An Introduction to Algebraic Coding Theory and Polynomial Codes

John Blackwelder and Steven DiSilvio

April 23, 2021

1 Introduction

Coding theory as a field began to develop at the end of the 1940s [4]. In large part driven by the question of how best to design telephone systems, Claude Shannon wrote a foundational article in 1948 titled "A Mathematical Theory of Communication," in which he described how sources of noise could interfere with a signal being sent across a channel and theorized an upper bound for how efficiently information could be conveyed [3]. Inspired by his findings, researchers soon began applying a variety of mathematical methods towards developing efficient codes. Linear algebra, group theory, and ring theory all proved to be very powerful tools towards this end, leading to the development of algebraic coding theory as a subfield of coding theory. As the name suggests, algebraic coding theory involves the use of algebraic methods to efficiently encode and decode information, usually expressed in the form of binary bits, over a potentially noisy channel [2]. This project will be dedicated to defining the fundamental concepts of algebraic coding theory with the ultimate goal of applying these to the topic of polynomial codes.

2 Motivation

Due to advances in communications technology and the development of new forms of media over past decades, the problem of how best to encode information to be sent across a potentially noisy medium so as to allow for accurate error detection and correction by the receiver prior to decoding has become extremely relevant. This is the main question which Algebraic Coding Theory attempts to solve.

However, some sort of coding scheme is vital even for messages transmitted over channels with low error rates. Suppose you wish to send a message containing 100 GB, or about $8 * 10^{11}$ binary digits, across the internet (this is approximately the size of a 4k resolution movie). Let A represent the event that at least one bit is transmitted incorrectly across a channel with a bit error rate of 10^{-13} (this error rate is what many consider to be the minimum acceptable rate for data communications applications [1]. Assuming that these $8 * 10^{11}$ events are independent, we have that

$$P(A) = 1 - (1 - 10^{-13})^{8*10^{11}} \approx .077.$$

This translates to a 7.7% likelihood of some sort of error occurring in our file. Such an error could completely break the file if no error checking or correcting steps are put in place. Although this is only one specific example, it nonetheless illustrates the degree to which we all depend on robust information transmission protocols in our daily lives.

One naive method which may immediately come to mind is to simply repeat each digit 3 consecutive times. Thus, to send the 4-bit string 0110 the coded message would be 000 111 111 000 (where spacing has been added for clarity). Thus, if an error were to occur in any single bit, such as in the first bit which would result in the message 100 111 111 000 being received, an algorithm could easily detect this message as invalid. Beyond error detection, error correction could occur if the assumption is made for any triplet of values containing an inconsistency, the specific digit which occurs twice (0 in the previous example) is the value of the bit in the original message corresponding to the triplet.

While such a method works, in practice it is ineffective due to the fact that 3 times as many bits of information are required to be sent, and it is not guaranteed to work when more than a single error occurs (i.e. in the above example if two errors occurred within the first triplet as a result of noise so that 110 111 111 000 was received, the original message would be incorrectly returned as 1110). Our discussion will thus focus on methods of encoding and decoding binary messages that allow for both efficient as well as accurate error detection and correction.

3 Preliminary Assumptions and Definitions

3.1 Assumptions

Throughout the rest of this paper, the following assumptions and simplifications will be made:

- Binary messages of length n are elements of $\mathbb{Z}_2^n$, and as such undergo addition in the typical fashion, where given any $(a_0, \ldots, a_{n-1}), (b_0, \ldots, b_{n-1}) \in \mathbb{Z}_2^n$ we have $(a_0, \ldots, a_{n-1}) + (b_0, \ldots, b_{n-1}) = (a_0 + b_0, \ldots, a_{n-1} + b_{n-1})$.
- Though binary messages of length n are elements of $\mathbb{Z}_2^n$ we will simply represent them as strings of consecutive digits. As an example, the binary message $(1, 1, 0, 1)$ will be represented throughout this paper as 1101.
- Errors resulting from noise occur rarely ($< 10^{-9}$ chance for any given bit), and the event of an error occurring in any particular bit is independent of an error occurring in any other bits
- All bits have the same likelihood of having an error during transmission
- The receiver can infer what message the sender intended to transmit based on which valid code word is closest to the one that was received (an assumption which Judson defines to be maximum likelihood decoding).

3.2 Formal Definition of Block Code

A code system is an (n, m)-**block code**, $n > m$, if the sender divides the bits that they wish to transmit into blocks of m-tuples, encodes each block into an n-tuple using an injective encoding function

$$E : \mathbb{Z}_2^m \to \mathbb{Z}_2^n,$$

and then the receiver uses a decoding function

$$D : \mathbb{Z}_2^n \to \mathbb{Z}_2^m$$

to ideally obtain the original m-tuple that was sent[2]. A **codeword** is simply any element in $E(\mathbb{Z}_2^m)$, the image of our encoding function[2]. Notice that D must be surjective, as given any m-tuple $\mathbf{y} = (b_0, b_1, \ldots, b_{m-1})$ such that $\mathbf{y} \in \mathbb{Z}_2^m$ we can consider the codeword $E(\mathbf{y})$, and it necessarily follows that $D(E(\mathbf{y})) = \mathbf{y}$ for our code to work properly.

Note: From this point on, many of our results will be adapted from Judson's Abstract Algebra, and we will not cite each individual instance of us referencing the textbook.

3.3 Hamming Distance

In order to formalize the notion of "distance" between two binary n-tuples, we introduce the Hamming Distance.

Definition The **Hamming Distance** between two n-tuples, $d(\mathbf{x}, \mathbf{y})$, is the number of bits that these n-tuples do not hold in common. For instance, if $\mathbf{x} = (1, 1, 1, 0)$ and $\mathbf{y} = (1, 1, 1, 1)$ are two 4-tuples, the hamming distance between them is 1.

Important Result At this point we will prove a result that will be relevant in the coming paragraphs and should help us better understand Hamming Distance more intuitively. Suppose $\mathbf{x}, \mathbf{y}, \mathbf{z}$ are three n-tuples. Let $d(\mathbf{x}, \mathbf{z}) = j$ and $d(\mathbf{z}, \mathbf{y}) = k$.

This means that we can obtain $\mathbf{z}$ by flipping j bits in $\mathbf{x}$, and then we can obtain $\mathbf{y}$ by flipping k bits in $\mathbf{z}$. Since we might have flipped some bits more than once, we have

$$j + k \geq \text{the minimum number of bits we needed to change to obtain } \mathbf{y} \text{ from } \mathbf{x}.$$

The right side of the inequality is precisely the definition of $d(\mathbf{x}, \mathbf{y})$. Thus, we now know that for any $\mathbf{x}, \mathbf{y}, \mathbf{z}$,

$$d(\mathbf{x}, \mathbf{y}) \leq d(\mathbf{x}, \mathbf{z}) + d(\mathbf{z}, \mathbf{y}).$$

Minimum Hamming Distance The minimum hamming distance of a block code, $d_{min} = min\{d(\mathbf{x}, \mathbf{y}) \mid \mathbf{x}, \mathbf{y} \in E(\mathbb{Z}_2^n) \wedge \mathbf{x} \neq \mathbf{y}\}$, where we define the encoding function E as before, can tell us a lot about a code's error detection potential. If a code has minimum distance $d_{min} = 2s + 1$, then we are guaranteed to detect up to $2s$ errors as long as our maximum likelihood assumption holds. This is because if no more than $d_{min} - 1$ errors occur, and $\mathbf{x}$ is the codeword sent by the sender, then the message recipient will receive some n-tuple $\mathbf{z}$ such that $d(\mathbf{x}, \mathbf{z}) < d_{min}$, violating the minimality of d_{min}. At this point the receiver will know that $\mathbf{z}$ cannot be a code-word and thus at least one error occurred as the code-word travelled through the channel.

Additionally, let $\mathbf{y} \in E(\mathbb{Z}_2^n)$ be some code-word not equal to $\mathbf{x}$. Necessarily $2s + 1 \leq d(\mathbf{x}, \mathbf{y})$ by assumption of d_{min}. By the inequality that we proved above, we know that $d(\mathbf{x}, \mathbf{y}) \leq d(\mathbf{x}, \mathbf{z}) + d(\mathbf{z}, \mathbf{y})$. Now, suppose that no more than s errors occurred when $\mathbf{x}$ got sent across the channel. This implies that

$$d(\mathbf{x}, \mathbf{z}) + d(\mathbf{z}, \mathbf{y}) \leq s + d(\mathbf{z}, \mathbf{y}).$$

Subtracting s from the right side and from $2s+1$, we find that

$$d(\mathbf{z},\mathbf{y}) \geq s+1.$$

In other words, $\mathbf{z}$ is closer to $\mathbf{x}$ than it is to any other valid code-word as long as no more than s errors occurred while transmitting $\mathbf{x}$. Thus, we can correct this error under the maximum likelihood assumption by deducing that $\mathbf{x}$ was the intended codeword.

We now see that obtaining d_{min} is a potential topic of interest.

4 Linear Codes

The first class of codes we will consider are linear codes.

Definition An (n,m) block-code is a **linear code** if the image of its encoding function (its set codewords) is determined by the null space of some matrix $H \in \mathbb{M}_{m\times n}[\mathbb{Z}_2]$.

As an example of this, consider the image of the encoding function $E : \mathbb{Z}_2^3 \to \mathbb{Z}_2^6$ defined such that

$$E(a_1a_2a_3) = a_1a_2a_3c_1c_2c_3$$

where $c_1 = a_1 + a_2, c_2 = a_1 + a_3$ and $c_3 = a_2 + a_3$. Considering the 3×6 matrix H given by

$$H = \begin{pmatrix} 1 & 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 0 & 1 \end{pmatrix}$$

note that

$$\begin{pmatrix} 1 & 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} a_1 \\ a_2 \\ a_3 \\ c_1 \\ c_2 \\ c_3 \end{pmatrix} = \begin{pmatrix} a_1 + a_2 + c_1 \\ a_1 + a_3 + c_2 \\ a_2 + a_3 + c_3 \end{pmatrix} = \begin{pmatrix} (a_1 + a_2) + c_1 \\ (a_1 + a_3) + c_2 \\ (a_2 + a_3) + c_3 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}$$

by the stipulation on c_1, c_2 and c_3. Thus, $E(\mathbb{Z}_2^3)$ is a subset of the null space of H. Additionally, examining the matrix multiplication above it is easily seen that any element $(a_1, a_2, a_3, a_4, a_5, a_6) \in \mathbb{Z}_2^6$ that is within the null set of H would have to satisfy $a_4 = -(a_1 + a_2) = a_1 + a_2, a_5 = -(a_1 + a_3)$ and $a_6 = a_2 + a_3$. It follows that such an element $(a_1, a_2, a_3, a_4, a_5, a_6)$ would have to be in $E(\mathbb{Z}_2^3)$ as given by $E(a_1, a_2, a_3)$, and by all of the above we have that $E(\mathbb{Z}_2^3)$ is the null set of H.
Thus, the image of the encoding function E is the null space of H, and hence the above coding scheme represents a linear code.

Next, we will prove an important result about linear codes.

Theorem The set of codewords of a linear code C form a group under addition.

Suppose we are given some linear code with a set of codewords C in $\mathbb{Z}_2^n$ and we denote H as the $m \times n$ matrix with elements in $\mathbb{Z}_2$ whose null space is C. Clearly every element in C

has an additive inverse in C, as any element in $\mathbb{Z}_2^n$ is its own additive inverse by the nature of $\mathbb{Z}_2$. Additionally, the zero matrix is in the null space of every matrix, and hence the zero matrix is in C. Thus, all that remains to show is that C is closed under addition. To do so, suppose we are given any $x, y \in C$. It then follows that

$$H(x+y) = Hx + Hy = 0 + 0 = 0$$

and so $x + y$ is in the null space of H, and hence is in C. □

5 Polynomial Codes

Although knowledge of group theory suffices in order to construct H in a way such that we can easily derive a linear code possessing some degree of error detection and error correction from it, building upon this framework with ring theory instead will allow us to create more powerful and efficient codes. The rest of the paper will use the above concepts/ideas as groundwork to delve into polynomial codes, and the error detection/correction that they permit. First, we will introduce the concept of cyclic codes.

Definition Let $E : \mathbb{Z}_2^k \to \mathbb{Z}_2^n$ be the encoding function for a binary (n, k)-block code with image $E(\mathbb{Z}_2^k) = C$. Then this code is a **cyclic code** if for every codeword $(a_0, a_1, \ldots, a_{n-1}) \in C$, the n-tuple $(a_{n-1}, a_0, a_1, \ldots, a_{n-2})$ is a codeword in C as well.

As we will soon see, cyclic codes possess certain desirable properties. Thus, we would like to find a way to construct cyclic codes.

Consider some binary n-tuple $(a_0, a_1, \ldots, a_{n-1})$. Clearly we can uniquely identify a given n-tuple with a polynomial in $\mathbb{Z}_2[x]$, namely $f(x) = a_0 + a_1 x + \cdots + a_{n-1}x^{n-1}$. Likewise, given any polynomial $f(x)$ of degree less than n in $\mathbb{Z}_2[x]$, clearly a unique $n - tuple$ can be generated in which the i^{th} element corresponds to the i^{th} coefficient of the $f(x)$.

This fact will help us create a cyclic (n, k)-code. First, fix some $g(x) \in \mathbb{Z}_2[x]$ such that $deg(g(x)) = n-k$. Now, consider the map $E : \mathbb{Z}_2[x] \to \mathbb{Z}_2[x]$ defined by $E(f(x)) = g(x)f(x)$. Clearly E is injective, since

$$E(f_1) = E(f_2) \implies g(x)f_1(x) = g(x)f_2(x) \implies f_1(x) = f_2(x).$$

If we restrict the domain of E to only polynomials of degree less than k, then the image of E will only contain polynomials of degree less than n. From what we showed above, each element of the domain of this restricted form of E corresponds uniquely with some k-tuple, and each element of the image corresponds uniquely with some n-tuple. Thus, we can consider E to be an injective encoding function mapping $\mathbb{Z}_2^k$ to $\mathbb{Z}_2^n$.

Define the subsets $K, N \subset \mathbb{Z}_2[x]$ such that

$$K = \{f(x) \in \mathbb{Z}_2[x] \mid deg(f(x)) < k\}$$

$$N = \{f(x) \in \mathbb{Z}_2[x] \mid deg(f(x)) < n\}.$$

Next consider the subset $W \subset N$ such that

$$W = \{h(x) \in \mathbb{Z}_2[x] \mid h(x) = g(x)f(x) \text{ for some } f(x) \in K\}.$$

Define $D : N \to K$ such that $\forall h(x) \in W, D(h(x)) = \frac{h(x)}{g(x)}$. Proper decoding of elements of N not in W depends on the error-correcting capabilities of our code, which we will show how to approximate later on. We can say that D is our decoding function by again invoking the correspondence between polynomials and tuples. Additionally, clearly D is surjective, since for any $f(x) \in M$, we can say that $f(x) = D(f(x)g(x))$.

We can say more about polynomial codes by applying our knowledge of ring theory. Consider the quotient ring

$$R_n = \mathbb{Z}_2[x]/(x^n - 1).$$

Every element of R_n is of the form $f(x)+(x^n-1)$, where $f(x) \in \mathbb{Z}_2[x]$ such that $deg(f(x)) < n$. To see this, simply note that clearly any element of R_n is of the form $f'(x) + (x^n - 1)$ for some $f'(x) \in \mathbb{Z}_2[x]$, and $f(x)$ such that $deg(f(x)) < n$ could be obtained simply by taking the remainder when $f'(x)$ is divided by x^n-1. In this case clearly $f'(x)+(x^n-1) = f(x)+(x^n-1)$ as $f'(x) - f(x) \in (x^n - 1)$.

Given any such $f(x) = a_0 + a_1x + \cdots + a_{n-1}x^{n-1}$, letting $t = x + (x^n - 1)$ we find that

$$\begin{aligned} f(x) + (x^n - 1) &= a_0 + a_1x + \cdots + a_{n-1}x^{n-1} + (x^n - 1) \\ &= a_0 + (a_1x + (x^n - 1)) + \cdots + (a_{n-1}x^{n-1} + (x^n - 1)) \\ &= a_0 + a_1(x + (x^n - 1)) + \cdots + a_{n-1}(x + (x^n - 1))^{n-1} \\ &= f(t). \end{aligned}$$

This means that every element of R_n can be expressed as a polynomial $f(t)$, where $f(t)$ is a polynomial of t of degree less than n. As before, there is a correspondence between elements of $\mathbb{Z}_2^n$ and the set of all $f(t)$, and hence there is a correspondence between elements of $\mathbb{Z}_2^n$ and R_n.

To provide some motivation for discussion of this correspondence in relation to cyclic codes, note that if we are given any element of R_n and represent it in the form $f(t) = a_0+a_1t+\ldots+a_{n-2}t^{n-2}+a_{n-1}t^{n-1}$ we find that multiplying by t leads to $tf(t) = a_0t+a_1t^2+\ldots+a_{n-2}t^{n-1}+a_{n-1}t^n = a_{n-1} + a_0t + a_1t^2 + \ldots + a_{n-2}t^{n-1}$. Thus, $f(t)$ corresponds to $(a_0, a_1, \ldots a_{n-1}, a_{n-1})$ in $\mathbb{Z}_2^n$ and $tf(t)$ corresponds to $(a_{n-1}, a_0, a_1, \ldots a_{n-1})$ in $\mathbb{Z}_2^n$, so that multiplying an element in R_n by t effectively maps it to the element which represents its own n-tuple left shifted one position.

At this point, we are ready to show how we can use R_n to generate cyclic codes.

Theorem *A linear code with a set of codewords C in $\mathbb{Z}_2^n$ is cyclic if and only if C is an ideal in $R_n = \mathbb{Z}[x]/(x^n - 1)$.*
Proof: Let C be the set of codewords, i.e. $C = imE$, for a cyclic linear code and suppose that $\mathbf{x} = (a_0, a_1, \ldots, a_{n-1})$ is a codeword. We know that we can interpret $\mathbf{x}$ as $f(t)$, an element of R_n. Since this code is cyclic and $\mathbf{x} \in C$, we know that $(a_{n-1}, a_1, \ldots, a_{n-2}) \in C$.

This means that if $f(t) = a_0 + \cdots + a_{n-1}t^{n-1}$, then $tf(t) = a_{n-1} + a_0t + \cdots + a_{n-2}t^{n-1}$, the element which corresponds with $(a_{n-1}, a_1, \ldots, a_{n-2}) \in C$, is also a codeword. Thus, $t^kf(t)$ is in C for all $k \in \mathbb{N}$.

Now, recall that the codewords of a linear code form a group under addition. Combining this with the fact that $t^kf(t)$ is in C for all $k \in \mathbb{N}$, we can see that

$$f(t) + tf(t) + \cdots + t^{n-1}f(t) \in C.$$

In fact, since every element a_i in $\mathbb{Z}_2^n$ is either the multiplicative identity or 0, it also follows that, for any $a_0, \ldots, a_{n-1} \in \mathbb{Z}_2^n$,

$$a_0f(t) + a_1tf(t) + \cdots + a_{n-1}t^{n-1}f(t) \in C.$$

Letting $p(t) = a_0 + a_1t + \cdots + a_{n-1}t^{n-1}$, we find that

$$p(t)f(t) \in C \ \forall p(t) \in R_n.$$

Thus, we've shown that C is an ideal in R_n.

Now we will show the converse. Suppose $C \subset R_n$ is an ideal in R_n.
Let $f(t) = a_0 + a_1t + \cdots + a_{n-1}t^{n-1}$ be an element of C. Since t is an element of R_n, we know that $tf(t) \in C$ as well. Since $f(t)$ represents the codeword $(a_0, \ldots . a_{n-1})$ and $tf(t)$ represents $(a_{n-1}, a_1, \ldots, a_{n-2})$, we have therefore shown that C satisfies the cyclic property. □

Generating Ideals We now see that the question of how to generate cyclic linear codes boils down to generating ideals of R_n. Our knowledge of ring theory will help us do this.

We know that $\pi : \mathbb{Z}_2[x] \to R_n$ defined by $\pi[f(x)] = f(t)$ is the canonical homomorphism. Since every element of R_n is simply defined as $f(x) + (x^n - 1)$ for some $f(x) \in \mathbb{Z}_2[x]$, we can see that π is surjective. Additionally, $ker\pi = (x^n - 1)$, since precisely the polynomial multiples of $g(x) = x^n - 1$, these being the $f(x)$ of the form $f(x) = g(x) * h(x)$ for some $h(x) \in \mathbb{Z}_2[x]$, obey the property $\pi[f(x)] = f(x) + (x^n - 1) = 0$.

The ring correspondence theorem tells us that the ideals of some ring R which contain I, where I is some ideal of R, correspond uniquely with the ideals of the quotient ring R/I. This means that every ideal in R_n is of the form $\pi(J)$, where $(x^n - 1) \subseteq J$ and J is an ideal in $\mathbb{Z}_2$.

Additionally, we know that if F is a field, then every ideal of $F[x]$ is a principal ideal. $\mathbb{Z}_2$ is a field since 2 is prime. Thus, we can say that every ideal of R_n is the image in π of $(g(x))$, $(g(x))$ being the ideal in $\mathbb{Z}[x]$ generated by some function $g(x) \in \mathbb{Z}[x]$. Since the ideals of R_n correspond with the ideals J of $\mathbb{Z}_2[x]$ such that $(x^n - 1) \subset J$, we know that $g(x)$ necessarily divides $(x^n - 1)$. Thus, every ideal in R_n is of the form

$$\pi[(g(x))] = \{\pi[g(x)h(x)] \mid h(x) \in \mathbb{Z}_2[x]\} = \{g(t)h(t) \mid h(t) \in R_n\} = (g(t)),$$

where $g(x) \in \mathbb{Z}_2[x]$ is some polynomial which divides $x^n - 1$ and $g(t)$ is its image in R_n.

Example Consider the polynomial $x^5 - 1 \in \mathbb{Z}_2[x]$. We have

$$x^5 - 1 = (x + 1)(x^4 + x^3 + x^2 + x + 1).$$

$g(t) = t + 1$ generates an ideal in R_5. Let our encoding function $E : \mathbb{Z}_2^4 \to \mathbb{Z}_2^5$ be defined by $E(\mathbf{x}) = \mathbf{y}$, where $\mathbf{x}$ is a 4-tuple, and $\mathbf{y}$ is the 5-tuple represented by $f(t) * g(t)$, with $f(t)$ being the polynomial representation of $\mathbf{x}$. We have formed a $(5, 4)$-block code.

We can easily construct an $(5, 4)$-matrix to represent E by viewing $\mathbb{Z}_2^4, \mathbb{Z}_2^5$ as vector spaces. More specifically, we will consider $\mathbb{Z}_2^4$ as R_4 and $\mathbb{Z}_2^5$ as R_5. Consider how E acts upon each of the basis elements of $\mathbb{Z}_2^4$.

$E(1) = g(t) = t + 1$
$E(t) = tg(t) = t^2 + t$
$E(t^2) = t^2 g(t) = t^3 + t^2$
$E(t^3) = t^3 g(t) = t^4 + t^3$.

The following matrix, which we will call G, represents E:

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

Generator and Check Matrices The above examples lead to an interesting observation, namely that for any $n \in \mathbb{N}$ and $g(x), h(x) \in \mathbb{Z}_2[x]$ such that $x^n - 1 = g(x)h(x)$ where $deg(g(x)) = n - k$ and $deg(h(x)) = k$ there exists a $n \times k$ generator matrix for an (n, k)-code given by the following:

$$G = \begin{bmatrix} g_0 & 0 & 0 & 0 & \dots & 0 \\ g_1 & g_0 & 0 & 0 & \dots & 0 \\ g_2 & g_1 & g_0 & 0 & \dots & 0 \\ \vdots & \ddots & \ddots & \ddots & \ddots & \vdots \\ g_n - k & g_{n-k-1} & \dots & \dots & \dots & g_0 \\ 0 & g_{n-k} & \ddots & \ddots & \ddots & g_1 \\ \vdots & \ddots & \ddots & \ddots & \ddots & \vdots \\ 0 & 0 & \dots & \dots & 0 & g_{n-k} \end{bmatrix}$$

The $a_{i,j}$ entry of G is given by g_{i-j}, where $g_m = 0$ for $m < 0$ and $m > n - k$. Using this matrix, it is possible to map any element in $\mathbb{Z}_2^k$ to an element in $\mathbb{Z}_2^n$, and the resulting encoding function that this matrix gives yields a cyclic code by all that has previously been shown.

However, another particularly useful aspect of these types of codes is their error-checking capabilities by way of the $(n - k) \times n$ matrix H given below:

$$H = \begin{bmatrix} 0 & \dots & 0 & 0 & h_k & \dots & h_0 \\ 0 & \dots & 0 & h_k & \dots & h_0 & 0 \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ h_k & \dots & h_0 & \dots & \dots & 0 & 0 \end{bmatrix}$$

The $b_{i,j}$ entry of H is given by $h_{(n+1)-i-j}$, where $h_m = 0$ for $m < 0$ and $m > k$. Using this matrix, it is possible to perform efficient error detection as an n-bit string $\mathbf{y}$ is a valid codeword in the image of G if and only if $H\mathbf{y} = \mathbf{0}$.

While a strict proof will not be given, there is an intuitive explanation why any codeword $\mathbf{y}$ must be in the null space of H. To see this, first note that necessarily there exists some $\mathbf{x}$ in $\mathbb{Z}_2^k$ such that $G\mathbf{x} = \mathbf{y}$. Thus, $H\mathbf{y}$ can be represented in the form $HG\mathbf{x}$. However, note that to compute HG, each row of H can be seen as representing the polynomial $t^a h(t)$ for some a while each column of G can be seen as representing the polynomial $t^b g(t)$ (as they are the tuples corresponding to the coefficients of these polynomials shifted a and b positions respectively). Based on the condition that $g(x)h(x) = x^n - 1$, it follows that $(t^a h(t))(t^b g(t)) = t^a t^b h(t)g(t) = (t^a t^b)(0) = 0$. As this occurs for each row/column multiplied, it follows that $HG = \mathbf{0}$, and thus $H\mathbf{y} = HG\mathbf{x} = \mathbf{0}$ as desired.

As a result of the above, if $H\mathbf{x}$ is computed and is not the zero matrix, it follows that it is not a valid codeword and hence at least one error occurred. Error correction is then necessary to determine the most likely codeword the altered message represents.

Minimum Distance of Polynomial Codes We now have the tools to determine the approximate error-detecting and error-correcting capabilities of the polynomial code generated by any given $g(x) \in \mathbb{Z}_2[x]$. Coming up with a way to find the lower bound of d_{min}, the minimum distance of the code, will allow us to do this.

Theorem Let $C = (g(t))$ be the set of codewords of a cyclic code in R_n and suppose that ω is a primitive n-th root of unity over $\mathbb{Z}_2$. If s consecutive powers of ω are roots of $g(x)$, then the minimum distance of our code is at least $s + 1$.
Proof: Suppose that we have

$$g(\omega^r) = g(\omega^{r+1}) = \cdots = g(\omega^{r+s-1}) = 0.$$

Let $f(x) \in C$ be some polynomial with s or fewer nonzero coefficients. Recall that $d_{min} = \{d(\mathbf{x}, \mathbf{y}) : \mathbf{x} \neq \mathbf{y}\}$.

$\implies$ $d_{min} = \{d(\mathbf{x}, \mathbf{y}) : \mathbf{x} + \mathbf{y}) \neq 0\}$, since the subtraction operation is equivalent to the addition operation in $\mathbb{Z}_2^n$.

$\implies$ $d_{min} = \{w(\mathbf{x} + \mathbf{y}) : \mathbf{x} + \mathbf{y}) \neq 0\}$, where $w(\mathbf{x} + \mathbf{y})$, the weight of $\mathbf{x} + \mathbf{y}$, is defined to be the number of nonzero digits of the n-tuple. This is again due to the fact that subtraction is equivalent to addition; we are essentially calculating the distance between $\mathbf{x} - \mathbf{y}$ and 0.

$\implies$ $d_{min} = \{w(\mathbf{z}) : \mathbf{z} \in C\}$, since every element of $\mathbb{Z}_2^n$ can be expressed as the sum of two other elements.

$\implies$ $d_{min} = w_{min}$, the minimum weight of any nonzero element in our set of codewords

C.

Thus, for the purpose of proving that $d_{min} > s + 1$, it suffices to show that any element of C with weight less than $s + 1$ must necessarily be the zero element. In other words, we wish to show that all of the a_i's, the coefficients of $f(x)$, are zero.

Since $g(\omega^r) = g(\omega^{r+1}) = \cdots = g(\omega^{r+s-1}) = 0$ and we know that every element in $(g(x))$ has $g(x)$ as a factor, necessarily

$$f(\omega^r) = f(\omega^{r+1}) = \cdots = f(\omega^{r+s-1}) = 0.$$

Consider the following system of equations:

$$\begin{aligned}
(\omega^{i_0})^r x_0 + (\omega^{i_1})^r x_1 + \cdots + (\omega^{i_{s-1}})^r x_{s-1} &= 0 \\
(\omega^{i_0})^{r+1} x_0 + (\omega^{i_1})^{r+1} x_1 + \cdots + (\omega^{i_{s-1}})^{r+1} x_{s-1} &= 0 \\
&\vdots \\
(\omega^{i_0})^{r+s-1} x_0 + (\omega^{i_1})^{r+s-1} x_1 + \cdots + (\omega^{i_{s-1}})^{r+s-1} x_{s-1} &= 0
\end{aligned}$$

Clearly $x_0 = x_1 = \cdots = x_{s-1} = 0$ is a solution to this system of equations. Furthermore, representing this system of equations in matrix form,

$$\begin{bmatrix}
(\omega^{i_0})^r & (\omega^{i_1})^r & \dots & (\omega^{i_{s-1}})^r \\
(\omega^{i_0})^{r+1} & (\omega^{i_1})^{r+1} & \dots & (\omega^{i_{s-1}})^{r+1} \\
\vdots & \vdots & \ddots & \vdots \\
(\omega^{i_0})^{r+s-1} & (\omega^{i_1})^{r+s-1} & \dots & (\omega^{i_{s-1}})^{r+s-1}
\end{bmatrix}$$

it follows from properties of the Vandermonde matrix as well as properties of the determinant that this matrix has a nonzero determinant, and hence this must be the unique solution. Thus, the minimum distance between any two elements in C is $s + 1$. □

Recall that knowledge of d_{min} has concrete implications with regards to a linear code's error-detecting and error-correcting capabilities per the section on hamming distance.

6 Conclusion

We now have some understanding of what polynomial codes are, how to construct them, and how to determine their error-detecting and error-correcting capabilities. While group theory on its own allows for the development of codes that can correct more than a single error, the error-correcting capabilities of polynomial codes are much more sophisticated.

As we have just shown, it is vital that we choose a generating polynomial, $g(t)$, with specific properties, as these are what determine a polynomial code's effectiveness. However, the choice of generating polynomial is far from a trivial matter, and their study continues to be an active research area. We encourage the reader to further explore coding theory, and we hope that this paper has provided enough of a grounding in the subject in order to do so.

References

[1] Edward Chang and Richard Taborek. Recommendation of 10^{-13} bit error rate for 10 gigabit ethernet, 1999.

[2] Tom Judson. *Abstract Algebra: Theory and Applications*. Open Source, 2020.

[3] George Markowsky. Information theory, Jun 2017.

[4] Richard Pinch. Coding theory: the first 50 years, Sep 1997.