# Coding Theory Lecture Notes

Nathan Kaplan and members of the tutorial

September 7, 2011

These are the notes for the 2011 Summer Tutorial on Coding Theory. I have not gone through and given citations or references for all of the results given here, but the presentation relies heavily on two sources, van Lint's *Introduction to Coding Theory* and the book of Huffman and Pless *Fundamentals of Error-Correcting Codes*. I also used course notes written by Sebastian Pancratz from a Part II course given at Cambridge on Coding Theory and Cryptography given by Professor Tom Fisher, and my own course notes from a Cambridge Part III Arithmetic Combinatorics course given by Professor Tim Gowers. I also included notes from Henry Cohn's MIT course, Applied Math for the Pure Mathematician. The last section of the notes is based on Nigel Boston's survey article, "Graph Based Codes".

All mistakes in these notes are my fault.

## Contents

# 1 Introduction

Thanks to Tony Feng and Joseph Moon for help typing this section

## 1.1 The Main Problem of Coding Theory

Suppose two parties are trying to communicate over a noisy channel. Consider a first example. All we want to do is send a single bit as our message, $\{0\}$ or $\{1\}$. When we send a bit there is a probability $p$ that the bit received does not match the bit sent. The main problem of coding theorem can be phrased as follows:

**How do we build in redundancy into messages so that the errors introduced by the channel can be identified and corrected?**

**Example.** One solution is to just send chunks of bits repeated many many times. For example, suppose we agree to send 000 or 111, and we receive 001. We know that we have at least one error. The probability that we send 000 and receive 001 is $(1-p)^2p$, and the probability that we send 111 and receive 001 is $p^2(1-p)$. If $p < \frac{1}{2}$ then it is likelier that we sent 000 than that we sent 111. We decode this message as 000.

This is a type of decoding where we take our received word as given and have each bit 'vote' on what the message sent was. We call this 'Maximum Likelihood Decoding'. Given our received word, we determine which of the message words is most likely to have been sent.

Next we ask for the probability that we make an error in decoding. We do this exactly in the case where there are two or more errors. The probability of this occurring is:

$$\binom{3}{2}(1-p)p^2 + \binom{3}{3}p^3 \approx 3p^2 \ll p,$$

since when $p$ is small, the $p^3$ terms are much smaller than our $p^2$ term. If we do no encoding and just send a single bit then the probability it is received incorrectly is $p$. Therefore, we see that when $p$ is small, this repetition helps us decode correctly.

We can also ask for the expected number of errors after decoding. If we have any errors then we must have three. So the expected number is

$$3(3(1-p)p^2 + p^3) \ll p,$$

where $p$ is the expected number of errors with no encoding.

More generally, suppose we agree to send each bit $n$ times, where for convenience we assume that $n$ is odd. With the same analysis as above, we see that the probability of decoding incorrectly is

$$\binom{n}{\frac{n+1}{2}}p^{\frac{n+1}{2}}(1-p)^{\frac{n-1}{2}} + \ldots + \binom{n}{n}p^n \approx \binom{n}{\frac{n-1}{2}}p^{\frac{n+1}{2}},$$

when $p$ is small.

For example, when $n = 5$ this probability is $p^3(6p^2 - 15p + 10)$. In the exercises you will show that when $p < \frac{1}{2}$ the probability of making an error after decoding approaches 0 as $n$ goes to infinity.

**Definition.** The subset of $\{0,1\}^n$ consisting of the two words $(0,0,\ldots,0)$ and $(1,1,\ldots,1)$ is known as the binary repetition code of length $n$.

There is a clear tradeoff here. We can decrease our probability of decoding incorrectly at the price of sending longer and longer transmissions.

Suppose we have an *alphabet* of $q$ symbols. Usually we will take $q$ to be a prime power and the set of symbols will correspond to the elements of the finite field of $q$ elements. We will discuss finite fields more in the next lecture. For today we will always take $q = 2$.

**Definition.** An error-correcting code is a subset of $\mathbb{F}_q^n$. We will send messages in blocks of $n$ symbols from our alphabet. This gives a block code of length $n$.

**Definition.** Let $u = (u_1, \dots, u_n)$ and $v = (v_1, \dots, v_n)$. The Hamming distance $d(u, v)$ is defined as

$$d(u, v) = \#\{i \mid u_i \neq v_i\}.$$

The weight of a word is its distance from 0.

**Example.** Suppose we want to send a 3-bit message $x_1, x_2, x_3$. We also send three extra check-bits, defined as $x_4 := x_2 + x_3,\ x_5 := x_3 + x_1$ and $x_6 := x_1 + x_2$.

The resulting messages form a code of length 6. There are 8 codewords since given any $(x_1, x_2, x_3)$ the other three bits are determined.

Suppose we transmit $c$ and receive $b := c + e$. We call this vector $e$ the the error pattern. We will see that there is a straightforward way to decode received messages, giving a code better than the repetition code in some sense. Since $x_4 + x_2 + x_3 = 0$, we see that $b_2 + b_3 + b_4 = e_4 + e_2 + e_3$. Similar relations hold for the other three check-bits. We define

$$\begin{aligned} s_1 &= b_2 + b_3 + b_4 = e_2 + e_3 + e_4 \\ s_2 &= b_1 + b_3 + b_5 = e_1 + e_3 + e_5 \\ s_3 &= b_1 + b_2 + b_6 = e_1 + e_2 + e_6. \end{aligned}$$

Given $(s_1, s_2, s_3)$ we need to choose the most likely error pattern $(e_1, \dots, e_6)$ which leads to this 3-tuple. It is easy to see that the most likely error pattern is the one with the smallest weight.

If $(s_1, s_2, s_3) \neq (1, 1, 1)$ then there's a unique $e$ of lowest weight which gives $(s_1, s_2, s_3)$. For example $(s_1, s_2, s_3) = (1, 0, 0)$ gives $e = (000100)$. When $(s_1, s_2, s_3) = (1, 1, 1)$, there are three equally likely possibilities: $\{(100100), (010010), (001001)\}$. Suppose we agree to choose the smallest weight error when there is one and fix an arbitrary choice otherwise. What's the probability of decoding correctly? If there are no errors or just one, we are always correct; if there are two, we are correctly in only one of three cases. So,

$$\text{Prob(decoding correctly)} = (1-p)^6 + 6(1-p)^5 p + (1-p)^4 p^2 \gg (1-p)^3.$$

For example, when $p = \frac{1}{10}$ the left hand side is 0.892 while the right hand side is 0.729.

This code and the repetition code both help us to correct errors in transmission, but we would like some way to say which one is better. One important measure of the effectiveness of a code is the rate.

**Definition.** The rate $R$ of a code of length $n$ with an alphabet of $q$ symbols is defined as

$$R = \frac{\log_q |C|}{n}.$$

If $|C| = q^k$, then $R = \frac{k}{n}$.

For example, we see that the rate of the repetition code of length $n$ is $\frac{1}{n}$. The rate of the code of length 6 defined above is $\frac{3}{6} = \frac{1}{2}$.

We want three things out of a good code.

1. We want a large *minimum distance*. For example if $d = 2e + 1$ we can then automatically correct errors of weight $\leq e$.

2. We want $|C|$ to be as large as possible so that we can send many possible messages.

3. We want good encoding and decoding algorithms.

This third point is a little tricky. Usually given a message deciding how to send it as a codeword, the encoding problem, is easy. Decoding is often difficult. Given a received message in $\mathbb{F}_q^n$ how do we determine which codeword is closest to it? If our code has no structure then this is often a very difficult problem, but when our code does have structure, we can often exploit it to solve this problem.

If you are not convinced that finding the closest codeword can be difficult, think about this analogous problem in Euclidean space. Given an infinite set $S$ of points in $\mathbb{Z}^n$ and any other point $p$ in $\mathbb{Z}^n$, how do we determine which point of $S$ is closest to $p$? The following version of the problem sounds easier, but is still difficult. Given an $n$-dimensional subspace of $\mathbb{Z}^n$, which we can think of as being given by an $n \times n$ integer matrix with rows giving generators for this subspace, and any point $p \in \mathbb{Z}^n$, how do we determine which point in our space is closest to $p$? This is equivalent to finding the nonzero point in our space closest to 0. This is the shortest vector problem for lattices, and the exact version is NP-complete, (although there is a very interesting approximate solution given by the LLL-algorithm).

## 1.2 Shannon's Theorem

In most math classes you will take at Harvard you will not learn anything proven in the past 100 years. This is emphatically not true for coding theory, which is a very young subject. It really only goes back to 1948 or so and Claude Shannon's landmark paper 'A Mathematical Theory of Communication'. A quick search reveals that this paper has been cited over 25,000 times. This paper is seen as the foundation of Information Theory, an area quite related to the theory of error correcting codes.

Shannon asked whether there is some theoretical limit to how well we can solve this coding problem. Consider all binary codes of rate $R$. Can we find one so that the probability of making an error in decoding is as small as we want?

**Theorem 1** (Shannon, 1948). *Let $c_1, \ldots, c_{|C|}$ be our codewords. Let $P_i$ be the probability that if $c_i$ is sent, it is decoded incorrectly.*

$$P_C = \frac{\sum_{i=1}^{|C|} P_i}{|C|}$$

*Let $P^*(|C|, n, p)$ be the minimum value of $P_C$. If $0 < R < 1 + p\log p + (1-p)\log(1-p)$, and $M_n = 2^{\lfloor R \cdot n \rfloor}$,*

$$P^*(M_n, n, p) \to 0 \quad \text{as } n \to \infty.$$

This says that if the rate is less than the channel capacity (which is the term $1+p\log p+(1-p)\log(1-p)$), then for sufficiently large length $n$ there exist codes of rate $R$ with arbitrarily small probability of decoding error. There is a sort of converse to this theorem as well. When the rate is larger than the channel capacity, we cannot get the probability of a decoding error to be arbitrarily small. Shannon's theorem tells us that good codes exist, but gives no indication of how to construct them. We will return to Shannon's theorem in about a week and a half.

Another of the earlier founders of coding theory was Richard Hamming, a colleague of Shannon's at Bell Labs. He was less interested in the theoretical limits of coding theory than in actually building codes that effectively corrected errors to be used in practice. They did work for the telephone company, after all. This course will focus mostly on constructing codes with interesting properties, the Hamming point of view.

Coding theory is a subject that did develop in part to address practical communication problems, such as transmission of images from deep space. Hamming also wrote a fundamental paper around the same time as Shannon, 'Error Detecting and Error Correcting Codes'. We will see the codes named for Hamming later in this lecture.

## 1.3 Linear Codes

**Definition.** A linear code $C$ of dimension $k$ is a $k$-dimensional linear subspace of $\mathbb{F}_q^n$. In particular, this says that $u, v \in C$ implies $u + v \in C$.

Notation: if the code is linear, it is denoted as $[n, k, d]$. Otherwise, it is denoted as $(n, M, d)$, where $M$ is the number of words in the code.

**Example.** The repetition code is $[n, 1, \lfloor \frac{n-1}{2} \rfloor]$. The second code we considered as $[6, 3, 3]$.

We can see that this code is linear by writing down a basis for it. One option is

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

**Definition.** A generator matrix for $C$ is a $k \times n$ matrix whose rows form a basis for the elements of $C$.

Note that by elementary row and column operators, we can always write a generator matrix in 'standard form' $[I_k \mid A]$.

A linear subspace of $\mathbb{F}_q^n$ (or more generally, any vector space) is the kernel of a linear transformation, so there exists an $(n-k) \times n$ matrix $H$ such that

$$C = \{x \in \mathbb{F}_q^n \mid Hx^T = 0\}.$$

This is the **parity check** matrix of $C$.

**Example.** For our $[n, 1, \lfloor \frac{n-1}{2} \rfloor]$ code, the parity check matrix is a column of 1s followed by the $(n-1)\times(n-1)$ identity matrix. For the $[6, 3, 3]$ code, the parity check matrix is

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

Notice that this is just a rearragement of the generator matrix!

**Definition.** Two codes $C_1, C_2$ are permutation equivalent if there is a permutation of $\{1, \ldots, n\}$ which sends $C_1$ to $C_2$.

The automorphism group of a code $\mathrm{Aut}(C) \subset S_n$ is the set of all permutations of the coordinates $\{1, \ldots, n\}$ which fix the elements of $C$ as a set.

We see that the automorphism group of the repetition code is all of $S_n$. We also see that the $[6, 3, 3]$ code is permutation equivalent to its dual.

**Theorem 2.** *If $G = [I_k \mid A]$ is a generator matrix for an $[n, k]$ code, then*

$$H = [-A^T \mid I_{n-k}].$$

*Proof.* We first note that we can express $C$ as the set of all length $n$ vectors $xG$ as we vary over all all possible vectors $x \in \mathbb{F}_q^n$.

Note that $HG^T = -A^T + A^T = 0$, so $C$ is contained in the kernel of the map defined by $H$ since $H(xG)^T = HG^T x^T = 0$.

Since $H$ has rank $n-k$, the dimension of the kernel is $k = \dim C$, implying that $C$ is exactly equal to the kernel. $\square$

**Theorem 3.** *For a linear code $C$, the minimum distance between any two codewords in $C$ is equal to the minimum weight vector in $C$.*

*Proof.* Since $C$ is a linear code, $d(x, y) = d(x-y, 0) = \mathrm{wt}(x-y)$. $\square$

**Definition.** Let $C$ be an $[n, k]$ code with generator matrix $G$ and parity check matrix $H$. The dual of $C$, denoted $C^\perp$, is the $[n, n-k]$ code which has generator matrix $H$ and parity check matrix $G$.

Note that this does not imply that $C \cap C^\perp = \emptyset$. In fact, there will be interesting cases where $C = C^\perp$.

Here is another way to think of the dual. Consider the regular dot product on $\mathbb{F}_q^n$. We have

$$C^\perp = \{x \in \mathbb{F}_q^n \mid x \cdot c = 0 \text{ for all } c \in C\}.$$

So actually, this is and orthogonal complement with respect to the dot product.

**Example.** The dual of the $[n, 1]$ repetition code is the set of all vectors of $\mathbb{F}_q^n$ of even length. We can check that the $[6, 3, 3]$ code is permutation equivalent to its dual with this interpretation.

## 1.4 Hamming Codes

**Definition.** The binary Hamming code of length $2^n - 1$ has as its parity check matrix all nonzero vectors in $\mathbb{F}_2^n$.

The $q$-ary Hamming code of length $\frac{q^k-1}{q-1}$, denoted $\mathcal{H}_{q,k}$, has as its parity check matrix one nonzero vectors from each one dimensional subspace of $\mathbb{F}_q^k$. We can take all nonzero $q$-ary vectors of length $k$ and then not include any two which differ only by a scalar factor.

**Example.** The Hamming code $[7, 4]$ is given by the generator and parity check matrices,

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

and

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

Note that the $[6, 3, 3]$ code has generator matrix equal to the first three rows of $G$ minus the fourth column which now has all zeros.

**Definition.** The extended binary Hamming code comes from adjoining an extra column to the generator matrix so that the weight of each row is even.

More generally, let $C$ be a linear code with generator matrix $G$. The extended code is defined by adding a column to the generator matrix so that the sum of the entries of each row of $G$ is zero in $\mathbb{F}_q$.

We form the extended Hamming code $[8, 4, 4]$ by adding an overall parity check bit column $[1, 1, 1, 0]^T$. As an exercise, you should show that this code is self-dual.

In general, the dual of the $q$-ary Hamming code of length $n$ is called the $q$-ary simplex code of length $n$. In the exercises you will investigate these codes a little. You will also compute the expected number of errors after decoding using the extended binary Hamming code.

## 1.5 The MacWilliams Theorem and Gleason's Theorem

For the rest of this lecture and some of the next, the idea is to give some of the highlights that will arise in the rest of the course.

**Definition.** The weight distribution of a code $C$, is given by $\{A_0, \ldots, A_n\}$ where

$$A_i := \#\{c \in C \mid \mathrm{wt}(c) = i\}.$$

Note that a linear code always has $A_0 = 1$.

The weight enumerator is the homogeneous degree $n$ polynomial in $x, y$

$$\sum_{i=0}^{n} A_i x^{n-i} y^i.$$

**Example.** For the $[n, 1]$ repetition code the weight enumerator is $x^n + y^n$. For the $[6, 3, 3]$ code, the weight enumerator is $x^6 + 4x^3y^3 + 3x^4y^2$. For the $[8, 4, 4]$ code, the weight enumerator is $x^8 + 14x^4y^4 + y^8$.

**Theorem 4** (MacWilliams)**.** *Let $W_C(x, y)$ be the weight enumerator of $C$. Then*

$$W_{C^\perp}(x, y) = \frac{1}{|C|} W_C(x + (q-1)y, x - y).$$

The MacWilliams theorem says that you can determine the weights enumerator of a dual code from the weight enumerator of a code. This can be very useful in practice. There are codes for which it is much easier to compute the weight enumerator of one than the other.

We will prove this in about two weeks. There are many different generalizations of the MacWilliams Theorem, (some of which even come up in my own grad student life), and pursuing these would make for a good final paper topic.

We have already seen some examples of codes which are self-dual, or at least permutation equivalent to their duals. What can we say about such codes? For example, what can we say about the weight enumerators?

**Theorem 5** (Gleason)**.** *Let $g_1(x,y) = x^2 + y^2$ and $g_2(x,y) = x^8 + 14x^4y^4 + y^2$. If $C$ is a binary $[n, \frac{n}{2}]$ with even weight codewords, then the weight enumerator is a linear combination of $g_1$ and $g_2$:*

$$W_C(x,y) = \sum_{i=0}^{\lfloor \frac{n}{8} \rfloor} a_i g_1(x,y)^{\frac{n}{2}-4i} g_2(x,y)^i,$$

*where the $a_i$ are rational and $\sum a_i = 1$.*

Andrew Gleason was a long time member of the Harvard math department, and is probably will be the last person to serve as president of the American Mathematical Society and not have a Ph.D. This is supposed to inspire you to go look him up online. Self-dual codes have been studied extensively. Gleason's Theorem also has many generalizations which would make for a good paper topic.

## 1.6 Finite Fields

The finite field of $p$ elements for a prime $p$ is probably familiar to everyone. It is just $\mathbb{Z}/p\mathbb{Z}$ where arithmetic is done modulo $p$.

**Proposition 1.** *We have $\mathbb{Z}/p\mathbb{Z}$ is a field of $p$ elements.*

*Proof.* There are two short ways to see this. First, we note that $\mathbb{Z}$ is a ring and $p\mathbb{Z}$ is a maximal ideal of $\mathbb{Z}$. We recall that a ring modulo a maximal ideal is a field.

Second, we recall the following fact. In a finite ring if $ab = 0$ implies that either $a = 0$ or $b = 0$, then the ring is a field. Now by the definition of a prime, this condition holds and $\mathbb{Z}/p\mathbb{Z}$ is a field. $\square$

Next we will construct a field of $p^r$ elements. We will usually just let $q = p^r$ and write $\mathbb{F}_q$. Consider the polynomial ring $\mathbb{F}_p[x]$ and take an irreducible polynomial $g(x)$ of degree $r$. Next consider $\mathbb{F}_p[x]/g(x)$. This is a field since $g(x)$ is an irreducible polynomial. This is because the ideal generated by an irreducible polynomial is prim in $\mathbb{F}_p[x]$. Equivalently, $a(x) \cdot b(x) = 0$ in $\mathbb{F}_p[x]/g(x)$ implies g(x) divides $a(x)$ or $b(x)$.

Representatives for this quotient are given by all polynomials of degree strictly less than $r$. This is because if we have any polynomial in $x$ of degree at least $r$, we can subtract some multiple of $g(x)$ and get a polynomial of degree less than $r$. This remainder is unique. So $\mathbb{F}_p[x]/g(x)$ is a finite field with $p^r$ elements.

**Theorem 6.** *If $\mathbb{F}$ is a finite field with $n$ elements, then $n$ is a prime power.*

*Proof.* Since $\mathbb{F}$ is a field we have a multiplicative identity 1. Add 1 to itself and name every element we have not seen before, for example call $1+1$, '2'. We add 1 to itself $k$ times until we come across some element $l$ that we have seen before. This shows $k - l = 0$. It is also easy to see that $l$ must be equal to zero, since otherwise we would have already come across the element 0 after adding 1 to itself $k - l$ times. If $k$ is composite, say $k = ab$, then $k = ab = 0$ but $a, b \neq 0$ by the definition of $k$. This contradicts $\mathbb{F}$ being a field, so we can suppose $k$ is prime. This is called the characteristic of the field. This shows $\mathbb{F}_p \subset \mathbb{F}$.

Now take a maximal subset of linearly independent elements of $\mathbb{F}$, $\{k_1, \ldots, k_r\}$. That is, a maximal subset such that

$$a_1k_1 + \cdots + a_rk_r = 0, \text{ with } a_1, \ldots, a_r \in \mathbb{F}_p \text{ implies that } a_1 = \cdots = a_r = 0.$$

Now take any $x \in \mathbb{F}$. Since $\{x, k_1, \ldots, k_r\}$ is not linearly independent we can write

$$x + a_1 k_1 + \cdots + a_r k_r = 0$$

for some collection of $a_i \in \mathbb{F}_p$. We see that such a representation is unique since $\{k_1, \ldots, k_r\}$ are linearly independent. Therefore, since there are exactly $p^r$ tuples $(a_1, \ldots, a_r)$ we see that $|\mathbb{F}| = p^r$. □

Now, to show that there exists a finite field of order $p^r$, we need to have an irreducible polynomial of degree $r$. In fact, we will count the number of such irreducible polynomials. We will use the Möbius Inversion Formula. First we recall a definition.

**Definition.** The Möbius function is defined by

$$\mu(n) = \begin{cases} 1 & n = 1 \\ (-1)^k & n = p_1 \cdots p_k \text{ distinct primes} \\ 0 & \text{otherwise} \end{cases}.$$

**Theorem 7.** *If $f, g$ are functions on $\mathbb{N}$ such that $g(n) = \sum_{d|n} f(d)$ then $f(n) = \sum_{d|n} \mu(d) g(\frac{n}{d})$.*

**Theorem 8.** *Fix a prime $p$. Let $I_r$ be the number of monic irreducible polynomials over $\mathbb{F}_p$ of degree $r$. Then $r \geq 1$.*

**Corollary 1.** *Given a prime $p$ and any $r \geq 1$, there exists a finite field of order $p^r$.*

*Proof.* Recall that as a formal power series $\frac{1}{1-x} = 1 + x + x^2 + \cdots$. We claim that

$$\frac{1}{1-pz} = \prod_{r=1}^{\infty} \frac{1}{(1-z^r)^{I_r}}.$$

We compare coefficients on both sides. On the left hand side the coefficient of the $z^n$ term is $p^n$. On the right hand side we note that each monic polynomial of degree $n$ factors into irreducibles in a unique way and that there are exactly $p^n$ monic polynomials of degree $n$.

Now we take the logarithm of both sides and differentiate, recalling that $(\log f)' = \frac{f'}{f}$. Multiplying by $z$ gives

$$\frac{pz}{1-pz} = \sum_{r=1}^{\infty} r I_r \frac{r z^r}{(1-z^r)}.$$

We now equate the $z^n$ coefficients of both sides and see that $p^n = \sum_{r|n} r \cdot I_r$. We apply Möbius Inversion and get

$$I_r = \frac{1}{r} \sum \mu(d) p^{r/d} > \frac{1}{r} \left( p^r - (p^{r/2} + p^{r/3} + \cdots + p^1) \right).$$

Noting $p^r - (1 + p + \cdots p^{\frac{r}{2}}) \geq p^r(1 - p^{1-\frac{r}{2}}) > 0$, when $r > 2$, and that $\sum \mu(d) p^{r/d} = p^2 - p > 0$ when $r = 2$, completes the proof. □

Here is an aside about why I like Möbius inversion so much.

**Definition.** The $n$th cyclotomic polynomial $\Phi_n(x)$ is the unique monic polynomial whose roots are the primitive $n$th roots of unity.

We note that $x^n - 1 = \prod_{d|n} \Phi_d(x)$. We now see that we can express $\Phi_n(x)$ using the Möbius Inversion formula. As an undergraduate I had a summer research project about coefficients of these polynomials.

# 2 Hamming Codes, Hadamard Codes and Combinatorial Designs

Thanks to Joseph Moon and Tony Feng for help typing this section

## 2.1 Minimal Weights and Perfect Codes

How do we actually determine the minimal weight of a code? Here is a proposition I should have proved in the first lecture.

**Proposition 2.** *Let $H$ be a parity check matrix for a linear code $C$. The minimum distance of $C$ the minimum number of columns of $H$ that are linearly dependent.*

*Proof.* Let $h_1, \dots h_n$ be columns of $H$. Let $c \neq 0$ be a codeword of weight 0 with nonzero entries in positions $1 \leq i_1 \leq i_w$. Equivalently, we say that the support of $c$ is the set $\{i_1, \dots, i_w\}$. Since $Hc^T = 0$, we have

$$c_{j_1} h_{j_1} + \cdots c_{j_w} h_{j_w} = 0.$$

Therefore this collection of $w$ columns is linearly dependent.

Now suppose that $h_{j_1}, \dots, h_{j_w}$ are dependent. So there exists nonzero elements $a_{j_1}, \dots, a_{j_w}$ of $\mathbb{F}_q$ such that $\sum_{i=1}^{w} a_{j_i} h)j_i = 0$. We construct a codeword $c$ which satisfies $c_{j_i} = a_{j_i}$ and is zero in all other positions. This is a codeword of weight $w$. □

**Example.** We see that the minimum distance of the $q$-ary Hamming code of dimension $k$ is 3. We know that the rows of the parity check matrix are representatives of 1-dimensional subspaces of $\mathbb{F}_q^k$. Therefore, any pair of columns is independent. However, given columns $h_1$ and $h_2$, their sum generates a 1-dimensional subspace distinct from these two spaces, and so there is another column $h_3$ so that $h_1, h_2$ and $h_3$ are linearly dependent. We can use this type of reasoning to determine the weight enumerator of the $q$-ary Hamming code.

**Definition.** Given $x \in \mathbb{F}_q^n$ we define the sphere (or more accurately the ball) of radius $r$ around $x$ as

$$B_r(x) = \{y \in \mathbb{F}_q^n : d(x, y) \leq r\}.$$

We let $V_q(n, r) = |B_r(x)|$ denote the size of the ball.

**Proposition 3.** *We have $V_q(n, r) = \sum_{i=0}^{r} \binom{n}{i} (q - i)^i$.*

*Proof.* We count the number of words which are distance exactly $i$ from $x$. There are $\binom{n}{i}$ ways to choose the $i$ positions that will be different and for each of these positions there are $q - 1$ choices for which symbol will be in that position. □

**Definition.** Let $A(n, d)$ be max number of codewords in a code of length $n$, minimum distance $d$.

The author of my favoriate coding theory textbook, J.H. van Lint, refers to establishing bounds on $A(n, d)$ as "the most important problem in combinatorial coding theory". Discussing upper and lower bounds will be a major topic in the course, with the Linear Programming Bound being the highlight. We first give one of the simplest possible bounds.

**Proposition 4** (Sphere Packing Bound)**.** *We have*

$$A(n, d) \leq \frac{q^n}{V_q(n, \lfloor \frac{d-1}{2} \rfloor)}.$$

*Proof.* Let $C$ be a code of length $n$ and minimum distance $d$. By assumption we can correct up to $\lfloor \frac{d-1}{2} \rfloor$ errors, so the spheres of radius $\lfloor \frac{d-1}{2} \rfloor$ around each codeword are disjoint. We see that the union of the sizes of these spheres is $|C| V_q(n, \lfloor d - 12 \rfloor)$, completing the proof. □

**Definition.** When equality holds in the above proposition, that is $|C| = \frac{q^n}{V_q(n, \lfloor \frac{d-1}{2} \rfloor)}$, we say that $C$ is a perfect $\lfloor \frac{d-1}{2} \rfloor$-error correcting code.

We somewhat trivially note that the codes consisting of all words of $\mathbb{F}_q^n$ and the code consisting of only the zero word are perfect, but correct no errors. A binary repetition code of odd length is perfect.

**Proposition 5.** *The $q$-ary Hamming code of length $n = \frac{q^r-1}{q-1}$ is perfect.*

*Proof.* We recall that size number of rows of the parity check matrix is $k$, so the dimension of the code is $n-r$. Also, we have already seen that these codes have minimum distance 3, so are 1-error correcting. We see that

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

□

Perfect codes give us codes with as many words as possible given a fixed length and minimum distance, so satisfy one of the main things we want out of our codes in the best possible way. Much energy has gone into classifying perfect codes, and the following theorem will be a highlight of the course.

**Theorem 9** (Tietäväinen, van Lint)**.** *If $C$ is binary perfect $e$-error correcting code, $e > 1$, then either $C$ is a repetition code, or binary Golay code.*

The Golay code mentioned in this theorem presents another highlight of this course. It is a perfect 3-error correcting linear code of length 23 and dimension 12. The extended binary code is also self-dual. This amazing highly symmetric object is also very important in the study of combinatorial designs, which is the next subject in this lecture. It also has an amazing automorphism group, one of the sporadic simple groups. We will discuss this code in a few weeks.

Before we move on to a new subject we will give a lower bound for $A(n,d)$.

**Proposition 6** (Gilbert Bound)**.** *We have*

$$A(n,d) \geq \frac{q^n}{V_q(n,d-1)}.$$

*Proof.* Let $C$ be a length $n$ minimum distance $d$ code with $M$ codewords, where $M$ is maximal among all such codes. No word in $\mathbb{F}_q^n$ is distance at least $d$ from every codeword because then we could add it to $C$ and get a length $n$ minimum distance $d$ code with $M+1$ words. Therefore if we put a ball of radius $d-1$ around each codeword in $C$, we must cover all of $\mathbb{F}_q^n$. □

## 2.2 Hadamard Matrices and Combinatorial Designs

Let $a_1, \ldots, a_N$ be columns of an $N \times N$ real matrix and let $A$ be the corresponding matrix. How large can the determinant of $A$ be?

**Theorem 10** (Hadamard)**.** *We have*

$$|\det A| \leq \prod_{i=1}^{N} \|a_i\|,$$

*and moreover, equality holds if and only if the columns are pairwise orthogonal.*

We did not give the proof in lecture but will give them here in the notes for completeness. This argument comes from a homework set of Fernando Rodriguez-Villegas.

*Proof.* By Gram-Schmidt there is an orthonormal basis $b_1, \ldots, b_N$ such that the span of these vectors is the same as the span of the columns of $A$. Let $B$ be the corresponding matrix. We see that $BB^T = I$. Each $z \in \mathbb{R}^N$ can be written

$$z = \sum_{i=1}^{N} \langle z, b_i \rangle b_i,$$

and therefore

$$a_m = \sum_{i=1}^{m} \langle a_m, b_i \rangle b_i.$$

Now let the matrix $C$ be defined so that the $(k,l)$ entry is $\langle a_l, b_k \rangle$ for $1 \leq k \leq l$, and is 0 otherwise. This matrix is upper triangular so the determinant is the product of the diagonal entries $\langle a_k, b_k \rangle$. We also see that $A = BC$.

Now

$$\begin{aligned} \det(A)^2 &= \det(A^T A) = \det(C^T B^T BC) = \det(C^T C) = \prod_{i=1}^{N} |\langle a_i, b_i \rangle|^2 \\ &\leq \prod_{i=1}^{N} \left( \sum_{m=1}^{i} |\langle a_m, b_i \rangle|^2 \right) = \prod_{i=1}^{N} \|a_i\|^2. \end{aligned}$$

We see that equality holds if and only if

$$|\langle a_i, b_i \rangle|^2 = \sum_{m=1}^{i} |\langle a_m, b_i \rangle|^2,$$

which occurs if and only if the columns are pairwise orthogonal. $\square$

We note that there is a version of this result for complex matrices, but we will not pursue this here.

**Definition.** An $N \times N$ matrix $A$, with all of its entries equal to 1 or $-1$, and $AA^T = nI_n$. $A$ is called a Hadamard matrix of order $n$.

Two Hadamard matrices are equivalent if we can get from one to the other by permuting rows and columns and by multiplying some rows or columns by $-1$.

**Example.**

$$H_1 = [1],\ H_2 = \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix},\ H_4 = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \end{pmatrix} = H_2 \otimes H_2.$$

The $H_2 \otimes H_2$ signifies the tensor product of matrices.

**Definition.** Let $A = [a_{i,j}]$ be an $n \times n$ matrix, and $B$ be an $m \times m$ matrix. Then

$$A \otimes B = \begin{pmatrix} a_{11}B & a_{12}B & \cdots \\ \vdots & & \\ a_{n1}B & \cdots & a_{nn}B \end{pmatrix}.$$

**Exercise.** Let $A$ be an $n \times n$ Hadamard matrix and $B$ be a $m \times m$ Hadamard matrix. Then $A \otimes B$ is an $nm \times nm$ Hadamard matrix.

The tensor product construction immediately gives the following result.

**Corollary 2.** *There exists a Hadamard matrix of order* $2^m$ *for all* $m$.

**Proposition 7.** *If $H$ is a Had matrix of order $n$, then $n = 1, 2$ or $n \equiv 0(\mod 4)$.*

*Proof.* By multiplying some columns by $-1$, we may assume that the first row consists entirely of 1s. Similarly, by multiplying some rows by $-1$ we may assume that the first row and column are entirely 1s. A Hadamard matrix in this form is called normalized. Since the rows of the matrix must be pairwise orthogonal, every other row must have $\frac{n}{2}$ entries 1 and $\frac{n}{2}$ entries $-1$. This shows that if $n > 1$ then $n$ must be even.

By permuting the columns we may assume that second row consists of $\frac{n}{2}$ 1s and then all of its $-1$s. Now by permuting columns within this first group of $\frac{n}{2}$ and then permuting columns within the second group of $\frac{n}{2}$ we can suppose that the third row fits into the following picture.

$$\begin{array}{|c|c|c|c|}
1 \cdots 1 & 1 \cdots 1 & 1 \cdots 1 & 1 \cdots 1 \\
1 \cdots 1 & 1 \cdots 1 & -1 \cdots -1 & -1 \cdots -1 \\
1 \cdots 1 & -1 \cdots -1 & 1 \cdots 1 & -1 \cdots -1 \\
\multicolumn{1}{c}{a} & \multicolumn{1}{c}{b} & \multicolumn{1}{c}{c} & \multicolumn{1}{c}{d}
\end{array}$$

The condition that the rows are pairwise orthogonal gives the following relations:

$$a + b + c + d = n$$

$$a + b = c + d = n/2 \quad .$$

$$a + c = b + d = n/2$$

Solving gives $a = b = c = d = \frac{n}{4}$, completing the proof.

□

**Conjecture.** *For all $k \geq 1$, there exists a Hadamard matrix of order $4k$.*

The first open case of this conjecture is currently $4k = 668$. The most recent improvement comes from Kharghani and Tayfe-Rezaie who constructed a Hadamard matrix of order 428 in 2004. This problem is interesting in its own right, but we are interested in it for its applications to coding theory.

**Definition.** Given a Hadamard matrix $H$ of order $n$ we get a Hadamard binary code of length $n$ by changing all of the 1s to 0s and the $-1$s to 1s, and taking the codewords to be the rows of the matrix. It is common to also include the complements of all of the codewords, that is, we take each codeword and add $(1, \ldots, 1)$. By the definition of a Hadamard matrix the distance between the codewords coming from any two rows is $\frac{n}{2}$. This gives a $(n, 2n, \frac{n}{2})$ code which is often nonlinear.

Consider the case $n = 8$ where we take the matrix $H_2 \otimes H_2 \otimes H_2$. You should check that the words of this code are exactly the words of the $[8, 4, 4]$ extended binary Hamming Code. This should be a little surprising since we do not even have good reasons to expect the code to be linear, let alone linear and interesting. However, we will see in the next lecture that there is a common framework in which both extended Hamming codes and Hadamard codes appear.

**Definition.** The Walsh-Hadamard code has generator matrix with columns equal to the $2^n$ binary vectors of length $n$. Note that this is the same as the parity check matrix of the length $2^n - 1$ Hadamard code, except with an extra all zero column.

We see that this code has $2^n$ codewords and that each differ in exactly $2^{n-1}$ places. One can show that going through the construction described above, listing the codewords in a $2^n \times 2^n$ matrix and then changing all of the 0s to 1s and 1s to $-1$s gives a Hadamard matrix of order $2^n$.

Part of the interest in these Walsh-Hadamard codes comes from the fact that they are locally decodable. That is, if we receive only a fraction of the received word, then we can still decode certain individual symbols with high probability. This is an important property for codes that are used in the real world. We will not really focus on this property here, but it is an active area of research (that I would be happy to learn more about) and might make a good final paper topic.

The next topic in the lecture is the theory of combinatorial designs. We will give a proposition related to Hadamard matrices which may seem a little unmotivated at first, but will become clear in terms of designs.

**Proposition 8.** *Let $H$ be a Hadamard matrix of order $n = 4s$. Take any 3 columns of $H$. There are exactly $s - 1$ rows other than the first, in which the symbols in these three columns are the same (either* (000) *or* (111)*).*

## 2.3 Combinatorial Designs

We will begin with the main definition and then give an interesting example.

**Definition.** A $t - (v, k, \lambda)$ design, often just called a $t$-design, is a pair $(P, B)$, where $|P| = v$, $B$ consists of $k$-element subsets of $P$, and any $t$-element subset of $P$ is contained in exactly $\lambda$ blocks. The set $P$ is usually called the set of points, and the set $B$ is called the set of blocks.

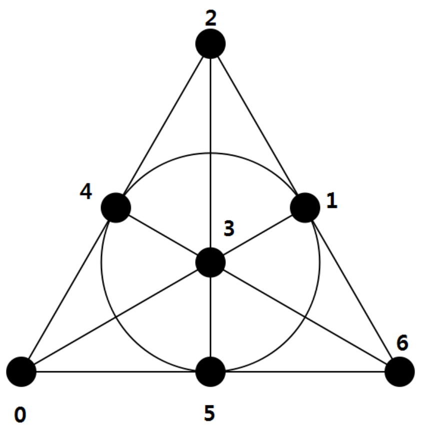
When $\lambda = 1$ this is called a Steiner system and is usually denoted $S(t, k, v)$.

This case of design theory is of particular importance, in part due to connections to finite projective planes, which we discuss below. There are many unsolved problems in this area. For example, it is unknown whether there are any Steiner systems $S(t, k, v)$ with $t \geq 6$.

We now see that we can phrase the above result about taking three columns of a Hadamard matrix as saying that the rows of $H$ excluding the first form a $3 - (4s, 2s, s - 1)$ design.

Design theory is a major topic in combinatorics and was the subject of a course, Professor Elkies' Math 155, last spring. Interestingly design theory has applications in statistics as well. We will discuss connections between codes and designs, but there is much more in the subject. The book Designs, Graphs, Codes and their Links by Cameron and van Lint is highly recommended and might be useful depending on what final project you choose.

**Example** (The Projective Plane of Order 2)**.**



This is a picture of the finite projective plane of order 2, or the Fano plane. Taking the points $P = \{0, \ldots, 6\}$ and the blocks $B = \{056, 024, 126, 013, 346, 235, 145\}$ gives a $2 - (7, 3, 1)$ design, or Steiner system $S(1, 3, 7)$. The blocks are referred to as 'lines' which makes sense in this case since each block is either a side or median of the triangle, or the one block consisting of the points on the circle. This design which has the same number of points and blocks is called a symmetric design.

**Definition.** A finite projective plane of order $n$ is defined as a set of $n$ points and $n$ lines satisfying the following properties:

1. Any two points are contained in a unique line.
2. Any two lines intersect at a unique point.
3. Every point lies on $n+1$ lines.
4. Every line contains $n+1$ points.

We note that this definition is equivalent to saying that a finite projective plane is a symmetric $2-(n^2+n+1, n+1, 1)$ design.

It is known that for every prime power $p^r$ a projective plane of that order exists. However, these are all the orders of projective planes known to exist. Certain cases, such as the nonexistence of a projective plane of order 6, can be determined without too much trouble. Proving that there is no finite projective plane of order 10 was a major achievement and took much computer computation. It is still not known whether there is a finite projective plane of order 12.

Often it is useful to look at a design through its incidence matrix. The rows of the matrix are indexed by the blocks, where the $i$th entry of a row is 1 if point $i$ is contained in that block, and is 0 otherwise.

For the projective plane of order 2 we get the following incidence matrix,

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

Inspecting this matrix we see that it is permutation equivalent to the set of codewords of the binary $[7,4]$ Hamming code of length 3. This is not a coincidence but is explained by one of the course highlights, the Assmus-Mattson Theorem.

**Definition.** Let $C$ be a code. If the codewords of $C$ of some fixed weight $i$ are the blocks of a $t$-design, then we say that $C$ 'holds a $t$-design'.

**Theorem 11** (Assmus-Mattson)**.** *Let $C$ be a binary $[n,k,d]$ code and suppose that the minimum weight of $C^\perp$ is $d^\perp$. Let $\{A_0,\ldots,A_n\}$ and $\{A_0^\perp,\ldots,A_n^\perp\}$ give the weight distributions of $C$ and $C^\perp$, respectively. Fix $t<d$ and let $s$ be the number of nonzero $A_i^\perp$ with $0<i\le n-t$. If $s\le d-t$,*

1. *The vectors of weight $i$ in $C$ hold a $t$-design if $A_i\ne 0$ and $d\le i\le n$.*
2. *The vectors of weight $i$ in $C^\perp$ hold a $t$-design if $A_i^\perp\ne 0$ and $d^\perp\le i\le n-t$.*

We will return to this theorem near the end of the course. In many cases it gives the best ways that we know of constructing interesting designs. This will be particularly interesting when we consider the Golay codes.

## 3 Reed-Muller and Reed-Solomon Codes

Thanks to Tony Feng for help typing this section

So far we've seen repetition codes, Hamming codes, and Hadamard codes. These all fit into a more general class of codes called Reed-Muller codes. These are easy to implement and have good decoding algorithms, but don't have the best minimal distance. We can describe them in terms of

1. Generator matrices.
2. Affine geometry.

Later we will discuss Reed-Solomon codes. (Yes, it is the same Reed.) These codes are used a lot in practice since they are easy to implement and have good decoding algorithms.

## 3.1 Reed-Muller Codes by Generator Matrices

Before defining Reed-Muller codes we will give two ways of producing new codes from codes that we already have.

**Definition.** For $i \in \{1,2\}$, let $C_i$ be a $[n_i, k_i, d_i]$ code. Then

$$C_1 \oplus C_2 = \{(c_1, c_2) \ : \ c_1 \in C_1, \, c_2 \in C_2\}.$$

**Exercise.** Show that this is an $[n_1+n_2, k_1+k_2, \min(d_1,d_2)]$ code.

Show that if $C_i$ has generator matrix $G_i$ and parity check matrix $H_i$ then $C_1 \oplus C_2$ has generator matrix $\begin{pmatrix} G_1 & 0 \\ 0 & G_2 \end{pmatrix}$, and parity check matrix $\begin{pmatrix} H_1 & 0 \\ 0 & H_2 \end{pmatrix}$

We give one slightly more complicated construction.

**Definition** ($(u \mid u+v)$ construction)**.** Let $C_1, C_2$ be linear codes with parameters $[n, k_i, d_i]$. Then let

$$C = \{(u, u+v) \ : \ u \in C_1, \text{and } v \in C_2\}.$$

**Exercise.** Show that this is a $[2, k_1+k_2, \min(2d_1, d_2)]$ code.

Show that if $C_i$ has generator matrix $G_i$ and parity check matrix $H_i$ then $C_1 \oplus C_2$ has generator matrix $\begin{pmatrix} G_1 & G_1 \\ 0 & G_2 \end{pmatrix}$, and parity check matrix $\begin{pmatrix} H_1 & 0 \\ -H_2 & H_2 \end{pmatrix}$

We can now give the most straightforward definition of the Reed-Muller codes.

**Definition.** Let $m$ be a positive integer and $0 \le r \le m$. Let $\mathcal{R}(0,m)$ be the binary repetition code of length $2^m$. Let $\mathcal{R}(m,m)$ be all of $\mathbb{F}_2^{2^m}$. We define

$$\mathcal{R}(r,m) = \{(u, u+v) \ : \ u \in \mathcal{R}(r, m-1), \, v \in \mathcal{R}(r-1, m-1)\}.$$

Using the above observations, we see that

$$\begin{aligned} G(0,m) &= [1,1,\dots,1] \\ G(m,m) &= I_{2^m} \\ G(r,m) &= \begin{pmatrix} G(r,m-1) & G(r,m-1) \\ 0 & G(r-1,m-1) \end{pmatrix} \end{aligned}$$

We consider some easy examples:

$$G(1,2) = \left(\begin{array}{cc|cc} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ \hline 0 & 0 & 1 & 1 \end{array}\right).$$

This is the code of all even weight vectors in $\mathbb{F}_2^4$. We see that this has dimension 3 and all the generators have even weight.

Next,

$$G(1,3) = \left(\begin{array}{cccc|cccc} 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ \hline 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{array}\right).$$

This is an $[8,4,4]$ code – in fact, it is equivalent to the extended binary Hamming code.

**Exercise.** Prove that every $[8,4,4]$ binary code is equivalent to the binary Hamming code.

You may want to first prove that any $[7,4,3]$ binary code is equivalent to the binary Hamming code of length 7.

**Proposition 9.** *The Reed-Muller codes have the following properties:*

1. $\mathcal{R}(i,m) \subseteq \mathcal{R}(j,m)$ *for all* $0 \le i \le j \le m$.
2. $\dim(\mathcal{R}(r,m)) = \binom{m}{0} + \ldots + \binom{m}{r}$.
3. *The minimum weight of* $\mathcal{R}(r,m)$ *is* $2^{m-r}$.
4. $\mathcal{R}(m,m)^\perp = \{0\}$*, and for* $0 \le r < m$*,* $\mathcal{R}(r,m)^\perp = \mathcal{R}(m-r-1,m)$.

*Proof.* 1. This is true for $m = 1$ by a computation and is obviously true for $j = m$. We proceed by induction on the length. Suppose that $\mathcal{R}(k,m-1) \subseteq \mathcal{R}(l,m-1)$ for all $0 \le k \le l < m$. Now suppose $0 < i \le j < m$. Then

$$\mathcal{R}(i,m) = \{(u,u+v) \ : \ u \in \mathcal{R}(i,m-1), v \in \mathcal{R}(i-1,m-1)\}$$

By the induction hypothesis, this is contained in

$$\{(u,u+v) \ : \ u \in \mathcal{R}(j,m-1), v \in \mathcal{R}(j-1,m-1)\} = \mathcal{R}(j,m).$$

For $i = 0$, we simply have to show that $[1,\ldots,1] \in \mathcal{R}(1,m)$. We can again argue by induction. This is clearly true for $m = 1$. If it is true for $\mathcal{R}(1,m-1)$, then since $\mathcal{R}(1,m)$ contains all $(u,u)$ for $u \in \mathcal{R}(1,m-1)$, it holds for $\mathcal{R}(1,m)$.

2. For $r = m$ we use the fact that $\mathcal{R}(m,m) = \mathbb{F}_2^{2^m}$ and $\sum_{i=0}^m \binom{m}{i} = 2^m$.

This holds for $m = 1$ by a computation. Now suppose $\mathcal{R}(i,m-1)$ has dimension $\sum_{j=0}^i \binom{m-1}{i}$ for all $0 \le i < m$. By the $(u \mid u+v$ construction, the dimension of $\mathcal{R}(i,m)$ is the sum of the dimensions of $\mathcal{R}(i,m-1)$ and $\mathcal{R}(i-1,m-1)$. Now noting $\binom{m-1}{i-1} + \binom{m-1}{i} = \binom{m}{i}$ completes the proof.

3. Also follows from noting that it holds for $m = 1$ and for $r = 0$ and $r = m$ and then arguing by induction using the minimum distance arising from the $(u \mid u+v)$ construction.

4. We defer the proof until we have described Reed-Muller codes in terms of affine geometry. ☐

**Proposition 10.** *The following properties of Reed-Muller codes hold:*

1. $\mathcal{R}(m-2,m)$ *are extended Hamming codes of length* $2^m$.
2. $\mathcal{R}(1,m)$ *consists of the rows of the Hadamard matrix* $H_{2^m} = H_2 \otimes \cdots \otimes H_2$*, where we change the* 1 *to* 0 *and* $-1$ *to* 1*, together with their complements.*

*Proof.* 1. It is easy to write down a parity check matrix for the binary Hamming code of length $2^m - 1$. We can then write down a generator matrix for it, and add a column to the generator matrix to form the extended binary Hamming code. We can now go back and find a parity check matrix for this code. This is the generator matrix of the dual code. It is an exercise to show that this is permutation equivalent to the generator matrix for $\mathcal{R}(1,m)$. We now use part 4. of the previous proposition.

2. We prove this by induction on $m$. For $m = 1$ and $m = 2$ we can write down the matrices and check. We now note that

$$\mathcal{R}(1,m) = \{(u,u) \ \mid u \in \mathcal{R}(1,m-1)\} \cup \{(u,u+\mathbf{1}) \ \mid u \in \mathcal{R}(1,m-1)\}.$$

When we replace 1 by 0 and $-1$ by 1, the matrix $H_2$ becomes $\begin{pmatrix} 0 & 0 \\ 0 & 1 \end{pmatrix}$. Arguing by induction shows these sets of codewords are equal. ☐

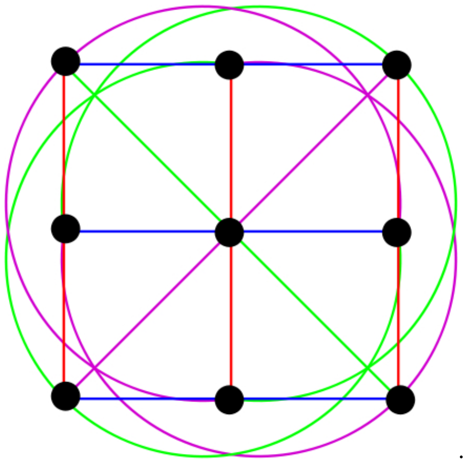
## 3.2 Affine Geometry and Reed-Muller codes

Now we will change perspectives and consider Reed-Muller codes from the point of view of affine and projective geometry.

**Definition.** The affine geometry of dimension $m$ over $\mathbb{F}_q$ is just the vector space $\mathbb{F}_q^m$, which we denote by $AG(m, q)$.

A $k$-dimensional affine subspace, or $k$-flat is a coset of a $k$-dimensional linear subspace, which we call a $k$-flat. An $m - 1$ flat is a hyperplane.

This is best explained with a picture:


.

This picture represents $AG(2, 3)$. We can identify this vector space with $(u, v)$ where $u, v \in \{0, 1, 2\}$. We can suppose that the center point is the origin $(0, 0)$. Each of the colored lines connecting three points represents a 1-flat, or a line, of the geometry. A line passing through the origin is a linear subspace, and line not passing through the origin is a coset of a linear subspace. If you take all three of the lines together of a single color, it shows how the entire plane $AG(2, 3)$ decomposes into three cosets, each of which is a line.

**Example.** $AG(3, 3) = \mathbb{F}_3^3$. We can partition a cube into 27 boxes and think of the center of each box giving a point. We can think of the center box as the origin. There are $\frac{1}{2}(27-1) = 13$ lines through the origin since each line through the origin is of the form $\{0, 0 + d, 0 + 2d\}$ where $d = (d_1, d_2, d_3)$ is a nonzero element of $\mathbb{F}_3^3$. We note that switching $d$ and $-d$ does not change the points on the line. There are 26 possible choices for $d$.

**Definition.** The projective geometry of dimension $m$ over $\mathbb{F}_q$ is the set of linear subspaces in $AG(m+1, q)$. Subspaces of dimension 1 are called points, subspaces of dimension 2 are called lines, and so on up to the $m$ dimensional subspaces, the hyperplanes.

The 13 lines through the origin of $AG(3, 3)$ give the points of $PG(2, 3)$. The convention is to write a projective point with the first non-zero coordinate being 1. In this example, the points of $PG(2, 3)$ look like

$$[1 : a : b], \quad [0 : 1 : a], \quad [0 : 0 : 1].$$

This notation identifies the projective point $[a : b : c]$ with all of the affine points that can be written as scalar multiples of $(a, b, c)$, so, the line passing through the origin and $(a, b, c)$.

**Exercise.** We saw in the lecture on designs that the Fano plane, nicely drawn as an equilateral triangle with a circle inscribed, can be identified with $PG(2, 3)$. These points can be identified with the lines of $AG(3, 2)$. Draw $AG(3, 2)$ and indicate where the points of the Fano plane occur.

Now we'll relate this to Reed-Muller codes. The idea is to write down a matrix whose columns index points in affine geometry. In particular, we can think of points in $AG(m,2)$ as elements of $\mathbb{F}_2^m$ and express them as column vectors. These correspond to binary representations of the numbers from 0 to $2^m - 1$. Represent $j$ as

$$x_j = \sum_{i=0}^{m-1} \xi_i u_i,$$

where $(u_i)$ gives the standard basis. We will do something a little unconventional. We think of our standard basis vectors as $u_0 = \begin{pmatrix} 0 \\ \vdots \\ 0 \\ 1 \end{pmatrix}, \ldots, u_{m-1} = \begin{pmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{pmatrix}$. This gives for example, when $m = 3$, we have $x_3 = \begin{pmatrix} 0 \\ 1 \\ 1 \end{pmatrix}$.

Let $n = 2^m$. Define the matrix $E$ having the column vectors $x_j$, $0 \le j < n$.

For example, for $m = 3$ we have

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

**Definition.** For $0 \le i < m$, let $A_i = \{x_j \ : \ \xi_{ij} = 1\}$. This gives a collection of points in $AG(m,2)$. We can also think of this as a collection of columns of our matrix.

This gives a hyperplane in $AG(m,2)$. By our ordering of the basis vectors, the columns which have points corresponding to this hyperplane are those for which the $m-i$th row has a 1 in that column.

For example, for $m = 3$ we have

$$A_0 = \left\{ \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}, \begin{pmatrix} 0 \\ 1 \\ 1 \end{pmatrix}, \begin{pmatrix} 1 \\ 0 \\ 1 \end{pmatrix}, \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix} \right\}.$$

**Definition.** For $1 \le i \le m$, let $v_i$ be the $i$th row of $E$. Note that $v_i$ is the characteristic function of $A_{m-i}$. The characteristic function of $\mathbb{F}_2^{2^m}$ is just $(1, \ldots, 1)$, which we denote by $v_0$.

For two vectors $a, b$ of length $n$, we let $ab = (a_0 b_0, a_1 b_1, \ldots, a_{n-1} b_{n-1})$. This is often called the wedge product of $a$ and $b$.

For example, when $m = 3$, $v_2 = (01010101)$.

**Lemma 1.** *1. Let $i_1, \ldots, i_s$ be distinct in $\{0, \ldots, m-1\}$. Then $v_{i_1} \ldots v_{i_s}$ is the characteristic function of $A_{i_1} \cap A_{i_2} \cap \ldots \cap A_{i_s}$.*

*2. The weight of $v_{i_1} \ldots v_{i_s}$ is $2^{m-s}$.*

*3. The characteristic function of $x_j$ is $e_j = \prod_{i=0}^{m-1} (V_i + (1 + \xi_{ij})1)$.*

*4. The products $v_{i_1} \ldots v_{i_s}$, $0 \le s \le m$, are a basis for $\mathbb{F}_2^{2^m}$.*

*Proof.* 1. The intersection $A_{i_1} \cap \ldots \cap A_{i_j}$ consists of vectors which have a 1 in rows $i_1, \ldots, i_j$, which is detected by the product of the characteristic functions.

2. The follows from part $(i)$ noting that $2^{m-s}$ is the cardinality of an $(m-s)$-flat.

3. We consider the matrix $E$. For each $i$ such that $\xi_{ij} = 0$ we replace the $m-i$th row of $E$ by its complement. This switches $v_i$ and $\mathbf{1} + v_i$. When we multiply the rows of this new matrix the only 1 will occur in position $j$ since every column occurs exactly once.

4. There are $\sum_{s=0}^{m} \binom{m}{s} = 2^m = n$ products $v_{i_1} \cdots v_{i_s}$. The result now follows from 3.

□

**Proposition 11.** *Let $0 \leq r < m$. The linear code of length $n = 2^m$ with $v_{i_1} \ldots v_{i_s}$ $0 \leq i_1 \leq \cdots \leq r$ as a basis, is $\mathcal{R}(r, m)$.*

*Proof.* We want to relate the basis vectors described above to the $(u \mid u + v)$ construction described earlier. We first note that give basis vectors $v_0 = (1 \cdots 1)$ and $v_i$ for $1 \leq i \leq m$ for $\mathcal{R}(1, m)$ we see that a basis for $\mathcal{R}(1, m+1)$ is $w_i = (v_i, v_i)$ for $0 \leq i \leq m$ and $w_{m+1} = (0 \cdots 01 \cdots 1)$.

This shows that a basis vector of $\mathcal{R}(r + 1, m + 1)$ of the form $w_{i_1} \cdots w_{i_s}$ is of the form $(u \mid u)$ with $u \in \mathcal{R}(r, m + 1)$ if $w_{m+1}$ does not occur in the product, and is of the form $(0 \mid v)$ if $w_{m+1}$ does occur. □

**Proposition 12.** *The dual of $\mathcal{R}(r, m)$ is $\mathcal{R}(m - r - 1, m)$.*

*Proof.* We first check that the sum of the dimensions of these two codes is $n$. We have that the dimension of $\mathcal{R}(r, m)$ is $\sum_{i=0}^{r} \binom{m}{r}$. Using properties of binomial coefficients completes this step. Next we check that given a basis vector of $\mathcal{R}(r, m)$, $u$, and a basis vector of $\mathcal{R}(m - r - 1, m)$, $v$, then $u$ and $v$ are orthogonal. We need only check that the product has even weight. This follows because the product is the characteristic function of some flat, which has weight $2^{m-k}$ for some $k$. □

It seems at first that this construction of Reed-Muller codes is more complicated than the $(u \mid u + v)$ construction, but hopefully we have shown that certain properties of the codes become more transparent when viewed through characteristic functions and affine geometry. We point out one more way of viewing these codes. Let $x = (x_1, \ldots, x_m)$ run through all binary vectors of length $m$. We see that $x_i$ gives a function from $\mathbb{F}_2^{2^m}$ to $\mathbb{F}_2$, so a word in $\mathbb{F}_2^{2^m}$. The function $x_{i_1} \cdots x_{i_s}$ has value 1 if and only if $x$ is in a certain $m - s$ flat. Therefore, we see that $\mathcal{R}(r, m)$ consists of sequences of values taken by polynomials in $x_1, \ldots, x_m$ of degree at most $r$. This is the idea which will give the generalized Reed-Muller codes which we will discuss in the next lecture.

## 3.3 Reed-Solomon Codes

The next topic for this lecture is Reed-Solomon codes. These are examples of BCH codes, which are a class of cyclic codes. We will discuss these two larger families in the next lecture.

Reed-Solomon codes are important in practice. Like Reed-Muller codes they are easy to implement and have good decoding algorithms. They are also our first examples of Minimum Distance Separable or MDS codes, which are particularly useful for the binary erasure channel. They are also useful channels which have burst errors, where the errors are not independently distributed throughout the codeword. The presence of one error makes it more likely that there are other errors nearby.

**Definition.** Take $n$ distinct points $x_1, \ldots, x_n \in \mathbb{F}_q$. (Usually, we take $(n, q) = 1$.) The Reed-Solomon code $RS(n, k)$ has codewords $(f(x_1), \ldots, f(x_n))$ where $f$ runs over all polynomials of degree less than $k$.

Note that polynomials of degree less than $k$ form a vector space, so this is a linear code of dimension $k$. Recall that two polynomials of degree $\leq k - 1$ whicih agree at $k$ or more points in a field must be the same. So the distance $d$ is at least $n - (k - 1)$.

We need a simple definition. This is another way of getting a new code from an old code.

**Definition.** Let $C$ be an $[n, k, d]$ code over $\mathbb{F}_q$. We can puncture the code $C$ by deleting the same coordinate $i$ from each codeword of $C$. The resulting code is often denoted $C^*$ and clearly has length $n - 1$.

**Proposition 13.** *Let $C$ be an $[n, k, d]$ code over $\mathbb{F}_q$ and let $C^*$ be the code punctured on the $i$th coordinate.*

1. *If $G$ is the generator matrix for $C$, then the generator matrix of $C^*$ is $G$ minus the $i$th column, where if there are two identical rows in the resulting matrix, we delete one of them, and if there is a zero row, we delete that as well. In particular, $C^*$ is linear.*

2. *If $d > 1$ then $C^*$ is a $[n-1, k, d^*]$ code where $d^* = d-1$ if $C$ has a minimum weight codeword with a nonzero $i$th coordinate and $d^* = d$ otherwise.*

*Proof.* Exercise. □

We now give a general bound on the dimension of codes. Along with the sphere packing bound this is one of the simplest bounds on codes.

**Theorem 12** (Singleton Bound)**.** *For $q, n, d \in \mathbb{N},\ q \geq 2$, we have $A(n, d) \leq q^{n-d+1}$.*

*Proof.* Pick any set of $d-1$ coordinates and puncture the code at these places. The codewords must remain distinct by the proposition about minimum distance of a punctured code. Clearly, there are at most $q^{n-d+1}$ possible codewords of length $n-(d-1)$. □

**Definition.** A code attaining the Singleton bound is called maximum distance separable (MDS).

We have already seen that Reed-Solomon codes satisfy $d \geq n-k+1$. The Singleton bound implies that $d \leq n-k+1$, and therefore $d = n-k+1$, so Reed-Solomon codes are MDS codes.

**Theorem 13.** *The dual of an MDS code is also MDS.*

*Proof.* Take $G = [I_k \mid P]$ to be the generator matrix of an MDS code. Recall that $H = [-P^T \mid I_{n-k}]$. Since the minimum distance is the minimum number of linearly dependent columns, every $n-k$ columns of $H$ are linearly independent. In particular, every $(n-k) \times (n-k)$ submatrix is non-singular. This means that no non-zero codeword of $C^\perp$ has $n-k$ zeros. So the minimum distance of the dual code, $d^\perp$ is at least $n-(n-k-1) = k+1$. Since the dimension of this code is $n-k$, the Singleton bound implies that $d^\perp = n-(n-k)+1$. □

## 3.4 Decoding

One common method for decoding is known as syndrome decoding. We consider each of the possibilities for our parity checks and find for each one an error vector of minimum weight. This is what we did in the first lecture when we considered the $[6, 3, 3]$ code and computed $(s_1, s_2, s_3)$. We will not focus on this type of decoding here. Instead we will consider the Berlekamp-Welch algorithm which decodes directly from the generator matrix of $C$. The presentation here follows a lecture of Henry Cohn. For more details, you can look up the course notes of Atri Rudra,

`http://codingtheory.wordpress.com/2009/04/03/lecture-30-berlekamp-welch-algorithm/`.

Suppose we have points $\alpha_1, \ldots, \alpha_n \in \mathbb{F}_q$. We receive $(y_1, \ldots, y_n) \in \mathbb{F}_q^n$. We want to find a polynomial $f(x)$ of degree less than $k$ such that $f(x_i) = y_i$ at all but at most $e$ places, where $e < \lfloor \frac{n-k+1}{2} \rfloor$. If no such polynomial exists, we want to output an error.

The idea is to find two polynomials $g(x)$ and $h(x)$ such that $\deg(g) = e$ and $\deg(h) < e + k$, and $g(\alpha_i) y_i = h(\alpha_i)$.

The problem here is that finding $g(x)$ is often exactly as hard as just finding the polynomial $f(x)$ giving the codeword. Finding $h(x)$ can also be that difficult. The trick is that finding a pair satisfying the above condition is easier.

First we claim that such a pair exists. We define $g(x)$ to be the *error-locating polynomial* for $P(x)$. Let $d$ be the distance between the received word $y$ and the word $(f(\alpha_1), \ldots, f(\alpha_n))$. We define the polynomial of degree exactly $e$ given by

$$g(x) = x^{e-d} \prod_{1 \leq i \leq n\ |y_i \neq f(x_i)} (x - \alpha_i).$$

We see that $g(x)$ satisfies $g(\alpha_i) = 0$ if and only if $y_i \neq P(\alpha_i)$. When $g(\alpha_i) \neq 0$ we have $f(\alpha_i) = y_i$. Now let $g(x)f(x) = h(x)$. We see that the pair $g(x), h(x)$ satisfies our conditions.

**Proposition 14.** *Any pairs $(g_1, h_1)$ and $(g_2, h_2)$ satisfying the conditions above give the same polynomial. That is,*

$$\frac{h_1(x)}{g_1(x)} = \frac{h_2(x)}{g_2(x)}.$$

*Proof.* Consider $h_1g_2 - h_2g_1$, which has degree at most $e+e+k-1 = 2e+k-1 < n$. So this is a polynomial of degree less than $n$ vanishing at the $n$ points $x_1, \ldots, x_n$, so it is zero. □

Once we find any valid pair $g, h$, we are done. We consider the coefficients of these polynomials as unknowns, so $g(x)$ has $e+1$ unknowns, and $h(x)$ has $e+k$ unknowns. For each $1 \leq i \leq$, the condition that $g(\alpha_i)y_i = h(\alpha_i)$ gives a linear equation in these unknowns. Therefore, we have a system of $n$ linear equations in $2e+k+1 < n+2$ unknowns. We already know that there is one such solution. We also want to require that the degree of $g(x)$ is exactly $e$. This fixes one of the coefficients of this polynomial. This gives $n+1$ linear equations in at most $n+1$ variables, which we can solve by representing them as a matrix and using Gaussian elimination.

Suppose we go through this process and find no such pair $(g, h)$. Then there's no codeword within $e$, so our algorithm outputs a failure. Perhaps we want to know not only if there is a single codeword within distance $e$, but the list of all codewords within some larger distance $t$. There is an algorithm **Sudan-Guruswami list-decoding** which does this efficiently.

Further study of decoding algorithms for Reed-Solomon codes, and the more general class of BCH codes, would make a very interesting final paper topic. In particular, there is an approach to this problem using Gröbner bases, one of the most important tools in computational commutative algebra and algebraic geometry.

# 4 Cyclic and BCH Codes

Thanks to Tony Feng for help typing up this section.

## 4.1 Cyclic Codes

We next introduce Cyclic Codes, a class which includes the Reed-Solomon codes discussed above. A Reed-Solomon code is actually a type of BCH code, which is a type of cyclic code. We will discuss this below.

**Definition.** A linear code $C$ of length $n$ is called cyclic if for all codewords $c = (c_0, \ldots, c_{n-1})$ the vector $(c_{n-1}, c_0, c_1, \ldots, c_{n-2})$ is also in $C$.

This definition says that a particular permutation of coordinates fixes the codes. That is, the permutation of the coordinates labeled $\{0, \ldots, n-1\}$ that sends $0 \to 1,\ 1 \to 2, \ldots, n-1 \to 0$ is in $\text{Aut}(C)$, the automorphism group of the code. Why is this type of permutation so special? We will show that we can learn a lot about this class of codes from identifying codewords with polynomials.

We first note that there is an isomorphism between $\mathbb{F}_q^n$ and $\mathbb{F}_q[x]/\langle x^n - 1\rangle$ where we identify the word $(a_0, \ldots, a_{n-1})$ with the polynomial $a_0 + a_1x + \cdots + a_{n-1}x^{n-1}$. We will freely refer to elements of a linear code both as codewords which we will usually call $c$, and as polynomials, which we usually call $f$.

**Proposition 15.** *A linear code $C$ is cyclic if and only if $C$ is an ideal in $\mathbb{F}_q[x]/\langle x^n - 1\rangle$.*

*Proof.* First suppose that $C$ gives an ideal in $\mathbb{F}_q[x]/\langle x^n - 1\rangle$. If $f \in C$ then since $C$ is an ideal $xf \in C$. We see that $xf$ is exactly the codeword which must be in $C$ if $C$ is cyclic.

Now suppose that $C$ is cyclic. Then for $f \in C$ we also have $x^if \in C$ for all $0 \leq i \leq n-1$. The polynomials $\{x^0, \ldots, x^{n-1}\}$ form a basis for $\mathbb{F}_q[x]/\langle x^n - 1\rangle$, so we see that $C$ is an ideal. □

We recall two well-known facts from abstract algebra. We will leave these as exercises.

**Exercise.**
1. Let $F$ be a field. Then $F[x]$ is a principal ideal domain. More generally, show that for $D$ an integral domain, $D[x]$ is a principal ideal domain if and only if $D$ is a field.

2. Show that the quotient ring of a principal ideal domain is a principal ideal domain.

**Proposition 16.** *Let $C$ be a cyclic code of length $n$. Then there is a polynomial $g(x) \in C$ which is the unique polynomial of minimal degree in $C$ and such that $C = \langle g(x) \rangle$. Also, $g(x) \mid x^n - 1$.*

*Proof.* The previous proposition and the above exercises imply that a cyclic code of length $n$ is a principal ideal of $\mathbb{F}_q[x]/\langle x^n - 1\rangle$.

Let $g(x)$ be a monic polynomial of minimal degree in $C$. First suppose there is another monic polynomial of the same degree, $h(x)$. Then $g(x) - h(x)$ is in $C$ and has strictly lower degree, which is a contradiction.

We now consider any $h(x) \in C$. By the division algorithm in $\mathbb{F}_q[x]$ we can write $h(x) = q(x)g(x) + r(x)$ where $r(x) = 0$ or the degree of $r(x)$ is strictly smaller than the degree of $g(x)$. Since $C$ is an ideal, $r(x) \in C$, and by our assumption about the degree of $g(x)$, we have $r(x) = 0$. This proves the second statement.

For the third statement, we again apply the division algorithm. We have $x^n - 1 = q(x)g(x) + r(x)$. Since $x^n - 1$ corresponds to the zero codeword, we again have $r(x) \in C$. Again, $r(x) = 0$ and so $g(x) \mid x^n - 1$. □

We call the polynomial $g(x)$ in the above proposition the generator polynomial of $C$.
We often assume that $(n, q) = 1$. This is to ensure that $x^n - 1$ factors into $t$ irreducibles with no repeated factors. Therefore, there are $2^t$ possible cyclic codes of length $n$ over $\mathbb{F}_q$. We see that the study of cyclic codes is closely related to the study of divisors of $x^n - 1$.

We next describe the dimension and generator matrix of a cyclic code. We leave the proof as an exercise.

**Proposition 17.** *Let $g(x) = g_0 + g_1x + \ldots + g_{n-k}x^{n-k}$ be the generator polynomial for a cyclic code $C$. Then $C$ has $\{g(x), xg(x), \ldots, x^{k-1}g(x)\}$ as a basis. In particular,* $\dim(C) = k$.

*The generator matrix for $C$ is*

$$G = \begin{pmatrix} g_0 & g_1 & \ldots & g_{n-k} & 0 & \ldots & 0 \\ 0 & g_0 & g_1 & \ldots & g_{n-k} & \ldots & 0 \\ \vdots & \ddots & \ddots & \ddots & \ddots & \ddots & \vdots \\ 0 & \ldots & 0 & g_0 & g_1 & \ldots & g_{n-k} \end{pmatrix}$$

**Exercise.** Prove the above proposition.

**Definition.** There exists a polynomial $h(x)$ such that $g(x)h(x) = x^n - 1$. This is called the parity check polynomial for $C$.

**Exercise.** Let $C$ be a cyclic code with generator polynomial $g(x)$ and parity check polynomial $h(x)$. Determine the parity check matrix for $C$.

Use this fact to prove that the dual of a cyclic code is cyclic.

We will next move on to defining the BCH codes, a particular type of cyclic code that includes the Reed-Solomon codes.

**Definition.** Let $q = p^r$. Recall from the previous exercise set that if the multiplicative group of $\mathbb{F}_q$ is cyclic. This implies that the elements of $\mathbb{F}_q$ are exactly the roots of the polynomial $x^q - x \in \mathbb{F}_q[x]$. An element $\beta \in \mathbb{F}_q$ such that $\beta^k = 1$ but $\beta^l \neq 1$ for any $1 \leq l < k$ is called a primitive $k$th roots of unity. If $\alpha \in \mathbb{F}_q$ then the minimal polynomial of $\alpha$ over $\mathbb{F}_p$ is the irreducible polynomial $f(x) \in \mathbb{F}_p[x]$ such that $f(\alpha) = 0$.

We leave three facts about minimal polynomials as exercises.

**Exercise.**

1. Suppose $\alpha \in \mathbb{F}_q^k$ and $f(\alpha) = 0$. Then $f(\alpha^q) = 0$.
2. Suppose $g(x)$ is a polynomial with coefficients in some extension field $\mathbb{F}_{q^k}$ of $\mathbb{F}_q$. If $g(\alpha^q) = 0$ for every $\alpha$ for which $g(\alpha) = 0$, then $g(x) \in \mathbb{F}_q[x]$.
3. Suppose that $\alpha$ has order $e$ in $\mathbb{F}_q$. Then the minimal polynomial of $\alpha$ is $\prod_{i=0}^{m-1}(x - \alpha^{p^i})$, where $m$ is the smallest integer such that $p^m \equiv 1 \pmod{e}$.

## 4.2 BCH Codes

Recall that Reed-Solomon codes were defined by evaluating all polynomials of degree less than $k$ at some points. We can obtain more codes by restricting this set of polynomials.

**Definition.** The cyclic code $\mathcal{C}$ with defining set $A = \{\alpha^{i_1}, \dots, \alpha^{i_l}\}$ where $A$ is a set of $n$th roots of unity is the set of all codewords $f(x)$ such that $f(a) = 0$ for all $a \in A$. If $A$ is the maximal defining set for $C$, then we say $A$ is complete.

We say that $A$ is a consecutive set of length $l$ if there is a primitive $n$th root of unity $\beta$ and an exponent $i$ such that $A = \{\beta^i, \beta^{i+1}, \dots, \beta^{i+l-1}\}$.

It is clear that the generator $g(x)$ will be the least common multiple of the minimal polynomials of the elements in $A$.

**Definition.** A BCH code of designed distance $\delta$ is a cyclic code with generator polynomial $g(x)$ equal to the least common multiple of the minimal polynomials of $\beta^l, \beta^{l+1}, \dots, \beta^{l+\delta-2}$ where $\beta$ is a primitive $n$th root of unity.

If $n = q^m - 1$, then we say that this BCH code is primitive. We often will take $l = 1$, in which case this is a narrow-sense BCH code.

**Proposition 18.** *The primitive BCH code with $n = q - 1$ and with generator polynomial*

$$g(x) = \prod_{i=1}^{n-k} (x - \alpha^i),$$

*where $\alpha$ is a primitive $n$th root of unity is a Reed-Solomon code of length $n = q - 1$ with $x_1 = 1, x_2 = \alpha, \dots, x_{q-1} = \alpha^{q-2}$.*

*Proof.* The words of the Reed-Solomon code were previously defined as $(f(x_1), \dots, f(x_n))$ where $f(x) \in \mathbb{F}_q[x]$ is a polynomial of degree at most $k-1$, and $x_1, \dots, x_n$ are $n$ distinct points in $\mathbb{F}_q$. Suppose $(f_0, f_1, \dots, f_{n-1}) \in C$. We note that the dimension of this code is $k$

The BCH code with generator polynomial $g(x) = \prod_{i=1}^{n-k}(x - \alpha^i)$ also has dimension $k$, so we need only show that this code contains the Reed-Solomon code to show that they are equal. The words of this code consist of all $a(x)$ such that $a(\alpha^i) = 0$ for all $1 \le i \le n-k$. Let $f(x) = \sum_{m=0}^{k-1} f_m x^m$ be some polynomial of degree at most $k-1$ and let $c = (f(1), f(\alpha), \dots, f(\alpha^{n-1}))$. This corresponds to the polynomial

$$c(x) = \sum_{j=0}^{n-1} f(\alpha^j) x^j.$$

We need only show that for all $1 \le i \le n-k$, $c(\alpha^i) = 0$. We have,

$$\begin{aligned} c(\alpha^i) &= \sum_{j=0}^{n-1} \left( \sum_{i=0}^{k-1} f_m \alpha^{jm} \right) \alpha^{ij} \\ &= \sum_{m=0}^{k-1} f_m \sum_{j=0}^{n-1} \alpha^{(i+m)j} = \sum_{m=0}^{k-1} f_m \frac{\alpha^{(i+m)n} - 1}{\alpha^{i+m} - 1}. \end{aligned}$$

We see that $\alpha^{(i+m)n} = 1$ but $\alpha^{i+m} \neq 1$ since $1 \le i + m \le (n-k) + (k-1) = n-1$ and $\alpha$ is a primitive $n$th root of unity. So $c(\alpha^i) = 0$ for all $1 \le i \le n-k$. $\square$

Note that if $n = q-1$ and the collection $\{x_1, \dots, x_n\}$ consists of all of the nonzero points of $\mathbb{F}_q$, then we can permute the coordinates of $C$ so that $x_1 = 1, x_2 = \alpha, \dots, x_n = \alpha^{n-1}$ for some $\alpha$. This proposition gives us another way to think of Reed-Solomon codes. The first definition is the original one. The interpretation of Reed-Solomon codes as BCH codes is useful because there are many results which apply to general BCH codes, including good decoding algorithms.

We will next consider the minimum distance of BCH codes. For this, we need a definition from linear algebra.

**Lemma 2** (Vandermonde Determinant). *We have*

$$\left|\begin{pmatrix} 1 & 1 & \dots & 1 \\ x_1 & x_2 & \dots & x_{n-1} \\ \vdots & \vdots & \vdots & \vdots \\ x_1^{n-1} & x_2^{n-1} & \dots & x_{n-1}^{n-1} \end{pmatrix}\right| = \prod_{1\le i<j\le n-1} (x_j - x_i).$$

**Exercise.** Prove this lemma.

**Theorem 14.** *The BCH code of designed distance $\delta$ has minimum distance at least $\delta$.*

*Proof.* If $c(x) = \sum_{j=1}^{w} c_{i_j} x^{i_j}$ is a codeword of $C$, then $c(\beta^l) = c(\beta^{l+1}) = \dots = c(\beta^{l+\delta-2}) = 0$. Suppose $w \le \delta - 1$.

Consider the matrix

$$H = \begin{pmatrix} \beta^{i_1 l} & \beta^{i_2 l} & \dots & \beta^{i_w l} \\ \beta^{i_1(l+1)} & \beta^{i_2(l+1)} & \dots & \beta^{i_w(l+1)} \\ \vdots & \vdots & & \\ \beta^{i_1(l+w-1)} & \beta^{i_2(l+w-1)} & \dots & \beta^{i_w(l+w-1)} \end{pmatrix}.$$

We must have $Hu^T = 0$ where $u = (c_{i_1}, \dots, c_{i_w})$. By assumption $u \neq 0$ and therefore $H$ is a singular matrix, so in particular it has determinant zero. Dividing out $\alpha^{i_j l}$ from each column shows that the determinant of this matrix is actually $\beta^{(i_1+\dots+i_w)l}\det(V)$ where $V$ is the Vandermonde matrix

$$V = \begin{pmatrix} 1 & 1 & \dots & 1 \\ \beta^{i_1} & \beta^{i_2} & \dots & \beta^{i_w} \\ \vdots & \vdots & & \\ \beta^{i_1(w-1)} & \beta^{i_2(w-1)} & \dots & \beta^{i_w(w-1)} \end{pmatrix}.$$

Since the $\beta^{i_j}$ are distinct, this determinant is nonzero, giving a contradiction. $\square$

This is known as the BCH bound. In some cases, BCH codes actually have larger minimum distance than is guaranteed by this bound. Much research is done on studying the minimum distance of certain types of BCH codes.

We will close this section by briefly mentioning how some of the ideas of Reed-Muller and Reed-Solomon codes generalize. Recall that in the previous lecture we defined the $r$th order binary Reed-Muller codes $\mathcal{R}(r,m)$ of length $2^m$ in two different ways:

1. Defining the generator matrix recursively using the $(u \mid u+v)$ construction.
2. By giving a basis for the code which consisted of the collection of vectors $v_{i_1} \cdots v_{i_s}$ where $s \le r$ and $v_i$ is the $i$th row of a matrix $E$ with $m$ rows which has columns representing all of the binary numbers $0 \le j < 2^m$. These $v_i$ were the characteristic functions of coordinate hyperplanes in $AG(m,2)$.

From this second description we can give an alternate interpretation of $\mathcal{R}(r,m)$ as sequences of values taken by polynomials in $v_0, v_1, \dots, v_m$ of degree at most $r$ evaluated on the $2^m$ points of $AG(m,2)$, where $v_0$ is the 'empty polynomial', the vector $(1,\dots,1)$. This motivates the following definition.

**Definition.** Let $m$ be positive, $n = q^m$. Let $P_1, \dots, P_n$ be an ordering of points of $\mathrm{AG}(m, q)$. For any $0 \le r \le m(q-1)$, consider the polynomials over $\mathbb{F}_q$ in the coordinate polynomials $x_0, \dots, x_{m-1}$ of degree at most $r$ (including the codeword $(1, \dots, 1)$ coming from the 'empty polynomial'). Let

$$\mathcal{R}_q(r, m) = \{(f(P_1), \dots, f(P_n))\},$$

where $f$ is in this vector space.

Note that $x_i^q = x_i$, so we can assume that each exponent is at most $q-1$.

There is also a $(u \mid u+v)$ generator matrix construction for this code and another construction involving generator polynomials.

There is a common idea here and in the development of Reed-Solomon codes. In both cases, we picked a set of points and a vector space of polynomials, and formed a code by evaluating all the polynomials on those points. This idea also forms the basis of Algebraic Geometry codes, which are constructed by picking points on some projective curve (or more generally, a projective variety) and evaluating the functions in the Riemann-Roch space $\mathcal{L}(D)$ for some fixed divisor $D$ of that variety.

# 5 Shannon's Noisy Channel Coding Theorem

Thanks to Tony Feng for typing this lecture!

## 5.1 Definitions

The goal of this section is to prove Shannon's Noisy Channel Coding Theorem which we stated in the first lecture.

We first consider three possible decoding rules:

1. Ideal observer rule: receive $x \in \mathbb{F}_q^n$, and we decode as $c \in \mathcal{C}$ maximizing
$$\mathbb{P}(c \text{ sent } \mid x \text{ received}).$$
2. Maximum likelihood deocding: decode as $c \in \mathcal{C}$ maximizing
$$\mathbb{P}(x \text{ received } \mid c \text{ sent}).$$
3. Minimum distance decoding: decode to minimize $d(x, c)$.

**Proposition 19.** *If all messages are equally likely to be sent, then the ideal observer rule and maximum likelihood rule are equivalent.*

*Proof.* Recall Bayes' rule:
$$\mathbb{P}(A \wedge B) = \mathbb{P}(A \mid B) \cdot \mathbb{P}(B) = \mathbb{P}(B \mid A) \cdot \mathbb{P}(A).$$

Therefore,

$$\mathbb{P}(c \text{ sent and } x \text{ received}) = \mathbb{P}(c \text{ sent } \mid x \text{ received})\mathbb{P}(x \text{ received}) = \mathbb{P}(x \text{ received } \mid c \text{ sent})\mathbb{P}(c \text{ sent}).$$

In our case,
$$\mathbb{P}(c \text{ sent } \mid x \text{ received}) = \frac{\mathbb{P}(c \text{ sent})}{\mathbb{P}(x \text{ received})}\mathbb{P}(x \text{ received } \mid c \text{ sent}),$$

so maximizing $\mathbb{P}(c \text{ sent } \mid x \text{ received})$ is the same as maximizing $\mathbb{P}(x \text{ received } \mid c \text{ sent})$.

□

**Proposition 1.** *If $p < \frac{1}{2}$, then Maximum likelihood and Minimum distance are equivalent.*

*Proof.* Let $r = d(x, c)$. Then $\mathbb{P}(x \text{ received} \mid c \text{ sent}) = p^r(1-p)^{n-r}$, so minimizing this is equivalent to minimizing $r$. $\square$

**Definition** (Discrete Memoryless channel.)**.** Let $\Sigma_1 = \{\alpha_1, \dots, \alpha_N\}$ and $\Sigma_2 = \{b_1, \dots, b_M\}$ be two alphabets, with sent messages in $\Sigma_1$ and received messages in $\Sigma_2$. Let

$$P_{ij} = \mathbb{P}(b_j \text{ received} \mid a_i \text{ sent}).$$

We assume that $P_{ij}$ is independent of all past and future use of the channel. It is often convenient to represent the probabilities as a matrix called the channel matrix. This is a $N \times M$ stochastic matrix.

**Example.**

- **Binary symmetric channel.** $\Sigma_1 = \Sigma_2 = \{0, 1\}$.

$$\mathcal{P} = \begin{pmatrix} 1-p & p \\ p & 1-p \end{pmatrix}$$

- **Binary erasure channel.** $\Sigma_1 = \{0, 1\}$, $\Sigma_2 = \{0, 1, *\}$. This channel does not allow for incorrect symbols to be transmitted, but symbols can be lost, resulting in erasures.

$$\mathcal{P} = \begin{pmatrix} 1-p & 0 & p \\ 0 & 1-p & p \end{pmatrix}$$

We could also use the binary symmetric channel with $\mathbb{F}_q$ as our alphabet. If each incorrect symbol is equally likely the be received, then it is an easy exercise to write down the channel matrix. However, this might not always be the case. Perhaps $\Sigma_1 = \Sigma_2 = \mathbb{F}_5$, and when you send a 0 it is likelier that a 1 or 4 is received than a 2 or 3. This motivates some of the variations of the classical Hamming weight enumerator. We will discuss some of these issues in the next lecture.

Let $X$ be a (discrete) random variable taking values $x_1, \dots, x_N$, and let $P_i = \mathbb{P}(X = x_i)$. Recall the following basic definitions from probability theory:

- The expected value of $x$ is
$$\mu = \mathbb{E}[X] = \sum_{i=1}^{n} p_i x_i.$$
- The standard deviation $\sigma$ is given by
$$\sigma^2 = \sum_i p_i x_i^2 - \mu^2 = \mathbb{E}[X - \mu]^2.$$
The variance is $\sigma^2$.
- We say that $X$ and $Y$ are independent if for all $x_i$ and $y_j$ we have $\mathbb{P}(x_i \mid y_j) = \mathbb{P}(x_i)$.
If $x, y$ are independent,
$$\mathbb{E}[XY] = \mathbb{E}[X]\mathbb{E}[Y].$$
- The Gaussian distribution of mean $\mu$ and variance $\sigma^2$ is defined by
$$P(a \le x \le b) = \int_a^b \frac{1}{\sqrt{2\pi\sigma^2}} e^{\frac{-(x-\mu)^2}{2\sigma^2}}\, dx.$$

**Definition** (Additive Gaussian white noise channel)**.** Send signal with amplitude equal to some fixed $s > 0$ for the bit 1, and $-s$ for the bit 0. We receive receive $r = s + n$ where $n$ is drawn from the Gaussian distribution with mean 0 and variance $\sigma^2$. In decoding, we compare $r$ to 0. If it is positive, we decode as 1, and if it is negative we decode as 0.

We next define the binary entropy function. This arises in the statement of Shannon's Theorem as the channel capacity for the binary symmetric channel. Our goal here is to demonstrate that this function arises naturally given what we already know about coding theory. All logarithms in this section will be taken in base 2. We could easily adapt these definitions and results to general $q$, but leave this as an exercise for the interested reader.

**Definition.** Let

$$H(\delta) = -\delta \log \delta - (1-\delta)\log(1-\delta)$$

for $0 < \delta \leq \frac{1}{2}$.

We recall some standard analysis notation that will appear throughout the next few arguments.

**Definition.** Let $f, g$ be functions from $\mathbb{Z}$ to $\mathbb{Z}$. We say $f(n) = O(g(n))$ if $\exists\ C > 0$ such that $f(x) < Cg(x)$ for all $x \in \mathbb{N}$.
We say $f(n) = o(g(n))$ if $\lim_{n\to\infty} \frac{f(n)}{g(n)} = 0$.
We say $f(n) \sim g(n)$ if $\exists\ 0 < c < C$ such that $c < \lim_{n\to\infty} \frac{f(n)}{g(n)} < C$. Some authors use this notation only for the case where the limit is exactly equal to 1.

**Theorem 15** (Stirling)**.** *We have*

$$\ln n! = n \ln n - n + O(\ln n).$$

Recall that $V(n,r) = |B(n,r)|$ denotes the size of the sphere of radius $r$ around a particular point in $\{0,1\}^n$. We have seen that

$$V(n,r) = \sum_{i=0}^{r} \binom{n}{i}.$$

We will next show that the binary entropy function arises naturally when considering spheres of increasingly large size.

**Proposition 2.**

$$\lim_{n\to\infty} \frac{V(n, \lfloor n\delta \rfloor)}{n} = H(\delta).$$

*Proof.* Without loss of generality, we assume $0 < \delta \leq \frac{1}{2}$, so $0 \leq r \leq \frac{n}{2}$. So

$$\binom{n}{r} \leq V(n,r) \leq (r+1)\binom{n}{r}.$$

By Stirling,

$$\log \binom{n}{r} = -r \log\left(\frac{r}{n}\right) - (n-r)\log\left(\frac{n-r}{n}\right) + O(\log n) = nH\left(\frac{r}{n}\right) + O(\log n).$$

By the inequality on $V(n,r)$, we have

$$H\left(\frac{r}{n}\right) + O\left(\frac{\log n}{n}\right) \leq \frac{\log V(n,r)}{n} \leq H\left(\frac{r}{n}\right) + O\left(\frac{\log n}{n}\right).$$

Now let $r = \lfloor n\delta \rfloor$. In the case where $\frac{1}{2} < \delta < 1$, we can use the symmetry of binomial coefficients to swap $\delta$ and $1-\delta$ in the statement. □

## 5.2 The Proof of Shannon's Theorem

We will now recall some facts that will arise in the proof of Shannon's Theorem:

1. The probability of some error pattern with $w$ errors is $p^w(1-p)^{n-w}$. This depends only on the number of errors, not where the errors occur.
2. The expected number of errors is a random variable with mean $np$ and variance $np(1-p)$.
3. Chebyshev's inequality: Let $x$ be a random variable with mean $\mu$ and variance $\sigma^2$. For any $k > 0$,

$$\mathbb{P}(|x-\mu| \geq k\sigma) < \frac{1}{k^2}.$$

We want to use Chebyshev to give an upper bound on the probability that more than a certain number of errors occur in transmission. Let $\epsilon > 0$, $b = \left(\frac{np(1-p)}{\epsilon/2}\right)^{1/2}$. By Chebyshev,

$$\mathbb{P}(w > np + b) \leq \frac{\epsilon}{2}.$$

Since $p < \frac{1}{2}$, $\rho = \lfloor np + b \rfloor$ is less than $\frac{n}{2}$ for $n$ sufficiently large.

We next define two functions that will be useful in the proof. Let

$$f : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$$

defined by

$$f(u,v) = \begin{cases} 1 & \text{if } d(u,v) \leq p, \\ 0 & \text{otherwise.} \end{cases}$$

For $x_i \in C$, $y \in \{0,1\}^n$, define

$$g_i(y) = 1 - f(y, x_i) + \sum_{j \neq i} f(y, x_j).$$

Note that if $x_i$ is the only codeword within $\rho$ of $y$, this is zero; otherwise, it is at least 1.

**Theorem 16** (Shannon)**.** *Let $C$ be a code of $M$ words, $x_1, \ldots, x_M$, each sent with equal probability. Let $P_i$ be the probability of decoding incorrectly if $x_i$ is sent. Let*

$$P_C = \frac{1}{M}\sum_{i=1}^{M} P_i \quad \textit{and set} \quad P^*(M,n,p) = \min\{P_C\}$$

*over all codes $C$ with $M$ words of length $n$. Suppose $0 < R < 1 + p\log p + (1-p)\log(1-p) = 1 - H(p)$. If $M_n = 2^{\lfloor Rn \rfloor}$, then*

$$\lim_{n\to\infty} P^*(M_n, n, p) \to 0.$$

Recall that for a code of length $n$, the information rate is $R = \frac{\log_q |\mathcal{C}|}{n}$. So, for a code of since $2^{\lfloor Rn \rfloor}$, we have rate $\frac{\lfloor Rn \rfloor}{n}$ which is approximately $R$. Shannon's Theorem says if we consider codes of a fixed rate which is less than $1 - H(p)$ of increasing length, then we can find a family of codes which has the probability of making a decoding error approaching zero. In some sense, it guarantees that reliable communication over the binary symmetric channel is possible, but it does not tell us how to find the right family of codes.

*Proof.* We first outline the main ideas of the proof. We want to pick codewords randomly, uniformly from the space of all possible codewords. Suppose we receive $y$. If there is a unique codeword $x_i$ with $d(x_i, y) \leq \rho$, then we decode as $x_i$. Otherwise, we declare an error. We will show that the probability that we make a mistake with this simple decoding algorithm goes to zero.

We want to count how often there is a unique such codeword. The second idea is that $P^*(M,n,p)$ is at most $\mathbb{E}[P_C]$ as we vary over all possible $C$. There must exist some code which performs at least as well as the average of the performance of all such codes. We need only compute a certain expectation.

So

$$\begin{aligned} P_i &= \sum_{y\in\{0,1\}^n} \mathbb{P}(y \text{ received} \mid x_i \text{ sent}) g_i(y) \\ &= \sum_{y\in\{0,1\}^n} \mathbb{P}(y \mid x_i) \left(1 - f(x_i,y) + \sum_{y\in\{0,1\}^n} \mathbb{P}(y \mid x_i) \sum_{j\neq i} f(x_j,y)\right) \end{aligned}$$

The first term is the probability that $y$ is not in $B_\rho(x_i)$, which is less than $\frac{\epsilon}{2}$ by the above remarks. Summing, we find

$$P_C \leq \frac{\epsilon}{2} + \frac{1}{M}\sum_{i=1}^{M} \sum_{y\in\{0,1\}^n} \sum_{j\neq i} \mathbb{P}(y \mid x_i) f(x_j,y).$$

As we vary over all possible collections of $M$ codewords we see that the random variables $f(x_j,y)$ and $\mathbb{P}(y \mid x_i)$ are independent. Taking expectations,

$$\begin{aligned} P^* &\leq \frac{\epsilon}{2} + \frac{1}{M}\sum_{i=1}^{M} \sum_{y\in\{0,1\}^n} \sum_{j\neq i} \mathbb{E}[\mathbb{P}(y \mid x_i)]\mathbb{E}[f(x_j,y)] \\ &\leq \frac{\epsilon}{2} + \frac{1}{M}\sum_{i=1}^{m} \sum_{y\in\{0,1\}^n} \sum_{j\neq i} \mathbb{E}[\mathbb{P}(y \mid x_i)]\frac{V(n,\rho)}{2^n} \\ &= \frac{\epsilon}{2} + (M-1)\frac{V(n,\rho)}{2^n} \end{aligned}$$

The second inequality follows from the observation that sum of the term $\mathbb{E}[f(x_j,y)]$ counts the number of points within a ball of radius $\rho$ around any fixed $x_j$. Now taking logs of both sides and dividing by $n$,

$$\begin{aligned} \frac{\log(P^* - \frac{\epsilon}{2})}{n} &\leq \frac{\log(M-1)}{n} + \frac{\log V(n,\rho)}{n} - 1 \\ &\leq \frac{\log M}{n} - (1 + p\log p + (1-p)\log(1-p)) + O(n^{-1/2}), \end{aligned}$$

where the last expression follows from our bound on the size of $V(n,\rho)$.

Let $M = M_n$, so $\frac{\log M}{n} \approx R$, and is actually equal to $R$ plus a small error term. The above expression at most

$$R - (1 + p\log p + (1-p)\log(1-p)) < -\beta < 0,$$

for some $\beta > 0$. This implies Therefore,

$$P^*(M,n,p) < \frac{\epsilon}{2} + 2^{-\beta n},$$

which goes to zero as $n$ goes to infinity. □

Shannon's Noisy Channel Coding Theorem is just the beginning of the huge subject of Information Theory. We could ask for analogs of this theorem for other channels. In order to even state them, we need to determine the channel capacity of these channels, the upper bound for the largest rate at which reliable transmission can occur. We may also want to know other things. For example, what if we want to use codes not primarily to detect and correct errors, but as a method of data compression. How effectively can we use codes to compress messages when the cost of transmission is high? This is the motivation for Noiseless coding, just one of many other subjects in the area.

# 6 Weight Enumerators and the MacWilliams Theorem

Thanks to Tony Feng and Osbert Bastani for help typing this section.

In this lecture we will investigate weight enumerators of codes. This is a homogeneous polynomial that keeps track of the number of codewords of weight $i$ for each $i$. We will prove the MacWilliams Theorem stated in the first lecture and talk about some generalizations of this idea. Since this is one of my personal favorite subjects in the area and since it relates to some other interesting areas of mathematics, we will spend relatively more time on it.

## 6.1 Weight enumerators

Let $C$ be a linear code of length $n$. For $0 \leq i \leq n$, let

$$A_i = \#\{c \in \mathcal{C} \mid \text{wt}(c) = i\}.$$

The set $\{A_0, A_1, \ldots, A_n\}$ is known as the weight distribution of the code.

Question: What does the weight distribution of $C$ tell us about the weight distribution of $C^\perp$?

We will show that the answer to this question is, everything.

**Definition.** The weight enumerator of a not necessarily linear code $C$ is

$$W(x,y) = \sum_{i=0}^{n} A_i x^{n-i} y^i.$$

Recall that the $q$-ary Hamming code of length $\frac{q^r-1}{q-1}$, denoted $\mathcal{H}_{q,r}$, is defined as the code whose parity check matrix has columns consisting of one nonzero vector from each one dimensional subspace of $\mathbb{F}_q^r$, or equivalently, the columns consist of the points of $PG(r-1,q)$. Its dual is known as the $q$-ary Simplex code of length $\frac{q^r-1}{q-1}$ and is often denoted by $\mathcal{S}_{q,r}$. It is easy to see that the dimension of this code is $r$.

**Example.** For $\mathcal{S}_{3,3}$, the generator matrix is

$$G = \begin{pmatrix} 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 & 2 & 2 & 2 \\ 1 & 0 & 1 & 2 & 0 & 1 & 2 & 0 & 1 & 2 & 0 & 1 & 2 \end{pmatrix}.$$

**Proposition 3.** *Every non-zero element of $\mathcal{S}_{q,r}$ has weight $q^{r-1}$.*

*Proof.* We can express $C$ as $\{xG \mid x \in \mathbb{F}_q^r\}$. If $x \neq 0$, $\text{wt}(x \cdot G)$ is the number of columns $y_j$ of $G$ such that $xy^T \neq 0$. For any $x \in \mathbb{F}_q^r$, the space orthogonal to $x$ is isomorphic to $\mathbb{F}_q^{r-1}$. There are $\frac{q^{r-1}}{q-1}$ one-dimensional subspaces of $\mathbb{F}_q^{r-1}$. For any $x$, $\text{wt}(xG) = \frac{q^r-1}{q-1} - \frac{q^{r-1}-1}{q-1} = q^{r-1}$. □

Therefore we have computed

$$W_{\mathcal{S}_{q,r}}(X,Y) = X^{\frac{q^r-1}{q-1}} + (q^r-1) X^{\frac{q^{r-1}-1}{q-1}} Y^{q^r-1}.$$

Now let us try computing the weight enumerator for a binary hamming code $\mathcal{H}_{2,r}$. This will be easier than computing the weight enumerator of $\mathcal{H}_{q,r}$, but will be significantly tougher than the above computation for $\mathcal{S}_{q,r}$.

The length is $n = 2^r - 1$ and the parity check matrix consists of all non-zero binary vectors of length $r$. Note that if a word of this code has 1s in positions $i_1, \ldots, i_s$ only, then the sum of columns $i_1, \ldots, i_s$ is zero. Choose $i-1$ columns randomly. There are $\binom{n}{i-1}$ ways to do this. We now consider the possibilities for the sum of these columns. of three possibilities:

1. The sum of the columns is zero. This happens exactly $A_{i-1}$ times.
2. The sum of the columns is a column that has been chosen in the group of $i-1$. If that column is omitted, the remaining $i-2$ columns sum to zero, so this is a codeword of weight $i-2$. We could start with any codeword of weight $i-2$ and add any of the $n-(i-2)$ remaining columns. This happens $(n-(i-2)) \cdot A_{n-2}$ times.
3. The sum of the columns is one of the remaining columns. We choose the column corresponding to the sum, we obtain a codeword of weight $i$. Given any codeword of weight $i$, we can omit any column and end up in this situation. This happens $iA_i$ times.

Therefore,

$$iA_i = \binom{n}{i-1} - (n-(i-2))A_{i-2} - A_{i-1}.$$

Since we also know that $A_0 = 1$ and $A_1 = 0$, we can compute all of the values using this recurrence. This allows us to compute the weight enumerator, but is a little tedious. We get a much easier way to compute this weight enumerator from the MacWilliams Theorem.

## 6.2 Characters

In order to prove the MacWilliams Theorem we will develop some of the theory of characters on finite abelian groups. This is the beginning of a large area of research, discrete harmonic analysis.

**Definition.** A character on a finite abelian group $G$ is a homomorphism $\psi : G \to \mathbb{C}^*$, the multiplicative group of $\mathbb{C}$.

We say that $\psi$ is trivial if $\psi(x) = 1$ for all $x \in G$.

We will often use additive notation in the group $G$ and multiplicative notation in $\mathbb{C}^*$. So for a character $\psi$ and two elements $a, b \in G$, we have $\psi(a+b) = \psi(a)\psi(b)$.

Since $\psi(0) = \psi(0+0) = \psi(0)^2$, we must have $\psi(0) = 1$. If $|G| = n$, then for every $x \in G$ we have $nx = 0$, so $\psi((x)^n = 1$ and therefore $\psi(x)$ is an $n$th root of unity.

**Example.** 1. If $G = \mathbb{Z}/N\mathbb{Z}$, let $\omega = e^{2\pi i/N}$. Then for every $r \in G$ the function which sends $x$ to $\omega^{rx}$, which we call $\psi_r$, is a character. We note that if we add $N$ to $x$ then the function is unchanged.

All characters on $G$ have this form since $\psi(x)$ is determined by $\psi(1)$, as $\psi(1) = \omega^r$ implies that $\psi(x) = \omega^{rx}$.

2. If $G = \mathbb{F}_p^n$ we write $r \cdot x$ for $r_1x_1 + \cdots + r_nx_n$, where $r = (r_1, \ldots, r_n)$ and $x = (x_1, \ldots, x_n)$ are both in $G$. Now let $\omega = e^{2\pi i/p}$ The map which sends $x$ to $\omega^{r \cdot x}$, which we call $\psi_r$, is a character on $G$.

Again, all characters are of this form since $\psi$ is determined by its values at the $n$ vectors $(1, 0, \ldots, 0), (01, 0, \ldots, 0), \ldots, (0, \ldots, 0, 1)$.

We will leave it as an exercise to determine what happens for a general finite abelian group $G$.

It is very straightforward to show that the set of characters actually has a group structure.

**Proposition 20.** *Let $\widehat{G}$ be the set of characters of $G$. Then $\widehat{G}$ is a group under multiplication. That is, $(\psi_1\psi_2)(x) = \psi_1(x)\psi_2(x)$ for all $x \in G$. The trivial character is the unit of the group.*

It is also easy to see that in both of the cases described above the map taking $\psi_r$ to $r$ gives an isomorphism from $\widehat{G}$ to $G$.

We write $\overline{\psi}$ for the character given by the complex conjugate of the character $\psi$. It is an exercise to check that this is the inverse of $\psi$ in $\widehat{G}$.

**Proposition 21.** *If $\psi$ and $\chi$ are distinct characters on a finite abelian group $G$, then*

$$\sum_{x\in G} \psi(x)\overline{\chi(x)} = 0.$$

*Proof.* We write $\phi = \psi\overline{\chi}$. Since $\psi$ and $\chi$ are distinct, $\phi$ is nontrivial. So

$$\sum_{x\in G} \psi(x)\overline{\chi(x)} = \sum_{x\in G} \phi(x) = \sum_{x\in G} \phi(x+y) = \phi(y)\sum_{x\in G} \phi(x),$$

for any $y \in G$. Since $\phi$ is nontrivial, we can find some $y \in G$ such that $\phi(y) \neq 0$. This implies that $\sum_{x\in G} \phi(x) = 0$. □

We will now give an important definition in this area.

**Definition.** Let $f:\ G \to \mathbb{C}$. The Fourier Transform of $f$, $\hat{f}:\ \widehat{G} \to \mathbb{C}$ is given by

$$\hat{f}(\psi) = \frac{1}{|G|}\sum_{x\in G} f(x)\psi(x).$$

We compare this to the definition of the Fourier Transform of a function on $\mathbb{R}$,

$$\hat{f}(r) = \int_0^1 f(x)e^{2\pi i r x}dx.$$

We also note that this is slightly different from the definition of the Fourier Transform that we will use below. Translating from one to the other is an exercise.

**Proposition 22.** *1. Plancherel Identity: We have*

$$\sum_{\psi\in\widehat{G}} \hat{f}(\psi)\overline{\hat{g}(\psi)} = \frac{1}{G}\sum_{x\in G} f(x)\overline{g(x)}.$$

*2. Inversion Formula: We have*

$$f(x) = \sum_{\psi\in\widehat{G}} \hat{f}(\psi)\overline{\psi(x)}.$$

*3. We have $\widehat{\widehat{G}} \cong G$, but not canonically.*

*Proof.* Exercise. □

We note that there are corresponding statements for function on $\mathbb{R}^n$ for the first two facts. The discrete versions are much easier to prove.

## 6.3 Discrete Poisson Summation and the Proof of the MacWilliams Theorem

We now aim to prove the main result that will lead us to the proof of the MacWilliams Theorem, the Discrete Poisson Summation formula.

Let $f$ be a function on $G = \mathbb{F}_q^n$ and define

$$\hat{f}(\psi) = \sum_{x\in G} \psi(x)f(x),$$

for some fixed nontrivial character $\psi$ on $\mathbb{F}_q$. Note that $\psi(x) \neq 0$ for $x \neq 0$.

We will state and prove Discrete Poisson Summation in the form that we will need for the proof of the MacWilliams Theorem but will state more general version below.

**Proposition 23** (Discrete Poisson Summation)**.** *Let $C$ be a linear code of length $n$ defined over $\mathbb{F}_q$ and $f$ a function on $\mathbb{F}_q^n$. Then*

$$\sum_{u\in C^\perp} f(u) = \frac{1}{|C|}\sum_{v\in C}\hat{f}(v).$$

*Proof.* We identify $\widehat{G}$ with $G$ by identifying $g=(g_1,\dots,g_n)$ with the character $\psi_g$ defined such that

$$\psi(g,h) = \psi(\langle g,h\rangle) = \psi(g_1h_1+\cdots+g_nh_n) = \prod_{i=1}^n \psi(g_ih_i).$$

Now we consider the sum of the Fourier transform taken over the code $C$,

$$\sum_{v\in C}\hat{f}(v) = \sum_{v\in C}\sum_{h\in G}\psi_v(h)f(h) = \sum_{v\in C}\sum_{h\in G}\psi(\langle v,c\rangle)f(u).$$

This is a finite sum which stays the same if we switch the order of summation. We get

$$\sum_{v\in C}\hat{f}(v) = \sum_{h\in G} f(h)\sum_{v\in C}\psi_h(v).$$

We see that $\psi_h(v)$ is a character on the subgroup $C\subset G$. If this character is nontrivial, then $\sum_{v\in C}\psi_h(v)=0$, as we saw above in the proof of orthogonality of characters. If this character is trivial then the sum is $|C|$.

We see that $\psi_h(v)$ is trivial on $C$ if and only if $\langle v,h\rangle = 0$ for all $v\in C$, which occurs if and only if $h\in C^\perp$. Therefore

$$\sum_{v\in C}\hat{f}(v) = |C|\sum_{u\in C^\perp} f(u),$$

completing the proof. □

We now have all of the tools that we need to prove the MacWilliams Theorem.

**Theorem 17** (MacWilliams)**.** *We have*

$$W_{C^\perp}(X,Y) = \frac{1}{|C|}W_C(X+(q-1)Y, X-Y).$$

*Proof.* Let

$$\phi(c) = (X+(q-1)Y)^{n-\mathrm{wt}(c)}(X-Y)^{\mathrm{wt}(c)}.$$

We have

$$\sum_{c\in C}\phi(c) = W_C(X+(q-1)Y, X-Y).$$

By Discrete Poisson Summation, this is also equal to

$$\frac{1}{|C^\perp|}\sum_{u\in C^\perp}\hat{\phi}(u).$$

By definition, for $u\in C^\perp$,

$$\hat{\phi}(u) = \sum_{g\in\mathbb{F}_q^n}\psi(\langle g,u\rangle)\phi(g).$$

Writing $g=(g_1,g_2,...,g_n)$, note that $\phi(g)=\prod_{i=1}^n h(g_i)$, where

$$h(g_i) = \begin{cases} X-Y \text{ if } g_i\neq 0, \\ X+(q-1)Y \text{ if } g_i = 0 \end{cases}.$$

We now have

$$\begin{aligned}\hat{\phi}(u) &= \sum_{g_1\in\mathbb{F}_q}\sum_{g_2\in\mathbb{F}_q}\cdots\sum_{g_n\in\mathbb{F}_q}\psi(u_1g_1)\psi(u_2g_2)\cdots\psi(u_ng_n)h(g_1)h(g_2)\cdots h(g_n)\\ &= \prod_{i=1}^{n}\sum_{g_i\in\mathbb{F}_q}\psi(u_ig_i)h(g_i).\end{aligned}$$

If $u_i = 0$, then

$$\sum_{g\in\mathbb{F}_q}\psi(u_ig_i)h(g_i) = \sum_{g\in\mathbb{F}_q}h(g_i) = X+(q-1)Y+(q-1)(X-Y) = qX.$$

If $u_i \neq 0$, then

$$\begin{aligned}\sum_{g_i\in\mathbb{F}_q}\psi(u_ig_i)h(g_i) &= X+(q-1)Y+(X-Y)\sum_{g_i\neq 0}\psi(u_ig_i)\\ &= X+(q-1)Y-(X-Y) = qY,\end{aligned}$$

where the second to last step follows because

$$\sum_{g_i\in\mathbb{F}_q}\psi(u_ig_i) = 0,$$

and $\psi(0) = 1$.

Then we have

$$\hat{\phi}(u) = \prod_{i\ |u_i=0} qX \prod_{i\ |u_i\neq 0} qY = q^nX^{n-\mathrm{wt}(u)}Y^{\mathrm{wt}u}.$$

Therefore

$$\frac{1}{|C^\perp|}\sum_{u\in C^\perp}q^n\hat{\phi}(u) = |C|W_{C^\perp}(X,Y),$$

since

$$|C| = \frac{q^n}{|C^\perp|}.$$

□

## 6.4 Some Applications of the MacWilliams Theorem

Recall that the simplex code $\mathcal{S}_{q,r}$ is the dual of the Hamming code $\mathcal{H}_{q,r}$, has length $n = \frac{q^r-1}{q-1}$, minimum distance $d = q^{r-1}$, and weight enumerator

$$W_{\mathcal{S}_{q,r}}(X,Y) = X^{\frac{q^r-1}{q-1}} + (q^r-1)X^{\frac{q^{r-1}-1}{q-1}}Y^{q^{r-1}}.$$

We saw that it was somewhat difficult to determine the weight enumerator of the Hamming code directly. Applying the MacWilliams Theorem, we see that

$$W_{\mathcal{H}_{q,r}}(X,Y) = \frac{1}{q^r}((X+(q-1)Y)^{\frac{q^r-1}{q-1}} + (X+(q-1)Y)^{\frac{q^{r-1}-1}{q-1}}(X-Y)^{q^{r-1}}.$$

We would like to be able to expand this out and determine how many codewords of $\mathcal{H}_{q,r}$ have weight $i$ for each $0 \leq j \leq \frac{q^r-1}{q-1}$. In order to give a nice form for this, we need the Krawtchouk polynomials. These are a special class of orthogonal polynomials that will play a major role when we talk about the Linear Programming upper bound for the length of a code.

**Definition.** Suppose that $q$ and $n$ are fixed. The we define the $k$th Krawtchouk polynomial

$$K_k^{n,q}(x) = \sum_{j=0}^{k} (-1)^j (q-1)^{k-j} \binom{x}{j} \binom{n-x}{k-j}.$$

**Exercise.** Show that for $0 \leq j \leq n$, the number of words of $\mathcal{H}_{q,r}$ of weight $j$ is

$$A_j = q^{-r} \left( K_j^{n,q}(0) + (q^r - 1) K_j^{n,q}(q^{r-1}) \right) = q^{-r} \left( (q-1)^j \binom{n}{j} + (q^r-1) K_j^{n,q}(q^{r-1}) \right).$$

We note that the MacWilliams Theorem can be used to give all sorts of results about weight enumerators of codes. For example, the weight enumerator of a Maximal Distance Separable code of length $n$ and dimension $k$ is completely determined by $n$ and $k$.

**Theorem 18.** *Let $C$ be an $[n, k, n-k+1]$ linear code. Recall that $C^\perp$ is also an MDS code. Then $A_0 = 1$, $A_i = 0$ for $1 \leq i \leq d-1$, and*

$$A_i = \binom{n}{i} \sum_{j=0}^{i-d} (-1)^j \binom{i}{j} (q^{i+1-d-j} - 1).$$

We will leave the proof of this as an exercise.

We have determined the weight enumerators of some codes and will show how to find the weight enumerators of several more. We do know the weight enumerator of $RM(1, m)$ from our earlier discussions on Hadamard matrices. For general Reed Muller codes $\mathcal{R}(r, m)$, we know the minimum distance, but determining the second smallest weight that occurs in the code is a very recent result. In all but a few cases, the entire weight enumerator is not known.

**Exercise.** Let $C_1$ and $C_2$ be not necessarily linear codes. We have

$$W_{C_1 \oplus \mathcal{C}_2}(X, Y) = W_{C_1}(X, Y) W_{C_2}(X, Y).$$

If two codes are equivalent, then they have the same weight enumerator, but the converse is false. Consider the binary codes $C_1$ and $C_2$ with generator matrices:

$$G_1 = \begin{pmatrix} 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 \end{pmatrix}, \quad G_2 = \begin{pmatrix} 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 \end{pmatrix}.$$

Both of these codes have $n = 6$ and $k = 3$. Replacing the third row of $G_2$ by the sum of all three rows gives

$$G_2' = \begin{pmatrix} 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 \end{pmatrix}.$$

**Exercise.** Check that

$$W_{C_2}(X, Y) = (X^2 + Y^2)^3 = X^6 + 3X^4Y^2 + 3X^2Y^4 + Y^6 = W_{C_1}(X, Y).$$

Applying MacWilliams, we have

$$W_{C_2^\perp}(X, Y) = (X^2 + Y^2)^3 = W_{C_2}(X, Y).$$

Since $C_1$ and $C_2$ have the same weight enumerator, their duals have the same weight enumerator as well. We see that $C_1,\ C_2,\ C_1^\perp,\ C_2^\perp$ all have the same weight enumerator.

How can we determine whether $C_1$ is equivalent to $C_2$? Similarly, we would like to know whether either code is self-dual. We could just compute the dual of each code, but we will instead will answer these questions by looking at weights of subcodes.

Look at 2-dimensional subcodes of these 3-dimensional codes. Take 2 of the 7 nonzero codewords $a, b$, which generate a two dimensional space with codewords, 0, $a$, $b$, and $a+b$. There are $\binom{7}{2}/\binom{3}{2} = 7$ 2-dimensional subcodes. These correspond to lines in $PG(2,2)$, which are planes in $AG(3,2)$.

We define the weight of a subspace as the size of the support of that subspace. That is, the number of coordinates of the code for which there is some codeword with a nonzero entry in that coordinate. We can also think of this as the number of nonzero columns of any generator matrix of the code. For example,

$$\mathrm{wt}\left(\begin{pmatrix} 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 \end{pmatrix}\right) = 3,$$

and

$$\mathrm{wt}\left(\begin{pmatrix} 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 \end{pmatrix}\right) = 4.$$

We can tell codes apart by look at 'higher weight distributions' of the code. We take all $m$-tuples of codewords from $C$ and write a generator matrix for the subcode of $C$ that it generates and determine the weight of that subspace. This gives us $|C|^m$ weights.

For example, the weight distribution for 2-dimensional subcodes of $C_1$ and $C_2$ are

$$\begin{pmatrix} 0 & 0 & 0 & 1 & 0 & 3 & 3 \end{pmatrix}, \quad \begin{pmatrix} 0 & 0 & 0 & 0 & 3 & 0 & 4 \end{pmatrix}.$$

We also consider the one-dimensional subcodes generated by pairs of vectors. If we have some $c \in C$ then the pairs $(0,c)$, $(c,0)$, $(c,c)$ all generate the same one-dimensional code. We also note the subcode generated by $(a,b)$ is also generated by $(b,a)$. The trivial subcode is only generated by $(0,0)$. We therefore see that the weight distribution from taking all 2-tuples of vectors from $C_2$ is given by

$$(1,0,0,0,0,0,0) + 3(0,0,3,0,3,0,1) + 6(0,0,0,1,0,3,3) = (1,0,9,6,9,18,21).$$

There is a MacWilliams theorem for these higher weight distributions.

**Definition.** Let $C$ be a linear code of length $n$, and let $\mathrm{el}(c_1,\ldots,c_m)$, where $c_1,\ldots,c_m \in C$, be the number of nonzero columns of the $m \times n$ matrix with rows $c_1,\ldots,c_m$. Let the $m$-tuple support weight enumerator of $C$ be given by

$$W_C^[m](x,y) = \sum_{c_1,\ldots,c_m \in C} X^{n-\mathrm{el}(c_1,\ldots,c_m)} Y^{\mathrm{el}(c_1,\ldots,c_m)}.$$

Note that when $m = 1$ this is exactly equal to the classical Hamming weight enumerator.

**Theorem 19** (MacWilliams for $m$-tuple support weight enumerators)**.** *We have*

$$W_{C^\perp}^{[m]}(X,Y) = \frac{1}{|C|^m} W_C^{[m]}(X + (q^m - 1)Y, X - Y).$$

Again, note that when $m = 1$ this is exactly the MacWilliams Theorem.

**Exercise.** Use this theorem to show that

$$W_{C_1^\perp}^{[2]}(X,Y) = X^6 + 9X^4Y^2 + 6X^3Y^3 + 9X^2Y^4 + 18XY^5 + 21Y^6 = W_{C_1}^{[2]}(X,Y).$$

Show that even though the higher weight enumerators of $C_1$ and $C_1^\perp$ are all the same, $C_1$ is not self-dual.

Also compute $W_{C_2}^{[2]}(X,Y)$ and use this version of the MacWilliams Theorem to compute $W_{C_2^\perp}^{[2]}(X,Y)$. Show that $C_1$ and $C_2$ are not equivalent, and show that $C_2$ is self-dual.

One of the reasons that we have spent comparatively more time discussing weight enumerators and the MacWilliams Theorem is that the weight distributions of codes arising in algebraic geometry give us interesting geometric information.

Consider the set of linear forms $aX + bY + cZ$, where $a, b, c \in \mathbb{F}_q$, on $\mathbb{P}^2(\mathbb{F}_q)$, which is just $PG(2, q)$. We can choose explicit representatives for the points of $\mathbb{P}^2(\mathbb{F}_q)$, for example the set $[1 : a : b]$, $[0 : 1 : a]$, $[0 : 0 : 1]$. This notation identifies an affine line with a projective point by identifying $[a : b : c]$ with the line through the origin in $AG(3, q)$ which is the set of points $\alpha(a, b, c)$ where $\alpha \in \mathbb{F}_q$.

Choose an ordering of the $q^2 + q + 1$ points. A linear form gives a codeword in $\mathbb{F}_q^{q^2+q+1}$ where the $i$th entry is that form evaluated on the affine point $(a, b, c)$ if the $i$th projective point in our ordering is $[a : b : c]$. Choose a linear form and evaluate it at each of these $q^2 + q + 1$ points. Linear forms give a vector space of dimension 3, so this set forms a linear code with $k = 3$ and $n = q^2 + q + 1$. Every codeword has exactly $q + 1$ zeros, since the zero set of each hyperplane in $\mathbb{P}^2(\mathbb{F}_q)$ is a line, which has $q + 1$ points, so we have

$$W_C(X, Y) = X^{q^2+q+1} + (q^3 - 1)X^{q+1}Y^{q^2}.$$

This is exactly the simplex code $\mathcal{S}_{q,3}$. Its dual is a Hamming code $\mathcal{H}_{q,r}$. Since the codewords of the simplex code come from something geometric, linear forms in projective space, dual codewords should also correspond to something geometric. Here are the first few terms of the dual code ordered by weight:

$$\frac{1}{q^3} W_C(X + (q - 1)Y, X - Y) = X^{q^2+q+1} + (q - 1)\frac{(q^3 - 1)(q + 1)q}{6} X^{q^2+q-2}Y^3 + O(Y^4).$$

The first term comes from the zero codeword. The coefficient of the next term is exactly $q - 1$, the number of nonzero scalars in $\mathbb{F}_q$, times the number of collections of three collinear points in $\mathbb{P}^2(\mathbb{F}_q)$. If we continue this analysis, we find similar coefficient that come from collections of points in projective space that impose dependent conditions on linear forms. Some main tools in this area come from the generalizations of the classical Hamming weight enumerator described above.

# 7 Bounds on Codes

Thanks to Osbert Bastani and Azalea Vo for help typing up this section.

## 7.1 An Introduction to Bounds

Recall that $A_q(n, d)$ is the maximal number of words of a code of length $n$ and distance $d$ over $\mathbb{F}_q$. We will first discuss the bounds which we have already seen.

Earlier, we proved the sphere packing bound:

$$A_q(n, d) \le \frac{q^n}{\sum_{i=0}^{\lfloor \frac{d-1}{2} \rfloor} \binom{n}{i}(q - 1)^i}.$$

We proved this by putting a sphere of radius $e := \lfloor \frac{d-1}{2} \rfloor$ around each word. If a code achieves equality, then the code is perfect. We have already seen some perfect codes, the Hamming codes and binary repetition codes of odd length, and we will later see the Golay codes.

We also saw a lower bound, the Gilbert bound:

$$A_q(n, d) \ge \frac{q^n}{\sum_{i=0}^{d-1} \binom{n}{i}(q - 1)^i}.$$

We proved this by putting a sphere of radius $d - 1$ around each codeword, which necessarily covers $\mathbb{F}_q^n$. We note that this is equivalent to

$$A_q(n, d) \ge q^{n - \log_q(\sum_{i=0}^{d-1} \binom{n}{i}(q-1)^i},$$

which is closer to the form of the Varshamov bound, another slightly stronger lower bound that we will prove in this section.

We have seen the Singleton bound, $A_q(n,d) \leq q^{n-d+1}$, and recall that a code which give equality for this bound is known as a Maximum Distance Separable or MDS code. We proved this bound by taking a code of minimal distance $d$ and puncturing it $d-1$ times. We will see a related bound on the length of linear codes, the Griesmer bound, which uses a similar puncturing idea.

We will also see a few other bounds, the Plotkin bound and its more difficult generalization the Elias bound, and also the linear programming bound.

We first prove an elementary property which allows us to focus on computing $A_2(n,d)$ for only odd values of $d$.

**Theorem 20.** *Let $d > 1$. Then*

1. $A_q(n,d) \leq A_q(n-1,d-1)$,
2. *if $d$ is even, then* $A_2(n,d) = A_2(n-1,d-1)$.

*Proof.* For the first part, let $C$ be a code of length $n$, distance $d$, with $M$ words, where $M$ is maximal. Puncturing once gives $M$ different codewords, because if two were the same, then they would have distance 1 from each other. This proves the first statement.

For the second statement, we need to show that $A_2(n,d) \geq A_2(n-1,d-1)$. Let $C$ be a binary code with $M$ words, where $M$ is maximal, length $n-1$, and distance $d-1$. Extend $C$ by adding a coordinate so that the weight of every word is even. Since $d-1$ is odd, two words with distance $d-1$ from each other have distance $d$ when extended. This is because two words of distance $d-1$ must have an odd number of places in which one word has a 1 and the other has a 0, which implies that one word has odd weight and the other word has even weight. $\square$

For $A_2(n,d)$, we only need to consider $d$ even. It is known for all $n \leq 16$ and $d \leq 10$. It was determined that $A_2(17,8) = 36$ in a recent paper of Östergård, after decades of knowing that $36 \leq A_2(17,8) \leq 37$. There are still a huge number of unknown values of $A_2(n,d)$ the smallest of which are

$$64 \leq A_2(18,8) \leq 72, \qquad 256 \leq A_2(17,6) \leq 340, \qquad 2720 \leq A_2(16,6) \leq 3276.$$

Any improvements to these bounds would be quite interesting to coding theorists.

For $q = 2$ and even $d$, $A_2(n,d) = A_2(n-1,d-1)$. For the rest of this section we define $r := 1 - \frac{1}{q} = \frac{q-1}{q}$.

## 7.2 The Plotkin Bound

The following bound is often is better than the sphere packing bound when $d$ is close to $n$.

**Proposition 4.** *[Plotkin] Let $C$ be an $(n, M, d)$ code over $\mathbb{F}_q$ with $rn < d$. Then $M \leq \lfloor \frac{d}{d-rn} \rfloor$.*

*Proof.* Let $S := \sum_{x\in C}\sum_{y \in C} d(x,y)$. For $x \neq y,\ d(x,y) \geq d$, so $M(M-1)d \leq S$.

We consider the $M \times n$ matrix where the rows are the codewords. For $1 \leq i \leq n$, let $m_{i,\alpha}$ be the number of times $\alpha \in \mathbb{F}_q$ appears in column $i$. Our plan is to count the total distance between ordered pairs of codewords by looking one column at a time. We see that

$$\sum_{\alpha \in \mathbb{F}_q} m_{i,\alpha} = M,$$

for each $1 \leq i \leq n$, so

$$S = \sum_{i=1}^{n} \sum_{\alpha\in\mathbb{F}_q} m_{i,\alpha}(M - m_{i,\alpha}) = nM^2 - \sum_{i=1}^{n}\sum_{\alpha\in\mathbb{F}_q} m_{i,\alpha}^2.$$

By Cauchy-Schwarz

$$\left(\sum_{\alpha\in\mathbb{F}_q} m_{i,\alpha}\right)^2 \leq q \sum_{\alpha\in\mathbb{F}_q} m_{i,\alpha}^2.$$

Now we have

$$S \leq nM^2 - \frac{1}{q}\sum_{i=1} n \left(\sum_{\alpha\in\mathbb{F}_q} m_{i,\alpha}\right)^2 = nM^2 - \frac{nM^2}{q} = nrM^2.$$

This implies

$$M(M-1)d \leq nrM^2, \text{ and therefore, } M \leq \frac{d}{d-nr}.$$

Since $M$ is an integer, we are free to take the floor of the righthand side, completing the proof. □

When $q = 2$ we can improve on this slightly.

**Proposition 5.** *For $q = 2$, $A_q(n,d) \leq \lfloor \frac{d}{2d-n} \rfloor$ when $n < 2d$.*

We leave the proof as an exercise.

**Example.** We know $A_2(13,5) = A_2(14,6)$. Consider a $(14, M, 6)$ code. We choose any three positions. There are eight possibilities for the 3-tuple of symbols occurring in these positions. We take the one that occurs most often and call it $v$. Consider only the codewords which have $v$ in the three positions. Then puncture the code in these three places, leaving a code with parameters $(11, M', 6)$. This construction is known as 'shortening' a code and is useful for constructing several kinds of interesting codes.

We have $M \leq 8M' \leq 8\frac{6}{6-\frac{11}{2}} = 96$. Compare this to the sphere packing bound which gives us $A_2(13,5) \leq 89$.

Similarly, consider shortening a $(17, M, 8)$ code in two positions, giving $A_2(17,8) \leq 4 \cdot A_2(15,8) \leq 64$. We will return to this example throughout this section.

**Example.** Let $q = 3, n = 13, d = 9$, so $r = \frac{2}{3}$. The Plotkin bound implies $A_3(13,9) = \left\lfloor \frac{9}{9-\frac{26}{3}} \right\rfloor = 27$.

We consider the simplex code $\mathcal{S}_{3,3}$ which has the columns of its generator matrix given by the $3^2+3+1$ points of the projective plane of order 3. We have seen that this code has minimum distance 9 and 27 words, so it gives equality in the Plotkin bound.

## 7.3 Griesmer Bound and the Varshamov Bound

In order to prove this upper bound we must develop the properties of residual codes. This is another way of taking a longer code and getting a shorter code from it with related parameters. Let $C$ be a linear $[n,k]$ code and $c \in C$ with weight $w$, and $I$ the set of coordinates of $c$ that are nonzero.

**Definition.** Let $C$ be an $[n,k,d]$ code over $\mathbb{F}_q$ and let $c \in C$ have weight $w$. The residual code, $\mathrm{Res}(C,c)$ is the code of the length $n-w$ punctured on all coordinates of $I$.

**Theorem 21.** *Let $C$ be an $[n,k,d]$ code over $\mathbb{F}_q$ and let $c \in C$ have weight $w < \frac{dq}{q-1}$. Then $\mathrm{Res}(C,c)$ is an $[n-w, k-1, d']$ code with $d' \geq d - w\lceil \frac{w}{q} \rceil$*

*Proof.* Without loss of generality, by rearranging coordinates and multiplying some columns of the generator matrix by a nonzero scalar we can find an equivalent code to $C$ where $c = (1,1,\cdots,1,0,\cdots,0)$ has weight $w$. We can take a generator matrix for $C$ with $c$ as its first row. Puncturing on these initial $w$ positions gives the 0 vector as the first row of our new generator matrix, so $dim(\mathrm{Res}(C,c)) \leq k-1$. Now we need to show that $\dim(\mathrm{Res}(C,c)) \geq k-1$.

Suppose not. Then we claim there exists a codeword of $C$, $x = (x_1, x_2, \ldots, x_n)$ which is not a multiple of $c$ and has all zeros in its last $n-w$ coordinates. If the dimension of the residual code decrease by more than one, then either there is a second row of this new generator matrix equal to zero, or there is a linear dependence among the nonzero rows. In this first case, this second vector which reduces to zero proves our claim, since if it was a multiple of $C$ then the original word of length $n$ could not be a row of the original generator matrix. So, we can suppose the second case holds. There must be two codewords of the residual code with the same $x_{w+1}\cdots x_n$ and take the difference between them. This proves the claim.

Now, by the Pigeonhole Principle, among the first $w$ coordinates, $x_1 \cdots x_w$ with $q$ symbols to distribute, there must be some $\alpha$ that occurs at least $\lceil \frac{w}{q} \rceil$ times. Then we have

$$d \leq \mathrm{wt}(x - \alpha c) \leq w - \frac{w}{q} = w\frac{q-1}{q},$$

which is a contradiction since we assumed that $w < \frac{dq}{q-1}$. Therefore, $dim(\mathrm{Res}(C, c)) = k - 1$.

Let $x_{w+1} \cdots x_n \in \mathrm{Res}(C, c)$ and $x_1 \cdots x_n$ be a corresponding $x \in C$. There exists $\alpha \in \mathbb{F}_q$ that occurs at least $\lceil \frac{w}{q} \rceil$ times in $x_1 \cdots x_w$. Then we have

$$d < \mathrm{wt}(x - \alpha c) \leq w - \left\lceil \frac{w}{q} \right\rceil + \mathrm{wt}(x_{w+1} \cdots x_n).$$

This implies $d' \geq d - w + \lceil \frac{w}{q} \rceil$ as desired. □

The following corollary is immediate.

**Corollary 3.** *If $C$ is an $[n, k, d]$ code over $\mathbb{F}_q, c \in C$ and $wt(c) = d$, then $\mathrm{Res}(C, c)$ is an $[n - d, k - 1, d']$ code with $d' \geq \lceil \frac{d}{q} \rceil$.*

We can now give our next upper bound.

**Theorem 22** (Griesmer Bound)**.** *We have*

$$n \geq \sum_{i=0}^{k-1} \left\lceil \frac{d}{q^i} \right\rceil.$$

*Proof.* We argue by induction using our results on residual codes. It is trivial to check $k = 1$.

Suppose $k > 1$ so $\mathrm{Res}(C, c)$ is $[n - d, k - 1, d'], d' \geq \lceil \frac{d}{q} \rceil$. By induction

$$n - d \geq \sum_{i=0}^{k-2} \left\lceil \frac{d}{q^{i+1}} \right\rceil.$$

This completes the proof. □

**Example.** The Griesmer bound implies that the maximum dimension of a $[13, k, 5]$ linear code is 6. In fact, no such code exists, but we will see later that a $(13, 64, 5)$ nonlinear code can be constructed from the extended binary Golay code.

**Example.** Consider the simplex code $\mathcal{S}_{q,r}$ with $d = q^{r-1},\ n = \frac{q^r - 1}{q-1}$. Then

$$n = \sum_{i=0}^{r-1} q^i = \sum_{i=0}^{r-1} \left\lceil \frac{q^{r-1}}{q^i} \right\rceil.$$

Therefore, we have equality in the Griesmer bound.

**Example.** Suppose we have $q = 3$ and an $[11, 6, 5]$ or a $[12, 6, 6]$ code. Then $\sum_{i=0}^{5} \lceil \frac{5}{3^i} \rceil = 5+2+1+1+1+1 = 11$ and $\sum_{i=0}^{6} \lceil \frac{6}{3^i} \rceil = 6 + 2 + 1 + 1 + 1 + 1 = 12$. A code with these parameters would give equality in the Griesmer bound. We will see in the next section that these codes are the ternary Golay code and the extended ternary Golay code.

We next state two results whose proofs we will leave as exercises. These are interesting since meeting the Griesmer bound tells you something interesting about the form of the generator matrix of such a code.

**Theorem 23.** *Let $C$ be a binary $[n,k,d]$ code with equality in the Griesmer bound. Then $C$ has a basis of codewords of minimal weight $d$.*

We note that the analog of this result over $\mathbb{F}_q$ is true as well. This result does not hold for general codes.

The next result shows that not every code has a basis of minimal weight vectors, but there is a code with the same parameters with a basis of minimal weight vectors.

**Theorem 24.** *Suppose $C$ is a general $[n,k,d]$ code over $\mathbb{F}_q$ then there is some $C'$, an $[n,k,d]$ code with a basis of vectors of weight $d$.*

We will give an idea of how to prove these results in the exercises.

We next turn to a lower bound that is slightly better than the Gilbert bound. The main lemma of the proof will be left as an exercise which we will give hints for in the exercises.

**Lemma 3.** *Let $n,k,d$ be such that $2 \le d \le n$ and $1 \le k \le n$, and $q$ a prime power. If $\sum_{i=0}^{d-2} \binom{n-1}{i}(q-1)^i < q^{n-k}$ , then there exists an $(n-k)\times n$ matrix $H$ over $\mathbb{F}_q$ such that every $d-1$ columns of $H$ are linearly independent.*

The proof is left as an exercise. The idea is that we will construct the columns of the matrix one at a time using a kind of greedy procedure. Each choice of the previous columns will rule out some possibilities for the next column, but the numerical condition is enough to show that there is always some possibility for the next column.

We note that given a code of minimum distance $d$ all of its subcodes have minimum distance at least $d$. This corollary follows immediately.

**Corollary 4.** *Let $2 \le d \le n$ and $1 \le k \le n$, then there is an $[n,k]$ code over $\mathbb{F}_q$ with minimum distance at least $d$ if $1+\sum_{i=0}^{d-2}\binom{n-1}{i}(q-1)^i \le q^{n-k}$.*

**Theorem 25** (Varshamov Lower Bound)**.** *We have*

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

*Proof.* Let $L = 1+\sum_{i=0}^{d-2}\binom{n-1}{i}(q-1)^i$. There is an $[n,k]$ code with minimum distance at least $d$ provided that $\log_q(L) \le n-k$. The largest $k$ such that $k \le n-\log_q(L)$ is $n-\lceil \log_q(L)\rceil$.
So $A_q(n,d) \ge q^{n-\lceil \log_q(L)\rceil}$. □

**Example.** The Varshamov bound implies that $A_2(13,5) \ge \lfloor \frac{8192}{1093} \rfloor = 8$. This is a very poor lower bound in this case since we can easily construct a binary code of length 13 and minimum distance 5 and more than 8 words.

Consider $\mathcal{R}(1,4)$ of length 16 and with $d=8$. We puncture this code three times and get a code with $n=13,\ d=5$, and $k=\dim(\mathcal{R}(1,4))=5$. This gives 32 codewords, so $A_2(13,5)\ge 32$.

## 7.4 Elias Bound

We will now give a bound which is a generalization of the Plotkin bound. The idea of the proof is similar. We will determine the total distance between ordered pairs of codewords by considering how many times each symbol occurs in each columns of the matrix with rows given by our $M$ codewords. We will consider the symbol 0 and nonzero symbols separately. The key idea is that we will only consider the codewords which lie in a certain sphere around the zero codeword.

**Theorem 26** (Elias Bound)**.** *Let $q,n,d,w\in\mathbb{N},\ q\ge 2,\ r := 1-q^{-1}$ with $w \le rn$ and $w^2-2nrw+rnd>0$. Then,*

$$A_q(n,d) \le \frac{rnd}{w^2-2rnw+rnd}\cdot\frac{q^n}{V_q(n,w)}.$$

We will first give applications of this bound in the example we have considered so far. We give the value of $w$ and then the corresponding upper bounds on $A_2(13,5)$ and on $A_2(14,6)$. We get

| $w$ | $A_2(13,5)$ | $A_2(14,6)$ |
|---|---|---|
| 0 | 8192 | 16384 |
| 1 | 927 | 1581 |
| 2 | 275 | 360 |
| 3 | 281 | 162 |
| 4 | $-$ | 233 |

.

It is an exercise to check that these are the correct values. We know that in fact $A_2(13,5) = A_2(14,6)$ and see that the best upper bound is given by $w = 3$ for which we see that $A_2(14,6) \leq 162$. This is worse than the sphere packing bound in this case.

In order to prove the Elias bound we begin with a lemma showing that we can choose a certain translation of our code and then focus on codewords which lie in a certain sphere.

**Lemma 4.** *If $A, C \subseteq \mathbb{F}_q^n$ there is some $x \in \mathbb{F}_q^n$ such that $\frac{|(x+A)\cap C|}{|A|} \geq \frac{|C|}{q^n}$.*

*Proof.* Take $x_0 \in F_q^n$ which maximizes $|(x_0 + A)\cap)|$. We have

$$\begin{aligned} |(x_0+A)\cap C| &\geq \frac{1}{q^n}\sum_{xin\mathbb{F}_q^n} |(x+A)\cap C| \\ &= \frac{1}{q^n}\sum_{xin\mathbb{F}_q^n}\sum_{a\in A}\sum_{c\in C}\{x+a\}\cap\{c\} = \frac{1}{q^n}\sum_{a\in A}\sum_{c\in C} 1 = \frac{|A||C|}{q^n}. \end{aligned}$$

□

We can now give the proof of the Elias bound.

*Proof.* Let $A = B_w(0)$, the sphere of radius $w$ around 0 and let $C$ be an $(n, M, d)$ code with $M$ maximal. By translating $C$ we can suppose $x_0 = (0, \cdots, 0)$ in the above lemma. Consider $A \cap C$ which is an $(n, K, d)$ code with $K \geq M \cdot \frac{V_q(n,w)}{q^n}$.

We list these $K$ codewords as the rows of a $K \times n$ matrix. Let $m_{i,j}$ be the number of times the symbol $j$ occurs in column $i$. We know $\sum_{j=0}^{q-1} m_{i,j} = K$, and we define the total number of zeros in our codewords as $S$. So $\sum_{i=1}^{n} m_{i,0} = S$. Each of our $K$ codewords is in a sphere of radius $w$ around 0 by assumption and so has at least $n - w$ coordinates equal to zero. This gives $S \geq K(n-w)$.

We now have

$$\sum_{j=1}^{q-1} m_{i,j}^2 \geq \frac{1}{q-1}\left(\sum_{j=1}^{q-1} m_{i,j}\right)^2 = \frac{1}{q-1}(K - m_{i,0})^2,$$

by Cauchy-Schwarz. We also have

$$\sum_{i=1}^{n} m_{i,0}^2 \geq \frac{1}{n}\left(\sum_{i=1}^{n} m_{i,0}\right)^2 = \frac{S^2}{n}.$$

We now take the sum of the distances coming from all ordered pairs of rows by considering one coordinate at a time. From the bound we already have we see that

$$\begin{aligned} \sum_{i=1}^{n}\sum_{j=0}^{q-1} m_{i,j}(K - m_{i,j}) &= nK^2 - \sum_{i=1}^{n}\left(m_{i,0}^2 + \sum_{j=1}^{q-1} m_{i,j}^2\right) \\ &\leq nK^2 - \frac{1}{q-1}\left(\sum_{i=1}^{n}(q-1)m_{i,0}^2 + (K - m_{i,0})^2\right) \leq nK^2 - \frac{1}{q-1}\left(\frac{qS^2}{n} + nK^2 - 2KS\right). \end{aligned}$$

Now choose $w \leq rn$, so $S \geq K(n-w)$ implies that $S \geq \frac{nK}{q}$. We substitute this value for $S$. We expand and collect terms, which implies that the left hand side of the above inequality is at most

$$\begin{aligned} K^2 n - \frac{1}{q-1}\left(\frac{qK^2(n-w)^2}{n} + nK^2 - 2K^2(n-w)\right) &= K^2\left(\frac{-qw^2}{(q-1)n} + \frac{2wnq}{(q-1)n} - \frac{2w}{q-1} + n - \frac{n-2n+qn}{q-1}\right) \\ &= K^2\left(2w - \frac{w^2 q}{n(q-1)}\right) = K^2 w\left(2 - \frac{w}{rn}\right). \end{aligned}$$

Since there are $K(K-1)$ ordered pairs and each pair with $x \neq y$ have distance of at least $d$, we have

$$K(K-1)d \leq K^2 w(2 - \frac{w}{rn}).$$

Then $d \leq \frac{Kw}{K-1}(2 - \frac{w}{rn})$. So $\frac{K-1}{K} \leq \frac{2wrn - w^2}{drn}$. This implies

$$K \leq \frac{rnd}{w^2 - 2rnw + rnd},$$

completing the proof. □

Note that if we substitute $w = rn$ and suppose $d > rn$, then we get $\frac{d}{d-rn}$ as our upper bound, so the Elias bound implies the Plotkin bound.

We included the Elias bound in order to show that by building on the techniques we have seen earlier in more intricate ways we can get stronger bounds. The next bounds we will see involve totally new ideas. These are the linear programming bounds which give many of the best upper bounds on the size of codes that are currently known.

## 7.5 Krawtchouk Polynomials and the Linear Programming Bound

In this section we will give an abbreviated introduction to Krawtchouk polynomials. These are a type of orthogonal polynomials which have several different applications in coding theory. They show up in the statement of the linear programming bound and we have already seen them appear in the weight enumerator of the Hamming codes. They also play a major role in the classification of binary perfect codes. We will state several of their properties, leaving the proofs as exercises. Many of the properties of Krawtchouk polynomials follow from more general properties of orthogonal polynomials.

First we recall the definition.

**Definition.** Let $q \geq 2$ and $1 \leq k \leq n$. We let

$$K_k(x; n, q) = K_k^{n,q}(x) = \sum_{j=0} (-1)^j \binom{x}{j}\binom{n-x}{k-j}(q-1)^{x-j},$$

where $\binom{x}{j} = \frac{x(x-1)\dots(x-(j-1))}{j!}$ for $x \in \mathbb{R}$.

We often think of $n$ and $q$ as being fixed and write $K_k^{n,q}(x)$ and will sometimes omit the $n$ and $q$.

**Proposition 24.** *$K_k(x)$ is a polynomial in $x$ of degree $k$ with leading coefficient $\frac{(-q)^k}{k!}$.*

*Proof.* We expand the binomial coefficients and need only consider the terms with the maximum power of $x$. This gives

$$\sum_{j=0}^{k} (-1)^j \cdot \frac{x^j}{j!} \cdot \frac{(-x)^{k-j}}{(k-j)!} \cdot (q-1)^{k-j} = \sum_{j=0}^{k} \frac{(-1)^k x^k}{j!(k-j)!} \cdot (q-1)^{k-j},$$

and since $\binom{k}{j} = \frac{k!}{(k-j)!j!}$ by the binomial theorem this is $x^k \frac{(-1)^k(1+(q-1))^k}{k!}$. □

It is a little tedious to compute these polynomials directly from the definition, but it is a good exercise to try a few.

**Exercise.** For $q = 2$, we have

$$K_0(x;n) = 1;\ \ K_1(x;n) = -2x+n;\ \ K_2(x;n) = 2x^2-2nx+\binom{n}{2};\ \ K_3 = -\frac{4}{3}x^3+2nx^2-\left(n^2-n+\frac{2}{3}\right)x+\binom{n}{3}.$$

Also show that when $q = 2$ we have $K_k(x) = (-1)^k K_k(n-x)$.

**Exercise.** Recall the Taylor series for the exponential:

$$(1-z)^x = \sum_{j=0}^{\infty}(-1)^j\binom{x}{j}z^j.$$

1. Use this to show that

$$\sum_{k=0}^{\infty} K_k(x)z^k = (1+(q-1)z)^{n-x}(1-z)^x.$$

2. Use this to prove that

$$\sum_{i=0}^{n} K_l(i)K_i(k) = \delta_{l,k}q^n,$$

   where $\delta_{l,k} = 1$ if $l = k$ and is 0 otherwise.

   This is the relation that justifies the term 'othogonal polynomials' in this case.

3. Now replace $x$ with $x-1$ in the above expression and get $K_k(i) = K_k(i-1)-(q-1)K_{k-1}(i)-K_{k-1}(i-1)$. This relation shows that we can produce values $K_k(i)$ if we know smaller values of $K_k(x)$ and $K_{k-1}(x)$.

4. Differentiate to get

$$(k+1)K_{k+1}(x) = (k+(q-1)(n-k)-qx)K_k(x)-(q-1)(n-k+1)K_{k-1}(x).$$

   This three term recurrence is often the easiest way to compute $K_k(x)$ in practice.

We will next show how Krawtchouk polynomials arise natural when considering certain sums of characters.

**Proposition 25.** *Let $Q = \mathbb{Z}/q\mathbb{Z}$, let $\omega$ be a primitive $q$th root of unity, and let $x \in Q^n$ a fixed word of weight $i$. Then*

$$\sum_{y \in Q^n:\ \mathrm{wt}(y)=k} \omega^{\langle x,y\rangle} = K_k(i).$$

*Proof.* Without loss of generality, by permuting coordinates we can suppose $x = (x_1, ..., x_i, 0, ..., 0)$. Choose $k$ positions for $y$ to be nonzero. We label them $0 < h_1 < h_2 < ... < h_j \leq i$ and $i < h_{j+1} < ... < h_k \leq n$. Let $D$ be the set of words of weight $k$ in $Q^i$ with nonzero coordinates exactly $h_1, ..., h_k$. Then

$$\sum_{y\in D}\omega^{\langle x,y\rangle} = \sum_{y\in D}\omega^{x_1y_1+...+x_ny_n} = \sum_{y\in D}\omega^{x_{h_1}y_{h_1}}\cdots\omega^{x_{h_j}y_{h_j}} = (q-1)^{k-j}\prod_{l=1}^{j}\sum_{y_{h_l}\in\{Q\backslash 0\}}\omega^{x_{h_l}y_{h_l}},$$

since $y_{h_{j+1}}, \ldots, y_{h_k}$ can be any one of $q-1$ symbols. The last sum is always $-1$ since we are summing the character which sends $y$ to $\omega^{xy}$ over all nonzero $y \in Q$. Therefore the sum is $(-1)^j(q-1)^{k-j}$. We note that there are $\binom{i}{j}\binom{n-i}{k-j}$ ways to choose positions in $D$, completing the proof. □

For a linear code $C$ and any $c \in C$, the number of $x \in C$ such that $d(x,c) = i$ equals the number of $x \in C$ of weight $i$, since $d(x,c) = \mathrm{wt}(x-c)$ and $C$ is linear.

**Definition.** Let $C \subset Q^n$ be a possibly nonlinear code of $M$ words. Let

$$B_i = \frac{1}{M}\#\{(x,y):\ x,y \in C,\ d(x,y) = i\}.$$

The sequence $\{B_0, \dots, B_n\}$ is called the distance distribution of $C$. We note that when $C$ is linear, the distance distribution and weight distribution are identical.

Let $Q$ be any alphabet of size $q$, with some element labeled '0'. Let $\alpha:\ \mathbb{F}_q \to Q$ be a bijection satisfying $\alpha(0) = 0$. If $c = (c_1, ..., c_n)$, then define $\alpha(c) = (\alpha(c_1), ..., \alpha(c_n))$. Clearly $\mathrm{wt}(c) = \mathrm{wt}(\alpha(c))$ and $d(x,y) = d(\alpha(x), \alpha(y))$. Therefore this bijection preserves the distance distribution of a code. The following proposition then follows immediately.

**Proposition 26.** *There exists an $(n,M)$ code with a given distance distribution over $\mathbb{F}_q$ if and only if there exists an $(n,M)$ code over $\mathbb{Z}/q\mathbb{Z}$, or more generally, any alphabet of size $q$, with the same distance distribution.*

We need one more lemma involving Krawtchouk polynomials before stating the linear programming bound.

**Lemma 5.** *Let $\{B_i\}_{i=0}^n$ be the distance distribution of $C \subset (\mathbb{Z}/q\mathbb{Z})^n$. Then*

$$\sum_{i=0}^n B_i K_k(i) \geq 0,\ \textit{for}\ k \in \{0, ..., n\}.$$

*Proof.* By the previous lemma,

$$\begin{aligned} M\sum_{i=0}^n B_i K_k(i) &= \sum_{i=0}^n \sum_{\substack{x,y\in C\\ d(x,y)=i}} \sum_{\substack{z\in Q^n\\ \mathrm{wt}(z)=k}} \omega^{\langle x-y,z\rangle} \\ &= \sum_{\substack{z\in Q^n\\ \mathrm{wt}(z)=k}} \left(\sum_{x\in C}\omega^{\langle x,z\rangle}\right)\left(\sum_{y\in C}\omega^{\langle -y,z\rangle}\right) = \sum_{\substack{z\in Q^n\\ \mathrm{wt}(z)=k}} \left|\sum_{x\in C}\omega^{\langle x,z\rangle}\right|^2 \geq 0. \end{aligned}$$

□

Before proving our upper bound on codes, we give some general definitions and facts related to linear programming.

**Definition.** We describe a basic kind of linear programming problem. Let $b$ and $c$ be known coefficient vectors and let $A$ be a matrix of coefficients. Let $x$ be a vector with nonnegative entries. The problem is to maximize $c^T x$ subject to $Ax \leq b$.

Suppose $x$ is a vector in $\mathbb{R}^n$. Each inequality of $Ax \leq b$ gives a half-space in $\mathbb{R}^n$, where equality occurs along some hyperplane. The intersection of these halfspaces is a convex polyhedral cone. If this cone is bounded then it is a convex polytope.

**Example.** Maximize $f(x_1,x_2) = c_1x_1 + c_2x_2$ subject to $x_1, x_2 \geq 0$ and $a_{i1}x_1 + a_{i2}x_2 \leq b_i$ for $1 \leq i \leq 3$.

We also consider the example where we want to maximize $f(x,y) = x+y$ subject to $x, y \geq 0$, $\frac{4}{3}x + y \leq 8$, and $-\frac{2}{3}x + y \leq 2$. This gives a convex polygon in $\mathbb{R}^2$ which has vertices at $(0,0), (0,2), (3,4)$, and $(6,0)$. It is a general fact of linear programming that the maximum of this function must occur at one of these extreme points. It turns out that the maximum is achieved at $(3,4)$.

One basic way to solve linear programming problems is to use the Simplex Algorithm. The idea is that we first introduce 'slack variables' so that we can write our constraints as $Ax = b$. We then use the fact that our objective function reaches its maximum on some extreme point. We first have to find an extreme point. We then have to determine whether this point maximizes our function. If it does not, then there must be a boundary direction along which we can travel to a new extreme point and along which our function is non-decreasing. For details on this process, or for more sophisticated linear programming techniques consult one of the many books on the subject.

We now state the linear programming bound.

**Theorem 27** (Linear Programming Bound). *Let $q, n, d \in \mathbb{N}$, with $q$ a prime power. Then*

$$A_q(n,d) \leq \max\left\{\sum_{i=0}^{n} B_i :\ B_0 = 1,\ B_1 = ... = B_{d-1} = 0,\ B_i \geq 0,\ \sum_{i=0}^{n} B_i K_k(i) \geq 0\right\},$$

*for all $k \in \{0, ..., n\}$.*

*If $q = 2$ and $d$ is even, we can also suppose that $B_i = 0$ for $i$ odd.*

*Proof.* Let $\{B_0, ..., B_n\}$ be the distance distribution of a code. The above lemma gives

$$\sum_{i=1}^{n} B_i K_k(i) \geq 0.$$

All other constraints in the first part of the statement are obvious.

To see the remark, take $q = 2$ and $d$ even. Then puncture the code in some coordinate $i$ for which there exists a pair $x, y$ such that $d(x,y) = d$ and $x_i \neq y_i$. The only pairs $x, y$ in the punctured code with $d(x', y') = d - 1$ must have had distance $d$ in the original code and different values in the coordinate we punctured. It is clear that since $d - 1$ is odd, one of these words has odd weight and one has even weight. Therefore, when we extend this puncture code to $\overline{C}'$ we get an $(n, M, d)$ code with all codewords of even weight. □

We will state a very similar version of the Linear Programming bound. This comes from applying the duality theorem of linear programming. We will not prove this here, but note that it is not difficult given the results we have already established.

**Theorem 28** (LP Bound, Dual Version). *Let $\beta(x) = 1 + \sum_{k=1}^{n} \beta_k K_k(x)$, with $\beta_k \geq 0$ for $1 \leq k \leq n$ and $\beta(j) \leq 0$ for $j \in \{d, d+1, ..., n\}$. Then $A(n,d) \leq \beta(0)$.*

We close this section by applying the linear programming bound to the example we have considered throughout our discussion of bounds on codes.

**Example.** Earlier we used the sphere-packing bound to show that $A_2(13,5) = A_2(14,6) \leq 89$. Using the recurrence for Krawtchouk polynomials given above, we can compute the values which occur in the statement of the linear programming bound. This gives coefficient matrix

$$A = \begin{bmatrix} 14 & 2 & -2 & -6 & -10 & -14 \\ 91 & -5 & -5 & 11 & 43 & 91 \\ 364 & -12 & 12 & 4 & -100 & -364 \\ 1001 & 9 & 9 & -39 & 121 & 1001 \\ 2002 & 30 & -30 & 38 & -22 & -2002 \\ 3003 & -5 & -5 & 27 & -165 & 3003 \\ 3432 & -40 & 40 & -72 & 264 & -3432 \end{bmatrix},$$

and using the constraints from the linear programming bound gives a unique maximum subject to $B_0 = 1,\ B_i = 0$ for $1 \leq 5$ and $i \in \{7, 9, 11, 13\}$. This is $1 + B_6 + B_8 + B_{10} + B_{12} + B_{14} = 64$, with $B_6 = 42$, $B_8 = 7$, $B_{10} = 14$, and $B_{12} = B_{14} = 0$. In particular, this shows that $A_2(13,5) = A_2(14,6) \leq 64$. We know that any code of length 14 and minimum distance 6 with 64 words must have exactly this distance distribution.

We will see in the next section that in fact there is a $(13, 64, 5)$ nonlinear code, called $Y$, which we can construct from the extended binary Golay code.

## 7.6 Asymptotic Bounds

We first define the function which will be the focus of our study of asymptotic bounds.

**Definition.** Let $q$ be a prime power and

$$\alpha_q(\delta) = \lim_{n\to\infty} \sup \frac{\log_q(A_q(n, \delta n))}{n}.$$

Note that for a give $n$, the rate of a $(n, A_q(n, \delta n), \delta n)$ code is $\alpha_q(\delta)$. A code of length $n$ and $\delta n$ can correct approximately $\frac{\delta n}{2}$ errors. If we could compute $\alpha_q(\delta)$ then we would know the maximum possible rate that can be achieved as $n$ goes to infinity for a code that is able to correct a $\frac{\delta}{2}$ proportion of errors. Unfortunately, we do not know any non-trivial exact values of $\alpha_q(\delta)$, for any $q$. We will instead settle for upper and lower bounds which we will establish in this section.

First we generalize the entropy function that we first considered in the binary case during the proof of Shannon's Theorem.

**Definition.** Let $r := 1 - q^{-1}$ and define $H_q(0) = 0$ and

$$H_q(x) = x\log_q(q-1) - x\log_q(x) - (1-x)\log_q(1-x),$$

on $(0, r]$.

Note that this coincides with the binary entropy function when $q = 2$. We see that $H_q(x)$ is continuous and increases for 0 to 1 as $x$ increases from 0 to $r$.

We will first work towards proving the asymptotic version of the sphere-packing bound.

**Lemma 6.** *Let $0 \le \delta \le r$ and $q \ge 2$. Then*

$$\lim_{n\to\infty} \log_q(V_q(n, \lfloor \delta n \rfloor) = H_q(\delta).$$

We note that we have already proved a very similar result in the $q = 2$ case when we were proving Shannon's Theorem, so we will not give all of the details.

*Proof.* We recall that

$$V_q(n, \lfloor \delta n \rfloor) = \sum_{i=0}^{\lfloor \delta n \rfloor} \binom{n}{i} (q-1)^i.$$

As $n$ goes to infinity this sum is dominated by its largest term, which is the last. We have

$$\binom{n}{\lfloor \delta n \rfloor}(q-1)^{\lfloor \delta n \rfloor} \le V_q(n, \lfloor \delta n \rfloor) \le (1 + \lfloor \delta n \rfloor)\binom{n}{\lfloor \delta n \rfloor}.$$

Now taking logarithms and dividing by $n$ as we did in the binary case gives the result. □

**Proposition 27** (Asymptotic Sphere-Packing Bound)**.** *Let $0 < \delta \le r$. Then $\alpha_q(\delta) \le 1 - H_q(\frac{\delta}{2})$.*

*Proof.* The sphere-packing bound gives

$$A_q(n, \lfloor \delta n \rfloor) \le \frac{q^n}{V_q\left(n, \frac{\lfloor \lceil \delta n \rceil - 1 \rfloor}{2}\right)}.$$

For $n \geq N$ we have

$$\left\lfloor \frac{\lceil \delta n \rceil - 1}{2} \right\rfloor \geq \left\lfloor \frac{\delta n}{2} \right\rfloor \geq \left\lfloor \left(\delta - \frac{N}{2}\right) \frac{n}{2} \right\rfloor.$$

Therefore,

$$\alpha_q(\delta) \leq 1 - \frac{\log_q \left(V_q(n, \lfloor \frac{1}{2}(\delta - \frac{2}{N}) \frac{n}{2} \rfloor)\right)}{n} = 1 - H_q\left(\frac{1}{2}(\delta - \frac{2}{N})\right).$$

As $n$ goes to infinity, we take $N$ to infinity as well, which gives the result.

□

We next recall the Singleton bound $A_q(n,d) \leq q^{n-d+1}$. We can rewrite this as $A_q(n, \delta n) \leq q^{n(1-\delta)} q$. Taking logarithms and dividing by $n$ gives the following easy result.

**Proposition 28** (Asymptotic Singleton Bound). *We have $\alpha_q(\delta) \leq 1 - \delta$.*

We next recall the Gilbert bound,

$$A_q(n,d) \geq \frac{q^n}{V_q(n, d-1)}.$$

We also proved the Varshamov lower bound, but asymptotically, these lower bounds give the same result.

**Proposition 29** (Gilbert-Varshamov Bound). *We have $\alpha_q(\delta) \geq 1 - H_q(\delta)$.*

*Proof.* By the Gilbert bound,

$$\alpha_q(\delta) \geq \lim_{n \to \infty} 1 - \frac{\log_q V_q(n, \delta n)}{n} = 1 - H_q(\delta).$$

□

We next give as asymptotic version of the Plotkin bound, which says

$$A_q(n,d) \leq \left\lfloor \frac{d}{d - rn} \right\rfloor.$$

**Proposition 30.** *We have*

$$\alpha_q(\delta) = \begin{cases} 0 \ \textit{if}\ r \leq \delta \leq 1 \\ \leq 1 - \frac{\delta}{r} \ \textit{if}\ 0 \leq \delta < r \end{cases}.$$

*Proof.* Suppose that $r \leq \delta \leq 1$. Then the Plotkin bound gives

$$A_q(n, \delta n) \leq \frac{\delta}{\delta - r}$$

independently of $n$. As $n \to \infty$, taking the logarithm of both sides and dividing by $n$ gives 0.

Now suppose $0 \leq \delta < r$. Let

$$n' = \left\lfloor \frac{\delta n - 1}{r} \right\rfloor,$$

so $n' < n$. The main idea is to shorten the code in the right way. Take an $(n, M, \delta n)$ code and choose any $n - n'$ positions. There are $q^{n-n'}$ possible $n - n'$ tuples in these spots, and by the pigeonhole principle one has to occur at least $\frac{M}{q^{n-n'}}$ times. Choose the tuple occurring the most and take all of the words with this tuple in these places. This gives $M'$ words, and then puncture the code in these $n - n'$ positions. The minimum distance of this punctured code does not decrease.

We have $rn' < \delta n$. Apply the Plotkin bound to get

$$\frac{M}{q^{n-n'}} \leq M' \leq \frac{\delta n}{\delta n - rn'} \leq \delta n,$$

since $\delta n - rn' \geq 1$ by the definition of $n'$. Therefore $A_q(n, \delta n) \leq q^{n-n'}\delta n$, which gives

$$\alpha_q(\delta) \leq \lim_{n\to\infty} \sup 1 - \frac{n'}{n} + \frac{\log_q(\delta)}{n} + \frac{\log_q(n)}{n} = 1 - \frac{\delta}{r}.$$

□

We will state two more asymptotic bounds without proof. The first is the asymptotic version of the Elias bound.

**Proposition 31** (Asymptotic Elias bound)**.** *Let $0 < \delta < r$. Then*

$$\alpha_q(\delta) \leq 1 - H_q(r - \sqrt{r(r-\delta)}).$$

The second proposition comes from applying the dual version of the linear programming bound with a carefully chosen function. Let $t \in [1, \frac{n}{2})$, $a \in \mathbb{R}$ and define

$$\alpha(x) = \frac{1}{a-x}\left(K_t(a)K_{t+1}(x) - K_{t+1}(a)K_t(x)\right)^2.$$

**Proposition 32.** *For $q = 2$, we have*

$$\alpha_2(\delta) \leq H_2\left(\frac{1}{2} - \sqrt{\delta(1-\delta)}\right).$$

Many of the best known asymptotic upper bounds we have come from applying the linear programming bound with carefully chosen parameters. We note that there are stronger bounds than this once, but the above bound is easier to state than most.

We have given lower and upper bounds for $\alpha_q(\delta)$ and it is currently unknown which end of this range is closer to the true values of this function. For a long time, it was thought that the Gilbert-Varshamov lower bound was the best possible, that there was no infinite family of codes of increasing length correcting a $\frac{\delta}{2}$ proportion of errors and rate exceeding this lower bound. Around 1982, there was a major breakthrough coming from Algebraic-Geometry codes. The Tsfasman-Vlăduţ-Zink theorem gives a lower bound which is better in the case where $q$ is a square and at least 49. The key is to show that there exist algebraic curves over $\mathbb{F}_q$ of relatively low genus with many rational points and to construct codes from them. Here is a version of that result.

**Theorem 29.** *For $q \geq 49$ a square prime power, we have*

$$\alpha_q(\delta) \geq 1 - \delta - \frac{1}{\sqrt{q} - 1}.$$

It is also interesting to graph the bounds given for different values of $q$ and see which is best in each case.

## 8 The Golay Codes

Thanks to Osbert Bastani for helping type this section.

In this section we will discuss the amazing Golay codes which have already appeared in several places throughout these notes.

## 8.1 Constructions of the Golay codes

Consider the binary code with parameters $(23, 2^{12}, 7)$. The sphere of radius 3 in $\mathbb{F}_2^{23}$ has

$$1 + \binom{23}{1} + \binom{23}{2} + \binom{23}{3} = 2^{11}$$

words. Note that $2^{11} \cdot 2^{12} = 2^{23} = |\mathbb{F}_2^{23}|$, so if such a code exists, then it is perfect. Indeed, such a code exists and is unique. It is called binary Golay code. By adding a parity check bit, we get a $(24, 2^{12}, 8)$ code which is known as the extended binary Golay code.

Also consider a code over $\mathbb{F}_3$ with parameters $(11, 3^6, 5)$. The sphere of radius 2 around a codeword has

$$1 + 2\binom{11}{1} + 2^2\binom{11}{2} = 243 = 3^5$$

words. We see that $3^5 \cdot 3^6 = 3^{11} = |\mathbb{F}_3^{11}|$, so if such a code exists, then it is perfect. This code also exists, is unique, and is known as the ternary Golay code. In the previous section we also showed that if such a code exists, it gives equality in the Griesmer bound. If we extend this code so that the sum of the coordinates of every word is zero, then we get a $(12, 3^6, 6)$ code which is known as the extended ternary Golay code. We have seen that a code with these parameters gives equality in the Griesmer bound as well.

We will sketch several constructions of the binary Golay code. The first is easy to remember, but it is not so easy to see how to derive some of the key properties of the code from this description.

Consider an icosahedron, a regular Platonic solid with 20 equilateral triangular faces, 30 edges and 12 vertices. Note that this satisfies Euler's formula, $F - E + V = 2$. We construct the adjacency matrix of the icosahedron. This is a $12 \times 12$ matrix with entries in $\mathbb{F}_2$. We label the 12 vertices. The $(i, j)$ entry is 1 if and only if there is an edge from $v_i$ to $v_j$. We take the complement $\bar{A}$ of $A$ which is defined by replacing every 1 of $A$ with a 0 and every 0 with a 1. Now consider the $12 \times 24$ matrix $G = [I_{12}\ \bar{A}]$.

**Proposition 33.** *The matrix $G$ is the generator matrix of a $[24, 12, 8]$ code. This is the extended binary Golay code. Puncturing this code in any coordinate gives the binary Golay code.*

We will not prove that this construction works here. We will return later to the fact that puncturing the extended binary Golay code gives the binary Golay code.

This construction suggests that the code has a large automorphism group, since the symmetry group of the icosahedron, not including reflections, is $A_5$ (the alternating group on 5 letters). There are some interesting connections between the symmetries of the icosahedron and some properties of the symmetric group on 6 letters, $S_6$. For more details see the article of John Baez, "Some thoughts on the number six" or Professor Elkies' course notes for Math 155.

We will give one direct way to construct the ternary Golay code which is not too difficult to remember, but it does not really indicate what is going on that makes this code special. The ternary Golay code can be defined as the code which has parity check matrix given by the $5 \times 11$ matrix

$$\begin{pmatrix} 1 & 1 & 1 & 2 & 2 & 0 & \\ 1 & 1 & 2 & 1 & 0 & 2 & \\ 1 & 2 & 1 & 0 & 1 & 2 & I_5 \\ 1 & 2 & 0 & 1 & 2 & 1 & \\ 1 & 0 & 2 & 2 & 1 & 1 & \end{pmatrix}.$$

We now give an interesting aspect of the history of the ternary Golay code. Suppose you want to place bets on a soccer match. A game can end in a Win, Loss or Draw, so there are three outcomes. Imagine there are 11 matches a week. This would make sense if a league had 22 teams, which is true in some cases, for example Serie B of Italian professional soccer. If you make one bet per match, we see that there are $3^{11}$ possible bets you could place. Now suppose you are in a gambling league where you can submit a sheet consisting of one bet on each match each week. The sheet is considered a winner if you make at least 9 correct bets. How many different sheets do you have to submit before you are guaranteed to have at least one that is a winner? It is completely clear that you can do this with $3^9$ sheets. Just pick two matches where you are convinced of the result. Then submit a sheet for each of the other $3^9$ possible matches. Now, even if your two matches that you were sure about go badly, you are guaranteed to have a sheet which covers the other 9 results exactly, and is therefore a winner. It is not hard to see that this is overkill.

It turns out that the ternary Golay code gives the best possible strategy for submitting these sheets. Let 0 correspond to a home team win, 1 correspond to a home team loss, and 2 correspond to a draw. Now give the 11 matches some arbitrary ordering. If you submit $3^6$ sheets, one corresponding to each word of the ternary Golay code, then no matter what the real outcome is there is guaranteed to be a codeword within distance 2 of this word, and the sheet corresponding to a codeword will be a winner.

It is somewhat amusing that important results in coding theory can be phrased in terms of gambling problems. It is not so hard to come up with other 'football-pool' type problems that have nice answers coming from codes. What is amazing about this particular example is that it was published by a Finnish gambling enthusiast named Virtakallio in 1947, two years before Golay published any academic work about the ternary code which is named for him. Over sixty years later I think we can be confident that the mathematicians have passed the gamblers in their knowledge of error-correcting codes, but the situation was not so clear in the 1940s.

Golay codes have many interesting properties. They are perfect, which means that they have the best possible error correcting capabilities given their length and number of codewords. They also have good decoding algorithms and have been used effectively in practice, for example by NASA in space transmission and in radio transmissions. They are also important because they arise in several major theorems of coding theory.

**Theorem 1** (Tietäväinen, van Lint)**.** *If $C$ is a perfect $e$-error correcting binary code with $e > 1$, then $C$ is a repetition code of odd length, or $C$ is a binary Golay code.*

We stated this theorem at the beginning of the course. If we had another few weeks of lectures, we would go through the proof in detail. Instead, here is a quick sketch. The largest part of the proof is Lloyd's theorem, a statement which says that if a binary perfect code of length $n$ exists, a certain Krawtchouk polynomial must have zeros of a certain type. Another key idea is that given a binary code, we construct a matrix from it in a particular way, and then study the eigenvalues of that matrix.

We now turn to Gleason's Theorem and self-dual codes. This is another major result that we would prove if we had more time.

We begin with a proposition that is not difficult.

**Proposition 34.** *The extended binary and ternary Golay codes are self-dual.*

We recall that if $C$ and $C^\perp$ have identical weight distributions then $C$ is formally self-dual. We note that there are formally self-dual codes which are not self-dual. We have already seen an example of such a $[6,3]$ code in our discussion of higher weight enumerators.

**Theorem 2** (Gleason)**.** *In each of the following statements suppose that the $a_i$ are rational numbers which sum to 1. We have*

1. *If $q = 2$, all weights of $C$ are even, and $C$ is formally self-dual, then*

$$W_C(X,Y) = \sum_{i=0}^{\lfloor \frac{n}{8} \rfloor} a_i (X^2+Y^2)^{\frac{n}{2}-4i}(X^8+14X^4Y^4+Y^8)^i.$$

2. *If $q = 2$, all the weights of $\mathcal{C}$ are divisible by 4, and $C$ is self-dual, then*

$$W_C(X,Y) = \sum_{i=0}^{\lfloor \frac{n}{24} \rfloor} a_i (X^8+14X^4Y^4+Y^8)^{\frac{n}{8}-3i}(X^{24}+759X^{16}Y^8+2576X^{12}Y^{12}+759X^8Y^{16}+Y^{24})^i.$$

3. *If $q = 3$ and $C$ is self-dual, then*

$$W_C(X,Y) = \sum_{i=0}^{\lfloor \frac{n}{12} \rfloor} a_i (X^4 + 7X^3Y)^{\frac{n}{4}-3i} (X^{12} + 264X^6Y^6 + 440X^3Y^9 + 24Y^{12})^i.$$

We note that $X^2 + Y^2$ is the weight enumerator of the binary repetition code of length 2 and $X^8 + 14X^4Y^4 + Y^8$ is the weight enumerator of the extended binary Hamming code of length 8. We will soon see that $X^{24} + 759X^{16}Y^8 + 2576X^{12}Y^{12} + 759X^8Y^{16} + Y^{24}$ is the weight enumerator of the extended binary Golay code, and that $X^{12} + 264X^6Y^6 + 440X^3Y^9 + 24Y^{12}$ is the weight enumerator of the extended ternary Golay code. We call a code satisfying the hypotheses of the first statement a Type I code, a code satisfying the hypotheses of the second statement is a Type II code, and a code satisfying the hypotheses of the third statement is a Type III code.

We will next give two other constructions of Golay codes which are particular instances of more general classes of codes. We begin with lexicodes.

**Definition.** Consider $\mathbb{F}_2^n$. Order all $n$-tuples lexicographically:

$$(00\ldots00), \quad (00\ldots01), \quad (0\ldots010), \quad (0\ldots011), \ \ldots\ , (11\ldots11).$$

Fix $d \geq 1$. We construct a set $\mathcal{L}$, known as the lexicode $(n,d)$ by adding one codeword at a time according to the following process:

1. Place $0 \in \mathcal{L}$.

2. Include the first nonzero vector of weight $d$.

3. Add the next element which has distance at least $d$ from every element already in $\mathcal{L}$.

4. Repeat the previous step until all $n$-tuples have been considered.

Alternatively we can construct a linear version of the lexicode by replacing the third step with 'Add the next element which is distance at least $d$ from all linear combinations of vectors already in $\mathcal{L}$'. We then take the nonzero words of $\mathcal{L}$ to be the rows of a generator matrix of our code.

We state a theorem which shows that we can deduce much information about a lexicode from $n$ and $d$.

**Theorem 3.** *Label the vectors in the lexicode in the order in which they are added, with $c_0$ is the zero vector and $c_1, \ldots, c_{|\mathcal{L}|-1}$ a listing of the other vectors. Then*

1. *$\mathcal{L}$ is a linear code, and $\{c_{2^i}\}$ gives a basis for it.*

2. *After $c_{2^i}$ is chosen, the next $2^i - 1$ vectors chosen are $c_1 + c_{2^i},\ c_2 + c_{2^i}, \ldots, c_{2^i-1} + c_{2^i}$.*

3. *Let $\mathcal{L}_i = \{c_0, c_1, \ldots, c_{2^i-1}\}$. Then $\mathcal{L}_i$ is an $[n,i,d]$ linear code.*

Asymptotically these codes give equality in the Gilbert-Varshamov bound. In the particular case where $n = 24,\ d = 12$ using the linear version of this construction we get a 12 element basis $w_1, ..., w_{12}$, which span the extended binary Golay code. We will not verify these claims here, but point out that lexicodes give another interesting example of codes that have been the subject of extensive study.

We next turn to the construction of Golay codes as a type of quadratic residue code. These are examples of BCH codes with particular defining sets and in fact are an instance of another class of codes, the duadic codes.

Let $n$ be an odd prime and let $q$ be a quadratic residue modulo $n$. This is equivalent to $q^{\frac{n-1}{2}} \equiv 1 \pmod{n}$. Let $R_0 = \{i^2 \pmod{n} :\ i \in \mathbb{F}_q^*\}$ be the set of nonzero quadratic resides modulo $n$ and $R_1 = \mathbb{F}_q^* \setminus R_0$, be the set of quadratic nonresidues. Let $\alpha$ be a primitive $n$th root of unity in some extension field of $\mathbb{F}_q$. That is, let $\alpha \in \mathbb{F}_{q^r}$ satisfy $\alpha^n = 1$ in $\mathbb{F}_{q^r}$ but $\alpha^m \neq 1$ for any $0 < m < n$. It is an exercise to note that this is possible given the conditions on $n$ and $q$ given above.

We define two key polynomials for this construction. Let

$$g_0(x) = \prod_{r \in R_0} x - \alpha^r, \quad \text{and} \quad g_1(x) = \prod_{r \in R_1} x - \alpha^r.$$

**Definition.** The cyclic code of length $n$ over $\mathbb{F}_q$ with generator polynomial $g_0(x)$ is called a quadratic residue code of length $n$. The code with generator polynomial $(x-1)g_0(x)$ is also often referred to as a quadratic residue code.

It is an exercise to show that when $q = 2$ the second quadratic residue code here is just the even weight subcode of the first.

We next see that several interesting examples of codes arise using this construction.

**Example.** Suppose $q = 2,\ n = 7$ and factor $x^7 - 1 = (x-1)(x^3+x+1)(x^3+x^2+1)$. It is an exercise to find an appropriate choice of $\alpha$ to define $g_0(x)$. We can check that in the above definitions $g_0(x) = 1 + x + x^3$. The cyclic code with this generator polynomial is the $[7, 4]$ Hamming code.

**Example.** Suppose $q = 2,\ n = 23$ and factor

$$x^{23} - 1 = (x-1)(x^{11} + x^9 + x^7 + x^6 + x^5 + x + 1)(x^{11} + x^{10} + x^6 + x^5 + x^4 + x^2 + 1).$$

It is an exercise to find an appropriate choice of $\alpha$ to define $g_0(x)$. With such an $\alpha$ we can see that $g_0(x)$ is the first degree 11 polynomial in this factorization.

Consider the cyclic code with this generator polynomial. We can apply the BCH code that we proved when first discussing cyclic codes and see that it shows that this code has minimum distance at least 5. In fact, it has minimum distance 7 which shows that it is a $[23, 12, 7]$ code and we will soon see that this implies that it is equivalent to the binary Golay code.

Letting $q = 3$ and $n = 11$ using a similar construction gives the ternary Golay code. In this case the BCH bound implies that the minimum distance is at least 4, when in fact it is equal to 5.

Quadratic residue codes have been studied extensively and remain somewhat mysterious. It is not unusual for them to have better minimum distance than is guaranteed by the BCH bound. They also tend to have large and interesting automorphism groups. Many open questions remain about their weight enumerators and also how to decode them effectively.

## 8.2 Uniqueness of the Golay Codes

Any code with the parameters $(2, 2^{12}, 7)$ is equivalent to the binary Golay code. This is what we mean when we say that the binary Golay code is unique. Similarly, the extended binary Golay code is unique. The two ternary Golay codes are also unique. In this section we will sketch a proof that the binary Golay codes are unique.

We begin with the weight enumerator of the binary Golay code.

**Proposition 35.** *Let $C$ be a $(23, 2^{12}, 7)$ binary code containing $0$. Then*
$A_0 = A_{23} = 1,\ A_7 = A_{16} = 253,\ A_8 = A_{15} = 506$*, and* $A_{11} = A_{12} = 1288$*. In other words,*

$$W_C(X, Y) = X^{23} + 253X^{16}Y^7 + 506X^8Y^{15} + 506X^{15}Y^8 + 253X^{16}Y^7 + Y^{23}.$$

*Proof.* We will give a sketch of this proof, outlining the major steps and completing the first few. It will be clear how to proceed from here.

A code with these parameters is a perfect 3-error correcting code, so spheres of radius 3 around codewords are disjoint and cover $\mathbb{F}_2^{23}$. Let $N(w, c)$ be the number of vectors of weight $w$ in a sphere of radius 3 around a codeword of weight $c$. It is clear that this does not depend on which codeword of weight $c$ is chosen. We also note that $N(w,c) = 0$ whenever $|w-c| > 3$. Every vector of weight $w$ must be in some sphere of radius 3 around some unique codeword $c$. Therefore,

$$\sum_{c=0}^{23} N(w,c) \cdot A_c = \binom{23}{w}.$$

The idea of the proof is to compute the $A_i$ one at a time. In order to compute $A_i$ we count the number of words of weight $i-3$ in spheres of radius 3 around each codeword. We use the above relation over and over. Note that $A_0 = 1,\ A_i = 0$ for $1 \leq i \leq 6$.

First we compute $A_7$. We note that any $x \in \mathbb{F}_2^{23}$ of weight 4 is in a unique sphere of radius 3 around a codeword of weight 7. It is also clear that a weight 4 codeword which is distance at most 3 from a weight 7 codeword must have its support contained within the support of the weight 7 codeword. Therefore, $N(4,7) = \binom{7}{4}$. Now using the above relation, $N(4,7)A_7 = \binom{23}{4}$, which implies $A_7 = 253$.

Next we compute $A_8$. Any $x \in \mathbb{F}_2^{23}$ of weight 5 must be contained in a sphere of radius 3 around some codeword of weight 7 or 8. This implies $N(5,7) = \binom{7}{5}$ and $N(5,8) = \binom{8}{5}$. We can see that the support of such a word must be contained within the support of this codeword. Applying the above proposition, $\binom{7}{5}A_7 + \binom{8}{5}A_8 = \binom{23}{5}$. This implies $A_8 = 506$.

In order to compute $A_9$ we have to consider one new difficulty. A word $x \in \mathbb{F}_2^{23}$ can be in a sphere of radius 3 around a codeword of weight 7 if only 5 of its six 1s are contained in the support of the codeword. This gives $N(6,7) = \binom{7}{6} + \binom{7}{5}\binom{16}{1} = 343$. We now continue as above which gives us $A_9$.

Continuing in this way, we can solve for each $A_i$, completing the proof. We could also note that each $A_i$ is determined by the $A_j$ for $i < j$ and the values of $N(w,c)$, which we can compute. We can then see that the weight enumerator for any $(23, 2^{12}, 7)$ code is the same, and compute it for one of them. □

We now see that adding an extra parity check bit gives the weight enumerator of the extended binary Golay code. This also shows that starting with the extended binary Golay code and puncturing in any of the 24 coordinates gives a code with the parameters of the binary Golay code.

A similar argument shows that the weight enumerator of the ternary Golay code is determined by its parameters. The key fact to use here is

$$\sum_{c=0}^{11} N(w,c)A_c = \binom{11}{w}2^w.$$

It is a little more difficult to show that the weight enumerator of the extended ternary Golay code is also determined by its parameters, but this is still true.

We next show that any code with the weight enumerator of the binary Golay code is linear.

**Proposition 36.** *Let $C$ be a $(23, 2^{12}, 7)$ code containing $0$, and $\widehat{C}$ the code we get from adding a parity check bit to every codeword. Then $C$ and $\widehat{C}$ are both linear.*

*Proof.* We see from the previous discussion of the weight enumerators of the binary Golay codes that every codeword of the extended binary Golay code has weight divisible by 4, that it is a doubly even code. We now want to show that distinct words of this code are orthogonal.

Let $c_1 \in C$ have extension $\widehat{c_1}$. We define a translated version of $C$, $C_1 := c_1 + C$. We see that this code contains 0 and is a $(23, 2^{12}, 7)$ code. Therefore, we know its weight enumerator.

We also consider $\widehat{C_1} := \widehat{c_1} + \widehat{C}$ also has the weight of each codeword divisible by 4. We now note that for $\widehat{c_1}, \widehat{c_2} \in \widehat{C}$, the inner product $\langle \widehat{c_1}, \widehat{c_2} \rangle = |\operatorname{Supp}(\widehat{c_1}) \cap \operatorname{Supp}(\widehat{c_2})|$. This implies that

$$\operatorname{wt}(\widehat{c_1}) + \operatorname{wt}(\widehat{c_2}) = \operatorname{wt}(\widehat{c_1} + \widehat{c_2}) + \langle \widehat{c_1}, \widehat{c_2} \rangle.$$

Since the lefthand side is divisible by 4 and $\operatorname{wt}(\widehat{c_1} + \widehat{c_2})$ is even, we see that $\langle \widehat{c_1}, \widehat{c_2} \rangle$ is also even. Therefore, $\widehat{c_1}$ and $\widehat{c_2}$ are orthogonal.

Now we have $2^{12}$ elements of $\mathbb{F}_2^{24}$ which are pairwise orthogonal. If we take the span of these $2^{12}$ words we see that the dual of this code has dimension at least 12. However the sum of the dimension of a code and the dimension of its dual is the length of a codeword, which implies that this code has dimension 12. Therefore, these $2^{12}$ words for a linear subspace of $\mathbb{F}_2^{24}$ and therefore $\widehat{C_1}$ is linear. This implies $\widehat{C} = \widehat{C_1}$. We also note that if we puncture a linear code, we get a linear code. This completes the proof.

□

In order to conclude that the binary Golay code is unique we will first prove that the extended binary Golay code is unique and then use a simple fact about its automorphism group.

The easiest way to prove the uniqueness of the extended binary Golay code is to use the uniqueness of a particular combinatorial design. It is not difficult to show that the 759 words of weight 8 hold a 5-design, in fact, a $5-(24,8,1)$ design or Steiner system $S(5,8,24)$. This means that any five coordinates are contained in the support of a unique codeword of weight 8. We now use the fact that this design is unique in order to prove that the extended binary Golay code is unique. We point out that this design is the object of intense study since very few Steiner systems with $t=5$ are known, and none are known with $t \geq 6$.

The extended binary Golay code has a transitive automorphism group. This can be seen from considering the automorphism group of the design coming from words of weight 8. Therefore when we puncture the code in one place it does not matter which place we choose, all of the resulting codes will be equivalent. This proves the uniqueness of the binary Golay code.

When we stated that the automorphism group of the extended binary Golay code is transitive this was a huge understatement. It is actually 5-transitive. That is, given any two 5-tuples of coordinate positions, there is an automorphism of the code which takes the first 5-tuple to the second. From this fact we can show that the automorphism group acts transitively on the 759 codewords of weight 8, which are sometimes known as octads.

**Theorem 30.** *The automorphism group of the extended binary Golay code is $M_{24}$, one of the Mathieu groups, a sporadic finite simple group. It has order $24 \cdot 23 \cdot 22 \cdot 21 \cdot 20 \cdot 16 \cdot 3$.*

*The automorphism group of the binary Golay code is $M_{23}$, another Mathieu finite simple group.*

The Mathieu groups play an important role in the classification of finite simple groups, one of the major mathematical achievements of the 20th century. The Golay codes are an interesting part of this story. They can be used to construct several other finite simple groups. For example, one can use the extended binary Golay code to construct the Leech lattice, a highly symmetric and dense lattice in $\mathbb{R}^{24}$. The automorphism group of this lattice is one of Conway's finite simple groups, from which we can construct the Monster group, the largest sporadic finite simple group. For more information on this story, see Conway and Sloane's Sphere Packing Lattices and Groups.

## 8.3 $A_2(13,5)$

In this section we will tie up a loose end from our discussion of bounds on codes. In that section of the notes we used $A_2(13,5)$ as a running example for the strength of various bounds. We saw that the sphere-packing bound gave an upper bound of 89 and that several bounds we saw failed to improve this result. The linear programming bound gave us $A_2(13,5) \leq 64$ and gave strict conditions on the distance distribution of such a $(13,64,5)$ code. In fact, such a code does exist and can be constructed from the extended binary Golay code.

Choose your favorite generator matrix for the extended binary Golay code. It is an exercise to show that there exists a permutation of coordinates such that there are 32 words with 0s in the first 8 positions, and for each $1 \leq i \leq 7$ there are 32 words with 1s in positions $i$ and 8 and 0s in each of the other first 8 positions. If you choose the right construction of this code, this is obviously true.

Take these $8 \cdot 32 = 256$ words and delete the first 8 coordinates. This gives a $(16,256,6)$ nonlinear code known as the Nordstrom-Robinson code. This is the first example in an infinite family of nonlinear codes with interesting properties. They arise as the image of a $\mathbb{Z}/4\mathbb{Z}$ linear code under the Gray map, which is a way of taking an element of $\mathbb{Z}/4\mathbb{Z}$ to a pair of elements of $\mathbb{F}_2$ in a way that maintains interesting properties of some codes. The Gray map and codes over $\mathbb{Z}/4\mathbb{Z}$ have been studied extensively.

We now take the Nordstrom-Robinson code and shorten the last two coordinates, choosing for example all of the codewords which end in two zeros, and then puncturing on these two coordinates. We now puncture one more time. This gives a $(13, 64, 5)$ code known as Y. It is also possible to show that this code is unique.

## 8.4 Extremal Type I and Type II Codes

Recall that a Type I code is a binary code with all even weights that is formally self-dual and that a Type II code has all weights divisible by 4 and is self-dual. Gleason's Theorem tells us about the weight enumerators of such codes, that they are linear combinations of certain products of weight enumerators of some special codes. We can use Gleason's Theorem to give a bound on the minimum distances of these codes.

**Theorem 31.**
- *Let $C$ be a Type I code. Then $d \leq 2\left\lfloor \frac{n}{8} \right\rfloor + 2$.*
- *Let $C$ be a Type II code. Then $d \leq 2\left\lfloor \frac{n}{24} \right\rfloor + 4$.*

We will not prove this here. We note that a code which gives equality in the second statement is known as an Extremal Type II code. We end this section with some open problems.

- Do there exist $[56, 28, 12]$, $[70, 35, 14]$ or $[72, 36, 14]$ Type I codes?
- Does there exist a $[72, 36, 16]$ Type II code?

# 9 Codes and Lattices

In this section we will discuss a topic that we did not get to in class, the connection between codes and lattices in $\mathbb{R}^n$. This will only be a brief introduction. For a more extensive treatment that connects to many of the topics we have already discussed, see Elkies' two-part article in the Notices of the AMS, "Lattices, Linear Codes and Invariants", or the book of Conway and Sloane, Sphere Packing, Lattices and Groups.

## 9.1 Some Basics of Lattices

**Definition.** A lattice $\Lambda$ is a free abelian group of rank $n$ given by

$$\left\{ \sum_{j=1}^{n} c_j v_j \quad \text{where } c_j \in \mathbb{Z}, \quad \{v_1, \ldots, v_n\} \text{ generate } \mathbb{R}^n \right\}.$$

The generator matrix $M$ of this lattice has columns $v_1, \ldots, v_n$. The Gram matrix is $A = MM^T$. We see that the $(i, j)$ entry of this matrix is $\langle v_i, v_j \rangle$.

We say that a lattice is integral if the inner product between any two vectors is an integer. We see that we need only check that the entries of the Gram matrix are integers.

The Gram matrix comes from a quadratic form. If $v = \sum_{i=1}^{n} m_i v_i$ where $m = (m_1, \ldots, m_n) \in \mathbb{Z}^n$, then

$$\langle v_i, v_j \rangle = \sum_{i=1}^{m} \sum_{j=1}^{n} A_{ij} m_i m_j.$$

The inner products are the values of a particular quadratic form evaluated at $m$. Much of the early work by Gauss and others on lattices was motivated by the study of quadratic forms.

**Definition.** The determinant of $\Lambda$ is $\det(\Lambda) = \det(A) = \det(M)^2$. This is also the volume of a fundamental domain for $\mathbb{R}^n/\Lambda$.

Let $N(\Lambda)$ denote the minimum value of $\langle v, v\rangle$ among all nonzero $v \in \Lambda$. We define the density of the lattice as

$$\Delta(\Lambda) = \frac{\text{Vol}\left(\text{Sphere of Radius } \frac{N(\Lambda)^{\frac{1}{2}}}{2}\right)}{\det(\Lambda)^{\frac{1}{2}}}.$$

This definition is motivated by the study of sphere packings. We would like to know how efficiently we can pack spheres of fixed radius into $n$-dimensional space. This question is very difficult. In fact, we only know the complete answer in dimensions $1, 2$ and $3$.

If we impose the extra requirement that the centers of our spheres must lie on the points of a lattice, then we know a lot more. If we want to find dense lattice packings then we want to maximize $\frac{N(\Lambda)^{\frac{n}{2}}}{\det(\Lambda)^{\frac{1}{2}}}$. The best lattice packings are known in dimensions $1 \le i \le 8$ and in dimension 24. This amazing 24 dimensional lattice packing, the Leech Lattice, is closely related to the binary Golay code!

**Definition.** Let $\Lambda$ be a lattice and define the dual lattice

$$\Lambda^* = \{y \in \mathbb{R}^n \ \mid \ x \cdot y \in \mathbb{Z} \ \text{ for all } x \in \Lambda\}.$$

**Exercise.** A lattice is integral if and only if it is contained in its dual.

**Definition.** If $\Lambda$ is integral, then we say that it is unimodular if and only if $\det(\Lambda) = \pm 1$.

**Exercise.** Check from the definitions that an integral lattice is unimodular if and only if it is equal to its dual.

The analogy between codes and lattices is particularly strong in the case of Type II codes and the following class of lattices which we also call Type II.

**Definition.** We say that a lattice $\Lambda$ is even if and only if $x \cdot x$ is an even integer for all $x \in \Lambda$. A lattice is called odd if it is integral but not even. We say that $\Lambda$ is a Type I lattice if it is odd and unimodular, and a Type II lattice if it is even and unimodular.

**Example.** We consider the hexagonal lattice in $\mathbb{R}^2$. This gives not only the best lattice packing, but the densest sphere packing of any kind in $\mathbb{R}^n$. A basis for this lattice is given by the vectors $(1, 0)$ and $(\frac{-1}{2}, \frac{\sqrt{3}}{2})$, so $M = \begin{pmatrix} 1 & \frac{-1}{2} \\ 0 & \frac{\sqrt{3}}{2} \end{pmatrix}$.

This generator matrix gives Gram matrix $\begin{pmatrix} 1 & \frac{-1}{2} \\ \frac{-1}{2} & 1 \end{pmatrix}$. We can scale all of the vectors in our lattice by $\sqrt{2}$, which gives a new Gram matrix $\begin{pmatrix} 2 & -1 \\ -1 & 2 \end{pmatrix}$, with $N(\Lambda) = 2, \ \det(\Lambda) = 3$ and $\Delta(\Lambda) = \frac{\pi}{4\sqrt{3}}$.

**Exercise.** Consider the face-centered-cubic lattice which is given by $M = \begin{pmatrix} 1 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 1 \end{pmatrix}$. Compute the Gram matrix, determinant and density of this lattice. This gives the sphere packing in three dimensions.

## 9.2 Theta Functions and Extremal Lattices

There is an analog of the weight enumerator of a code in the case of lattices. This is a function which keeps track of the norms of the vectors of the lattice.

**Definition.** Let $q = 2^{i\pi\tau}$ where $\tau$ is in the upper half plane. We define the theta function of a lattice $\Lambda$ by

$$\Theta_\Lambda(q) = \sum_{x\in\Lambda} q^{\langle x,x\rangle}.$$

If $\Lambda$ is integral then let $N_m$ be the number of lattice points of norm $m$, so

$$\Theta_\Lambda(q) = \sum_{m=0}^{\infty} N_m q^m.$$

We will now state but not prove some results that demonstrate the connection between Type II codes and lattices. In this case, the theta function obeys some interesting symmetry relations which are very useful in attempting to classify Type II lattices.

**Proposition 37.** *If $\Lambda$ is an even unimodular (Type II) lattice, then $\Theta_\Lambda(q)$ is a modular form of weight $\frac{n}{2}$.*

This result is the foundation which allows us to give an analog of Gleason's Theorem on weight enumerators known as Hecke's Theorem. We know quite a lot about the space of modular forms of given weight and we can use these results to say something about the first nonzero norm which must occur.

**Theorem 32.** *For a Type I lattice, $N(\Lambda) \le \lfloor \frac{n}{8} \rfloor + 1$.*
*For a Type II lattice, $N(\Lambda) \le 2\lfloor \frac{n}{24} \rfloor + 2$.*

We can also show that Type II lattices exist only in dimensions that are multiples of 8.

If equality holds in either of the above conditions, we say that $\Lambda$ is extremal. Just as much research is done on extremal Type I and Type II codes, extremal lattices have been the subject of intense study over the years. Last summer, the first extremal Type II lattice in 72-dimensions was found by Gabriele Nebe. While there is a strong connection between Type II lattices and Type II codes, nobody has yet been successful in constructing an extremal Type II code of length 72.

## 9.3 Lattices from Codes: Construction A

In this section we will describe a simple way to take a binary code of length $n$ and construct a lattice in $\mathbb{R}^n$ from it. This is known as Construction A and is due to Sloane. This construction has been generalized in several directions, for example to non-binary codes, but we will not go into this here.

**Definition.** Let $C$ be an $[n, k, d]$ binary code. Then $\Lambda(C)$ is the lattice consisting of all $x \in \mathbb{R}^n$ which are obtained from a codeword $c \in C$ viewed as integer vector in $\mathbb{Z}^n$ plus a vector of $\mathbb{Z}^n$ which has all even integer components, and then dividing each component of the resulting vector by $\sqrt{2}$. More succinctly,

$$\Lambda(C) = \left\{ x \in \mathbb{R}^n \mid \sqrt{2}x \pmod 2 \in C \right\}.$$

The following proposition is pretty straightforward, but we leave the proof as an exercise.

**Proposition 38.** *Let $C$ be a binary code. Then $\Lambda(C^\perp) = \Lambda(C)^*$. Also, $\Lambda(C)$ is Type I/II if and only if $C$ is Type I/II.*

**Example.** Let $C$ be the trivial code of length $n$. Applying this construction, we see that $\Lambda(C)$ consists of all vectors which are $\sqrt{2}$ times a vector in $\mathbb{Z}^n$. This gives a scaled copy of the $\mathbb{Z}^n$ lattice.

Let $C$ be the repetition code of length $2n$. Applying this construction we get a lattice called $D_{2n}$, the checkerboard lattice.

**Example.** Let $C$ be the $[8, 4, 4]$ extended Hamming code. If we apply Construction $A$ we get a lattice known as $E_8$. This is the unique extremal Type II lattice in dimension 8 and gives the best lattice packing in 8 dimensions.

We can also construct $E_8$ by starting with $D_8$. We can take a scaled copy of $D_8$ to be the set of vectors in $\mathbb{Z}^8$ which have even sum. We note that the minimum distance between two vectors of this lattice is 2. The vector $(\frac{1}{2}, \frac{1}{2}, \ldots, \frac{1}{2})$ has distance 2 to the closest vector of $D_8$. We can take another copy of $D_8$ shifted by this vector, and luckily the union of $D_8$ and this shifted copy is still a lattice! This is also $E_8$.

If we apply Construction A to the extended binary Golay code, and then take its union with a shifted copy of itself just as we did with $D_8$ above, we get the Leech lattice. This is the unique extremal Type II lattice in dimension 24 and gives the densest lattice packing of spheres in this dimension. This fact is a recent and difficult result of Cohn and Kumar, and relies on sophisticated Linear Programming techniques, not unlike the type of linear programming we saw in our study of bounds on codes. The Leech lattice has a gigantic automorphism group, one of Conway's sporadic simple groups, from which we can construct the largest sporadic simple group, the Monster Group.

This is all we will say about codes and lattices for now, but we have only seen the surface of a deep and fascinating subject.

## 10 Graph Based Codes

In this final section we will explore a different view of coding theory. This is a more applied view of the subject and is a key component of many modern industrial applications. For much more on the subject see Nigel Boston's survey article, "Graph Based Codes", MacKay's book, or Modern Coding Theory by Richardson and Urbanke. The presentation below is heavily taken from Boston's article. Unfortunately, in the printed version of the article in the collection edited by Winnie Li, there are many problems with the diagrams (including the fact that two of them are switched).

Boston begins his article with some warnings for mathematicians. In this course we have been guilty of some of the common problems that he says afflict mathematicians learning coding theory.

- Decoding algorithms matter. Often mathematicians underemphasize the decoding side of the story. Instead of thinking of it as a kind of inconvenience to be left to the engineers, it would be better to use it as a kind of springboard to new and interesting mathematics.
- In practice, minimum distance is not the one-and-only issue in coding theory. In practice, a few low-weight codewords may be tolerable and lead to higher probability of transmitting information correctly than a code with higher minimum distance but many more codewords close to that distance.
- Low Density Parity Check codes (LDPC) and their variants are one of the major areas of coding theory research right now. So far, we (and many mathematical coding theory courses) have focused on more classical code constructions, even though in many situations LDPC-type codes perform much better.

Shannon's Theorem shows the existence of codes with increasing length, rates arbitrarily close to channel capacity, and error probability using maximum likelihood decoding going to zero. It does not say how to find these good codes. Below is a timeline highlighting some progress towards this goal with a focus on methods involving graphs.

- 1960- Gallagher develops the theory of LDPC codes in his MIT PhD thesis. This achievement goes largely ignored for decades. Gallagher's codes require block lengths in the tens of thousands and so, are dismissed as much too long to be practical.
- 1981- LDPC codes are rediscovered by Tanner who introduced new concepts from graph theory into their study.
- 1993- Turbo codes with rates approaching the Shannon limit are introduced.
- 1999- MacKay gives new improvements for LDPC codes and also gives proofs of their effectiveness in many situations.
- 1997- Luby, Mitzenmacher, Shokrollahi, Spielman and Stemann reach the Shannon limit for the binary erasure channel.

Shannon's challenge has been answered for the binary erasure channel, which in several ways is easier to work with than the binary symmetric channel. New families of codes have also been developed that come close to the Shannon limit for the AWGN (Additive-Gaussian-White-Noise) channel.

In the rest of this section we will go through some of the ways in which codes are represented with graphs. Unfortunately, in class we were able to draw several useful diagrams, but we have not made digital versions yet.

## 10.1 Trellises

Consider the $[4,3,2]$ parity check code. We list the codewords as the rows of a matrix:

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

Suppose we send a codeword $x = (x_1, x_2, x_3, x_4)$ over a channel with bitwise error probability $p$. If we receive $y = (y_1, y_2, y_3, y_4) = (0,0,1,0)$. There are four codewords with distance 1 from $y$ and we note that three of them have $x_2 = 0$. Suppose we want to figure out the probability that $x_2$ was a 0. We expect this to be approximately $\frac{3}{4}$ by the argument given above.

By Bayes' Theorem

$$\mathbb{P}(x_2 = 0 \mid y) = \sum_{\substack{x \in C \\ x_2 = 0}} \frac{\mathbb{P}(y \mid x)\, \mathbb{P}(x)}{\mathbb{P}(y)}.$$

Suppose $p = .1$. Therefore, we compute

$$\sum_{\substack{x \in C \\ x_2 = 0}} \mathbb{P}(y \mid x)\, \mathbb{P}(x) = \frac{1}{8}(3p(1-p)^3 + p^3(1-p)) = .02745.$$

Similarly,

$$\sum_{\substack{x \in C \\ x_2 = 1}} \mathbb{P}(y \mid x)\, \mathbb{P}(x) = \frac{1}{8}(p(1-p)^3 + 3p^3(1-p)) = .00945.$$

In order to finish the calculation, we note that

$$\mathbb{P}(y) = \sum_{x \in C} \mathbb{P}(y \mid x) = .02745 + .00945 = .0369.$$

Now,

$$\mathbb{P}(x_2 = 0 \mid y) = \frac{.02745}{.0369} = .744, \quad \text{and, } \mathbb{P}(x_2 = 1 \mid y) = \frac{.00945}{.0369} = .256.$$

We can always carry out this type of computation to do bitwise decoding, but in general it is too much work. This requires $O(n2^k)$ operations, which is much to slow. In order to fix this problem we introduce the idea of representing a code with a graph known as a trellis.

**Definition.** A trellis is a directed graph in which every node is a well-defined depth from the initial node. Vertical slices are known as times, and the maximum width of the trellis is known as its state complexity.

We now introduce the BCJR or Forward-Backward Algorithm for bitwise decoding. In
Each edge is labeled with a 0 or 1, and as we go from the start to the finish in $n$ steps, this gives a binary word of length $n$. Varying over all possible paths gives the code. Every codeword appears exactly once as a path from the start node to the finish node.

To determine $\mathbb{P}(x_2 \mid y)$, we make a forward compuation and a backward compuation, multiplying node values and edge values. Every edge is labeled with a 0 or 1. The edge value for an edge which corresponds to $x_i$ being equal to 0 (for example) is $\mathbb{P}(x_i = 0)\ \mathbb{P}(y_i = 0 \mid x_i = 0)$. In this example $\mathbb{P}(x_i = 0)$ is $\frac{1}{2}$ for all $i$, and $\mathbb{P}(y_i = 0 \mid x_i)$ is either $p$ or $1 - p$ depending on whether $y_i = 0$ or not.

For the forward computation, we want to assign a probability to each node. We ask, given the edge values we have computed, what is the probability that we end up here beginning at the start. For the backward computation we do something similar, starting from the end node and going backwards along the edges.

We then combine information from our edge values, forward values and backward values to figure out our desired probability. For example, for each edge corresponding to $x_2 = 0$, we look at the forward value at the beginning of this edge, and the backward value at the end of this edge, and multiply these three quantities together. We compute that $\mathbb{P}(x_2 = 0 \mid y) = .744$, the same result we got with the more direct approach described above.

Using a trellis representation of our code significantly cuts down on our total number of multiplications. This is basically because we are saving multiplications by using the distributive law. There are a few other important issues related to trellises which we will not discuss further here.

- Computing the bitwise probability for each bit and combining this information does not always give a valid codeword. In the cases where it does give a codeword, it is not guaranteed to be the most likely codeword.
- It is not clear how to draw a trellis efficiently given a generator matrix for a code.
- Different generator matrices give different trellises with different levels of complexity. For example, the state complexity can vary.
- Permuting the coordinates of a code gives an equivalent code, but the resulting trellis can look very difficult. How do we find a good choice of permutation of coordinates?
- How do trellises arise in applications? Trellises play a large role in the theory of convolutional codes, which are a class of codes that are not memoryless. The next symbol transmitted depends on the last $D$ sent, for some $D > 0$.

## 10.2 Tanner Graphs

**Definition.** A Tanner graph is a bipartite graph where there is one node for each coordinate and one node for each parity check. The first set of $n$ nodes are known as message nodes and the second are the $r$ check nodes. We draw an edge from message node $i$ to check node $j$ if $x_i$ is involved in parity check $j$.

**Example.** Consider the $[7, 4, 2]$ binary code with parity checks $c_1, c_2, c_3$ given by

$$\begin{array}{ccccccc} x_1 & + & x_2 & + & x_2 & = & 0 \\ x_1 & + & x_4 & + & x_5 & = & 0 \\ x_1 & + & x_6 & + & x_7 & = & 0 \end{array}.$$

This gives the Tanner graph below where circles represent message nodes and squares represent check nodes.
Include a picture of this graph.

We next describe the belief propagation algorithm which uses this graph for iterative decoding of individual symbols. One key idea is that decoding here happens with probabilities. Initially, each message bit is given a probability which represents its current belief that the corresponding message bit was a 1.

In the next step of the algorithm, every message bit simultaneously sends this probability to its neighboring check nodes. Each check nodes takes in this information from each of its neighboring message nodes and combines it to give new updated probabilities for each of these message nodes and sends them back. Each message nodes takes in these updated estimates from each connected check node and combines them to form a new estimate.

This process repeats either for a prespecified number of iterations, or until the probability for each bit surpasses some prespecified threshold. Much of the effectiveness of this algorithm comes from the fact that it works in parallel. In many situations the probabilities converge to near certainty over time.

There is a refinement of the Tanner graph known as a factor graph, which also includes ideas from trellises for more effective decoding, but we will not discuss this further.

## 10.3 Low Density Parity Check Codes

We will give a construction of low density parity check codes from sparse $\{0,1\}$ matrices. We begin with a mathematical definition of sparse.

**Definition.** A sequence of $r_i \times n_i$ matrices is sparse if $r_i n_i \to \infty$ and there exists some $K > 0$ such that the number of nonzero elements of each matrix is bounded by $K \max\{r_i, n_i\}$.

The codes constructed by taking sparse matrices as parity check matrices are known as Low Density Parity Check codes. An $(s,t)$-regular LDPC code is a linear code with parity-check matrix $H$ that has $s$ 1s in each column and $t$ 1s in each row.

Note that the total number of 1s in this matrix is $rt = ns$. It is not necessarily true that the parity checks are independent (that $H$ has full rank), but this is enough to give a lower bound on the rate of such a code. We see that the rate is at least $\frac{n-r}{n} = 1 - \frac{s}{t}$.

Here is an early example of Gallagher of a $(3,6)$-regular LDPC. Suppose that $6 \mid n$. The parity check matrix we construct has $\frac{n}{2}$ rows and $n$ columns.

The first row is $1111110\ldots0$. The next row is $0000001111110\ldots0$, and so on, so the first $\frac{n}{6}$ rows have 6 consecutive rows and then zeros everywhere else. Each row is given by the row above it shifted by six places.

Now choose a random permutation to apply to the columns. We apply this permutation to the first $\frac{n}{6}$ rows, giving a new set of $\frac{n}{6}$ rows which we place under our first $\frac{n}{6}$ rows. Finally, for the last $\frac{n}{6}$ rows we choose a second random permutation and apply it to the first $\frac{n}{6}$ rows. We see that this gives a $(3,6)$-regular parity check matrix.

These $(s,t)$-regular LDPC codes constructed from random permutations perform surprisingly well using decoding algorithms like the belief propagation described above. There has also been much research about adapting these constructions. For example, it has been shown that in many situations slightly irregular parity check matrices are superior to the regular ones. Also, there has been much interest in developing non-random constructions of such codes. Some of these are related to expander graphs and to deep results in number theory.

We close these notes with a statement of the result of Luby, Mitzenmacher, Shokrollahi, Spielman, and Stemann mentioned above.

**Theorem 33.** *For the binary erasure channel there exists a family of LDPC codes whose rates approach the channel capacity and which can be decoded using belief propagation with arbitrarily small error probability.*