6], [47, 35, 6], [47, 36, 5]
[47, 37, 5], [47, 38, 5], [47, 39, 4*], [47, 40, 4*], [48, 4, 31*], [48, 5, 28], [48, 6, 28]
[48, 7, 23], [48, 8, 23], [48, 9, 22], [48, 10, 20], [48, 11, 19], [48, 12, 18], [48, 13, 17]
[48, 14, 16], [48, 15, 16], [48, 16, 15], [48, 17, 14], [48, 18, 14], [48, 19, 13], [48, 20, 12]
[48, 21, 11], [48, 22, 11], [48, 23, 10], [48, 24, 10], [48, 25, 10], [48, 26, 9], [48, 27, 9]
[48, 28, 10], [48, 29, 8], [48, 30, 7], [48, 31, 7], [48, 32, 8], [48, 33, 7], [48, 34, 7]
[48, 35, 6], [48, 36, 6], [48, 37, 5], [48, 38, 5], [48, 39, 5*], [48, 40, 4*], [48, 41, 4*]
[49, 4, 31], [49, 5, 30], [49, 6, 26], [49, 7, 28], [49, 8, 23], [49, 9, 22], [49, 10, 21]
[49, 11, 20], [49, 12, 18], [49, 13, 18], [49, 14, 17], [49, 15, 19], [49, 16, 16], [49, 17, 15]
[49, 18, 14], [49, 19, 13], [49, 20, 13], [49, 21, 12], [49, 22, 12], [49, 23, 11], [49, 24, 10]
[49, 25, 10], [49, 26, 10], [49, 27, 10], [49, 28, 10], [49, 29, 10], [49, 30, 9], [49, 31, 7]
[49, 32, 7], [49, 33, 8], [49, 34, 7], [49, 35, 7], [49, 36, 6], [49, 37, 6], [49, 38, 5]
[49, 39, 5], [49, 40, 5*], [49, 41, 4*], [49, 42, 4*], [50, 4, 30], [50, 5, 30], [50, 6, 27]
[50, 7, 28], [50, 8, 23], [50, 9, 23], [50, 10, 21], [50, 11, 20], [50, 12, 19], [50, 13, 18]
[50, 14, 18], [50, 15, 17], [50, 16, 17], [50, 17, 15], [50, 18, 14], [50, 19, 14], [50, 20, 13]
[50, 21, 13], [50, 22, 12], [50, 23, 11], [50, 24, 11], [50, 25, 10], [50, 26, 10], [50, 27, 10]
[50, 28, 10], [50, 29, 9], [50, 30, 9], [50, 31, 8], [50, 32, 8], [50, 33, 8], [50, 34, 7]
[50, 35, 7], [50, 36, 7], [50, 37, 6], [50, 38, 5], [50, 39, 5], [50, 40, 5], [50, 41, 4]
[50, 42, 4*], [50, 43, 4*]

CHAPTER 5

ALGEBRAIC CONSTRUCTION OF CYCLIC AND QUASI-CYCLIC LCD CODES

A linear code C of length n over $\mathbb{F}_q$ is called a cyclic code if its codewords remain in C after a cyclic shift by 1 unit. More precisely, $(c_0, c_1, \cdots, c_{n-1}) \in C$ implies $(c_{n-1}, c_0, c_1, \cdots, c_{n-2}) \in C$. This property allows us to represent the codewords of the cyclic code C as polynomials in $\mathbb{F}_q[x]$, which simplifies their construction, representation, analysis and decoding. By mapping $(c_0, c_1, \cdots c_{n-1}) \in C$ to the polynomial $c(x) = c_0 + c_1 x + \cdots + c_{n-1}x^{n-1} \in \mathbb{F}_q[x]/\langle x^n - 1\rangle$, any cyclic code C of length n over $\mathbb{F}_q$ can be represented as an ideal in $\mathbb{F}_q[x]/\langle x^n - 1\rangle$, since the cyclic shift invariance corresponds to $x \cdot c(x) \in C$. Moreover, every ideal of $\mathbb{F}_q[x]/\langle x^n - 1\rangle$ is principal, which implies $C = \langle g(x)\rangle$ such that $g(x)$ is a monic divisor of $x^n - 1$. The polynomial $g(x)$ is called the generator polynomial of the cyclic code C.

In [24], Yang and Massey characterize LCD cyclic codes. Following their notation, let C be a q-ary cyclic code of length $n = \tilde{n} \cdot p^e$, where $\tilde{n}$ and p are coprime where $q = p^m$. In $\mathbb{F}_q[x]$, the polynomial $x^n - 1$ can be written as

$$x^n - 1 = x^{\tilde{n}p^e} - 1 = (x^{\tilde{n}} - 1)^{p^e}$$

Since $x^{\tilde{n}} - 1$ has no repeated irreducible factors when $\tilde{n}$ and p are coprime, each irreducible factor of $x^{\tilde{n}} - 1$ appears exactly once. Then, raising $x^{\tilde{n}} - 1$ to p^e'th power means that every irreducible factor in $x^n - 1$ appears exactly p^e times. Consequently, all monic irreducible factors of $x^n - 1 \in \mathbb{F}_q[x]$ have multiplicity p^e. Given the monic polynomial $f(x) = c_0 + c_1 x + \cdots + c_{r-1}x^{r-1} + x^r \in \mathbb{F}_q[x]$ with $c_0 \neq 0$, the normalized

monic reciprocal of $f(x)$ is

$$\tilde{f}(x) = c_0^{-1}(c_r + c_{r-1}x + c_{r-2}x^2 + \cdots + c_0x^r) = c_0^{-1}x^r f(x^{-1})$$

Lemma 5.1. *[24] If $g(x)$ is the generator polynomial of an $[n, k]$ cyclic code C of length $n = \tilde{n} \cdot p^e$, where $p = char(\mathbb{F}_q)$ and $\tilde{n}$ and p are coprime, then C is LCD if and only if $\gcd(g(x), \tilde{h}(x)) = 1$ where $\tilde{h}(x)$ is the monic reciprocal polynomial of $h(x) = \frac{x^n-1}{g(x)}$.*

Theorem 5.2. *[24] If $g(x)$ is the generator polynomial of a q-ary $[n, k]$ cyclic code C of length n, then C is LCD if and only if $g(x)$ is monic self-reciprocal, which means $\tilde{g}(x) = g(x)$, and all the monic irreducible factors of $g(x)$ have the same multiplicity in $g(x)$ and in $x^n - 1$.*

5.1 LCD Cyclic Code Construction Algorithm

In our search, we assume that $\gcd(n, p) = 1$. From Theorem 5.2 and Lemma 5.1, we know that in order to create LCD cyclic codes, we need to generate all self-reciprocal factors of $x^n - 1$. Suppose that the factorization of $x^n - 1$ is

$$x^n - 1 = g_1(x) \cdots g_s(x) \cdot h_1(x)h_1^*(x) \cdots h_t(x)h_t^*(x)$$

where $g_i(x)$ is self-reciprocal, for $i \in \{1, \ldots, s\}$, and $h_j(x), h_j^*(x)$ are reciprocal pairs, for $j \in \{1, \ldots, t\}$. Let F be the set of all self-reciprocal factors and let E be the set of products of the reciprocal pairs such that

$$F = \{g_1(x), \cdots, g_s(x)\},\ E = \{h_1(x) \cdot h_1^*(x), \ldots, h_t(x) \cdot h^*t(x)\}.$$

Note that any nonempty subset $S \subseteq F \cup E$ generates a self-reciprocal factor of $x^n - 1$ if we take the product of elements in S. This way, we can generate all LCD cyclic codes of length n over $\mathbb{F}_q$.

5.2 LCD Quasi-Cyclic Code Decomposition

Now let m be a positive integer coprime with q. A linear code of length $n = m\ell$ over $\mathbb{F}_q$ is called quasi-cyclic (QC) of index ℓ if it is closed under shifting codewords by

ℓ units, and ℓ is the smallest positive integer satisfying this property. In particular, cyclic codes amount to the special case $\ell = 1$. If we let $R := \mathbb{F}_q[x]/\langle x^m - 1\rangle$, then the code C can be viewed as an R-module in R^ℓ ([18, Lemma 3.1]). To observe this, we first view codewords of C as $m \times \ell$ arrays as follows

$$\mathbf{c} = \begin{pmatrix} c_{00} & \cdots & c_{0,\ell-1} \\ \vdots & & \vdots \\ c_{m-1,0} & \cdots & c_{m-1,\ell-1} \end{pmatrix}. \tag{5.1}$$

Here, being invariant under shift by ℓ units amounts to being closed under the row shift where each row is moved downward one row and the bottom row is moved to the top.

To an element $\mathbf{c} \in \mathbb{F}_q^{m\times\ell} \simeq \mathbb{F}_q^{m\ell}$ as in (5.1), we associate an element of R^ℓ as

$$\mathbf{c}(x) := (c_0(x), c_1(x), \ldots, c_{\ell-1}(x)) \in R^\ell,$$

where, for each $0 \le j \le \ell - 1$,

$$c_j(x) := c_{0,j} + c_{1,j}x + c_{2,j}x^2 + \cdots + c_{m-1,j}x^{m-1} \in R.$$

Thus, the following map is an R-module isomorphism:

$$\begin{aligned} \phi : \quad & \mathbb{F}_q^{m\ell} & \longrightarrow \quad & R^\ell \\ \mathbf{c} = & \begin{pmatrix} c_{00} & \cdots & c_{0,\ell-1} \\ \vdots & & \vdots \\ c_{m-1,0} & \cdots & c_{m-1,\ell-1} \end{pmatrix} & \longmapsto \quad & \mathbf{c}(x). \end{aligned} \tag{5.2}$$

Note that, when $\ell = 1$, we obtain the classical polynomial representation of cyclic codes. Also, the row shift on $\mathbb{F}_q^{m\times\ell}$ corresponds to the componentwise multiplication by x in R^ℓ. Therefore, a q-ary QC code C of length $m\ell$ and index ℓ can be viewed as an R-submodule in R^ℓ.

Observe that the reciprocal of the monic polynomial $x^m - 1$ is $-(x^m - 1)$. We assume the following factorization into irreducible polynomials in $\mathbb{F}_q[x]$

$$x^m - 1 = g_1(x) \cdots g_s(x) h_1(x) h_1^*(x) \cdots h_t(x) h_t^*(x), \tag{5.3}$$

where $g_i(x)$'s are self-reciprocal and $h_j^*(x)$ denotes the reciprocal of $h_j(x)$. There are no repeating factors in eqn. (5.3) since $\gcd(m, q) = 1$. Let ξ be a primitive m^{th} root

of unity in some extension field of $\mathbb{F}_q$. For each i, j, assume that $g_i(\xi^{u_i}) = 0$ and $h_j(\xi^{v_j}) = 0$, for some nonnegative integers u_i and v_j. Then, we also have $h_j^*(\xi^{-v_j}) = 0$. By the Chinese Remainder Theorem (CRT), R decomposes as

$$R \simeq \left(\bigoplus_{i=1}^{s} \mathbb{F}_q[x]/\langle g_i(x) \rangle \right) \oplus \left(\bigoplus_{j=1}^{t} \Big(\mathbb{F}_q[x]/\langle h_j(x) \rangle \oplus \mathbb{F}_q[x]/\langle h_j^*(x) \rangle \Big) \right) \tag{5.4}$$

Since each $g_i(x)$ and $h_j(x), h_j^*(x)$ are irreducible, the summands in eqn. (5.4) above are (isomorphic to) field extensions of $\mathbb{F}_q$. We let $\mathbb{G}_i = \mathbb{F}_q[x]/\langle g_i(x) \rangle$, $\mathbb{H}'_j = \mathbb{F}_q[x]/\langle h_j(x) \rangle$ and $\mathbb{H}''_j = \mathbb{F}_q[x]/\langle h_j^*(x) \rangle$ for simplicity. The map that sends $a(x) \in R$ to the decomposition as follows:

$$a(x) \mapsto \left(\bigoplus_{i=1}^{s} a(\xi^{u_i}) \right) \oplus \left(\bigoplus_{j=1}^{t} \Big(a(\xi^{v_j}) \oplus a(\xi^{-v_j}) \Big) \right).$$

This decomposition naturally extends to R^ℓ as

$$R^\ell \simeq \left(\bigoplus_{i=1}^{s} \mathbb{G}_i^\ell \right) \oplus \left(\bigoplus_{j=1}^{t} (\mathbb{H}'_j)^\ell \oplus (\mathbb{H}''_j)^\ell \right)$$

and then $C \subset R^\ell$ decomposes as

$$C = \left(\bigoplus_{i=1}^{s} C_i \right) \oplus \left(\bigoplus_{j=1}^{t} \Big(C'_j \oplus C''_j \Big) \right), \tag{5.5}$$

where each component code is a linear code of length ℓ over the base field ($\mathbb{G}_i, \mathbb{H}'_j$ or $\mathbb{H}''_j$) (see [18, Section IV]). The component codes C_i, C'_j, C''_j are called the constituents of C.

The constituents can be described in terms of the generators of C ([11, Lemma 2.1]). Namely, if C is an r-generator QC code with generators

$$\{ \big(a_{1,1}(x), \ldots, a_{1,\ell}(x)\big), \ldots, \big(a_{r,1}(x), \ldots, a_{r,\ell}(x)\big) \} \subset R^\ell,$$

then

$$\begin{aligned}
C_i &= \mathrm{Span}_{G_i} \big\{ \big(a_{b,1}(\xi^{u_i}), \ldots, a_{b,\ell}(\xi^{u_i})\big) : 1 \le b \le r \big\}, \text{ for } 1 \le i \le s, \\
C'_j &= \mathrm{Span}_{H'_j} \big\{ \big(a_{b,1}(\xi^{v_j}), \ldots, a_{b,\ell}(\xi^{v_j})\big) : 1 \le b \le r \big\}, \text{ for } 1 \le j \le t, \\
C''_j &= \mathrm{Span}_{H''_j} \big\{ \big(a_{b,1}(\xi^{-v_j}), \ldots, a_{b,\ell}(\xi^{-v_j})\big) : 1 \le b \le r \big\}, \text{ for } 1 \le j \le t.
\end{aligned} \tag{5.6}$$

For $i \in \{1, \ldots, s\}$, let θ_i be the generating primitive idempotent for the q-ary minimal cyclic code of length m, whose check polynomial is $g_i(x)$. Then, $\langle \theta_i \rangle$ is isomorphic

to the field $\mathbb{G}_i$ (see [11, Section III]). Similarly, let θ'_j and θ''_j denote the primitive idempotent generators for the minimal cyclic codes which are isomorphic to the fields $\mathbb{H}'_j$ and $\mathbb{H}''_j$ (for $1 \leq j \leq t$), respectively. By Jensen's work ([16]), it was shown in [11] that the QC code C above also has a concatenated decomposition

$$C = \left(\bigoplus_{i=1}^{s} \langle \theta_i \rangle \Box \mathfrak{C}_i \right) \oplus \left(\bigoplus_{j=1}^{t} \left(\langle \theta'_j \rangle \Box \mathfrak{C}'_j \right) \oplus \left(\langle \theta''_j \rangle \Box \mathfrak{C}''_j \right) \right), \tag{5.7}$$

where the outer codes $\mathfrak{C}_i, \mathfrak{C}'_j, \mathfrak{C}''_j$ are length ℓ linear codes over $\mathbb{G}_i, \mathbb{H}'_j, \mathbb{H}''_j$, respectively, and where $\Box$ denotes standard concatenation. More importantly, the outer codes and the constituents (in the CRT decomposition) are the same ([11, Theorem 4.1]). The converse statement holds as well. Namely, if you start with arbitrary length ℓ outer codes (constituents) over the fields $\mathbb{G}_i, \mathbb{H}'_j, \mathbb{H}''_j$ and form the concatenation above, the resulting code is a QC code over $\mathbb{F}_q$ of length $m\ell$ and index ℓ.

Since each $g_i(x)$ is self-reciprocal, the cardinality of $\mathbb{G}_i$, say q_i, is an even power of q for all $1 \leq i \leq s$ with two exceptions. One of these exceptions, for all m and q, is the field coming from the irreducible factor $x-1$ of x^m-1. When q is odd and m is even, $x+1$ is another self-reciprocal irreducible factor of x^m-1. In these cases, $q_i = q$. Except for these two cases, we equip each $\mathbb{G}_i^\ell$ with the inner Hermitian product

$$\langle \mathbf{c}, \mathbf{d} \rangle := \sum_{j=0}^{\ell-1} c_j d_j^{\sqrt{q_i}}, \tag{5.8}$$

where $\mathbf{c} = (c_0, \ldots, c_{\ell-1}), \mathbf{d} = (d_0, \ldots, d_{\ell-1}) \in \mathbb{G}_i^\ell$. For the two exceptions, in which case the corresponding field $\mathbb{G}_i$ is $\mathbb{F}_q$, we equip $\mathbb{G}_i^\ell$ with the usual Euclidean inner product. With the appropriate inner product on $\mathbb{G}_i$, $\perp_{\mathbb{G}}$ denotes the dual on $\mathbb{G}_i^\ell$. For each $1 \leq t \leq p$, $\mathbb{H}_t'^\ell = \mathbb{H}_t''^\ell$ is also equipped with the Euclidean inner product; $\perp_e$ denotes the dual on $\mathbb{H}_t'^\ell = \mathbb{H}_t''^\ell$.

For a QC code C with the CRT decomposition as in (5.5), the (Euclidean) dual in $\mathbb{F}_q^{m\ell}$ is of the form

$$C^\perp = \left(\bigoplus_{i=1}^{s} C_i^{\perp_{\mathbb{G}}} \right) \oplus \left(\bigoplus_{j=1}^{\top} \left(C_j''^{\perp_e} \oplus C_j'^{\perp_e} \right) \right). \tag{5.9}$$

The characterization of QC-LCD codes can be obtained via their constituents immediately.

Theorem 5.3. *[12, Theorem 3.1] Let C be a q-ary QC code of length $m\ell$ and index ℓ whose CRT decomposition is as in (5.5) and (5.7). Then C is LCD if and only if C_i is Hermitian or Euclidean LCD for all $1 \leq i \leq s$ and $C'_j \cap C''^{\perp_e}_j = \{0\}$, $C''_j \cap C'^{\perp_e}_j = \{0\}$, for all $1 \leq j \leq t$.*

Proof. Immediate from the CRT decomposition of the dual code $C^{\perp}$ in (5.9). □

5.3 LCD Quasi-Cyclic Code Construction Algorithm

In our search, we assume that $\gcd(m, q) = 1$. Suppose that the factorization of $x^n - 1$ is

$$x^m - 1 = g_1(x) \cdots g_s(x) \cdot h_1(x)h_1^*(x) \cdots h_t(x)h_t^*(x)$$

where $g_i(x)$ is self-reciprocal, for $i \in \{1, \ldots, s\}$, and $h_j(x), h_j^*(x)$ are reciprocal pairs, for $j \in \{1, \ldots, t\}$. We let $\mathbb{G}_i = \mathbb{F}_q[x]/\langle g_i(x)\rangle$, $\mathbb{H}'_j = \mathbb{F}_q[x]/\langle h_j(x)\rangle$ and $\mathbb{H}''_j = \mathbb{F}_q[x]/\langle h_j^*(x)\rangle$ as described above. For the two exceptions $x \pm 1$, in which case the corresponding field $\mathbb{G}_i$ is $\mathbb{F}_q$, we equip $\mathbb{G}_i^\ell$ with the usual Euclidean inner product. Except for these two cases, we equip each $\mathbb{G}_i^\ell$ with the inner Hermitian product. With the appropriate inner product on $\mathbb{G}_i$, $\perp_{\mathbb{G}}$ denotes the dual on $\mathbb{G}_i^\ell$. For each $1 \leq t \leq p$, $\mathbb{H}_t'^\ell = \mathbb{H}_t''^\ell$ is also equipped with the Euclidean inner product; $\perp_e$ denotes the dual on $\mathbb{H}_t'^\ell = \mathbb{H}_t''^\ell$.

Using this setup, we generate random constituents C_i, C'_j, C''_j of length ℓ over $\mathbb{G}_i, \mathbb{H}'_j, \mathbb{H}''_j$, respectively, which satisfy the conditions of Theorem 5.3, for each $i \in \{1, \ldots, s\}$ and $j \in \{1, \ldots, t\}$. That is, $C_i \cap C_i^{\perp_{\mathbb{G}}} = \{0\}$, for all $1 \leq i \leq s$ and $C'_j \cap C''^{\perp_e}_j = C''_j \cap C'^{\perp_e}_j = \{0\}$, for all $1 \leq j \leq t$.

5.4 Cyclic and Quasi-Cyclic Code Search Results

Since including all the polynomial results would take up too much space, we present only a few representative examples in this section, along with a complete list of the cyclic and quasi-cyclic results. The full set of polynomial results can be found on [17]

When running the search program, all previously known results are filtered out. The remaining candidate codes are evaluated using the *BDLC* method in the *Magma V2.28-19* [6] database. A candidate code is accepted if its corresponding *Magma BDLC* code is not LCD, or if it differs in dimension or length, or if it achieves a better minimum distance.

Example 5.4. There is a binary $[63,\ 38,\ 10]$ LCD cyclic code with generator polynomial

$$x^{25} + x^{24} + x^{22} + x^{20} + x^{18} + x^{17} + x^{16} + x^9 + x^8 + x^7 + x^5 + x^3 + x + 1.$$

Example 5.5. There is a binary $[42,\ 21,\ 9]$ LCD quasi-cyclic code of index 2 has polynomial generators by the rows of the following generator polynomial matrix

$$\begin{pmatrix} x^4 + x^2 + x + 1 & x^{16} + x^{15} + x^{14} + x^{12} + x^7 + x^6 + x^5 + x^3 + x^2 + x + 1 \\ 0 & x^{17} + x^{15} + x^{14} + x^{10} + x^8 + x^7 + x^3 + x + 1 \end{pmatrix}.$$

Example 5.6. There is a ternary $[73,\ 37,\ 16]$ LCD cyclic code with generator polynomial

$$x^{36} + x^{35} + 2x^{30} + 2x^{29} + x^{28} + 2x^{27} + 2x^{25} + 2x^{24} + 2x^{21} + x^{19} + x^{17} + 2x^{15} + 2x^{12} + 2x^{11} + 2x^9 + x^8 + 2x^7 + 2x^6 + x + 1.$$

Example 5.7. There is a ternary $[28,\ 8,\ 13]$ LCD quasi-cyclic code of index 4 has polynomial generators by the rows of the following generator polynomial vectors

$$\left(1,\ 2x^5 + x^2 + 2,\ x^5 + 2x^3 + 2x^2,\ 2x^6 + x^5 + 2x^4 + 2x^3 + 2x^2 + x\right),$$
$$\left(0,\ x^6 + x^5 + x^4 + x^3 + x^2 + x + 1,\ x^6 + x^5 + x^4 + x^3 + x^2 + x + 1.\ 0\right).$$

5.4.1 Binary LCD Cyclic Codes

We have found the following codes.

$[31, 21, 5],\ [33, 13, 10],\ [63, 15, 19],\ [63, 38, 10],\ [73, 19, 19],\ [75, 7, 15],\ [85, 8, 30],$
$[85, 9, 25],\ [85, 12, 30],\ [89, 44, 16],\ [89, 45, 15],\ [89, 67, 7],\ [91, 7, 13],\ [91, 31, 13],$
$[99, 11, 33].$

5.4.2 Binary LCD Quasi-Cyclic Codes

We have found the follwing code.

$$[42,\ 21,\ 9].$$

5.4.3 Ternary LCD Cyclic Codes

We have found the following codes.

$[26, 13, 8],\ [28, 15, 8],\ [35, 6, 10],\ [35, 10, 10],\ [37, 18, 11],\ [37, 19, 10],\ [40, 25, 8]$
$[49, 6, 14],\ [52, 14, 14],\ [55, 4, 22],\ [56, 8, 20],\ [56, 9, 20],\ [56, 12, 12],\ [56, 13, 12],$
$[56, 14, 12],\ [56, 15, 12],\ [56, 16, 12],\ [56, 18, 12],\ [56, 19, 12],\ [56, 20, 12],\ [56, 21, 12],$
$[56, 22, 12],\ [56, 24, 12],\ [56, 25, 12],\ [56, 26, 12],\ [56, 27, 12],\ [61, 31, 14],\ [61, 41, 9],$
$[64, 4, 16],\ [64, 5, 16],\ [64, 6, 16],\ [64, 7, 16],\ [65, 7, 25],\ [73, 36, 16],\ [73, 37, 16],$
$[73, 48, 11],\ [80, 4, 32],\ [80, 5, 32],\ [80, 6, 32],\ [80, 7, 32],\ [80, 8, 32],\ [80, 9, 32],$
$[80, 10, 32],\ [80, 11, 32],\ [80, 12, 28],\ [80, 12, 32],\ [80, 13, 28],\ [80, 13, 32],\ [80, 14, 28],$
$[80, 14, 32],\ [80, 15, 28],\ [80, 15, 32],\ [80, 16, 32],\ [80, 17, 32],\ [80, 18, 28],\ [80, 18, 32],$
$[80, 19, 28],\ [80, 20, 28],\ [80, 21, 28],\ [80, 22, 28],\ [82, 8, 44],\ [82, 9, 40],\ [82, 9, 44],$
$[82, 10, 40],\ [82, 32, 18],\ [82, 33, 18],\ [82, 34, 18],\ [85, 4, 34],\ [85, 32, 20],\ [85, 33, 20],$
$[91, 12, 44],\ [91, 13, 44],\ [91, 18, 33],\ [91, 19, 33],\ [91, 31, 28],\ [91, 36, 17],\ [95, 4, 38],$
$[95, 5, 19],\ [97, 48, 24],\ [98, 6, 28],\ [98, 7, 28].$

5.4.4 Ternary LCD Quasi-Cyclic Codes

We have found the following codes.

$[20, 9, 8],\ [21, 6, 11],\ [21, 7, 10],\ [26, 12, 9],\ [26, 13, 8],\ [28, 8, 13],\ [28, 12, 10],$
$[30, 4, 19],\ [33, 10, 14],\ [35, 4, 22],\ [39, 6, 22].$

REFERENCES

[1] M. Araya and M. Harada, “On the classification of linear complementary dual codes”, Discrete Mathematics, vol. 342, no. 1, pp. 270–278, doi: 10.1016/j.disc.2018.09.034, 2019.

[2] M. Araya and M. Harada, “On the minimum weights of binary linear complementary dual codes”, Cryptography and Communications, vol. 12, no. 2, pp. 285–300, doi: 10.1007/s12095-019-00402-5, 2019.

[3] M. Araya, M. Harada, K. Ishizuka, and Y. Tanaka, “Characterizations of the minimum weights of LCD codes of large dimensions”, IEEE Transactions on Information Theory, vol. 70, no. 12, pp. 8758–8769, doi: 10.1109/tit.2024.3483218, 2024.

[4] M. Araya, M. Harada, and K. Saito, “Characterization and classification of optimal LCD codes”, Designs, Codes and Cryptography, vol. 89, no. 4, pp. 617–640, doi: 10.1007/s10623-020-00834-8, 2021.

[5] M. Araya, M. Harada, and K. Saito, “On the minimum weights of binary LCD codes and ternary LCD codes”, Finite Fields and Their Applications, vol. 76, 101925, doi: 10.1016/j.ffa.2021.101925, 2021.

[6] W. Bosma, J. Cannon and C. Playoust, “The MAGMA algebra system. I. The user language", Journal of Symbolic Computation, vol. 24, pp. 1179–1260,doi: 10.1006/jsco.1996.0125, 1997.

[7] S. Bouyuklieva, “Optimal binary LCD codes”, Designs, Codes and Cryptography, vol. 89, no. 11, pp. 2445–2461, doi: 10.1007/s10623-021-00929-w, 2021.

[8] C. Carlet, S. Mesnager, C. Tang, Y. Qi, and R. Pellikaan, “Linear codes over $\mathbb{F}_q$ are equivalent to LCD codes for $q > 3$”, IEEE Transactions on Information Theory, vol. 64, no. 4, pp. 3010–3017, doi: 10.1109/TIT.2018.2789347, 2018.

[9] S. T. Dougherty, J.-L. Kim, B. Özkaya, L. Sok, and P. Solé, “The combinatorics of LCD codes: Linear Programming bound and orthogonal matrices”, International Journal of Information and Coding Theory, vol. 4, no. 2/3, pp. 116–128, doi: 10.1504/IJICOT.2017.083827, 2017.

[10] Gurobi Optimization, LLC, Gurobi Optimizer Reference Manual. available online at: https://www.gurobi.com, 2025.

[11] C. Güneri and F. Özbudak, “The concatenated structure of quasi-cyclic codes and an improvement of Jensen’s bound”, IEEE Transactions on Information Theory, vol. 59, no. 2, pp. 979–985, doi: 10.1109/tit.2012.2225823, 2013.

[12] C. Güneri, B. Özkaya, and P. Solé, “Quasi-cyclic complementary dual codes”, Finite Fields and Their Applications, vol. 42, pp. 67–80, doi: 10.1016/j.ffa.2016.07.005, 2016.

[13] M. Harada, “Construction of binary LCD codes, ternary LCD codes and quaternary Hermitian LCD codes”, Designs, Codes and Cryptography, vol. 89, no. 10, pp. 2295–2312, doi: 10.1007/s10623-021-00916-1, 2021.

[14] M. Harada and K. Saito, “Binary linear complementary dual codes”, Cryptography and Communications, vol. 11, no. 4, pp. 677–696, doi: 10.1007/s12095-018-0319-0, 2018.

[15] W. C. Huffman and V. Pless, *Fundamentals of Error-Correcting Codes*, Cambridge University Press; pp. 75–79, doi: 10.1017/CBO9780511807077, 2003.

[16] J. Jensen, “The concatenated structure of cyclic and Abelian codes”, IEEE Transactions on Information Theory, vol. 31, no. 6, pp. 788–793, doi: 10.1109/tit.1985.1057109, 1985.

[17] E. Karabakla, *code_search*, GitHub Repository, available online at: `https://github.com/karabakla/code_search`, 2025

[18] S. Ling and P. Sole, “On the algebraic structure of quasi-cyclic codes I: Finite fields”, IEEE Transactions on Information Theory, vol. 47, no. 7, pp. 2751–2760, doi: 10.1109/18.959257, 2001.

[19] S. Ling and P. Sole, “On the algebraic structure of quasi-cyclic codes III: Generator Theory”, IEEE Transactions on Information Theory, vol. 51, no. 7, pp. 2692–2700, doi: 10.1109/tit.2005.850142, 2005.

[20] J. Macwilliams, “A theorem on the distribution of weights in a systematic code", in The Bell System Technical Journal, vol. 42, no. 1, pp. 79-94, doi: 10.1002/j.1538-7305.1963.tb04003.x, 1963.

[21] J. L. Massey, “Linear codes with complementary duals", Discrete Mathematics, vol. 106–107, pp. 337–342, doi: 10.1016/0012-365x(92)90563-u, 1992.

[22] S. Li, M. Shi, and H. Liu, “Several constructions of optimal LCD codes over small finite fields”, Cryptography and Communications, vol. 16, no. 4, pp. 779–800, doi: 10.1007/s12095-024-00699-x, 2024.

[23] SageMath, the Sage Mathematics Software System (Version 10.4.0), The Sage Developers, available online at: https://www.sagemath.org, 2025.

[24] X. Yang and J. L. Massey, “The condition for a cyclic code to have a complementary dual”, Discrete Mathematics, vol. 126, no. 1–3, pp. 391–393, doi: 10.1016/0012-365x(94)90283-6, 1994.

[25] G. Wang, S. Liu, and H. Liu, “New constructions of optimal binary LCD codes”, Finite Fields and Their Applications, vol. 95, 102381, doi: 10.1016/j.ffa.2024.102381, 2024.