8-2001

Z_4-linear codes

Terri Annette Bedford

To the Graduate Council:

I am submitting herewith a thesis written by Terri Annette Bedford entitled "Z_4-linear codes." I have examined the final electronic copy of this thesis for form and content and recommend that it be accepted in partial fulfillment of the requirements for the degree of Master of Science, with a major in Mathematics.

Gretchen Matthews, Major Professor

We have read this thesis and recommend its acceptance:

David F. Anderson, Pavlos Tzermias

Accepted for the Council:
Carolyn R. Hodges

Vice Provost and Dean of the Graduate School

(Original signatures are on file with official student records.)

To the Graduate Council:

I am submitting herewith a thesis written by Terri Annette Bedford entitled "Z_4-Linear Codes." I have examined the final copy of this thesis for form and content and recommend that it be accepted in partial fulfillment of the requirements for the degree of Masters of Science, with a major in Mathematics.

Gretchen L Matthews

Gretchen Matthews, Major Professor

We have read this thesis
and recommend its acceptance:

David F Anderson

Pavlos Tzermias

Accepted for the Council:

Interim Vice Provost and
Dean of the Graduate School

$\mathbb{Z}_4$-Linear Codes

A Thesis

Presented for the

Master of Science

Degree

The University of Tennessee, Knoxville

Terri Annette Bedford

August 2001

Acknowledgements

First and foremost, thank you to my heavenly Father for giving me the strength to finish this. I would like to thank my family for their prayers and encouragement. They believed in me when I didn't believe in myself. Thank you to my buddies in the UT math department, especially to Joel Mejeur for his help with all my computer problems and to Marti McClard for her proofing reading and suggestions. Thank you to Dr. Stubbs of Huntingdon College for his encouragement to attend graduate school and sparking my interest in coding theory. I would also like to thank my thesis committee, Dr. Matthews, Dr. Anderson, and Dr. Tzermias, for their time spent proof reading and listening to my defense. I especially want to thank my major professor, Dr. Matthews, for her time spent reading and rereading this thesis and for her knowledge of coding theory and algebra that kept me on the right track. She motivated me to keep working when I had lost desire. I could not have completely this without her.

Abstract

Coding theory is a branch of mathematics that began in the 1940's to correct errors caused by noise in communication channels. It is know that certain nonlinear codes satisfy the MacWilliams Identity. Much research has been done to explain this relationship. In the early 1990's, five coding theorists discovered that nonlinear codes have linear properties if viewed under the alphabet $\mathbb{Z}_4$ rather than the usual alphabet $\mathbb{F}_2$. In 1994, these coding theorists published their results in a joint paper in *IEEE Transactions on Information Theory*. In this study, linear codes and nonlinear codes are introduced and characterized as $\mathbb{Z}_4$-linear codes to understand the importance of this discovery.

Contents

Chapter 1 Introduction to Coding Theory

With the use of high speed computers, digital phones, and credit cards, the transfer of information from one source to another is going on continuously. This information is sent from one source to another through a communication channel. Examples of a communication channel are telephone lines, computer cables, and even the atmosphere. Suppose we want to send a message through a communication channel. Because of noise in the channel, the message received may not be the same as the message that was sent. Noise is caused by many different things. Some of these include lightning, a scratch on a compact disc, and even poor speech or hearing. Coding theory, which began in the late 1940's, aims to correct the errors caused by noise so that messages are communicated reliably.

Definition 1.1: An *alphabet* A is a finite set of symbols.

Definition 1.2: A *code* C over an alphabet A is a subset of $A^n := A \times ... \times A$ (n copies). The *length* of the code C is n. The elements of C are called *codewords*.

Often, we consider codes over $\mathbb{F}_2 = \{0,1\}$, called binary codes, or codes over extension fields of $\mathbb{F}_2$. More generally, we will use the alphabet $A = \mathbb{F}_q$, where $\mathbb{F}_q$ denotes the finite field with q elements and q is a power of some prime number.

Figure 1 below provides a description of how a message is sent from one source to another. Suppose we have a message in an alphabet A that we want to send. Let C be a code of length n over alphabet A. The message to be sent is broken into message words of equal length k, for some $k \in \mathbb{N}$ such that $k < n$. With each message word $\mathbf{w} = (w_1, ..., w_k)$, $w_i \in$ A, we associate a codeword $\mathbf{x} = (x_1, ..., x_n) \in$ C. The codeword $\mathbf{x}$ is sent through the channel. Let $\mathbf{y} = (y_1, ..., y_n)$ denote the received word. Note that we will assume the received word has the same length as the codeword. If no errors have occured, then $\mathbf{y} = \mathbf{x}$. However, it is possible that $\mathbf{y} \neq \mathbf{x}$. Regardless, it is the now the job of the decoder to determine from the received word $\mathbf{y}$ the message word $\mathbf{w}$ that was originally sent. We assume that the channel is not too noisy. In this case, it is the most likely that only a few errors have occured. Thus, the decoder selects a codeword $\mathbf{x}' \in$ C that most closely resembles $\mathbf{y}$. If there is not a unique codeword closest to $\mathbf{y}$, then either a choice is

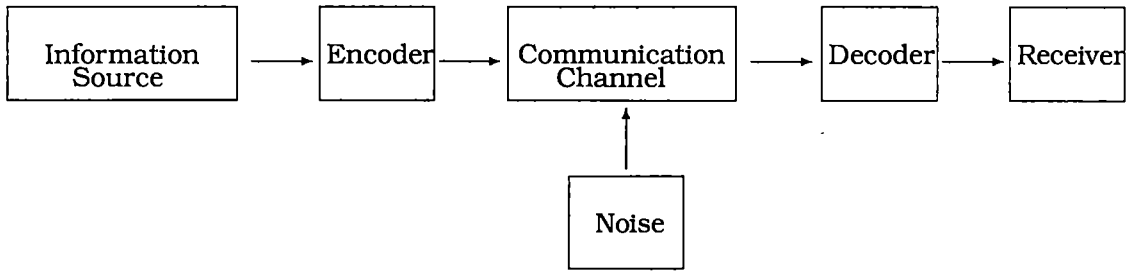

Figure 1. The basic setup of coding theory.

made between codewords or we ask for a retransmission. This type of decoding is called maximum likelihood decoding. To make this more precise, we need a notion of distance.

Definition 1.3: The *Hamming distance* between two words $\mathbf{x} = (x_1, ..., x_n)$ and $\mathbf{y} = (y_1, ..., y_n)$ in A^n is defined to be d(**x**,**y**) = $|\{i : x_i \neq y_i\}|$.

Using this terminology, the received word **y** is decoded as **x**, where d(**x**,**y**) = min{d(**x**′,**y**) | **x**′ ∈ C}. A choice is made if more than one **x**′ satisfies this condition. From this, the decoder determines which message word $\mathbf{w}' = (w_1', ..., w_k') \in A^k$ was orginally sent.

Definition 1.4: The *Hamming weight* of a word $\mathbf{x} = (x_1, .., .x_n) \in A^n$ is the number of nonzero coordinates, x_i, denoted wt(**x**). That is, wt(**x**) = $|\{i : x_i \neq 0\}|$.

Example 1.5: Let **x** = (0,1,1,0) ∈ $\mathbb{F}_2^4$. Then wt(**x**) = $|\{i : x_i \neq 0\}| = |\{2,3\}| = 2$.

Note that if **z** = **x**−**y** for words **x**,**y** ∈ A^n, then wt(**z**) = wt(**x**−**y**) = d(**x**,**y**).

Definiton 1.6: Let **v** be the codeword that was sent and **y** be the received word. Then **e** = **v**−**y** is called the *error vector.*

Note that if **v** is sent and **y** is received, then the number of errors that occurred is d(**v**,**y**). That is, the number of errors is wt(**e**), where **e** = **v**−**y** denotes the error vector.

Given an alphabet A, one would like to construct codes over A which correct many errors. If a received word **y** is closest to a unique codeword **x**, then by maximum likelihood decoding we assume that **x** was the codeword sent through the channel. This means that d(**x**,**y**) ≠ d(**x**′,**y**) for all **x**′ ∈ C\{**x**}.

Definition 1.7: The *minimum distance* of a code C is

$$d := \min\{d(\mathbf{x}, \mathbf{y}) \mid \mathbf{x}, \mathbf{y} \in C \text{ and } \mathbf{x} \neq \mathbf{y}\}.$$

The minimum distance of a code tells how many errors the code can detect and correct.

Theorem 1.8: A code C with minimum distance d will detect all errors in $d-1$ or fewer positions. Moreover, there is at least one error in d positions that will not be detected by C.

Proof: Suppose C is a code with minimum distance d. Let $\mathbf{y}$ be a received word such that $\mathbf{y} = \mathbf{v}-\mathbf{e}$, where $\mathbf{v} \in C$ and $\mathbf{e}$ is a nonzero error vector of weight $wt(\mathbf{e}) \leq d-1$. Then $d(\mathbf{v},\mathbf{y}) = wt(\mathbf{v}-(\mathbf{v}-\mathbf{e})) = wt(\mathbf{e}) < d$. Since C has minimum distance d and the distance between $\mathbf{y}$ and the codeword $\mathbf{v}$ is less than d, $\mathbf{y}$ is not an element of C. Therefore C detects any error in $d-1$ positions.

By the definition of distance, there are codewords $\mathbf{v}$ and $\mathbf{w}$ in C with $d(\mathbf{v},\mathbf{w}) = d$. Suppose $\mathbf{v}$ is the codeword sent and $\mathbf{w}$ is received. Then the error vector $\mathbf{e} = \mathbf{v}-\mathbf{w}$ is a nonzero vector of weight $wt(\mathbf{e}) = d$. Since $\mathbf{w}$ is a codeword, C will not detect this error of weight d. □

Notice that a code C may detect some error vectors of weight d or more, but C does not detect all error vectors $\mathbf{e}$ such that $wt(\mathbf{e}) \geq d$.

Example 1.9: Consider the International Standardized Book Number (ISBN) Code. The ISBN is used to catalog books. Each book is assigned a ten digit number, $a_1, ..., a_{10}$. The first nine digits, $a_1, ..., a_9 \in \mathbb{Z}_{10}$, contain information about the book. The last digit, $a_{10} \in \mathbb{Z}_{11}$, is called a check digit. It is chosen by computing $a'_{10} := (a_1 + 2a_2 + ... + 9a_9)$. If $a'_{10} \equiv i \pmod{11}$ for some $0 \leq i \leq 9$, we set $a_{10} = i$. If $a'_{10} \equiv 10 \pmod{11}$, we set a_{10} to be the symbol "X". The check digit is used to detect a mistake, or error, in the number entered.

One can verify that the ISBN code has minimum distance 2. Therefore by Theorem 1.8, ISBN code detects all single-digit errors. Also, it detects if two positions are transposed. Suppose $a_1, a_2, a_3, ..., a_i, a_{i+1}, ..., a_{10}$ is the correct ISBN and

$a_1, a_2, ..., a_{i+1}, a_i, ..., a_{10}$ is the number entered. Since $a_{10} = (a_1 + 2a_2 + ... + ia_i + (i + 1)a_{i+1} + ... + 9a_9) \pmod{11} \neq (a_1 + 2a_2 + ... + (i+1)a_i + ia_{i+1} + ... + 9a_9) \pmod{11}$, the code detects the transposition error. See [16].

The ISBN code cannot correct any errors, but it is efficient since only one non-information symbol is used for every nine-symbol piece of data. Since the correct ISBN can be reentered easily, detecting single errors and transposition errors is satisfactory.

In some applications, asking the source to resend the message could be very expensive or not even possible. For example, the Voyager II spacecraft sent pictures of Jupiter and Saturn in the early 1980's. Each picture was broken up into pieces and the pieces were sent one by one through the atmosphere. Asking the spacecraft to resend a piece of the picture that is not readable would mean that the spacecraft was resending the piece continously since the atmosphere is a very noisy channel. This would take time, and the Voyager II had a limited life span. In this case we need a code that not only detects errors but can also correct errors.

Theorem 1.10: A code C with minimum distance $d = 2t + 1$ can correct all errors in t or fewer positions. Moreover, there is at least one error in $t+1$ positions which the code C cannot correct.

Proof: Let C be a code in the alphabet A of minimum distance $d = 2t + 1$. Given $\mathbf{c} \in$ C, let $B_t(\mathbf{c}) = \{\mathbf{x} \in A^n \mid d(\mathbf{x},\mathbf{c}) \leq t\}$ be the sphere of radius t about $\mathbf{c}$. Since $d = 2t + 1$, the distance between any two distinct codewords is at least $2t + 1$. Therefore, $B_t(\mathbf{c}) \bigcap B_t(\mathbf{c}') = \emptyset$ for all $\mathbf{c},\mathbf{c}' \in$ C such that $\mathbf{c} \neq \mathbf{c}'$. Suppose a codeword $\mathbf{u}$ is transmitted and t or fewer errors occur. Let $\mathbf{y}$ be the received word. Then $\mathbf{y} \in B_t(\mathbf{u})$ since $d(\mathbf{y},\mathbf{u}) \leq t$. Note that $\mathbf{y} \notin B_t(\mathbf{c})$ for any $\mathbf{c} \in$ C\{$\mathbf{u}$} since $B_t(\mathbf{c}) \bigcap B_t(\mathbf{c}') = \emptyset$. Then $\mathbf{y}$ is decoded to the codeword $\mathbf{u}$, and the t errors are corrected.

Since $d = 2t + 1$, there exists $\mathbf{c},\mathbf{c}' \in$ C and $\mathbf{c} \neq \mathbf{c}'$ such that $d(\mathbf{c},\mathbf{c}') = 2t + 1$. Therefore, $B_{t+1}(\mathbf{c}) \bigcap B_{t+1}(\mathbf{c}') \neq \emptyset$. In particular, there exists $\mathbf{y} \in B_{t+1}(\mathbf{c}) \bigcap B_{t+1}(\mathbf{c}')$. If $\mathbf{y}$ is a received word such that $\mathbf{e} = \mathbf{y}-\mathbf{c}$ is the error vector of weight $t + 1$, then a choice must be made in decoding $\mathbf{y}$ since $\mathbf{y}$ is not contained in a unique sphere of radius $t + 1$. Therefore the $t + 1$ errors are not always corrected. □

Example 1.11: Let C = {(0,0,0), (1,1,1)}. Since min{d(**x**,**y**) | **x**,**y** $\in$ C and **x** $\neq$ **y**} = 3, C has minimum distance $d = 3$. Thus, C will detect all error vectors of weight 1 or 2. Suppose **x** = (0,0,0) is the word sent and **y** = (1,1,1) is the received word. Since (1,1,1) $\in$ C, C cannot detect the error of weight 3. Since $\frac{d-1}{2} = 1$, C will correct all error vectors of weight 1. Suppose **y** = (1,0,1) is the received word. Since **y** $\in$ $B_t(1,1,1) = \{\mathbf{x} \in \mathbb{F}_2^3 \mid d(\mathbf{x},(1,1,1)) \leq 1\}$, **y** is decoded to the codeword (1,1,1).

Corollary 1.12: Let C be a code of minimum distance d. If d is even, the code can detect all errors in $\frac{d}{2}$ positions and correct all errors in $\lfloor \frac{d-1}{2} \rfloor$ positions.

Proof: Suppose the minimum distance d of a code C is even. By definition of minimum distance, spheres of radius $\frac{d-1}{2}$ about the codewords of C are disjoint. Therefore, C can correct error vectors **e** such that wt(**e**) = $\frac{d-1}{2}$. If wt(**e**) = $\frac{d}{2}$, the received word **y** may be equidistant from two codewords, that is **y** $\in$ $B_t(\mathbf{u}) \bigcap B_t(\mathbf{v})$ for some **u**,**v** $\in$ C and **u** $\neq$ **v**. In this case, the code detects that $\frac{d}{2}$ errors occurred but does not correct them. □

Chapter 2 Linear Codes

One of the goals of coding theory is to construct codes which can be decoded by algorithms which are faster than the brute force decoding scheme of comparing a received word with all of the codewords in C. In this chapter, we will construct codes with algebraic structure that make these codes easier to implement. The additional structure may aid in both encoding and decoding.

Here, we take the alphabet to be $\mathbb{F}_q$, the finite field with q elements. Recall that a code of length n over $\mathbb{F}_q$ is a subset of $\mathbb{F}_q^n$.

Definition 2.1: Let C be a code of length n over the alphabet $\mathbb{F}_q$. The code C is called a *linear code* if C is an $\mathbb{F}_q$-subspace of $\mathbb{F}_q^n$. In other words, a subset C of $\mathbb{F}_q^n$ is a linear code if for all $\mathbf{u},\mathbf{v} \in$ C and $a \in \mathbb{F}_q$, $\mathbf{u}+\mathbf{v}$ and $a\mathbf{u}$ are codewords in C.

Being a vector space, a linear code can be described using a basis. Since a basis for a linear code C spans the entire code, it provides a more efficient way of describing the code than listing every codeword. Let $\{\mathbf{v}_1,\mathbf{v}_2,\ldots,\mathbf{v}_k\}$ be a basis for a linear code C. Then any codeword $\mathbf{w} \in$ C can be written uniquely as a linear combination of the basis elements. In other words, C = $\{a_1\mathbf{v}_1+a_2\mathbf{v}_2+\ldots+a_k\mathbf{v}_k: a_i \in \mathbb{F}_q\}$. Thus, to store a linear code C we need only to store a basis for C, rather than a list of all codewords of C.

Definition 2.2: Let C be a linear code of length n over the alphabet $\mathbb{F}_q$ with basis $\{\mathbf{v}_1,\mathbf{v}_2,\ldots,\mathbf{v}_k\}$. The $k \times n$ matrix $M = \begin{bmatrix} \mathbf{v}_1 \\ \mathbf{v}_2 \\ \vdots \\ \mathbf{v}_k \end{bmatrix}$ is called a *generator matrix* for the code C.

A generator matrix for a linear code is not unique. For example, a nontrivial permutation of the rows of the generator matrix of a linear code C would give rise to another generator matrix for C. However, there is a standard form for a generator matrix of a linear code. Such a generator matrix is of the form $\begin{bmatrix} I_k & X \end{bmatrix}$, where X is a $k \times (n-k)$ matrix with entries in the alphabet $\mathbb{F}_q$. We will see that using a generator matrix in row reduced echelon form as above makes it easier to encode message words.

Definition 2.3: Two linear codes of length n are *equivalent* if one code can be obtained from the other by permuting certain coordinates of all codewords and adding a particular vector $\mathbf{x} = (x_1, ..., x_n) \in \mathbb{F}_q^n$ to all codewords.

Any linear code C is equivalent to a code that has a generator matrix in standard form. This can easily be seen by row reduction of the generator matrix for the code C and permuting the columns if necessary.

Definition 2.4: The *dimension* k of a code C is its dimension as an $\mathbb{F}_q$-vector space.

Let C be a linear code of length n and dimension k. To encode a message using the code C, the length of the message words must equal k. The reason for this will become clear when we discuss encoding using a generator matrix in Section 2.1.

The code C uses n symbols to send k information symbols. Thus, it is said to have information rate $R = \frac{k}{n}$. We would like R to be large so that the code is efficient.

Recall that the minimum distance of a code C is d:= min{d(**x**,**y**) | **x**,**y** $\in$ C and **x** $\neq$ **y**}. If C is a linear code, then the following proposition gives an equivalent description of minimum distance.

Proposition 2.5: Let C be a linear code, the minimum distance d of C is the minimum weight of a nonzero codeword in C. That is, d = min{wt(**c**) | **c** $\in$ C, **c** $\neq$ **0**}.

Proof: Let **u** and **v** be codewords in a linear code C of minimum distance d. Let **w** = **u**−**v**. Since C is a vector space, **w** = **u**−**v** is also a codeword in C. Thus d(**u**,**v**) = wt(**u**−**v**) = wt(**w**). It follows that d = min{d(**u**,**v**) | **u**,**v** $\in$ C and **u** $\neq$ **v**} = min{wt(**w**) | **w** $\in$ C and **w** $\neq$ 0}. □

Definition 2.6: A linear code of length n, dimension k, and minimum distance d is called an $[n, k, d]$ *code*.

The code parameters length, dimension, and minimum distance describe the efficiency and reliability of the code. According to Theorem 1.10, we would like to construct codes with large minimum distance d so we can correct many errors. As seen above, we would like the dimension k to be large with respect to the code length n so that the information rate is high. This would give an efficient code. As the following theorem shows, these goals are somewhat conflicting.

Theorem 2.7 (Singleton Bound): For any $[n, k, d]$ linear code C,

$$k + d \leq n + 1.$$

Proof: Let C be an $[n, k, d]$ code. Let $W := \{(a_1, ..., a_{d-1}, 0..., 0) \in \mathbb{F}_q^n \mid a_i \in \mathbb{F}_q\}$. Clearly W is a linear subspace. By Proposition 2.5, $C \bigcap W = \emptyset$ since all words in W have weight less than d and C has minimum distance d. Since a basis for W is {(1,0,...,0), (0,1,0,...,0), (0,0,1,0,...,0),...,(0,...,0,1,0,...,0)}, the dimension of W is $d-1$. Therefore, we obtain $k + (d-1) = \dim C + \dim W = \dim(C+W) + \dim(C \bigcap W) = \dim(C+W) \leq n$. See [14]. □

Notice that this theorem provides an upper bound on the cardinality of a linear code with a given length and minimum distance. For a code over $\mathbb{F}_q$, the cardinality of a linear code is q^k. By the Singleton Bound, $q^k \leq q^{n-d+1}$.

Given two words $\mathbf{v} = (v_1, v_2, ..., v_n)$ and $\mathbf{w} = (w_1, w_2, ..., w_n)$ in $\mathbb{F}_q^n$, one defines $\mathbf{v} \cdot \mathbf{w} = v_1 w_1 + ... + v_n w_n$. If $\mathbf{v} \cdot \mathbf{w} = 0$, then $\mathbf{v}$ and $\mathbf{w}$ are said to be orthogonal.

Definition 2.8: Let C be an $[n, k, d]$ code over $\mathbb{F}_q$. The *dual code* of C is defined to be $C^\perp := \{\mathbf{u} \in \mathbb{F}_q^n \mid \mathbf{v} \cdot \mathbf{u} = 0 \text{ for all } \mathbf{v} \in C\}$. If $C = C^\perp$, then C is said to be *self dual.*

Theorem 2.9: The dimension of the dual code of C is $n - k$.

Proof: Let M(respectively M′) be a generator matrix for C(respectively $C^\perp$). Since M′ is the nullspace of M, the dimension of $C^\perp$ is equal to the number of columns of M minus the dimension of C. Since there are n columns and the dimension of C is k, the dimension of $C^\perp$ is $n - k$. □

Let C be an $[n, k, d]$ code. Then its dual code $C^\perp$ has length n and dimension $n - k$. However, there is not a formula relating the minimum distances of C and $C^\perp$.

Definition 2.10: Let C be an $[n, k, d]$ code. An $n \times (n-k)$ matrix H is called a *parity-check matrix* for C if the columns of H form a basis for the dual code $C^\perp$.

By definition, if H is a parity-check matrix for C, then $H^\perp$ is a generator matrix for $C^\perp$.

Proposition 2.11: Let C be linear code with parity-check matrix H. Then $\mathbf{c}H = \mathbf{0}$ if and only if $\mathbf{c}$ is a codeword in C.

Proof: Let C be a code in the alphabet $\mathbb{F}_q$ of length n. Let $\mathbf{c} = (c_1, c_2, ..., c_n) \in \mathbb{F}_q^n$ and H $= \begin{bmatrix} \mathbf{H}_1 \\ \vdots \\ \mathbf{H}_n \end{bmatrix}$ be the parity-check matrix for the code C. Suppose $\mathbf{c}H = \mathbf{0}$. Then $\mathbf{c}$ is orthogonal to the columns of H. Since the columns of H form a basis for $C^{\perp}$, $\mathbf{c}$ is an element of the dual code of $C^{\perp}$. Therefore, $\mathbf{c} \in (C^{\perp})^{\perp} = C$.

Now suppose $\mathbf{c}$ is a codeword in C. Since the columns of H form a basis for the dual code $C^{\perp}$, $\mathbf{c}H = \mathbf{0}$ by definition of H. □

Theorem 2.12: Let H be a parity-check matrix for a linear code C. Then C has minimum distance d if and only if any set of $d-1$ rows of H is linearly independent, and at least one set of d rows of H is linearly dependent.

Proof: Let $H = \begin{bmatrix} \mathbf{H}_1 \\ \mathbf{H}_2 \\ \vdots \\ \mathbf{H}_n \end{bmatrix}$ be a parity-check matrix for a $[n, k, d]$ code C. First, assume C has minimum distance d. By Proposition 2.5, there exists a codeword $\mathbf{c} = (c_1, c_2, ..., c_n)$ of C such that wt($\mathbf{c}$) = d. Then $\mathbf{c}H = c_1\mathbf{H}_1 + c_2\mathbf{H}_2 + ... + c_n\mathbf{H}_n = \mathbf{0}$. Since $\mathbf{c} \neq \mathbf{0}$, there is at least one coefficient c_i that must be nonzero. In fact, since wt($\mathbf{c}$) = d, there are exactly d coefficients that are nonzero. Choose the set of d rows of H with nonzero coefficients and renumber, if necessary, so that $c_1\mathbf{H}_1 + c_2\mathbf{H}_2 + ... + c_d\mathbf{H}_d = \mathbf{0}$. This implies that $\{\mathbf{H}_1, ..., \mathbf{H}_d\}$ is a set of d rows of H that are linearly dependent.

Suppose there exists a set of $d-1$ rows of H that is linearly dependent. Then there exists $c_1, ..., c_{d-1} \in F_q$ not all zero such that $c_1\mathbf{H}_1 + c_2\mathbf{H}_2 + ... + c_{d-1}\mathbf{H}_{d-1} = \mathbf{0}$. Let $\mathbf{c} = (c_1, c_2, ..., c_{d-1}, 0, ..., 0)$. Then $\mathbf{c}H = c_1\mathbf{H}_1 + c_2\mathbf{H}_2 + ... + c_{d-1}\mathbf{H}_{d-1} = \mathbf{0}$ implies that $\mathbf{c} \in C$ by Proposition 2.11. Since the minimum distance of C is d and wt($\mathbf{c}$) = $d-1$, this is a contradiction. Therefore, any set of $d-1$ rows of H is linearly independent.

Now assume any set of $d-1$ rows of H is linearly independent, and at least one set of d rows of H is linearly dependent. Then the minimum distance of C is less than or equal to d. Let $\mathbf{v} = (v_1, v_2, ..., v_n) \in C$. Suppose that wt($\mathbf{v}$) < d. Then $\mathbf{v}$ has at most $d-1$ nonzero coordinates. Let $v_{i_1}, v_{i_2}, ..., v_{i_j}$, where $j \leq d-1$, be the set of nonzero

coordinates of **v**. By Proposition 2.11, **vH** = **0**. Thus $v_{i_1}\mathbf{H}_{i_1} + v_{i_2}\mathbf{H}_{i_2} + ... + v_{i_j}\mathbf{H}_{i_j} = \mathbf{0}$, which is a contradiction. Thus, the wt(**v**) cannot be less than d. By Proposition 2.5, the minimum distance of C must be d. See [9]. □

If we are given a generator matrix for a linear code C, then we can find a parity-check matrix H for C using the following algorithm.

Algorithm 2.13: Let C be an $[n, k, d]$ code. Let A be the matrix whose rows are all the words in C. Use elementary row operations to reduce A to reduced row echelon form. Since the dimension of C is k, the matrix with all nonzero rows has dimensions $k \times n$. Let M be the $k \times n$ matrix consisting of all nonzero rows of the reduced row echelon form of A. Let X be the $k \times (n-k)$ matrix obtained from M by deleting the leading columns of M. Form an $n \times (n-k)$ matrix H as follows:

i. In the rows of H corresponding to the leading columns of M, place in order, the rows of X. The first k rows of H are the rows (in order) of X.

ii. The remaining $n-k$ rows of H (in order) are the rows of the $(n-k) \times (n-k)$ identity matrix $I_{(n-k)}$.

See [9].

Let M_C and $M_{C^\perp}$ denote the generator matrices for the code C and the dual code $C^\perp$, respectively. Let H_C and $H_{C^\perp}$ denote the parity-check matrices for the code C and the dual code $C^\perp$, respectively. Given either the generator matrix M or the parity-check matrix H, we can form the remaining matrices using the diagram in Figure 2.

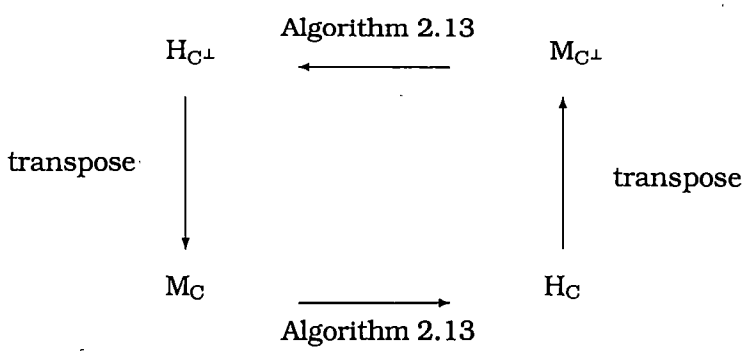

Figure 2. A diagram for forming generator matrices.

Example 2.14: Let C be a linear code with generator matrix

$$M = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 1 & 1 \end{bmatrix}. \text{ Then } X = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 1 \end{bmatrix}.$$

By the algorithm, $H = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$ is the parity-check matrix for the code C.

2.1 Encoding

Let C be an $[n, k, d]$ code over $\mathbb{F}_q$ with generator matrix $M = \begin{bmatrix} c_{11} & c_{12} & \cdots & c_{1n} \\ c_{21} & c_{22} & \cdots & c_{2n} \\ \vdots & & & \vdots \\ c_{k1} & c_{k2} & \cdots & c_{kn} \end{bmatrix}$. Suppose we wish to send a message using the code C. The message is broken into message words of equal length k. Then matrix multiplication is used to encode the message words. In particular, the message word $\mathbf{w} = (w_1, w_2, ..., w_k) \in \mathbb{F}_q$ is encoded as the codeword $\mathbf{w}M = (w_1 \cdot c_{11} + w_2 \cdot c_{21} + \cdots + w_k \cdot c_{k1}, w_1 \cdot c_{12} + w_2 \cdot c_{22} + \cdots + w_k \cdot c_{k2}, \ldots, w_1 \cdot c_{1n} + w_2 \cdot c_{2n} + \cdots + w_k \cdot c_{kn})$. Thus the code C is the set of words $\{\mathbf{w}M | \ \mathbf{w} \in \mathbb{F}_q^k\}$.

For codes that are not linear codes, there is no easy way to associate a codeword with a message word. In fact, a table may be used to encode codewords. Using matrix multiplication to encode messages is advantageous since it is faster than using a table and less storage space is needed to store the generator matrix than a list of codewords.

2.2 Decoding

Linear codes also have the advantage of decoding without using a table to lookup the most likely codeword sent through the channel. We decode received words using cosets.

Let C be an $[n, k, d]$ code over $\mathbb{F}_q$ with generator matrix M. Given a word $\mathbf{u} \in \mathbb{F}_q^n$, the set $\mathbf{u}+C = \{\mathbf{u}+\mathbf{x} \mid \mathbf{x} \in C\}$ is called the coset of C determined by $\mathbf{u}$. Note that two words $\mathbf{u}$ and $\mathbf{v}$ in $\mathbb{F}_q^n$ determine the same coset of C if and only if $\mathbf{u}-\mathbf{v} \in C$. Let $\mathbf{w}$ be a message word. Then $\mathbf{w}$ is encoded as the codeword $\mathbf{v} = \mathbf{w}M$. The codeword $\mathbf{v}$ is sent through the channel. Let $\mathbf{y}$ denote the received word. Due to noise in the channel, it may be the case that $\mathbf{y} \neq \mathbf{v}$. Let $\mathbf{e} = \mathbf{y}-\mathbf{v}$ denote the error vector. Then $\mathbf{y}$ and $\mathbf{e}$ determine the same coset of C since $\mathbf{y}-\mathbf{e} = \mathbf{y}-(\mathbf{y}-\mathbf{v}) = \mathbf{v} \in C$. Therefore the possible error vectors are exactly the vectors in the coset of C determined by $\mathbf{y}$. Since we use maximum likelihood decoding, we assume that errors of small weight are most likely to occur. Therefore we choose a minimum weight vector $\mathbf{e}$ in the coset containing $\mathbf{y}$ and decode $\mathbf{y}$ as $\mathbf{y}-\mathbf{e} = \mathbf{v}$.

Note that there exists a unique vector of minimum weight if the weight of $\mathbf{e}$ is less than $t = \lfloor \frac{d-1}{2} \rfloor$. This allows correct decoding up to minimum distance d.

Finding the coset containing the received word $\mathbf{y}$ and then finding an error vector of least weight in that coset may make the decoding procedure difficult. To ease the process, we can use a parity-check matrix for C.

Definition 2.15: Let H be a parity-check matrix for an $[n, k, d]$ code C. Given a word $\mathbf{u}$ in $\mathbb{F}^n$, the *syndrome* of $\mathbf{u}$ is defined to be the word $\mathbf{u}H$ in $\mathbb{F}^{n-k}$.

Syndromes help find the most likely positions where errors occurred. In the next theorem, we see that if the syndrome of a received word $\mathbf{y}$ equals the syndrome of the most likely error vector $\mathbf{e}$, then $\mathbf{y}$ and $\mathbf{e}$ are in the same coset.

Theorem 2.16: Let C be an $[n, k, d]$ code over $\mathbb{F}_q$ with parity-check matrix H. Then $\mathbf{u}H = \mathbf{v}H$ if and only if $\mathbf{u}$ and $\mathbf{v}$ lie in the same coset of C.

Proof: Let $\mathbf{u},\mathbf{v} \in \mathbb{F}_q^n$. Suppose $\mathbf{u}H = \mathbf{v}H$. Then $(\mathbf{u}-\mathbf{v})H = \mathbf{0}$. By Proposition 2.11, $\mathbf{u}-\mathbf{v} \in C$. Therefore $\mathbf{u}$ and $\mathbf{v}$ are in the same coset.

Now suppose $\mathbf{u}$ and $\mathbf{v}$ are in the same coset. Then $\mathbf{u}-\mathbf{v} \in C$. By Proposition 2.11, $(\mathbf{u}-\mathbf{v})H = \mathbf{0}$. Thus $\mathbf{u}H-\mathbf{v}H = \mathbf{0}$ implies $\mathbf{u}H = \mathbf{v}H$. □

According to Theorem 2.16, a coset can be identified by its syndrome. For an error vector $\mathbf{e} = (e_1, \ldots, e_n)$ and parity-check matrix $H = \begin{bmatrix} \mathbf{H}_1 \\ \vdots \\ \mathbf{H}_n \end{bmatrix}$, the syndrome $\mathbf{e}H = e_1\mathbf{H}_1 + \ldots + e_n\mathbf{H}_n$

is the sum of the rows of H such that $e_i \neq 0$. Thus, the syndrome equals the sum of the rows of H corresponding to the location of the most likely errors. Since a received word **y** has syndrome equal to the error vector **e** of least weight, **y**H indicates the positions of the most likely errors.

Example 2.17: Let C be a $[n, k, d]$ linear code with parity-check matrix H given in Example 2.14. Suppose **y** = (1,1,1,0,0,0,1) is the received word. The syndrome is **y**H = (1,0,1). Since **y**H $\neq$ **0**, **y** $\notin$ C implies an error occurred. Since (1,0,1) is the third row of H, the most likely error is in position 3. We can conclude that **e** = (0,0,1,0,0,0,0) is the error vector. Note that **e**H = **y**H. Thus **y** is decoded to **v** = (1,1,1,0,0,0,1)+(0,0,1,0,0,0,0) = (1,1,0,0,0,0,1). Since **v**H = **0**, **v** $\in$ C.

2.3 Cyclic Codes

Definition 2.18: A $[n, k, d]$ linear code C is called a *cyclic code* if $\mathbf{c} = (c_0, c_1, ..., c_{n-1}) \in$ C implies $(c_{n-1}, c_0, c_1 ..., c_{n-2}) \in$ C.

Given a word $\mathbf{v} = (v_0, v_1, ..., v_{n-1}) \in \mathbb{F}_q^n$, define $v(x) = v_0 + v_1 x + ... + v_{n-1}x^{n-1} \in \mathbb{F}_q[x]$ to be the polynomial with coefficients from **v**. There is a one-to-one correspondence between a word $\mathbf{v} \in \mathbb{F}_q^n$ and polynomial $v(x) \in \mathbb{F}_q[x]$ of degree less than n defined by

$$\mathbf{v} = (v_0, v_1, ..., v_{n-1}) \longleftrightarrow v(x) = v_0 + v_1 x + ... + v_{n-1}x^{n-1}.$$

In this section, we will use the polynomial interchangeably with the codeword. Using polynomials, we give an equivalent definition of cyclic codes.

Definition 2.19: A $[n, k, d]$ code C is *cyclic* if $c(x) \in$ C implies $xc(x) \text{mod}(x^n - 1) \in$ C.

Example 2.20: Let C be a linear code over $\mathbb{F}_2^6$ with **v** = (1,1,0,1,0,0) $\in$ C. Then **v** corresponds to the polynomial $v(x) = 1 + x + x^3$. Then the following are also codewords in C:

$x^0 v(x) \equiv 1 + x + x^3 mod(x^7 - 1) \longleftrightarrow (1,1,0,1,0,0,0)$

$x^1 v(x) \equiv x + x^2 + x^4 mod(x^7 - 1) \longleftrightarrow (0,1,1,0,1,0,0)$

$x^2 v(x) \equiv x^2 + x^3 + x^5 mod(x^7 - 1) \longleftrightarrow (0,0,1,1,0,1,0)$

$x^3v(x) \equiv x^3 + x^4 + x^6 mod(x^7 - 1) \longleftrightarrow (0,0,0,1,1,0,1)$

$x^4v(x) \equiv 1 + x^4 + x^5 mod(x^7 - 1) \longleftrightarrow (1,0,0,0,1,1,0)$

$x^5v(x) \equiv x + x^5 + x^6 mod(x^7 - 1) \longleftrightarrow (0,1,0,0,0,1,1)$

$x^6v(x) \equiv 1 + x^2 + x^6 mod(x^7 - 1) \longleftrightarrow (1,0,1,0,0,0,1)$

Theorem 2.21: A linear code in $\mathbb{F}_q$ is cylic if and only if C is an ideal in $\mathbb{F}_q[x]/(x^n - 1)$.

Proof: Suppose C is a cyclic code. Then $c(x) \in$ C implies that $xc(x)\text{mod}(x^n - 1) \in$ C. Therefore, $x^i c(x)\text{mod}(x^n - 1) \in$ C for all $i \in \mathbb{N}$. Since C is linear, $a(x)c(x) \in$ C for all $a(x) \in \mathbb{F}_q[x]$. Thus, C is an ideal in $\mathbb{F}_q[x]/(x^n - 1)$.

Suppose C is an ideal in $\mathbb{F}_q[x]/(x^n - 1)$. Let $c(x) \in$ C. Then for $x \in \mathbb{F}_q[x]$, $xc(x) \in$ C. Thus, C is a cyclic code. □

Definition 2.22: Let C be a cyclic code of length n. A monic polynomial $g(x) \in$ C of minimal degree such that C = $\langle g(x) \rangle$ is called the *generator polynomial* of C.

Theorem 2.23: Let C be a cyclic code of length n. There exists a unique monic generator polynomial $g(x)$ of minimal degree.

Proof: Every ideal in $\mathbb{F}_q[x]/(x^n - 1)$ is a principal ideal. Thus, there exists a monic polynomial $g(x) \in$ C such that C = $\langle g(x) \rangle$.

Suppose $f(x), g(x) \in$ C are both monic generator polynomials of minimum degree r. Since C is a linear code, $f(x) - g(x) \in$ C. However, deg$(f(x) - g(x)) < r$. This is a contradiction unless $f(x) = g(x)$. □

Theorem 2.24: The generator polynomial for a linear cyclic code of length n is a factor of $x^n - 1$.

Proof: Let C be a cyclic code of length n and $g(x)$ be the generator polynomial of C. By the division algorithm we write $x^n - 1 = f(x)g(x) + r(x)$, where $g(x)$, $f(x)$, and $r(x)$ are polynomials in $\mathbb{F}_q[x]$ and the deg$(r(x)) <$ deg$(g(x))$. Then $(x^n - 1) = (f(x)g(x) + r(x))\text{mod}(x^n - 1)$, which implies that $r(x) \equiv (-f(x)g(x))\text{mod}(x^n - 1)$. Since $f(x)g(x) \in$ C, $r(x) \in$ C. But deg$(r(x)) <$ deg$(g(x))$ implies that $r(x) = 0$. Thus $x^n - 1 = f(x)g(x)$. This shows that $g(x)$ divides $x^n - 1$. □

Let $g(x)$ be the generator polynomial for a cyclic code C of length n. By Theorem 2.24, $g(x)$ divides $x^n - 1$. Therefore, there exists a unique $h(x) \in \mathbb{F}_q[x]$ such that $h(x)g(x) = x^n - 1$. It follows that $h(x)g(x) \equiv 0 \bmod(x^n - 1)$. Thus, $h(x) = \frac{x^n-1}{g(x)}$ implies that the $\deg(h(x)) = n - \deg(g(x))$.

Definition 2.25: Let $g(x)$ be the generator polynomial for a cyclic code C. If $h(x)g(x) \equiv 0 \bmod(x^n - 1)$, then $h(x)$ is called the *check-polynomial.*

Let $g(x)$ be a generator polynomial and $h(x)$ be the check-polynomial for a cyclic code C. Notice that $c(x) \equiv a(x)g(x) \bmod(x^n - 1)$ for all $c(x) \in$ C for some $a(x) \in \mathbb{F}_q[x]$. Therefore, $c(x)h(x) \equiv 0 \bmod(x^n - 1)$ since $c(x)h(x) = a(x)g(x)h(x) \equiv 0 \bmod(x^n - 1)$ for all $c(x) \in$ C.

Theorem 2.26: Let C be a cyclic code of length n and let $g(x)$ be the generator polynomial. Let $k = n - \deg(g(x))$. Then the codewords corresponding to $g(x), xg(x), ..., x^{k-1}g(x)$ form a basis for C. Thus C has dimension k.

Proof: Let $g(x) = b_0 + b_1x + ... + b_{n-k}x^{n-k}$. Suppose $a_0g(x) + a_1xg(x) + ... + a_{k-1}x^{k-1}g(x) = 0$. Then $a_0(b_0 + b_1x + ... + b_{n-k}x^{n-k}) + a_1x(b_0 + b_1x + ... + b_{n-k}x^{n-k}) + ... + a_{k-1}x^{k-1}(b_0 + b_1x + ... + b_{n-k}x^{n-k}) = 0$. This implies $a_0b_0 + (a_0b_1 + a_1b_0)x + (a_0b_2 + a_1b_1 + a_2b_0)x^2 + ... + a_{k-1}b_{n-k}x^{n-1} = 0$. Clearly, $\{1, x, x^2, ..., x^{n-1}\}$ is a linearly independent set over $\mathbb{F}_q$. Thus, $a_0b_0 = 0$, $a_0b_1 + a_1b_0 = 0$, $a_0b_2 + a_1b_1 + a_2b_0 = 0$,..., and $a_{k-1}b_{n-k} = 0$. Consider $a_0b_0 = 0$. If $b_0 = 0$, then $g(x) = b_0 + b_1x + ... + b_{n-k}x^{n-k} = x(b_1 + ... + b_{n-k}x^{n-k-1})$. This implies that $b_1 + ... + b_{n-k}x^{n-k-1} \in$ C with degree less than $\deg(g(x))$. This is a contradiction. Thus, $b_0 \neq 0$ and $a_0 = 0$. Now consider $a_0b_1 + a_1b_0 = 0$. We know that $b_0 \neq 0$ and $a_0 = 0$. This implies that $a_1 = 0$. Similarly for $a_0b_2 + a_1b_1 + a_2b_0 = 0$,..., and $a_{k-1}b_{n-k} = 0$. Thus, $a_j = 0$ for all $0 \leq j \leq k - 1$. Therefore, $\{g(x), xg(x), ..., x^{k-1}g(x)\}$ is a linearly independent set.

For all $c(x) \in$ C, $c(x) = f(x)g(x)$ for some $f(x) = f_0 + f_1x + ... + f_{k-1}x^{k-1} \in \mathbb{F}_q[x]$. Note that $\deg(f(x)) = k-1$ since $\deg(c(x)) \leq n-1$. Thus, $c(x) = (f_0+f_1x+...+f_{k-1}x^{k-1})g(x) = f_og(x) + f_1xg(x) + ... + f_{k-1}x^{k-1}g(x)$. Therefore, $c(x) \in \langle\{g(x), xg(x), ..., x^{k-1}g(x)\}\rangle$. This implies that $\{g(x), xg(x), ..., x^{k-1}g(x)\}$ is a basis for C. □

We can form a generator matrix for a cyclic code C using the generator polynomial $g(x)$. From Theorem 2.26, we know that $\{g(x), xg(x), ..., x^{k-1}g(x)\}$ forms a basis for C. Thus

$$\begin{bmatrix} g(x) \\ xg(x) \\ \vdots \\ x^{k-1}g(x) \end{bmatrix}$$ is a generator matrix for C.

Example 2.27: Since $v(x)$ from Example 2.20 divides $x^7 - 1$, $v(x)$ is the generator polynomial for the linear cyclic code $C = \langle v(x) \rangle$. $n = 7$ and $\deg(v(x)) = 3$ imply that the dimension of C is $k = 4$. Therefore the codewords corresponding to $v(x)$, $xv(x)$, $x^2v(x)$, $x^3v(x)$ form a basis for C. Thus, a generator matrix for C is $\begin{bmatrix} 1 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 1 \end{bmatrix}$.

Encoding for cyclic codes is done by polynomial mulitplication. For a message word $a(x) \in \mathbb{F}_q[x]$, $a(x)g(x) \bmod (x^n - 1)$ is the corresponding codeword. The advantage of using a cyclic code is that less storing space is required. Instead of storing a $k \times n$ generator matrix, we need only to store the generator polynomial $g(x)$.

The additional structure of a cyclic code also aids in decoding. Finding the message word corresponding to the closest codeword $c(x)$ to the received word consists of dividing polynomials, namely $c(x)$ by $g(x)$. As in all linear codes, decoding consists of finding the syndrome. For cyclic codes, we call this the syndrome polynomial. See [12] for more information about decoding cyclic codes.

2.4 Hamming Codes

Now we will look at some examples of linear codes. We first consider an important family of codes discovered by R.W. Hamming in the late 1940's. These codes, called Hamming codes, are among the first error correcting codes discovered. All codes considered in this section are binary.

Definition 2.28: Let $r \geq 2$ and H_r be a $(2^r - 1) \times r$ matrix whose rows are all nonzero binary words of length r. The binary code with parity-check matrix H_r is called a *Hamming code*, denoted Ham(r)[9].

Theorem 2.29: The Hamming code Ham(r) has length $n = 2^r - 1$ and dimension $k = 2^r - r - 1$.

Proof: By definition, the length n of Ham(r) is $2^r - 1$.

Since H_r is the parity-check matrix, the r columns of H_r form a basis for the dual code of Ham(r). By Theorem 2.9, the dimension of the dual code of Ham(r) is $r = n-k$, where k is the dimension of Ham(r). Since $n = 2^r - 1$, $k = 2^r - 1 - r$. □

Corollary 2.30: Ham(r) has minimum distance $d = 3$.

Proof: Since no row of H_r is the zero word, no single row of H_r is linearly dependent. By Theorem 2.12, $d > 1$. Let R_i and R_j be two distinct rows of H_r. Since $R_i \neq R_j$ and R_i,$R_j \neq 0$, $aR_i + bR_j = 0$ with $a, b \in \{0, 1\}$ implies that $a = b = 0$. Thus any two rows of H_r are linearly independent. Note that (1,0,0,...,0), (0,1,0,...,0), and (1,1,0,...,0) are rows of H_r by construction. These three rows form a linearly dependent set . By Theorem 2.12, Ham(r) has distance $d = 3$. See [13]. □

Recall from Theorem 1.10 that a code with minimum distance $d = t + 1$ can correct all errors in t or fewer positions. The ISBN code in Example 1.9 shows that some codes may be able to detect certain kinds of errors.

However, there are $[n, k, d]$ codes that cannot correct any error vectors of weight greater than $t = \lfloor \frac{d-1}{2} \rfloor$. Such codes are called perfect codes. Perfect codes have maximum dimension among the codes that can correct error vectors of weight less than t.

Definition 2.31: A code C of length n and odd distance $d = 2t + 1$ is called a *perfect code* if $|C| = \frac{2^n}{\binom{n}{0}+\binom{n}{1}+\cdots+\binom{n}{t}}$.

Since a Hamming code is a binary code with dimension $k = 2^r - r - 1$, there are 2^k codewords in the code. For a Hamming code of length $n = 2^r - 1$ and distance $d = 3$, $|\text{Ham}(r)| = \frac{2^n}{\binom{n}{0}+\binom{n}{1}} = \frac{2^{2^r-1}}{1+n} = \frac{2^{2^r-1}}{2^r} = 2^{2^r-r-1}$, since $t = \lfloor \frac{d-1}{2} \rfloor = \lfloor \frac{3-1}{2} \rfloor = 1$. Thus Ham($r$) is a perfect code.

Theorem 2.32: If C is a perfect code of length n and minimum distance $d = 2t + 1$, then C will correct all error vectors of weight less than or equal to t, and no other error vectors.

Proof: By Theorem 1.10, C will correct all error vectors of weight less than or equal to t. Therefore $B_t(\mathbf{u}) \bigcap B_t(\mathbf{v}) = \emptyset$ for all $\mathbf{u},\mathbf{v} \in$ C with $\mathbf{u} \neq \mathbf{v}$. Recall that $\binom{n}{t}$ is the number of ways an unordered collection of t objects can be chosen from a set of n objects. Thus, $\binom{n}{t}$ is the number of binary codewords of length n and weight t. For each $\mathbf{u} \in$ C, $B_t(\mathbf{u})$ contains $\binom{n}{0} + \binom{n}{1} + ... + \binom{n}{t}$ words of length n and weight less than or equal to t. Since there are $|C|$ disjoint spheres of radius t around the codewords of C, there are exactly $|C| \cdot (\binom{n}{0} + \binom{n}{1} + ... + \binom{n}{t})$ words in the union of the spheres. By definition of perfect codes, $|C| \cdot (\binom{n}{0} + \binom{n}{1} + ... + \binom{n}{t}) = 2^n$, which is the number of all possible words of length n. This implies that C corrects only error vectors $\mathbf{e}$ such that wt($\mathbf{e}$) $\leq t$. □

Since Hamming codes correct error vectors of weight $t = 1$, Hamming codes are perfect single-error correcting codes.

Example 2.33: Let $H_3 = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$. H_3 is the parity-check matrix for Ham(3). Using Algorithm 2.13, we find that the generator matrix for Ham(3) is

$$M = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 1 & 1 \end{bmatrix}.$$ Ham(3) has length $n = 2^r - 1 = 7$, dimension $k = 2^r - 1 - r = 4$, and minimum distance $d = 3$. Thus Ham(3) is a [7,4,3] linear code.

If $\mathbf{y}$ = (1,1,0,1,0,0,1) is the word received, then $\mathbf{y}$H =(0,1,1) is the syndrome. Notice that $\mathbf{y}$H = (0,1,1) is the fourth row of H. Thus the error occurred in the fourth position. Since Ham(3) is a binary code, we know that the error vector is $\mathbf{e}$ = (0,0,0,1,0,0,0). Therefore we decode $\mathbf{y}$ as $\mathbf{y}-\mathbf{e}$ = (1,1,0,0,0,0,1) to be the word sent.

2.5 Reed-Muller Codes

Our next family of codes to consider is the Reed-Muller codes. I.S. Reed and D.E. Muller discovered these codes in 1954. The first order Reed-Muller codes are good codes for very noisy channels. In fact, the first order Reed-Muller code RM(1,5) was used in 1972 by the Mariner 9 spacecraft to transmit a photograph of Mars. Beyond the first order Reed-Muller codes, the minimum distances of Reed-Muller codes are less than the minimum distances of BCH codes discovered in 1960. For more information on BCH codes see [9], [12], [13], or [14]. Therefore BCH codes are used instead of Reed-Muller codes. However, Reed-Muller codes are used to form families of codes which we will discuss in later Chapter 3.

If $\mathbf{x} = (x_1, ..., x_m) \in A^m$ and $\mathbf{y} = (y_1, ..., y_n) \in A^n$, then $(\mathbf{x},\mathbf{y}) = (x_1, ..., x_m, y_1, ..., y_n) \in A^{m+n}$. This notation will be used to describe codewords in Reed-Muller codes.

Definition 2.34: Let $0 \leq r \leq m$. The r^{th} order *Reed-Muller* code of length 2^m will be denoted by RM(r, m). RM(r, m) is defined recursively as follows: RM$(0, m)$ = {(0,0,...,0), (1,1,...,1)}, RM(r, m) = $\{(\mathbf{x},\mathbf{x}+\mathbf{y}) \in \mathbb{F}_2^{2^m} \mid \mathbf{x} \in \text{RM}(r, m-1), \mathbf{y} \in \text{RM}(r-1, m-1)\}$ for $0 < r < m$, and RM$(m, m) = \mathbb{F}_2^{2^m}$.

Proposition 2.35: Let $0 \leq r \leq m$. RM$(r-1, m) \subseteq$ RM(r, m).

Proof: We induct on m. Suppose RM$(r-1, m') \subseteq$ RM(r, m') is true for all $m' < m$. Let $\mathbf{u} \in$ RM$(r-1, m)$. Then $\mathbf{u} = (\mathbf{x},\mathbf{x}+\mathbf{y})$ where $\mathbf{x} \in$ RM$(r-1, m-1)$ and $\mathbf{y} \in$ RM$(r-2, m-1)$. By hypothesis RM$(r-1, m-1) \subseteq$ RM$(r, m-1)$ and RM$(r-2, m-1) \subseteq$ RM$(r-1, m-1)$. Therefore, $\mathbf{x} \in$ RM$(r, m-1)$ and $\mathbf{y} \in$ RM$(r-1, m-1)$, which implies that $\mathbf{u} \in$ RM(r, m).

□

Since RM(r, m) is a linear code, it has a generator matrix, which we will denote by G(r, m).

Proposition 2.36: Let $0 \leq r \leq m$. G$(0, m)$ = [1 1 ... 1], G$(m, m) = \begin{bmatrix} G(m-1, m) \\ 0 \cdots 01 \end{bmatrix}$, and G$(r, m)$

$= \begin{bmatrix} G(r, m-1) & G(r, m-1) \\ 0 & G(r-1, m-1) \end{bmatrix}$ for $0 < r < m$.

Proof: Let $r = 0$. Then RM$(0, m)$ = {(0,0,...,0), (1,1,...,1)} is clearly generated by G$(0, m)$. Next, let $r < m$. To prove RM(r, m) is the code generated by G(r, m), we induct on

m. Suppose the hypothesis is true for all $m' < m$. Let $\mathbf{u} = (u_1, u_2, ..., u_k) \in \mathbb{F}_2^k$. Then $\mathbf{u}\begin{bmatrix} G(r,m-1) \\ 0 \end{bmatrix} = \mathbf{x}+\mathbf{0}$, where $\mathbf{x} \in \text{RM}(r,m-1)$ by hypothesis and

$\mathbf{u}\begin{bmatrix} G(r,m-1) \\ G(r-1,m-1) \end{bmatrix} = \mathbf{x}+\mathbf{y}$, where $\mathbf{y} \in \text{RM}(r-1,m-1)$ by hypothesis. Thus

$\mathbf{u}\begin{bmatrix} G(r,m-1) & G(r,m-1) \\ 0 & G(r-1,m-1) \end{bmatrix} = (\mathbf{x},\mathbf{x}+\mathbf{y}) \in \text{RM}(r,m)$. This implies that $\text{RM}(r,m) \subseteq \text{G}(r,m)$. Since $|\text{RM}(r,m)| = |\text{G}(r,m)|$, $\text{RM}(r,m)$ is the code generated by $\text{G}(r,m)$. The proof is similar for $r = m$. □

Example 2.37:

i. RM(0,2) = {(0,0,0,0), (1,1,1,1)} and G(0,2) = [1111]

ii. RM(1,2) = {(**x**,**x**+**y**) | **x** ∈ RM(1,1), **y** ∈ RM(0,1)}

= {(**x**,**x**+**y**) | **x** ∈ {(0,0), (0,1), (1,0), (1,1)}, **y** ∈ {(0,0), (1,1)}}

= {(0,0,0,0), (0,0,1,1), (0,1,0,1), (0,1,1,0), (1,0,0,0), (1,0,0,1), (1,1,0,0), (1,1,1,1)}

and $\text{G}(1,2) = \begin{bmatrix} G(1,1) & G(1,1) \\ 0 & G(0,1) \end{bmatrix} = \begin{bmatrix} G(0,1) & G(0,1) \\ 0...01 & 0...01 \\ 0 & 1...11 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 \end{bmatrix}$

iii. $\text{RM}(2,2) = \mathbb{F}_2^4$ and $\text{G}(2,2) = \begin{bmatrix} G(1,2) \\ 0...01 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 \end{bmatrix}$

Proposition 2.38: The Reed-Muller code $\text{RM}(r,m)$ has dimension $k = \sum_{i=0}^{r} \binom{m}{i}$ and minimum distance $d = 2^{m-r}$.

To prove this proposition, we will use the following lemma. Let $d_{\text{RM}(r,m)}$ denote the minimum distance of $\text{RM}(r,m)$. Note that $d_{\text{RM}(0,m)} = 2^m$ since (0,...,0) and (1,...,1) are the only two codewords in $\text{RM}(0,m)$.

Lemma 2.39: The minimum distance of RM(r,m) is $d = \min\{2 \cdot d_{\text{RM}(r,m-1)}, d_{\text{RM}(r-1,m-1)}\}$.

Proof of Lemma 2.39: Recall that RM(r,m) = {(**x**,**x**+**y**) $\in \mathbb{F}_q^{2^m}$ | **x** $\in$ RM$(r,m-1)$, **y** $\in$ RM$(r-1,m-1)$}. Let **a** = (**u**,**u**+**v**) , **b** = (**u'**,**u'**+**v'**) be distinct codewords of RM(r,m) where **u**,**u'** $\in$ RM$(r,m-1)$, **v**,**v'** $\in$ RM$(r-1,m-1)$. If **v** = **v'**, then d(**a**,**b**) = d(**u**,**u'**)+d(**u**+**v**,**u'**+**v'**) = d(**u**,**u'**)+d(**u**+**v**,**u'**+**v**) = 2d(**u**,**u'**) $\geq 2d_{\mathrm{RM}(r,m-1)}$, as **u**,**u'** $\in$ RM$(r,m-1)$. Since $d_{\mathrm{RM}(r,m-1)}$ is the minimum distance of RM$(r,m-1)$, the minimum distance of RM(r,m) = $2d_{\mathrm{RM}(r,m-1)}$. Now suppose **v** $\neq$ **v'**. Then d(**a**,**b**) = wt(**a**−**b**) = wt((**u**,**u**+**v**)−(**u'**,**u'**+**v'**)) = wt(**u**−**u'**) +wt(**u**−**u'**+**v**−**v'**) $\geq$ wt(**u**−**u'**)+ wt(**v**−**v'**)−wt(**u**−**u'**) = wt(**v**−**v'**) $\geq d_{\mathrm{RM}(r-1,m-1)}$. See [12]. □

Proof of Proposition 2.38: Let k denote the dimension of RM(r,m) and d denote the minimum distance of RM(r,m). We will prove $k = \sum_{i=0}^{r} \binom{m}{i}$ and $d = 2^{m-r}$ by induction on m. Suppose the proposition holds for all RM(r',m'), where $r' \leq r$ and $m' \leq m$. Then RM(r',m') has dimension $\sum_{i=0}^{r'} \binom{m'}{i}$ and minimum distance $2^{m'-r'}$.

Now consider RM(r,m). If $m = r$, then RM$(m,m) = \mathbb{F}_2^{2^m}$. Clearly, $d = 1$ since (1,0,...,0) $\in$ RM(m,m) and wt(1,0,...,0) = 1. RM(m,m) has generator matrix G(m,m) = $\begin{bmatrix} G(m-1,m) \\ 0\cdots01 \end{bmatrix}$. The matrix G$(m,m)$ has rank equal to the dimension of RM$(m-1,m)$ plus one. Thus, the dimension of RM(m,m) is $\sum_{i=0}^{m-1} \binom{m}{i} + 1 = \sum_{i=0}^{m-1} \binom{m}{i} + \binom{m}{m} = \sum_{i=0}^{m} \binom{m}{i}$.

If $r < m$, $d = \min\{2 \cdot d_{RM(r,m-1)}, d_{RM(r-1,m-1)}\} = \min\{2 \cdot 2^{m-1-r}, 2^{m-1-(r-1)}\} = 2^{m-r}$. G$(r,m)$ = $\begin{bmatrix} G(r,m-1) & G(r,m-1) \\ 0 & G(r-1,m-1) \end{bmatrix}$ has rank equal to the sum of the dimension of RM$(r,m-1)$ and the dimension of RM$(r-1,m-1)$. Thus, the dimension of RM(r,m) is $\sum_{i=0}^{r} \binom{m-1}{i} + \sum_{i=0}^{r-1} \binom{m-1}{i}$. Using the well-known fact that $\binom{m}{i} = \binom{m-1}{i} + \binom{m-1}{i-1}$ and $\binom{m-1}{0} = \binom{m}{0} = 1$, we have $\sum_{i=0}^{r} \binom{m-1}{i} + \sum_{i=0}^{r-1} \binom{m-1}{i} = \sum_{i=1}^{r} \left(\binom{m-1}{i} + \binom{m-1}{i-1}\right) + \binom{m-1}{0} = \sum_{i=0}^{r} \binom{m}{i}$. See [12]. □

Example 2.40: RM(1,2) is a [4,3,2] linear code.

2.6 Golay Code

The Golay code was constructed by M.J.E. Golay in 1949. The Golay code was used to transmit photographs of Jupiter and Saturn in the Voyager spacecraft program in the

early 1980's.

Definition 2.41: The *extended Golay code* G_{24} is the linear code with generator matrix

$$G = \left[\begin{array}{ccccccccccccc}
 & 1 & & 1 & 1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 0 \\
 & 1 & & 0 & 1 & 1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \\
 & 1 & & 1 & 0 & 1 & 1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\
I_{7\times 7} & 1 & 0_{7\times 5} & 0 & 1 & 0 & 1 & 1 & 0 & 1 & 1 & 1 & 0 & 0 \\
 & 1 & & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 1 & 1 & 1 & 0 \\
 & 1 & & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 1 & 1 & 1 \\
 & 1 & & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 1 & 1 \\
 & 1 & & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 1 \\
 & 1 & & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 0 \\
0_{5\times 7} & 1 & I_{5\times 5} & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 1 \\
 & 1 & & 1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 1 \\
 & 0 & & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1
\end{array}\right].$$

Proposition 2.42: The Golay code G_{24} is a [24,12,8] code.

Proof: Looking at the generator matrix G, it is clear that the length of the codewords is 24 and the dimension of the code is 12. To see that the minimum distance is 8 see [12]. □

Theorem 2.43: G_{24} is self-dual.

Proof: If **u** and **v** are rows of G, then by inspection, one sees that wt(**u** · **v**) =0(mod 2). Therefore every row of G is orthogonal to every other row and so $G_{24} \subseteq G_{24}^{\perp}$. But the dimension of G_{24} is 12 which is also the dimension of $G_{24}^{\perp}$(since $n - k = 12$). This implies that $G_{24} = G_{24}^{\perp}$. □

2.7 MacWilliams Identity for Binary Linear Codes

Let C be a binary $[n, k, d]$ code. By Proposition 2.5, C has at least one codeword of weight d. One may be interested in how many codewords have weight d. More generally, we may ask how many codewords there are of weight i, for $d \leq i \leq n$. Let A_i denote the number of

codewords in C of weight i. Clearly, $A_0 = 1$, and $A_i = 0$ for all i, $0 < i < d$, by Proposition 2.5.

Definition 2.44: The polynomial $W_C(X,Y) = \sum_{i=0}^{n} A_i X^{n-i} Y^i \in \mathbb{Z}[X,Y]$ is called the *weight enumerator* of C.

In the early 1960's, F. J. MacWilliams noticed an interesting relationship between the weight enumerator for the code C and the weight enumerator for its dual code $C^\perp$. The following identity allows us to find $W_{C^\perp}(X,Y)$ given $W_C(X,Y)$. Thus, we are able to determine the structure of the dual code $C^\perp$ without knowing the actual code $C^\perp$.

Theorem 2.45 (MacWilliams Identity): If C is an $[n,k,d]$ binary linear code with dual code $C^\perp$, then $W_{C^\perp}(X,Y) = \frac{1}{|C|} W_C(X+Y, X-Y)$. Equivalently, $\sum_{i=0}^{n} A_i' X^{n-i} Y^i = \frac{1}{|C|} \sum_{i=0}^{n} A_i (X+Y)^{n-i} (X-Y)^i$.

The proof of this theorem can be found in [12].

Example 2.46: Let C = {(0,0,0), (0,1,1), (1,0,1), (1,1,0)}. $W_C(X,Y) = \sum_{i=0}^{n} A_i X^{n-1} Y^i = X^3 + 3XY^2$. Using the MacWilliams Identity, we can find the weight enumerator for the dual code $C^\perp$. $W_{C^\perp}(X,Y) = \frac{1}{|C|} W_C(X+Y, X-Y) = \frac{1}{4}(X+Y)^3 + 3(X+Y)(X-Y)^2 = X^3 + Y^3$. Therefore $C^\perp$ has one codeword of weight 0 and one codeword of weight 3.

Using Algorithm 2.13, we find that $H = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}$ is the parity-check matrix for C. By definition, the columns of H form a basis for $C^\perp$. Thus $C^\perp$ = {(0,0,0), (1,1,1)} as expected.

Although Theorem 2.45 only applies to binary codes, there are MacWilliams Identities for codes over arbitrary alphabets. Since the codes we consider are only defined over $\mathbb{F}_2$, we will not include the MacWilliams Identity theorem for codes over $\mathbb{F}_q$. For this theorem and its proof, see [12].

Chapter 3 Nonlinear Codes

Recall that a code C over an alphabet A is a subset of A^n. Thus, such a code is not necessarily a vector space. Codes which are not vector spaces are called nonlinear codes. We have already defined length and minimum distance for an arbitrary code. However, since nonlinear codes are not vector spaces, there is no concept of dimension. Instead we use the parameter m which will denote the number of codewords in the code. Notice that $m = q^k$ for a linear code over $\mathbb{F}_q$ with dimension k.

Definition 3.1: A nonlinear code of length n with m codewords and minimum distance d is called an (n, m, d) *code.*

While a nonlinear code lacks vector space structure, it may contain more codewords than any linear code of the same length and minimum distance. For this reason, nonlinear codes are very appealing in practice.

First, we will look at three examples of nonlinear codes. The Golay code defined in Chapter 2 will be used to produce the nonlinear Nordstrom-Robinson code. We will also define the Kerdock and Preparata codes and look at the relationship between these two codes.

3.1 Nordstrom-Robinson Code

In 1967, A.W. Nordstrom and J. Robinson constructed a nonlinear double error correcting code using the extended Golay code G_{24}.

Definition 3.2: The *Nordstrom-Robinson code* N_{16} is obtained by deleting the first 8 coordinates from those codewords in G_{24} with first 8 coordinates (0,0,0,0,0,0,0,0), (1,0,0,0,0,0,0,1), (0,1,0,0,0,0,0,1), ... , or (0,0,0,0,0,0,1,1)[12].

By doing this, we get a nonlinear code that contains more codewords than any other linear code with the same length and minimum distance[12].

Theorem 3.3: The Nordstrom-Robinson code is a (16,256,6) code.

Proof: N_{16} has length 16, since codewords in N_{16} are defined by deleting the first 8 coordinates from codewords in G_{24}, a code of length 24.

By looking at the generator matrix for G_{24}, there are 2^7 possibilities for the first 7 coordinates for codewords in G_{24}. G_{24} has dimension $k = 12$ and thus 2^{12} codewords. Separating these 2^{12} codewords in G_{24} according to the first 7 coordinates, we would obtain 2^7 sets of coordinates with $\frac{2^{12}}{2^7} = 32$ codewords in each set. By definition of N_{16}, the codewords in N_{16} are formed from those codewords in G_{24} that begin with one of (0,0,0,0,0,0,0,0), (1,0,0,0,0,0,0,1), (0,1,0,0,0,0,0,1), ... , or (0,0,0,0,0,0,1,1). Thus, there are $8 \cdot 32 = 256$ codewords in N_{16}.

Let $\mathbf{x}$ and $\mathbf{y}$ be distinct codewords in N_{16}. Then there exist $\mathbf{x}'$,$\mathbf{y}' \in G_{24}$ such that $\mathbf{x}$ (respectively $\mathbf{y}$) is obtained from deleting the first 8 coordinates of $\mathbf{x}'$ (respectively $\mathbf{y}'$). Then $d(\mathbf{x}',\mathbf{y}') \geq 8$ as G_{24} has minimum distance 8. Therefore, $d(\mathbf{x},\mathbf{y}) \geq 6$ since the distance in the first 8 coordinates is no greater than two. □

Corollary 3.4: The Nordstrom-Robinson code is a double error correcting code.

Proof: Since the minimum distance is 6, the Nordstrom-Robinson code will correct $\lfloor \frac{d-1}{2} \rfloor = \lfloor \frac{6-1}{2} \rfloor = 2$ errors by Corollary 1.12. □

The Nordstrom-Robinson code gives an example for the most important advantage to using nonlinear codes. The best known linear code that corrects 2 errors and has length 16 contains 128 codewords[4]. Because of the lack of vector space structure, nonlinear codes contain more codewords while maintaining the same minimum distance between words.

This lack of structure that enables a code to contain more codewords is also the biggest disadvantage in using nonlinear codes. Since nonlinear codes have no algebraic structure, there is no easy encoding and decoding scheme for these types of codes. This is a barrier to using nonlinear codes in applications.

3.2 Kerdock Codes

The Kerdock codes were defined by A.M. Kerdock in 1972[10]. Since then, several equivalent definitions arose. See references [2] and [11].

To define the Kerdock code, we need to use the trace map.

Definition 3.5: The *trace map* $Tr : \mathbb{F}_{2^m} \to \mathbb{F}_2$ is defined by

$Tr(\Pi) = \Pi + \Pi^2 + \Pi^4 + ... + \Pi^{2^{m-1}}$ for any $\Pi \in \mathbb{F}_{2^m}$.

Definition 3.6: Let $m \geq 3$, where m is an odd integer. Let $h(x) \in \mathbb{Z}_2[x]$ be a primitive polynomial of degree m and ξ be a root of $h(x)$ in some splitting field. The *Kerdock code* K_{m+1} of length 2^{m+1} is the binary code which consists RM$(1, m+1)$ together with the $2^m - 1$ elements of the coset space RM$(2, m+1)$/RM$(1, m+1)$ with coset representatives

$$\left(L_\Pi\left(\xi^\infty\right), L_\Pi\left(\xi^0\right), ..., L_\Pi\left(\xi^{n-1}\right), R_\Pi\left(\xi^\infty\right), R_\Pi\left(\xi^0\right), ..., R_\Pi\left(\xi^{n-1}\right)\right)$$

where $L_\Pi(\xi^j) = \sum_{i=1}^{(m-1)/2} Tr(\Pi\xi^j)^{1+2^j}$ and $R_\Pi(\xi^j) = L_\Pi(\xi^j) + Tr(\Pi\xi^j)$ for j $\in \{0,1,...,2^m - 1\}$ and Π runs through $\mathbb{F}_{2^m}^*$. See [12] and [16].

In other words, the Kerdock code $K_{m+1} = \{\mathbf{u} + \text{RM}(1, m+1) \mid \mathbf{u}$ is a coset representative in the form $\left(L_\Pi\left(\xi^\infty\right), L_\Pi\left(\xi^0\right), ..., L_\Pi\left(\xi^{n-1}\right), R_\Pi\left(\xi^\infty\right), R_\Pi\left(\xi^0\right), ..., R_\Pi\left(\xi^{n-1}\right)\right)\}$. Notice that RM$(1, m+1) \subset K_{m+1} \subset$ RM$(2, m+1)$.

Proposition 3.7: The Kerdock code K_{m+1} contains $2^{2(m+1)}$ codewords and has minimum distance $2^m - 2^{(m-1)/2}$.

Proof: RM$(1, m+1)$ contains 2^k codewords, where $k = \binom{m+1}{0} + \binom{m+1}{1} = m+2$. There are 2^m cosets (including the (0,...,0) coset) of RM$(2, m+1)$. Thus there are $2^m(2^{m+2}) = 2^{2m+2}$ codewords in K_{m+1}.

For a proof that K_{m+1} has minimum distance $2^m - 2^{(m-1)/2}$, see [12]. □

Example 3.8: Let $m = 3$. The primitive polynomial of degree 3 is $h(x) = x^3 + x + 1$. Then $\xi^3 + \xi + 1 = 0$. Thus $\mathbb{F}_8^* = \{1, \xi, \xi^2, 1+\xi, \xi+\xi^2, 1+\xi+\xi^2, 1+\xi^2\}$. The Kerdock Code K_4 of length $n = 2^{m+1} = 16$ is the binary code which consists of RM(1,4) together with

$2^m - 1 = 7$ cosets of RM(2,4) with coset representatives

(0, 1, 0, 0, 1, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0)
(0, 0, 0, 1, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0)
(0, 0, 1, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0)
(0, 1, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0)
(0, 0, 1, 1, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0)
(0, 1, 1, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0)
(0, 1, 1, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0).

3.3 Preparata Codes

In 1968, the Preparata Codes were introduced by F.P. Preparata. An equivalent definition was given by R.D Baker and R.M. Wilson in the early 1980's. We will use the definition of Baker and Wilson[1].

Definition 3.9: Let m be an odd integer such that $m \geq 3$. The *Preparata code* P_{m+1} of length 2^{m+1} consists of all codewords **(x,y)**, where $\mathbf{x} = (x_0, x_\xi, x_{\xi^2}, ..., x_{\xi^{2^m-1}})$, $\mathbf{y} = (y_0, y_\xi, y_{\xi^2}, ..., y_{\xi^{2^m-1}}) \in F_2^{2^m}$, satisfying the following properties:

i. wt(**x**) and wt(**y**) are even

ii. $\sum_{x_\alpha=1} \alpha = \sum_{y_\alpha=1} \alpha$

iii. $\sum_{x_\alpha=1} \alpha^3 + \left(\sum_{x_\alpha=1} \alpha\right)^3 = \sum_{y_\alpha=1} \alpha^3$.

Proposition 3.10: The Preparata code, P_{m+1}, has minimum distance 6 and contains $2^{2^{m+1}-2m-2}$ codewords.

Proof: See [12]. □

Since the smallest Preparata code P_4 consists of 256 codewords, we will not list all codewords of a Preparata code. However, we will give an example of a codeword in P_4 and a word that is not in P_4.

Example 3.11: Consider the code P_4, where $m = 3$. Let **x** = (1,1,1,1,0,0,0,0) and **y** = (0,0,1,0,0,0,0,1). Since wt(**x**) = 4 and wt(**y**) = 2, condition (i) is satisfied. Now,

$\sum_{x_\alpha=1} \alpha = 0+\xi+\xi^2+\xi^3 = \xi^6$ and $\sum_{y_\alpha=1} \alpha = \xi^2+\xi^7 = \xi^6$ satisfies condition (ii). Since $\sum_{x_\alpha=1} \alpha^3 + \left(\sum_{x_\alpha=1} \alpha\right)^3 = \xi^3+\xi^6+\xi^9+\xi^{18} = \xi^2$ and $\sum_{y_\alpha=1} \alpha^3 = (\xi^2)^3+(\xi^7)^3 = 1+\xi^6 = \xi^2$ condition (iii) is satisfied. Thus $(1,1,1,1,0,0,0,0,0,0,1,0,0,0,0,1) \in P_4$.

Now let $\mathbf{x}$ = (1,0,1,0,1,0,1,0) and $\mathbf{y}$ = (1,1,1,0,0,0,0,1). Although condition (i) is satisfied with wt($\mathbf{x}$) = wt($\mathbf{y}$) = 4, $\sum_{x_\alpha=1} \alpha^3 + \left(\sum_{x_\alpha=1} \alpha\right)^3 = \xi^6 + \xi^{12} + \xi^{15} + \xi^{18} = \xi^4$ and $\sum_{y_\alpha=1} \alpha^3 = 1 + \xi^3 + \xi^6 = \xi^5$ imply that condition (iii) is not satisfied. Thus $(1,0,1,0,1,0,1,0,1,1,1,0,0,0,0,1) \notin P_4$.

Although for nonlinear codes there is no formal dual code, the Preparata code acts in some sense like a dual to the Kerdock code. In fact, the weight distributions of P_{m+1} and K_{m+1} are related by the MacWilliams Identity just as the weight distribution of a linear code and its dual[12]. That is, $W_{P_{m+1}}(X,Y) = \frac{1}{|K_{m+1}|} W_{K_{m+1}}(X+Y, X-Y)$. This fascinating relationship was explained by two separate groups of coding theorists in the early 1990's with the concept of $\mathbb{Z}_4$-linear codes[8]. The theory of $\mathbb{Z}_4$-linear codes will be the focus of Chapters 4 and 5.

Chapter 4 Quaternary Codes

In this chapter, we will consider codes in the alphabet $\mathbb{Z}_4$. Recall that a code of length n in alphabet $\mathbb{Z}_4$ is a subset of $\mathbb{Z}_4^n = \{(x_1, ..., x_n) \mid x_i \in \mathbb{Z}_4\}$. We will be interested in codes with additional structure, called quaternary codes.

Definition 4.1: Let C be a code of length n over the alphabet $\mathbb{Z}_4$. The code C is called a *quaternary code* if C is a subgroup of $\mathbb{Z}_4^n$.

A quaternary code is not linear in the sense of Chapter 2. Quaternary codes are not vector spaces due to the fact that $\mathbb{Z}_4$ is not a field. However, they do share some of the nice properties of linear codes. For example, in a quaternary code, the sum of any two codewords and any scalar multiple of a codeword is a codeword.

Since quaternary codes are not vector spaces, we do not have the concept of dimension. However, type is similar to dimension in that it describes the number of codewords in a code.

Definition 4.2: Let p be a prime. A finite abelian p-group G is said to be of *type* $(p^{r_1}, ..., p^{r_s})$ if G is isomorphic to the product of cyclic groups of orders p^{r_i} where $i = 1, .., s$.

The additive abelian group $\mathbb{Z}_4^n$ is of type $(2^2)^n$ since it is a direct sum of n cyclic subgroups of order 2^2. Since a quaternary code of length n is a subgroup of $\mathbb{Z}_4^n$, it has type $4^{k_1}2^{k_2}$ for some k_1 and k_2.

Another nice property of quaternary codes is that such codes can be described by a generator matrix.

Definition 4.3: A $k \times n$ matrix M with entries in $\mathbb{Z}_4$ is said to be a *generator matrix* for a quaternary code C if each codeword of C is a linear combination of the rows of M and no proper subset of the rows of G generates C.

For words $\mathbf{x} = (x_1, ..., x_n)$,$\mathbf{y} = (y_1, ..., y_n) \in \mathbb{Z}_4^n$, the componentwise product of $\mathbf{x}$ and $\mathbf{y}$ is defined to be the word $\mathbf{x}*\mathbf{y} = (x_1y_1, ..., x_ny_n)$.

Definition 4.4: Two quaternary codes are said to be *equivalent* if one code can be obtained from the other by a permutation of coordinates and componentwise product

by a vector. Two quaternary codes that differ only by a permutation of coordinates are said to be *permutation-equivalent.*

Clearly, quaternary codes that are permutation-equivalent are certainly equivalent codes.

Example 4.5: Let C_1 = {(0,0,0,0), (1,1,1,1), (2,2,2,2), (3,3,3,3), (1,3,1,3), (3,1,3,1), (0,2,0,2), (2,0,2,0)}. By permuting the coordinates x_0 and x_1, we get a permutation-equivalent code C_2 = {(0,0,0,0), (1,1,1,1), (2,2,2,2), (3,3,3,3), (3,1,1,3), (1,3,3,1), (2,0,0,2), (0,2,2,0)}. If we then multiply C_2 by (1,1,-1,1), we get an equivalent code C_3={(0,0,0,0), (1,1,3,1), (2,2,2,2), (3,3,1,3), (3,1,3,3), (1,3,1,1), (2,0,0,2), (0,2,2,0)}.

Notice that componentwise multiplication by (1,1,-1,1) changes the sign of the x_2 coordinate.

The following theorem is analogous to Definition 2.3. A quaternary code with a generator matrix as in following theorem has type $4^{k_1}2^{k_2}$ and contains $4^{k_1}2^{k_2} = 2^{2k_1+k_2}$ codewords, where $k_1, k_2 \in \mathbb{N}$.

Theorem 4.6: Any nonzero quaternary code C is permutation-equivalent to a quaternary code with a generator matrix of the form

$$M = \begin{bmatrix} I_{k_1} & A & B \\ 0 & 2I_{k_2} & 2D \end{bmatrix}$$

where I_{k_1} and I_{k_2} denote the $k_1 \times k_1$ and $k_2 \times k_2$ identity matrices, respectively, A and D are matrices with entries in $\mathbb{Z}_2$, and B is a matrix with entries in $\mathbb{Z}_4$[8].

Note that if C is of length n, A is a $k_1 \times k_2$ matrix , B is a $k_1 \times (n - k_1 - k_2)$ matrix, and D is a $k_2 \times (n - k_1 - k_2)$ matrix.

Proof: Let C be a nonzero quaternary code of length n. Since C is a subgroup of $\mathbb{Z}_4^n$, all elements of C have order 2 or 4. That is, each codeword has order 2 or 4. We induct on the code length n. We distinguish two cases depending on whether or not C has a codeword of order 4.

First, assume there is a codeword of order 4 in C. After permuting the coordinates of the codewords and (if necessary) multiplying the codeword by -1, we can assume that

the codeword of order 4 is of the form $(1,c_2,...,c_n)$ where $c_i \in \mathbb{Z}_4$. Let $C' = \{(0,x_2,...,x_n) \in C\}$. Clearly, C' is also a quaternary code and can be regarded as a code of length $n-1$ by deleting the first coordinate of each codeword. By the induction hypothesis, C' has a generator matrix of the form

$$\begin{bmatrix} 0 & I_{k_1-1} & A_1 & B_1 \\ 0 & 0 & 2I_{k_2} & 2D \end{bmatrix},$$

where A_1 and D are matrices with entries in $\mathbb{Z}_2$ and B_1 is a matrix with entries in $\mathbb{Z}_4$. Then C has a generator matrix of the form

$$\begin{bmatrix} 1 & c_2,...,c_{k_1} & c_{k_1+1},...,c_{k_1+k_2} & c_{k_1+k_2+1},...,c_n \\ 0 & I_{k_1-1} & A_1 & B_1 \\ 0 & 0 & 2I_{k_2} & 2D \end{bmatrix}.$$

Row reducing the first k_1 rows will result in $\begin{bmatrix} I_{k_1} \\ 0 \end{bmatrix}$ for the first k_1 columns. By adding certain linear combinations of the last k_2 rows we obtain $[c'_{k_1+1},...,c'_{k_1+k_2}]$ such that $c'_i \in \mathbb{Z}_2$. Thus, we get a matrix in the form

$$\begin{bmatrix} I_{k_1} & A & B \\ 0 & 2I_{k_2} & 2D \end{bmatrix}.$$

This completes the proof in the case that C contains a codeword of order 4.

Now suppose that C has no codewords of order 4. Then all nonzero codewords in C are of order 2. Since $C \neq \{\mathbf{0}\}$, there is a codeword of order 2 in C. We can assume that this codeword is of the form $(2, 2c_2,..., 2c_n)$ where $c_i \in \mathbb{Z}_4$. As done previously, define $C' = \{(0,x_2,...,x_n) \in C\}$. Then C' is also a quaternary code without codewords of order 4. Thus the type of C' is $4^0 2^{k_2}$ which implies that $k_1 = 0$. C' can be regarded as a code of length $n-1$. By the induction hypothesis, C' has a generator matrix of the form

$$\begin{bmatrix} 0 & 2I_{k_2-1} & 2D_1 \end{bmatrix},$$

where D_1 is a $\mathbb{Z}_2$ matrix. Then C has a generator matrix of the form

$$\begin{bmatrix} 2 & 2c_2, ..., 2c_{k_2} & 2c_{k_2+1}, ..., 2c_n \\ 0 & 2I_{k_2-1} & 2D_1 \end{bmatrix}.$$

After adding a certain linear combination of the last $k_2 - 1$ rows of the above matrix to the first row, we obtain a matrix of the form

$$\begin{bmatrix} 2I_{k_2} & 2D \end{bmatrix},$$

which is a matrix of the form

$$\begin{bmatrix} I_{k_1} & A & B \\ 0 & 2I_{k_2} & 2D \end{bmatrix}$$

with $k_1 = 0$. See [16]. □

This allows for encoding of message words via matrix multiplication as is the case with linear codes. Let C be a quaternary code with generator matrix M. Thus, the quaternary code C is C = {**u**M: $u_i \in \mathbb{Z}_4$ for $1 \leq i \leq k_1$ and $u_i \in \mathbb{Z}_2$ for $k_1 + 1 \leq i \leq k_1 + k_2$}.

The inner product of two words $\mathbf{x} = (x_1, ..., x_n)$ and $\mathbf{y} = (y_1, ..., y_n)$ in $\mathbb{Z}_4^n$ is $\mathbf{x} \cdot \mathbf{y} = x_1y_1 + ... + x_ny_n$. If $x_1y_1 + ... + x_ny_n = 0$, then **x** and **y** are orthogonal.

Definition 4.7: Given a quaternary code C of length n, let $C^{\perp} := \{ \mathbf{x} \in \mathbb{Z}_4^n \mid \mathbf{x} \cdot \mathbf{y} = 0 \text{ for all } \mathbf{y} \in C \}$. $C^{\perp}$ is called the *dual code* of C.

Proposition 4.8: $C^{\perp}$ is a subgroup of $\mathbb{Z}_4^n$, and thus is a quaternary code.

Proof: This is immediate from Definition 4.1 and the definition of inner product. □

Theorem 4.9: Let C be a quaternary code of length n with generator matrix M as in Theorem 4.6. The dual code $C^{\perp}$ has generator matrix

$$M^{\perp} = \begin{bmatrix} -B^t - D^tA^t & D^t & I_{n-k_1-k_2} \\ 2A^t & 2I_{k_2} & 0 \end{bmatrix}.$$

See [8].

Proof: Consider the quaternary code C′ defined by the generator matrix $M^{\perp}$ above. We claim that C′ = $C^{\perp}$. Since $M(M^{\perp})^t = 0$, we have that C′ ⊂ $C^{\perp}$.

Let $\mathbf{c} = (c_1, ..., c_n) \in C^{\perp}$. We can choose a linear combination of the first $n-k_1-k_2$ rows of $M^{\perp}$ to $\mathbf{c}$ to obtain a codeword of $C^{\perp}$, which is of the form $\mathbf{c}' = (c_1, ..., c_{k_1}, c_{k_1+1}, ..., c_{k_1+k_2}, 0, ..., 0)$. Since $\mathbf{c}'$ is orthogonal to the last k_2 rows of M, each $c_{k_1+1}, ..., c_{k_1+k_2}$ is 0 or 2. We can also choose a linear combination of the last k_2 rows of $M^{\perp}$ to $\mathbf{c}'$ to obtain a codeword of $C^{\perp}$, which is of the form $\mathbf{c}'' = (c_1, ..., c_k, 0, ..., 0, 0)$. Since $\mathbf{c}''$ is orthogonal to the first k_1 rows of M, $c_1 = \cdots = c_k = 0$. Therefore $\mathbf{c} \in C'$. Thus C′ = $C^{\perp}$ with generator matrix $M^{\perp}$. See [16]. □

Definition 4.10: Let C be a quaternary code with generator matrix M. Then $(M^{\perp})^t$ is called the *parity-check matrix* for C, where $M^{\perp}$ is the generator matrix for the dual code of C.

For $3 \in \mathbb{Z}_4$, notice that 3 ≡ -1(mod 4). Therefore, the distance of 3 from 0 is the same as the distance of 1 from 0. We take this property into account and define weight and distance for quaternary codes. For equivalent codes, we do not distinguish between 1 and 3 since multiplication by -1 is allowed.

Definition 4.11: The *Lee weights*, wt_L, of 0, 1, 2, 3 $\in \mathbb{Z}_4^n$ are defined to be 0, 1, 2, 1, respectively. The Lee weight of $\mathbf{x} = (x_1, ..., x_n) \in \mathbb{Z}_4^n$ is $wt_L(\mathbf{x}) = \sum_{i=0}^{n} wt_L(a_i)$. The weight of $\mathbf{x}$ at $a \in \mathbb{Z}_4$ is defined to be $w_a(\mathbf{x}) = |\{i \mid x_i = a\}|$. See [8] .

Definition 4.12: The *Lee distance* between $\mathbf{x},\mathbf{y} \in \mathbb{Z}_4^n$ is $d_L(\mathbf{x},\mathbf{y}) = wt_L(\mathbf{x}-\mathbf{y})$. The minimum Lee distance for quaternary code C is defined to be $d_L = \min\{d_L(\mathbf{x},\mathbf{y}) \mid \mathbf{x},\mathbf{y} \in \mathbb{Z}_4^n$ and $\mathbf{x} \neq \mathbf{y}\}$[8].

Recall Proposition 2.5. The relationship between Lee distance and Lee weight of a quaternary code is similar to that of distance and weight of a linear code.

Proposition 4.13: $wt_L(\mathbf{x})=w_1(\mathbf{x})+2w_2(\mathbf{x})+w_3(\mathbf{x})$ for all $\mathbf{x} \in \mathbb{Z}_4^n$[16].

Proof: Let $x_i = 0$. Then $wt_L(0) = w_1(0)+2w_2(0)+w_3(0) = 0$. For $x_i = 1$. Then $wt_L(1) = w_1(1)+2w_2(1)+w_3(1) = 1$. Now let $x_i = 2$. Then $wt_L(2) = w_1(2)+2w_2(2)+w_3(2) = 2$. Let $x_i = 3$. Then $wt_L(3) = w_1(3)+2w_2(3)+w_3(3) = 1$. Extend this to $\mathbb{Z}_4^n$. □

4.1 MacWilliams Identity for Quaternary Codes

As with codes over alphabet $\mathbb{F}_q$, it is useful to describe the structure of a quaternary code by a weight enumerator. The complete weight enumerator classifies codewords $\mathbf{c} \in C$ according to the number of times each element $a \in \mathbb{Z}_4$ appears in $\mathbf{c}$.

Definition 4.14: Let C be a quaternary code. The *complete weight enumerator*, *cwe*, of C is

$$cwe_C(X_0, X_1, X_2, X_3) = \sum_{\mathbf{c} \in C} X_0^{w_0(\mathbf{c})} X_1^{w_1(\mathbf{c})} X_2^{w_2(\mathbf{c})} X_3^{w_3(\mathbf{c})} \in \mathbb{Z}[X_0, X_1, X_2, X_3].[8]$$

Example 4.15: Let C = {(0,0), (2,2)}. Then $w_0((0,0)) = |\{i \mid x_i = 0\}| = 2$, $w_1((0,0)) = |\{i \mid x_i = 1\}| = 0$, $w_2((0,0)) = |\{i \mid x_i = 2\}| = 0$, $w_3((0,0)) = |\{i \mid x_i = 3\}| = 0$, $w_0((2,2)) = |\{i \mid x_i = 0\}| = 0$, $w_1((2,2)) = |\{i \mid x_i = 1\}| = 0$, $w_2((2,2)) = |\{i \mid x_i = 2\}| = 2$, $w_3((2,2)) = |\{i \mid x_i = 3\}| = 0$. Thus $cwe_C(X_0, X_1, X_2, X_3) = \sum_{\mathbf{c} \in C} X_0^{w_0(\mathbf{c})} X_1^{w_1(\mathbf{c})} X_2^{w_2(\mathbf{c})} X_3^{w_3(\mathbf{c})} = X_0^2 + X_2^2$.

The following theorem gives a MacWilliams Identity for quaternary codes.

Theorem 4.16: Let C be a quaternary code. Then $cwe_{C^\perp}(X_0, X_1, X_2, X_3) =$

$$\frac{1}{|C|} cwe_C(X_0 + X_1 + X_2 + X_3, X_0 + iX_1 - X_2 - iX_3, X_0 - X_1 + X_2 - X_3, X_0 - iX_1 - X_2 + iX_3).$$

We omit the proof, which can be found in [8].

Example 4.17: Let C = {(0,0), (2,2)}. Then by the MacWilliams Identity for quaternary codes, $cwe_{C^\perp}(X_0, X_1, X_2, X_3)$

$= \frac{1}{2}\left[(X_0 + X_1 + X_2 + X_3)^2 + (X_0 - X_1 + X_2 - X_3)^2\right]$

$= X_0^2 + X_1^2 + X_2^2 + X_3^2 + 2X_0X_2 + 2X_1X_3$

is the complete weight enumerator for the dual code $C^\perp$. Note that this classifies all codewords in $C^\perp$ = {(0,0), (1,1), (2,2),(0,2), (2,0), (1,3), (3,1)}.

Permutation-equivalent codes have the same complete weight enumerator. Recall that permutation-equivalent is a stronger property than equivalence. As the next example shows, equivalent codes may have different complete weight enumerators.

Example 4.18: C_1 = {(0,0,0,0), (1,1,1,1), (2,2,2,2), (3,3,3,3), (1,3,1,3), (3,1,3,1), (0,2,0,2), (2,0,2,0)} and C_2 = {(0,0,0,0), (1,1,1,1), (2,2,2,2), (3,3,3,3), (3,1,1,3), (1,3,3,1), (2,0,0,2),

(0,2,2,0)} are permutation-equivalent. $cwe_{C_1}(X_0, X_1, X_2, X_3) = X_0^4 + X_1^4 + X_2^4 + X_3^4 + 2X_1^2X_3^2 + 2X_0^2X_2^2 = cwe_{C_2}(X_0, X_1, X_2, X_3)$. C_1 is equivalent to C_3 = {(0,0,0,0), (1,1,3,1), (2,2,2,2), (3,3,1,3), (3,1,3,3), (1,3,1,1), (2,0,0,2), (0,2,2,0)} which has

$cwe_{C_3}(X_0, X_1, X_2, X_3) = X_0^4 + X_2^4 + 2X_1^3X_3 + 2X_1X_3^3 + 2X_0^2X_2^2 \neq cwe_{C_1}(X_0, X_1, X_2, X_3)$.

Since for equivalent codes we identify 1 with 3, we want a weight enumerator that also identifies 1 with 3.

Definition 4.19: Let C be a quaternary code. The *symmetrized weight enumerator*, *swe*, is defined as $swe_C(X_0, X_1, X_2) = cwe_C(X_0, X_1, X_2, X_1)$; that is, $swe_C(X_0, X_1, X_2, X_3)$ is obtained by identifying 1 with 3 by setting $X_1 = X_3$ in the complete weight enumerator[8].

Example 4.20: Let C_1 and C_2 be as in Example 4.18. $swe_{C_1}(X_0, X_1, X_2) = X_0^4 + 4X_1^4 + X_2^4 + 2X_0^2X_2^2 = swe_{C_3}(X_0, X_1, X_2)$.

Note that the symmetrized weight enumerator does not give as much information about the codewords in C_2 as does the complete weight enumerator. Equivalent codes have the same symmetrized weight enumerator since the symmetrized weight enumerator does not distinguish between 1 and 3.

Definition 4.21: Let C be a quaternary code. The *Lee weight enumerator* is defined as

$$Lee_C(X, Y) = \sum_{c \in C} X^{2n - w_L(c)} Y^{w_L(c)}[8].$$

The following gives a MacWilliams identity for the Lee weight enumerator.

Theorem 4.22: For a quaternary code C,

$$Lee_{C^\perp}(X, Y) = \frac{1}{|C|} Lee_C(X + Y, X - Y)[8].$$

Proof: By using Definition 4.17 and 4.12, we find that $swe_C(X^2, XY, Y^2)$ $= cwe_C(X^2, XY, Y^2, XY) = \sum_{c \in C} (X^2)^{w_0(c)} (XY)^{w_1(c) + w_3(c)} (Y^2)^{w_2(c)}$. It follows by Proposition 4.11 that $\sum_{c \in C} (X^2)^{w_0(c)} (XY)^{w_1(c) + w_3(c)} (Y^2)^{w_2(c)}$

$= \sum_{c \in C} X^{2w_0(c)+w_1(c)+w_3(c)} Y^{w_1(c)+w_3(c)+2w_2(c)} = \sum_{c \in C} X^{2n-w_L(c)} Y^{w_L(c)} = Lee_C(X,Y)$.

Thus, since $swe_{C^\perp}(X^2, XY, Y^2) = cwe_{C^\perp}(X^2, XY, Y^2, XY)$

$= \frac{1}{|C|} cwe_C(X^2+2XY+Y^2, X^2-Y^2, X^2-2XY+Y^2, X^2-Y^2)$, we have that

$Lee_{C^\perp}(X,Y) = swe_{C^\perp}(X^2, XY, Y^2) = \frac{1}{|C|} swe_C(X^2+2XY+Y^2, X^2-Y^2, X^2-2XY+Y^2)$

$= \frac{1}{|C|} swe_C\left[(X+Y)^2, (X+Y)(X-Y), (X-Y)^2\right] = \frac{1}{|C|} Lee_C(X+Y, X-Y)$. See [16]. □

4.2 Quaternary Cyclic Codes

Definition 4.23: A quaternary code C of length n is *cyclic* if

$(c_0, c_1, ..., c_{n-1}) \in C \Rightarrow (c_{n-1}, c_0, c_1, ..., c_{n-2}) \in C$.

Given a word $\mathbf{v} \in \mathbb{Z}_4^n$, we define a polynomial in $\mathbb{Z}_4[x]$ with coefficients coming from $\mathbf{v}$. There is a one-to-one correspondence between words in $\mathbb{Z}_4^n$ and polynomials of degree less than n in $\mathbb{Z}_4[x]$ defined by $v = (v_0, v_1, ..., v_{n-1}) \longleftrightarrow v(x) = v_0 + v_1 x + ... + v_{n-1}x^{n-1}$. Thus, we identify a codeword in a quaternary code with a polynomial.

Definition 4.24: A quaternary code C of length n is *cyclic* if $c(x) \in C \Rightarrow xc(x) \bmod (x^n - 1) \in C$.

Definition 4.25: Let C be a quaternary cyclic code of length n. $g(x)$ is called the *generator polynomial* of C if $g(x)$ is a monic polynomial dividing $x^n - 1$ such that $C = \langle g(x) \rangle$.

As in the linear case, we know that a polynomial $g(x)$ such that $C = \langle g(x) \rangle$ exists since C is a principal ideal of $\mathbb{Z}_4$. Any codeword $c(x) \in C$ can be written as $c(x) = f(x)g(x)$ where $f(x) \in \mathbb{Z}_4[x]$.

We can see that $\{g(x), xg(x), ..., x^{k-1}g(x)\}$ forms a basis for the cyclic code of length n, where $k = n -$ deg$(g(x))$. Therefore $\begin{bmatrix} g(x) \\ xg(x) \\ \vdots \\ x^{k-1}g(x) \end{bmatrix}$ is a generator matrix for the quaternary cyclic code $C = \langle g(x) \rangle$.

Since $g(x)$ divides $x^n - 1$, there exists an $h(x) \in \mathbb{Z}_4[x]$ such that $h(x)g(x) \equiv 0 \bmod (x^n - 1)$.

Definition 4.26: Let C be a quaternary cyclic code of length n such that $C = \langle g(x) \rangle$. Then $h(x) = \frac{x^n-1}{g(x)} \in \mathbb{Z}_4[x]$ is called the *check-polynomial*.

Let $h(x)$ be a check-polynomial for a quaternary cyclic code C. Then $c(x)h(x) = 0$ for all $c(x) \in$ C.

4.3 Quaternary Kerdock Codes

In this section we will need the following map from $\mathbb{Z}_4[x]$ to $\mathbb{Z}_2[x]$. Let $\psi : \mathbb{Z}_4[x] \longrightarrow \mathbb{Z}_2[x]$ be the map defined by $a_0 + a_1x + ... + a_nx^n \longmapsto \overline{a_0} + \overline{a_1}x + ... + \overline{a_n}x^n$ where $\overline{0} = \overline{2} = 0$ and $\overline{1} = \overline{3} = 1$.

Definition 4.27: Let $m \geq 2$ be an integer and $h(x) \in \mathbb{Z}_4[x]$ be a monic polynomial of degree m. Then $h(x)$ is a *basic primitive polynomial* over $\mathbb{Z}_4[x]$ if $\psi(h(x))$ is a primitive polynomial over $\mathbb{Z}_2[x]$.

Definition 4.28: Let $h(x)$ be a monic basic primitive polynomial of degree $m \geq 2$ over $\mathbb{Z}_4[x]$ such that $h(x) \mid (x^{2^m-1} - 1)$ and $g(x) = x^{2^m-m-2}\widetilde{g}\left(\frac{1}{x}\right)$, where $\widetilde{g}(x) = \frac{(x^{2^m-1}-1)}{(x-1)h(x)}$. The *shortened quaternary Kerdock code* $K(m)^-$ is the quaternary cyclic code of length n with generator polynomial $g(x)$. The coordinates of codewords of $K(m)^-$ are numbered as $0, 1, ..., n-1$. The *quaternary Kerdock code* $K(m)$ is the code obtained from $K(m)^-$ by adding a zero-sum check symbol to each codeword of $K(m)^-$ at the position ∞, which is situated in front of the position 0. See [16].

Proposition 4.29: Let $\delta = 2^m - m - 2$, $g(x) = g_0 + g_1x + ... + g_\delta x^\delta \in \mathbb{Z}_4[x]$, and $g_\infty = -(g_0 + g_1 + ... + g_\delta) \in \mathbb{Z}_4$. Then the $(m+1) \times 2^m$ matrix

$$\begin{bmatrix} g_\infty & g_0 & g_1 & \cdots & g_\delta & & \\ g_\infty & & g_0 & g_1 & \cdots & g_\delta & \\ \vdots & & & \ddots & & & \ddots \\ g_\infty & & & & g_0 & g_1 & \cdots & g_\delta \end{bmatrix}$$

is a generator matrix for $K(m)$[8].

Example 4.30: Let $m = 3$. A basic primitive polynomial of degree 3 over $\mathbb{Z}_4$ is $h(x) = x^3+2x^2+x+3$. Then $\widetilde{g}(x) = \frac{(x^7-1)}{(x-1)h(x)} = x^3+3x^2+2x+3$ implies that $g(x) = 3x^3+2x^2+3x+1$ is a generator polynomial for $K(3)$ with generator matrix

$$\begin{bmatrix} 1 & 3 & 2 & 3 & 1 & 0 & 0 & 0 \\ 1 & 0 & 3 & 2 & 3 & 1 & 0 & 0 \\ 1 & 0 & 0 & 3 & 2 & 3 & 1 & 0 \\ 1 & 0 & 0 & 0 & 3 & 2 & 3 & 1 \end{bmatrix}.$$

Notice that if we change the signs of the all coordinates, we will get an equivalent code with generator polynomial $g'(x) = x^3 + 2x^2 + x + 3$ and generator matrix

$$\begin{bmatrix} 1 & 3 & 1 & 2 & 1 & 0 & 0 & 0 \\ 1 & 0 & 3 & 1 & 2 & 1 & 0 & 0 \\ 1 & 0 & 0 & 3 & 1 & 2 & 1 & 0 \\ 1 & 0 & 0 & 0 & 3 & 1 & 2 & 1 \end{bmatrix}.$$

In Chapter 3, we defined the trace map, Tr, from $\mathbb{F}_{2^m}$ to $\mathbb{F}_2$. Here we define a similar map for the Galois ring $\mathbb{Z}_4[x]/(h(x))$, where $h(x) \in \mathbb{Z}_4[x]$ is a basic irreducible polynomial of degree m.

Definition 4.31: The *relative trace map* $T : \mathbb{Z}_4/(h(x)) \to \mathbb{Z}_4$ is defined by

$$T(\xi) \longmapsto \xi + \xi^2 + \xi^4 + ... + \xi^{2^{m-1}}.$$

It is easy to verify that the relative trace map is a ring homomorphism since $T(c + c') = T(c) + T(c')$ and $T(cc') = T(c)T(c')$ for all $c, c' \in \mathbb{Z}_4/(h(x))$.

Proposition 4.32: Let T be the relative trace map and Tr be the trace map. Then $\psi \circ T = Tr \circ \psi$[8].

Proposition 4.33: The trace description over the ring $\mathbb{Z}_4$ of the quaternary Kerdock code $K(m)$ is $K(m) = \{\varepsilon 1^{n+1} + \mathbf{u}^\lambda \mid \varepsilon \in \mathbb{Z}_4, \lambda \in \mathbb{Z}_4[\xi]\}$, where 1^{n+1} is the $(n+1)$-tuple, $\mathbf{u}^\lambda = (T(\lambda\xi^\infty), T(\lambda\xi^0), ..., T(\lambda\xi^{n-1}))$, and ξ is a root of the basic primitive polynomial $h(x) \in \mathbb{Z}_4[x]$.

Proof: See [8] and [16]. □

4.4 Quaternary Preparata Codes

Using the same notation as for Kerdock codes, let $h(x)$ be a basic primitive polynomial of degree $m \geq 2$ such that $h(x) \mid (x^{2^m-1} - 1)$.

Definition 4.34: The quaternary cyclic code of length $n = 2^m - 1$ with generator polynomial $h(x)$ is called the *shortened quaternary Preparata code* $P(m)^-$. The quaternary

code obtained from $P(m)^-$ by adding a zero-sum check symbol to each codeword of $P(m)^-$ is called the *quaternary Preparata code* $P(m)$. See [16].

Proposition 4.35: Let $h(x) = h_0 + h_1x + ... + h_mx^m$ be the basic primitive polynomial of degree m, where m is an integer ≥ 2. Let $h_\infty = -h(1) = \pm 1$. Then the following $(2^m - m - 1) \times 2^m$ matrix is a generator matrix for $P(m)$

$$\begin{bmatrix} h_\infty & h_0 & h_1 \cdots & h_m & & & \\ h_\infty & & h_0 & h_1 \cdots & h_m & & \\ \vdots & & & \ddots & & \ddots & \\ h_\infty & & & & h_0 & h_1 \cdots & h_m \end{bmatrix}$$ [8].

Example 4.36: Let $m = 3$. The basic primitive polynomial of degree 3 is $h(x) = x^3 + 2x^2 + x - 1$. Now $-h(1) = -3 = 1$ implies that $h_\infty = 1$. Thus

$$\begin{bmatrix} 1 & 3 & 1 & 2 & 1 & 0 & 0 & 0 \\ 1 & 0 & 3 & 1 & 2 & 1 & 0 & 0 \\ 1 & 0 & 0 & 3 & 1 & 2 & 1 & 0 \\ 1 & 0 & 0 & 0 & 3 & 1 & 2 & 1 \end{bmatrix}$$ is a generator matrix for the Preparata code $P(3)$.

Notice that $g'(x) = h(x)$ from Example 4.30. This may give rise to the question of whether the quaternary Kerdock and quaternary Preparata codes have any relationship.

Theorem 4.37: The quaternary Preparata code $P(m)$ is the dual code of the quaternary Kerdock code $K(m)$.

Proof: See [16]. □

Thus, the quaternary Preparata code and quaternary Kerdock code satisfy the MacWilliams identity for quaternary codes. In other words, $cwe_{P(m)}(X_0, X_1, X_2, X_3) = \frac{1}{|K(m)|} cwe_{K(m)}(X_0 + X_1 + X_2 + X_3, X_0 + iX_1 - X_2 - iX_3, X_0 - X_1 + X_2 - X_3, X_0 - iX_1 - X_2 + iX_3)$. For $m = 3$, $K(3)$ is self-dual since $P(3) = K(3)$, which is the octacode O_8 in Example 5.16.

We will see in the next chapter that Theorem 4.37 helps explain the dual like relationship between the binary Kerdock and Preparata codes.

Chapter 5 $\mathbb{Z}_4$-linear Codes

Beginning in early 1990's, an important discovery was made by five coding theorists, Robert Calderbank and Neil Sloane of AT&T Bell Laboratories, Roger Hammons of Hughes Aircraft, Vijay Kumar of the University of South Carolina, and Patrick Sole of the Centre National De La Recherche Scientifique in Valbonne, France. They discovered that binary nonlinear codes viewed in a different way have many algebraic properties. The fact that the binary Kerdock and Preparata codes act like dual codes even though it doesn't make sense for nonlinear codes to have duals can be explained by looking at these codes under the alphabet $\mathbb{Z}_4$ rather than the usual alphabet $\mathbb{Z}_2$. In this chapter we will explain the dual relationship between these codes using the $\mathbb{Z}_4$ alphabet. To do this, we need the gray map.

Definition 5.1: The *gray map* $\phi : \mathbb{Z}_4 \to \mathbb{Z}_2^2$ is defined by $0 \mapsto (0,0)$, $1 \mapsto (0,1)$, $2 \mapsto (1,1)$, and $3 \mapsto (1,0)$.

Clearly, ϕ is a bijection. However, ϕ is not an additive group homomorphism since $\phi(1+3) \neq \phi(1) + \phi(3)$.

Definition 5.2: The following three maps from $\mathbb{Z}_4$ to $\mathbb{Z}_2$ are defined using the chart

Z_4	α	β	γ
0	0	0	0
1	1	0	1
2	0	1	1
3	1	1	0

The map α is an additive group homomorphism since $\alpha(u+v) = \alpha(u) + \alpha(v)$ for all $u, v \in \mathbb{Z}_4$. However, β and γ are not group homomorphisms since $\beta(1+3) \neq \beta(1) + \beta(3)$ and $\gamma(1+3) \neq \gamma(1) + \gamma(3)$.

Note that $\phi(x) = (\beta(x),\gamma(x))$ for all $x \in \mathbb{Z}_4$. We extend the gray map in Definition 5.1 to a map $\phi : \mathbb{Z}_4^n \to \mathbb{Z}_2^{2n}$ in the following way. Let $\mathbf{x} = (x_1, ..., x_n) \in \mathbb{Z}_4^n$. Define $\phi(\mathbf{x}) = (\beta(\mathbf{x}),\gamma(\mathbf{x}))$, where $\beta(\mathbf{x}) = (\beta(x_1),\beta(x_2),...,\beta(x_n))$ and $\gamma(\mathbf{x}) = (\gamma(x_1),\gamma(x_2),...,\gamma(x_n))$. We refer to $\phi(\mathbf{x})$ as the binary image of $\mathbf{x}$ under ϕ.

Example 5.3: Let $\mathbf{x}$ = (1,2,3,0). Then $\phi(\mathbf{x}) = \phi(1,2,3,0) = (\beta(1,2,3,0),\gamma(1,2,3,0))$

= (0,1,1,0,1,1,0,0).

Definition 5.4: For all $x \in \mathbb{Z}_4$, $x = \alpha(x) + 2\beta(x)$ is called the *2-adic representation* of x[8].

Proposition 5.5: If $x = \alpha(x) + 2\beta(x)$ is the 2-adic representation of $x \in \mathbb{Z}_4$, then $\phi(x) = (\beta(x), \alpha(x) + \beta(x))$[16].

Proof: Notice that $\alpha(x)+\beta(x)+\gamma(x) = 0$ for all $x \in \mathbb{Z}_4$. Then $\gamma(x) = \alpha(x)+\beta(x)$ implies that $\phi(x) = (\beta(x),\gamma(x)) = \phi(x) = (\beta(x),\alpha(x) + \beta(x))$ for all $x \in \mathbb{Z}_4$. □

Let C be a quaternary code of length n. Then C is a subgroup of $\mathbb{Z}_4^n$. We can consider the image of C under the gray map $\phi : \mathbb{Z}_4^n \to \mathbb{Z}_2^{2n}$. Clearly, ϕ(C) is a binary code of length $2n$. We will explore the properties of ϕ(C). First, we will see that the gray map preserves weight and distance.

Lemma 5.6: $w_L(x) = w(\phi(x))$ and $d_L(x,y) = d(\phi(x),\phi(y))$ for all $x, y \in \mathbb{Z}_4$[8].

Proof: We will show that $w_L(x) = w(\phi(x))$ holds for all $x \in Z_4$. $w_L(0) = 0 = w((0,0)) = w(\phi(0))$. $w_L(1) = 1 = w((0,1))= w(\phi(1))$. $w_L(2) = 2 = w((1,1)) = w(\phi(2))$. $w_L(3) = 1 = w((1,0)) = w(\phi(3))$.

Now, $d_L(x,y) = w_L(x-y) = w_L(x)-w_L(y) = w(\phi(x))-w(\phi(y)) = w(\phi(x)-\phi(y)) = d(\phi(x),\phi(y))$.

□

Theorem 5.7: The gray map ϕ is a weight and distance preserving map from $\mathbb{Z}_4^n$ to $\mathbb{Z}_2^{2n}$[8].

Proof: Let $\mathbf{a} = (a_1,...,a_n) \in \mathbb{Z}_4^n$. Then $w_L(\mathbf{a}) = \sum_{i=1}^n w_L(a_i)$ and $w(\phi(\mathbf{a})) = w(\beta(\mathbf{a}),\gamma(\mathbf{a}))$
$=w(\beta(\mathbf{a}))+w(\gamma(\mathbf{a})) = \sum_{i=1}^n w(\beta(a_i)) + \sum_{i=1}^n w(\gamma(a_i)) = \sum_{i=1}^n w(\beta(a_i),\gamma(a_i))$
$= \sum_{i=1}^n w(\phi(a_i)) = \sum_{i=1}^n w_L(a_i) = w_L(\mathbf{a})$.

Now let $\mathbf{b} \in Z_4^n$. $d(\phi(\mathbf{a}),\phi(\mathbf{b})) = w(\phi(\mathbf{a})-\phi(\mathbf{b})) = w(\phi(\mathbf{a}))-w(\phi(\mathbf{b})) = w_L(\mathbf{a})-w_L(\mathbf{b}) = w_L(\mathbf{a}-\mathbf{b}) = d_L(\mathbf{a},\mathbf{b})$. See [16]. □

Theorem 5.8: Let C be a quaternary code and C'=ϕ(C) be the binary image of C under the gray map. Then the minimum Lee distance of C is equal to the minimum Hamming distance of C'[8].

Proof: From Theorem 5.7, we know that the gray map is a distance preserving map such that $d(\phi(a),\phi(b)) = d_L(a,b)$ where $a, b \in C$. Let d_L denote the minimum Lee distance of C and d denote the minimum Hamming distance of C'. Then $d_L = \min\{d_L(c,c') \mid c,c' \in C, c \neq c'\} = \min\{d(\phi(c),\phi(c')) \mid c,c' \in C, c \neq c'\} =$

$\min\{d(\phi(c),\phi(c')) \mid \phi(c),\phi(c') \in C', \phi(c) \neq \phi(c')\} = d.$ □

Definition 5.9: A binary code C' is called a $\mathbb{Z}_4$-*linear code* if it is the image of some quaternary code under the gray map. That is, C' is $\mathbb{Z}_4$-linear if $C' = \phi(C)$ for some quaternary code C.

Let C be a quaternary code. The binary image of C under the gray map is not necessarily a linear code in the alphabet $\mathbb{Z}_2$. See Example 5.16. Therefore $\phi(C)$ need not have a dual code in the usual sense. However, we can define the dual of $\phi(C)$ in the following manner.

Definition 5.10: Let C be a quaternary code of length n with dual $C^{\perp}$. Let $C' = \phi(C) \in \mathbb{Z}_2^{2n}$ denote the binary image of C. The *quaternary dual* of C' is defined to be $\phi(C^{\perp}) \in \mathbb{Z}_2^{2n}$; that is, $C'_{\perp} = \phi(C^{\perp})$. If $C' = C'_{\perp}$, then C' is called a *self-dual* $\mathbb{Z}_4$-*linear code.*

Theorem 5.11: Let C and $C^{\perp}$ be dual quaternary codes, and $C' = \phi(C)$ and $C' = \phi(C^{\perp})$ be their binary images. Then the weight enumerators $W_{C'}(X,Y)$ and $W_{C'_{\perp}}(X,Y)$ of C' and $C'_{\perp}$, respectively, are related by the binary MacWilliams Identity

$$W_{C'_{\perp}}(X,Y) = \frac{1}{|C'|} W_{C'}(X+Y, X-Y)[8].$$

Proof: Let C be a quaternary code and C' be its binary image under the gray map. Since ϕ is a bijection, $|C| = |C'|$. Since ϕ is weight preserving,

$W_{C'_{\perp}}(X,Y) = Lee_{C^{\perp}}(X,Y) = \frac{1}{|C|} Lee_C(X+Y, X-Y) = \frac{1}{|C'|} W_{C'}(X+Y, X-Y).$ □

Since $C' = \phi(C)$ is not necessarily linear, it is natural to ask under what conditions C' is linear. We would also like to know when any binary code C' is the image of a quaternary code. To answer these questions, we need the notion of a swap map.

Definition 5.12: Let $(x_1, ..., x_n, x_{n+1}, ..., x_{2n}) \in A^{2n}$ be a word of length $2n$. The permutation $\sigma : A^{2n} \to A^{2n}$ defined by

$$(x_1, ..., x_n, x_{n+1}, ..., x_{2n}) \longmapsto (x_{n+1}, ..., x_{2n}, x_1, ..., x_n)$$

is called the *swap map*[8].

The swap map is an automorphism that interchanges the left and right halves of each word. For all $\mathbf{x} \in \mathbb{Z}_4^n$, $\sigma(\phi(\mathbf{x})) = \sigma(\beta(\mathbf{x}),\gamma(\mathbf{x})) = (\gamma(\mathbf{x}),\beta(\mathbf{x})) = \phi(-\mathbf{x})$[7].

Lemma 5.13: For all $\mathbf{x}, \mathbf{y} \in \mathbb{Z}_4^n$, we have $\phi(\mathbf{x}+\mathbf{y}) = \phi(\mathbf{x})+\phi(\mathbf{y})+(\phi(\mathbf{x})+\sigma(\phi(\mathbf{x})))*(\phi(\mathbf{y})+\sigma(\phi(\mathbf{y})))$.

Proof: Let $\mathbf{x}, \mathbf{y} \in \mathbb{Z}_4^n$. Then $(\phi(\mathbf{x})+\sigma(\phi(\mathbf{x})))*(\phi(\mathbf{y})+\sigma(\phi(\mathbf{y})))=$

$$= ((\beta(\mathbf{x}), \gamma(\mathbf{x})) + (\gamma(\mathbf{x}), \beta(\mathbf{x}))) * ((\beta(\mathbf{y}), \gamma(\mathbf{y})) + (\gamma(\mathbf{y}), \beta(\mathbf{y})))$$

$$= (\beta(\mathbf{x}) + \gamma(\mathbf{x}), \gamma(\mathbf{x}) + \beta(\mathbf{x})) * (\beta(\mathbf{y}) + \gamma(\mathbf{y}), \gamma(\mathbf{y}) + \beta(\mathbf{y}))$$

$$= (\alpha(\mathbf{x}), \alpha(\mathbf{x})) * (\alpha(\mathbf{y}), \alpha(\mathbf{y}))$$

$$= (\alpha(\mathbf{x}) * \alpha(\mathbf{y}), \alpha(\mathbf{x}) * \alpha(\mathbf{y}))$$

We claim that $\phi(\mathbf{x})+\phi(\mathbf{y})+\phi(\mathbf{x}+\mathbf{y}) = (\alpha(\mathbf{x}) * \alpha(\mathbf{y}), \alpha(\mathbf{x}) * \alpha(\mathbf{y}))$. Now,

$\phi(\mathbf{x})+\phi(\mathbf{y})+\phi(\mathbf{x}+\mathbf{y}) = (\beta(\mathbf{x}) + \beta(\mathbf{y}) + \beta(\mathbf{x}+\mathbf{y}), \gamma(\mathbf{x}) + \gamma(\mathbf{y}) + \gamma(\mathbf{x}+\mathbf{y}))$. This implies that if $\beta(\mathbf{x}) + \beta(\mathbf{y}) + \beta(\mathbf{x}+\mathbf{y}) = \gamma(\mathbf{x}) + \gamma(\mathbf{y}) + \gamma(\mathbf{x}+\mathbf{y}) = \alpha(\mathbf{x}) * \alpha(\mathbf{y})$ for all $\mathbf{x}, \mathbf{y} \in \mathbb{Z}_4^n$, we are done. It is easy to verify this for all $x, y \in \mathbb{Z}_4$. Now extend to $\mathbb{Z}_4^n$. See [16]. □

Corollary 5.14: For all $\mathbf{x}, \mathbf{y} \in \mathbb{Z}_4^n$, we have $\phi(\mathbf{x}+\mathbf{y}) = \phi(\mathbf{x})+\phi(\mathbf{y})+\phi(2\alpha(\mathbf{x}) * \alpha(\mathbf{y}))$[8].

Proof: Let $\mathbf{x} = (x_1, ..., x_n)$ and $\mathbf{y} = (y_1, ..., y_n) \in \mathbb{Z}_4^n$. If x_i and y_i are both odd for some i, then $\alpha(x_i)*\alpha(y_i) = 1$. This implies that $\beta(2\alpha(x_i) * \alpha(y_i)) = \beta(2) = 1 = \alpha(x_i) * \alpha(y_i)$ and $\gamma(2\alpha(x_i) * \alpha(y_i)) = \gamma(2) = 1 = \alpha(x_i) * \alpha(y_i)$. If x_i or y_i are even for some i, then $\alpha(x_i)*\alpha(y_i) = 0$. This implies that $\beta(2\alpha(x_i) * \alpha(y_i)) = \beta(0) = 0 = \alpha(x_i) * \alpha(y_i)$ and $\gamma(2\alpha(x_i) * \alpha(y_i)) = \gamma(0) = 0 = \alpha(x_i) * \alpha(y_i)$. Thus $\beta(2\alpha(\mathbf{x}) * \alpha(\mathbf{y})) = \alpha(\mathbf{x}) * \alpha(\mathbf{y})$ and $\gamma(2\alpha(\mathbf{x}) * \alpha(\mathbf{y})) = \alpha(\mathbf{x}) * \alpha(\mathbf{y})$.

Now $\phi(2\alpha(\mathbf{x}) * \alpha(\mathbf{y})) = (\beta(2\alpha(\mathbf{x}) * \alpha(\mathbf{y})), \gamma(2\alpha(\mathbf{x}) * \alpha(\mathbf{y}))) = (\alpha(\mathbf{x}) * \alpha(\mathbf{y}), \alpha(\mathbf{x}) * \alpha(\mathbf{y}))$
$= (\alpha(\mathbf{x}), \alpha(\mathbf{x}) * \alpha(\mathbf{y}), \alpha(\mathbf{y})) = (\beta(\mathbf{x}) + \gamma(\mathbf{x}), \gamma(\mathbf{x}) + \beta(\mathbf{x})) * (\beta(\mathbf{y}) + \gamma(\mathbf{y}), \gamma(\mathbf{y}) + \beta(\mathbf{y}))$

$= (\phi(\mathbf{x}) + \phi(-\mathbf{x})) * (\phi(\mathbf{y}) + \phi(-\mathbf{y})) = (\phi(\mathbf{x}) + \sigma(\phi(\mathbf{x}))) * (\phi(\mathbf{y}) + \sigma(\phi(\mathbf{y})))$. By Lemma 5.13, $(\phi(\mathbf{x}) + \sigma(\phi(\mathbf{x}))) * (\phi(\mathbf{y}) + \sigma(\phi(\mathbf{y}))) = \phi(\mathbf{x}) + \phi(\mathbf{y}) + \phi(\mathbf{x}+\mathbf{y})$.

Therefore $\phi(\mathbf{x}+\mathbf{y}) = \phi(\mathbf{x}) + \phi(\mathbf{y}) + \phi(2\alpha(\mathbf{x}) * \alpha(\mathbf{y}))$. See [16]. □

The following theorem answers our first question.

Theorem 5.15: The binary image C′ = ϕ(C) of a quaternary code C is linear if and only if $\mathbf{a}, \mathbf{b} \in$ C implies $2\alpha(\mathbf{a}) * \alpha(\mathbf{b}) \in$ C[8].

Proof: Assume C′ is linear. Since C is a quaternary code, $\mathbf{x}+\mathbf{y} \in$ C for all $\mathbf{x}, \mathbf{y} \in$ C. Then $\phi(\mathbf{x})$, $\phi(\mathbf{y})$, and $\phi(\mathbf{x}+\mathbf{y})$ are elements of C′. Since C′ is linear, $\phi(\mathbf{x})+\phi(\mathbf{y})+\phi(\mathbf{x}+\mathbf{y}) \in$ C′. By Corollary 5.14, $\phi(\mathbf{x}) + \phi(\mathbf{y}) + \phi(\mathbf{x}+\mathbf{y}) = \phi(2\alpha(\mathbf{x}) * \alpha(\mathbf{y}))$. This implies that $\phi(2\alpha(\mathbf{x}) * \alpha(\mathbf{y})) \in$ C′. Since ϕ is a bijection, $2\alpha(\mathbf{x}) * \alpha(\mathbf{y}) \in$ C..

Now assume that $2\alpha(\mathbf{c}) * \alpha(\mathbf{c}') \in$ C for all $\mathbf{c}, \mathbf{c}' \in$ C. Let $\mathbf{u}, \mathbf{v} \in$ C′. There exists some $\mathbf{x}, \mathbf{y} \in$ C such that $\phi(\mathbf{x}) = \mathbf{u}$ and $\phi(\mathbf{y}) = \mathbf{v}$. Since C is linear, $\mathbf{x}+\mathbf{y}+2\alpha(\mathbf{x}) * \alpha(\mathbf{y}) \in$ C and $\phi(\mathbf{x}+\mathbf{y}+2\alpha(\mathbf{x}) * \alpha(\mathbf{y})) \in$ C′. By Corollary 5.14, $\phi(\mathbf{x}+\mathbf{y}+2\alpha(\mathbf{x}) * \alpha(\mathbf{y})) = \phi(\mathbf{x}+\mathbf{y}) + \phi(2\alpha(\mathbf{x}) * \alpha(\mathbf{y})) + \phi(2\alpha(\mathbf{x}+\mathbf{y}) * \alpha(2\alpha(\mathbf{x}) * \alpha(\mathbf{y}))) \in$ C′. Since $\alpha(2) = 0$, $\alpha(2\alpha(\mathbf{x}) * \alpha(\mathbf{y})) = 0$ implies that $\phi(2\alpha(\mathbf{x}+\mathbf{y}) * \alpha(2\alpha(\mathbf{x}) * \alpha(\mathbf{y}))) = 0$. Therefore, $\phi(\mathbf{x}+\mathbf{y}) + \phi(2\alpha(\mathbf{x}) * \alpha(\mathbf{y})) \in$ C′. By Corollary 5.14, $\phi(2\alpha(\mathbf{x}) * \alpha(\mathbf{y})) = \phi(\mathbf{x}) + \phi(\mathbf{y}) + \phi(\mathbf{x}+\mathbf{y})$. Thus $\phi(\mathbf{x}+\mathbf{y}) + \phi(\mathbf{x}) + \phi(\mathbf{y}) + \phi(\mathbf{x}+\mathbf{y}) = \phi(\mathbf{x}) + \phi(\mathbf{y}) = \mathbf{u}+\mathbf{v} \in$ C′. See [16].

□

Example 5.16: The octacode O_8 is defined to be the quaternary code with generator matrix

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 3 & 1 & 2 & 1 \\ 0 & 1 & 0 & 0 & 1 & 2 & 3 & 1 \\ 0 & 0 & 1 & 0 & 3 & 3 & 3 & 2 \\ 0 & 0 & 0 & 1 & 2 & 3 & 1 & 1 \end{bmatrix}.$$

Let $\mathbf{x}_1$ = (1,0,0,0,3,1,2,1) and $\mathbf{x}_2$ = (0,1,0,0,1,2,3,1). $2\alpha(\mathbf{x}_1) * \alpha(\mathbf{x}_2)$ = (0,0,0,0,2,0,0,2) $\notin O_8$. This implies that $\phi(O_8)$ is nonlinear. In fact, $\phi(O_8)$ is the Nordstrom-Robinson code. Therefore, the Nordstrom-Robinson code is $\mathbb{Z}_4$-linear[8].

In Chapter 3, we found that the Nordstrom-Robinson N_{16} code is equal to the Kerdock code K_{m+1} and Preparata code P_{m+1} when $m = 3$. Since K_{m+1} and P_{m+1} satisfy the

MacWilliams Identity, $W_{N_{16}} = \frac{1}{|N_{16}|} W_{N_{16}}(X+Y, X-Y)$. Therefore, The Nordstrom-Robinson code acts like a self-dual code even though duality for nonlinear codes does not make sense. From Example 5.16, we see that N_{16} is the image of a self-dual quaternary code O_8 under the gray map. By Definition 5.10, N_{16} is a self-dual $\mathbb{Z}_4$-linear code. Therefore, Theorem 5.11 explains why the Nordstrom-Robinson code has this "dual" like relationship.

Next, we focus on the second question: when is a binary code C' a $\mathbb{Z}_4$-linear code? To answer this question we have the following theorem.

Theorem 5.17: A binary linear code C' of even length is $\mathbb{Z}_4$-linear if and only if its coordinates can be permuted so that $\mathbf{u}, \mathbf{v} \in C'$ implies $(\mathbf{u} + \sigma(\mathbf{u}))*(\mathbf{v} + \sigma(\mathbf{v})) \in C'$[8].

Proof: Suppose C' is a $\mathbb{Z}_4$-linear code. Then for $\mathbf{u}, \mathbf{v} \in C'$, $\phi(\mathbf{x}) = \mathbf{u}$ and $(\mathbf{y}) = \mathbf{v}\phi$ for some $\mathbf{x}, \mathbf{y} \in C$, where C is a quaternary code. Since C is linear, $\mathbf{x}+\mathbf{y} \in C$. Thus $\phi(\mathbf{x}+\mathbf{y}) = \mathbf{w}$ for some $\mathbf{w} \in C'$. By Lemma 5.13, $(\mathbf{u}+\sigma(\mathbf{u}))*(\mathbf{v}+\sigma(\mathbf{v})) = (\phi(\mathbf{x})+\sigma(\phi(\mathbf{x})))*(\phi(\mathbf{y})+\sigma(\phi(\mathbf{y})))$ $= \phi(\mathbf{x})+\phi(\mathbf{y})+\phi(\mathbf{x}+\mathbf{y}) = \mathbf{u}+\mathbf{v}+\mathbf{w} \in C'$ since C' is a binary linear code.

Now suppose the coordinates can be permuted so that $\mathbf{u}, \mathbf{v} \in C'$ implies $(\mathbf{u}+\sigma(\mathbf{u}))*(\mathbf{v}+\sigma(\mathbf{v})) \in C'$. Let the length of $C' = 2n$. Define $C = \{\mathbf{c} \in \mathbb{Z}_4^n \mid \phi(\mathbf{c}) \in C'\}$. We want to show that C is a quaternary code. Let $\mathbf{x}, \mathbf{y} \in C$. Then $\phi(\mathbf{x}), \phi(\mathbf{y}) \in C'$ implies $(\phi(\mathbf{x})+\sigma(\phi(\mathbf{x})))*(\phi(\mathbf{y})+\sigma(\phi(\mathbf{y}))) \in C'$. By Lemma 5.13, $\phi(\mathbf{x})+\phi(\mathbf{y})+(\phi(\mathbf{x})+\sigma(\phi(\mathbf{x})))*(\phi(\mathbf{y})+\sigma(\phi(\mathbf{y}))) = \phi(\mathbf{x}+\mathbf{y}) \in C'$. Therefore, $\mathbf{x}+\mathbf{y} \in C$ which implies that C is a quaternary code. □

Example 5.18: Let $\mathbf{u}$ = (0,0,0,0,0,0,1,1,0,0,0,0,0,1,0,0,0,1,0,1,1,0,1,1) and $\mathbf{v}$ = (0,0,0,0,0,0,0,1,1,0,0,0,0,0,1,1,0,0,0,1,0,1,1,0,1) to be the 7th and 8th rows of the generator matrix for G_{24}. Then $\mathbf{u}, \mathbf{v} \in G_{24}$. It follows that $(\mathbf{u} + \sigma(\mathbf{u}))*(\mathbf{v} + \sigma(\mathbf{v})) = \mathbf{w}$ = (0,1,0,0,0,0,1,0,0,0,0,1,0,1,0,0,0,0,1,0,0,0,0,1) $\notin G_{24}$ since the wt(w) = 6 < 8, the minimum distance of G_{24}. By Theorem 5.17, the Golay code G_{24} is a [24,12,8] linear code that is not a $\mathbb{Z}_4$-linear code.

Certain Reed-Muller codes are examples of linear codes that are $\mathbb{Z}_4$-linear codes.

Theorem 5.19: The r^{th} order binary Reed-Muller code RM(r,m) of length $n = 2^m$, $m \geq 1$ is $\mathbb{Z}_4$-linear for $r = 0, 1, 2, m-1$, and m.

Proof: See [8]. □

Let ZRM$(r, m-1) = \begin{bmatrix} G(r-1, m-1) \\ 2G(r, m-1) \end{bmatrix}$ denote the generator matrix for the quaternary code that maps to RM(r, m) by the gray map.

Example 5.20: Let ZRM(1,1) $= \begin{bmatrix} G(0,1) \\ 2G(1,1) \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ 2 & 2 \\ 0 & 2 \end{bmatrix}$. G(1,2) is the image of ZRM(1,1) under the gray map since ϕ(ZRM(1,1)) $= \begin{bmatrix} 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 \end{bmatrix} = G(1,2)$. Thus, RM(1,2) is also the image of the quaternary code generated by ZRM(1,1).

Theorem 5.17 gives a requirement for linear codes to be $\mathbb{Z}_4$-linear. Since linear codes have nice algebraic properties, it is not useful information. However, for codes that do not have algebraic structure, this information could be very useful is explaining certain properties, namely duality, of these types of codes. Theorem 5.21 allows us to qualify a nonlinear code as a $\mathbb{Z}_4$-linear.

Theorem 5.21: A binary, not necessarily linear, code C′ of even length is $\mathbb{Z}_4$-linear if and only if after a permutation of its coordinates, $\mathbf{u}, \mathbf{v} \in$ C′ implies

$\mathbf{u} + \mathbf{v} + (\mathbf{u} + \sigma(\mathbf{u})) * (\mathbf{v} + \sigma(\mathbf{v})) \in$ C′[8].

Proof: Suppose C′ $= \phi$(C), where C is a quaternary code. Let $\mathbf{u}, \mathbf{v} \in$ C′, then there are $\mathbf{x}, \mathbf{y} \in$ C such that $\phi(\mathbf{x}) = \mathbf{u}$ and $\phi(\mathbf{y}) = \mathbf{v}$. Since C is a quaternary code, $\mathbf{x} + \mathbf{y} \in$ C. Thus by Lemma 5.13, $\mathbf{u} + \mathbf{v} + (\mathbf{u} + \sigma(\mathbf{u})) * (\mathbf{v} + \sigma(\mathbf{v})) = \phi(\mathbf{x}) + \phi(\mathbf{y}) + (\phi(\mathbf{x}) + \sigma(\phi(\mathbf{x})) * (\phi(\mathbf{y}) + \sigma(\phi(\mathbf{y}))$ $= \phi(\mathbf{x} + \mathbf{y})$. Since ϕ is a bijection and $\mathbf{x} + \mathbf{y} \in$ C, $\phi(\mathbf{x} + \mathbf{y}) \in$ C′, which implies that $\mathbf{u} + \mathbf{v} + (\mathbf{u} + \sigma(\mathbf{u})) * (\mathbf{v} + \sigma(\mathbf{v})) \in$ C′.

Now suppose that $\mathbf{u}, \mathbf{v} \in$ C′ implies $\mathbf{u} + \mathbf{v} + (\mathbf{u} + \sigma(\mathbf{u})) * (\mathbf{v} + \sigma(\mathbf{v})) \in$ C′. Let the length of C′ equal $2n$. Define C $= \{\mathbf{c} \in \mathbb{Z}_4^n \mid \phi(\mathbf{c}) \in$ C′$\}$. We want to show that C is a quaternary code. Let $\mathbf{x}, \mathbf{y} \in$ C. Then $\phi(\mathbf{x}), \phi(\mathbf{y}) \in$ C′. Thus, $\phi(\mathbf{x}) + \phi(\mathbf{y}) + (\phi(\mathbf{x}) + \sigma(\phi(\mathbf{x})) * (\phi(\mathbf{y}) + \sigma(\phi(\mathbf{y})) \in$ C′. By Lemma 5.13, $\phi(\mathbf{x} + \mathbf{y}) \in$ C′. Therefore, $\mathbf{x} + \mathbf{y} \in$ C, which implies that C is a quaternary code. See [16]. □

In the next two subsections, we will show examples of nonlinear codes that are $\mathbb{Z}_4$-linear.

5.1 Kerdock Code as a $\mathbb{Z}_4$-linear code

Let $m \geq 2$ be an integer. Denote the binary image of the quaternary Kerdock code $K(m)$ by $\phi(K(m))$, which is a $\mathbb{Z}_4$-linear code.

Theorem 5.22: Let $m \geq 2$. The $\mathbb{Z}_4$-linear code $\phi(K(m))$ is a nonlinear binary code of length 2^{m+1} with 4^{m+1} codewords.

Proof: Since the length of $K(m)$ is 2^m, $\phi(K(m))$ has length $2(2^m) = 2^{m+1}$.

Since the gray map ϕ is a bijection, $|K(m)| = |\phi(K(m))|$. $K(m)$ has type 4^{m+1} and thus 4^{m+1} codewords. Therefore $\phi(K(m))$ has 4^{m+1} codewords. □

Theorem 5.23: Let $m \geq 3$ be odd. Then $\phi(K(m)) = K_{m+1}$, where K_{m+1} is the nonlinear binary Kerdock code defined in Chapter 3[16].

To prove this theorem, we will need the following lemma.

Lemma 5.24: Let $m \geq 3$ be an odd integer and c = $(c_\infty, c_0, c_1, ..., c_{n-1}) \in K(m)$. Then the 2-adic representation of $c_t = a_t + 2b_t$ is given by $a_t = A + Tr(\Pi\overline{\xi}^{\,t})$ and $b_t = B + Tr(\eta\overline{\xi}^{t}) + \sum_{j=1}^{(m-1)/2} Tr(\Pi\overline{\xi}^{t})$, where $t \in \{\infty, 0, 1, ..., n-1\}$, $A, B \in \mathbb{Z}_2$, and $\Pi, \eta \in \mathbb{F}_{2^m}$ [8].

Recall that $\overline{\xi}$ is the image of ξ under the map defined by $\psi : \mathbb{Z}_4[x] \longrightarrow \mathbb{Z}_2[x]$.

Proof: Let $\lambda = \xi^r + 2\xi^s \in \mathbb{Z}_4[\xi]$ where $r, s \in \{\infty, 0, 1, ..., n-1\}$. By Proposition 4.33, $c_t = \varepsilon + T((\xi^r + 2\xi^s)\xi^t) = \varepsilon + T(\xi^{r+t} + 2\xi^{s+t}) = \varepsilon + T(\xi^{r+t}) + T(2\xi^{s+t}) = \varepsilon + T(\xi^{r+t}) + 2T(\xi^{s+t})$. Since $c_t = a_t + 2b_t$ is the 2-adic representation of c_t, $a_t + 2b_t = \varepsilon + T(\xi^{r+t}) + 2T(\xi^{s+t})$. Applying the map $\psi := \mathbb{Z}_4[x] \to \mathbb{Z}_2[x]$, we obtain $\psi(a_t + 2b_t) = \psi(\varepsilon + T(\xi^{r+t}) + 2T(\xi^{s+t}))$. Since the map ψ is a homomorphism, $\psi(a_t) + \psi(2)\psi(b_t) = \psi(\varepsilon) + \psi(T(\xi^{r+t})) + \psi(2)\psi(T(\xi^{s+t}))$. Note that $a_t, b_t \in \mathbb{Z}_2$. Thus, $\psi(a_t) = a_t$ and $\psi(b_t) = b_t$. Since $\psi(2) = 0$, we obtain $a_t = \psi(\varepsilon) + \psi(T(\xi^{r+t}))$. Since $\varepsilon \in \mathbb{Z}_4$, $\psi(\varepsilon) = \overline{\varepsilon} \in \mathbb{Z}_2$. By Proposition 4.32, $\psi \circ T = Tr \circ \psi$. Thus, $\psi(T(\xi^{r+t})) = Tr(\psi(\xi^{r+t}) = Tr(\overline{\xi}^{\,r+t})$. Therefore, $a_t = A + Tr(\Pi\overline{\xi}^{\,t})$, where $A = \overline{\varepsilon}$ and $\Pi = \overline{\xi}^{\,r}$.

Now $c_t - c_t^2 = a_t + 2b_t - (a_t + 2b_t)^2 = a_t + 2b_t - a_t^2$. Since $a_t \in \mathbb{Z}_2$, $a_t = a_t^2$. Thus, $2b_t = c_t - c_t^2$. Therefore, $2b_t = \varepsilon + T(\xi^{r+t}) + 2T(\xi^{s+t}) - (\varepsilon + T(\xi^{r+t}) + 2T(\xi^{s+t}))^2 = \varepsilon + T(\xi^{r+t}) + 2T(\xi^{s+t}) - (\varepsilon^2 + 2\varepsilon T(\xi^{r+t}) + T^2(\xi^{r+t})) = \varepsilon - \varepsilon^2 + T(\xi^{r+t}) - T^2(\xi^{r+t}) - 2\varepsilon T(\xi^{r+t}) + 2T(\xi^{s+t})$.

Since $\varepsilon \in \mathbb{Z}_4$, $\varepsilon = \alpha(\varepsilon) + 2\beta(\varepsilon)$ is the 2-adic representation for ε. Thus $\varepsilon - \alpha(\varepsilon) = 2\beta(\varepsilon)$. By checking all possible choices for ε, we see that $\alpha(\varepsilon) = \varepsilon^2$. Therefore, $\varepsilon - \varepsilon^2 = 2\beta(\varepsilon)$.

One can verify that, $T(\xi^{r+t}) - T^2(\xi^{r+t}) = T(\xi^{r+t})(1 - T(\xi^{r+t})) = -2\sum_{0\le j\le k}(\xi^{r+t})^{2^j+2^k} = 2\sum_{0\le j\le k}(\xi^{r+t})^{2^j+2^k} = 2\sum_{j=1}^{(m-1)/2} Tr(\overline{\xi}^{\,r+t})^{1+2^j}$ since m is odd.

Now, $-2\varepsilon T(\xi^{r+t}) + 2T(\xi^{s+t}) = 2\varepsilon T(\xi^{r+t}) + 2T(\xi^{s+t}) = 2(\varepsilon T(\xi^{r+t}) + T(\xi^{s+t}))$. Since T is a homomorphism, we obtain $2(\varepsilon T(\xi^{r+t}) + T(\xi^{s+t})) = 2(T(\varepsilon\xi^{r+t}) + T(\xi^{s+t})) = 2T(\varepsilon\xi^{r+t} + \xi^{s+t}) = 2(T(\varepsilon\xi^r + \xi^s)\xi^t)$.

Thus, $2b_t = 2\beta(\varepsilon) + 2(T(\varepsilon\xi^r + \xi^s)\xi^t) + 2\sum_{j=1}^{(m-1)/2} Tr(\overline{\xi}^{\,r+t})^{1+2^j}$. Dividing through by 2, we obtain $b_t = \beta(\varepsilon) + (T(\varepsilon\xi^r + \xi^s)\xi^t) + \sum_{j=1}^{(m-1)/2} Tr(\overline{\xi}^{\,r+t})^{1+2^j}$. $\psi(b_t) = \psi(\beta(\varepsilon) + (T(\varepsilon\xi^r + \xi^s)\xi^t) + \sum_{j=1}^{(m-1)/2} Tr(\overline{\xi}^{\,r+t})^{1+2^j}) = \psi(\beta(\varepsilon)) + \psi((T(\varepsilon\xi^r + \xi^s)\xi^t)) + \psi(\sum_{j=1}^{(m-1)/2} Tr(\overline{\xi}^{\,r+t})^{1+2^j})$ $= \beta(\varepsilon) + Tr(\psi(\varepsilon\xi^r + \xi^s)\xi^t)) + \psi(\sum_{j=1}^{(m-1)/2} Tr(\overline{\xi}^{\,r+t})^{1+2^j})$. This implies that $b_t = B + Tr(\eta\overline{\xi}^{\,t}) + \sum_{j=1}^{(m-1)/2} Tr(\Pi\overline{\xi}^{\,t})^{1+2^j}$, where $B = \beta(\varepsilon)$, $\eta = \overline{\varepsilon}\overline{\xi}^{\,r} + \overline{\xi}^{\,s}$, and $\Pi = \overline{\xi}^{\,r}$. See [16].□

Proof of Theorem 5.23: Let $\mathbf{c} = (c_\infty, c_0, c_1, ..., c_{n-1}) \in K(m)$. Then $\mathbf{c} = \mathbf{a} + 2\mathbf{b}$ is the 2-adic representation of $\mathbf{c}$ for some $\mathbf{a} = (a_\infty, a_0, ..., a_{n-1}), \mathbf{b} = (b_\infty, b_0, ..., b_{n-1}) \in \mathbb{Z}_2^{n+1}$. By Lemma 5.24, there exists $A, B \in \mathbb{Z}_2$ and $\Pi, \eta \in \mathbb{Z}_{2^m}$ such that $c_t = a_t + 2b_t$ for $t = \{\infty, 0, 1, ..., n-1\}$, where $a_t = A + Tr(\Pi\overline{\xi}^t)$ and $b_t = B + Tr(\eta\overline{\xi}^t) + \sum_{j=1}^{(m-1)/2} Tr(\Pi\overline{\xi}^t)^{1+2^j}$. Let $Q(\Pi\overline{\xi}^{\,t}) = \sum_{j=1}^{(m-1)/2} Tr(\Pi\overline{\xi}^{\,t})^{1+2^j}$.

Consider $\phi(\mathbf{c}) - (Q(\Pi\overline{\xi}^{\infty}), Q(\Pi\overline{\xi}^{0}), ..., Q(\Pi\overline{\xi}^{n-1}), Tr(\Pi\overline{\xi}^{\infty}) + Q(\Pi\overline{\xi}^{\infty}), Tr(\Pi\overline{\xi}^{0}) + Q(\Pi\overline{\xi}^{0}), ...,$ $Tr(\Pi\overline{\xi}^{n-1}) + Q(\Pi\overline{\xi}^{n-1})$. By Proposition 5.3, $\phi(\mathbf{c}) = (\mathbf{b}, \mathbf{a}+\mathbf{b})$. Thus, $\phi(\mathbf{c}) - (Q(\Pi\overline{\xi}^{\infty}), Q(\Pi\overline{\xi}^{0}),$ $..., Q(\Pi\overline{\xi}^{\,n-1}), Tr(\Pi\overline{\xi}^{\,\infty}) + Q(\Pi\overline{\xi}^{\,\infty}), Tr(\Pi\overline{\xi}^{\,0}) + Q(\Pi\overline{\xi}^{\,0}), ..., Tr(\Pi\overline{\xi}^{\,n-1}) + Q(\Pi\overline{\xi}^{\,n-1})$

$= (\mathbf{b}, \mathbf{a} + \mathbf{b}) - (Q(\Pi\overline{\xi}^{\,\infty}), Q(\Pi\overline{\xi}^{\,0}), ..., Q(\Pi\overline{\xi}^{\,n-1}), Tr(\Pi\overline{\xi}^{\,\infty}) + Q(\Pi\overline{\xi}^{\,\infty}), Tr(\Pi\overline{\xi}^{\,0}) + Q(\Pi\overline{\xi}^{\,0}),$ $..., Tr(\Pi\overline{\xi}^{\,n-1}) + Q(\Pi\overline{\xi}^{\,n-1}) = (b_\infty, b_0, ..., b_{n-1}, a_\infty + b_\infty, a_0 + b_0, ..., a_{n-1} + b_{n-1})$

$-(Q(\Pi\overline{\xi}^{\,\infty}), Q(\Pi\overline{\xi}^{\,0}), ..., Q(\Pi\overline{\xi}^{\,n-1}), Tr(\Pi\overline{\xi}^{\,\infty}) + Q(\Pi\overline{\xi}^{\,\infty}), Tr(\Pi\overline{\xi}^{\,0}) + Q(\Pi\overline{\xi}^{\,0}), ..., Tr(\Pi\overline{\xi}^{\,n-1}) +$ $Q(\Pi\overline{\xi}^{n-1})) = (b_\infty - Q(\Pi\overline{\xi}^{\infty}), b_0 - Q(\Pi\overline{\xi}^{0}), ..., b_{n-1} - Q(\Pi\overline{\xi}^{n-1}), a_\infty + b_\infty - Tr(\Pi\overline{\xi}^{\infty}) - Q(\Pi\overline{\xi}^{\infty}),$ $a_0 + b_0 - Tr(\Pi\overline{\xi}^{\,0}) - Q(\Pi\overline{\xi}^{\,0}), ..., a_{n-1} + b_{n-1} - Tr(\Pi\overline{\xi}^{\,n-1}) - Q(\Pi\overline{\xi}^{\,n-1}))$.

Let $\mathbf{u} = (b_\infty - Q(\Pi\overline{\xi}^{\,\infty}), b_0 - Q(\Pi\overline{\xi}^{\,0}), ..., b_{n-1} - Q(\Pi\overline{\xi}^{\,n-1}))$ and $\mathbf{v} = (a_\infty - Tr(\Pi\overline{\xi}^{\,\infty}), a_0 - Tr(\Pi\overline{\xi}^{\,0}), ..., a_{n-1} - Tr(\Pi\overline{\xi}^{\,n-1}))$. Since $a_t = A + Tr(\Pi\overline{\xi}^{\,t})$ and $b_t = B + Tr(\eta\overline{\xi}^{\,t}) + Q(\Pi\overline{\xi}^{\,t})$, $\mathbf{u} = B1^{2^m} + Tr(\eta\overline{\xi}^{\,t})$ and $\mathbf{v} = A1^{2^m}$. By Definition 3.6, $L_\eta = Q(\eta\overline{\xi}^{\,t})$ is the left side and $R_\eta(\overline{\xi}^{\,t}) = Q(\eta\overline{\xi}^{\,t}) + Tr(\eta\overline{\xi}^{\,t})$ is the right side of a codeword in RM$(2, m+1)$. This

implies that $Tr(\eta\overline{\xi}^t) \in \mathrm{RM}(1,m)$. Clearly $B1^{2^m} \in \mathrm{RM}(1,m)$ and $A1^{2^m}\mathrm{RM}(0,m)$. Thus $\mathbf{u} \in \mathrm{RM}(1,m)$ and $\mathbf{v} \in \mathrm{RM}(0,m)$. This implies that $(\mathbf{u}, \mathbf{u}+\mathbf{v}) = \phi(\mathbf{c}) - (Q(\Pi\overline{\xi}^t), Tr(\Pi\overline{\xi}^t) + Q(\Pi\overline{\xi}^t)) \in \mathrm{RM}(1,m+1)$. Thus, $\phi(\mathbf{c}) \in (Q(\Pi\overline{\xi}^t), Tr(\Pi\overline{\xi}^t) + Q(\Pi\overline{\xi}^t)) + \mathrm{RM}(1,m+1)$. Since $(Q(\Pi\overline{\xi}^t), Tr(\Pi\overline{\xi}^t) + Q(\Pi\overline{\xi}^t))$ is a coset representative for K_{m+1} as in Definition 3.6, $\phi(\mathbf{c}) \in K_{m+1}$. Thus $K(m) \subseteq K_{m+1}$. Since $|K(m)| = |K_{m+1}|$, $K(m) = K_{m+1}$. See [16]. □

Thus the Kerdock code $K_{m+1}$is $\mathbb{Z}_4$-linear. By Definition 5.10 , K_{m+1} has a quaternary dual code. Since the quaternary Kerdock code $K(m)$ and the quaternary Preparata code $P(m)$ are $\mathbb{Z}_4$ duals, the quaternary dual code for K_{m+1} is $\phi(P(m))$.

5.2 Preparata Code as a $\mathbb{Z}_4$-linear code

Let $m \geq 2$ be an integer. Denote the binary image of the quaternary Preparata code $P(m)$ by $\phi(P(m))$, which is a $\mathbb{Z}_4$ -linear code.

Theorem 5.25: Let $m \geq 2$. The Z_4-linear code $\phi(P(m))$ is a binary code of length 2^{m+1} with 4^{2^m-m-1} codewords.

Proof: Since the length of $P(m)$ is 2^m, $\phi(P(m))$ has length $2(2^m) = 2^{m+1}$. Since the gray map ϕ is a bijection, $|P(m)| = |\phi(P(m))|$. $P(m)$ has type 4^{2^m-m-1} and thus 4^{2^m-m-1} codewords. Therefore $\phi(P(m))$ has 4^{2^m-m-1} codewords. □

Since the quaternary Preparata code $P(m)$ is the dual to the quaternary Kerdock code $K(m)$, $\phi(P(m))$ is the quaternary dual of $K_{m+1} = \phi(K(m))$. Thus, $W_{\phi(P(m))}(X,Y) = \frac{1}{|K_{m+1}|}W_{K_{m+1}}(X+Y, X-Y)$ by Theorem 5.11. In Chapter 3 we found that the binary Preparata code P_{m+1} satisfies the MacWilliams Identity, $W_{P_{m+1}}(X,Y) = \frac{1}{|K_{m+1}|}W_{K_{m+1}}(X+Y, X-Y)$. Does this mean that P_{m+1} is the image of the quaternary Preparata code $P(m)$ under the gray map; i.e. does $\phi(P(m)) = P_{m+1}$? The answer is yes[8]. The algebraic structure of $\mathbb{Z}_4$-linear codes explains why P_{m+1} acts like a dual code for the binary Kerdock code K_{m+1} even though duality does not make sense for nonlinear codes.

With this new structure of being $\mathbb{Z}_4$-linear, the Kerdock code K_{m+1} and the Preparata code P_{m+1} have overcome the downfall of being nonlinear codes. Since the quaternary Kerdock and Preparata codes have nice decoding algorithms[8], the same decoding algorithms are used for the binary nonlinear Kerdock and Preparata codes. Now these codes,

which contain more codewords than any linear code with the same length and minimum distance, are predicted to be used in such applications as modems and digital cellular radios. This is only the beginning. There is no stopping these codes from being used in many other areas of communication.

References

References

[1] Baker, Ronald D., van Lint, Jacubus H., and Wilson, Richard M. (May 1983). On the Preparata and Goethals Codes, *IEEE Transactions on Information Theory* **29**, 342-354.

[2] Carlet, Claude. (1989). A Simple Description of Kerdock Codes, *Lecture Notes in Computer Science* **388**, 202-208.

[3] Carlet, Claude. (1999). On Kerdock Codes, *Contemporary Mathematics* **225**, 155-163.

[4] Cipra, Barry. (October 29, 1993). Nonlinear Codes Straighten up- and Get to Work, *Science*, 262.

[5] Cipra, Barry. (February 1994). Coding Theorists Writing Linearity out of Nonlinear Codes, *SIAM News*, **27**, 1.

[6] Cipra, Barry. (1994). Straightening Out Nonlinear Codes, *What's Happening in the Mathematical Sciences* **2**, 37-40.

[7] Hammons, A. R., Jr., Kumar, P. V., Calerbank, A.R., Sloane, N. J. A. and Sole, P. (October 1993). A Linear Construction for Certain Kerdock and Preparata Codes, *American Mathematical Society* **29**, 218-222.

[8] Hammons, A. R., Jr., Kumar, P. V., Calerbank, A.R., Sloane, N. J. A. and Sole, P. (1994). The $\mathbb{Z}_4$-linearity of Kerdock, Preparata, Goethals, and related codes, *IEEE Transactions on Information Theory* **40**, 301-309.

[9] Hoffman, D.G., Leonard, D.A., Lindner, C.C., Phelps, K.T., Rodger, C.A., and Wall, J.R. (1991). *Coding Theory: The Essentials*. New York: Marcel Dekker, Inc.

[10] Kerdock, A.M. (1972). A class of Low-Rate Nonlinear Binary Codes, *Information and Control* **20**, 182-197.

[11] Leonard, D.A. and Rodger, C.A. (1984). Explicit Kerdock Codes over GF(2), *Lecture Notes in Computer Science* **228**, 130-135.

[12] Mac Williams, F.J. and Sloane, N.J.A. (1977). *The Theory of Error-correcting Codes.* Amsterdam: North-Holland.

[13] Pretzel, Oliver. (1992). *Error-Correcting Codes and Finite Fields.* Oxford: Clarendon Press.

[14] Stichtenoth, Henning. (1993). *Algebraic Function Fields and Codes.* Springer-Verlag.

[15] van Lint, Jacubus H. (1982). *Introduction to Coding Theory.* New York: Springer-Verlag.

[16] Wan, Zhe-Xian. (1997). *Quaternary Codes.* Singapore: World Scientific Publishing Co.

[17] Walker, Judy. *Codes and Curves.* American Mathematical Society.

Vita

Terri Annette Bedford was born in Mobile, Alabama, on January 8, 1977. She graduated from Mary G. Montgomery High School in Semmes, Alabama, in June 1995. She entered Huntingdon College, in Montgomery, Alabama, during August 1995 where in May 1999, she received the Bachelor of Arts in Mathematics. Following graduation, she entered the graduate program at the University of Tennessee, Knoxville in August 1999 as a graduate teaching assistant. The masters degree was received August 2001. She is presently teaching at A. Crawford Mosley High School in Panama City, Florida.