UPPER BOUNDS ON MINIMUM DISTANCE OF NONBINARY QUANTUM STABILIZER CODES

A Thesis

by

SANTOSH KUMAR

Submitted to the Office of Graduate Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

August 2004

Major Subject: Computer Science

UPPER BOUNDS ON MINIMUM DISTANCE OF NONBINARY QUANTUM STABILIZER CODES

A Thesis

by

SANTOSH KUMAR

Submitted to Texas A&M University
in partial fulfillment of the requirements
for the degree of

MASTER OF SCIENCE

Approved as to style and content by:

Andreas Klappenecker
(Chair of Committee)

Rabi Mahapatra
(Member)

Sue Geller
(Member)

Valerie Taylor
(Head of Department)

August 2004

Major Subject: Computer Science

ABSTRACT

Upper Bounds on Minimum Distance of Nonbinary Quantum Stabilizer Codes.

(August 2004)

Santosh Kumar, B.E., Mumbai University, India

Chair of Advisory Committee: Dr. Andreas Klappenecker

The most popular class of quantum error correcting codes is stabilizer codes. Binary quantum stabilizer codes have been well studied, and Calderbank, Rains, Shor and Sloane (July 1998) have constructed a table of upper bounds on the minimum distance of these codes using linear programming methods. However, not much is known in the case of nonbinary stabilizer codes. In this thesis, we establish a bridge between self-orthogonal classical codes over the finite field containing q^2 elements and quantum codes, extending and unifying previous work by Matsumoto and Uyematsu (2000), Ashikhmin and Knill (November 2001), Kim and Walker (2004). We construct a table of upper bounds on the minimum distance of the stabilizer codes using linear programming methods that are tighter than currently known bounds. Finally, we derive code construction techniques that will help us find new codes from existing ones. All these results help us to gain a better understanding of the theory of nonbinary stabilizer codes.

To my wonderful parents for always being there when I need them

ACKNOWLEDGMENTS

First and foremost, I would like to thank my advisor, Dr. Klappenecker, for always being a source of inspiration and motivation and for giving a definite direction to my work. I would also like to thank my committee members, Dr. Mahapatra and Dr. Geller, for invaluable suggestions for my proposal and my thesis. I would like to express my sincere gratitude to my colleagues, Pradeep Sarvepalli and Avanti Ketkar, for helping me out whenever I faced a problem. Last but not least, I would like to thank all those who patiently edited my thesis. This research has been supported by the NSF grant CCR 0218582 and a Texas A&M TITF initiative.

TABLE OF CONTENTS

CHAPTER Page

CHAPTER Page

LIST OF TABLES

CHAPTER I

INTRODUCTION

A. Quantum Computation and Information

Quantum computing has emerged as a new computing paradigm in the last decade. Using the principles of quantum mechanics, computations can be performed simultaneously, rather than separately, as on traditional classical systems. It is conjectured that this property of quantum parallelism helps to achieve an exponential speedup in certain quantum algorithms over their classical counterparts. Shor's factorization algorithm [1] supports this conjecture. This algorithm could have applications in breaking the RSA public key exchange algorithm, which is based on the difficulty in factoring large numbers.

While the unit of computation in classical computers is a *bit*, the unit of computation in quantum computers is a *qubit*, which is a nonzero vector in the 2-dimensional vector space of complex numbers, $\mathbf{C}^2$. We denote the standard basis of $\mathbf{C}^2$ by $\{|0\rangle, |1\rangle\}$. Any arbitrary state $|\psi\rangle$ in $\mathbf{C}^2$ can be expressed as a linear combination of the basis states $|0\rangle$, $|1\rangle$ as given below:

$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle, \tag{1.1}$$

where α and β are complex numbers that are usually assumed to be normalized to unit norm, i.e., $|\alpha|^2 + |\beta|^2 = 1$. In vector notation, $|\psi\rangle$ can be represented as

$$|\psi\rangle = \begin{pmatrix} \alpha \\ \beta \end{pmatrix}. \tag{1.2}$$

The journal model is *IEEE Transactions on Automatic Control.*

Analogous to the classical case, quantum computers have gates or operators acting on the qubits. Quantum circuits are comprised of quantum gates acting on the qubits in a logical fashion. An operator acting on a single qubit system is a unitary 2×2 matrix containing complex values. The above concepts can be generalized to q-ary systems in which the unit of computation is a *qudit*, which is a vector over $\mathbf{C}^q$. Also, operators acting on q-ary systems are $q \times q$ matrices containing complex values. Later, we will derive a basis for the set of $q \times q$ matrices. For a more complete exposition of the fundamentals of quantum computing, see [2].

In spite of quantum computation holding much promise, its practical realization has been hampered by the lack of good quantum error-correcting-codes. Errors in quantum computation occur due to decoherence, i.e., noise resulting from the interaction of the quantum computer with the environment. Also, quantum gates (in contrast to classical gates) are unitary transformations chosen from a continuum of possible values and, hence, are more susceptible to errors than their classical counterparts. Classical error correction works on the principle of duplicating information. However, because an arbitrary quantum state cannot be cloned, see [2, page 532], the same technique cannot be applied to quantum error correction. Also, while classical error correction needs to guard against bit flip errors, quantum error correction needs to protect against both bit flip and phase flip errors.

The first breakthrough in the field of quantum-error-correction came when Shor showed that quantum error correcting codes do exist [3]. He did this by constructing a quantum error correcting code that encodes 1 logical qubit into 9 qubits and allows for correction of one error. Thereafter, Gottesman [4] and Calderbank, Rains, Shor and Sloane [5] developed a special class of quantum-error-correcting codes, the so-called binary stabilizer codes. The class of stabilizer codes is important because it bears a strong resemblance to classical codes.

Calderbank, Rains, Shor and Sloane [5] have transformed the problem of finding binary quantum stabilizer codes into that of finding additive classical codes over $\mathbf{F}_4$ that are self-orthogonal with respect to a certain trace inner product. $\mathbf{F}_4$ denotes the finite field containing 4 elements. They have also constructed a table of upper bounds of the minimum distance of such codes using linear programming methods. Another feature of this paper is that it also outlines numerous code construction methods that help discover new codes from existing ones.

The results established in [5] help in understanding the theory of binary stabilizer codes. But less is known in the case of nonbinary stabilizer codes. For some applications, such as proof-of-concept implementation in certain ion trap models (R. Laflamme, personal communication), nonbinary quantum error correction codes would be more useful. Also, codes over alphabets of size 2^m, where $m \geq 2$, could be useful for constructing easily decodable binary codes, via concatenation [6]. Hence, understanding the theory of nonbinary codes is of primary interest. Therefore, the main focus of this research is to gain a deeper understanding of nonbinary stabilizer codes by generalizing the results in [5] to the nonbinary case. In Chapters I-II, we review some key fundamentals that will be required for an understanding of the material covered in the later chapters. Also, in Chapter II, the problem of finding $q-$ary stabilizer codes is transformed to that of finding self-orthogonal classical codes over $\mathbf{F}_{q^2}$. In Chapter III, we determine numerous linear programming constraints that such classical codes over $\mathbf{F}_{q^2}$ need to satisfy. In Chapter IV, these constraints are then modelled using a linear optimization package to determine whether a solution exists. Non-existence of a solution implies the non-existence of the corresponding classical self-orthogonal code over the field $\mathbf{F}_{q^2}$. And non-existence of the classical code will automatically imply the non-existence of a corresponding $q-$ary quantum code. In Chapter IV, we also determine, using linear programming methods, whether we can have linear stabilizer codes meeting the bounds derived for additive stabilizer codes. In Chapter V, we outline a few code construction methods for nonbinary stabilizer codes that will help to derive new codes from existing ones. These techniques will help in finding codes that meet or nearly meet the upper bounds derived earlier. Appendix A contains the code in Mathematica for modelling the constraints derived for additive codes in Chapter III. Appendix B contains the code in Maple for modelling the constraints derived for linear codes in Chapter III using Integer Linear Programming.

B. Review of Mathematical Concepts

In this section, we review some basic mathematical concepts and introduce some notations that will be used frequently in the later chapters.

1. Galois Fields

A field of finite cardinality is often called a Galois field or a finite field. If p is the cardinality of the field, then the field is denoted by $\mathbf{F}_p$. If p is a prime, then the field $\mathbf{F}_p$ is given by the set $\mathbf{F}_p = \{0, 1, \ldots, p-1\}$ with addition and multiplication modulo p. From now on, p will be used to denote a prime, unless mentioned otherwise. If $q = p^m$ for some integer $m \geq 0$, then a field $\mathbf{F}_q$ with q elements is given by the set

$$\mathbf{F}_q = \{f(x) \in \mathbf{F}_p[x] | \deg f(x) < m\}. \tag{1.3}$$

The addition of elements in $\mathbf{F}_q$ is performed by adding in $\mathbf{F}_p[x]$ and multiplication in $\mathbf{F}_q$ is done by multiplying in $\mathbf{F}_p[x]$ and taking the remainder modulo a fixed irreducible polynomial $m(x)$ in $\mathbf{F}_p[x]$ of degree m.

Let β be a non-zero element in $\mathbf{F}_q$. The order of β is the smallest positive integer

k such that $\beta^k = 1$. An element with order $q-1$ in $\mathbf{F}_q$ is called a *primitive element* in $\mathbf{F}_q$. All nonzero elements in $\mathbf{F}_q$ can be represented by $q-1$ consecutive powers of a primitive element,

$$1, \alpha, \alpha^2, \alpha^3, \ldots, \alpha^{q-2}, \alpha^{q-1} = 1, \alpha^q = \alpha, \text{ and so on.}$$

2. Character Theory

Let $(G, +)$ be a finite abelian group and 0 the identity element of G. A character of G is a mapping χ, from G to the set of non-zero complex numbers, satisfying the following conditions:

1. $\chi(x+y) = \chi(x)\chi(y) \quad \text{for all}\, x, y \in G,$
2. $|\chi(x)| = 1 \quad \text{for all}\, x \in G.$

There exist $|G|$ distinct characters of the abelian group G. We can use the elements of G to index the set of characters ψ_x, $x \in G$. We assume that the trivial character $\psi_0(x) \equiv 1$ is indexed by the identity element 0.

A symmetric set of characters is one in which for a particular manner in which we index the set of characters, we have $\psi_x(y) = \psi_y(x)$ for all $x, y \in G$. An asymmetric set of characters is one in which for a particular manner in which we index the set of characters, we have $\psi_x(y) = \psi_y(-x)$ for all $x, y \in G$. where $x \neq y$ and $x, y \neq 0$. Group characters satisfy certain orthogonality relations as given below:

1. $\sum_{y \in G} \chi_x(y) = 0, \text{ where}\, \chi_x$ is a non-trivial character,
2. $\sum_{x \in G} \chi_x(y) = 0, \text{ where}\, y \neq 0 \in G.$

C. Classical Coding Theory

We now review some well known concepts from classical coding theory. Consider a code $C \subseteq \mathbf{F}_2^n$ that encodes k bits into n bits. Let T be the set of all vectors of length k over $\mathbf{F}_2$. The code C is then given by

$$C = \{Gt \,|\, \text{for all}\, t \in T\}, \tag{1.4}$$

where G is some $n \times k$ matrix (with entries from $\mathbf{F}_2$), called the generator matrix of the code. Its columns form a basis for the k-dimensional coding sub-space of the n-dimensional binary vector space. Given a generator matrix G, we can calculate the dual matrix P, which is an $(n-k) \times n$ matrix of 0's and 1's with $PG = 0$ and maximal rank $n-k$. The matrix P is called the parity check matrix for the code. It can be used to test if a given vector is a valid codeword, since $Pv = 0 \Leftrightarrow v \in C$.

The Hamming weight of a code vector is the number of nonzero coordinates in the code vector. The Hamming distance between two vectors $x = (x_1, x_2, \ldots, x_n)$ and $y = (y_1, y_2, \ldots, y_n)$ in C is the number of coordinates in which they differ, i.e.,

$$d(x, y) =\mid \{i \mid x_i \neq y_i\} \mid . \tag{1.5}$$

The minimum distance d of the code C is then defined as:

$$d = \min\{d(x, y)|x, y \in C\}. \tag{1.6}$$

A binary code to encode k bits in n bits with minimum distance d is said to be an $[n, k, d]_2$ code. For a code to correct t single bit errors, it must have distance at least $2t + 1$ between any two code words. The same concepts can be extended to the nonbinary alphabet of size q, where a code encoding k bits in n bits with minimum distance d, is said to be an $[n, k, d]_q$ code.

The dual code of C, denoted as $C^{\perp}$, is given by

$$C^{\perp} = \{u|u \in \mathbf{F}_2^n, u.v = 0 \, \text{for all} \, v \in C\}, \tag{1.7}$$

where $u.v$ is some inner product between the vectors u and v. The actual specifics of this inner product may vary from one code to another, as we shall see in later chapters. A code C is said to be self-orthogonal if $C \subseteq C^{\perp}$. In other words, C is contained in its dual space with repect to a certain notion of duality. For a more detailed exposition on classical coding theory, see [7].

CHAPTER II

CONNECTING NONBINARY QUANTUM CODES AND CLASSICAL CODES

The ultimate goal of any coding theorist is to construct as many codes as possible. An effective strategy employed in [5] is to translate the problem of finding binary quantum codes to that of finding quaternary classical codes that are self-orthogonal with respect to a certain trace inner product. The rationale behind this idea is that it is easier to construct equivalent classical codes and, hence, a bridge to classical codes would help us derive many quantum codes. Also, by means of such a bridge, we can translate the existence and non-existence of many quantum codes to that of classical codes and vice versa. In this chapter, we construct a similar bridge that connects q-ary quantum codes to q^2-ary self-orthogonal classical codes. The first part of the bridge connects q-ary quantum codes to q-ary classical codes and the second part of the bridge connects q-ary classical codes to q^2-ary self-orthogonal classical codes. Though the first part of the bridge may seem sufficient to address the problem at hand, it becomes cumbersome to operate in the domain of q-ary classical codes, as it involves use of symplectic weights as opposed to the Hamming weights. We will look at the symplectic weights later in this chapter. Hence, for convenience, we have the second part of the bridge. The bridge presented in this chapter attempts to extend and unify [8, 9, 10, 11]. First, we review some of the fundamental concepts of quantum-error-correction and also a special class of quantum error-correcting-codes, called the stabilizer codes.

A. Quantum Error Correction

1. Review of Fundamental Concepts

The central idea of quantum-error-correction is to introduce redundancy, not necessarily through duplication. A q-ary quantum-error-correction code, Q, that encodes k qubits into n qubits is a q^k-dimensional subspace of $\mathbf{C}^{q^n}$. Errors on a single qudit system are $q \times q$ operators. Because of the linearity of quantum mechanics, we need only consider a basis for the set of $q \times q$ operators. Later we will derive a set of basis operators acting on a single qudit system. If $\mathcal{E}$ is the set of basis operators acting on a single qudit system, then the set of basis error operators, $\mathcal{E}_n$, for n-qudit quantum systems is the set of all possible n-fold tensor products of operators in $\mathcal{E}$ and is given by

$$\mathcal{E}_n = \{\sigma_1 \otimes \sigma_2 \otimes \sigma_3 \ldots \sigma_n \,|\, \sigma_i \in \mathcal{E}\}. \tag{2.1}$$

We say that a quantum system that can correct a set of basis error operators can correct any arbitrary linear combination of these error operators. The weight of an error operator e is defined as the number of errors that differ from the identity matrix

$$w(e) = |\{\sigma_i : \sigma_i \neq I\}|. \tag{2.2}$$

The error correction and detection capabilities of a quantum error correcting code Q are the most crucial aspects of the code. A result by Knill and Laflamme [12] shows that a quantum code Q is able to detect an error E in $U(q^n)$, the set of all $q^n \times q^n$ unitary matrices, if and only if the condition $\langle c_1|E|c_2\rangle = \lambda_E \langle c_1|c_2\rangle$ holds for all states $|c_1\rangle$ and $|c_2\rangle$ in Q, where λ_E is a complex scalar. In addition to the detectable errors, there are undetectable errors that move vectors to another valid code vector in the same subspace.

2. Stabilizer Codes

Suppose G_n is the group generated by the error operators in $\mathcal{E}_n$. Then the stabilizer is an abelian subgroup S of G_n, such that every operator of S leaves every vector in the code space fixed. The quantum code is then defined as the joint eigenspace of all the operators in S. The centralizer of S, denoted $C(S)$, is a subgroup of G_n such that operators in $C(S)$ commute with all the operators in the stabilizer. Now, the set of detectable errors are those that lie outside $C(S)$. The set of undetectable errors are those that lie in $C(S) - S$. Finally, the operators that lie in S have trivial effect on any vector in the code space. If a code is to have distance d, then there should not be any non-zero operators of weight less than d in $C(S) - S$.

A main advantage of this stabilizer formalism is that the code can be defined in terms of the set of generating elements of the group S rather than describing it in terms of the code vectors. Two conditions that must be satisfied by S in order that it stabilizes a non-trivial vector space are:

1. The elements of S commute, and
2. $-I$ is not an element of S.

We illustrate the above concepts with the help of a simple example for the binary case.

Example 1. Consider a repetition code that encodes 1 qubit into 3 qubits as given below:

$$|0\rangle \mapsto |000\rangle, \quad |1\rangle \mapsto |111\rangle. \tag{2.3}$$

For a single qubit system, the set of basis error operators are the identity operator I, bit flip operator X, phase flip operator Z, and the Y operator, which is a combination of the bit flip and the phase flip operators. This set of I, X, Z and Y operators is also called the *Pauli basis*. The set of error operators $\mathcal{E}_n$, acting on the encoded subspace, will be the set of all 3-fold tensor products of the Pauli matrices. If G_n is the group generated by the operators in $\mathcal{E}_n$, then the stabilizer S is a subgroup of G_n, consisting of all operators that leave the code vectors fixed. For the repetition code above, S will consist of the operators $I \otimes I \otimes I$, $Z \otimes I \otimes Z$, $Z \otimes Z \otimes I$ and $I \otimes Z \otimes Z$, where $\otimes$ denotes the tensor product. It can be seen that the subgroup S is generated by the group elements $Z \otimes I \otimes Z$ and $Z \otimes Z \otimes I$.

Thus, we see that the entire code can be compactly described in terms of these generator elements rather than in terms of the code vectors themselves. In this regard, stabilizer codes are analogous to classical codes.

Definition 2. Pure Stabilizer codes: A stabilizer code is said to be pure if there are no operators of weight less than d in $C(S)$, else it is said to be impure.

For a more complete treatment of stabilizer codes, see [4].

B. Connecting q-ary Quantum Codes to Classical Codes over $\mathbf{F}_{q^2}$

1. Error Bases

As discussed earlier, a quantum error correcting code Q is a q^k-dimensional subspace of $\mathbf{C}^{q^n} = \mathbf{C}^q \otimes \cdots \otimes \mathbf{C}^q$. We denote by $|x\rangle$ the vectors of a distinguished orthonormal basis of $\mathbf{C}^q$, where the labels x range over the elements of a finite field $\mathbf{F}_q$ with q elements. First, we need to select an appropriate error model so that we can measure the performance of a code. To simplify matters, we choose a basis $\mathcal{E}_n$ of the vector space of complex $q^n \times q^n$ matrices to represent a discrete set of errors. To reiterate, the stabilizer code then is defined as the joint eigenspace of a subset of $\mathcal{E}_n$, so the error operators play a crucial role.

Let a and b be elements of the finite field $\mathbf{F}_q$. We define the unitary operators $X(a)$ and $Z(b)$ on $\mathbf{C}^q$ by

$$X(a)|x\rangle = |x+a\rangle, \qquad Z(b)|x\rangle = \omega^{\mathrm{tr}(bx)}|x\rangle,$$

where tr denotes the trace operation from the extension field $\mathbf{F}_q$ to the prime field $\mathbf{F}_p$, and $\omega = \exp(2\pi i/p)$ is a primitive pth root of unity.

We form the set $\mathcal{E} = \{X(a)Z(b)\,|\,a, b \in \mathbf{F}_q\}$ of error operators. The set $\mathcal{E}$ has some interesting properties, namely (a) it contains the identity matrix, (b) the product of two matrices in $\mathcal{E}$ is a scalar multiple of another element in $\mathcal{E}$, and (c) the trace $\mathrm{Tr}(A^\dagger B) = 0$ for distinct elements A, B of $\mathcal{E}$ (see Lemma 3). A finite set of q^2 unitary matrices that satisfies the properties (a), (b), and (c) is called a *nice error basis* [13].

The set $\mathcal{E}$ of error operators forms a basis of the set of complex $q \times q$ unitary matrices, thanks to property (c).

Lemma 3. *The set $\mathcal{E} = \{X(a)Z(b)\,|\,a, b \in \mathbf{F}_q\}$ is a nice error basis of $\mathbf{C}^q$.*

Proof. The matrix $X(0)Z(0)$ is the identity matrix, so property (a) holds. We have $\omega^{\mathrm{tr}(ba)}X(a)Z(b) = Z(b)X(a)$, which implies that the product of two error operators is given by

$$X(a)Z(b)\,X(a')Z(b') = \omega^{\mathrm{tr}(ba')}X(a+a')Z(b+b'). \tag{2.4}$$

This is a scalar multiple of an operator in $\mathcal{E}$, hence, property (b) holds.

Suppose that the error operators are of the form $A = X(a)Z(b)$ and $B = X(a)Z(b')$ for some $a, b, b' \in \mathbf{F}_q$. Then

$$\mathrm{Tr}(A^\dagger B) = \mathrm{Tr}(Z(b'-b)) = \sum_{x \in \mathbf{F}_q} \omega^{\mathrm{tr}((b'-b)x)}.$$

The map $x \mapsto \omega^{\mathrm{tr}((b'-b)x)}$ is a character of $\mathbf{F}_q$. The sum of all character values is 0 unless the character is trivial; thus, $\mathrm{Tr}(A^\dagger B) = 0$ when $b' \neq b$.

On the other hand, if $A = X(a)Z(b)$ and $B = X(a')Z(b')$ are two error operators satisfying $a \neq a'$, then the diagonal elements of the matrix $A^\dagger B = Z(-b)X(a' - a)Z(b')$ are 0, which implies $\mathrm{Tr}(A^\dagger B) = 0$. Thus, whenever A and B are distinct elements of $\mathcal{E}$, then $\mathrm{Tr}(A^\dagger B) = 0$, which proves (c). □

Example 4. We give an explicit construction of a nice error basis with $q = 4$ levels. The finite field $\mathbf{F}_4$ consists of the elements $\mathbf{F}_4 = \{0, 1, \alpha, \overline{\alpha}\}$. We denote the four standard basis vectors of the complex vector space $\mathbf{C}^4$ by $|0\rangle, |1\rangle, |\alpha\rangle$, and $|\overline{\alpha}\rangle$. Let $\mathbf{1}_2$ denote the 2×2 identity matrix, $\sigma_x = \left(\begin{smallmatrix} 0 & 1 \\ 1 & 0 \end{smallmatrix}\right)$, and $\sigma_z = \left(\begin{smallmatrix} 1 & 0 \\ 0 & -1 \end{smallmatrix}\right)$. Then

$$\begin{aligned} &X(0) = \mathbf{1}_2 \otimes \mathbf{1}_2, \quad X(1) = \mathbf{1}_2 \otimes \sigma_x, \quad X(\alpha) = \sigma_x \otimes \mathbf{1}_2, \quad X(\overline{\alpha}) = \sigma_x \otimes \sigma_x, \\ &Z(0) = \mathbf{1}_2 \otimes \mathbf{1}_2, \quad Z(1) = \sigma_z \otimes \mathbf{1}_2, \quad Z(\alpha) = \sigma_z \otimes \sigma_z, \quad Z(\overline{\alpha}) = \mathbf{1}_2 \otimes \sigma_z. \end{aligned}$$

We see that this nice error basis is obtained by tensoring the Pauli basis, a nice error basis on $\mathbf{C}^2$. The next lemma shows that this is a general design principle for nice error bases.

Lemma 5. *If $\mathcal{E}_1$ and $\mathcal{E}_2$ are nice error bases, then*

$$\mathcal{E} = \{E_1 \otimes E_2 \,|\, E_1 \in \mathcal{E}_1, E_2 \in \mathcal{E}_2\}$$

is a nice error basis as well.

The proof of this simple observation follows directly from the definitions.

Let $\mathbf{a} = (a_1, \ldots, a_n) \in \mathbf{F}_q^n$. We write $X(\mathbf{a}) = X(a_1) \otimes \cdots \otimes X(a_n)$ and $Z(\mathbf{a}) = Z(a_1) \otimes \cdots \otimes Z(a_n)$ for the tensor products of n error operators. Our aim was to provide an error model that conveniently represents errors acting locally on one quantum system. Using the new notations, we can easily formulate this model.

Corollary 6. *The set $\mathcal{E}_n = \{X(\mathbf{a})Z(\mathbf{b}) \,|\, \mathbf{a}, \mathbf{b} \in \mathbf{F}_q^n\}$ is a nice error basis on the complex vector space $\mathbf{C}^{q^n}$.*

Let G_n denote the group generated by the matrices of the nice error basis $\mathcal{E}_n$. It follows from equation (2.4) that

$$G_n = \{\omega^c X(\mathbf{a})Z(\mathbf{b}) \,|\, \mathbf{a}, \mathbf{b} \in \mathbf{F}_q^n, c \in \mathbf{F}_p\}.$$

Note that G_n is a finite group of order pq^{2n}. We call G_n the *error group* associated with the nice error basis $\mathcal{E}_n$.

It turns out that a stabilizer code Q with stabilizer S can detect all errors in G_n that are scalar multiples of elements in S or that do not commute with some element of S; see Lemma 10. The following characterization of the commutation properties is instrumental in the construction of stabilizer codes.

Lemma 7. *Two elements $E = \omega^c X(\mathbf{a})Z(\mathbf{b})$ and $E' = \omega^{c'} X(\mathbf{a}')Z(\mathbf{b}')$ of the error group G_n satisfy the relation*

$$EE' = \omega^{\operatorname{tr}(\mathbf{b}\cdot\mathbf{a}'-\mathbf{b}'\cdot\mathbf{a})} E'E,$$

where $b \cdot a' = \sum_{i=1}^n b_i a_i'$ and $b' \cdot a = \sum_{i=1}^n b_i' a_i$, is the standard inner product. In particular, the elements E and E' commute if and only if the trace symplectic form $\operatorname{tr}(\mathbf{b}\cdot\mathbf{a}'-\mathbf{b}'\cdot\mathbf{a})$ vanishes.

Proof. It follows from equation (2.4) that $EE' = \omega^{\operatorname{tr}(\mathbf{b}\cdot\mathbf{a}')} X(\mathbf{a}+\mathbf{a}')Z(\mathbf{b}+\mathbf{b}')$ and $E'E = \omega^{\operatorname{tr}(\mathbf{b}'\cdot\mathbf{a})} X(\mathbf{a}+\mathbf{a}')Z(\mathbf{b}+\mathbf{b}')$. Therefore, multiplying $E'E$ with the scalar $\omega^{\operatorname{tr}(\mathbf{b}\cdot\mathbf{a}'-\mathbf{b}'\cdot\mathbf{a})}$ yields EE', as claimed. □

An error $\omega^c X(\mathbf{a})Z(\mathbf{b})$ of G_n is said to have weight d if and only if it has d tensor components that are different from the identity,

$$d = |\{\, k \,|\, (a_k, b_k) \neq (0, 0)\}|.$$

Remark: The definition of the weight of an error as stated above corresponds to the symplectic weight, denoted as swt, which is different from the Hamming weight of an error discussed earlier.

Let us define a map Fix that associates to a subgroup S of G_n its joint eigenspace with eigenvalue 1,

$$\mathrm{Fix}(S) = \bigcap_{E \in S} \{v \in \mathbf{C}^{q^n} \mid Ev = v\}. \tag{2.5}$$

We also define a map Stab that associates to a quantum code Q its stabilizer group $\mathrm{Stab}(Q)$,

$$\mathrm{Stab}(Q) = \{E \in G_n \mid Ev = v \text{ for all } v \in Q\}. \tag{2.6}$$

Lemma 8. *If Q is a nonzero subspace of $\mathbf{C}^{q^n}$, then its stabilizer $S = \mathrm{Stab}(Q)$ is an abelian group satisfying $S \cap Z(G_n) = \{1\}$.*

Proof. Suppose that E and E' are non-commuting elements of $S = \mathrm{Stab}(Q)$. By Lemma 7, we have $EE' = \omega^k E'E$ for some $\omega^k \neq 1$. A nonzero vector v in Q would have to satisfy $v = EE'v = \omega^k E'Ev = \omega^k v$, a contradiction. Therefore, S is an abelian group. The stabilizer cannot contain any element $\omega^k \mathbf{1}$, unless $k = 0$, which proves the second assertion. □

Lemma 9. *Suppose that S is the stabilizer of a vector space Q. An orthogonal projector onto the joint eigenspace* $\mathrm{Fix}(S)$ *is given by*

$$P = \frac{1}{|S|} \sum_{E \in S} E.$$

Proof. A vector v in $\mathrm{Fix}(S)$ satisfies $Pv = v$, hence, $\mathrm{Fix}(S)$ is contained in the image of P. Conversely, note that $EP = P$ holds for all E in S, hence, any vector in the image of P is an eigenvector with eigenvalue 1 of all error operators E in S. Therefore, $\mathrm{Fix}(S) = \mathrm{image}\, P$. The operator P is idempotent because

$$P^2 = \frac{1}{|S|} \sum_{E \in S} EP = \frac{1}{|S|} \sum_{E \in S} P = P$$

holds. The inverse $E^\dagger$ of E is contained in the group S, hence, $P^\dagger = P$. Therefore, P is an orthogonal projector onto $\mathrm{Fix}(S)$. □

Remark: If S is a nonabelian subgroup of the group G_n, then it necessarily contains the center $Z(G_n)$ of G_n; it follows that P is equal to the all-zero matrix. Note that the image of P has dimension $\operatorname{tr} P$.

2. From q-ary Quantum Codes to q-ary Classical Codes

We are now ready to construct the first part of our bridge that connects q-ary quantum stabilizer codes to q-ary self-orthogonal classical codes.

Lemma 10. *Suppose that $S \le G_n$ is the stabilizer group of a stabilizer code Q. An error E in G_n is detectable by the code Q if and only if either E is an element of $SZ(G_n)$ or E does not belong to $C(S)$.*

Proof. An element E in $SZ(G_n)$ is a scalar multiple of a stabilizer; thus, it acts by multiplication with a scalar λ_E on Q. It follows that E is a detectable error.

Suppose now that E is an error in G_n that does not commute with some element F of the stabilizer S; it follows that $EF = \lambda FE$ for some complex number $\lambda \neq 1$; see Lemma 7. All vectors u and v in Q satisfy the condition

$$\langle u|E|v\rangle = \langle u|EF|v\rangle = \lambda\langle u|FE|v\rangle = \lambda\langle u|E|v\rangle;$$

hence, $\langle u|E|v\rangle = 0$. It follows that the error E is detectable.

Finally, suppose that E is an element of $C(S)\setminus SZ(G_n)$. Seeking a contradiction, we assume that E is detectable; this implies that there exists a complex scalar λ_E such that $Ev = \lambda_E v$ for all v in Q. Let S^* denote the abelian group generated by $\lambda_E^{-1}E$ and by the elements of S. The joint eigenspace of S^* with eigenvalue 1 has dimension $q^n/|S^*| < \dim Q = q^n/|S|$. This implies that not all vectors in Q remain invariant under $\lambda_E^{-1}E$, in contradiction to the detectability of E. □

Theorem 11. *An $[[n, q^k, d]]_q$ stabilizer code exists if and only if there exists an additive code $C \leq \mathbf{F}_q^{2n}$ of size $|C| = q^n/q^k$ such that $C \leq C^{\perp_s}$ and $\operatorname{swt}(C^{\perp_s} \setminus C) = d$.*

Proof. Suppose that an $[[n, q^k, d]]_q$ stabilizer code Q exists. By Lemma 8, there exists an abelian subgroup $S \leq G_n$, such that $S \cap Z(G_n) = 1$ and $G = \operatorname{Fix}(S)$. Also, from Lemma 9, we have $|S| = q^n/q^k$. Let $SZ(G_n)$ be the abelian group generated by the elements in S and $Z(G_n)$. The quotient $C \cong SZ(G_n)/Z(G_n)$ is an additive subgroup of $\mathbf{F}_q^{2n}$ such that $|C| = |S|$ because $SZ(G_n)$ trivially intersects the center $Z(G_n)$. Since the group $SZ(G_n)$ is abelian, C is contained in its trace-symplectic dual $C^{\perp_s}$ by Lemma 7. The stabilizer code Q has minimum distance d; therefore, the elements in $C(S) \setminus SZ(G_n)$ have at least weight d by Lemma 10. We have $C^{\perp_s} = C(S)/Z(G_n)$, so $\operatorname{swt}(C^{\perp_s} \setminus C) = d$ because the weight of an element $\omega^c X(\mathbf{a})Z(\mathbf{b})$ is equal to $\operatorname{swt}(\mathbf{a}|\mathbf{b})$.

Conversely, suppose that C is an additive subcode of $\mathbf{F}_q^{2n}$ such that $|C| = q^n/q^k$, $C \leq C^{\perp_s}$, and $\operatorname{swt}(C^{\perp_s} \setminus C) = d$. Let

$$N = \{\omega^c X(\mathbf{a})Z(\mathbf{b}) \,|\, c \in \mathbf{F}_p \text{ and } (\mathbf{a}|\mathbf{b}) \in C\}.$$

Notice that N is an abelian normal subgroup of G_n because it is the pre-image of $C = N/Z(G_n)$. Choose a character χ of N such that $\chi(\omega^c \mathbf{1}) = \omega^c$. Then

$$P_N = \frac{1}{|N|} \sum_{E \in N} \chi(E^{-1})E$$

is an orthogonal projector onto a vector space Q because P_N is an idempotent in the group ring $\mathbf{C}[G_n]$. We have

$$\dim Q = \operatorname{tr} P_N = |Z(G_n)|q^n/|N| = q^n/|C| = q^k.$$

Each coset of N modulo $Z(G_n)$ contains exactly one matrix E such that $Ev = v$ for all v in Q. Set $S = \{E \in N \mid Ev = v \text{ for all } v \in Q\}$. Then S is an abelian subgroup of G_n of order $|S| = |C| = q^n/q^k$. We have $Q = \mathrm{Fix}(S)$ because Q is clearly a subspace of $\mathrm{Fix}(S)$, but $\dim Q = q^n/|S| = q^k$. An element $\omega^c X(\mathbf{a})Z(\mathbf{b})$ in $C(S) \setminus SZ(G_n)$ cannot have weight less than d because this would imply that $(\mathbf{a}|\mathbf{b}) \in C^{\perp_s} \setminus C$ has weight less than d, which is impossible. Therefore, Q is an $[[n, q^k, d]]_q$ stabilizer code. □

3. From q-ary Codes to Codes over $\mathbf{F}_{q^2}$

We now construct the second part of the bridge that connects q-ary classical codes to q^2-ary classical codes. Let (β, β^q) denote a normal basis of $\mathbf{F}_{q^2}$ over $\mathbf{F}_q$. We define a trace-alternating form of two vectors v and w in $\mathbf{F}_{q^2}^n$ by

$$\langle v|w\rangle_a = \mathrm{tr}_{q/p}\left(\frac{v \cdot w^q - v^q \cdot w}{\beta^{2q} - \beta^2}\right). \tag{2.7}$$

We note that the argument of the trace is invariant under the Galois automorphism $x \mapsto x^q$, so it is indeed an element of $\mathbf{F}_q$, which shows that (2.7) is well-defined.

The trace-alternating form is bi-additive, i.e, $\langle u+v|w\rangle_a = \langle u|w\rangle_a + \langle v|w\rangle_a$ and $\langle u|v+w\rangle_a = \langle u|v\rangle_a + \langle u|w\rangle_a$ holds for all $u, v, w \in \mathbf{F}_{q^2}^n$. It is $\mathbf{F}_p$-linear, but not $\mathbf{F}_q$-linear unless $q = p$. And it is alternating in the sense that $\langle u|u\rangle_a = 0$ holds for all $u \in \mathbf{F}_{q^2}^n$. We write $u \perp_a w$ if and only if $\langle u|w\rangle_a = 0$ holds.

We define a bijective map ϕ that takes an element $(\mathbf{a}|\mathbf{b})$ of the vector space $\mathbf{F}_q^{2n}$ to a vector in $\mathbf{F}_{q^2}$ by setting $\phi((\mathbf{a}|\mathbf{b})) = \beta\mathbf{a} + \beta^q\mathbf{b}$. The map ϕ is isometric in the sense that the symplectic weight of $(\mathbf{a}|\mathbf{b})$ is equal to the Hamming weight of $\phi((\mathbf{a}|\mathbf{b}))$.

Lemma 12. *Suppose that c and d are two vectors of $\mathbf{F}_q^{2n}$. Then*

$$\langle c|d\rangle_s = \langle \phi(c)|\phi(d)\rangle_a.$$

In particular, c and d are orthogonal with respect to the trace-symplectic form if and only if $\phi(c)$ and $\phi(d)$ are orthogonal with respect to the trace-alternating form.

Proof. Let $c = (\mathbf{a}|\mathbf{b})$ and $d = (\mathbf{a}'|\mathbf{b}')$. We calculate

$$\begin{aligned} \phi(c) \cdot \phi(d)^q &= \beta^{q+1}\,\mathbf{a} \cdot \mathbf{a}' + \beta^2\,\mathbf{a} \cdot \mathbf{b}' + \beta^{2q}\,\mathbf{b} \cdot \mathbf{a}' + \beta^{q+1}\,\mathbf{b} \cdot \mathbf{b}' \\ \phi(c)^q \cdot \phi(d) &= \beta^{q+1}\,\mathbf{a} \cdot \mathbf{a}' + \beta^{2q}\,\mathbf{a} \cdot \mathbf{b}' + \beta^2\,\mathbf{b} \cdot \mathbf{a}' + \beta^{q+1}\,\mathbf{b} \cdot \mathbf{b}'. \end{aligned}$$

Therefore, the trace-alternating form of $\phi(c)$ and $\phi(d)$ is given by

$$\langle \phi(c)|\phi(d)\rangle_a = \operatorname{tr}_{q/p}\left(\frac{\phi(c) \cdot \phi(d)^q - \phi(c)^q \cdot \phi(d)}{\beta^{2q} - \beta^2}\right) = \operatorname{tr}_{q/p}(\mathbf{b} \cdot \mathbf{a}' - \mathbf{a} \cdot \mathbf{b}'),$$

which is precisely the trace-symplectic form $\langle c \,|\, d\rangle_s$. □

Theorem 13. *An $[[n, q^k, d]]_q$ stabilizer code exists if and only if there exists an additive subcode D of $\mathbf{F}_{q^2}^n$ of cardinality $|D| = q^n/q^k$ that satisfies the weight condition $\mathrm{wt}(D^{\perp_a} \setminus D) = d$, and is self-orthogonal $D \leq D^{\perp_a}$.*

Proof. Theorem 11 shows that there exists an additive code $C \leq \mathbf{F}_q^{2n}$ satisfying $C \leq C^{\perp_s}$ and $\mathrm{swt}(C^{\perp_s} \setminus C) = d$. We obtain the result by applying the isometry ϕ, $D = \phi(C)$. □

Definition 14. Linear Stabilizer Code: If the associated classical code D over $\mathbf{F}_{q^2}$ is also linear, then the stabilizer code is said to be linear.

CHAPTER III

LINEAR PROGRAMMING BOUNDS

In this chapter, we derive constraints that additive and linear classical self-orthogonal codes over $\mathbf{F}_{q^2}$ need to satisfy. These constraints will be used to derive upper bounds on the minimum distance d of q-ary stabilizer codes where $q = 3, 4$, using linear programming methods. The minimum distance is an important parameter of a code because the error correcting capabilities of a code are directly related to its minimum distance d. To recap, for a code to correct t errors, its minimum distance should be at least equal to $2t+1$. For given values of n and k, a code with a higher value of d is obviously better than a code with a lower value of d. The bounds are useful because they automatically prove the non-existence of codes having certain values of n, k, d and place an upper limit on the highest achievable value of d for given values of the parameters n and k.

A. Linear Programming Bounds for Additive Codes

Knill and Laflamme [12] have shown that the distance d of a quantum code is related to the parameters n and k by

$$d \leq \frac{n-k+2}{2}. \tag{3.1}$$

Equation 3.1 is the quantum analog of the Singleton bound for classical codes. Using linear programming methods, we derive bounds that are tighter than those obtained for the quantum Singletom bound 3.1.

1. Notations and Terminology

Suppose an $[[n,k,d]]_q$ code exists. Let D be the corresponding (n, q^{n-k}) code over $\mathbf{F}_{q^2}$ and let $D^{\perp_a}$, an (n, q^{n+k}) code, be its dual with respect to the trace-alternating form. The weight distribution $A = A(D) = (A_0, A_1, \ldots, A_n)$ is a vector of dimension $n+1$, where

$$A_i =\mid \{\overline{x} \in D \mid w(\overline{x} = i\} \mid, \tag{3.2}$$

that is, the i^{th} component of A is the number of codewords of Hamming weight i in D. Analogously, the weight distribution $A' = A'(D^{\perp_a}) = (A'_0, A'_1, \ldots, A'_n)$ of the dual code $D^{\perp_a}$ is also a vector of dimension $n+1$, where the the i^{th} component of A' is the number of codewords of weight i in $D^{\perp_a}$.

Krawtchouk polynomials are a family of orthogonal polynomials that have been used extensively in coding theory. The generating function of Krawtchouk polynomials [14, page 20] is given by

$$\sum_{j=0}^{n} P_j(x,n) z^j = (1 + (q^2-1)z)^{n-x}(1-z)^x. \tag{3.3}$$

In other words, the coefficient of z^j in $(1+(q^2-1)z)^{n-x}(1-z)^x$ is given by the Krawtchouk polynomial $P_j(x,n)$, where $P_j(x,n)$ is defined as

$$P_j(x,n) = \sum_{s=0}^{j} (-1)^s (q^2-1)^{j-s} \begin{pmatrix} x \\ s \end{pmatrix} \begin{pmatrix} n-x \\ j-s \end{pmatrix}. \tag{3.4}$$

2. Constraints for Additive Codes

Theorem 15. *Let $D \leq \mathbf{F}_{q^2}^n$ be an additive code with weight enumerator $A(z)$, and let $D^{\perp_a}$ denote its trace-alternating dual code. The weight-enumerator $A'(z)$ of the code*

$D^{\perp_a}$ *is given by*

$$A'(z) = \frac{(1+(q^2-1)z)^n}{|D|} A\left(\frac{1-z}{1+(q^2-1)z}\right).$$

Proof. We define for $b \in \mathbf{F}_{q^2}^n$ and $c \in D$, a character $\chi_b(c)$ of the additive group D, such that

$$\chi_b(c) = \chi(\langle c|b\rangle_a).$$

The character χ_b is trivial if and only if b is an element of $D^{\perp_a}$. Therefore, we obtain from the orthogonality relations of characters that

$$\sum_{c\in D} \chi_b(c) = \begin{cases} |D| & \text{for } b \in D^{\perp_a}, \\ 0 & \text{otherwise.} \end{cases}$$

The following relation for polynomials is an immediate consequence

$$\sum_{c\in D}\sum_{b\in\mathbf{F}_{q^2}^n} \chi_b(c) z^{\mathrm{wt}(b)} = \sum_{b\in\mathbf{F}_{q^2}^n} z^{\mathrm{wt}(b)} \sum_{c\in D}\chi_b(c) = |D|A'(z). \tag{3.5}$$

The right hand side is a multiple of the weight enumerator of the code $D^{\perp_a}$. Let us have a closer look at the inner sum of the left-hand side. If we express the vector $c \in D$ in the form $c = (c_1, \dots, c_n)$, then we obtain

$$\begin{aligned} \sum_{b\in\mathbf{F}_{q^2}^n} \chi_b(c) z^{\mathrm{wt}(b)} &= \sum_{(b_1,\dots,b_n)\in\mathbf{F}_{q^2}^n} z^{\sum_{k=1}^n \mathrm{wt}(b_k)} \prod_{k=1}^n \chi_{b_k}(c_k) \\ &= \sum_{(b_1,\dots,b_n)\in\mathbf{F}_{q^2}^n} \prod_{k=1}^n z^{\mathrm{wt}(b_k)} \chi_{b_k}(c_k) \\ &= \prod_{k=1}^n \sum_{b_k\in\mathbf{F}_{q^2}} z^{\mathrm{wt}(b_k)} \chi_{b_k}(c_k). \end{aligned}$$

Now, from the orthogonality relations of a character, it follows that

$$\sum_{b_k \in \mathbf{F}_{q^2}} z^{\mathrm{wt}(b_k)} \chi_{b_k}(c_k) = \begin{cases} 1 + (q^2 - 1)z & \text{if } c_k = 0, \\ 1 - z & \text{if } c_k \neq 0. \end{cases}$$

It follows that

$$\sum_{b \in \mathbf{F}_{q^2}^n} \chi_b(c) z^{\mathrm{wt}(b)} = (1 - z)^{\mathrm{wt}(c)} (1 + (q^2 - 1)z)^{n - \mathrm{wt}(c)}.$$

Substituting this expression into equation (3.5), we find that

$$\begin{aligned} A'(z) &= |D|^{-1} \sum_{c \in D} \sum_{b \in \mathbf{F}_{q^2}^n} \chi_b(c) z^{\mathrm{wt}(b)} \\ &= \frac{(1 + (q^2 - 1)z)^n}{|D|} \sum_{c \in D} \left(\frac{1 - z}{1 + (q^2 - 1)z} \right)^{\mathrm{wt}(c)} \\ &= \frac{(1 + (q^2 - 1)z)^n}{|D|} A\left(\frac{1 - z}{1 + (q^2 - 1)z} \right), \end{aligned}$$

which proves the claim. □

Lemma 16. *The weight distribution of the dual code $D^{\perp_a}$ and the weight distribution of the code D are related as*

$$A_j^{'} = \sum_{r=0}^{n} A_r P_j(r, n). \tag{3.6}$$

Proof. From the above theorem, the weight-enumerator $A'(z)$ of the code $D^{\perp_a}$ is given by

$$A'(z) = \frac{(1 + (q^2 - 1)z)^n}{|D|} A\left(\frac{1 - z}{1 + (q^2 - 1)z)} \right).$$

Hence,

$$\begin{aligned} \sum_{j=0}^{n} A_j' z^j &= \frac{(1 + (q^2 - 1)z)^n}{|D|} \sum_{r=0}^{n} A_r \left(\frac{1 - z}{1 + (q^2 - 1)z)} \right)^r \\ &= \frac{1}{|D|} \sum_{r=0}^{n} A_r (1 - z)^r (1 + (q^2 - 1)z)^{n - r}. \end{aligned} \tag{3.7}$$

Equating the coefficients of z^j on both sides, we get

$$A'_j = \sum_{r=0}^{n} A_r P_j(r, n). \tag{3.8}$$

□

Lemma 17. *If D is an $[n, k, d]_{q^2}$ classical linear code, then*

$$(q^2 - 1) \mid A_i \quad (1 \leq i \leq n). \tag{3.9}$$

Proof. If $0 \neq x \in D$, then for each non-zero $\alpha \in \mathbf{F}_{q^2}$, $\alpha x \in D$ and $w(\alpha x) = w(x)$, where w denotes the Hamming weight. Hence, the number of codewords of weight i is a multiple of $q^2 - 1$. □

Lemma 18. *If D is an $[n, k, d]_{q^2}$ additive code, then*

$$(p - 1) \mid A_i \quad (1 \leq i \leq n). \tag{3.10}$$

Theorem 19. *If an $[[n, k, d]]_q$ quantum-error-correcting code exists such that the associated additive (n, q^{n-k}) code D is self-orthogonal with respect to the trace-alternating form and $D^{\perp_a} \backslash D$ contains no vectors of weight $< d$, then there is a solution to the following set of linear equations.*

1. $A_0 = 1, A_j \geq 0 \quad (1 \leq j \leq n)$
2. $A_0 + A_1 + \cdots + A_n = q^{n-k}$
3. $A_j^{'} = \frac{1}{q^{n-k}} \sum_{r=0}^{n} P_j(r, n) A_r \quad (0 \leq j \leq n)$
4. $A_j = A_j^{'} \quad (0 \leq j \leq d - 1),\ A_j \leq A_j^{'}\ (d \leq j \leq n)$
5. $(p - 1) \mid A_j \quad (1 \leq j \leq n),$

where A_j and $A_j^{'}$ denote the weight distribution, i.e., the number of vectors in D and $D^{\perp_a}$ of weight j, respectively.

Proof. Constraint (1) follows from the fact that D is additive and, hence, the identity element, which is the all *zero vector*, has to be present, and the weight distribution is non-negative. Constraint (2) follows from the fact the weight distributions should sum up to the number of vectors in D. Constraint (3) is a direct consequence of Lemma 16. Constraint (4) follows from the facts that $D \subset D^{\perp_a}$ and all vectors in $D^{\perp_a}$ of weights between 1 and $d-1$ inclusive must also be in D. Constraint (5) follows from lemma 18. □

We observe that the constraints depend only on the parameters of the code, namely n, k, d and q. The way we have applied the theorem in finding the upper bounds is to find whether a solution exists for a given set of values of n, k, d and q. If a solution does not exist, then we conclude that no code can exist for the given parameters. We then find whether a solution exists for the next lower value of d. We repeat this process until we find a solution for some set of input values n, k, d and q. The value of d obtained at this point is then the upper bound for the given values of n, k and q. At this point, the bounds obtained are for the classical codes. However, we can easily translate these bounds to the quantum domain by means of Theorem 13. This is because from Theorem 13, the non-existence of a classical code automatically implies the non-existence of the corresponding quantum code.

B. Linear Programming Bounds for Linear Codes

The constraints outlined in the previous section determine the non-existence of certain additive codes and, hence, the non-existence of the corresponding additive quantum codes by means of Theorem 13. These constraints then help to derive upper bounds on d. However, it is of practical interest to see if we can have linear stabilizer codes meeting the bounds derived for the additive stabilizer codes. This is because it is conjectured that linear codes have a more convenient form as compared to additive codes, and this property facilitates the design of encoding circuits for the same. For example, the stabilizer of the linear five qubit code [4, page 97] has 16 elements: the identity, and 3×5 cyclic permutations of $X \otimes Z \otimes Z \otimes X \otimes I$, $Y \otimes X \otimes X \otimes Y \otimes I$ and $Z \otimes Y \otimes Y \otimes Z \otimes I$. In this section, we derive constraints that classical $\mathbf{F}_{q^2}$-linear codes need to satisfy. These constraints will be modelled using an integer linear optimization package such as Maple [15], to get more information about the code such as the weight distribution. This information about the weight distribution can be used to optimize programs that exhaustively search for codes.

1. Background

First, we review some fundamental concepts that will be required for an understanding of the proofs later. Let $\mathbf{F}_{q^2} = \{\alpha_i | i = 0, 1, \ldots, q^2 - 1\}$ denote the set of field elements. For every vector $x \in \mathbf{F}_{q^2}^n$, we define $w_i(x) = |\{i | x_i = \alpha_i\}|$.

Definition 20 (Complete Weight Enumerator:). The complete weight enumerator of a code D is a homogeneous polynomial of degree n in q^2 variables given by

$$W_D(X_0, X_1, \ldots, X_{q^2-1}) = \sum_{x \in D} X_0^{w_0(x)} X_1^{w_1(x)} \ldots X_{q^2-1}^{w_{q^2-1}(x)}. \tag{3.11}$$

Let $\chi_a(b)$ be an asymmetric character defined over $\mathbf{F}_{q^2}^n$ as follows:

$$\chi_a(b) = \exp\left(\frac{j2\pi}{p}\langle a, b\rangle_a\right) = \exp\left(\frac{j2\pi}{p}\operatorname{tr}(\frac{ab^q - a^q b}{\beta^2 - \beta^{2q}})\right), \tag{3.12}$$

where $a, b \in \mathbf{F}_{q^2}^n$ and $\{\beta, \beta^2\}$ form a normal basis of $\mathbf{F}_{q^2}$ over $\mathbf{F}_q$.

Let f be a function defined on $\mathbf{F}_{q^2}^n$. The Fourier transform of f, denoted by $\hat{f}$, is defined in terms of the characters over $\mathbf{F}_{q^2}^n$,

$$\hat{f}(x) = \sum_{y \in \mathbf{F}_{q^2}^n} \chi_x(y) f(y). \tag{3.13}$$

Lemma 21 (Poisson's theorem). *For any code D and its trace-alternating dual code $D^{\perp_a}$, we have*

$$\sum_{x \in D^{\perp_a}} f(x) = \frac{1}{|D|} \sum_{y \in D} \hat{f}(y). \tag{3.14}$$

Proof. Our proof is similar to the proof in [14, page 11] for duality with respect to the standard inner product. We note that

$$\begin{aligned} \sum_{y \in D} \hat{f}(y) &= \sum_{y \in D} \sum_{x \in \mathbf{F}_{q^2}^n} \chi_y(x) f(x) = \sum_{x \in \mathbf{F}_{q^2}^n} f(x) \sum_{y \in D} \chi_y(x) \\ &= \sum_{x \in \mathbf{F}_{q^2}^n} f(x) \sum_{y \in D} \chi_x(-y) = \sum_{x \in \mathbf{F}_{q^2}^n} f(x) \sum_{z \in D} \chi_x(z). \end{aligned}$$

If x in $D^{\perp_a}$, then χ_x is the trivial character and, hence, $\sum_{z \in D} \chi_x(z) = |D|$. If x is not in $D^{\perp_a}$, then there exists u in D such that $\chi_x(u) \neq 1$. Hence,

$$(1 - \chi_x(u)) \sum_{z \in D} \chi_x(z) = \sum_{z \in D} \chi_x(z) - \sum_{z \in D} \chi_x(u+z) = 0.$$

Therefore, we get

$$\sum_{y \in D} \hat{f}(y) = |D| \sum_{x \in D^{\perp_a}} f(x). \tag{3.15}$$

□

Theorem 22 (MacWilliams). *For any linear code D over $\mathbf{F}_{q^2}^n$, the complete weight enumerator of $D^{\perp_a}$, dual with respect to the trace-alternating form is given by the following relation:*

$$W_{D^{\perp_a}}(X_0, X_1, \ldots, X_{q^2-1}) = \frac{1}{|D|} W_D(Z_0, Z_1, \ldots, Z_{q^2-1}), \tag{3.16}$$

where $Z_i = \sum_{j=0}^{q^2-1} \chi_{\alpha_i}(\alpha_j) X_j$.

Proof. Our proof is similar to the proof in [14, page 11] for duality with respect to the standard inner product. We have

$$\hat{f}(u) = \sum_{x \in \mathbf{F}_{q^2}^n} \chi_u(x) X_0^{w_0(x)} X_1^{w_1(x)} \dots X_{q^2-1}^{w_{q^2-1}(x)}.$$

Now, $\chi_u(x) = \prod_{i=1}^n \chi_{u_i}(x_i)$ and for $\alpha \in \mathbf{F}_{q^2}$,

$$w_\alpha(x) = \delta_{\alpha,x_1} + \delta_{\alpha,x_2} + \dots + \delta_{\alpha,x_n}, \tag{3.17}$$

where δ is the Kronecker delta. Then $\hat{f}(u)$ can be written as

$$\begin{aligned} \hat{f}(u) &= \sum_{x \in \mathbf{F}_{q^2}^n} \prod_{i=1}^n \left(\chi_{u_i}(x_i) X_0^{\delta_{0,x_i}} X_1^{\delta_{1,x_i}}, \dots, X_{q^2-1}^{\delta_{q^2-1},x_i} \right) \\ &= \sum_{x_1 \in \mathbf{F}_{q^2}} \cdots \sum_{x_n \in \mathbf{F}_{q^2}} \prod_{i=1}^n \left(\chi_{u_i}(x_i) X_0^{\delta_{0,x_i}} X_1^{\delta_{1,x_i}}, \dots, X_{q^2-1}^{\delta_{q^2-1},x_i} \right) \end{aligned} \tag{3.18}$$

Expanding the product in equation 3.18, we get

$$\begin{aligned} \hat{f}(u) &= \left(\sum_{j=0}^{q^2-1} \chi_{u_1}(\alpha_j) X_j \right) \times \left(\sum_{j=0}^{q^2-1} \chi_{u_2}(\alpha_j) X_j \right) \times \dots \times \left(\sum_{j=0}^{q^2-1} \chi_{u_n}(\alpha_j) X_j \right) \\ &= \prod_{i=1}^n \left(\sum_{j=0}^{q^2-1} \chi_{u_i}(\alpha_j) X_j \right) = \prod_{i=0}^{q^2-1} \left(\sum_{j=0}^{q^2-1} \chi_{u_i}(\alpha_j) X_j \right)^{w_i(u)} = \prod_{j=0}^{q^2-1} Z_i^{w_i(u)} \\ &= Z_0^{w_0(u)} Z_1^{w_1(u)} \dots Z_{q^2-1}^{w_{q^2-1}(u)}. \end{aligned} \tag{3.19}$$

From Poisson's theorem, we have

$$\sum_{x \in D^{\perp a}} f(x) = \frac{1}{|D|} \sum_{u \in D} \hat{f}(u).$$

Consequently,

$$\sum_{x \in D^{\perp a}} X_0^{w_0(x)} X_1^{w_1(x)} \dots X_{q^2-1}^{w_{q^2-1}(x)} = \frac{1}{|D|} \sum_{u \in D} Z_0^{w_0(u)} Z_1^{w_1(u)} \dots Z_{q^2-1}^{w_{q^2-1}(u)},$$

which proves that

$$W_{D^{\perp_a}}(X_0, X_1, \ldots, X_{q^2-1}) = \frac{1}{|D|} W_D(Z_0, Z_1, \ldots, Z_{q^2-1}) \tag{3.20}$$

holds. □

Definition 23 (Even-like codes:). A codeword $c \in D$ is said to be *even-like* if $\sum_{i=0}^{n} c_i = 0$, else it is said to be *odd-like*. A code D is even-like if all the codewords are even-like, else it is called odd-like. The complete weight enumerator can be used to derive an important relationship between the even-like codewords of a code D and the weight enumerator of the dual code $D^{\perp_a}$.

Theorem 24 (MacWilliams). *For an $\mathbf{F}_{q^2}$-linear code D, let A_{0_i} be the number of even-like codewords of weight i, $A_i - A_{0_i}$ the number of odd-like codewords of weight i. Let $B_i^{(1)}$ be the number of codewords in $D^{\perp_a}$, which have i components equal to 1. Then, we have*

$$\sum_{i=0}^{n} B_i^{(1)} X^i Y^{n-i} = \frac{1}{|D|} \sum_{i=0}^{n} \left(A_{0_i} - \frac{A_i' - A_{0_i}}{q^2 - 1} \right) \big(X + (q^2 - 1)Y\big)^{n-i} (X - Y)^i. \tag{3.21}$$

Proof. Our proof is similar to the proof in [7] for duality with respect to the standard inner product. From Theorem 22, the complete weight enumerator is given by:

$$W_{D^{\perp_a}}(X_0, X_1, \ldots, X_{q^2-1}) = \frac{1}{|D|} W_D(Z_0, Z_1, \ldots, Z_{q^2-1}). \tag{3.22}$$

Replace $X_1 = X$ and $X_i = Y$ for $i \neq 1$. Then,

$$\begin{aligned} W_{D^{\perp_a}}(Y, X, Y \ldots, Y) &= \sum_{x \in D^{\perp_a}} X^{w_1(x)} Y^{n - w_1(x)} \\ &= \sum_{i=0}^{n} B_i^{(1)} X^i Y^{n-i}. \end{aligned} \tag{3.23}$$

Note that from the orthogonality relations of the characters, we have $Z_0 = (X + (q^2 -$

$1)Y)$ and $Z_i = (X-Y)\chi_{\alpha_i}(1) = (X-Y)\chi_1(-\alpha_i)$ for $i \geq 1$. Also,

$$W_D(Z_0, Z_1, \ldots, Z_{q^2-1}) = \sum_{x \in D} Z_0^{w_0(x)} Z_1^{w_1(x)} \ldots Z_{q^2-1}^{w_{q^2-1}(x)} \tag{3.24}$$

$$\begin{aligned} &= \sum_{x \in D} (X + (q^2-1)Y)^{n-w(x)} (X-Y)^{w(x)} \left[\prod_{j=1}^{q^2-1} (\chi_1(-\alpha_j))^{w_j(x)} \right] \\ &= \sum_{x \in D} (X + (q^2-1)Y)^{n-w(x)} (X-Y)^{w(x)} \chi_1(-\sum_{j=1}^{q^2-1} \alpha_j w_j(x)) \end{aligned}$$

$$= \sum_{x \in D} (X + (q^2-1)Y)^{n-w(x)} (X-Y)^{w(x)} \chi_1(-\sum_{j=1}^{n} x_j). \tag{3.25}$$

Since D is $\mathbf{F}_{q^2}$-linear, for any x in D and α in $\mathbf{F}_{q^2}$, αx is also in D. All these codewords have same terms in equation (3.23). So we have

$$\sum_{x \in D, w(x)=i} (X + (q^2-1)Y)^{n-w(x)} (X-Y)^{w(x)} \chi_1(-\sum_{j=1}^{n} x_j)$$

$$= \sum_{x \in D, w(x)=i} (X + (q^2-1)Y)^{n-w(\alpha x)} (X-Y)^{w(\alpha x)} \chi_1(-\sum_{j=1}^{n} \alpha x_j) \tag{3.26}$$

$$= \frac{1}{q^2-1} \sum_{\alpha \in \mathbf{F}_{q^2}^{\times}} \sum_{x \in D, w(x)=i} (X + (q^2-1)Y)^{n-w(\alpha x)} (X-Y)^{w(\alpha x)} \chi_1(-\sum_{j=1}^{n} \alpha x_j).$$

Where the last equality is obtained by summing equation (3.26) over all the non-zero field elements. Hence,

$$\frac{1}{q^2-1} \sum_{\alpha \in \mathbf{F}_{q^2}^{\times}} \sum_{x \in D, w(x)=i} (X + (q^2-1)Y)^{n-w(\alpha x)} (X-Y)^{w(\alpha x)} \chi_1(-\sum_{j=1}^{n} \alpha x_j)$$

$$= \frac{1}{q^2-1} (A_{0_i}(q^2-1) - (A_i - A_{0_i}))(X + (q^2-1)Y)^{n-i} (X-Y)^i, \tag{3.27}$$

where the last equality follows from the orthogonality relations of a character, viz.

$$\sum_{\alpha \in \mathbf{F}_{q^2}^{\times}} \chi_1(-\alpha \sum_{j=1}^{n} x_j) = \begin{cases} -1 & \text{if } \sum_{j=1}^{n} x_j \neq 0, \\ q^2 - 1 & \text{otherwise.} \end{cases}$$

Summing over all i and substituting in equation 3.24 and equating to equation 3.23 gives us the desired result. □

2. Constraints for Linear Codes

From [16], we also have the following important lemma, which we state without proof.

Lemma 25. *Let D be a linear $[n, k, d]_{q^2}$ classical code. Let D_e be the set of even-like codewords. Then either*

1. $|D_e| = |D|$, *or*
2. $|D_e| = \frac{|D|}{q^2}$.

Lemma 26. *Let D be a linear $[n, k, d]_{q^2}$ classical even-like code and $D^{\perp_a}$ its dual with respect to the trace-alternating form. Then $D^{\perp_a}$ contains the all one vector.*

Proof. Since D is even-like, for any $c \in D$, we have

$$\sum_{i=1}^{n} c_i = 0 = \sum_{i=1}^{n} c_i^q. \tag{3.28}$$

Hence,

$$\sum_{i=1}^{n} \frac{c_i^q - c_i}{\beta^2 - \beta^{2q}} = 0 = \langle 1, c \rangle_a. \tag{3.29}$$

□

Lemma 27. *Let D be a linear $[n, k, d]_{q^2}$ classical code and $D^{\perp_a}$ be its dual with respect to the trace-alternating form. Then $D^{\perp_a}$ is also linear.*

Proof. Suppose $x \in D$ and $y \in D^{\perp_a}$. Hence $\langle x, y\rangle_a = 0$. Since D is linear, $\langle \alpha x, y\rangle_a = 0$, where $\alpha \in \mathbf{F}_{q^2}$. Hence, we have $\mathrm{tr}_{q/p}\left(\frac{\langle \alpha x, \overline{y}\rangle - \langle \overline{\alpha x}, y\rangle}{\beta^2 - \beta^{2q}}\right) = \mathrm{tr}_{q/p}\left(\frac{\langle x, \overline{\alpha y}\rangle - \langle \overline{x}, \alpha y\rangle}{\beta^2 - \beta^{2q}}\right) = \langle x, \alpha y\rangle_a$ $= 0$. □

With all the requisite background at our disposal, we now present a set of constraints that linear classical self-orthogonal codes over $\mathbf{F}_{q^2}$ need to satisfy.

Theorem 28. *If an $[[n,k,d]]_q$ linear stabilizer code exists such that the associated linear (n, q^{n-k}) code D is self-orthogonal with respect to the trace-alternating form and $D^{\perp_a}\backslash D$ contains no vectors of weight $< d$, then there is a solution to at least one of the following three sets of linear equations: The first set of equations apply to the case when both D and $D^{\perp_a}$ are even-like. The second set applies to the case when D is even-like and $D^{\perp_a}$ is odd-like, and the third set applies to the case when both D and $D^{\perp_a}$ are odd-like. Constraints (1)-(5) are common to each of the three cases. In addition, constraints (6a)-(6b) apply to the first case, constraints (7a)-(7d) apply to the second case, and constraints (8a)-(8g) apply to the third case.*

1. $A_0 = 1, A_j \geq 0 \quad (1 \leq j \leq n)$
2. $A_0 + A_1 + \cdots + A_n = q^{n-k}$
3. $A_j^{'} = \frac{1}{q^{n-k}} \sum_{r=0}^{n} P_j(r,n) A_r \quad (0 \leq j \leq n)$
4. $A_j = A_j^{'} \quad (0 \leq j \leq d-1),\ A_j \leq A_j^{'}\ (d \leq j \leq n)$
5. $(q^2-1) \mid A_j \quad (1 \leq j \leq n)$
6. *(a)* $n \equiv 0 \bmod p$, *where* q *is a power of a prime* p

   (b) $A_n \geq q^2 - 1$
7. *(a)* $\sum_{i=0}^{n} B_{0_i} = q^{n+k-2}$

   (b) $A_i = B_{0_i} = A_i^{'}\,(0 \leq i \leq d-1)\ A_i \leq B_{0_i} \leq A_i^{'}\,(d \leq i \leq n)$

(c) $\sum_{i=0}^{n} A_i^{(1)} = q^{n-k}$

(d) $A_{n-i}^{(1)} = \sum_{j=0}^{n} \left(\left(B_{0_j} - \frac{A_j^{'} - B_{0_j}}{q^2-1} \right) P_i(j,n) \right)$

8. *(a)* $\sum_{i=0}^{n} B_{0_i} = q^{n+k-2}$

(b) $\sum_{i=0}^{n} A_i^{(1)} = q^{n-k}$

(c) $A_{n-i}^{(1)} = \sum_{j=0}^{n} \left(\left(B_{0_j} - \frac{A_j^{'} - B_{0_j}}{q^2-1} \right) P_i(j,n) \right)$

(d) $\sum_{i=0}^{n} A_{0_i} = q^{n-k-2}$

(e) $\sum_{i=0}^{n} B_i^{(1)} = q^{n+k}$

(f) $B_{n-i}^{(1)} = \sum_{j=0}^{n} \left(\left(A_{0_j} - \frac{A_j^{'} - A_{0_j}}{q^2-1} \right) P_i(j,n) \right)$

(g) $A_{0_j} = B_{0_j} \quad (0 \leq j \leq d-1),\ A_{0_j} \leq B_{0_j}\ (d \leq j \leq n)$

Proof. From Lemma 26, D contains the *all one* vector and also since D is even-like, constraint (6a) follows. Constraint (6b) follows from the fact that D is linear and, hence, all the $q^2 - 1$ non-zero multiples of the all one codeword are in D. Constraint (7a) follows from Lemma 25 and constraint (7d) follows from Theorem 24. Constraint (7b) is obtained by applying distance relationships to the weight enumerators of the even-like code D and the even-like subcode of dimension $\frac{|D|}{q^2}$ within $D^{\perp_a}$. Constraint (7c) follows from the fact that the sum of the weight distributions that count the number of 1's within codewords in D should be equal to the number of codewords in D. Constraints (8a) - (8g) are obtained by applying the same logic to the even-like subcodes of dimension $\frac{|D|}{q^2}$ and $\frac{|D^{\perp_a}|}{q^2}$ within D and $D^{\perp_a}$, respectively. □

CHAPTER IV

TABLE OF UPPER BOUNDS ON MINIMUM DISTANCE OF NONBINARY STABILIZER CODES

A. Table of Upper Bounds for Additive Codes

The set of linear equations for additive codes outlined in the previous chapters were modelled using the linear optimization package, Mathematica [17]. The package dynamically formulates the constraints given the input parameters n, k, d and q. The program then attempts to find a solution to the set of constraints formulated. If a solution exists, then the program displays the solutions or else reports that no feasible solution exists, in which case we can conclude that no $[[n, k, d]]_q$ code exists. In the case that a solution exists, we may also also learn some additional properties about the code like whether it is pure or impure, or whether it contains vectors of a particular weight or not. We have computed the table of upper bounds for values of n up to 30. We now present the table of upper bounds on d for q-ary additive quantum codes, where $q = 3, 4$. Table I, Table II and Table III outline the upper bounds for q = 3 and Table IV, Table V and Table VI outline the upper bounds for $q = 4$. Note that the entries in the tables that have a superscript of β are the ones for which we have derived upper bounds that are tighter than Singleton bound.

Table I. Upper bounds on d for a $[[n,k,d]]_3$ error-correcting-code (n:3..15, k:1..15)

n/k	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
3	2	1	1	-	-	-	-	-	-	-	-	-	-	-	-
4	2	2	1	1	-	-	-	-	-	-	-	-	-	-	-
5	3	2	2	1	1	-	-	-	-	-	-	-	-	-	-
6	3	3	2	2	1	1	-	-	-	-	-	-	-	-	-
7	4	3	3	2	2	1	1	-	-	-	-	-	-	-	-
8	4	4	3	3	2	2	1	1	-	-	-	-	-	-	-
9	5	4	4	3	3	2	2	1	1	-	-	-	-	-	-
10	5	5	4	4	3	3	2	2	1	1	-	-	-	-	-
11	6	5	5	4	4	3	2^β	2	2	1	1	-	-	-	-
12	6	6	5	5	4	3^β	3	2^β	2	2	1	1	-	-	-
13	7	6	6	5	4^β	4	3^β	3	2^β	2	2	1	1	-	-
14	7	7	6	5^β	5	4^β	4	3^β	3	2^β	2	2	1	1	-
15	8	7	6^β	6	5^β	5	4^β	4	3^β	3	2^β	2	2	1	1

Table II. Upper bounds on d for a $[[n,k,d]]_3$ error-correcting-code (n:16..30, k:1..15)

n/k	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
16	8	7^{β}	7	6^{β}	6	5^{β}	5	4^{β}	4	3^{β}	3	2^{β}	2	2	1
17	8^{β}	8	7^{β}	7	6^{β}	6	5^{β}	5	4^{β}	4	3^{β}	3	2^{β}	2	2
18	9	8^{β}	8	7^{β}	7	6^{β}	6	5^{β}	5	4^{β}	4	3^{β}	3	2^{β}	2
19	9^{β}	9	8^{β}	8	7^{β}	7	6^{β}	6	5^{β}	5	4^{β}	4	3^{β}	3	2^{β}
20	10	9^{β}	9	8^{β}	8	7^{β}	7	6^{β}	6	5^{β}	5	4^{β}	4	3^{β}	3
21	10^{β}	10	9^{β}	9	8^{β}	8	7^{β}	7	6^{β}	6	5^{β}	5	4^{β}	4	3^{β}
22	11	10^{β}	10	9^{β}	9	8^{β}	8	7^{β}	7	6^{β}	6	5^{β}	5	4^{β}	4
23	11^{β}	11	10^{β}	10	9^{β}	9	8^{β}	8	7^{β}	7	6^{β}	6	5^{β}	5	4^{β}
24	11^{β}	11^{β}	11	10^{β}	10	9^{β}	9	8^{β}	8	7^{β}	7	6^{β}	5^{β}	5^{β}	4^{β}
25	12^{β}	11^{β}	11^{β}	11	10^{β}	10	9^{β}	9	8^{β}	8	7^{β}	6^{β}	6^{β}	5^{β}	5^{β}
26	12^{β}	12^{β}	11^{β}	11^{β}	10^{β}	10^{β}	9^{β}	9^{β}	8^{β}	8^{β}	7^{β}	7^{β}	6^{β}	6^{β}	5^{β}
27	13^{β}	12^{β}	12^{β}	11^{β}	11^{β}	10^{β}	10^{β}	9^{β}	9^{β}	8^{β}	8^{β}	7^{β}	7^{β}	6^{β}	6^{β}
28	13^{β}	13^{β}	12^{β}	12^{β}	11^{β}	11^{β}	10^{β}	10^{β}	9^{β}	9^{β}	8^{β}	8^{β}	7^{β}	7^{β}	6^{β}
29	14^{β}	13^{β}	13^{β}	12^{β}	12^{β}	11^{β}	11^{β}	10^{β}	10^{β}	9^{β}	9^{β}	8^{β}	8^{β}	7^{β}	7^{β}
30	14^{β}	14^{β}	13^{β}	13^{β}	12^{β}	12^{β}	11^{β}	11^{β}	10^{β}	10^{β}	9^{β}	9^{β}	8^{β}	8^{β}	7^{β}

Table III. Upper bounds on d for a $[[n,k,d]]_3$ error-correcting-code (n:16..30, k:16..30)

n/k	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30
16	1	-	-	-	-	-	-	-	-	-	-	-	-	-	-
17	1	1	-	-	-	-	-	-	-	-	-	-	-	-	-
18	2	1	1	-	-	-	-	-	-	-	-	-	-	-	-
19	2	2	1	1	-	-	-	-	-	-	-	-	-	-	-
20	2^β	2	2	1	1	-	-	-	-	-	-	-	-	-	-
21	3	2^β	2	2	1	1	-	-	-	-	-	-	-	-	-
22	3^β	3	2^β	2	2	1	1	-	-	-	-	-	-	-	-
23	4	3^β	3	2^β	2	2	1	1	-	-	-	-	-	-	-
24	4^β	4	3^β	3	2^β	2	2	1	1	-	-	-	-	-	-
25	4^β	4^β	4	3^β	3	2^β	2	2	1	1	-	-	-	-	-
26	5^β	4^β	4^β	4	3^β	3	2^β	2	2	1	1	-	-	-	-
27	5^β	5^β	4^β	4^β	4	3^β	3	2^β	2	2	1	1	-	-	-
28	6^β	5^β	5^β	4^β	4^β	4	3^β	3	2^β	2	2	1	1	-	-
29	6^β	6^β	5^β	5^β	4^β	4^β	4	3^β	3	2^β	2	2	1	1	-
30	7^β	6^β	6^β	5^β	5^β	4^β	4^β	3^β	3^β	2^β	2^β	2	2	1	1

Table IV. Upper bounds on d for a $[[n, k, d]]_4$ error-correcting-code (n:3..15, k:1..15)

n/k	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
3	2	1	1	-	-	-	-	-	-	-	-	-	-	-	-
4	2	2	1	1	-	-	-	-	-	-	-	-	-	-	-
5	3	2	2	1	1	-	-	-	-	-	-	-	-	-	-
6	3	3	2	2	1	1	-	-	-	-	-	-	-	-	-
7	4	3	3	2	2	1	1	-	-	-	-	-	-	-	-
8	4	4	3	3	2	2	1	1	-	-	-	-	-	-	-
9	5	4	4	3	3	2	2	1	1	-	-	-	-	-	-
10	5	5	4	4	3	3	2	2	1	1	-	-	-	-	-
11	6	5	5	4	4	3	3	2	2	1	1	-	-	-	-
12	6	6	5	5	4	4	3	3	2	2	1	1	-	-	-
13	7	6	6	5	5	4	4	3	3	2	2	1	1	-	-
14	7	7	6	6	5	5	4	4	3	3	2	2	1	1	-
15	8	7	7	6	6	5	5	4	4	3	3	2	2	1	1

Table V. Upper bounds on d for a $[[n,k,d]]_4$ error-correcting-code (n:16..30, k:1..15)

n/k	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
16	8	8	7	7	6	6	5	5	4	4	3	3	2	2	1
17	9	8	8	7	7	6	6	5	5	4	4	3	3	2	2
18	9	9	8	8	7	7	6	6	5	5	4	4	3	2^{β}	2
19	10	9	9	8	8	7	7	6	6	5	5	4	3^{β}	3	2^{β}
20	10	10	9	9	8	8	7	7	6	6	5	4^{β}	4	3^{β}	3
21	11	10	10	9	9	8	8	7	7	6	5^{β}	5	4^{β}	4	3^{β}
22	11	11	10	10	9	9	8	8	7	6^{β}	6	5^{β}	5	4^{β}	4
23	12	11	11	10	10	9	9	8	7^{β}	7	6^{β}	6	5^{β}	5	4^{β}
24	12	12	11	11	10	10	9	8^{β}	8	7^{β}	7	6^{β}	6	5^{β}	5
25	13	12	12	11	11	10	9^{β}	9	8^{β}	8	7^{β}	7	6^{β}	6	5^{β}
26	13	13	12	12	11	10^{β}	10	9^{β}	9	8^{β}	8	7^{β}	7	6^{β}	6
27	14	13	13	12	11^{β}	11	10^{β}	10	9^{β}	9	8^{β}	8	7^{β}	7	6^{β}
28	14	14	13	12^{β}	12	11^{β}	11	10^{β}	10	9^{β}	9	8^{β}	8	7^{β}	7
29	15	14	13^{β}	13	12^{β}	12	11^{β}	11	10^{β}	10	9^{β}	9	8^{β}	8	7^{β}
30	15	14^{β}	14	13^{β}	13	12^{β}	12	11^{β}	11	10^{β}	10	9^{β}	9	8^{β}	8

Table VI. Upper bounds on d for a $[[n,k,d]]_4$ error-correcting-code (n:16..30, k:16..30)

n/k	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30
16	1	-	-	-	-	-	-	-	-	-	-	-	-	-	-
17	1	1	-	-	-	-	-	-	-	-	-	-	-	-	-
18	2	1	1	-	-	-	-	-	-	-	-	-	-	-	-
19	2	2	1	1	-	-	-	-	-	-	-	-	-	-	-
20	2^{β}	2	2	1	1	-	-	-	-	-	-	-	-	-	-
21	3	2^{β}	2	2	1	1	-	-	-	-	-	-	-	-	-
22	3^{β}	3	2^{β}	2	2	1	1	-	-	-	-	-	-	-	-
23	4	3^{β}	3	2^{β}	2	2	1	1	-	-	-	-	-	-	-
24	4^{β}	4	3^{β}	3	2^{β}	2	2	1	1	-	-	-	-	-	-
25	5	4^{β}	4	3^{β}	3	2^{β}	2	2	1	1	-	-	-	-	-
26	5^{β}	5	4^{β}	4	3^{β}	3	2^{β}	2	2	1	1	-	-	-	-
27	6	5^{β}	5	4^{β}	4	3^{β}	3	2^{β}	2	2	1	1	-	-	-
28	6^{β}	6	5^{β}	5	4^{β}	4	3^{β}	3	2^{β}	2	2	1	1	-	-
29	7	6^{β}	6	5^{β}	5	4^{β}	4	3^{β}	3	2^{β}	2	2	1	1	-
30	7^{β}	7	6^{β}	6	5^{β}	5	4^{β}	4	3^{β}	3	2^{β}	2	2	1	1

Comments on the tables: The entries in the table are to be interpreted as follows: From Table I, we see that no $[[11,7,3]]_3$ quantum stabilizer code exists. However, a $[[11,6,3]]_3$ code may exist. This corroborates our initial conjecture that using linear programming methods, we can derive upper bounds tighter than those obtained using equation 3.1. Codes that meet the upper bounds obtained via equation 3.1 are called *Maximum Distance Separable* (MDS) codes. They are called MDS because the Singleton bound gives the largest value of d for the given parameter values n and k, and these codes have a value of distance equal to the highest achievable minimum distance.

We also investigate the possible existence of pure codes by setting A_1 through A_{d-1} equal to 0. In all cases up to $n = 25$, the bounds for pure codes were equal to the bounds for impure codes.

To further substantiate the results outlined above, we scanned literature extensively and we could not find a single example that proves the non-existence of a code meeting the upper bounds derived by us. This may lead us to believe that the bounds derived by us are pretty tight. However, to support this conjecture, we need to actually find codes that meet or nearly meet the bounds derived. Also, for none of the values of n and k, did we get an upper bound that was greater than the Singleton bound.

B. Table of Upper Bounds for Linear Codes

Mathematica does not include a package for integer linear programming, hence, most of the solutions to the constraints for additive codes that were obtained were real numbers. But weight distributions are integer valued and this constraint does not get implicitly incorporated while modelling with Mathematica. On the other hand, integer linear programming is computationally very expensive and, hence, it would be possible to model a set of constraints only for very small values of n.

We modelled the set of linear equations for linear codes outlined in the previous chapter using the integer linear optimization package, *ilp* [15], developed by Pathria, which is part of the Maple Share library. We have computed the table of upper bounds for values of n upto 13. We now present the table of upper bounds on d for 3-ary linear quantum stabilizer codes.

Table VII. Upper bounds on d for $[[n,k,d]]_3$ linear error-correcting-codes

n/k	1	2	3	4	5	6	7	8	9	10	11	12	13
3	2	-	1	-	-	-	-	-	-	-	-	-	-
4	-	2	-	1	-	-	-	-	-	-	-	-	-
5	3	-	2	-	1	-	-	-	-	-	-	-	-
6	-	3	-	2	-	1	-	-	-	-	-	-	-
7	4	-	3	-	2	-	1	-	-	-	-	-	-
8	-	4	-	3	-	2	-	1	-	-	-	-	-
9	5	-	4	-	3	-	2	-	1	-	-	-	-
10	-	5	-	4	-	3	-	2	-	1	-	-	-
11	6	-	5	-	4	-	2	-	2	-	1	-	-
12	-	6	-	5	-	3	-	2	-	2	-	1	-
13	7	-	6	-	4	-	3	-	2	-	2	-	1

Lemma 29. *If $n-k$ is odd, then there does not exist a linear quantum stabilizer code.*

Proof. Let D be the associated $\mathbf{F}_{q^2}$-linear classical subspace over $\mathbf{F}_{q^2}$ with q^{n-k} vectors. From Constraint 1 and 2 in Theorem 28, we have

$$\sum_{i=1}^{n} A_i = q^{n-k} - 1 = q^z - 1, \tag{4.1}$$

where $z = n-k$ is odd. Also, from Constraint 5 in Theorem 28 and equation 4.1, we have

$$(q^2 - 1) \,|\, q^z - 1. \tag{4.2}$$

The above equation holds only when z is even and, hence, from definition 14 it follows that a linear $[[n,k,d]]_q$ quantum stabilizer code cannot exist when $n-k$ is odd. $\square$

Comments on the table: From Table VII, we see that the bounds derived for linear codes coincide with those derived for additive codes for values of n up to 13 and for $q = 3$ when $n - k$ is odd. This means that we can potentially have linear quantum stabilizer codes that have the same error correcting capabilities as additive quantum stabilizer codes whenever $n - k$ is even. This is useful because it is conjectured that building encoding circuits for linear codes is simpler than building encoding circuits for additive codes because of the rich structure associated with linear codes. For cases when $n - k$ is odd, we can only have additive quantum stabilizer codes.

CHAPTER V

CODE CONSTRUCTIONS

The bounds established previously only prove the non-existence of codes having certain parameters. To actually prove the existence of codes having certain parameter values, we need to explore the literature to find families of codes and derive code construction theorems that help construct new codes from existing ones. Ketkar [18] has constructed a table of lower bounds on the minimum distance d for additive codes using the techniques stated above. This table of lower bounds will help evaluate the tightness of the upper bounds derived previously. In this chapter, we present a few code construction techniques that will help in the construction of new additive nonbinary quantum codes from existing ones.

A. General Constructions

The technique for constructing new additive quantum codes from existing ones will involve the following steps:

1. Find the associated classical code over $\mathbf{F}_{q^2}$ of the existing quantum code.
2. Derive a new classical code from the one established in step 1, satisfying all the required properties.
3. Prove the existence of the quantum code corresponding to the classical code derived in step 2 by means of Theorem 11.

Lemma 30. *An additive* $[[n,k,1]]_q$ *code exists for all* $0 \leq k \leq n$, $n \geq 3$.

Theorem 31. *Given two additive codes* $[[n_1,k_1,d_1]]_q$ *and* $[[n_2,k_2,d_2]]_q$ *with* $k_2 \leq n_1$, *we can construct an additive* $[[n_1+n_2-k_2,k_1,d]]_q$ *code, where* $d \geq \min\{d_1, d_1+d_2-k_2\}$.

Proof. Let the associated codes be $D_1, D_1^{\perp_a}$ with parameters $(n_1, q^{n_1-k_1}), (n1, q^{n_1+k_1})$ and $D_2, D_2^{\perp_a}$ with parameters $(n_2, q^{n_2-k_2}), (n_2, q^{n_2+k_2})$. Let ρ be the composition of the natural map from $D_2^{\perp_a}$ to $D_2^{\perp_a}/D_2$ with any inner-product-preserving homomorphism from the additive group $D_2^{\perp_a}/D_2$ to $\mathbf{F}_{q^2}^{k_2}$. Construct a new code D of the form

$$D = \{uv : v \in D_2^{\perp_a}, u\rho(v) \in D_1\} \tag{5.1}$$

and $D^{\perp_a}$ of the form

$$D^{\perp_a} = \{uv : v \in D_2^{\perp_a}, u\rho(v) \in D_1^{\perp_a}\}. \tag{5.2}$$

We now have to show that

1. D is additive.

2. $D \subseteq D^{\perp_a}$.

3. The corresponding quantum code obtained from D has minimum weight d given by $d \geq \min\{d_1, d_1 + d_2 - k_2\}$.

4. Number of code words in $D = q^{n_1+n_2-k_2-k_1}$.

We note that D is additive because ρ is a homomorphism of additive groups. Let $(u_1|v_1)$, $(u_2|v_2) \in D$ meaning that $(u_1|\rho(v_1))$, $(u_2|\rho(v_2)) \in D_1$. Now,

$$\begin{aligned} \langle (u_1|v_1), (u_2|v_2) \rangle_a &= \langle u_1, u_2 \rangle_a + \langle v_1, v_2 \rangle_a \\ &= \langle u_1, u_2 \rangle_a + \langle \rho(v_1), \rho(v_2) \rangle_a \\ &= \langle (u_1|\rho(v_1)), (u_2|\rho(v_2)) \rangle_a \\ &= 0, \end{aligned}$$

where the last equality holds because $(u_1|\rho(v_1))$ and $(u_2|\rho(v_2))$ are elements of the self-orthogonal code D_1; hence $D \subseteq D^{\perp_a}$.

To prove the distance property, we find the minimum weight of a vector in $D^{\perp_a} \backslash D$. Any vector in $D^{\perp_a} \backslash D$ is of the form

$$D^{\perp_a} \backslash D = \{uv \,:\, v \in D_2^{\perp_a} \backslash D_2, u\rho(v) \in D_1^{\perp_a} \backslash D_1\}$$

Note that if $\rho(v) \neq 0$, v contributes at least d_2 to the weight of uv, but u need have weight only $d_1 - k_2$. If $\rho(v) = 0$, and $uv \neq 0$, $wt(u) \geq d_1$.

To show that $|D| = q^{n_1+n_2-k_2-k_1}$, we proceed as follows: The mapping ρ maps $q^{n_2+k_2}$ elements to q^{2k_2} elements. This means that each one of the q^{2k_2} sets of $q^{n_2-k_2}$ elements each is mapped to the same element in $\mathbf{F}_{q^2}^{k_2}$. Let us denote the ith set of elements as Set_i. Let A_i be the number of code words in D_1 that have suffix $\rho(v_j)$, where v_j is any element in Set_i. Then the number of code words in D is given by

$$|D| = q^{n_2-k_2} \times (A_1 + A_2 + \cdots + A_{2k_2}). \tag{5.3}$$

Since ρ generates all possible vectors in $\mathbf{F}_{q^2}^{k_2}$, $(A_1 + A_2 + \cdots + A_{2k_2}) = q^{n_1-k_1}$. Hence, we have shown that the number of vectors in D is $q^{n_1+n_2-k_2-k_1}$. Note that if the initial codes are pure, then the code so obtained as a result of the direct sum construction is also pure. □

B. Concatenated Quantum Codes

In this section, we give a brief exposition on the construction of quantum codes using a powerful technique that is analogous to the concatenated codes in the classical setting. The discussion presented here is basically a generalization of the theory of binary concatenated codes in [4]. Concatenated codes are a special class of codes that use a combination of an inner encoder and an outer encoder to encode the data. We illustrate it with the help of an example below. Suppose the inner encoder uses a code that is a $[[n,1,d]]_q$ quantum code and the outer code is an $[[N,K,D]]_q$ quantum code. The encoding is done as follows: First, the K information symbols are encoded into N qubits by the outer encoder. Then, each qubit in the output produced by the outer encoder is encoded into further n qubits using the inner encoder. Thus, the final code word that is transmitted is of length nN and has minimum distance dD. Decoding is done as follows: The error syndrome for each level is calculated by performing parallel computations on different blocks that make up that level. The information about the error syndromes from each level is then combined to perform the error recovery operations. The ability to combine information about the error syndromes from different levels is a crucial requirement in order to be able to achieve the full minimum distance of the code. For example, using the above procedure, we can construct a $[[25,1,9]]_3$ code from two $[[5,1,3]]_3$ codes and also a $[[25,1,9]]_4$ code from two $[[5,1,3]]_4$ codes.

CHAPTER VI

CONCLUSION AND FUTURE WORK

A. Future Work

In this work, we have derived upper bounds for nonbinary additive quantum stabilizer codes where $q = 3, 4$ and also for 3-ary linear quantum stabilizer codes. A logical extension to this work would be to derive upper bounds for higher values of q and n. While we have not come across a single example in literature that proves the non-existence of a code meeting the upper bounds derived by us, we still cannot prove that our bounds are indeed tight. In [5], the authors have tightened the bounds by applying the theory of shadow codes. Hence, exploring the generalized theory of shadow codes as explained in [19, 20] would be good starting point that would help us find more constraints and, hence, help us to tighten the upper bounds. Once we have a table of upper bounds that are pretty tight, the next step would be to find actual codes meeting the upper bounds derived in the table. One strategy to find codes is to exhaustively search for codes satisfying a particular set of properties. However, this method is exponential in time and unless it is highly optimized, we cannot use the exhaustive search strategy to search for codes having large values of n. The solutions in the form of the weight distribution of the codes returned by the linear programming solver can be used to optimize the exhaustive search program and, hence, help us to search for codes for larger values of n. What we ultimately need is a table of the best possible codes for all possible parameter values up to a certain limit along with the encoding circuits for the same. This table would then help us to compare the nonbinary quantum codes and binary quantum codes on the basis of their error correcting capabilities.

B. Conclusion

The main focus of this research was to generalize the results in [5] for the binary case to the nonbinary case. We have constructed a table of upper bounds on the minimum distance of additive nonbinary stabilizer codes using linear programming methods. The bounds derived by us are tighter than those obtained using the Singleton bound equation. We then attempted to find out whether we could have linear codes meeting the bounds derived for additive codes. Our linear programming results show that for values of n up to 13 and for $q = 3$, the bounds on the distance for linear codes coincide with the bounds for additive codes for the case when $n - k$ is even. We also discussed a couple of techniques for deriving new additive quantum codes from existing additive quantum codes. All the results put together have helped us gain a deeper understanding of the theory of nonbinary quantum stabilizer codes. We now have the groundwork ready for the work ahead that will help us answer the question of whether we can have nonbinary quantum stabilizer codes that are better than binary stabilizer codes in terms of the error correction capabilities.

REFERENCES

[1] P.W. Shor, "Algorithms for quantum computation. Discrete logarithms and factoring," in *35th Annual Symposium on Foundations of Computer Science*, Santa Fe, New Mexico, 1994, pp. 124–134.

[2] M.A. Nielsen and I.L. Chuang, *Quantum Computation and Quantum Information*, Cambridge, UK, Cambridge University Press, 2000.

[3] P.W. Shor, "Scheme for reducing decoherence in quantum memory," *Phys. Rev. A*, vol. 52, pp. 2493–2496, 1995.

[4] D. Gottesman, "Stabilizer codes and quantum error correction," Ph.D. dissertation, California Institute of Technology, Pasadena, California, 2001.

[5] A.R. Calderbank, E.M. Rains, P.W. Shor, and N.J.A. Sloane, "Quantum error correction via codes over GF(4)," *IEEE Trans. Inform. Theory*, vol. 44, pp. 1369–1387, 1998.

[6] E.M. Rains, "Nonbinary quantum codes," *IEEE Trans. Inform. Theory*, vol. 45, pp. 1827–1832, 1999.

[7] F.J. MacWilliams and N.J.A. Sloane, *The Theory of Error-Correcting Codes*, Amsterdam, The Netherlands: North-Holland, 1977.

[8] A. Ashikhmin and E. Knill, "Nonbinary quantum stabilizer codes," *IEEE Trans. Inform. Theory*, vol. 47, no. 7, pp. 3065–3072, 2001.

[9] R. Matsumoto and T. Uyematsu, "Constructing quantum error correcting codes for p^m-state systems from classical error correcting codes," *IEICE Trans. Fundamentals*, vol. E83-A, no. 10, pp. 1878–1883, 2000.

[10] J. Bierbrauer and Y. Edel, "Quantum twisted codes," *J. Comb. Designs*, vol. 8, pp. 174–188, 2000.

[11] J.E. Kim and J. Walker, "Nonbinary quantum error-correcting codes from algebraic curves," *IEEE Trans. Inform. Theory*, 2004.

[12] E. Knill and R. Laflamme, "A theory of quantum error–correcting codes," *Phys. Rev. A*, vol. 55, no. 2, pp. 900–911, 1997.

[13] E. Knill, "Non-binary unitary error bases and quantum codes," Los Alamos National Laboratory Report LAUR-96-2717, 1996.

[14] Z. Wan, *Quaternary Codes*, China, World Scientific Pub. Co. Inc., 1997.

[15] A. Pathria, "Integer linear programming package," June 2001, Available at http://www.mapleapps.com/maplelinks/share/ilp.shtml.

[16] W.C. Huffman and V. Pless, *Fundamentals of Error-Correcting Codes*, Cambridge, UK, Cambridge University Press, 2003.

[17] S. Wolfram, *The Mathematica Book*, 5th Edition, Champaign, Illinois, Wolfram Media Inc., 2003.

[18] A.U. Ketkar, "Code constructions and encoding circuits for nonbinary stabilizer codes," M.S. thesis, Texas A&M University, 2004.

[19] E.M. Rains, "Quantum weight enumerators," LANL e-print quant-ph/9612015, 1996.

[20] E.M. Rains, "Polynomial invariants of quantum codes," LANL e-print quant-ph/9704042, 1997.

APPENDIX A

MATHEMATICA CODE FOR LINEAR PROGRAMMING

This appendix contains the source code for the implementation of the linear programming constraints for additive codes using MATHEMATICA. The program takes as input the parameters n, k, d and q of the code and dynamically generates the constraints. These constraints are then modelled using the *LinearProgramming* function which takes as arguments the objective function, c, which is a list, of dimension 1 × numVar, the matrix, m of dimension numCon × numVar that contains the coefficients of the variables and another numCon × 2 matrix, b that specifies the type of constraints ($\leq$, $=$, $\geq$) and the right hand side values of the constraints. Here, numCon is used to represent the number of constraints and numVar is used to represent the number of variables. We now present the source code along with adequate documentation.

```
**************************************************************

(* This method returns the coefficient of the Krawtchouk polynomial *)
P[j_, x_, n_, q_,  k_]:=
    ((Sum[(( -1 ) ^ i) * ((q * q - 1) ^ (j - i)) * Binomial[x, i] * Binomial[n - x, j - i],
                    {i, 0, j}]) / (q ^ (n - k)))


**************************************************************
```

```
******************************************************************
(* This method takes as input the parameters of the code and then attempts
to find a solution to the set of linear programming constraints *)
Calc[n_, k_, d_, q_]:=(
    numCon = 2 * n + 4; (* Initialize the number of constraints *)
    numVar = 2 * n + 2; (* Initialize the number of variables *)
    curCon = 1; (* Number of the current constraint *)

    c = Table[obj[i], {i, numVar}];
    (* Initialize the objective function to be 0 since we are only interested
    in finding a solution *)
    For[i = 1, i ≤ numVar, i++, c[[i]]=0;];
    (* The table b is numCon by 2 and b[i,2] =
    0 if equality constraint =, 1 if ≥ constraint and -1 otherwise *)

    b = Table[con[i, j], {i, numCon}, {j, 2}];
    (* m is numCon by numVar and contains the coeffs. of the variables *)

    m = Table[a[e, f], {e, numCon}, {f, numVar}];

    (* Initialize the m matrix to contain all 0's *)
    For[i = 1, i ≤ numCon, i++,
        For[j = 1, j ≤ numVar, j++,
            m[[i, j]] = 0;]
    ];
```

```
(* Constraint number 1 *)
m[[curCon, 1]] = 1;
b[[curCon, 1]] = 1;
b[[curCon, 2]] = 0;
curCon = curCon + 1; (* Increment the curCon count *)

(* Constraint number 2 *)
b[[curCon, 1]] = q ^ (n - k);
b[[curCon, 2]] = 0;
For[j = 1, j ≤ n + 1, j++, m[[curCon, j]] = 1;];
curCon = curCon + 1; (* Increment the curCon count *)

(* Next n+1 constraints involving the Krawtchouk polynomial *)
For[i = 0, i ≤ n, i++, b[[curCon, 1]] = 0; b[[curCon, 2]] = 0;
    m[[curCon, i + 1 + n + 1]] = 1;
    For[j = 0, j ≤ n, j++, m[[curCon, j + 1]] = -1 * P[i, j, n, q, k];
    ];
    curCon = curCon + 1; (* Increment the curCon count *)
];

(* Next n+1 constraints involving the distance relationships *)
For[i = 1, i ≤ d, i++, m[[curCon, i]] = 1;
    m[[curCon, i + n + 1]] = -1;
    b[[curCon, 1]] = 0;
    b[[curCon, 2]] = 0;
    curCon = curCon + 1;  (* Increment the curCon count *)
    ];

    For[i = d + 1, i ≤ n + 1, i++, m[[curCon, i]] = 1;
        m[[curCon, i + n + 1]] = -1;
        b[[curCon, 1]] = 0;
        b[[curCon, 2]] = -1;
        curCon = curCon + 1; (* Increment the curCon count *)
    ];

    LinearProgramming[c,m,b]
)
*********************************************
```

APPENDIX B

MAPLE CODE FOR INTEGER LINEAR PROGRAMMING

This appendix contains the source code for the implementation of the linear programming constraints for linear codes using MAPLE. The program takes as input the parameters n, k, d and q of the code and dynamically generates the constraints. These constraints are then modelled using the *ilp* function which takes as arguments the objective function, a list of constraints in symbolic form and a third argument which specifies whether the solutions are NONNEGATIVE or not. This function attempts to find a set of integeral solutions satisfying the set of constraints. We include the source code only for the set of constraints corresponding to the case when both the code D and its dual $D^{\perp_a}$ are even-like. We now present the source code along with adequate documentation.

```
***************************************************************
# Function that returns the coefficient of the Krawtchouk polynomial
Kraw := proc(j, x, n, q)
    local coeff, k;
    coeff:= sum((-1) ˆ k * (q * q - 1) ˆ (j - k) * binomial(x, k)
                * binomial(n - x, j - k), k = 0..j);
    return coeff;
end proc;
***************************************************************
```

```
**************************************************************

# This method takes as input the parameters of the
# code and then attempts to find a set of integral solutions
# to the set of linear programming constraints

CalcBounds := proc(n, k, d, q, p)
    numCon := n + 4; # Variable that denotes number of constraints
    numVar := n + 1; # Variable that denotes number of variables
    curCon := 1; # Current constraint number

    # Generate a matrix numVar ×1 that contains the variables.
    # This part of the code will change depending on the value of $n$
    # For example, if the value of n is 13 then the list will hold values
    # up to [a13]
    var := Matrix(numVar, 1, [[a0], [a1], [a2], [a3], [a4], [a5], [a6]]);

    # Generate the matrix that contains the coeffs. of all the variables
    # in the constraints. This matrix contains all 0's initially
    coeff := Matrix(numCon, numVar);

    # Constraint corresponding to $A_0 = 1$
    coeff[curCon, 1] := 1;
    curCon := curCon + 1;

    # Constraint corresponding to $\sum_{i=0}^{n} A_i = q^{n-k}$
    for i from 1 to n + 1 do
        coeff[curCon, i] := 1;
    end do;
    curCon := curCon + 1;

    # Next set of constraints involving the distance relationships
    for i from 0 to n do
        coeff[curCon, i + 1] := q ^ (n - k);
        for j from 0 to n do
            coeff[curCon, j + 1] := coeff[curCon, j + 1] - Kraw(i, j, n, q);
        end do;
        curCon := curCon + 1;
    end do;

    # Constraint corresponding to $A_n \geq q^2 - 1$
    coeff[curCon, n + 1] := 1;
    curCon := curCon + 1;

    # Implicitly include the divisibility criteria here
    coeff := (q * q - 1) * coeff;
    for i from 1 to numCon do
        coeff[i, 1] := coeff[i, 1] / (q * q - 1);
    end do;

    D := coeff.var;
```

```
#Declare a list here that will hold all the constraints
L := list(numCon);
curCon := 1;

#Generate the first constraint
L[curCon] := (D[curCon, 1] = 1 );
curCon := curCon + 1;

#Generate the next constraint
L[curCon] := (D[curCon, 1] = q ^ (n - k));
curCon := curCon + 1;
#Generate the next d constraints
for i from 0 to d - 1 do
    L[curCon] := (D[curCon, 1] = 0);
    curCon := curCon + 1;
end do;

# Generate constraints d to n
for i from d to n do
    L[curCon] := (D[curCon, 1] ≤  0);
    curCon := curCon + 1;
end do;

# Generate the next constraint
L[curCon] := (D[curCon, 1] ≥ (q * q - 1));
curCon := curCon + 1;
```

```
    # Now convert the list of constraints to a set
    # because the ilp function requires that
    constraints := convert(L, set);

    # Now call the ilp solver
    s := ilp(0, constraints, NONNEGATIVE);
end proc;
*****************************************************************
```

VITA

Name: Santosh Kumar

Education: B.E. Computer Science, Mumbai University, Aug' 02

Address: 301 Harvey R. Bright Bldg., Computer Science Dept., Texas A&M Univ.,
College Station, TX 77843-3112