Introduction to Coding Theory[1]

Spring Semester 2010

[1]Compiled June 4, 2010

The following lecture notes have been initially prepared by Amin Shokrollahi for the course Modern Coding Theory (Spring 2006) ; part of the material has been typed by the students of this course, including Mahdi Cheraghchi, Shirin Saeedi (and ?). The notes have been slightly re-arranged for the purpose of this course by Bertrand Meyer.

Contents

Lecture 1
Information Theory

1.1. What This Class is About

Traditionally, coding theory has been about methods for reliable transmission of information through unreliable media. The transmission media are called *communication channels*. Infidelities of the communication channels leads to corruption of the transmitted data. The role of coding theory is to pre-process the data in such a way as to provide reliable recovery after corruption. This pre-processing is often called "encoding". The pre-processing has obviously to take into account the nature of infidelities of the communication channel: for example, if a channel has the property that it occasionally flips a bit, then one might want to employ a different coding technique, than in the case of a very unreliable channel which flips bits more often than not.

One of the immediate questions that comes to mind is that of fundamental limits on reliable communication when the channel is unreliable. This is answered by Shannon's coding theorems. Although this theorem governs pretty much all the digital communication today, its proof is somewhat unsatisfactory, in that it does not present a constructive way for coding data so as to achieve these limits. Moreover, the classical proof is far from any attempt to provide efficient algorithmic solutions for the coding, and the corresponding decoding tasks. In essence, this is what this course is about: design and analysis of codes and corresponding encoding/decoding algorithms that are extremely efficient in terms of (a) the way they can cope with channel infidelities (so called error-correction capability), and (b) and in terms of their encoding/decoding algorithms.

Coding theory very naturally borrows techniques from various mathematical disciplines, among them probability and statistics, algebra, and combinatorics. We will not be able to develop all the basics in this course, and we will assume that the audience is somewhat familiar with these notions.

Before we dive into the design and analysis of codes, we have to start with the basics. This lecture will provide some of the basic definitions and notions that will be used throughout this class.

1.2. Channels

1.2.1. Definition

Definition 1.1. A *(finite input) memoryless communication channel* $\mathcal{C}$ is a triple (Σ, I, p), where

1. Σ is a finite set
2. I is a measurable set with measure μ
3. $p\colon I \times \Sigma \to \mathbb{R}_{\geq 0}$ is a function such that for any $x \in \Sigma$, $\int_I p(y|x)\mathrm{d}\mu(y) = 1$.

Often $p(y|x)$ is interpreted as the "probability of receiving y given that x was sent." We will always deal with memoryless channels, so that we extend p to a function on $\Sigma^n \times I^n$ as $p((y_1, \ldots, y_n)|(x_1, \ldots, x_n)) := \prod_{i=1}^n p(y_i|x_i)$.

1.2.2. Symmetry

A *permutation* of a set S is a bijective map on S. Suppose that μ is a permutation on S such that $\mu^t = \mathrm{id}_S$ for some integer t. This means that if we apply μ t times to any element of S, we obtain that element back. The *orbit* of an element a in S under μ is the set $\{a, \mu(a), \ldots, \mu^{t-1}(a)\}$. The size of the orbit is at most t, but can be smaller.

A channel is called *input* (respectively *output*) *symmetric* if there exist permutations σ on Σ and τ on I, with $\sigma^q = \mathrm{id}_\Sigma$ and $\tau^q = \mathrm{id}_I$, where $q = |\Sigma|$, such that all orbits of σ (respectively all orbits of τ) have q elements and such that for all $x \in \Sigma$ and $y \in I$ we have $p(y|x) = p(\tau(y)|\sigma(x))$. A channel is called *symmetric* if it is input and output symmetric.

A channel is called *binary* if Σ has only two elements. Combining the notation introduced so far, we can talk of binary input symmetric or binary output symmetric, or binary symmetric channels. A *binary symmetric* channel is a binary channel that is symmetric. In all these cases, the permutations σ and τ giving rise to the definitions are involutions, i.e., $\sigma^2 = \mathrm{id}_\Sigma$ and $\tau^2 = \mathrm{id}_I$.

A channel is called *discrete* if the output alphabet I is a discrete set.

We define the *error probability* of an input symmetric binary channel as

$$\frac{1}{2}\int_I \min(p(\tau(y)|1), p(y|1))\mathrm{d}y,$$

where we assume that $1 \in \Sigma$.

Example 1.2.

1. Let $\Sigma = I = \{0,1\}$, and $p(0|0) = p(1|1) = 1$, $p(0|1) = p(1|0) = 0$. This channel is symmetric and it is called the trivial channel.

2. Let $\Sigma = \{0,1\}$ and $I = \{0,1,E\}$, $\varepsilon \in [0,1]$, and $p(0|0) = p(1|1) = 1-\varepsilon$, $p(0|E) = p(1|E) = \varepsilon$, and $p(0|1) = p(1|0) = 0$. This channel is called the *binary erasure channel* with probability ε, denote $\mathrm{BEC}(\varepsilon)$. It is input symmetric via the maps $\sigma(0) = 1, \sigma(1) = 0$, and $\tau(0) = 1, \tau(1) = 0$, and $\tau(E) = E$. Its error probability is equal to $\varepsilon/2$.

3. Let $\Sigma = I = \{0,1\}$, $\varepsilon \in [0, 1/2]$, and $p(0|0) = p(1|1) = 1-\varepsilon$, and $p(0|1) = p(1|0) = \varepsilon$. The channel is symmetric via $\sigma = \tau$, and $\sigma(0) = 1, \sigma(1) = 0$. This channel is called the *binary symmetric channel* with crossover probability ε, and is denoted by $\mathrm{BSC}(\epsilon)$. Its error probability is ε.

4. Let $\Sigma = \{-1,+1\}$, $I = \mathbb{R}$, $\sigma \in \mathbb{R}_{>0}$, and $p(y|a) = \frac{1}{2\sqrt{\pi\sigma}}\mathrm{e}^{-(x-a)^2/4\sigma}$ for $a \in \{-1,+1\}$. This channel is called the *binary input additive white Gaussian noise* channel with variance σ, and is denoted by $\mathrm{AWGN}(\sigma)$. The error probability of this channel is $\frac{1}{2\sqrt{\pi\sigma}}\int_{-\infty}^{0}\mathrm{e}^{-(x-1)^2/4\sigma}\mathrm{d}x$. Using the Q-function[1], the error probability can be expressed as $Q(1/\sqrt{2\sigma})$. This channel is symmetric via $\sigma(a) = -a$, $a \in \{-1,1\}$, and $\tau(y) = -y$, $y \in \mathbb{R}$.

5. Let $\Sigma = \{0,1,\ldots,q-1\} = I$, $\varepsilon \in [0, \frac{q-1}{q}]$, and $p(a|b) = \varepsilon/(q-1)$ if $b \neq a$, and $p(a|a) = 1-\varepsilon$. This channel is symmetric via the functions $\sigma = \tau$ with $\sigma(x) = (x+1) \bmod q$. This channel is called the q-ary symmetric channel with probability ε, and is denoted by q-$\mathrm{SC}(\varepsilon)$.

The first four communication channels given above are binary.

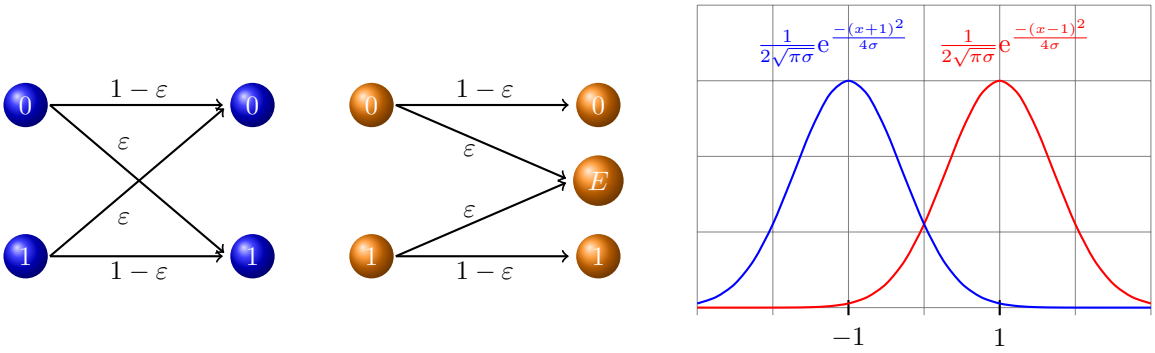
Suppose that $\mathcal{C}$ is a discrete memoryless channel. The "transition matrix" of the channel is the matrix $P(\mathcal{C}) = (p_{xy})_{x\in\Sigma, y\in I}$, where $p_{xy} = p(y|x)$. For example,

$$P(\mathrm{BSC}(\varepsilon)) = \begin{pmatrix} 1-\varepsilon & \varepsilon \\ \varepsilon & 1-\varepsilon \end{pmatrix}, \qquad P(\mathrm{BEC}(\varepsilon)) = \begin{pmatrix} 1-\varepsilon & \varepsilon & 0 \\ 0 & \varepsilon & 1-\varepsilon \end{pmatrix}.$$

The following remark is a little mental exercise.

Remark 1.3. *If the discrete channel $\mathcal{C}$ is input symmetric then all rows of $P(\mathcal{C})$ are permutations of the first row, and if it is output symmetric then all columns of $P(\mathcal{C})$ are permutations of the first column.*

[1] The Q-function is the tail probability of the standard normal distribution : $Q(x) = \frac{1}{\sqrt{2\pi}}\int_x^\infty \exp\left(\frac{t^2}{2}\right)\mathrm{d}t$.



(a) (b) (c)

Figure 1.1: (a) The binary symmetric channel, (b) The binary erasure channel, (c) The binary input AWGN channel

1.2.3. Operational Meaning of a Channel

Operationally, a communication channel models an ideal transmission medium, over which we wish to transmit the elements of the alphabet Σ. At every channel use, we transmit one of the elements of $x \in \Sigma$, and we receive an element $y \in I$. The transmitted element induces a probability distribution on I. For any $y \in I$ the value $p(y|x)$ describes the probability of receiving y if x is sent. The task is to estimate x from the received value y. If the channel has a nonzero average error probability π, then on average (over the choice of x) any estimate for x from y will have an error probability of at least π. To obtain error-free communication, we proceed as follows: Instead of sending x, we send a vector $(x_1, \ldots, x_n) \in \Sigma^n$ using the channel n times consecutively. If the x_i are chosen independently, then we still have the nonzero probability of estimation. However, if there is a dependency among the x_i, then the error probability of the estimation may be made arbitrarily small.

1.3. Codes and Maximum-Likelihood Decoding

1.3.1. Codes and rate

Definition 1.4. A *code* C over the alphabet Σ is a subset of Σ^n.

The (binary) *rate* of the code is the quantity $\log_2(|C|)/n$, and the *symbol rate* of the code is the quantity $\log_{|\Sigma|}(|C|)/n$. The binary rate measures the average number of bits sent over the communication channel when elements of C are used for n consecutive channel uses. The symbol rate measures the average number of alphabet symbols sent over the channel in n consecutive channel uses.

1.3.2. Maximum-Likelihood Decoding

For a code $C \subseteq \Sigma^n$, we define the *Maximum-Likelihood (ML)-decoding* as follows: for a received word $y \in I^n$, the codeword $x \in C$ maximizing the probability $p(y|x)$ is called the ML-decoding of y.

Given two elements $x, y \in \Sigma^n$, we define the *Hamming distance* $d(x, y)$ by the number of coordinates on which x and y differ.

Theorem 1.5. *If* $\mathcal{C} = \mathrm{BSC}(\varepsilon)$, $\varepsilon < 1/2$, *then ML-decoding is equivalent to finding a codeword that has the smallest Hamming distance to the received word.*

Proof. Suppose that y is received. Then, for all $z \in C$ we have

$$p(y|z) = \prod_{i=1}^{n} p(y_i|z_i) = (1-\varepsilon)^n \left(\frac{\varepsilon}{1-\varepsilon}\right)^{d(y,z)},$$

where $d(y, z)$ is the Hamming distance between y and z. Since $\varepsilon < 1/2$, the quantity $\varepsilon/(1-\varepsilon) < 1$, and the quantity $p(y|z)$ is largest when $d(y, z)$ is smallest. □

1.4. Channel Capacity

Given a communication channel, what is the maximum rate at which we can send symbols through the channel, and expect the error probability of the ML-decoder to go to zero?

To that end, we shall need to quantifiy the uncertainty on the source X of input symbols and introduce a notion of entropy $H(X)$. We shall also introduce a notion of relative entropy, $H(X|Y)$, where X is a source of random messages and Y is the output of the channel. Thus $H(X|Y)$ measures the uncertainty on the sent message X given that we have received Y. This is a measure of the uncertainty introduced by the channel, a measure of its noise.

We define the capacity $\mathrm{Cap}(\mathcal{C})$ of the channel by $\max_X H(X) - H(X|Y)$ (the difference for any source between the quantity of information sent by X and the uncertainty introduced by the use of the channel). Shannon proves that transmission is possible at any rate $R < \mathrm{Cap}(\mathcal{C})$ with an error probability of the ML-decoder that approaches zero as the length of the underlying code goes to infinty.

1.4.1. Entropy

As for a rigorous definition, let X be a random variable on Σ, and Y be a random variable on I induced by the channel law p. This means that $\Pr[Y = y|X = x] = p(x|y)$, and hence $\Pr[Y = y] = \sum_{x\in\Sigma} \Pr[Y = y \mid X = x] \Pr[X = x]$.

Definition 1.6. The *entropy* of the random variable X, denoted $H(X)$, is given by

$$H(X) = -\sum_{x\in X} \Pr[X = x] \log_2(\Pr[X = x]),$$

and the *conditional entropy* of X given Y is

$$H(X|Y) = -\sum_{x\in X, y\in Y} \Pr[X = x, Y = y] \log_2(\Pr[X = x|Y = y]).$$

Notice that the entropy is also the expectancy $E[-\log p(X)]$.

Example 1.7. Suppose X is a Bernoulli variable, *i.e.* $X = 1$ with probability p and $X = 0$ with probability $1 - p$. Then

$$H(X) = H(p) = -p \log p - (1 - p) \log(1 - p)$$

The graph of $H(p)$ (see Figure 1.2) shows that that entropy (the uncertainty) is maximal when $p = \frac{1}{2}$ and vanishes when $p = 0$ or $p = 1$.

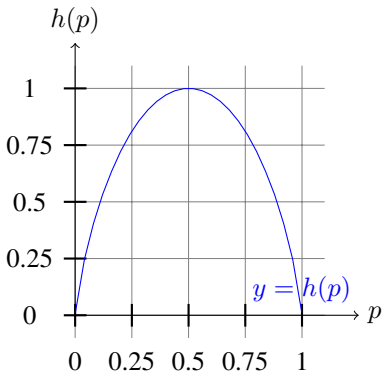


Figure 1.2: Binary entropy.

Definition 1.8. The *mutual information* $I(X;Y)$ is defined as $H(X) - H(X|Y)$.

Since $H(X) \geq H(X|Y)$, the mutual information is nonnegative. In addition, the mutual information is zero iff $H(X) = H(X|Y)$, and the latter holds iff X and Y are independent. Moreover, a simple manipulation using Bayes rule reveals that $I(X;Y)$ is symmetric in its arguments.

1.4.2. Channel capacity

Definition 1.9. The capacity $\mathrm{Cap}(\mathcal{C})$ of the channel $\mathcal{C}$ is defined as $\max_{p(X)} I(X;Y)$, where the maximum is taken over all probability distributions on Σ.

If $\mathcal{C}$ is an input symmetric channel, then it can be shown that the maximum is achieved for the uniform distribution on the input alphabet. In this case, we will have two interesting sets of probability distributions on I associated with $\mathcal{C}$, which can be visualized using the transition matrix $P(\mathcal{C})$: each row of the transition matrix is a probability distribution on I, and since the channel is input symmetric, the rows are permutations of one another, by Remark 1.3. In particular, these probability distributions have the same entropy. Denoting by Π the probability distribution on I induced by the first row of $P(\mathcal{C})$, the common entropy of all the rows of the transition matrix is $H(\Pi)$.

The second distribution is obtained by calculating the marginal distribution on the output symbols, given that we have the uniform distribution on the input. In this case, we take the average of all the rows of the transition matrix, which gives us another distribution on I, and we call this distribution Δ.

Theorem 1.10. *If $\mathcal{C}$ is a discrete memoryless input symmetric channel, then the capacity of $\mathcal{C}$ is $H(\Delta) - H(\Pi)$, where Π and Δ are defined above.*

Proof. Let X be the uniform distribution on Σ and Y be the distribution on I induced by X, i.e., for every $y \in I$, we have $\Pr[Y = y] = \frac{1}{|\Sigma|}\sum_{x\in\Sigma} p(y|x)$. Then, Δ is exactly this probability distribution, and $H(\Delta) = H(Y)$.

How about $H(Y|X)$? Using the uniformity of the distribution of X, we obtain

$$\begin{aligned} H(Y|X) &= -\sum_{x\in\Sigma}\sum_{y\in I}\frac{1}{|\Sigma|}p(y|x)\log_2 p(y|x) \\ &= \frac{1}{|\Sigma|}\sum_{x\in\Sigma} H(\Pi) \\ &= H(\Pi), \end{aligned}$$

since for every x, the probability distribution on I given by $p(y|x)$ has the same entropy $H(\Pi)$. □

Example 1.11.

1. $\mathrm{Cap}(\mathrm{BEC}(\varepsilon)) = 1 - \varepsilon$. To see this, we appeal to the transition matrix of $\mathrm{BEC}(\varepsilon)$. In this case, the distribution Π is given as the vector $(1-\varepsilon, \varepsilon, 0)$, where the first entry is the probability of picking the element 0, the second is the probability of E, and the third is the probability of 1. The entropy of this distribution is $h(\varepsilon)$, where $h(x)$ is the binary entropy function. The average of the rows of the transition matrix is $((1-\varepsilon)/2, \varepsilon, (1-\varepsilon)/2)$, which has an entropy equal to $-(1-\varepsilon)\log_2((1-\varepsilon)/2) - \varepsilon\log_2(\varepsilon)$ which in turn is equal to $(1-\varepsilon) + h(\varepsilon)$. In total, the capacity is $H(\Delta) - H(\Pi) = 1 - \varepsilon + h(\varepsilon) - h(\varepsilon) = 1 - \varepsilon$.
2. $\mathrm{Cap}(\mathrm{BSC}(\varepsilon)) = 1 - h(\varepsilon)$, where $h(x) = -x\log_2(x) - (1-x)\log_2(1-x)$ is the binary entropy function. In this case, Π is given by $(1-\varepsilon, \varepsilon)$, and Δ is given by $(1/2, 1/2)$. The entropy of Δ is 1, and that of Π is $h(\varepsilon)$.
3. $\mathrm{Cap}(\mathrm{AWGN}(\sigma)) = 1 - \frac{1}{2\sqrt{\pi m}}\int_{-\infty}^{\infty}\log_2(1+\mathrm{e}^{-x})\mathrm{e}^{-\frac{(x-m)^2}{4m}}\mathrm{d}x$, where $m = 2/\sigma^2$. (Note that this is not a consequence of the previous theorem, since this channel is not discrete.)

Example 1.12. We compare the capacity of $\mathrm{AWGN}(\sigma)$ with $\mathrm{BSC}(\varepsilon)$, where ε is the error probability of $\mathrm{AWGN}(\sigma)$. The result is given in Figure 1.3. As can be seen, the capacity of $\mathrm{AWGN}(\sigma)$ is bigger. We often phrase this as saying that "using soft information is better than hard-decision decoding."

1.4.3. Shannon's Channel Coding Theorem

Shannon's Channel Coding Theorem shows that the capacity is an achievable upper bound for reliable communication. Its proof requires the following lemma and the Chernoff bounds on the tail distribution of sum of independent random variables.

Lemma 1.13 (Volume of the Hamming ball)**.** *Let n be an integer, and $e \le n/2$. Then*

$$\sum_{i=0}^{e}\binom{n}{i} = 2^{nh(e/n)+o(n)},$$

where $h(x) = -x\log_2 x - (1-x)\log_2(1-x)$ is the binary entropy function.

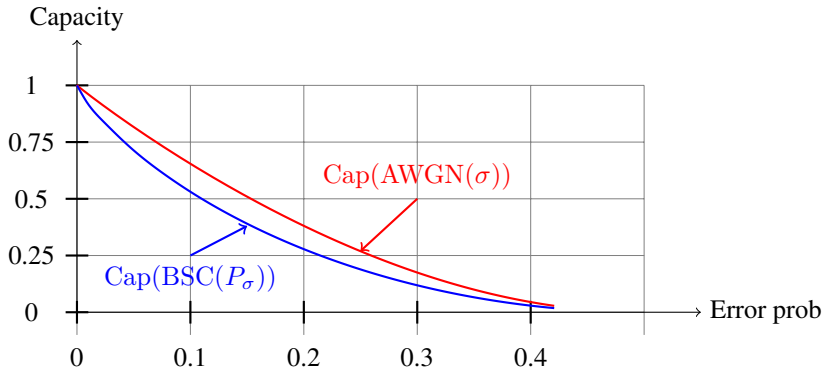


Figure 1.3: Comparison of the capacities of $\mathrm{AWGN}(\sigma)$ and a $\mathrm{BSC}(\varepsilon)$ with equal error probability.

Proof. Notice that : $2^{nh(e/n)} = \left(\frac{e}{n}\right)^{-e}\left(\frac{n-e}{n}\right)^{-(n-e)}$. We start with an upper bound of the sum. We have

$$\begin{aligned} 1 &= \left(\frac{e}{n}+\frac{n-e}{e}\right)^n \\ &= \sum_{i=0}^{n}\binom{n}{i}\left(\frac{e}{n}\right)^i\left(\frac{n-e}{n}\right)^{n-i} \\ &\geq \sum_{i=0}^{e}\binom{n}{i}\left(\frac{e}{n}\right)^i\left(\frac{n-e}{n}\right)^{n-i} \end{aligned}$$

Since, $e \leq n/2$, $\frac{e/n}{(n-e)/n} \leq 1$ and so

$$1 \geq \sum_{i=0}^{e}\binom{n}{i}\left(\frac{e}{n}\right)^e\left(\frac{n-e}{n}\right)^{n-e}$$

This provides directly $\sum_{i=0}^{e}\binom{n}{e} \leq 2^{nh(e/n)}$.

On the other hand, recall that $n! \sim \sqrt{2\pi n}\left(\frac{n}{e}\right)^n$ (Stirling's formula). We set $\alpha = \frac{e}{n}$. We have thus

$$\binom{n}{e} = \frac{n!}{e!(n-e)!} \sim \frac{1}{\sqrt{2\pi\alpha(1-\alpha)n}\alpha^e(1-\alpha)^{n-e}} \geq 2^{nh(\alpha)+o(n)}$$

But $\sum_{i=0}^{e}\binom{n}{i} \geq \binom{n}{e} \geq 2^{nh(\alpha)+o(n)}$, which achieves the proof. □

Proposition 1.14 (Chernoff bounds)**.** *Let $(Z_i)_{1\leq i\leq n}$ be i.i.d random variables in $\{0,1\}$ of expectation p, then there is a constant $\gamma > 0$ such that*

$$\Pr\left(\frac{1}{n}\sum_{i=1}^{n}Z_i - p > \epsilon\right) \leq e^{-\gamma np\epsilon^2} \quad \textit{and} \quad \Pr\left(\frac{1}{n}\sum_{i=1}^{n}Z_i - p < -\epsilon\right) \leq e^{-\gamma np\epsilon^2}.$$

Theorem 1.15 (Channel Coding Theorem for the BSC)**.** *Given a discrete memoryless communication channel $\mathcal{C}$ with input alphabet Σ, then for any $R < \mathrm{Cap}(\mathcal{C})$ there exists a series of codes $C_m \subset \Sigma^{n_m}$ of rate R such that the error probability of the ML-decoder for C_m is upper bounded by $\mathrm{e}^{-\gamma n_m R}$ for some $\gamma > 0$ (depending on the channel) called the error exponent.*

Proof. We will sketch the proof of this theorem for the case $\mathcal{C} = \mathrm{BSC}(p)$ with $p < 1/2$. Given a small $\epsilon > 0$, we set $\rho = p + \epsilon$ and $R < 1 - h(\rho)$. Note that $R < \mathrm{Cap}(\mathcal{C})$.

We start to select a code C by picking uniformly at random $M = 2^{\lfloor nR \rfloor}$ codewords $x_1, x_2, \ldots, x_M$ in $\{0,1\}^n$. Note that on average, decoding one of the x_i should give an error on np bits. According to Chernoff's bounds, when we send an x_i, the probability of an error on more than $n\rho$ bits is bounded by $e^{-\gamma np\epsilon^2}$. We modify our ML-decoder and decode now in the following way. When y is received and there is exactly one codeword x_i such that $d(y, x_i) \leq \rho$, we decode y as x. Otherwise, we declare an error.

Let P_i be the probability of making an incorrect decision when decoding the transmission of x_i. We have

$$P_i \leq \sum_{y\in\{0,1\}^n} p(y|x_i)\left(\mathbb{1}_{d(y,x_i)>n\rho} + \sum_{j\neq i}\mathbb{1}_{d(y,x_j)\leq n\rho}\right).$$

We know from the Chernoff bound that

$$\sum_{y\in\{0,1\}^n} p(y|x_i)\mathbb{1}_{d(y,x_i)>n\rho} \leq e^{-\gamma np\epsilon^2}.$$

Let P_C be the average error probability on the code C : $P_C = \frac{1}{M}\sum_{i=1}^{M} P_i$. We have

$$P_C \leq e^{-\gamma np\epsilon^2} + \frac{1}{M}\sum_{i=1}^{M}\sum_{y\in\{0,1\}^n}\sum_{i\neq j} p(y|x_i)\mathbb{1}_{d(y,x_j)\leq n\rho}.$$

and on average over the different codes :

$$E[P_C] \leq e^{-\gamma np\epsilon^2} + \frac{1}{M}E\left[\sum_y\sum_{i\neq j} p(y,x_i)\mathbb{1}_{d(y,x_j)\leq n\rho}\right]$$

Now remember that C is chosen at random and that the x_i are also independant and uniformly distributed. So for a fixed y,

$$E_{x_i}\left[p(y,x_i)\right] = \sum_{x\in\{0,1\}^n} p^{d(x,y)}(1-p)^{n-d(x,y)} = 1.$$

On the other hand, using Lemma 1.13,

$$E_{x_j}\left[\mathbb{1}_{d(y,x_j)\leq n\rho}\right] = \frac{1}{2^n}\sum_{x\in\{0,1\}^n}\mathbb{1}_{d(y,x_j)\leq n\rho} = \frac{1}{2^n}\sum_{i=1}^{\rho n}\binom{n}{i} = 2^{-n(1-h(\rho))+o(n)}.$$

So using independance, we have $\frac{1}{M}E\left[\sum_{i\neq j} p(y,x_i)\mathbb{1}_{d(y,x_j)\leq n\rho}\right] = (M-1)E\left[p(y,x_1)\right]E\left[\mathbb{1}_{d(y,x_1)\leq n\rho}\right]$ and we obtain

$$E[P_C] \leq e^{-\gamma np\epsilon^2} + (M-1)2^{-n(1-h(\rho))+o(n)}$$

.

Let $P^*(M,n,p)$ be the the minimum of P_C over all the codes C, then

$$P^*(M,n,p) \leq e^{-\gamma np\epsilon^2} + 2^{-n(1-h(\rho)+R)+o(n)}$$

A simple manipulation yields the final result. □

We remark that random codes of rate R approach the Shannon capacity with high probability. We also remark — without proof — the following converse to the Shannon coding theorem.

Remark 1.16. *For any code of rate $R > \mathrm{Cap}(\mathcal{C})$ the error probability of the ML-decoding is bounded from below by a constant. More precisely, for any sequence of codes of length n and rate $R > \mathrm{Cap}(\mathcal{C})$ there is a constant $\gamma > 0$ such that the error probability of the ML-decoder is at least $1 - \mathrm{e}^{-\gamma n}$.*

The fundamental problems associated with Shannon's coding theorem are:

1. Find codes that approach the Shannon capacity using ML-decoding.

2. Find codes that approach the Shannon capacity using a polynomial time decoding algorithm.

As mentioned above, random codes give an affirmative solution to the first problem (though they are not explicit; more on explicit codes later). A much more difficult problem is finding codes that approach the Shannon capacity arbitrarily closely, but have polynomial time decoding complexity. In a way, this class is about solving the second problem.

In the following, we will give a solution to the second problem for the binary erasure channel.

1.5. Codes Achieving the Capacity of the BEC with Polynomial Decoding Complexity

Throughout this section, $\mathcal{C} = \mathrm{BEC}(\varepsilon)$. The capacity of this channel is $1-\varepsilon$. Let R be smaller than $1-\varepsilon$. We will show the existence of codes of rate R with polynomial time (in the length n of the codewords) decoding complexity over $\mathrm{BEC}(\varepsilon)$ such that the error probability of the decoder is at most $\mathrm{e}^{-\gamma n}$ for some $\gamma > 0$ (depending on R).

Let C be a random subspace of $\mathbb{F}_2^n$ obtained in the following way: choose Rn vectors $x_1, \ldots, x_{Rn}$ of $\mathbb{F}_2^n$ uniformly at random. The subspace C is then $\langle x_1, \ldots, x_{Rn} \rangle$.

Exercise 1.1. Show that for any given δ the dimension of C is larger than $(R-\delta)n$ with probability at least $1-\mathrm{e}^{-c_\delta n}$ for some $c_\delta > 0$.

Let c be an element in C, and let y be the vector obtained from c after transmission through the channel $\mathcal{C}$.

Exercise 1.2. Show that for any $\delta > 0$ the vector y coincides with c on at least $(1-\epsilon-\delta)n$ positions, with probability at least $1-\mathrm{e}^{-\gamma n}$ for some $\gamma > 0$ (Chernoff bounds).

The task of the decoder is to recover c from y. This results to solving a system of linear equations. Let G denote the $Rn \times n$-matrix in which the rows are the vectors $x_1, \ldots, x_{Rn}$. The vector c equals $z \cdot G$ for some $z \in \mathbb{F}_2^n$, and the task is to recover z. Let $i_1, \ldots, i_m$ denote the non-erased positions of c, i.e., those positions in which $y_{i_j} \neq E$. Let G_1 denote the matrix obtained from G by considering only columns $i_1, \ldots, i_m$. Then we have $z \cdot G_1 = \hat{y}$, where $\hat{y}$ is the vector in which the coordinates of y are not E. This is a system of linear equations for z, which is uniquely solvable if G_1 is of rank Rn. Note that G_1 is a random binary matrix, i.e., the entries of G_1 are chosen independently and uniformly from $\{0,1\}$. The error probability of the decoder is then upper bounded by the probability that G_1 is of rank smaller than $(1-\varepsilon)Rn$.

Theorem 1.17. *Let G be a random matrix over $\mathbb{F}_2$ with k rows and n columns, $k < n$. The probability that the rank of G is smaller than k is at most 2^{k-n}.*

Proof. This probability is at most equal to the probability that there exists a nonzero vector z such that $z \cdot G = 0$. For every nonzero z, the probability that $z \cdot G = 0$ is $1/2^n$. Therefore, the expected number of nonzero z such that $z \cdot G = 0$ is $(2^k - 1)/2^n \leq 2^{k-n}$. By the union bound, this is an upper bound on the probability in question. □

References for this chapter :

- Claude E. Shannon, A mathematical source of communication.
- Thomas M. Cover, Joy A. Thomas, Elements of information theory.
- Pierre Brémaud, Introduction aux probabilités.

Lecture 2
Linear Codes

2.1. Linear Codes

2.1.1. Basic definitions

From now on we want to identify the alphabet Σ with a finite field $\mathbb{F}_q$. For general codes, introduced in the last section, the description is hard. For a code of rate R, we need to write down 2^{Rn} codewords, each of length n. Linear codes allow for an exponentially smaller description size.

Definition 2.1. A *linear code* C of *dimension* k is a set of vectors of $\mathbb{F}_q^n$ which forms a k-dimensional vector space over $\mathbb{F}_q$. The parameter n is called the block-length of C. A linear code of length n and dimension k over $\mathbb{F}_q$ is called an $[n,k]_q$-code. The ratio $\frac{k}{n}$ gives the *rate* of the code.

How can we guarantee that decoding on $\mathrm{BSC}(\varepsilon)$ is possible without (or with very little) errors? On the BSC the codeword that has the smallest Hamming distance to the received word is the Maximum-likelihood estimate of the transmitted word. If any two elements of C are "far apart" and if ε is small enough, then with very high probability there is a unique codeword that is closest to the received word, and this one is the maximum likelihood estimate.

We call the minimum Hamming distance between any two different codewords the *minimum distance* of the code. If the code is linear, this is the minimum distance of a nonzero codeword from the zero word, or the minimum Hamming weight of a nonzero codeword. Denoting the Hamming distance of x and y by $d(x,y)$, and the Hamming weight of x by $\mathrm{wgt}(x)$, we see that

$$\min_{x,y\in C, x\neq y} d(x,y) = \min_{0\neq x\in C} \mathrm{wgt}(x),$$

if C is linear.

An $[n,k]_q$-code with minimum distance d is called an $[n,k,d]_q$-code.

Example 2.2.
1. Any n dimensional linear space $\mathbb{F}_q^n$ is a $[n,n,1]_q$ code.
2. The parity code $P = \{(x_1,\ldots,x_n)\,;\, \sum_{i=1}^n x_i = 0\}$ is a $[n,n-1,2]_q$-code.

Lemma 2.3. *Let C be an $[n,k,d]_q$-code, and let $e = \lfloor (d-1)/2 \rfloor$. Given $c \in C$, let $B_e(c) = \{x \in \mathbb{F}_q^n, d(x,c) \leq e\}$ denote the Hamming ball of radius e. Then, for any $c, c' \in C$ with $c \neq c'$,*

$$B_e(c) \cap B_e(c') = \emptyset.$$

This lemma is usually stated as follows: if c is transmitted and $y \in B_e(c)$ is received, then ML-decoding will have zero error probability.

Proof. Given $c \neq c'$ in C and $x \in B_e(c) \cap B_e(c')$, we have $d \leq d(c,c') \leq d(c,x) + d(c',x)$ by the triangle inequality. So $d \leq 2e$ which is a contradiction. □

Definition 2.4. For a code C, the homogeneous polynomial $A_C(x,y) = \sum_{i=0}^n A_i x^i y^{n-i}$ is called the *weight enumerator* of C, where A_i is the number of codewords of weight i.

2.1.2. Generator and check matrices

Definition 2.5. Any matrix $G \in \mathbb{F}_q^{k\times n}$ whose rows form a basis of C is called a *generator matrix* for C. Any matrix $H \in \mathbb{F}_q^{(n-k)\times n}$ with independent rows and $\mathrm{rker}(H) = C$ is called a *check matrix* for C.

$$C = \{xG \in \mathbb{F}_q^n ;\ x \in \mathbb{F}_q^k\} = \{y \in \mathbb{F}_q^n ;\ Hy^\top = 0\}$$

A generator matrix for a code is called *systematic* if it is of the form $(I_k|G_1)$, where I_k is the $k \times k$-identity matrix.

Lemma 2.6. *Let C be an $[n,k]_q$-code with check matrix H. If any set of $d-1$ columns of H are linearly independent over $\mathbb{F}_q$, then the minimum distance of C is at least d.*

Proof. Let the columns of H be $H = [h_1|h_2\ldots|h_n]$ with $h_i \in \mathbb{F}_q^{n-k}$. Suppose without loss of generality that $h_1, \ldots, h_{d-1}$ are dependant and take $a_1, \ldots, a_{d-1}$ in $\mathbb{F}_q$ such that $\sum_{i=1}^{d-1} a_i h_i = 0$. Then

$$[h_1|h_2\ldots|h_n]\begin{pmatrix} a_1 \\ \vdots \\ a_{d-1} \\ 0 \\ \vdots \end{pmatrix} = 0$$

which means that $c = (a_1, \ldots, a_{d-1}, 0, \ldots, 0)$ is a codeword. But $\mathrm{wgt}(c) = d-1$. □

2.1.3. Duality

Definition 2.7. The *inner product* of vectors x and y in $\mathbb{F}_q^n$ is defined as $\langle x, y\rangle := \sum_{i=1}^n x_i y_i$. For a code C, the *dual code* $C^\perp$ of C is defined as

$$C^\perp = \{x \in \mathbb{F}_q^n ;\ \forall c \in C,\ \langle x, c\rangle = 0\}.$$

Proposition 2.8. *Let C is a linear code with generator matrix G and check matrix H, then $\left(C^\perp\right)^\perp = C$ and the dual code $C^\perp$ admits H as generator matrix and G as check matrix.*

The weight enumerator of C and the weight enumerator of $C^\perp$ are intimately related.

Theorem 2.9 (MacWilliams identities)**.** *Let C be an $[n,k]_q$-code. Then we have*

$$A_{C^\perp}(x,y) = \frac{1}{q^k} A_C(y-x, y+(q-1)x).$$

Proof. A function $\chi\colon \mathbb{F}_q \to \mathbb{C}$ is called an *additive character* if $\chi(a+b) = \chi(a)\chi(b)$ and $\chi(0) = 1$. It is called *trivial* if $\chi(x) = 1$ for all $x \in \mathbb{F}_q$. For a nontrivial additive character, we have

$$\sum_{\alpha\in\mathbb{F}_q} \chi(\alpha) = 0. \tag{2.1}$$

To see this, suppose that $\beta \in \mathbb{F}_q$ is such that $\chi(\beta) \neq 1$. Then the sum in question equals $\sum_{\alpha\in\mathbb{F}_q} \chi(\alpha+\beta)$ which in turn equals $\chi(\beta)$ times the sum in question. This implies $(1-\chi(\beta))\sum_{\alpha\in\mathbb{F}_q}\chi(\alpha) = 0$, from which we obtain $\sum_\alpha \chi(\alpha) = 0$.

For $x \in \mathbb{F}_q^n$ let

$$g(x) := \sum_{y\in\mathbb{F}_q^n} \chi(\langle x, y\rangle) z^{\mathrm{wgt}(y)}.$$

Then we have

$$\sum_{x\in C} g(x) = \sum_{y\in\mathbb{F}_q^n} z^{\mathrm{wgt}(y)} \sum_{x\in C} \chi(\langle x, y\rangle).$$

If $y \in C^\perp$, then $\chi(\langle x, y\rangle) = 1$. If $y \notin C^\perp$, then $\langle x, y\rangle$ takes every value in $\mathbb{F}_q$ the same number of times, say T, and hence $\sum_{x\in C}\chi(\langle x, y\rangle) = T\sum_{\alpha\in\mathbb{F}_q}\chi(\alpha) = 0$, by (2.1). Hence,

$$\sum_{x\in C} g(x) = |C| A_{C^\perp}(z, 1). \tag{2.2}$$

For $\alpha \in \mathbb{F}_q$ let $\mathrm{wgt}(\alpha) = 1$ if $\alpha \neq 0$, and $\mathrm{wgt}(0) = 0$, so that $\mathrm{wgt}(x_1, \ldots, x_n) = \sum_i \mathrm{wgt}(x_i)$. Then

$$\begin{aligned} g(x) &= \sum_{y_1,\ldots,y_n \in \mathbb{F}_q} z^{\mathrm{wgt}(y_1)+\cdots+\mathrm{wgt}(y_n)} \chi(\sum_i x_i y_i) \\ &= \sum_{y_1,\ldots,y_n \in \mathbb{F}_q} z^{\mathrm{wgt}(y_1)+\cdots+\mathrm{wgt}(y_n)} \prod_{i=1}^{n} \chi(x_i y_i) \\ &= \prod_{i=1}^{n} \sum_{\alpha \in \mathbb{F}_q} z^{\mathrm{wgt}(\alpha)} \chi(x_i \alpha). \end{aligned}$$

If $x_i = 0$, then the last expression is equal to $1 + (q-1)z$. If $x_i \neq 0$, then $\sum_\alpha z^{\mathrm{wgt}(\alpha)} \chi(x_i\alpha) = 1 + z \sum_{\alpha \neq 0} \chi(\alpha) = 1 - z$. Therefore,

$$g(x) = (1-z)^{\mathrm{wgt}(x)} (1 + (q-1)z)^{n - \mathrm{wgt}(x)},$$

and so $\sum_{x \in C} g(x) = A_C(1-z, 1+(q-1)z)$. Putting this and (2.2) together, we see that

$$A_C(1-z, 1+(q-1)z) = |C| A_{C^\perp}(z, 1).$$

Replacing z with x/y yields the result. □

Example 2.10. Consider the $[n, n-1, 2]_2$-parity code C. We have

$$A_C(x, y) = \sum_{i=0}^{\lfloor n/2 \rfloor} \binom{n}{2i} x^{2i} y^{n-2i}$$

The dual $C^\perp$ of C is one dimensional and equal to $\langle (1, 1, \ldots, 1) \rangle$, hence has the weight distribution $B(x, y) = y^n + x^n$. By the MacWilliams identities, we have

$$A_C(x, y) = \frac{1}{2} \left((y-x)^n + (y+x)^n \right),$$

which is true.

2.2. Relationship Between Distance Distribution and the Error Probability of the ML-decoder

The weight distribution of a linear code is a convenient means for upper bounding the error probability of the ML-decoder of a linear code on a binary symmetric channel.

For a discrete memoryless binary communication channel $\mathcal{C}$ with output alphabet I the quantity

$$\gamma(\mathcal{C}) = \sum_{y \in I} \sqrt{p(y,0) p(y,1)}$$

is called the *Bhattacharya parameter* of the channel. The goal of this section is to provide an upper bound on the word error probability of the ML-decoder using the weight distribution of a linear code.

Let the code C have weight distribution $\sum_i A_i x^i y^{n-i}$. If a vector $x \in C$ is transmitted over $\mathcal{C}$, let $P(x, z)$ denote the probability that the ML-decoder decodes the received word to a vector z of distance d from x. Because of the symmetry of the channel, this probability is the same for all pairs of distance d, and it will be denoted by P_d. By the union bound, the error probability P_e of the ML-decoder is upper bounded by

$$P_e \leq \sum_{d=1}^{n} A_d P_d.$$

Our task is thus to upper bound P_d.

Theorem 2.11. *We have $P_d \leq \gamma(\mathcal{C})^d$, hence $P_e \leq A(\gamma(\mathcal{C}), 1) - 1$.*

Proof. Let $D = \{x_1, \ldots, x_M\}$ be the set of codewords of distance d from z, and let $I_j := \{y \mid p(y, x_j) \geq p(y, z)\}$, i.e., the set of all received words that cause the ML-decoder to decode to x_j rather than z. Suppose that z was transmitted through the channel. Then

$$P_d = \sum_{y \in I_j} p(y, z) \leq \sum_{y \in I_j} \sqrt{p(y, z)p(y, x_j)},$$

where the inequality follows from $\sqrt{p(y, x_j)/p(y, z)} \geq 1$ by assumption. We thus have

$$\begin{aligned} P_d &\leq \sum_{y \in I_j} \sqrt{p(y, z)p(y, x_j)} \\ &\leq \sum_{y \in I^n} \sqrt{p(y, z)p(y, x_j)} \\ &= \sum_{y \in I^n} \prod_{i=1}^{n} \sqrt{p(y_i, z_i)p(y_i, x_{ji})} \\ &\leq \prod_{i=1}^{n} \sum_{y \in I} \sqrt{p(y, z_i)p(y, x_{ji})} \\ &= \gamma(\mathcal{C})^d \end{aligned}$$

For the last step, note that the inner sum is 1 if $z_i = x_{ji}$, and $\gamma(\mathcal{C})$ otherwise. □

This theorem means that the error probability of the ML-decoder is mostly influenced by the lower A_i's, i.e., by the number of pairs of words that have a small Hamming distance. This is one of the reasons why researchers often study codes for which any pair of vectors have a large Hamming distance (often also called codes with large minimum distance). The next lecture will make some of these concepts more precise.

2.3. First Examples of Codes

2.3.1. The Hamming Code

Consider the matrix H with r rows and $2^r - 1$ columns in which the columns are all the nonzero binary vectors of length r. We first show that

Lemma 2.12. *The matrix H described above has rank r.*

Proof. Note that the $r \times r$-identity matrix is a sub-matrix of H. □

The code which has H as its check matrix is called the *binary Hamming code*. It is a $[2^r - 1, 2^r - r - 1]_2$-code. Since all the columns of H are distinct, any two columns are independent. Hence, the minimum distance of C is at least 3. Moreover, there are 3 dependent columns (take any three binary vectors x_1, x_2, x_3 that sum up to zero), hence the minimum distance is exactly 3.

Example 2.13. The following is a systematic check matrix for the $[7, 4, 3]_2$-Hamming code:

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

Since Hamming codes have distance 3, they are one-error correcting. The following algorithm calculates the error position, provided that the columns are H are the binary expansions of the integers $1, 2, \ldots, 2^k - 1$:

1. If y denotes the received word, calculate $z = Hy^\top$ (This vector is often called the "syndrome" of the received word.)
2. Interpret z as the binary expansion of the integer i. Then the error is at position i.

It is easily seen that the algorithm is correct: if $y = c + e$ for a codeword c and an error vector e of weight one, then $H \cdot y^\top = H \cdot e^\top = h_j$, where j is the position of the nonzero entry of e, and h_j is the jth column of H. But according to the construction of H, the column h_j is the binary expansion of the integer j.

2.3.2. Hadamard Codes

Consider the vector space $\mathbb{F}_2^k$, and its dual space $(\mathbb{F}_2^k)^*$. This is the space of linear forms on $\mathbb{F}_2^k$, and is an k-dimensional vector space over $\mathbb{F}_2$. Define the following vector space morphism:

$$\begin{aligned} \varphi\colon (\mathbb{F}_2^k)^* &\to \mathbb{F}_2^{2^k-1} \\ \lambda &\mapsto (\lambda(x) \mid x \in \mathbb{F}_2^k \setminus \{0\}). \end{aligned}$$

The image of φ is called a Hadamard code.

Theorem 2.14. (1) *φ is injective.*

(2) *The minimum distance of* $\mathrm{Im}\varphi$ *is* 2^{k-1}.

Proof. (1) Obviously, only the zero linear form vanishes on all the elements of $\mathbb{F}_2^k$.

(2) Let λ be a nonzero linear form. Then the dimension of $\ker\lambda$ is $n-1$, and hence λ vanishes on exactly 2^{k-1} elements of $\mathbb{F}_2^k$, i.e., it does not vanish on exactly 2^{k-1} elements of this vector space. This shows that the weight of any nonzero element in the image of φ is 2^{k-1}. □

We will now show the following.

Theorem 2.15. (1) *The Hadamard code of length 2^k-1 and dimension k is the dual of the Hamming code of length 2^k-1 and co-dimension k.*

(2) *The weight distribution of the binary Hamming code of length 2^k-1 is*

$$\frac{1}{2^k}(y+x)^{2^k-1} + \frac{2^k-1}{2^k}(y^2-x^2)^{2^{k-1}-1}(y-x).$$

Proof. (1) First, note that for any nonzero point $x \in \mathbb{F}_2^n$ there is a linear form λ on $\mathbb{F}_2^n$ such that $\lambda(x) \neq 0$. Let $\lambda_1, \ldots, \lambda_n$ form a basis of the dual space $(\mathbb{F}_2^n)^*$. It follows that for any two $x, y \in \mathbb{F}_2^n \setminus \{0\}$, $x \neq y$, there is an i such that $\lambda_i(x) \neq \lambda_i(y)$. Therefore, in the matrix H in which the ith row is the vector $(\lambda_i(x) \mid x \in \mathbb{F}_2^n \setminus \{0\})$, all the columns are distinct. Moreover, H has 2^k-1 columns, so H is the check matrix of a binary Hamming code. On the other hand, H is clearly a generator matrix for the first order Reed-Muller code. Hence the duality result.

(2) The weight enumerator for the Hadamard code is $A_C(x,y) = y^{N-1} + (N-1)x^{N/2}y^{N/2-1}$, where $N = 2^k$. It follows by the MacWilliams identities that the weight enumerator for the Hamming code is

$$\frac{1}{N}(y+x)^{N-1} + \frac{N-1}{N}(y^2-x^2)^{N/2-1}(y-x).$$

□

2.3.3. ML-Decoding of the Hadmard Code

The normal algorithm for ML-decoding of Hadamard codes on $\mathrm{BSC}(\varepsilon)$ would do the following: upon reception of a word y, go over all the 2^k codewords, and check which one is closest to y. This operation uses $O(2^{2k})$ time: for each check, we need to compare the 2^k-1 coordinate positions. Here we introduce a faster algorithm, which uses only $O(k2^k)$ operations.

In the received vector y replace each 0 by $+1$ and each 1 by -1. Call the resulting vector $\hat{y}$. Note that $\hat{y}_i = (-1)^{y_i}$. Let H be the Hadamard matrix of size 2^k. The rows and columns of this matrix are indexed by elements $(u,v) \in \mathbb{F}_2^k \times \mathbb{F}_2^k$, and the entry (u,v) of the matrix is $(-1)^{\langle u,v\rangle}$. Let z denote the vector $H \cdot \hat{y}^\top$ (we view $\hat{y}$ as a row vector). This vector is called the *Hadamard-Walsh* transform of $\hat{y}$. The entry of z corresponding to $u \in \mathbb{F}_2^k$ is

$$\sum_{v\in\mathbb{F}_2^k} (-1)^{\langle u,v\rangle}(-1)^{y_v} = |\{v \mid \langle u,v\rangle = y_v\}| - |\{v \mid \langle u,v\rangle \neq y_v\}.$$

The vector $(\langle u,v\rangle \mid v \in \mathbb{F}_2^k)$ is a codeword of the Hadamard code. Call it $E(u)$. Then, the above says that

$$(H \cdot \hat{y})_u = N - 2d(y, E(u))$$

Hence, if we can calculate $H \cdot \hat{y}$ efficiently, then we can find vector closest to y by finding the maximum entry of the Hadamard-Walsh transform of $\hat{y}$. It is well-known that the computation of the Hadamard-Walsh transform of y can be accomplished with $O(n\log(n)) = O(k2^k)$ operations. The pre- and postprocessing steps are clearly $O(2^k)$. Hence the results follows.

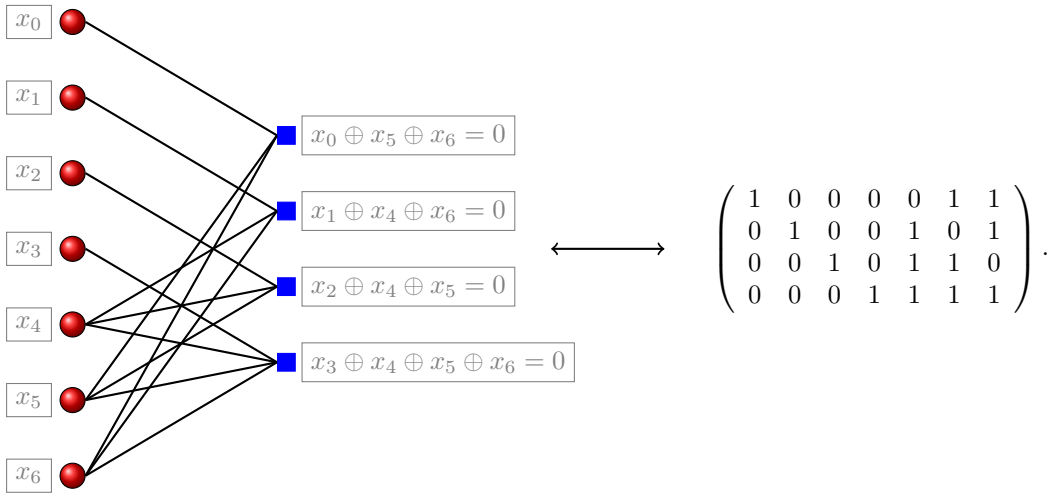


Figure 2.1: Graph representation of the $[7, 3, 4]_2$-Hadamard code

2.3.4. First Order Reed-Muller Code

Consider the space $V := \mathbb{F}_q[X_1, \ldots, X_m]_{\leq 1}$ of m-variate polynomials of degree at most 1 over $\mathbb{F}_q$. Let $N = q^m$, and map this space into $\mathbb{F}_q^N$ via

$$\begin{aligned} \varphi \colon V &\to \mathbb{F}_q^N \\ f &\mapsto (f(z_1), \ldots, f(z_N)), \end{aligned}$$

where $z_1, \ldots, z_N$ constitute the elements of $\mathbb{F}_q^m$ in some ordering. This is clearly a linear mapping, and hence its image is a linear code of length N. As for the dimension, note that the kernel of the map is trivial, hence the dimension is the dimension of V, i.e., it is $m + 1$. This image is called the first order Reed-Muller code over $\mathbb{F}_q$. Can you see how, in the case of $q = 2$, this case is related to the Hadamard code?

2.4. Graph Representation of Binary Codes

Let H be a check matrix for a binary code. We can interpret H as a bipartite graph with n left nodes (n being the number of columns of H) and r right nodes (r being the number of rows of H), in the following way: if the element at position (i, j) of H is one, then we put an edge between the ith right and the jth left node.

In this representation the left nodes are called the "variable nodes" while the right nodes are called the "check nodes". Given such a graph, the code is defined as the set of all binary settings on the variable nodes such that for all check nodes the sum of the settings of the adjacent variable nodes is zero.

If this graph is sparse (i.e., if we have a sequence of such graphs for which the number of variable and check nodes goes to infinity, the ratio between the variable and check nodes is fixed, and and the average variable node degree is constant), then these codes are called low-density parity-check (LDPC) codes. We will deal with these codes later in the course.

Figure 2.1 shows a graph representation of the Hadamard code represented by the check matrix

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

Sometimes, certain properties of the code can be inferred from graph properties of the graph representation. Here is a very simple example.

Lemma 2.16. *Suppose that the variable nodes of a graph corresponding to a binary code C have weight at least 2, and that the graph does not have cycles of length 4. Then the minimum distance of the code is at least 3.*

Proof. The minimum distance is not one: otherwise, there is a variable node that is not connected to any check node, which is a contradiction to the fact that the degree of the variable nodes is larger than one.

Suppose that the minimum distance is 2, and w.l.o.g. assume that minimum weight word is $(1, 1, 0, \ldots, 0)$. Consider the subgraph induced by the two first variable nodes. All check nodes in this graph must have even degree (or else they would not be satisfied). Moreover, there are at least two check nodes in this graph of degree greater than zero, since the degrees of the variable nodes is supposed to be ≥ 2. Then the graph formed by the two first variable nodes, and these two check nodes, is a cycle of length 4, contrary to the assumption. □

2.5. Creating New Codes from Old Ones

Let C be an $[n, k, d]_q$-code. In this section we describe methods to obtain $[n', k', d']_q$-codes from this code for values $n' \leq n$, $k' \leq k$, and $d' \leq d$.

Puncturing: The punctured code at position i, denoted C^i is the set of words $(x_1, x_2, \ldots, x_{i-1}, x_{i+1}, \ldots, x_n)$. The length of this code is $n - 1$, its dimension is $k - 1$ or k, and its minimum distance is $d - 1$ or d. Note that the dimension of this code is $k - 1$ iff there is a codeword of weight 1 with the nonzero entry at position i. Moreover, the minimum distance of the punctured code is $d - 1$ if and only if there is a codeword of weight d which has a nonzero entry at position i.

Shortening: In this construction the length and the dimension are decreased by 1, and the minimum distance will at least remain at its old value. Let C' be the intersection of C with the hyperplane of words that are zero at position i. The shortened version C_i of C at position i is the puncturing of C' at position i. Since C' is a sub-code of C, the minimum distance of C_i is at least d. Moreover, if C is such that there is a codeword with nonzero entry at position i, then the dimension of C_i is $k - 1$.

As a useful exercise, we propose that the reader prove the following:

Lemma 2.17. *The Hadamard code is obtained as a shortening of the binary first order RM-code.*

There is a simple connection between punctured and shortened versions of codes, due to P. Delsarte.

Proposition 2.18. *Let C be a linear code. Then*

$$\left(C^{\perp}\right)_i = \left(C^i\right)^{\perp}.$$

Proof. By inspection. □

2.6. Projecting Codes over Large Alphabets to Codes over Small Alphabets

In this section we assume that we have an $[n, k, d]_q$-code and that we want to produce from this code another coder over $\mathbb{F}_p$ where p is the characteristic of $\mathbb{F}_q$. For simplicity we will assume that $p = 2$.

The first method we discuss is the *binary image method*. Fix a vector space basis $\alpha_1, \ldots, \alpha_m$ of $\mathbb{F}_q/\mathbb{F}_2$ (so we assume that $q = 2^m$).

Definition 2.19. The *binary image* of C, denoted $\mathrm{Im}_{\mathbb{F}_2}(C)$ is obtained by replacing each entry in a codeword by its representation in the above basis.

The following is then obvious.

Proposition 2.20. *$\mathrm{Im}_{\mathbb{F}_2}(C)$ is an $[mn, mk, d']_2$-code where $d' \geq d$.*

The second method is the *subfield-subcode method*.

Definition 2.21. The *binary subfield-subcode* of C is the code $C|_{\mathbb{F}_2} := C \cap \mathbb{F}_2^n$, i.e., we consider in C vectors that happen to have binary entries.

In this method the length of the code is not altered, and obviously, the minimum distance cannot be smaller than that of the original code, simply because the subfield-subcode is in particular a subcode of C. As far as the dimension goes, we have the following result.

Proposition 2.22. *The dimension of $C|_{\mathbb{F}_2}$ is at least $mk - (m - 1)n$.*

Proof. Suppose that $(x_1, \ldots, x_n)$ is an element of the subfield-subcode, and consider a check equation $\sum_i \mu_i x_i = 0$, wherein $\mu_i \in \mathbb{F}_q$. Then, for all automorphisms σ of $\mathbb{F}_q/\mathbb{F}_2$ we have $\sum_i \sigma(\mu_i) x_i = 0$. Moreover, $(x_1, \ldots, x_n)$ is in the subfield-subcode iff it satisfies all the automorphic images of all the checks. Since there are $r = n - k$ independent checks on C, there are at most mr independent checks on $C|_{\mathbb{F}_2}$, and hence the dimension of the subfield-subcode is at least $n - mr = mk - (m - 1)n$. □

The bound given in the proposition is sharp. Consider the $[m, m-1]_{\mathbb{F}_q}$-code with check matrix $(\mu_1, \mu_2, \ldots, \mu_m)$, and suppose that the elements in this row vector are independent over $\mathbb{F}_2$. A check matrix for $C|_{\mathbb{F}_2}$ is then given by the $m \times m$-matrix

$$\begin{pmatrix} \mu_1 & \mu_2 & \cdots & \mu_{m-1} & \mu_m \\ \mu_1^2 & \mu_2^2 & \cdots & \mu_{m-1}^2 & \mu_m^2 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ \mu_1^{2^{m-1}} & \mu_2^{2^{m-1}} & \cdots & \mu_{m-1}^{2^{m-1}} & \mu_m^{2^{m-1}} \end{pmatrix}.$$

The right kernel of this matrix does not contain any nontrivial binary vector, because of the independence assumption. Hence, the $\mathbb{F}_2$-rank of this matrix is full, and the dimension of $C|_{\mathbb{F}_2}$ is 0. On the other hand, $mk - (m-1)n = m(m-1) - (m-1)m = 0$, which shows the sharpness of the bound.

A much easier way for constructing the subfield-subcode is as follows. Fix a basis $\mu_1, \ldots, \mu_m$ of $\mathbb{F}_q/\mathbb{F}_2$. In the check matrix H of C, replace any entry by its representation in this basis as an m-dimensional column vector. Denote the resulting matrix by $\hat{H}$.

Proposition 2.23. *$\hat{H}$ is a check matrix for $C|_{\mathbb{F}_2}$.*

Proof. Let h_{ij} denote the (i,j)-entry of H, and $(x_1, \ldots, x_n) \in \mathbb{F}_2^n$. We need to show that if $\sum_j h_{ij} x_j = 0$ iff $\sum_j \tau_{ijk} x_j = 0$, where $h_{ij} = \sum_k \tau_{ijk} \mu_k$. This follows immediately from the independence of the μ_j. □

Example 2.24. Let $\mathbb{F}_4 = \{0, 1, \alpha, \alpha^2\}$, where $\alpha^2 + \alpha + 1 = 0$. Consider the $[8,3]_4$-code with check matrix

$$\begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & \alpha & \alpha & \alpha & \alpha^2 & \alpha^2 & \alpha^2 \\ 0 & 0 & 1 & \alpha & \alpha^2 & 1 & \alpha & \alpha^2 \end{pmatrix}.$$

It can be shown that the minimum distance of this code is 3. A basis of $\mathbb{F}_4/\mathbb{F}_2$ is given by (α, α^2). With respect to this basis a binary check matrix for $C|_{\mathbb{F}_2}$ is

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

The rank of this matrix is 5, so that $C|_{\mathbb{F}_2}$ is an $[8,3]_2$-code. The minimum distance of this code is at least 3. In fact, this code has minimum distance 4. Among all codes of length 8 and dimension 3, this code has the largest possible minimum distance.

The code on $\mathbb{F}_4$ is obtained from an algebraic geometric construction using a maximal elliptic curve over the field. Such codes will be the topic of one of the later classes.

Lecture 3
Bounds on Codes

3.1. Introduction

One of the fundamental problems in coding theory is to determine, for a given q, the possible set of values for triples (n, k, d) for which there exists an $[n, k, d]_q$-code. More precisely, we define the function

$$A_q(n, d) := \max\{k \mid \exists\ [n, k, d]_q\text{-code}\}.$$

Exact values for $A_q(n, d)$ are known only for small values of n and d. A good source for looking up results like this would be the server maintained by Andries Brouwer's which is available at

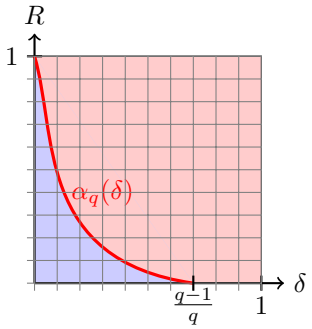
`http://www.win.tue.nl/~aeb/voorlincod.html`

Because of the intractability of exact values for this function, one might want to look at asymptotic assertion. Typically, this is done by fixing for example the ratio d/n, and looking at the upper limit of $A_q(n, d)/n$ as n goes to infinity. We define

$$\alpha_q(\delta) := \limsup_{n\to\infty} \frac{A_q(n, \lfloor \delta n \rfloor)}{n}.$$

The asymptotic theory is concerned with the determination of this function.

Despite its easy look, we do not as of yet know the values of this function on the interval $[0, (q-1)/q]$, save for two points: the point $\delta = 0$, and the point $\delta = (q-1)/q$, where the value is 1, and 0, respectively. Nevertheless, we know provable upper and lower bound for this function, and some coding theorists even believe that they know the shape of $\alpha_2(\delta)$. This lecture is devoted to some of the things that we can prove about $\alpha_q(x)$. The situation of the function $\alpha_q(x)$ is as depicted here.



The pink region (northeast of $\alpha_q(\delta)$) is not achievable, but the blue region (southwest of $\alpha_q(\delta)$) is achievable.

3.2. First Bounds

Theorem 3.1 (Singleton Bound)**.** *For an $[n, k, d]_q$-code we have $k + d \leq n + 1$. Codes for which equality holds are called Maximum Distance Separable (MDS) codes.*

Proof. Consider the vector space $V = \mathbb{F}_q^{d-1} \times 0^{n-d+1}$ of dimension $d-1$. Since the code C is of minimum distance d, we have $V \cap C = \{0\}$, so that $\dim V + \dim C \leq n$, i.e., $k + d - 1 \leq n$. □

Example 3.2.
1. The parity code is defined as $P = \langle (1, 1, \ldots, 1) \rangle^{\perp}$. The minimum distance of the code is 2, and its dimension is $n-1$. It is MDS.
2. Consider the dual of the parity code. Its distance is n and its dimension is 1. It is MDS.

The Singleton bound is of somewhat limited use, since it does not take into account the dependence on q. To get to more serious bounds, we look at the asymptotic behavior of the set of triples (n, k, d) for which there exists an $[n, k, d]_q$-code. The Singleton bound states that $\alpha_q(\delta) \leq 1 - \delta$.

We now consider a better bound. We start with the following inequality

Lemma 3.3. *Let C be an $[n, k, d]_q$-code and $\theta = (q-1)/q$. Assume that $d \geq \theta n$, then*

$$k \leq \log_q \left(\frac{d}{d - n\theta} \right)$$

Proof. We start by establishing the following

$$q^k(q^k - 1)d \leq \sum_{x,y \in C, x \neq y} d(x, y) \leq n\theta q^{2k}, \tag{3.1}$$

where $d(x, y)$ is the Hamming distance between x and y. The inequality $\sum_{x,y \in C, x \neq y} d(x, y) \geq q^k(q^k - 1)d$ is clear. The other inequality is obtained by writing down all the q^k codewords of C into a $q^k \times n$-matrix. Fix a column, and let n_j denote the number of times the element j of the alphabet occurs in that column. The contribution of the column to the sum is $\sum_{j=1}^{q} n_j(q^k - n_j)$, so that

$$\begin{aligned}
\sum_{x,y \in C, x \neq y} d(x, y) &= n \sum_{j=1}^{q} n_j(q^k - n_j) \\
&= n \left(q^{2k} - \sum_j n_j^2 \right) \qquad (\sum_j n_j = q^k) \\
&\leq n \left(q^{2k} - \frac{q^{2k}}{q} \right) \qquad \text{(Cauchy-Schwarz)} \\
&= n\theta q^{2k}.
\end{aligned}$$

Now Equation (3.1) can be rewritten as $(d - \theta n)q^k \leq d$, which leads to the result. □

Theorem 3.4 (Plotkin bound)**.** *We have*

$$\alpha_q(\delta) \begin{cases} \leq -\delta \frac{q}{q-1} + 1 & \text{if } \delta \leq \frac{q-1}{q}, \\ = 0 & \text{if } \delta \geq \frac{q-1}{q}. \end{cases}$$

Proof. Let $\theta := (q-1)/q$. We first observe that $\alpha_q(\delta) = 0$ for $\delta > \theta$ by a straitforward application of the preceeding lemma.

For the proof of the other assertion, we shorten C to a new code C' with parameters $[n', k', d]_q$ where $n' = \lfloor \frac{d-1}{\theta} \rfloor$. Now the preceeding lemma can be applied to the new code, yielding $k' \leq \log_q \left(\frac{d}{d - n'\theta} \right) \leq \log_q \left(\frac{d}{1-\theta} \right)$. On the other hand, the gap between the dimension $k - k'$ cannot exceed the gap between the length $n - n'$, so $k \leq n - n' + k' \leq n - \lfloor \frac{d-1}{\theta} \rfloor + \log_q \left(\frac{d}{1-\theta} \right)$. Taking the limsup of k/n clearly gives the expected result. □

The *Hamming bound* has a simple interpretation. Suppose that we have an $[n, k, d]_q$-code, and consider the Hamming balls of radius $(d-1)/2$ around the codewords (Hamming ball of radius $(d-1)/2$ around x = set of all words of distance $\leq (d-1)/2$ from x). Since the words of the code are distance d apart, these balls are mutually disjoint. If V denotes their volume, we have $q^k V \leq q^n$, and hence $k/n \leq 1 - \log_q(V)/n$. Let $H_q(x) := x \log_q(q-1) - x \log_q(x) - (1-x) \log_q(1-x)$ denote the q-ary entropy function. Then we have the following result the proof of which we leave as an exercise.

Proposition 3.5. *Let V be a Hamming ball of radius rn around the word $0 \in \mathbb{F}_q^n$. Then $|V| = q^{nH_q(r) + o(n)}$.*

From this proposition we obtain the following bound.

Theorem 3.6 (Hamming bound)**.** $\alpha_q(\delta) \leq 1 - H_q(\delta/2)$.

3.3. The Linear Programming Bound

The idea of the Linear Programming Bound is to collect as many information as one can on the distribution of the weights of a code and then to use then as constrains in a linear program where one try to maximize what would stand for the size of a code.

Let C be a $[n,k,d]_q$ -code. Recall the the weight enumerator polynomial is

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

where A_i is the number of codeword of weight i. Let $C^\perp$ be the dual code and remember that

$$A_{C^\perp}(x,y) = \frac{1}{q^k} A_C(y-x, y+(q-1)x).$$

In particular, if we expand that latter expression, we get

$$\begin{aligned} A_C(1-x,1+(q-1)x) &= \sum_{i=0}^{n} A_i \left(\sum_{j=0}^{i} \binom{i}{j} (-1)^j x^j \right) \left(\sum_{\ell=0}^{n} -i \binom{n-i}{\ell} (q-1)^\ell x^\ell \right) \\ &= \sum_{i=0}^{n} A_i \left(\sum_{s=0}^{n} \left(\sum_{j=0}^{s} \binom{i}{j} (-1)^j \binom{n-i}{s-j} (q-1)^{s-j} \right) x^s \right) \\ &= \sum_{s=0}^{n} \left[\sum_{i=0}^{n} A_i \sum_{j=0}^{s} (-1)^j \binom{i}{j} \binom{n-i}{s-j} (q-1)^{s-j} \right] x^s \end{aligned}$$

If we define the Krawtchouk polynomials for fixed n, q by

$$K_s(x) = \sum_{j=0}^{s} (-1)^j \binom{x}{j} \binom{n-x}{s-j} (q-1)^{s-j}$$

we have finally

$$A_C(1-x,1+(q-1)x) = \sum_{s=0}^{n} \left(\sum_{i=0}^{n} A_i K_s(i) \right) x^s$$

One observe thus that

- $|C| = \sum_{i=0}^{n} A_i$.
- for any $i = 0, \ldots, n$, $A_i \geq 0$.
- $A_0 = 1$, $A_1 = A_2 = \cdots = A_{d-1} = 0$.
- for any $s = 0 \ldots, n$, $\sum_{i=0}^{n} A_i K_s(i) \geq 0$.

It is now obvious that the following bound holds.

Theorem 3.7. *Let $q, n, d \in \mathbb{N}$, $q \geq 2$. Let m be the maximum of the linear program*

$$\max \left\{ \left\lfloor \left(\sum_{i=0}^{n} A_i \right) \right\rfloor \;\middle|\; A_0 = 1, A_1 = \cdots = A_{d-1} = 0, A_d, \ldots, A_n \geq 0, \right.$$

$$\left. \forall k = 0, \ldots, n-1 \colon \quad \sum_{i=0}^{n} A_i K_k(i) \geq 0 \right\}$$

Then we have

$$A_q(n,d) \leq \log_q(m)$$

Example 3.8. We prove the optimality of the $[8,5,3]_4$-code that we encountered in the last lecture. The weight distribution of this code involves only the nonnegative parameters $A_0, A_3, A_4, A_5, A_6, A_7, A_8$. The inequalities for these parameters are

$$\begin{aligned}
A_0 + A_3 + A_4 + A_5 + A_6 + A_7 + A_8 &\geq 0\\
24\,A_0 + 12\,A_3 + 8\,A_4 + 4\,A_5 - 4\,A_7 - 8\,A_8 &\geq 0\\
252\,A_0 + 48\,A_3 + 12\,A_4 - 8\,A_5 - 12\,A_6 + 28\,A_8 &\geq 0\\
1512\,A_0 + 44\,A_3 - 40\,A_4 - 28\,A_5 + 16\,A_6 + 28\,A_7 - 56\,A_8 &\geq 0\\
5670\,A_0 - 150\,A_3 - 74\,A_4 + 50\,A_5 + 30\,A_6 - 70\,A_7 + 70\,A_8 &\geq 0\\
13608\,A_0 - 252\,A_3 + 120\,A_4 + 44\,A_5 - 96\,A_6 + 84\,A_7 - 56\,A_8 &\geq 0\\
20412\,A_0 + 216\,A_3 + 108\,A_4 - 144\,A_5 + 100\,A_6 - 56\,A_7 + 28\,A_8 &\geq 0\\
17496\,A_0 + 324\,A_3 - 216\,A_4 + 108\,A_5 - 48\,A_6 + 20\,A_7 - 8\,A_8 &\geq 0\\
6561A_0 - 243A_3 + 81A_4 - 27A_5 + 9A_6 - 3A_7 + A_8 &\geq 0
\end{aligned}$$

The linear program which maximizes $A_0 + A_3 + A_4 + A_5 + A_6 + A_7 + A_8$ subject to the constraints above, and subject to the non-negativity of the parameters gives a solution

$$A_0 = 1, A_3 = 72, A_4 = 210, A_5 = 432, A_6 = 792, A_7 = \frac{4152}{7}, A_8 = \frac{1683}{7}.$$

The sum of these values is $16384/7$, so that $A_4(8,3) \leq \lfloor \log_4(16384/7) \rfloor = 5$, which shows the optimality of the code.

We mention without proof the following upper bound on α_2 due to McEliece, Rodemich, Ramsey, and Welch (called the MRRW-bound). Its proof is based on the linear programming approach.

Theorem 3.9 (MRRW-bound)**.** *We have*

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

3.4. The Gilbert-Varshamov Bound

The bounds we have discussed so far are all upper bounds on α_q, i.e., they lead to asymptotic non-existence theorems. The Gilbert-Varshamov bound below is a lower bound. We will prove it for nonlinear codes, and will later show that there are also "more explicit" linear codes that achieve this bound. However, to make matters more precise

Theorem 3.10 (Gilbert-Varshamov bound)**.** *For $\delta \in [0,(q-1)/q]$ we have $\alpha_q(\delta) \geq 1 - H_q(\delta)$.*

Proof. Fix $\delta \in [0,(q-1)/q]$, and $\varepsilon > 0$, and set $R = 1 - H_q(\delta) - \varepsilon$. Let H be a random $(1-R)n \times n$-matrix over $\mathbb{F}_q$, i.e., every entry of H is independently and uniformly distributed over $\mathbb{F}_q$. Let C be the right-kernel of H, i.e., H is a check matrix for C. It suffices to prove that

$$\Pr[d(C) < \delta n] \leq q^{-\varepsilon n + o(n)},$$

where $d(C)$ is the minimum distance of C. To this end, we use the union-bound to obtain

$$\Pr[d(C) < \delta n] \leq \sum_{\substack{0 \neq x \in \mathbb{F}_q^n \\ \mathrm{wgt}(x) < \delta n}} \Pr[x \in C].$$

Since H is random, any nonzero $x \in \mathbb{F}_q^n$ has probability $1/q^{n-Rn}$ to be in C, hence

$$\Pr[d(C) < \delta n] \leq |V(\delta)| q^{-(1-R)n},$$

where $V(\delta)$ is the set of all words in $\mathbb{F}_q^n$ of weight $\leq \delta n$. From Proposition 3.5 we thus obtain

$$\Pr[d(C) < \delta n] \leq q^{(H_q(\delta) - (1-R) + o(1))n} = q^{-n\varepsilon + o(n)},$$

which completes the proof. □

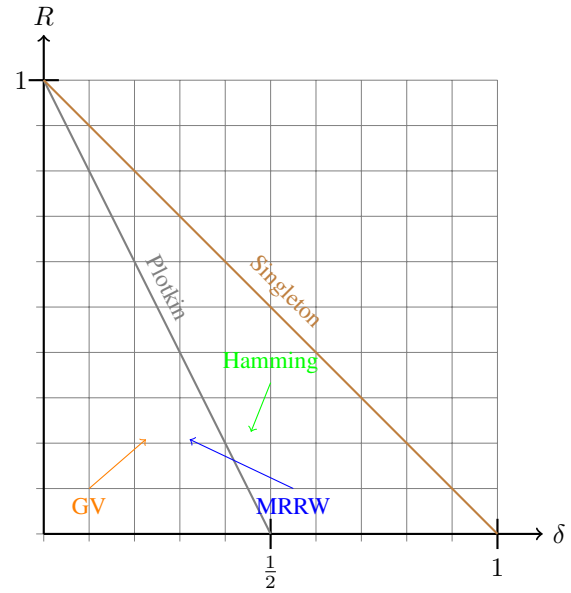


Figure 3.1: Upper and lower bounds for $\alpha_2(x)$.

Some of the upper bounds of the previous sections, and the lower GV-bound are depicted in Figure 3.1. The above theorem not only shows that the GV-bound is asymptotically achievable, it also shows that random codes achieve it. We now show a simpler construction of linear codes achieving the GV bound over $\mathbb{F}_2$. We first construct 2^n pairwise non-intersecting n-dimensional subspaces of $\mathbb{F}_2^{2n}$. Consider the field $\mathbb{F}_{2^n}$, and fix a basis of this field as an $\mathbb{F}_2$-vector space. Elements of this field can be represented as n-dimensional vectors over $\mathbb{F}_2$. Let $r(a)$ denote this binary representation of $a \in \mathbb{F}_{2^n}$. For $\alpha \in \mathbb{F}_{2^n}$, let $C_\alpha := \{(r(x), r(\alpha x)) \mid x \in \mathbb{F}_{2^n}\}$. This vector space is obviously n-dimensional (because $\mathbb{F}_{2^n}$ injects into it as a vector space). Moreover, suppose that $0 \neq (a, b) \in C_\alpha \cap C_\beta$. If $a = r(x)$, this means that $b = r(\alpha x) = r(\beta x)$, i.e., $\alpha x = \beta x$, which shows that $\alpha = \beta$, since $x \neq 0$ (otherwise $a = b = 0$).

Theorem 3.11. *Suppose that $1/2 > \varepsilon > 0$. If $h(\delta) = 1/2 - \varepsilon$, then for large enough n an overwhelming fraction of the C_α's have a minimum distance $\geq 2\delta n$.*

Proof. The number of words of relative weight $\leq \delta$ in $\mathbb{F}_2^{2n}$ is $2^{2nh(\delta)+o(n)} = 2^{n(1-2\varepsilon)+o(n)}$, and each such word can belong to at most one C_α. So the fraction of C_α's that have a low-weight word is $2^{-2n\varepsilon+o(n)}$, which is exponentially small. □

Using shortening the bound can be made to work for any rate (not just 1/2).

Despite this, we do not know of a single sequence of explicitly describable binary codes which meets the GV-bound! This is a challenging open problem. We will provide later in this class sequences of non-binary codes which surpass the GV-bound. For binary codes it is conjectured that the GV-bound is sharp, though I am not aware of deeper reasons for this conjecture. It is very much an open problem to prove or disprove this conjecture.

On another note, even though we know that there are sequences of binary codes attaining this bound, we do not know of a single explicit such sequence. This is another exciting open problem. Both these problems seem to be very very difficult (not a good idea to bank your PhD on them). An interesting question that may be possible to answer is the following:

Question 3.12. *Let $\delta \in (0, 1/2)$. Exhibit an explicit family of sets S_i of binary codes of length n_i such that $|S_i| = 2^{o(n_i)}$, $n_i \to \infty$, and such that for all i there is a code in S_i of rate at least $1 - h(\delta) - o(1)$ and minimum distance $\geq \lfloor \delta n_i \rfloor$.*

Of course, even if the $o(n_i)$ term above is $O(\log(n_i))$, one does not necessarily have a polynomial time algorithm (polynomial in n_i) to compute the correct codes, unless there is a way of telling what the minimum distance of the code in question is without actually going through all the codewords. Nevertheless, it would be interesting to exhibit such families of codes.

3.5. Final Remarks

The upper bounds derived in this lecture are also valid if we take nonlinear codes, though the proofs are slightly different than the ones we gave here. The lower bound given by the Gilbert-Varshamov curve is obviously also valid for nonlinear codes, and it is perhaps interesting to know that it is also conjectured to be sharp for nonlinear binary codes.

Lecture 4
Evaluation Codes

4.1. Definition

Definition 4.1. Let $S = \{P_1, \ldots, P_n\}$ be a finite set. Consider a subspace M of the $\mathbb{F}_q$-vector space of functions from S to $\mathbb{F}_q$. We call *evaluation code via* $(M, S, \mathbb{F}_q)$ the image of the map

$$\iota \colon M \to \mathbb{F}_q^n, \qquad \phi \mapsto (\phi(P_1), \ldots, \phi(P_n)). \tag{4.1}$$

Despite their appearance, evaluation codes are not peculiar, as the following simple result suggests:

Proposition 4.2. *Any linear code is an evaluation code.*

Proof. Consider a generator matrix G of an $[n, k, d]_q$-code. In this case, the vector space M will be the dual space $(\mathbb{F}_q^k)^*$ of $\mathbb{F}_q^k$, i.e., the $\mathbb{F}_q$-space of all linear forms on $\mathbb{F}_q^k$, and the set S will be the set of columns of G. The encoding of an element v of $\mathbb{F}_q^k$ is given as $v \cdot G$. Denoting by ϕ the linear form corresponding to v, the encoding of v equals $(\phi(c_1), \ldots, \phi(c_n))$. □

What can we say about the dimension and minimum distance of such a code? The answer is: essentially nothing meaningful, if we don't have an "interpretation" for M and for S. But at least we can rephrase the problem.

Proposition 4.3. *Let C be an evaluation code via $(M, S, \mathbb{F}_q)$.*

(1) *the dimension of C is $\dim_{\mathbb{F}_q}(M) - \dim_{\mathbb{F}_q}(T)$, where T is the subspace of M for which all the function disappear on all the points of S.*

(2) *Suppose that all $\phi \in M$ which do not identically disappear on S have the property that they have at most t zeros on S. Then the minimum distance of C is at least $n - t$.*

Proof. The kernel of ι from (4.1) is exactly equal to T, so the assertion on the dimension follows. The assertion on the minimum distance is trivial. □

Construction of "good" codes amounts to finding M and S such that M has many elements, points of S separate elements of M, and elements of M don't have too many zeros on S. What M and S are is anybody's best guess. A good method often uses interpretations for M and S so that the evaluation problem becomes some well-studied problem within that interpretation.

For example, remember the Hadamard codes we mentioned in one of the earlier lectures. This code is an evaluation code via $(M, S, \mathbb{F}_2)$, where M is the dual space of $\mathbb{F}_2^k$, and S is the set of all nonzero elements of $\mathbb{F}_2^k$. Any linear code of dimension k is equivalent to a punctured version of this code. The dimension of the code is k, since a nonzero linear form cannot vanish on all elements of $\mathbb{F}_2^k$. The minimum distance of the code is 2^{k-1}, since a linear form has exactly $2^{k-1} - 1$ zeros among the nonzero elements of $\mathbb{F}_2^k$.

In the next few lectures we will give ample examples of evaluation codes.

4.2. Cyclic Codes

4.2.1. Simple Evaluation Codes

In our first construction, we take $q = 2^k$ and identify $\mathbb{F}_q$ with $\mathbb{F}_2^k$. We select an element $\omega \in \mathbb{F}_q$. We consider M as the dual space of $\mathbb{F}_2^k$ and S as $\{1, \omega, \ldots, \omega^{n-1}\}$ for some n.

If S does not contain a basis of $\mathbb{F}_q/\mathbb{F}_2$, then there will be a linear form vanishing on all the points of S, an event we would like to exclude. If S contains a basis, and $n = k$, then the evaluation code is equal to $\mathbb{F}_2^k$, hence trivial. We therefore assume that the elements of S generate $\mathbb{F}_q$ as an $\mathbb{F}_2$-space, and that $n > k$.

It is more convenient to consider the dual code of this particular evaluation code. Consider the space of all "binary relations" on S. These are the set of all binary $(a_0, \ldots, a_{n-1})$ such that $\sum_i a_i\omega^i = 0$. This vector space is the dual space of the evaluation code we are considering. In fact, if $\sum_i a_i\varphi(\omega^i) = 0$ for all linear forms φ, then by linearity $\sum_i a_i\omega^i = 0$, and vice versa.

Lemma 4.4. *Let C be the evaluation code via $(M, S, \mathbb{F}_2)$ as described above. Then $C^\perp$ is the set (of coefficients) of all binary polynomials f of degree $< n$ such that $f(\omega) = 0$. Moreover, any such polynomial is a multiple of the minimal polynomial g of ω over $\mathbb{F}_2$.*

If $\mathcal{I}$ denotes the ideal $g(x)\mathbb{F}_2[x]$, and $\mathcal{I}_{<n}$ denotes the space of polynomials in $\mathcal{I}$ of degree less than n, then this space is exactly the dual of the evaluation code. The dimension of this space is $n - \deg(g)$ (why?), so that the dimension of the evaluation code is $\deg(g)$.

As for the minimum distance, we can prove the following.

Lemma 4.5. *Suppose that any subset of S of size $n - d + 1$ contains a basis of $\mathbb{F}_q/\mathbb{F}_2$. Then the minimum distance of the evaluation code is at least d.*

Proof. Suppose that there is a nonzero word of weight $d - 1$ or less. This word has at least $n - d + 1$ zeros, so there is a nonzero linear form that vanishes on a subset of S of size at least $n - d + 1$. But such a subset contains a basis, and a nonzero linear form cannot vanish on every element of a basis. This yields the desired contradiction. □

Example 4.6. Let $q = 16$, $\mathbb{F}_q = \mathbb{F}_2[x]/(f)$ with $f(x) = x^4 + x + 1$, $\omega := x \bmod f$ and $n = 8$. The minimal polynomial of ω is by definition $x^4 + x + 1$, and hence $\mathcal{I}$ is generated by this polynomial. The codewords of this code and the codewords of the dual code are given in Figure 4.1 As can be seen, the code and its dual are both $[8, 4, 3]_2$-codes.

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

(a)

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

(b)

Figure 4.1: (a) The codewords of the evaluation code of Example 4.6, and (b) the codewords of the dual code

4.2.2. Cyclic Codes

Suppose now that ω is an nth root of unity, i.e., $\omega^n = 1$. Then the minimal polynomial of ω is a divisor of $x^n - 1$. Let $\mathbb{F}_\ell = \mathbb{F}_2(\omega)$ be the smallest extension of $\mathbb{F}_2$ containing ω. We would like to study the evaluation code via $(M_\omega, S, \mathbb{F}_2)$, where $M_\omega = (\mathbb{F}_\ell)^*$. In this case, the evaluation map is injective, since S contains a basis of $\mathbb{F}_\ell$ (by construction).

Definition 4.7. A code C is said *cyclic* when $(c_0, \ldots, c_{n-1})$ is in the code iff $(c_{n-1}, c_0, \ldots, c_{n-2})$ is.

Proposition 4.8. *When ω is an nth root of unity, the evalutation code via $(M, S, \mathbb{F}_2)$ is cyclic.*

To see this, note that if $(a_0, \ldots, a_{n-1})$ is in the dual code, then $\sum_i a_i\omega^i = 0$, so

$$0 = \omega \sum_i a_i\omega^i = \sum_i a_i\omega^{i+1} = a_{n-1} + a_0\omega + \cdots + a_{n-2}\omega^{n-1}.$$

Hence $(a_0, \ldots, a_{n-1})$ is in the dual code iff its cyclic shift is. A moment's thought reveals that therefore the original code is cyclic as well.

The evaluation codes given above are sometimes called *irreducible cyclic codes*. This means that they are cyclic, and there is no subspace of them that is cyclic (this needs a proof, of course). Moreover, it turns out that *any* cyclic code is a direct sum of such simple evaluation codes (or irreducible cyclic codes). All these facts are naturally embedded in the representation theory of finite groups (in this case: cyclic groups), but we do not have the time to touch upon this very interesting tangent.

We will now proceed by proving some useful facts about cyclic codes.

Theorem 4.9. *Let C be a cyclic code of length n. Then there is a unique monic divisor $g(x)$ of $x^n - 1$ in $\mathbb{F}_2[x]$ such that C is the set of (coefficients of) polynomials of the form $f(x)g(x) \bmod x^n - 1$.*

Conversely, for any polynomial $g(x)$ the set of polynomials $f(x)g(x) \bmod x^n - 1$ forms a cyclic code.

Proof. Identify a vector $(a_0, a_1, \ldots, a_{n-1})$ in $\mathbb{F}_2^n$ with the polynomial $a(x) = a_0 + a_1x + \cdots + a_{n-1}x^{n-1}$. Since C is cyclic, $a(x) \in C$ iff $xa(x) \bmod x^n - 1$ is in C. Let $g(x)$ be a monic polynomial of lowest degree in C. Then $g(x)$ divides any $a(x)$ in C: otherwise, there is $0 \neq r(x)$ of degree lower than $\deg(g)$ such that $a(x) = g(x)q(x) + r(x)$ for some polynomial $q(x)$, and hence $r(x) \in C$ since $a(x)$ and $g(x)q(x)$ are. This also shows that $g(x)$ is unique (otherwise it divides another monic polynomial of same degree, which cannot be).

The polynomial $g(x)$ also divides $x^n - 1$, since otherwise $x^n - 1 = u(x)g(x) + v(x)$ for some nonzero $u(x)$, and some $v(x)$ of degree less than $g(x)$, and hence $v(x) \in C$, contradicting the minimality of $g(x)$.

Conversely, if $C = \{f(x)g(x) \bmod x^n - 1 \bmod f(x) \in \mathbb{F}_2[x]\}$, then with any $h(x) \in C$ also $xh(x) \bmod x^n - 1$ is in C, and the code is cyclic. □

The polynomial $g(x)$ from the previous theorem is called *the generator polynomial* of the code C. Obviously, all polynomials in the code are of the form $g(x)f(x)$ with $\deg(f)$ strictly less than $n - \deg(g)$ and all such polynomials are distinct. Hence, we have

Corollary 4.10. *If C is a cyclic code with generator polynomial $g(x)$ and length n, then* $\dim(C) = n - \deg(g)$.

4.2.3. Minimum Distance of Cyclic Codes

How about the minimum distance of a cyclic code? Suppose that n is odd. Then there exists a *primitive* nth root of unity ω over $\mathbb{F}_2$. This means that $\omega^i = 1$ iff $i \equiv 0 \bmod n$. (Why?) The following theorem gives a lower bound on the minimum distance of a cyclic code given by its generator polynomial.

Theorem 4.11. *Suppose that the generator polynomial g of a cyclic code satisfies $g(\omega^i) = \cdots = g(\omega^{i+d-2}) = 0$, for some i and d. Then the minimum distance of the code is at least d.*

Proof. For simplicity we will assume that $i = 0$. The reader can verify that trivial modifications make the proof work for the case of arbitrary i. Consider the matrix

$$H = \begin{pmatrix} 1 & 1 & 1 & \cdots & 1 & 1 \\ 1 & \omega & \omega^2 & \cdots & \omega^{n-2} & \omega^{n-1} \\ 1 & \omega^2 & \omega^4 & \cdots & \omega^{2(n-2)} & \omega^{2(n-1)} \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 1 & \omega^{d-2} & \omega^{2(d-2)} & \cdots & \omega^{(d-2)(n-2)} & \omega^{(d-2)(n-1)} \end{pmatrix}.$$

Because g is the generator polynomial of C, if $x = (x_0, \ldots, x_{n-1})$ is in C, then

$$H \cdot x^\top = 0,$$

which means that the code C' (over $\mathbb{F}_q = \mathbb{F}_2(\omega)$) which has H as a check matrix contains C. Note that any $d - 1$ columns of H are independent, since they form a Vandermonde matrix. Hence, the minimum distance of C' is at least d, which implies that the minimum distance of C is at least d. □

Example 4.12. (1) [Repetition code] Let n be an integer (not necessarily odd), and $g = x^{n-1} + x^{n-2} + \cdots + x + 1$. The corresponding code has dimension 1, and contains only the all-one and the all-zero codewords. It is the repetition code of length n, with minimum distance n.

(2) [Parity code] Let n be an integer, and $g = x - 1$. The corresponding code has dimension $n - 1$. Any codeword is a multiple of $x - 1$, and hence vanishes at 1, when considered as a polynomial. This means that the sum of all coordinates of any codeword is zero, and the code is the parity code of minimum distance 2.

(3) [Hamming code] Let $n = 7$, and $g(x) = x^3 + x + 1$. If ω is a root of g, then ω is a primitive 7th root of unity, and the other roots of g are ω^2 and ω^4. This code has dimension 4. By the previous theorem, the minimum distance of this code is at least 3. In fact, the minimum distance is exactly 3 since $g(x)$ is a codeword of that weight, and the code is equivalent to the $[7, 4, 3]_2$-Hamming code, i.e., after a possible one-time permutation of the codewords, we obtain the $[7, 4, 3]_2$-Hamming code.

(4) (Optimal 2-error correcting code of length 15) Let $n = 15$. The polynomial $x^{15} - 1$ has the factorization

$$x^{15} - 1 = (x - 1)(x^4 + x + 1)(x^4 + x^3 + 1)(x^4 + x^3 + x^2 + x + 1)(x^2 + x + 1).$$

Let $g(x) = (x^4 + x + 1)(x^4 + x^3 + x^2 + x + 1)$. Let ω be a root of $x^4 + x + 1$. Then ω is a primitive 15th root of unity, and the other roots of this polynomial are ω^2, ω^4, and ω^8. The roots of $x^4 + x^3 + x^2 + x + 1$ are $\omega^3, \omega^6, \omega^9$, and ω^{12}. Hence, $g(x)$ has $\omega, \omega^2, \omega^3, \omega^4$ as its roots, so that the minimum distance of C is at least 5 (so C is 2-error correcting). The dimension of C is $15 - 8 = 7$, so C is a $[15, 7, \geq 5]_2$-code.

One can check that the minimum distance of this code is indeed 5, and that the code is optimal, in the sense that it is not possible to find a binary linear code of shorter length and same dimension and minimum distance.

4.3. Weight Distribution of Irreducible Cyclic Codes

The weight distribution of irreducible cyclic codes has astonishing relationships to other areas of mathematics, most notably, algebraic geometry. In this section, we will describe one such connection. We will only sketch the proofs, and leave the detailed presentation to the exercises. To fix the notation, we let ω be a primitive nth root of unity over $\mathbb{F}_2$, and let $\mathbb{F}_{2^k} = \mathbb{F}_2(\omega)$. Throughout this section, we denote by C_n the simple evaluation code via $(M, S, \mathbb{F}_2)$, where M is the dual space of $\mathbb{F}_{2^k}$, and $S = \{1, \ldots, \omega^{n-1}\}$.

Starting point of our journey is the trace map. Recall that the trace of $\mathbb{F}_{2^k}$ to $\mathbb{F}_2$ of an element $\alpha \in \mathbb{F}_{2^k}$ is $\mathrm{Tr}(\alpha) = \alpha + \alpha^2 + \cdots + \alpha^{2^{k-1}}$.

Lemma 4.13. *Let φ be a linear form of $\mathbb{F}_{2^k}$ as an $\mathbb{F}_2$-space. Then there is an $\alpha \in \mathbb{F}_{2^k}$ such that for all $x \in \mathbb{F}_{2^k}$, we have $\varphi(x) = \mathrm{Tr}(\alpha x)$.*

Proof. (Sketch) It is easy to show that $\mathrm{Tr}(\alpha x)$ and $\mathrm{Tr}(\beta x)$ are different linear forms on $\mathbb{F}_{2^k}$ if α and β are different. Hence, the set of all $\mathrm{Tr}(\alpha x)$ coincides with the set of all linear forms of $\mathbb{F}_{2^k}$. □

Lemma 4.14. *If an element $\alpha \in \mathbb{F}_{2^k}$ has trace zero, then there exists $\beta \in \mathbb{F}_{2^k}$ such that $\alpha = \beta^2 - \beta$.*

Proof. (Sketch) The map $\beta \mapsto \beta^2 - \beta$ is a linear map of $\mathbb{F}_{2^k}$. Moreover, its image is clearly contained in the space of elements of trace 0. Being a quadratic polynomial over a field, it is at most 2-to-1, so its image has at least 2^{k-1} elements. Hence, its image is the set of elements of trace 0. □

Lemma 4.15.

$$C_n = \{(\mathrm{Tr}(\alpha), \mathrm{Tr}(\alpha\omega), \ldots, \mathrm{Tr}(\alpha\omega^{n-1})) \mid \alpha \in \mathbb{F}_{2^k}\}.$$

Proof. Follows from the two previous lemmas. □

Proposition 4.16. *Notations being as in the previous lemma, let $m = (2^k - 1)/n$, and let N_α denote the number of $\mathbb{F}_{2^k}$-solutions of the equation $\alpha x^m = y^2 - y$ for which $x \neq 0$. Then the weight distribution of C_n is*

$$\sum_{\alpha \in \mathbb{F}_{2^k}} x^{n - N_\alpha/2m} y^{N_\alpha/2m}.$$

Proof. The number of zeros of the vector $(\mathrm{Tr}(\alpha), \mathrm{Tr}(\alpha\omega), \ldots, \mathrm{Tr}(\alpha\omega^{n-1}))$ is the number of times $\mathrm{Tr}(\alpha\omega^i)$ is zero, i.e., the number of i such that there exists y with $\alpha\omega^i = y^2 - y$, or equivalently, the number of i such that there exists y with $\alpha(\tau^i)^m = y^2 - y$, where τ is a primitive element of $\mathbb{F}_{2^k}$. □

The equation $\alpha x^m = y^2 - y$ describes a *curve* over the field $\mathbb{F}_{2^k}$. There is an area of algebraic geometry which studies the number of points of curves over finite fields. We will briefly touch upon this area later in the course.

Example 4.17. (1) (Hadamard codes) Let k be a positive integer, and $n = 2^k - 1$. Then $m = 1$, and we need to look at the solution of the equation $\alpha x = y^2 - y$. For any nonzero α, the number of solutions of this equation with $x \neq 0$ is $2 \cdot (2^{k-1} - 1)$. Hence, the weight of the corresponding codeword is 2^{k-1}. The code in question is in fact the Hadamard code.

(2) (Lower bounds from elliptic cuves) Let $n = 21$. Then $k = 6$, i.e., the smallest extension of $\mathbb{F}_2$ containing a primitive 21st root of unity is $\mathbb{F}_{2^6} = \mathbb{F}_{64}$. In this case $m = (2^k - 1)/n = 3$, and the weight distribution of the irreducible cyclic code C_{21} is associated with the number of points of the curve $\alpha x^3 = y^2 - y$, for which $\neq 0$. The case $x = 0$ gives two points, so we need to subtract 2 from the total number of points. This curve is called an *elliptic curve* if $\alpha \neq 0$. By a general theorem from algebraic geometry (called the *Hasse bound*), the number of points of such a curve cannot exceed $63 + 2 \cdot 8 = 79$. As a result, a nonzero codeword in C_{21} has at least $21 - 77/6$ nonzero positions, which means that the weight of a nonzero codeword in C_{21} has weight at least 8. This makes the code a $[21, 6, \geq 8]_2$-code. Using the Griesmer bound (or checking Andries Brouwer's tables) shows that this code is optimal. (This result also could have been obtained differently, for example using Theorem 4.11.) On the other hand, the number of points of an elliptic curve over $\mathbb{F}_{64}$ cannot be less than $63 - 2 \cdot 8 = 47$, so a codeword in C_{21} cannot have weight more than $21 - 47/6$, i.e., more than 13. In fact, an inspection yields that this code has only weights 0,8, and 12.

Lecture 5
Decoding Binary BCH Codes

5.1. BCH Codes

5.1.1. Definition

Suppose that we are given an odd length n, and a designed minimum distance d, and we are asked to construct a cyclic code of length n and minimum distance at least d. The following procedure would construct such a code using Theorem 4.11. The code obtained this way is called a BCH-code, named after its inventors Bose, Ray-Chaudhuri, and Hocquenghem.

To construct the code, we first factor the polynomial $x^n - 1$ over $\mathbb{F}_2$. Let ω denote a primitive nth root of unity over $\mathbb{F}_2$. Then each of these factors has a set of zeros of the form $\{\omega^{i_1}, \ldots, \omega^{i_t}\}$, if the factor is of degree t. Next, we multiply factors together so that the resulting product has a "consecutive" set of roots, i.e., a subset of roots of the form $\{\omega^i, \omega^{i+1}, \ldots, \omega^{i+d-2}\}$. Then Theorem 4.11 implies that the minimum distance of the code is at least d.

Definition 5.1. The binary BCH code of length n and designed distance δ is the cyclic code of length n whose generator is the minimal polynomial of $\{\omega, \omega^2, \ldots, \omega^{\delta-1}\}$.

Of course, while combining the factors of $x^n - 1$, we need to ensure that we combine a minimum number of such factors so as to obtain the consecutive set of zeros. This can be at times challenging, with the problems compounded by the fact that the root of unity ω can be replaced by any other primitive nth root of unity, thereby adding to the challenge of checking whether a particular combination of factors is good.

5.1.2. Example

Suppose we want to enumerate all the BCH codes of length 15 on $\mathbb{F}_2$. We factor $x^{15} - 1$ into irreducible factors :

$$x^{15} - 1 = (x-1)\underbrace{(x^4 + x + 1)}_{g_1}\underbrace{(x^4 + x^3 + 1)}_{g_2}\underbrace{(x^4 + x^3 + x^2 + x + 1)}_{g_3}\underbrace{(x^2 + x + 1)}_{g_4}.$$

We need to choose a primitive 15th root of unity. Let us introduce ω such that $\omega^4 + \omega + 1 = 0$. We are going to work on $\mathbb{F}_{16} \simeq \mathbb{F}_2[\omega]$. Now the 15th roots of unity split as follows : $\{\omega^1, \omega^2, \omega^4, \omega^8\}$ are the roots of g_1, $\{\omega^7, \omega^{11}, \omega^{13}, \omega^{14}\}$ are the roots of g_2, $\{\omega^3, \omega^6, \omega^9, \omega^{12}\}$ are the roots of g_3 (note that these are the primitive 5th roots of unity) and eventually $\{\omega^5, \omega^{10}\}$ are the roots of g_4. (You can observe that the set of roots are indeed stable under the Frobenius map $x \mapsto x^2$ or equivalently that the exponents form stable classes of $\mathbb{Z}/15\mathbb{Z}$ under the multiplication by 2)

Now the different BCH codes are the following :

Designed distance δ	Exponent of the roots	Dimension	Error correction	Generator g
$2, 3$	$\{1, 2, 4, 8\}$	11	1	g_1
$4, 5$	$\{1, 2, 3, 4, 6, 8, 9, 12\}$	7	2	$g_1 g_2$
$6, 7$	$\{1, 2, 3, 4, 5, 6, 8, 9, 10, 12\}$	5	3	$g_1 g_2 g_3$
$8, \ldots, 15$	$\{1, \ldots, 14\}$	1	7	$g_1 g_2 g_3 g_4$

i	ω^i
0	1
1	ω
2	ω^2
3	ω^3
4	$\omega+1$
5	$\omega^2+\omega$
6	$\omega^3+\omega^2$
7	$\omega^3+\omega+1$

i	ω^i
8	ω^2+1
9	$\omega^3+\omega$
10	$\omega^2+\omega+1$
11	$\omega^3+\omega^2+\omega$
12	$\omega^3+\omega^2+\omega+1$
13	$\omega^3+\omega^2+1$
14	ω^3+1

Table 5.1: Table of values of ω^i

5.2. Decoding the $[15,7,5]_2$-BCH Code

Consider the $[15,7,5]_2$-code C we introduced in the last section. Its generator polynomial is $g(x)=(x^4+x+1)(x^4+x^3+x^2+x+1)$, and if ω denotes a zero of x^4+x+1, then the roots of $g(x)$ are

$$\omega,\omega^2,\omega^3,\omega^4,\omega^6,\omega^8,\omega^9,\omega^{12}.$$

Suppose that we have a received word $y=(y_0,\ldots,y_{14})$ of which we know that it has distance ≤ 2 from a codeword. Our task is to find the closest codeword c to y. We know that $y=c+e$, where e is of weight at most 2. Let us assume for now that e is of weight exactly 2, and it is one at positions i_1 and i_2, say. What do we know then?

Let $y(x)=\sum_i y_i x^i$, $c(x)=\sum_i c_i x^i$, and $e(x)=x^{i_1}+x^{i_2}$. The only polynomial in this list that we know is $y(x)$. But, we also know that $y(\alpha)=e(\alpha)$ whenever $c(\alpha)=0$, so that we know S_i for $i=1,2,3,4,6,8,9,12$, where

$$S_i := y(\omega^i).$$

Thus, set $X:=\omega^{i_1}$ and $Y:=\omega^{i_2}$, we know the following:

$$\begin{aligned} X+Y &= S_1 \\ X^2+Y^2 &= S_2 \\ X^3+Y^3 &= S_3 \\ X^4+Y^4 &= S_4. \end{aligned}$$

We have left out the other equations, because, as it turns out, these four are sufficient to determine the error positions i_1 and i_2. Our task is to find X and Y, two nonzero and different elements in $\mathbb{F}_{2^k}=\mathbb{F}(\omega)$ satisfying the above equations. Note that this description is also valid when the weight of e is 1: in that case, X or Y are zero.

Suppose that we could find XY from these equations. Then we would know $X+Y$ (which is S_1), and XY, so that we know the polynomial $(X-z)(Y-z)=z^2-(X+Y)z+XY$. Factoring this polynomial then reveals X and Y.

Our task should thus consist of finding XY. Note that

$$S_1^3=(X+Y)^3=(X^2+Y^2)(X+Y)=X^3+Y^3+XY(X+Y)=S_3+XYS_1. \tag{5.1}$$

Since $S_1\neq 0$, we find that $XY=S_1^2-S_3/S_1=S_2-S_3/S_1$, and we can thus decode the code.

Example 5.2. Let ω be a root of $x^4+x+1\in\mathbb{F}_2[x]$. For the following calculation, it is advantageous to keep a table of the various representations of powers of ω, which is given in Table 5.1.

Suppose that we have received the word

$$y=(0,0,0,0,1,0,1,1,1,0,0,0,0,1,0).$$

The polynomial $y(x)$ is then equal to

$$y(x)=x^4+x^6+x^7+x^8+x^{13}.$$

We therefore have

$$\begin{aligned} S_1=y(\omega) &= \omega^2+\omega^3=\omega^6 \\ S_2=y(\omega^2) &= 1+\omega+\omega^2+\omega^3=\omega^{12} \\ S_3=y(\omega^3) &= 1+\omega+\omega^3=\omega^7 \\ S_4=y(\omega^4) &= \omega+\omega^3=\omega^9 \end{aligned} \tag{5.2}$$

The polynomial to be factored is $z^2 + (\omega^2 + \omega^3)z + (\omega^{12} - \omega) = z^2 + \omega^6 z + \omega^{13}$. There are various ways to factor this polynomial, one of which is to try all possible values of ω^i for z. Such a search (called a *Chien Search* in the literature) reveals the two zeros

$$\omega^0, \omega^{13}$$

of this polynomial. This shows that the received word has errors at positions 0 and 13, so that the closest codeword to y is

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

5.3. The Newton Relations

The general method of decoding is about the same: suppose that the generator polynomial of the code has the consecutive set of zeros $\omega, \omega^2, \ldots, \omega^{2t}$. Then we can correct at most t errors. We form the polynomial $y(x)$ from the received word, and calculate

$$S_1 = y(\omega), S_2 = y(\omega^1), \ldots, S_{2t} = y(\omega^{2t}).$$

We call these values the *syndromes* of the received word y.

If the error positions are $i_1, \ldots, i_t$, then we write X_ℓ for ω^{i_ℓ}, and we obtain the relations

$$\begin{array}{rcl} X_1 + \cdots + X_t & = & S_1, \\ X_1^2 + \cdots + X_t^2 & = & S_2, \\ \vdots & \vdots & \vdots \\ X_1^{2t} + \cdots + X_t^{2t} & = & S_{2t}. \end{array}$$

How can we obtain the coefficients of the polynomial $L(z) = (z - X_1) \cdots (z - X_t)$ from these relations? This is expressed in terms of *Newton Relations*. To describe them, it is more convenient to work with the polynomial $H(z) = (1 - X_1 z) \cdots (1 - X_t z)$ instead. Of course, if we can find this polynomial (usually called the *error-locating polynomial*), then we can read off the locations of the errors from this polynomial.

Theorem 5.3. *[Newton Relations] Let $H(z) = (-1)^t \sigma_t z^t + (-1)^{t-1} \sigma_{t-1} z^{t-1} + \cdots - \sigma_1 z + 1$, and let $H'(z)$ denote the formal derivative of H with respect to z. Further, let $S(z) = S_1 + S_2 z + S_3 z^2 + \cdots$ be a power series with the S_i as coefficients. Then we have*

$$H(z) \cdot S(z) = -H'(z).$$

Proof. We work with formal power series. Then we have

$$\begin{array}{rcl} H'(z) & = & -\displaystyle\sum_{j=1}^{t} X_j \frac{H(z)}{1 - X_j z} \\ & = & -H(z) \displaystyle\sum_{j=1}^{t} X_j \sum_{\ell \geq 0} X_j^\ell z^\ell \\ & = & -H(z) \displaystyle\sum_{\ell \geq 0} \left(\sum_{j=1}^{t} X_j^{\ell+1} \right) z^\ell \\ & = & -H(z) \cdot S(z), \end{array}$$

which is equivalent to our assertion. □

Note that $(1 - zX_1) \cdots (1 - zX_t) = 1 - \sigma_1(X_1, \ldots, X_t) + \cdots + (-1)^{t-1} \sigma_{t-1}(X_1, \ldots, X_t) z^{t-1} + (-1)^t \sigma_t(X_1, \ldots, X_t) z^t$, where $\sigma_i(X_1, \ldots, X_t)$ is the ith elementary symmetric polynomial of $X_1, \ldots, X_t$. Hence, this theorem reveals the relationship between $S(z)$ and $H(z)$.

5.4. Recurrence Relations

Let $S = (S_1, S_2, \ldots, S_{2t})$ be a sequence of length $2t$ over a field $\mathbb{F}$. A polynomial $r(z)$ of degree $\leq t$ is called a *recurrence relation* for the sequence if there is a polynomial $\mu(z)$ of degree less than $\deg(r)$ such that $S(z) \equiv \mu(z)/r(z) \bmod z^{2t}$, where $S(z) = S_1 + S_2 z + \cdots + S_{2t} z^{2t-1}$. Note that S may not have a recurrence at all. However,

if it does, then it has a minimal recurrence, i.e., one of minimal degree. It turns out that every recurrence is a multiple of a minimal one, and hence, the minimal recurrence is unique up to scalar multiplication. Sums of recurrence relations are also recurrence relations,

Theorem 5.4. *Suppose that $S(z) = S_1 + S_2 z + \cdots + S_{2t} z^{2t-1}$ has a recurrence. Then it has a minimal recurrence $\mu(z)$, and any other recurrence is a multiple of $\mu(z)$. Moreover, if $S(z) \equiv v(z)/a(z) \bmod z^{2t}$ for co-prime polynomials $v(z)$ and $a(z)$ with $\deg(u) < \deg(a) \leq t$, then $a(z)$ is a minimal recurrence.*

Proof. Let $S(z) \equiv \eta(z)/\mu(z) \bmod z^{2t}$ with $\deg(\eta) < \deg(\mu)$. We can assume that $\eta(z)$ and $\mu(z)$ are co-prime, because otherwise the minimality of $\mu(z)$ would be violated.

Let $a(z)$ be another recurrence relation for S, with $S(z) \equiv v(z)/a(z) \bmod z^{2t}$. Then $a(z)\eta(z) - v(z)\mu(z) \equiv 0 \bmod z^{2t}$. It is easily seen that the degree of the left-hand side is less than $2t$, so that the polynomial on the left is zero. This means that $v(z)/a(z) = \eta(z)/\mu(z)$. Since μ is of minimal degree and $\eta(z)$ and $\mu(z)$ are co-prime, this shows that $a(z)$ is a multiple of $\mu(z)$.

Suppose now that $S(z) \equiv v(z)/a(z)$ with co-prime $a(z)$ and $v(z)$. Since $v(z)/a(z) = \eta(z)/\mu(z)$, the polynomial $a(z)$ must be a scalar multiple of $\mu(z)$, hence is a minimal recurrence. □

The Newton relations of the previous section prove that the error locator is a recurrence for the sequence of syndromes $S_1, S_2, \ldots$. In fact, it turns out that it is a minimal recurrence for this sequence.

Theorem 5.5. *Any recurrence relation of length $2t$ and degree $\leq t$ for the sequence of syndromes is a multiple of the error locator polynomial.*

Proof. Let $H(z)$ denote the error locator. By the Newton relations we have $S(z) \equiv -H'(z)/H(z) \bmod z^{2t}$. Since $H(z)$ does not have double roots, the polynomials $H'(z)$ and $H(z)$ are co-prime, hence $H(z)$ is a minimal recurrence. □

The previous result implies the following.

Corollary 5.6. *If $a(z)$ is a polynomial of degree $\leq t$ and $v(z)$ is a polynomial of degree $< t$, and if*

$$a(z)S(z) \equiv v(z) \bmod z^{2t}, \tag{5.3}$$

then $a(z)$ is a multiple of the error locator $H(z)$. Moreover, a minimal solution to (5.3) (i.e., a nonzero solution of minimal degree) is (up to scalar multiplication) the error locator.

How this can be used is demonstrated in the next sections.

5.5. Decoding BCH-Codes using Matrix Equations

The algorithm in this section is known as the Petersen-Gorenstein-Zierler decoder.

Since multiplication of both sides of (5.3) with a nonzero polynomial does not alter the equation, we can assume that $a(z)$ is of degree t. Let $a_0, a_1, \ldots, a_{t-1}$ denote the coefficients of $z^0, z^1, \ldots, z^{t-1}$ in $a(z)$, respectively. By considering the coefficients of $z^t, z^{t+1}, \ldots, z^{2t-1}$ on both sides of (5.3) we obtain the following system of equations:

$$\begin{pmatrix} S_{t+1} & S_t & \cdots & S_2 \\ S_{t+2} & S_{t+1} & \cdots & S_3 \\ \vdots & \vdots & \ddots & \vdots \\ S_{2t} & S_{2t-1} & \cdots & S_{t+1} \end{pmatrix} \begin{pmatrix} a_0 \\ a_1 \\ \vdots \\ a_{t-1} \end{pmatrix} = \begin{pmatrix} -S_1 \\ -S_2 \\ \vdots \\ -S_t \end{pmatrix}. \tag{5.4}$$

From the results of the last section we know that the above system has a solution, and that any solution is a multiple of the error locator.

The algorithm we present here is a modified version of the Peterson-Gorenstein-Zierler decoder.

1. Upon reception of y, check whether it is in the code. If so, then STOP.
2. Compute the syndromes $S_i = y(\omega^i)$, $i = 1, \ldots, 2t$.
3. Find a nonzero solution a of the system (5.4). Declare ERROR if it does not exist.
4. Form the polynomial $a(z)$ and calculate its roots $\omega^{-i_1}, \ldots, \omega^{-i_s}$ in $\{1, \omega, \ldots, \omega^{n-1}\}$.

5. Declare positions $i_1, \ldots, i_s$ as erasures. Use any erasure decoding algorithm for the BCH-code to recover these positions and obtain a codeword c.

The way it is described, this algorithm uses $O(t^3)$ steps for solving the system (5.4), and an additional $O(k^3)$ steps for the erasure decoding step. This can be improved in several ways. First, the system of equations given in (5.4) is *structured*. The matrix corresponding to this system is called a *Toeplitz* matrix and its inverse can be found with $O(t^2)$ operations over $\mathbb{F}_2(\omega)$. We will encounter structured matrices and methods for inverting them later in the class.

The second way in which this algorithm can be improved is the erasure decoding part, which could also be accomplished with $O(k^2)$ operations over $\mathbb{F}_2(\omega)$. Note that this part could be avoided altogether if we could immediately find the error locator in the first step.

The algorithm provided in the next section is more efficient. In fact, it finds the error locator immediately, though we will not prove this here.

Example 5.7. We use the same example as in the first section. Then, recalling (5.2), the system (5.4) becomes

$$\begin{pmatrix} \omega^7 & \omega^{12} \\ \omega^9 & \omega^7 \end{pmatrix} \cdot \begin{pmatrix} a_0 \\ a_1 \end{pmatrix} = \begin{pmatrix} \omega^6 \\ \omega^{12} \end{pmatrix}.$$

We obtain $a_0 = \omega^2$ and $a_1 = \omega^8$. The polynomial $a(z)$ is thus equal to $\omega^2 + \omega^8 z + z^2$. Its roots are $1 = \omega^0$ and $\omega^2 = \omega^{-13}$, and hence the error locations are contained in 0 and 13. In this case, the error locations are exactly these two values.

5.6. Decoding Binary BCH Codes via the Extended GCD-Algorithm

One way to obtain the polynomials $a(z)$ and $v(z)$ in (5.3) is the following. We apply the extended GCD-algorithm to $S(z)$ and z^{2t}. At each step i of the algorithm, we obtain polynomials $a_i(z)$, $b_i(z)$, and $v_i(z)$ such that

$$a_{i-1}(z)S(z) + b_{i-1}(z)z^{2t} = v_i(z).$$

As soon as the degree of $v_i(z)$ becomes smaller than that of $a_{i-1}(z)$, we have satisfied the assumptions of Corollary 5.6, and we have identified a super-set of the error positions. Erasure decoding, as described in the previous section, will then reveal the correct codeword.

The polynomials a_i, b_i, and v_i are obtained as follows. Set $v_0(z) = z^{2t}$, and $v_1(z) = S(z)$. Then, at each step $i \geq 0$, perform a division algorithm to compute q_i such that $v_i(z) = q_i(z)v_{i+1}(z) + v_{i+2}$. This way, we have

$$\begin{pmatrix} v_{i+1} \\ v_{i+2} \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ 1 & -q_i \end{pmatrix} \cdot \begin{pmatrix} 0 & 1 \\ 1 & -q_{i-1} \end{pmatrix} \cdots \begin{pmatrix} 0 & 1 \\ 1 & -q_0 \end{pmatrix} \cdot \begin{pmatrix} v_0 \\ v_1 \end{pmatrix} =: \begin{pmatrix} b_i & a_i \\ b_{i+1} & a_{i+1} \end{pmatrix} \cdot \begin{pmatrix} v_0 \\ v_1 \end{pmatrix}.$$

To have a matrix equation between $(v_i, v_{i+1})^\top$ and $(v_0, v_1)^\top$ for all values of $i \geq 0$, we define

$$\begin{pmatrix} b_{-1} & a_{-1} \\ b_0 & a_0 \end{pmatrix} := \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}. \tag{5.5}$$

Then we have the recurrence relations

$$\begin{pmatrix} b_i & a_i \\ b_{i+1} & a_{i+1} \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ 1 & -q_i \end{pmatrix} \cdot \begin{pmatrix} b_{i-1} & a_{i-1} \\ b_i & a_i \end{pmatrix} = \begin{pmatrix} b_i & a_i \\ -q_i b_i + b_{i-1} & -q_i a_i + a_{i-1} \end{pmatrix}.$$

The algorithm would thus proceed as follows:

1. Initialize a_i, b_i, $i = -1, 0$, as in (5.5).
2. Initialize $v_0 = z^{2t}$ and $v_1 = S(z)$, $i = 0$.
3. While $\deg(v_i) \geq \deg(a_{i-1})$ do
   (a) (Division step) Compute $q_i(z)$ and v_{i+2} from $v_i(z) = q_i(z)v_{i+1}(z) + v_{i+2}$.
   (b) (Update) Set $a_{i+1} = -q_i a_i + a_{i-1}$, and $b_{i+1} = -q_i b_i + b_{i-1}$.
   (c) Set $i = i + 1$.
4. The polynomial $a_{i-1}(z)$ is now a multiple of the error locator polynomial. Calculate its roots $\omega^{-i_1}, \ldots, \omega^{-i_s}$ in $\{1, \omega, \ldots, \omega^{n-1}\}$.

5. Declare positions $i_1, \ldots, i_s$ as erasures. Use any erasure decoding algorithm for the BCH-code to recover these positions and obtain a codeword c.

On the surface, this algorithm has not really improved the Peterson-Gorenstein-Zierler algorithm. However, it can be shown that the polynomial $a_{i-1}(z)$ found above is in fact a scalar multiple of the error locator, though we are not going to show it here. (This result is due to various researchers, among them Mc Eliece, Welch and Scholtz, Chenc, and Dornstetter).

The algorithm given above is essentially equivalent to the famous "Berlekamp-Massey" algorithm which finds a minimal recurrence of degree $\leq t$ for the first $2t$ values of a given sequence. We will discuss this algorithm in the exercises.

Example 5.8. We continue with the running example of Section 1. Setting $v_0(z) = z^4$ and $v_1(z) = S(z) = \omega^9 z^3 + \omega^7 z^2 + \omega^{12} z + \omega^6$, we obtain

$$\begin{array}{rcccccc}
v_0(z) & = & v_1(z) & \cdot & (\omega^6 z + \omega^4) & + & (\omega^5 z^2 + \omega^{11} z + \omega^{10}) \\
 & = & v_1(z) & \cdot & q_0(z) & + & v_2(z)
\end{array}$$

$$\begin{array}{rcc}
b_1(z) & = & 1 \\
a_1(z) & = & q_0(z)
\end{array}$$

$$\begin{array}{rcccccc}
v_1(z) & = & v_2(z) & \cdot & (\omega^4 z + \omega^7) & + & \omega^3 \\
 & = & v_2(z) & \cdot & q_1(z) & + & v_3(z)
\end{array}$$

$$\begin{array}{rcccccc}
b_2(z) & = & q_1(z) & & & & \\
a_2(z) & = & q_1(z) & \cdot & b_1(z) & + & b_0(z) \\
 & = & (\omega^6 z + \omega^4) & \cdot & (\omega^4 z + \omega^7) & + & 1 \\
 & = & \omega^{10} z^2 + \omega^3 z + \omega^{12}. & & & &
\end{array}$$

We see that $\deg(v_3) < \deg(a_2)$, and hence $a_2(z)$ is a multiple of the error locator. This polynomial has the roots ω^0 and ω^{-13}, which shows that the errors are at most in positions 0 and 13.

Remark 5.9. *Erasure decoding means in our case the following. We have received a word $y(x) = y_0 + \cdots + y_{n-1}x^{n-1}$ where the coefficients $y_{i_1}, y_{i_2}, \ldots, y_{i_s}$ are unknown (but $i_1, \ldots, i_s$ are known). We can fix the value of theses coefficients arbitrarily and thus need to decompose y into*

$$y(x) = c(x) + e(x),$$

where $c(x)$ is a codeword and $e(x) = e_{i_1}x^{i_1} + \cdots + e_{i_s}x^{i_s}$. Now by evaluating this equation for ω^i ($i = 1, \ldots, \delta - 1$) we obtain a linear system of Vandermonde type, which can be easily solved. This final step in not really necessary when working in the binary case but becomes inevitable on a general alphabet $\mathbb{F}_q$.

Lecture 6

Goppa codes

In this lecture, we continue our exploration of classical codes by broadening our view point on BCH codes.

6.1. Motivations

Let $\omega \in \mathbb{F}_{q^m}$ be a n-th root of unity. Let C be a q-ary BCH code of designed distance d. If g denotes the generator polynomial of C, we have in particular that $g(\omega^i) = 0$ for $i \in \{1, \ldots, d-1\}$. Thus a codeword $c(x) = c_0 + c_1 x + \cdots + c_{n-1}x^{n-1}$ belongs to the code if and only if $c(x) \equiv 0 \mod g(x)$ or equivalently iff $c(\omega^i) = 0$ for $i \in \{1, \ldots, d-1\}$.

One can easily derive from this the following check matrix.

$$H = \begin{pmatrix} 1 & \omega & \omega^2 & \cdots & \omega^{n-1} \\ 1 & \omega^2 & \omega^4 & \cdots & \omega^{2(n-1)} \\ \vdots & & & & \vdots \\ 1 & \omega^{d-1} & \omega^{2(d-1)} & \ldots & \omega^{(d-1)(n-1)} \end{pmatrix}$$

and see C as the subfield subcode of the code on $\mathbb{F}_{q^m}$ defined by H.

The fact that the minimal distance of C is lower bounded by d comes from any $d-1$ columns of H being a non singular Vandermonde matrix. Now if we replace H by the following matrix :

$$H = \begin{pmatrix} h_0\beta_0 & h_1\beta_1 & h_2\beta_2 & \cdots & h_{n-1}\beta_{n-1} \\ \vdots & & & & \vdots \\ h_0\beta_0^{d-1} & h_1\beta_1^{d-1} & h_2\beta_2^{d-1} & \cdots & h_{n-1}\beta_{n-1}^{d-1} \end{pmatrix}$$

the same argument concerning the minimal distance apply.

We can rewrite the fact that a codeword c belongs to C, *i.e.* that $\sum_{i=0}^{n-1} c_i(\omega^j)^i = 0$:

$$\frac{z^n - 1}{z - \omega^{-i}} = \frac{z^n - (\omega^{-i})^n}{z - \omega^{-i}} = \sum_{k=0}^{n-1} z^k \omega^{-i(n-1-k)} = \sum_{k=0}^{n-1} z^k \omega^{i(k+1)}$$

Thus

$$\sum_{i=0}^{n-1} \frac{c_i}{z - \omega^{-i}} = \frac{1}{z^n - 1} \sum_{i=0}^{n-1} \sum_{k=0}^{n-1} c_i z^k \omega^{i(k+1)} = \frac{1}{z^n - 1} \sum_{k=0}^{n-1} \underbrace{\left[\sum_{i=0}^{n-1} c_i \omega^{i(k+1)} \right]}_{=0 \text{ si } k+1<d} z^k$$

So for some polynomial $p(z)$, we have

$$\sum_{i=0}^{n-1} \frac{c_i}{z - \beta^{-i}} = \frac{z^{d-1} p(z)}{z^n - 1}.$$

Finally c belongs to C if and only if

$$\sum_{i=0}^{n-1} \frac{c_i}{z - \beta^{-i}} \equiv 0 \mod z^{d-1}.$$

6.2. Goppa codes

Definition 6.1. Let $g(z)$ be a monic polynomial in $\mathbb{F}_{q^m}[z]$ of degree t. Let $L = \{\gamma_0, \ldots, \gamma_{n-1}\} \subseteq \mathbb{F}_{q^m}$ distinct elements such that $g(\gamma_i) \neq 0$. We call *Goppa code* $\Gamma(L, g)$ with *Goppa polynomial* g the set of words $c = (c_0, \ldots, c_{n-1}) \in \mathbb{F}_q^n$ such that

$$\sum_{i=0}^{n-1} \frac{c_i}{z - \gamma_i} \equiv 0 \mod g(z).$$

Example 6.2. Taking $\beta \in \mathbb{F}_{q^m}$ such that $\beta^m = 1$ and $L = \{\beta^{-i}; i = 0, \ldots, m-1\}$ and $g(z) = z^{d-1}$ gives us back the BCH code of designed distance d.

Check matrix of $\Gamma(L, g)$?
Since $g(\gamma_i) \neq 0$, $z - \gamma_i$ is invertible modulo $g(z)$ and

$$\frac{1}{z - \gamma_i} = \frac{-1}{g(\gamma_i)^{-1}} \frac{g(z) - g(\gamma_i)}{z - \gamma_i}$$

So for $c = (c_0, c_1, \ldots, c_{n-1}) \in \Gamma(L, g)$,

$$c \left(\frac{-1}{g(\gamma_0)} \frac{g(z) - g(\gamma_0)}{z - \gamma_0}, \ldots, \frac{-1}{g(\gamma_{n-1})} \frac{g(z) - g(\gamma_{n-1})}{z - \gamma_{n-1}} \right) = 0$$

which gives t equations by looking at the coefficient in front of each monomial (one equation per degree).

Let's be more precise and fix $g(z) = \sum_{i=0}^{t} g_i z^i$. We have the general identity

$$\frac{z^i - x^i}{z - x} = x^{i-1} \sum_{k=0}^{i-1} (zx^{-1})^k$$

and therefore

$$\begin{aligned}
\frac{g(z) - g(x)}{z - x} &= \sum_{i=0}^{t} g_i \frac{z^i - x^i}{z - x} \\
&= \sum_{i=0}^{t} g_i x^{i-1} \sum_{k=0}^{i-1} z^k x^{-k} \\
&= \sum_{k=0}^{t-1} \left(\sum_{i=k+1}^{t} g_i x^{i-k-1} \right) z^k \\
&= \sum_{k=0}^{t-1} \left(\sum_{j=0}^{t-k-1} g_{j+k+1} x^j \right) z^k
\end{aligned}$$

So the following matrix is a check matrix

$$\begin{pmatrix}
h_0 g_t & \cdots & h_{n-1} g_t \\
h_0 \left(g_{t-1} + g_t \gamma_0 \right) & & h_{n-1} \left(g_{t-1} + g_t \gamma_{n-1} \right) \\
\vdots & & \vdots \\
h_0 \left(g_1 + g_2 \gamma_0 \cdots + g_t \gamma_0^{t-1} \right) & & h_{n-1} \left(g_1 + g_2 \gamma_{n-1} \cdots + g_t \gamma_{n-1}^{t-1} \right)
\end{pmatrix}$$

where $h_i = \frac{-1}{g(\gamma_i)}$ for $i = 0, \ldots, n-1$. This matrix can simplified by combining the rows and we get

$$H = \begin{pmatrix}
h_0 \gamma_0^0 & \ldots & h_{n-1} \gamma_{n-1}^0 \\
\vdots & & \vdots \\
h_0 \gamma_0^{t-1} & \ldots & h_{n-1} \gamma_{n-1}^{t-1}
\end{pmatrix}$$

Remark 6.3. *The check matrix H is not completely arbitrary since we have to set $h_i = g(\gamma_i)^{-1}$.*

Theorem 6.4. *The minimal distance of the Goppa code $\Gamma(L, g)$, where the degree of g is t satisfies $\geq t + 1$.*

Proof. Any set of t columns of the check matrix form a Vandermonde matrix and is invertible. □

Proposition 6.5. *The dimension of the Goppa code $\Gamma(L, g)$, where the degree of g is t is $\geq n - tm$.*

Proof. Apply Proposition 2.22. □

6.3. Improved bound on the minimum distance

We change our point of view on the setting to improve the bound on the minimum distance. On the one hand we define $R = \{(c_0, c_1, \ldots, c_{n-1}) \in \mathbb{F}_q^n\}$ equipped with the usual Hamming distance. On the other hand, we set $R^* = \{\sum_{i=0}^{n-1} \frac{c_i}{z-\gamma_i}; c \in R\}$ with the following distance.

Definition 6.6. If $\xi(z)$ and $\eta(z) \in R^*$, we define $d(\xi, \eta) = \deg(\delta(z))$ where $\xi(z) - \eta(z) = \dfrac{\nu(z)}{\delta(z)}$ and $\nu(z) \wedge \delta(z) = 1$.

One can prove (do it as an exercise) that d is a distance on R^* and that the map

$$\begin{array}{ccc} R & \to & R^* \\ c & \mapsto & \displaystyle\sum_{i=0}^{n-1} \frac{c_i}{z-\gamma_i} \end{array}$$

is an isometry.

Let $c(z) = \frac{n(z)}{d(z)}$ in $\Gamma(L, g)$, *i.e.* $\frac{n(z)}{d(z)} \equiv 0 \mod g(z)$ and $n(z) \wedge d(z) = 1$. Then $g(z) | n(z)$. Thus $\deg d \geq \deg n + 1 \geq \deg g + 1$.

Theorem 6.7. *On $\mathbb{F}_2$, if g has no multiple roots, then the minimal distance of $\Gamma(L, g)$ is $\geq 2t + 1$.*

Proof. If $c \in (\mathbb{F}_2)^n$, $\displaystyle\sum_{i=0}^{n-1} \frac{c_i}{z-\gamma_i} = \frac{f'(z)}{f(z)}$ where $f(z) = \prod_{i=0}^{n-1}(z-\gamma_i)^{c_i}$ (Remember that $c_i = 0$ or 1). But in characteristic 2, f' has only even powers (since differentiating z^{2p} yields $2pz^{2p-1}$). So f' is a perfect square, say $f' = h^2$. Now, if g divides f', g divides h^2. But g has no multiple roots, so g divides h. Thus $\deg f \geq \deg f' + 1 \geq 2\deg h + 1 \geq 2\deg g + 1 = 2t + 1$. □

6.4. Decoding Goppa codes

With no surprise, there exists a decoding algorithm very similar to BCH codes. Suppose we receive $y(z) = c(z) + e(z)$ with $|e| \leq \frac{t-1}{2}$ errors. We call *syndrome* the polynomial

$$S(z) = \sum_{i=0}^{n-1} \frac{y_i}{z-\gamma_i} \equiv \sum_{i=0}^{n-1} \frac{e_i}{z-\gamma_i} \mod g(z).$$

Let $M = \{i; e_i \neq 0\}$ be the set of positions where the errors are located. We also introduce the *error-locator polynomial*

$$\sigma(z) = \prod_{i \in M} (z - \gamma_i)$$

and

$$\omega(z) = \sum_{i \in M} e_i \prod_{j \neq i, j \in M} (z - \gamma_j).$$

We observe that

$$S(z)\sigma(z) = \sum_{i \in M} \frac{e_i}{z-\gamma_i} \prod_{i \in M} (z - \gamma_i) \equiv \omega(z) \mod g(z).$$

In this equation, σ and ω are the unknowns. The equation has the same shape as in the preceeding chapter. It is possible to find a solution (σ_1, ω_1) satisfying

$$S(z)\sigma_1(z) \equiv \omega_1(z) \mod g(z)$$

such that $\deg \omega_1 < \deg \sigma_1$ and $\deg \sigma_1$ is minimal. But then,

$$\underbrace{\omega_1(z)\sigma(z) - \omega(z)\sigma_1(z)}_{\deg < t} \equiv 0 \mod g(z).$$

So we have $\omega_1(z)\sigma(z) = \omega(z)\sigma_1(z)$. Since $\omega \wedge \sigma = 1$ and $\sigma | \sigma_1$, we have finally $\sigma = \sigma_1$. Now the knowledge of σ gives directly the locations of the errors.

Lecture 7
Reed-Solomon Codes

In this class we will study a very important class of evaluation codes: The Reed-Solomon codes. Reed-Solomon codes are being used as outer codes in many communication systems. Many of today's ubiquitous devices such as disk drives or CD/DVD players rely heavily on the error correction capabilities of Reed-Solomon codes. This class introduces the basic concepts, and some abstract decoding algorithms. Later lectures will focus on efficient implementations of these algorithms.

7.1. Reed-Solomon Codes as Evaluation Codes

Let $\mathbb{F}_q[x]_{<k}$ denote the $\mathbb{F}_q$-space of univariate polynomials of degree less than k over $\mathbb{F}_q$. Further, let $S = \{x_1, \ldots, x_n\}$ be a subset of $\mathbb{F}_q$. We consider the evaluation code with parameters $(\mathbb{F}_q[x]_{<k}, S, \mathbb{F}_q)$, where the action of $\mathbb{F}_q[x]_{<k}$ on S is given by polynomial evaluation. In other words, the code we are considering is the image of the homomorphism:

$$\begin{array}{rcl} \mathbb{F}_q[x]_{<k} & \to & \mathbb{F}_q^n, \\ f & \mapsto & (f(x_1), \ldots, f(x_n)). \end{array} \tag{7.1}$$

We call this image a *Reed-Solomon code* (or RS code), after the two inventors *Irving Reed* and *Gus Solomon* of these codes. If we want to be very precise, then we denote this code by $\mathrm{RS}(k; x_1, \ldots, x_n)$.

Why should this code be good? The reason is, that we can control the number of zeros in a codeword, using the following well-known theorem.

Theorem 7.1. *A nonzero polynomial f of degree r over a field $\mathbb{F}$ has at most r roots in $\mathbb{F}$.*

This means that if $k < n$, then the image of a nonzero polynomial $f \in \mathbb{F}_q[x]_{<k}$ has at most $k-1$ zero coordinates, and hence the minimum distance of the corresponding code is at least $n - k + 1$. Moreover, by the same token, no nonzero polynomial can be mapped to the zero codeword, hence the mapping is injective, hence the dimension of the code is k. Since by the Singleton inequality the minimum distance cannot be larger than $n - k + 1$, we see that the minimum distance is equal to $n - k + 1$.

Theorem 7.2. *If $k < n$, then the code $\mathrm{RS}(k; x_1, \ldots, x_n)$ has dimension k and minimum distance $n-k+1$. The code is thus MDS.*

It is clear that the following matrix *Vandermonde* matrix is a generator matrix for $\mathrm{RS}(k; x_1, \ldots, x_n)$:

$$\begin{pmatrix} 1 & 1 & 1 & \cdots & 1 \\ x_1 & x_2 & x_3 & \cdots & x_n \\ x_1^2 & x_2^2 & x_3^2 & \cdots & x_n^2 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ x_1^{k-1} & x_2^{k-1} & x_3^{k-1} & \cdots & x_n^{k-1} \end{pmatrix}.$$

RS-codes, as described here, are non-systematic. They can be made systematic in a variety of ways, some of which we will describe when we study the displacement method.

7.2. The Welch-Berlekamp Algorithm

The decoding problem is the following. We are given a word $y = (y_1, \ldots, y_n) \in \mathbb{F}_q^n$ with a promise that it has distance $\leq (n-k)/2$ from a q-ary RS-code $C = \mathrm{RS}(k; x_1, \ldots, x_n)$. The task is to find the closest codeword in C to y. We call this codeword c, and call f the corresponding polynomial in $\mathbb{F}_q[x]_{<k}$. In other words, for all i, $c_i = f(x_i)$.

We use the concept of an error locator described in the previous section. Let us call a polynomial $h(x)$ an *error locator* if $h(x_i) = 0$ if and only if the vector y and its closest codeword c differ in the i-th position. Finding the error locator is tantamount to decoding, since we can discard the positions in error, and perform an erasure decoding on the rest of the vector y.

What do we know about $h(x)$? We know that its degree is at most $(n-k)/2$, and that

$$\forall\, i = 1, \ldots, n: \quad (f(x_i) - y_i) \cdot h(x_i) = 0.$$

or stated differently

$$\forall\, i = 1, \ldots, n: \quad g(x_i) - y_i h(x_i) = 0, \tag{7.2}$$

where $g(x) := f(x)h(x)$ is a polynomial of degree $< k + (n-k)/2 = (n+k)/2$. It does not seem like this equation can give us anything useful, since we only know the y_i. However, it turns out that *any* nontrivial solution to (7.2) with $\deg(h) \leq (n-k)/2$ and $\deg(g) < (n+k)/2$ will solve the decoding problem, i.e., for any nonzero pair (g, h) with the given degree constraints we have $f = g/h$!

Theorem 7.3. *Suppose that $(g, h) \in \mathbb{F}_q[x]_{<(n+k)/2} \times \mathbb{F}_q[x]_{\leq (n-k)/2}$ is a nonzero pair of polynomials satisfying (7.2). Then $h(x)$ is an error locator, g is divisible by h, and $f(x) = g(x)/h(x)$.*

Proof. Consider the polynomial $F(x) := g(x) - f(x)h(x) \in \mathbb{F}_q[x]_{<(n+k)/2}$. Note that for all i we have

$$F(x_i) = g(x_i) - f(x_i)h(x_i) = h(x_i)(y_i - f(x_i)).$$

So, if i is such that $y_i = f(x_i)$, or in other words, if i is not in error, then $F(x_i) = 0$. Since there are at most $(n-k)/2$ error positions, there are at least $(n+k)/2$ positions not in error, so F has at least $(n+k)/2$ zeros. But its degree is less than $(n+k)/2$. So, $F(x)$ is identically zero, and hence $g(x) = f(x)h(x)$. The fact that $h(x)$ is an error locator follows from the above displayed equation. □

Does a nonzero solution to (7.2) exist, and if so, how can we calculate it? Suppose that $g(x) = g_0 + g_1 x + \cdots + g_{(n+k)/2-1} x^{(n+k)/2-1}$, and $h(x) = h_0 + h_1 x + \cdots + h_{(n-k)/2} x^{(n-k)/2}$, with unknown coefficients g_i and h_i. Then (7.2) translates to the following system of linear equations:

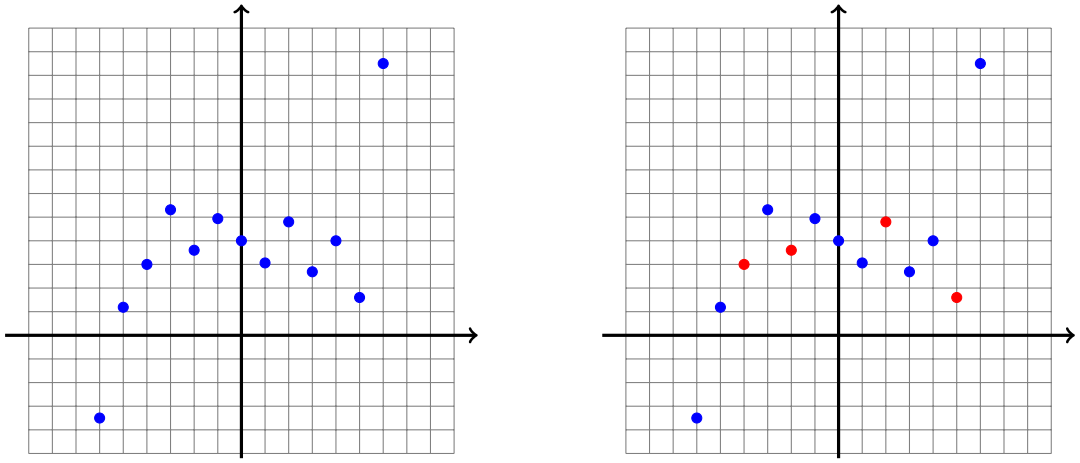
$$\left(\begin{array}{cccc|cccc} 1 & x_1 & \cdots & x_1^{(n+k)/2-1} & -y_1 & -y_1x_1 & \cdots & -y_1x_1^{(n-k)/2} \\ 1 & x_2 & \cdots & x_2^{(n+k)/2-1} & -y_1 & -y_2x_2 & \cdots & -y_2x_2^{(n-k)/2} \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_n & \cdots & x_n^{(n+k)/2-1} & -y_n & -y_nx_n & \cdots & -y_nx_n^{(n-k)/2} \end{array}\right) \cdot \left(\begin{array}{c} g_0 \\ g_1 \\ \vdots \\ g_{(n+k)/2-1} \\ \hline h_0 \\ h_1 \\ \vdots \\ h_{(n-k)/2} \end{array}\right) = 0. \tag{7.3}$$

This is a homogeneous system of n equations in $(n+k)/2 + (n-k)/2 + 1 = n+1$ unknowns, so it has a nonzero solution. Moreover, such a solution can be found by solving this system using, for example, the Gaussian elimination algorithm. However, since the matrix corresponding to this system is *structured*, there are much faster methods for its solution. We will review such methods later in the context of the displacement method.

Summarizing, the Welch-Berlekamp (WB) decoder can be described as follows:

1. Find a nonzero solution of the system (7.3) and form the polynomials $g(x)$ and $h(x)$.
2. Compute $f(x) = g(x)/h(x)$.
3. The closest codeword to the received word is $(f(x_1), \ldots, f(x_n))$.

The WB-decoder has an interesting geometric interpretation. Suppose that we are given n pairs (x_i, y_i) where the x_i are different, and we are asked to fit a polynomial of degree $< k$ through these points so that it passes through as many of the given points as possible. Then, as long as at least $(n+k)/2$ of the points are "correct," i.e., lie on the same polynomial of degree $< k$, we can find the polynomial using the WB-decoder. The situation is exemplified in Figure 7.1 for the case $n = 13$, and $k = 4$.



(a) (b)

Figure 7.1: (a) We are given 13 points of which we know that at least 9 lie on a third degree polynomial; (b) a plot of the unique polynomial passing through 9 of the points. It reveals the four outliers.

7.3. Decoding More Errors: List Decoding

We know that for a given RS-code with parameters $[n, k, n-k+1]_q$, we can uniquely decode up to $e = (n-k)/2$ errors. What happens, though, when the number of errors is larger? In that case we cannot possibly hope for unique decoding in general, but is it possible to at least find all codewords that are with a given Hamming-radius of the received word?

Given the received vector y, and a real number τ, we would like to find all codewords of distance $\leq \tau n$ from y. Phrased this way, the WB-decoder solves the problem when $\tau = (1-R)/2$, $R = k/n$ being the rate of the code. Can we do better?

Let us change our view a little, and consider the received vector as a set of points $(x_i, y_i) \in \mathbb{F}_q^2$ with pairwise distinct x-coordinates. Our task would be to find all polynomials $f(x) \in \mathbb{F}_q[x]_{<k}$ passing through at least $n - \tau n$ of the points.

Suppose that there are ℓ such polynomials, $f_1(x), \ldots, f_\ell(x)$. Then all the points (x_i, y_i) would be zeros of

$$Q(x, y) := (y - f_1(x)) \cdot (y - f_2(x)) \cdots (y - f_\ell(x)) \cdot E(x),$$

with a polynomial $E(x)$ of degree $< e$, which is a summary for all the points that are in error for all the f_i's. If we had $Q(x, y)$ from somewhere, then we could find the polynomials $f_1(x), \ldots, f_\ell(x)$ through finding all factors of the form $y - g(x)$ of $Q(x, y)$.

This is the Ansatz we are going to use: finding a nonzero polynomial $Q(x, y) = Q_\ell(x)y^\ell + Q_{\ell-1}(x)y^{\ell-1} + \cdots + Q_1(x)y + Q_0(x)$, with polynomials $Q_i(x)$ of degree $< e + (\ell - i)(k-1)$ such that

$$\forall\, i = 1, \ldots, n: \qquad Q(x_i, y_i) = 0. \tag{7.4}$$

(We have replaced $E(x)$ with $Q_\ell(x)$ for consistency reasons.) Under certain conditions on τ, k, and n, it turns out that *any* such polynomial Q will have the property that $Q(x, f(x)) = 0$ if $y - f(x)$ passes through at least $n - \tau n$ of the points (x_i, y_i). This idea for doing a *list decoding* is due to M. Sudan, and is known as the Sudan list-decoding algorithm.

Theorem 7.4. *Suppose that $Q(x, y) = Q_\ell(x)y^\ell + \sum_{i=0}^{\ell-1} Q_i(x)y^i$ with polynomials $Q_i(x)$ of degree less than $e + (\ell - i)(k-1)$, and that $Q(x, y)$ satisfies the equations in (7.4). Then any polynomial $f(x) \in \mathbb{F}_q[x]_{<k}$ for which $(f(x_1), \ldots, f(x_n))$ has a distance of at most $n - e - \ell(k-1)$ of the received vector y satisfies $Q(x, f(x)) = 0$.*

Proof. Suppose that $f(x)$ is such that $c = (f(x_1), \ldots, f(x_n))$ has a distance $\leq n - e - \ell(k-1)$ from the received word. Consider the polynomial $F(x) := Q(x, f(x))$. Then the degree of $F(x)$ is less than $e + \ell(k-1)$. Moreover,

$$c_i = y_i \implies F(x_i) = 0.$$

This means that $F(x)$ vanishes at all positions i at which c and y agree. By assumption, c and y agree on at least $e + \ell(k-1)$ positions. But the degree of $F(x)$ is less than $e + \ell(k-1)$, which shows that $F(x) = 0$. □

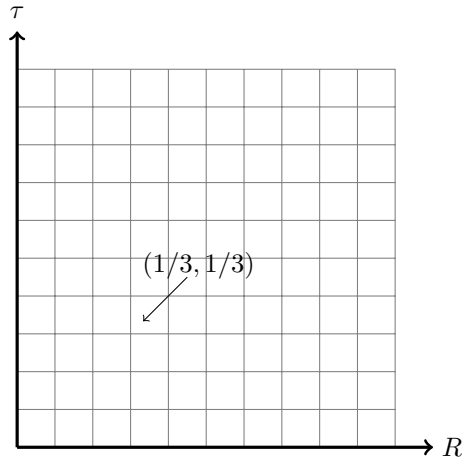


Figure 7.2: The upper curve is the relative error correction capability of Sudan's algorithm versus the rate of the code. It consists of pairwise linear functions. The lower curve is the plot of $1 - \sqrt{2R}$.

Does a nonzero polynomial $Q(x, y)$ satisfying (7.4) exist, and if so, how can we find it? As in the case of the WB-decoder, this problem is reduced to linear algebra. Let us denote by $A_0, A_1, \ldots, A_\ell$ the column vector consisting of the unknown coefficients of $Q_0, Q_1, \ldots, Q_\ell$, respectively. Furthermore, let

$$V_d := \begin{pmatrix} 1 & x_1 & \cdots & x_1^{d-1} \\ 1 & x_2 & \cdots & x_2^{d-1} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_n & \cdots & x_n^{d-1} \end{pmatrix}, \qquad D := \begin{pmatrix} y_1 & 0 & \cdots & 0 & 0 \\ 0 & y_2 & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & y_{n-1} & 0 \\ 0 & 0 & \cdots & 0 & y_n \end{pmatrix}. \tag{7.5}$$

Then (7.4) is equivalent to

$$\left(D^\ell V_e \mid D^{\ell-1} V_{e+k-1} \mid \cdots \mid D V_{e+(\ell-1)(k-1)} \mid V_{e+\ell(k-1)}\right) \cdot \left(\begin{array}{c} A_\ell \\ \hline A_{\ell-1} \\ \hline \vdots \\ \hline A_1 \\ \hline A_0 \end{array}\right) = 0. \tag{7.6}$$

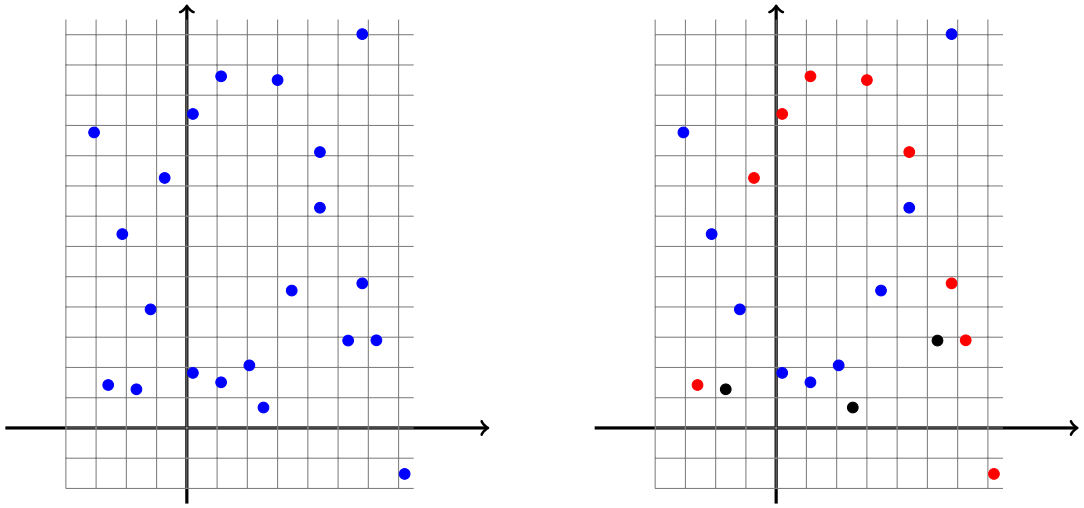
The number of unknowns in this system of equations is $(\ell+1)e + \frac{\ell(\ell+1)}{2}(k-1)$. The number of equations is n. Hence, if n is smaller than the number of unknowns, we will have a nonzero solution of this system. We therefore require

$$n \leq (\ell+1)e + \frac{\ell(\ell+1)}{2}(k-1) - 1. \tag{7.7}$$

From Theorem 7.4 we know that we would be able to find all codewords of distance at most $n - e - \ell(k-1) - 1$ from the received word. The task is to maximize this value, subject to the constraint in (7.7), i.e., to minimize $e + \ell(k-1)$ subject to those constraints. The minimal value for e subject to (7.7) is $n/(\ell+1) - \ell(k-1)/2 + 1$, so that the minimal value for $e + \ell(k-1)$ is $n/(\ell+1) + \ell(k-1)/2 + 1$.

If we phrase everything relative to n, and assume that n is large, we obtain that the algorithm is capable of correcting a fraction of

$$\tau = 1 - \frac{1}{\ell+1} - \frac{\ell}{2}R$$



(a) (b)

Figure 7.3: Demonstration of Sudan's algorithm. (a) We are asked to identify all polynomials of degree at most 2 that pass through at least 9 of the 21 points shown. (b) We find two quadratic polynomials, and three spurious points which don't lie on any of the quadratic curves.

errors with a list of size at most ℓ, provided that $e/n = 1/(\ell+1) - \ell R/2$ is positive, i.e., provided that $R < \frac{2}{\ell(\ell+1)}$. For each ℓ we obtain a fraction of recoverable errors from the previous displayed equation. Figure 7.2 summarizes a plot of this function, which consists of pairwise linear functions. Also plotted is $1 - \sqrt{2R}$, which approximates the error correction capability very well for low rates. As can be seen, the new algorithm is better than the WB-algorithm if the rate is lower than $1/3$.

7.4. A Brief Note on Factoring

Sudan's list decoding algorithm and the algorithm of Guruswami-Sudan described below need to find factors of the form $y - f(x)$ of $Q(x, y)$. One possibility to do this is to factor $Q(x, y)$ completely. This can be done in polynomial time using randomized algorithms. A different approach is to pointedly look for factors of the form $y - f(x)$. Several researchers have dealt with this problem, among them Augot-Pecquet, Gao-Shokrollahi, Høholdt-Nielsen, and Roth-Rückenstein.

There are essentially two approaches: one of them is to work in a large finite field $\mathbb{F}_{q^m}$, and specialize x by a generator of that field. In other words, we take an irreducible polynomial $p(x)$ of sufficiently large degree m, and reduce $Q(x, y)$ modulo that polynomial to obtain a polynomial to obtain a univariate polynomial in y over $\mathbb{F}_{q^m}$. Next, we find roots of that univariate polynomial to obtain factors of the form $y - (f(x) \bmod p(x))$. If the degree of $p(x)$ is larger than the largest degree of $f(x)$ possible, then we can read off the right roots.

The other approach uses something very similar to Newton iteration. We specialize the variable x to some element x_0 in the field, for example 0. Next, we find the roots of $Q(0, y)$. Let the roots be $y_0, y_1, \ldots, y_t$. Then we know that if $y - f(x)$ is a factor of $Q(x, y)$, then $f(0)$ is one of these values. If $Q(0, y)$ does not have multiple roots, then, fixing $f(0)$, we can determine the higher coefficients of $f(x)$ iteratively. Details of this algorithm will be provided in the exercises.

If $Q(0, y)$ has multiple roots, then other techniques have to be used, which we are not going to discuss here.

7.5. The List Decoding Algorithm of Guruswami-Sudan

How can we correct more errors than with the algorithm of Sudan. Guruswami and Sudan had the following idea to do this: suppose that we look for a polynomial that passes through all the points (x_i, y_i) with some given multiplicity. In other words, we are looking for a polynomial of the form

$$Q(x, y) = (y - f_1(x))^r (y - f_2(x))^r \cdots (y - f_\ell(x))^r \cdot E(x)$$

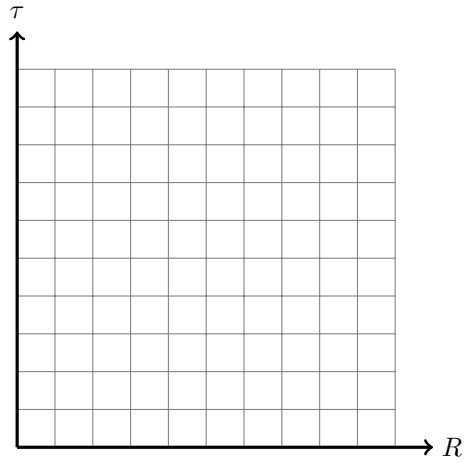


Figure 7.4: Comparison between the error correction capabilities of Sudan's algorithm (lower curve) and the algorithm of Guruswami and Sudan. The latter is better than the former and better than the WB-algorithm on the entire range for R.

describing the set of points. This polynomial has the property that for all the points (x_i, y_i), we have

$$Q(x + x_i, y + y_i) = \sum_{j \geq r} \sum_{s+k=j} q_{s,k}^{(i)} x^s y^k, \tag{7.8}$$

for some $q_{s,k}^{(i)} \in \mathbb{F}_q$. Suppose now that we have found a nonzero polynomial $Q(x, y)$ satisfying (7.8). Consider $F(x) := Q(x, f(x))$ and suppose that y and the evaluation vector of $f(x)$ agree on at least t positions. Then $F(x)$ has rt roots, since each agreement point gives rise to a zero of $F(x)$ of multiplicity r. If we can prove that we can choose $Q(x, y)$ such that the degree of $F(x)$ is less than rt, then this shows that $F(x) = 0$, and hence $(y - f(x))^r$ divides $Q(x, y)$.

To guarantee that the degree of $Q(x, f(x))$ is small, we need to design $Q(x, y)$ properly. For that, we introduce the notion of the (a, b)-degree of a monomial $x^i y^j$, defined as $ai + bj$. The normal degree is then simply the $(1, 1)$-degree. The degree of $Q(x, f(x))$ is thus at most the $(1, k - 1)$-degree of $Q(x, y)$.

Our task is to find a nonzero polynomial $Q(x, y)$ satisfying (7.8) and having a $(1, k - 1)$-degree that is as small as possible. For this, let us write

$$Q(x, y) = \sum_{i \geq 0} \sum_{j+k=i} q_{jk} x^j y^k$$

with unknown coefficients q_{ik}. Then the coefficients of $Q(x + x_i, y + y_i)$ are linear forms in q_{ik}, and Condition (7.8) translates to $r(r + 1)/2$ homogeneous equations for the q_{ik}, one for each of the "missing" monomials. In total, there are $nr(r + 1)/2$ equations.

If we assume that the $(1, k - 1)$-weighted degree of $Q(x, y)$ is less than or equal to D, then the number of monomials of which Q is potentially composed equals

$$\begin{aligned} \sum_{i=0}^{D} \sum_{j=0}^{\lfloor (D-i)/(k-1) \rfloor} 1 &= \sum_{i=0}^{D} \left\lfloor \frac{D-i}{k-1} + 1 \right\rfloor \\ &\geq \sum_{i=0}^{D} \frac{i}{k-1} \\ &= \frac{D(D+1)}{2(k-1)}. \end{aligned}$$

Thus, if we assume that

$$\frac{D(D+1)}{2(k-1)} \geq n \frac{r(r+1)}{2} + 1, \tag{7.9}$$

then we can guarantee the existence of a nonzero $Q(x, y)$ satisfying (7.8), and any polynomial $f(x)$ whose evaluation has an agreement of at least $(D + 1)/r$ with the received vector has the property that $(y - f(x))^r$ divides $Q(x, y)$.

Therefore, we would like to minimize D subject to (7.9). We loosen that condition, and require that

$$\frac{D^2}{2(k-1)} \geq n\frac{(r+1)^2}{2},$$

leading to $D \geq (r+1)\sqrt{n(k-1)}$. Choosing D to be equal to the expression on the right will lead to an agreement of at least

$$\frac{r+1}{r}(\sqrt{n(k-1)}+1).$$

Phrasing everything in terms of the error correction radius τ, this means that this algorithm is capable to find a short list of all polynomials that are within a relative distance of

$$1 - \frac{r+1}{r}\sqrt{R}$$

of the received word. Since there is no a-priori bound on r, in the limit, as r grows large, we obtain an algorithm that can "correct" up to a fraction of

$$1 - \sqrt{R}$$

errors.

Figure 7.3 shows a plot of the error correction capability of this algorithm versus the algorithm of Sudan. As can be seen, the new algorithm is better than the old one on the entire range for R. Moreover, the new algorithm is also better than the WB-algorithm on the entire range.

We finish this section with some comments on the computational complexity of this algorithm. We have to solve a system of equations with $O(nr^2)$ unknowns. Using normal Gaussian elimination, we would need $O(n^3r^6)$ operations. However, the matrix we are dealing with has structure, and using various techniques (such as the displacement method discussed later in the class), one can solve the system in $O(n^2r^4)$ operations. On the other hand, we have to choose r to be very large in order to fully benefiting from the algorithm, which may render the algorithm ineffective in practice. A Master's Thesis by Julie Marc, done at our lab, studies this algorithm from the point of view of practicability. Interested readers may want to consult that thesis (which will be available on our web site soon).

7.6. Further Developments

A natural question to ask is whether the $1 - \sqrt{R}$ bound of Guruswami-Sudan is the best possible, i.e., if there are examples in which the size of the list is too large when the number of errors is beyond this bound. The answer to this question is not known. What we do know is that extensions of Reed-Solomon codes over very large alphabets have list decoding algorithms that can decode well beyond the $1 - \sqrt{R}$ bound — almost as far as $1 - R$, which is the ultimate error correcting radius. More precisely, if the alphabet size is q^m, then one can design codes of length at most q and decoding algorithms that would be able to correct up to a fraction of $1 - \sqrt[m+1]{R^m}$ errors. For $m = 1$, we recover the Guruswami-Sudan bound, and as m goes to infinity, the error correction radius converges to $1 - R$.

The first such algorithm (with a slightly worse error correction radius than given above) was provided by to Parvaresh and Vardy, and it was vastly extended by Guruswami and Rudra to get the error correction radius reported above. These algorithms were inspired by another algorithm, due to Bleichenbacher, Kiyayias, and Yung. The latter algorithm studies Reed-Solomon for which the length is significantly smaller than the alphabet size, and provides error correction algorithms when these codes are used on the q-ary symmetric channel. These algorithms can correct almost up to a fraction of $1 - R$ of errors, with very small error probabilities. An extension of these algorithms, and a detailed analysis was provided by Brown, Minder, and the author, and is available from our web site.

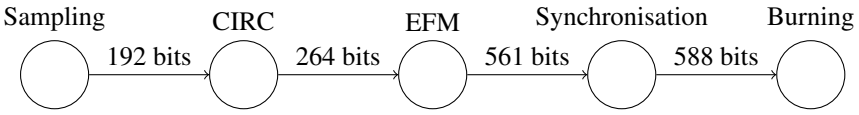
Lecture 8
Compact disc encoding

We present in this chapter how coding theory is used to provide reliable storage of music on compact disc.

8.1. General view

The sound track to be stored on the compact disc is sampled at a frequency of 44,1 kHz on 2^{16} levels (2 bytes) and two voices (stereo sound). It is thus necessary to use 4 bytes for a sample and 196 400 bytes per second of music. The samples are grouped by 6 to form frames of 24 bytes (or 192 bits). So the rate is $44100/6 = 7300$ frames per second.

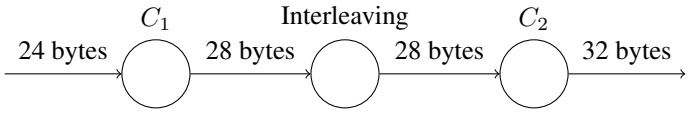
To write data on the CD, one proceeds according to the following scheme.



CIRC stands for "cross interleaved Reed-Solomon codes" ; EFM stands for eight to fourteen modulation. Burning and reading constraints requires to introduce binding bytes (to avoid for instance too long sequences of identical bits or periodicity that could lead to resonnance phenomena).

In terms of physical length, one frame is thus 588 bits which uses $588 \times 0.3 = 180\mu$m . In total, a disc can contain $2 \cdot 10^7$ frames which is 5.7 km or 74 minutes of music.

We shall only focus here on the CIRC part. The CIRC code mixes two error correcting codes and an interleaving and can be described as follows.



8.2. Tools

Each byte is represented by an element of $S = \mathbb{F}_{2^8} = \mathbb{F}_{256} \simeq (\mathbb{F}_2)^8$ (A vector base of $\mathbb{F}_{256}$ is fixed once and for all).

8.2.1. Interleaving with delay r

Given an integer n, we want to transmit a stream of elements from S^n (we shall use $n = 28$ in practice) : $\mathbf{m^{(0)}} = (m_0^{(0)}, m_1^{(0)}, \ldots, m_{n-1}^{(0)})$, $\mathbf{m^{(1)}}$, $\mathbf{m^{(2)}}$, ... We could write these symbols as a matrix and read the matrix column by

column.

$$\begin{pmatrix} m_0^{(0)} & m_0^{(1)} & m_0^{(2)} & m_0^{(3)} & m_0^{(4)} & m_0^{(5)} & m_0^{(6)} & \cdots \\ m_1^{(0)} & m_1^{(1)} & m_1^{(2)} & m_1^{(3)} & m_1^{(4)} & m_1^{(5)} & m_1^{(6)} & \cdots \\ \vdots & & & & & & & \\ m_{n-1}^{(0)} & m_{n-1}^{(1)} & m_{n-1}^{(2)} & m_{n-1}^{(3)} & m_{n-1}^{(4)} & m_{n-1}^{(5)} & m_{n-1}^{(6)} & \cdots \end{pmatrix}$$

Now interleaving with delay r means shifting each row by r coordinates with respect to the previous row and still reading the stream column by column. This adds some extra empty characters at the beginning, that we can fill in by a fixed symbol, say $\bullet$. Let's see an example with $r = 2$ and $n = 3$.

$$\begin{pmatrix} m_0^{(0)} & m_0^{(1)} & m_0^{(2)} & m_0^{(3)} & m_0^{(4)} & m_0^{(5)} & m_0^{(6)} & \cdots \\ \bullet & \bullet & m_1^{(0)} & m_1^{(1)} & m_1^{(2)} & m_1^{(3)} & m_1^{(4)} & \cdots \\ \bullet & \bullet & \bullet & \bullet & m_2^{(0)} & m_2^{(1)} & m_2^{(2)} & \cdots \end{pmatrix}$$

The sequence after interleaving becomes

$$(m_0^{(0)}, \bullet, \bullet, m_0^{(1)}, \bullet, \bullet, m_0^{(2)}, m_1^{(0)}, \bullet, m_0^{(3)}, m_1^{(1)}, \bullet, m_0^{(4)}, m_1^{(2)}, m_2^{(0)}, m_0^{(5)}, m_1^{(3)}, m_2^{(1)}, \dots)$$

8.2.2. Reed-Solomon codes

We consider α a primitive 255-th root of unity in $\mathbb{F}_{256}$. Remember that the Reed-Solomon code

$$C_0 = \text{RS}(251; 1, \alpha, \alpha^2, \dots, \alpha^{254})$$

is the set of codewords $\big(u(1), u(\alpha), \dots, u(\alpha^{254})\big)$ where $u(x)$ goes through all polynomial of degree less than k. The minimal distance of C_0 is 5 (Remember that RS codes are MDS). For $s = 28$ and $s = 32$, we shorten the code C_0 by keeping the first s coordinates of the codewords of C_0 whose $n - s$ last coordinates are zero. Specificly, we denote $C_1 = C_0 \cap (\mathbb{F}_{256}^{28} \times 0)$ and $C_2 = C_0 \cap (\mathbb{F}_{256}^{32} \times 0)$. C_1 is a $[28, 24, 5]_{256}$-code and C_2 is a $[32, 28, 5]_{256}$-code.

The next results show that C_1 and C_2 are also cyclic, a property that will help us deal with them practically.

Lemma 8.1. *Let K be a field and n a positive integer. Let α be a primitive n th root of unity. Let $m(x)$ and $u(z)$ be polynomials of degree less than n. The following assertions are equivalent :*

1. $n\, m(x) = \sum_{0 \le i < n} u(\alpha^i) x^i$
2. $u(z) = \sum_{0 \le j < n} m(\alpha^{-j}) z^j$.

Proof. We define the linear map $m \mapsto \phi_m(z) = \sum_{0 \le j < n} m(\alpha^{-j}) z^j$. We would like to prove that $\sum_{0 \le i < n} \phi_m(\alpha^i) x^i = nm(x)$. This would prove that the second assertion implies the first. For a monomial $m(x) = x^i$, we have $\phi_m(z) = \sum_{0 \le j < m} \alpha^{-ij} z^j$ so

$$\phi_m(\alpha^r) = \sum_{0 \le i < n} (\alpha^{r-i})^j = \begin{cases} n & \text{if } r = i \\ 0 & \text{otherwise.} \end{cases}$$

Thus if $m = \sum_{0 \le i < n} m_i x^i$, $\phi_m(\alpha^i) = n m_i$, and $\sum_{0 \le i < n} \phi_m(\alpha^i) x^i = nm(x)$ as we wanted. The converse implication is showed the same way. □

Proposition 8.2. *The code RS$(k; 1, \alpha, \alpha^2, \dots, \alpha^{q-2})$ defined on $\mathbb{F}_q$ is a cyclic code, whose generator polynomial is $g(x) = \prod_{i=1}^{q-1-k} (x - \alpha^i) \in \mathbb{F}_q[x]/\langle x^{q-1} - 1 \rangle$.*

Proof. A word of the form $(u(1), u(\alpha), \dots, u(\alpha^{q-1}))$ belongs to the Reed-Solomon code if $\deg u < k$, which is equivalent, according to the lemma, to the fact that $m(\alpha^j) = 0$ for $k \le j \le q - 1$ or $m(\alpha^i) = 0$ for $0 < i < d$. □

Encoding of the frames We present a way to encode the frames that doesn't hide the initial message (this is general to any cyclic code). Initially, a frame $\mathbf{m} = (m_0, m_1, \dots, m_{23}) \in (\mathbb{F}_{256})^{24}$ can be presented as the following polynomial of degree less than 24 :

$$m(x) = m_0 + m_1 x + \cdots + m_{23} x^{23} \in \mathbb{F}_{256}[x]$$

A codeword of C_1 is a polynomial $p(x)$ of degree less than 28 that is a multiple of $g(x) = (x - \alpha)(x - \alpha^2)(x - \alpha^3)(x - \alpha^4)$ or else such that $p(\alpha) = p(\alpha^2) = p(\alpha^3) = p(\alpha^4) = 0$ We can write the Euclidean division

$$x^4 m(x) = q(x) g(x) + r_0 + r_1 x + r_2 x^2 + r_3 x^3 + r_4 x^4.$$

Then $-r+x^4m(x)$ belongs to the code C_1 and is very easy to represent since it is simply the word $(-r_0, -r_1, -r_2, -r_3, m)$ The $(r_i)_{1\leq i\leq 4}$ are solutions of the system

$$\begin{cases} r_0 + r_1\alpha + r_2\alpha^2 + r_3\alpha^3 &= \alpha^4 m(\alpha) \\ r_0 + r_1\alpha^2 + r_2\alpha^4 + r_3\alpha^6 &= \alpha^8 m(\alpha^2) \\ r_0 + r_1\alpha^3 + r_2\alpha^6 + r_3\alpha^9 &= \alpha^{12} m(\alpha^3) \\ r_0 + r_1\alpha^4 + r_2\alpha^8 + r_3\alpha^{12} &= \alpha^{16} m(\alpha^4) \end{cases}$$

8.3. Details about encoding and decoding for CIRC

8.3.1. Encoding

First encoding : We use C_1 as presented. C_1 can correct up to 4 erasures.

Interleaving : The 28 bytes frames are interleaved with a delay of $r = 4$. After the operation, one frame is spread on $28 \cdot 4 + 1 = 113$ bits, so it can support the erasure of $4 \cdot 113 = 452$ consecutive bytes (length on the disc of 2.88 mm).

Second encoding : We use C_2, which can correct up to 2 errors.

8.3.2. Decoding

For C_2. We receive 32 bytes $\mathbf{m'} = \mathbf{m_0} + \mathbf{e_0}$ where $\mathbf{m_0}$ was the message initial encoded and $\mathbf{e_0}$ is the error that really occured. The decoder tries to find a decomposition $\mathbf{m'} = \mathbf{m} + \mathbf{e}$ with $\mathrm{wgt}(\mathbf{e})$ as small as possible. Of course, there is no garanty that $\mathbf{m} = \mathbf{m_0}$. Even though C_2 is able to correct up to 2 errors, we won't try to correct more than one error. There are three possible cases :

a. $\mathbf{m'} \in C_2$, i.e. $\mathrm{wgt}(\mathbf{e}) = 0$: we decode $\mathbf{m'}$ as $\mathbf{m}$

b. $\mathrm{wgt}(\mathbf{e}) = 1$: we decode $\mathbf{m'}$ as $\mathbf{m}$.

c. $\mathrm{wgt}(\mathbf{e}) \geq 2$: we declare $\mathbf{m}$ as unknown.

We have $\mathbf{m}+\mathbf{e} = \mathbf{m_0}+\mathbf{e_0}$. So falling in the case a. or b. when $\mathbf{m} \neq \mathbf{m_0}$ requires that $\mathrm{wgt}(\mathbf{e_0}) \geq \mathrm{wgt}(\mathbf{m}-\mathbf{m_0}) - \mathrm{wgt}(\mathbf{e_0}) \geq 5-1 = 4$. More precisely, let us try to upper bound the probability to be in case a. or b. while $\mathbf{m} \neq \mathbf{m_0}$ by assuming that the error e_0 is uniformely distributed among $\mathbb{F}_{256}^{32}$ (i.e. $\mathbf{m'}$ is any vector with equal probability). There are $1 + n(q-1)$ words at distance ≤ 1 from a codeword and q^k codewords in the code, while there are q^n words in total. So the probability to lie at distance ≤ 1 from the code is $q^{k-n}(1 + n(q-1))$ which is a rough upper bound of the probability to be in the case a. or b. wrongly. When $q = 256$ and $n = 32$, this evaluates to $1.9 \cdot 10^{-6}$.

Decision protocol for C_2 : We first compute the syndromes from the received message $\mathbf{m'}$: $s_1 = \mathbf{m'}(\alpha)$, $s_2 = \mathbf{m'}(\alpha^2)$, $s_3 = \mathbf{m'}(\alpha^3)$ and $s_4 = \mathbf{m'}(\alpha^4)$. Case a. would give

$$s_1 = s_2 = s_3 = s_4 = 0.$$

Case b. would give for some λ and j

$$s_1 = \lambda\alpha^j, \quad s_2 = \lambda\alpha^{2j}, \quad s_3 = \lambda\alpha^{3j}, \quad s_4 = \lambda\alpha^{4j}.$$

- If $s_1 = s_2 = s_3 = s_4 = 0$, return case a.
- If $s_1 \neq 0$, $s_2 \neq 0$, $s_2^2 = s_1 s_3$ and $s_1 s_4 = s_2 s_3$, return case b. and set $\lambda = s_1^2/s_2$, $\alpha^j = s_2/s_1$.
- Otherwise return case c.

Now it remains to de-interleave the decoded stream and apply an erasure decoding relative to C_1.

Decoding C_1 : We need to find a polynomial $p(x)$ divisible by $(x-\alpha)(x-\alpha^2)(x-\alpha^3)(x-\alpha^4)$. Now

$$p(x) = p^{\text{known}}(x) \cdot (x-\alpha)(x-\alpha^2)(x-\alpha^3)(x-\alpha^4) + \lambda_1 x^{n_1} + \cdots + \lambda_e x^{n_e}$$

with $e \leq 4$. Evaluating for $x = \alpha, \ldots, \alpha^4$ gives a Vandermonde system, that can be solved to recover the errors.

Lecture 9
The Displacement Method

The method of displacement is an algorithmic tool for solving systems of linear equations in which the matrix has a certain type of structure. In many cases, the structure can be used to considerably reduce the running time of algorithms based on Gaussian elimination. Such algorithms turn out the be very useful in the context of decoding evaluation codes, like the ones introduced in the last lectures.

9.1. Definition

A displacement operator $\nabla = \nabla_{A,B}$ on the space $\mathbb{F}^{m\times n}$ of $m \times n$-matrices over the field $\mathbb{F}$ is a linear map of the form

$$X \mapsto \nabla(X) := AX - XB$$

where $A \in \mathbb{F}^{m\times m}$ and $B \in \mathbb{F}^{n\times n}$ are *fixed* matrices. If the rank of $\nabla(X)$ is r, then $\nabla(X)$ can be written as

$$\nabla(X) = G \cdot H,$$

where $G \in \mathbb{F}^{m\times r}$ and $H \in \mathbb{F}^{r\times n}$.

If ∇ is an isomorphism, then X can be uniquely recovered from $\nabla(X)$, and if in addition $\nabla(X)$ has small rank, then the displacement operator gives a means of *compressing* the matrix X, since $r(m + N)$ entries of the G and H are sufficient to describe the mn entries of X. In this case, G and H are called the *generators* of X. The rank of $\nabla(X)$ is called the *displacement rank* of X (with respect to ∇, or with respect to (A, B)).

The question we will answer in this lecture is whether it is possible to perform the Gaussian elimination algorithm on the generators of X instead of on X directly: note that the complexity $O(m^2n)$ of the Gaussian elimination algorithm for X stems from the fact that the $O(mn)$ entries of X are updated m times. If we work with the generators instead, we might hope to update the $r(m+n)$ entries of the generators m times, thereby obtaining an algorithm with running time $O(rm(m+n))$. If r is significantly less than m and n, then we have saved an order of magnitude in the running time. We will derive such algorithms for certain types of displacement operators in this lecture.

9.2. First Examples

9.2.1. Vandermonde Matrices

Let

$$V_{m,n} := \begin{pmatrix} 1 & x_1 & \cdots & x_1^{n-1} \\ 1 & x_2 & \cdots & x_2^{n-1} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_m & \cdots & x_m^{n-1} \end{pmatrix},$$

and assume that $x_1 \cdots x_m \neq 0$. This Vandermonde matrix has a displacement structure obtained from the upper shift matrix Z_n and a diagonal matrix Δ_m:

$$Z_n := \begin{pmatrix} 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & 1 \\ 0 & 0 & 0 & \cdots & 0 \end{pmatrix} \in \mathbb{F}^{n \times n}, \qquad \Delta_m := \operatorname{diag}(1/x_1, \ldots, 1/x_m).$$

We have

$$\Delta_m V_{m,n} - V_{m,n} Z_n = \begin{pmatrix} 1/x_1 \\ 1/x_2 \\ \vdots \\ 1/x_m \end{pmatrix} \cdot (0, 0, \ldots, 0, 1). \tag{9.1}$$

The displacement rank of V with respect to (Δ_m, Z_n) is thus 1.

9.2.2. Matrices Arising from Sudan's Algorithm

Let $D = \operatorname{diag}(y_1, \ldots, y_n)$. In the last lecture we saw that Sudan's list decoding algorithm gives rise to a system of n equations in $n+1$ variables in which the matrix is given by

$$A := \left(D^\ell V_{n,e} \mid D^{\ell-1} V_{n,e+k-1} \mid \cdots \mid D V_{n,e+(\ell-1)(k-1)} \mid V_{n,e+\ell(k-1)} \right),$$

for some integers e and k. Let Z be the $(n+1) \times (n+1)$-block diagonal matrix with block diagonal entries $Z_e, Z_{e+(k-1)}, \ldots, Z_{e+\ell(k-1)}$. Then we have $\Delta_n A - AZ$ consists of $\ell+1$ blocks with $e, e+k-1, \ldots, e+\ell(k-1)$ columns, respectively, and each of these blocks, all but the last column is zero. The last column in the first block is $(y_1^\ell/x_1, \ldots, y_n^\ell/x_n)^\top$, the last column in the second block is $(y_1^{\ell-1}/x_1, \ldots, y_n^{\ell-1}/x_n)^\top$, and the last column in the last block is $(1/x_1, \ldots, 1/x_n)^\top$. This shows us that

$$\Delta_n A - AZ = \begin{pmatrix} y_1^\ell/x_1 & y_1^{\ell-1}/x_1 & \cdots & 1/x_1 \\ y_2^\ell/x_2 & y_2^{\ell-1}/x_2 & \cdots & 1/x_2 \\ \vdots & \vdots & \ddots & \vdots \\ y_n^\ell/x_n & y_n^{\ell-1}/x_n & \cdots & 1/x_n \end{pmatrix} \cdot H, \tag{9.2}$$

where H is the $(\ell+1) \times (n+1)$-matrix which consists of $\ell+1$ blocks with $e, e+k-1, \ldots, e+\ell(k-1)$ columns, respectively, and each of these blocks has only one nonzero entry. The indices of the nonzero entries in blocks $1, 2, \ldots, \ell+1$ are $(1,1), (2,1), \ldots, (\ell+1, 1)$, respectively, and the value of these entries is 1.

For example, suppose that $e = 3, \ell = 2$, and $k = 4$. Then H is

$$H = \left(\begin{array}{ccc|cccccc|ccccccccc} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{array} \right).$$

The displacement rank of the matrix A with respect to (Δ_n, Z) is $\ell+1$.

9.3. Gaussian Elimination and the PLU-Decomposition

How does Gaussian elimination work?

Suppose that $A \in \mathbb{F}^{m \times n}$ is partitioned as $A = \left(\begin{smallmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{smallmatrix}\right)$, with square matrices A_{11} and A_{22}, and suppose further that A_{11} is invertible. The *Schur-complement* of A with respect to A_{11} is the matrix $A_2 := A_{22} - A_{21} A_{11}^{-1} A_{12}$, and we have the identity

$$\begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix} = \begin{pmatrix} I & 0 \\ A_{21} A_{11}^{-1} & \tilde{I} \end{pmatrix} \begin{pmatrix} A_{11} & 0 \\ 0 & A_2 \end{pmatrix} \begin{pmatrix} I & A_{11}^{-1} A_{12} \\ 0 & \tilde{I} \end{pmatrix},$$

with identity matrices I and $\tilde{I}$ of appropriate sizes. This decomposition of A shows that if $A_{11} =: a$ is a 1×1-matrix, and if we know an LU-decomposition for A_2, say $A_2 = L_2 U_2$, then we can compute an LU decomposition for A as

follows:

$$A = \underbrace{\begin{pmatrix} 1 & 0 \\ A_{21}/a & L_2 \end{pmatrix}}_{=:L} \cdot \underbrace{\begin{pmatrix} a & A_{21} \\ 0 & U_2 \end{pmatrix}}_{=:U}.$$

This is the basis for the Gaussian elimination algorithm, which iteratively computes Schur-complements and an LU-decomposition thereof. By using pivoting, the algorithm additionally produces a permutation matrix Q such that QA has the property that the $(1,1)$-entry of all the Schur-complements are nonzero, and hence the above procedure is applicable. Altogether, the algorithm produces a decomposition $A = PLU$, where $P = Q^{-1}$.

9.4. Iterative Computation of the PLU-Decomposition from the Generators

As the following lemma suggests, we can efficiently compute the Schur complement of a given matrix from the compressed generators of the matrix with respect to a displacement operator.

Lemma 9.1. *Let $A \in \mathbb{F}^{m\times n}$, $\Delta \in \mathbb{F}^{m\times m}$ is upper triangular, $Z \in \mathbb{F}^{n\times n}$ is lower triangular, and that we have the decomposition*

$$G = (g_{11} \mid G_{21}),\ B = (h_{11} \mid H_{12}),\ \Delta = \begin{pmatrix} d_1 & \mathbf{0} \\ \star & D_2 \end{pmatrix}, Z = \begin{pmatrix} z_1 & \star \\ \mathbf{0} & Z_2 \end{pmatrix}, A = \begin{pmatrix} a_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix}.$$

Suppose that a_{11} is nonzero. Then the Schur complement $A_2 := A_{22} - A_{21}a_{11}^{-1}A_{12}$ of A satisfies the displacement equation

$$\Delta_2 A_2 - A_2 Z_2 = G_2 H_2,$$

where $G_2 = G_{21} - a_{11}^{-1}g_{11}A_{12}$ and $H_2 = H_{12} - a_{11}^{-1}h_{11}A_{12}$.

Proof. The proof is straightforward and is left as an exercise. □

It is possible to use the above lemma to design an algorithm for computing a PLU-decomposition of the matrix A. In our applications, we would often like to find an element in the right kernel of A, rather than a full PLU-decomposition. However, it is trivial to compute a nonzero element in the kernel of A once a PLU-decomposition is known, since the kernel of A and that of U conincide, and a nonzero element in the latter can be found using backward substitution.

Algorithm 9.2. *On input a diagonal matrix $D \in \mathbb{F}^{m\times m}$, an upper triangular matrix $Z \in \mathbb{F}^{n\times n}$, and a $\nabla_{D,Z}$-generator (G, H) for $A \in \mathbb{F}^{m\times n}$ in $DA - AZ = GH$, the algorithm outputs a permutation matrix P, a lower triangular matrix $L \in \mathbb{F}^{m\times m}$, and an upper triangular matrix $U \in \mathbb{F}^{m\times n}$, such that $A = PLU$.*

(1) *Recover from the generator the first column of A.*

(2) *Determine the position, say $(k, 1)$, of a nonzero entry of Ax. If it does not exist, then set the first column of L equal to $[1, \mathbf{0}]^\top$ and the first row of U equal to the first row of A, and go to Step (4). Otherwise interchange the first and the k-th diagonal entries of Z and the first and the k-th rows of G and call P_1 the permutation matrix corresponding to this transposition.*

(3) *Recover from the generator the first row of $P_1A =: \left(\begin{smallmatrix} a_{11} & A_{12} \\ A_{21} & A_{22} \end{smallmatrix}\right)$. Store $[1, A_{12}/a_{11}]^\top$ as the first column of L and $[a_{11}, A_{12}]$ as the first row of U.*

(4) *If $a_{11} \neq 0$, compute by Lemma 9.1 a generator of the Schur complement A_2 of P_1A. If $a_{11} = 0$, then set $A_2 := A_{22}$, $G_2 := G_{21}$, and $H_2 := H_{12}$.*

(5) *Proceed recursively with A_2 which is now represented by its generator (G_2, H_2) to finally obtain the factorization $A = PLU$, where $P = P_1 \cdots P_\mu$ with P_k being the permutation used at the k-th step of the recursion and $\mu = \min\{m, n\}$.*

The correctness of the above algorithm and its running time depend on steps (1) and (3). Note that it may not be possible to recover the first row and column of A from the matrices D, Z, G, H. We will explore these issues later in Section 9.5..

For now, we focus on developing a more memory efficient version of this algorithm for certain displacement structures. For this, the following result will be crucial.

Lemma 9.3. *Let $A \in \mathbb{F}^{m\times n}$ be partitioned as*

$$A = \begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix},$$

for some $1 \leq i < m$, where $A_{11} \in \mathbb{F}^{i\times i}$, $A_{12} \in \mathbb{F}^{i\times(n-i)}$, $A_{21} \in \mathbb{F}^{(m-i)\times i}$, and $A_{22} \in \mathbb{F}^{(m-i)\times(m-i)}$, and suppose that A_{11} is invertible. Further, let $\tilde{A} := \binom{A}{I_n}$, where I_n is the $n \times n$-identity matrix, and denote by $c = (c_1, \ldots, c_{m+n-i})^\top$ the first column of the Schur complement of $\tilde{A}$ with respect to A_{11}. Suppose that $c_1 = \cdots = c_{m-i} = 0$. Then the vector $(c_{m-i+1}, \ldots, c_m, 1, 0, \ldots, 0)^\top$ is in the right kernel of the matrix A.

Proof. The Schur complement of $\tilde{A}$ with respect to A_{11} is easily seen to be

$$\begin{pmatrix} A_{22} - A_{21}A_{11}^{-1}A_{12} \\ -A_{11}^{-1}A_{12} \\ I_{n-i} \end{pmatrix}.$$

Note that

$$A \cdot \begin{pmatrix} -A_{11}^{-1}A_{12} \\ I_{n-i} \end{pmatrix} = \begin{pmatrix} 0 \\ A_{22} - A_{11}^{-1}A_{12} \end{pmatrix}.$$

By assumption, the first column of the matrix on the right is zero, which implies the assertion. □

Suppose now that A has low displacement rank with respect to $\nabla_{D,Z}$ and suppose further that B is a matrix such that $B - Z$ has low rank:

$$DA - AZ = G_1H_1, \quad C - Z = G_2H_2.$$

Then we have

$$\begin{pmatrix} D & 0 \\ 0 & C \end{pmatrix} \cdot \begin{pmatrix} A \\ I_n \end{pmatrix} - \begin{pmatrix} A \\ I_n \end{pmatrix} Z = \begin{pmatrix} G_1 & 0 \\ 0 & G_2 \end{pmatrix} \cdot \begin{pmatrix} H_1 \\ H_2 \end{pmatrix}. \tag{9.3}$$

Algorithm 9.2 can now be customized in the following way.

Algorithm 9.4. *On input a diagonal matrix $D \in \mathbb{F}^{m\times m}$, an upper triangular matrix $Z \in \mathbb{F}^{n\times n}$, a matrix $A \in \mathbb{F}^{n\times n}$, and a $\nabla_{C,Z}$-generator (G, H) for $W = \binom{A}{I_n} \in \mathbb{F}^{(m+n)\times n}$ in $CW - WZ = GH$, where $C = \begin{pmatrix} D & 0 \\ 0 & A \end{pmatrix}$, the algorithm outputs a vector $x = (x_1, \ldots, x_i, 1, 0, \ldots, 0)^\top$ such that $Ax = 0$.*

(0) *Set $i := 0$.*

(1) *Recover from the generators the first column of W.*

(2) *Determine the position, say $(k, 1)$, of a nonzero entry of W. If k does not exist, or $k > m-i$, then go to Step (6). Otherwise interchange the first and the k-th diagonal entries of D, the first and the k-th rows of G, the first and the k-th entries of the first column of W, denoting the new matrix by W again.*

(3) *Recover from the generators the first row of W.*

(4) *Using the first row and the first column of W, compute by Lemma 9.1 a generator of the Schur complement W_2 of W.*

(5) *Proceed recursively with A_2 which is now represented by its generator (G_2, H_2), increase i by 1, and go back to Step (1).*

(6) *Output the vector $(x_1, \ldots, x_i, 1, 0, \ldots, 0)^\top$ where $x_1, \ldots, x_i$ are the $m - i + 1$-st through m-th entries of the first column of W.*

As was pointed out before, the correctness of the above algorithm and its running time depend on steps (1) and (3). Note that it may not be possible to recover the first row and column of W from the matrices D, B, G, H, Z. In fact, recovery from these data alone is only possible if $\nabla = \nabla_{C,Z}$ is an isomorphism. For simplicity we assume in the following that this is the case. In the general case one has to augment the (D, B, G, H, Z) by more data corresponding to the kernel of ∇, but we will not give details.

If ∇ is an isomorphism, then the algorithm has the correct output. Indeed, using Lemma 9.1, we see that at step (5), W_2 is the Schur-complement of the matrix PW with respect to the prinicipal $i\times i$-minor, where P is a permutation matrix (obtained from the pivoting transpositions in Step (2)). Once the algorithm reaches Step (6), the conditions of Lemma 9.3 are satisfied and the algorithm outputs the correct vector.

The running time of the algorithm is summarized in the following.

Lemma 9.5. *Suppose that steps* (1) *and* (3) *of Algorithm 9.4 run in time* $O(rm)$ *and* $O(rn)$*, respectively. Then the total running time of that algorithm is* $O(rmn)$*, where* r *is the displacement rank of* A *with respect to* $\nabla_{D,A}$*.*

Proof. The proof is obvious once one realizes that Step (4) runs in time $O(r(m+n))$, and that the algorithm is performed recursively at most $\min\{m,n\}$ times. □

9.5. Computing the First Row and the First Column

A major ingredient of Algorithm 9.4 is that of computing the first row and the first column of a matrix from its generators. The computational cost of this step depends greatly on the underlying displacement structure. In this section, we will present a solution to this problem in a special case, namely, in the case where the matrix $W \in \mathbb{F}^{(m+n)\times n}$ has the displacement structure

$$\begin{pmatrix} D & 0 \\ 0 & C \end{pmatrix} W - WZ = GH \tag{9.4}$$

where D is diagonal and $Z, C \in \mathbb{F}^{n\times n}$ are given by

$$Z := \begin{pmatrix} 0 & 1 & 0 & \cdots & 0 & 0 \\ 0 & 0 & 1 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & 1 & 0 \\ 0 & 0 & 0 & \cdots & 0 & 1 \\ 0 & 0 & 0 & \cdots & 0 & 0 \end{pmatrix}, \quad C := \begin{pmatrix} 0 & 1 & 0 & \cdots & 0 & 0 \\ 0 & 0 & 1 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & 1 & 0 \\ 0 & 0 & 0 & \cdots & 0 & 1 \\ 1 & 0 & 0 & \cdots & 0 & 0 \end{pmatrix}. \tag{9.5}$$

This case will be prototypical for our later applications. For the sake of simplicity, we will assume that all the diagonal entries of D are nonzero.

The attentive reader might have noticed that the matrices we use for our displacement operator do not satisfy the conditions of Lemma 9.1: the matrix C is not lower triangular, and hence in (9.4) the matrix on the far left will not be lower triangular. How will we be able to use Lemma 9.1, then? The answer is the following: we will use that lemma only to compute the Schur complement of the matrix W with respect to its upper $m \times m$-submatrix, at most. Details are provided in Algorithm 9.8.

Another question an attentive reader may have is why we did not choose C to be Z? In that case the rank of $C - Z$ is zero, and we can easily apply the remark preceding Algorithm 9.4. The answer to this question is also simple: if we had done that, would not be able to recover the first row and columns of W from the generators.

Algorithm 9.6. *On input a diagonal matrix* D *with nonzero diagonal entries* $x_1, \ldots, x_m$*, a matrix* $G = (G_{ij}) \in \mathbb{F}^{m\times r}$*, and a matrix* $H = (H_{ij}) \in \mathbb{F}^{r\times n}$*, this algorithm computes the first row and the first column of the matrix* $W \in \mathbb{F}^{(m+n)\times n}$ *which satisfies the displacement equation* (9.4)*.*

(1) *Compute the first row* $(r_1, \ldots, r_n)$ *and the first column* $(\gamma_1, \ldots, \gamma_m, \gamma_{m+1}, \ldots, \gamma_{m+n})^\top$ *of* GH*.*

(2) *For* $i = 1, \ldots, m$ *set* $v_i := \gamma_i / x_i$*.*

(3) *For* $i = m+1, \ldots, m+n-1$ *set* $v_{i+1} := \gamma_i$*.*

(4) *Set* $v_{m+1} := \gamma_{m+n}$*.*

(5) *Set* $c_1 := v_1$*. For* $i = 2, \ldots, n$ *set* $c_i := (r_i + c_{i-1})/x_i$*.*

(4) *Output row* $(c_1, \ldots, c_n)$ *and column* $(v_1, \ldots, v_m, v_{m+1}, \ldots, v_{m+n})^\top$*.*

Proposition 9.7. *The above algorithm correctly computes its output with* $O(r(m+n))$ *operations in the field* $\mathbb{F}$*.*

Proof. Correctness of the algorithm follows immediately from the following observation: let $(c_1, \ldots, c_n)$ and

$$(c_1, v_2, \ldots, v_m, v_{m+1}, \ldots, v_{m+n})^\top$$

be the first row and the first column of W, respectively. Then

$$\begin{pmatrix} D & 0 \\ 0 & A \end{pmatrix} W - WZ = \begin{pmatrix} x_1c_1 & x_1c_2 - c_1 & \cdots & x_1c_n - c_{n-1} \\ x_2v_2 & \star & \cdots & \star \\ \vdots & \vdots & \ddots & \vdots \\ x_mv_m & \star & \cdots & \star \\ v_{m+2} & \star & \cdots & \star \\ \vdots & \vdots & \cdots & \vdots \\ v_{m+n} & \star & \ddots & \star \\ v_{m+1} & \star & \cdots & \star \end{pmatrix},$$

where the $\star$'s denote elements in $\mathbb{F}$. As for the running time, observe first that computing the first row and first column of GH requires at most $r(m+n) + rn$ operations over $\mathbb{F}$. Step (2) can be done using m operations, steps (3) and (4) are free of charge, and Step (5) requires $2n$ operations. □

We can now combine all the results obtained so far to develop an efficient algorithm for computing a nonzero element in the kernel of a matrix $A \in \mathbb{F}^{m\times n}$ given indirectly as a solution to the displacement equation

$$DA - AZ = GH. \tag{9.6}$$

Here, D is a diagonal matrix, Z is an upper-shift matrix of format n, $G \in \mathbb{F}^{m\times r}$, and $H \in \mathbb{F}^{r\times n}$. Since we will use Algorithm 9.4, we need to have a displacement structure for $W = \binom{A}{I_n}$. It is given by

$$\begin{pmatrix} D & 0 \\ 0 & C \end{pmatrix} \binom{A}{I_n} - \binom{A}{I_n} Z = \binom{G}{E} H, \tag{9.7}$$

where C is defined as in (9.5) and $E = C - Z$. We assume that all the diagonal entries of D are non-zero.

Algorithm 9.8. *On input integers m, n, $m < n$, a diagonal matrix $D \in \mathbb{F}^{m\times m}$ all of whose diagonal entries $x_1, \ldots, x_m$ are nonzero, a matrix $G \in \mathbb{F}^{m\times r}$, and a matrix $H \in \mathbb{F}^{r\times n}$, the algorithm computes a nonzero vector $v \in \mathbb{F}^n$ in the kernel of the matrix A given by the displacement equation* (9.6).

(1) *Set $G_1 := \binom{G}{C-Z}$, $H_1 := H$, and $D_1 := D$, where C is given in* (9.5).

(2) *For $i = 1, \ldots, m$ do*

   (a) *Let D_i be the diagonal matrix with diagonal entries $x_i, \ldots, x_m$. Use Algorithm 9.6 with input D_i, G_i, and H_i to compute the column $c = (c_1, \ldots, c_{m+n-i+1})^\top$.*

   (b) *If $c_1 = \cdots = c_{m-i} = 0$, then* **output** $(c_{m-i+1}, \ldots, c_m, 1, 0, \ldots, 0)^\top$ *and* **stop**. *Otherwise, find an integer $k \le m - i$ such that $c_k \neq 0$. Interchange rows k and 1 of c and of G_i, interchange the first and the kth diagonal entries of D_i.*

   (c) *Use Algorithm 9.6 with input D_i, G_i and H_i to compute the row $r = (r_1, \ldots, r_{n-i+1})$.*

   (d) *For $j = 2, \ldots, m + n - i + 1$ replace row j of G_i by $-c_j/c_1$ times the first row plus the j-th row. Set G_{i+1} as the matrix formed from G_i by deleting the first row.*

   (e) *For $j = 2, \ldots, n - i + 1$ replace the j-th column of H_i by $-r_j/c_1$ times the first column plus the j-th column. Set H_{i+1} as the matrix formed from H_i by deleting the first column.*

Theorem 9.9. *The above algorithm correctly computes its output with $O(rmn)$ operations over the field $\mathbb{F}$.*

Proof. The correctness of the algorithm follows from Lemma 9.3 and Proposition 9.7. The analysis of the running time of the algorithm is straight-forward: Step (2a) runs in time $O(r(m+n-i))$ by Proposition 9.7. The same is true for steps (2d) and (2e). Hence, in the worst case, the running time will be $\sum_{i=1}^{m-1} O(r(m+n-i+1)) = O(rmn)$, since $m < n$. □

The memory requirements for this algorithm can be reduced by observing that the last $n - i$ rows of the matrix G_i are always known. As a result, the matrix G_i needs to store only mr elements (instead of $(m+n)r$). Further, it is easy to see that for the vector c computed in Step (2a), we have $c_{m+1} = 1, c_{m+2} = \cdots = c_{m+n-i} = 0$, hence, we do not need to store these results. Combining these observations, one can easily design an algorithm that needs storage for two matrices of sizes $m \times r$ and $r \times n$, respectively, and storage for three vectors of size n.

Lecture 10
Algebraic Geometric Codes I

In this lecture we will give a brief introduction to codes from algebraic geometry.

10.1. Introduction

We continue with our general discussion of evaluation codes. We saw that Reed-Solomon codes are evaluation codes that have the best possible minimum distance given their block length and their dimension. The drawback, however, is that they are restricted in their length: the block length is at most equal to the size of the field. In particular, it is not possible to get sequences of Reed-Solomon codes for which the length goes to infinity, while the field size is fixed.

One of the ways to overcome this difficulty was to look at Reed-Muller codes. For these codes the block length can be as large as desired. However, we saw that as we increase the block length, either the relative minimum distance, or the rate, converge to 0. In particular, we cannot obtain *asymptotically good* codes, i.e., sequences of codes for which the relative distance and the rate converge to nonzero values.

The new idea that helps overcome this difficulty is due to V.D. Goppa, from 1981, though our presentation differs in part substantially with the one provided in the established literature on this subject.

Looking back at Reed-Solomon codes, the main reason why they yielded good codes was our control on the number of zeros of the code. The reason why Reed-Muller codes are not as good is that the evaluation of an m-variate polynomial on $\mathbb{F}_q^m$ will lead to a large number of zeros. Would it be possible, though, to evaluate the polynomial on a subset of $\mathbb{F}_q^m$ as to gain better control on the number of zeros? For this we need a tool that limits the number of zeros of such an evaluation. We are going to describe the tool first in the most familiar setting, that of the 2-dimensional plane.

Suppose that $m = 2$, i.e., we want to evaluate a bivariate polynomial $h(x, y) \in \mathbb{F}_q[x, y]$ on a subset of the plane $\mathbb{F}_q^2$. Suppose that this subset is the set of zeros of a bivariate polynomial $f(x, y) \in \mathbb{F}_q[x, y]$. Do we have any chance to obtain an upper bound on the number of zeros of $h(x, y)$ on the set of zeros of f?

We can phrase this question in yet a different way: the zeroset of $f(x, y)$ is a *curve* in $\mathbb{F}_q^2$, as is the zeroset of $h(x, y)$. So, we are asking how many intersection points these two curves will have at most. Figure 10.1 gives an example of such a setting in the plane $\mathbb{R}^2$, where $h(x, y) = x^3 - 3xy + y^3 - 1.8$ and $f(x, y) = y^2 - x^3 + 2x - 2$. In this case, the number of intersection points is 5.

10.2. Irreducible Curves and Bézout's Theorem

In general, nothing much can be said about the number of intersection points of $h(x, y)$ and $f(x, y)$. In fact, over an infinite field, this number can be infinity. For example, suppose that $f(x, y) = (x - y)(x + y)$ and $h(x, y) = (x - y)(x^2 + y^2 - 1)$. In this case, both $f(x, y) = 0$ and $h(x, y) = 0$ have the *component* $x - y = 0$ in common, and over an infinite field, this component has infinitely many points (and over the field $\mathbb{F}_q$ it has q points).

In our applications, $f(x, y)$ will be a fixed curve, and we will evaluate polynomials $h(x, y)$ of various degrees on it. There is no guarantee, a priori, that $h(x, y)$ and $f(x, y)$ have no components in common, because we have little control over $h(x, y)$. However, if we assume that $f(x, y)$ is *irreducible*, i.e., that it cannot be factored in a nontrivial

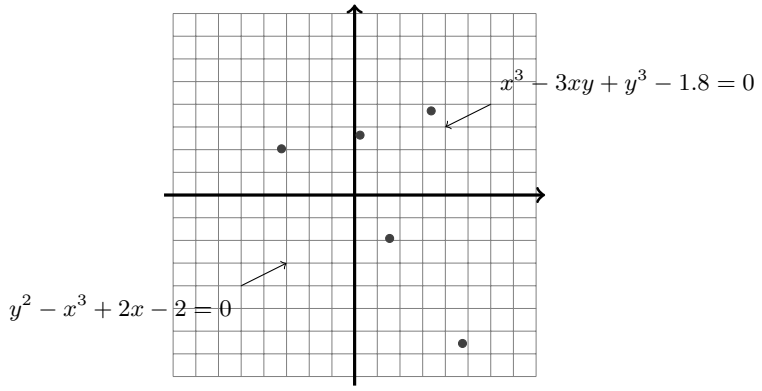


Figure 10.1: The curves given by $x^3 - 3xy + y^3 - 1.8 = 0$ and $y^2 - x^3 + 2x - 2 = 0$ intersect at 5 points in the real plane. By Bézout's theorem, the number of intersection points is at most 9.

way, then we can say a little bit more. Namely, in this case, $h(x, y)$ and $f(x, y)$ have a common factor iff h is a multiple of f. If this is not the case, i.e., if f is irreducible and h is not a multiple of f, then we can bound the number of intersection points of f and h.

For this, we need the concept of *degree*. The degree $\deg(g)$ of a polynomial $g \in \mathbb{F}_q[x, y]$ is the largest degree of a monomial in g. For example, the degree of $x^3y + xy + 3x - y$ is 4, since the degree of the monomial x^3y is 4. We have the following theorem.

Theorem 10.1 (Bézout's Theorem in the plane)**.** *Suppose that $f(x, y), h(x, y) \in \mathbb{F}_q[x, y]$, that $f(x, y)$ is irreducible, and that $h(x, y)$ is not a multiple of $f(x, y)$. Then the number of points $(a, b) \in \mathbb{F}_q^2$ such that $f(a, b) = h(a, b) = 0$ is at most $\deg(f) \deg(h)$.*

For a proof of this theorem, we refer the reader to the book *Algebraic Geometry* by Joe Harris, or to the book *Algebraic Complexity Theory* by Bürgisser et al., or to the book *Ideals, Varieties, and Algorithms* by Cox et al.

Let us check the theorem in the situation of Figure 10.1: here, the degree of $f(x, y) = y^2 - x^3 + 2x - 2$ is 3, and the degree of $h(x, y) = x^3 - 3xy + y^3 - 1.8$ is also 3. The polynomial $f(x, y)$ is irreducible (without proof), and $h(x, y)$ is not a multiple of $f(x, y)$. Hence, Bézout's theorem implies that the number of intersection points is at most 9. (In fact, the theorem bounds the number even over the field $\mathbb{C}$ of complex numbers. The reader is invited to show that over this field the number of intersection points is exactly 9.)

10.3. The Space of Functions

Plane AG codes are evaluation codes in which we evaluate certain bivariate polynomials on a subset of $\mathbb{F}_q^2$. Based on the discussions in the last sections, the subset we will be considering will be the zeroset of an irreducible bivariate polynomial $f(x, y) \in \mathbb{F}_q[x, y]$. The polynomials we want to evaluate are bivariate polynomials that are not multiples of $f(x, y)$. Our first goal is to find a basis for the vector space of such polynomials.

Let $R := \mathbb{F}_q[x, y]/(f(x, y)\mathbb{F}_q[x, y])$ be the residue class ring modulo $f(x, y)$. The polynomials we want to evaluate are nonzero elements of R. To find a basis for R over $\mathbb{F}_q$, we make the following simplifying assumption. If d denotes the degree of f, we assume that $f(x, y) = ax^d + f_1(x, y)$, wherein the x-degree of f_1 is smaller than d, and a is nonzero. We can always assume w.l.o.g. that f is of this form. Otherwise, if $f(x, y)$ has a monomial of the form $x^{d-k}y^k$, then we replace y by $y + x$, and obtain the desired form for f. This only results in a change of basis in $\mathbb{F}_q^2$ and does not change anything else. In particular, after this transformation, the new f is irreducible as well.

Theorem 10.2. *Suppose that $f \in \mathbb{F}_q[x, y]$ is of degree d, and that $f = ax^d + f_1(x, y)$, where the degree in x of $f_1(x, y)$ is smaller than d and a is nonzero. Then for any polynomial $h \in \mathbb{F}_q[x, y]$ there is a unique element h_1 in*

$$\Gamma := \mathbb{F}_q[y] \oplus x\mathbb{F}_q[y] \oplus \cdots \oplus x^{d-1}\mathbb{F}_q[y],$$

such that $h - h_1$ is a multiple of f.

Proof. The assertion is very easily seen, when we consider all the polynomials as polynomials in x over $\mathbb{F} := \mathbb{F}_q(y)$.

First, we show that if for $h, g \in \Gamma$ we have that $h - g$ is divisible by f, then $h = g$. Considering these polynomials and f as polynomial in $\mathbb{F}[x]$, we see that their x-degree of $g - h$ is smaller than that of f, and hence f divides $g - h$ iff $g = h$.

Next, take any $h \in \mathbb{F}_q[x,y]$, interpret it as a polynomial in $\mathbb{F}[x]$, and perform polynomial division with remainder by f to obtain a remainder h_1. Since the "leading coefficient" of f (in x) is in $\mathbb{F}_q$, h_1 is an element of Γ (i.e., we never divide by a polynomial in $\mathbb{F}_q[y]$).

Putting everything together, any h has an h_1 in Γ, and this h_1 is unique since otherwise the difference of two non-equal elements in Γ would be divisible by f. □

The polynomials we would like to evaluate on the zeroset of $f(x,y)$ will have to come from Γ. We need to have a control on their degree, which, by Bézout's theorem, gives us control on the number of zeros of nonzero codewords, and hence on the minimum distance of the code. For this, we need to calculate the dimension of $\Gamma_{<m}$, the space of polynomials in Γ of degree less than m.

Theorem 10.3. *For $m \geq d-1$, the dimension of the space of polynomials of degree $< m$ in Γ, called $\Gamma_{<m}$, is*

$$(m-1)d - \frac{(d-1)(d-2)}{2} + 1.$$

Proof. Note that we have

$$\Gamma_{<m} = \mathbb{F}_q[y]_{<m} \oplus x\mathbb{F}_q[y]_{<m-1} \oplus \cdots \oplus x^{d-1}\mathbb{F}_q[y]_{<m-d+1}.$$

This shows that if $m \geq d$, then the dimension of $\Gamma_{<m}$ is

$$\begin{aligned} \sum_{i=0}^{d-1}(m-i) &= md - \frac{d(d-1)}{2} \\ &= (m-1)d - \frac{(d-1)(d-2)}{2} + 1, \end{aligned}$$

and proves our assertion. □

10.4. AG Codes

Now we are ready to define plane AG codes. These are evaluation codes via $(M, S; \mathbb{F}_q)$.

We choose an irreducible polynomial $f(x,y) \in \mathbb{F}_q[x,y]$ of degree d, $f(x,y) = ax^d + f_1(x,y)$ with x-degree of f_1 smaller than d, and $a \neq 0$. The evaluation set S is equal to $\{(a,b) \in \mathbb{F}_q^2 \mid f(a,b) = 0\}$.

The space of functions M is equal to $\Gamma_{<m}$, and the evaluation is simply the evaluation of a polynomial at a point. The code obtained this way is called a *geometric Goppa code* or an *AG code*. We denote it by $C(f,m)$.

Theorem 10.4. *Let n be the number of $(a,b) \in \mathbb{F}_q^2$ such that $f(a,b) = 0$. If the degree of f is d, $m \geq d$, and $n > (m-1)d$, then*

$$\dim C(f,m) = (m-1)d - \frac{(d-1)(d-2)}{2} + 1,$$

and the minimum distance of $C(f,m)$ is at least $n - (m-1)d$.

Proof. If $h \in \Gamma_{<m}$ is nonzero, then by Bézout's Theorem the number of (a,b) such that $h(a,b) = f(a,b) = 0$ is at most $(m-1)d$. Therefore, the image of the evaluation map applied to f has at most $(m-1)d$ zeros, and thus at least $n - (m-1)d$ nonzeros. If $n > (m-1)d$, then the number of nonzeros is positive, and hence the evaluation map is injective. As a result, the dimension of $C(f,m)$ equals that of $\Gamma_{<m}$, which was determined in Theorem 10.3. By the same token, the minimum distance is at least $n - (m-1)d$. □

The AG codes we obtain are $[n,k,d]_q$-codes with

$$k + d \geq n - \frac{(d-1)(d-2)}{2} + 1.$$

On the other hand, the Singleton bound asserts that $k + d \leq n + 1$, so that we can think of the term $(d-1)(d-2)/2$ as some kind of a *defect* of the code.

In order to obtain good AG codes, what we have to do is to find polynomials $f(x,y)$ that

1. are irreducible;
2. have a lot of zeros in $\mathbb{F}_q^2$, and

3. do not have a large degree.

Unfortunately, the second and the third goal above compete with one another: a deep theorem in algebraic geometry, called the *Hasse-Weil Bound*, asserts that the number of zeros of f is at most $q + (d-1)(d-2)\sqrt{q}$.

Even though we will be able to exhibit polynomials that achieve this bound, one should note that this approach will not yield asymptotically good codes, since the number of zeros of $f(x, y)$ will never exceed q^2. In the next lecture, we will show how to overcome this difficulty. For now, we will be content with some examples of plane AG codes.

10.5. Eisenstein's Irreducibility Criterion

In order to apply our results to specific examples, we need to test whether a given polynomial $f(x, y)$ is irreducible. There are a variety of sufficient conditions for irreducibility, of which we highlight one, called the *Eisenstein Criterion*.

First, let us discuss a trivial case in which $f(x, y)$ is reducible. We write $f(x, y) = \sum_{i=0}^{m} f_i(x)y^i$. Suppose that $f_0(x), \ldots, f_m(x)$ have a nontrivial gcd, say $h(x)$ of degree larger than 0. Then $f(x, y) = h(x)f_1(x, y)$ for some other polynomial $f_1(x, y)$, and $f(x, y)$ becomes reducible.

We say that a polynomial $f(x, y)$ is primitive (with respect to x), if the gcd $h(x)$ is trivial, i.e., if the $f_0(x), \ldots, f_m(x)$ are co-prime. Clearly, if we want $f(x, y)$ to be irreducible, we definitely want it to be primitive.

Theorem 10.5. *Suppose that $f(x, y) = \sum_{i=0}^{m} f_i(x)y^i$ is primitive with respect to x, and that there is an irreducible polynomial $p(x) \in \mathbb{F}_q[x]$ such that $p(x)$ divides $f_0(x), f_1(x), \ldots, f_{m-1}(x)$, but $p(x)$ does not divide $f_m(x)$, and $p^2(x)$ does not divide $f_0(x)$. Then $f(x, y)$ is irreducible.*

Proof. Suppose that $f(x, y)$ is reducible. Then

$$f(x, y) = \left(\sum_{i=0}^{\ell} h_i(x)y^i\right) \cdot \left(\sum_{j=0}^{m-\ell} g_j(x)y^j\right),$$

with $h_\ell(x)$ and $g_{m-\ell}(x)$ nonzero. Since $f(x, y)$ is supposed to be primitive with respect to x, both ℓ and $m - \ell$ are larger than 0. We have:

$$f_0(x) \quad = \quad h_0(x)g_0(x)$$

Since $p(x)$ is irreducible, it divides $f_0(x)$, and $p(x)^2$ does not divide $f_0(x)$, this polynomial either divides $h_0(x)$ or $g_0(x)$, but not both. W.l.o.g. assume that it divides $h_0(x)$. Then, since

$$f_1(x) \quad = \quad h_0(x)g_1(x) + h_1(x)g_0(x)$$

$p(x)$ also divides $h_1(x)$, since it divides $f_1(x)$ by assumption, and it does not divide $g_0(x)$. Continuing this way, we see that since

$$f_\ell(x) \quad = \quad h_0(x)g_\ell(x) + \cdots + h_\ell(x)g_0(x),$$

and $\ell < m$, $p(x)$ divides $f_\ell(x)$, and it divides $h_0(x), \ldots, h_{\ell-1}(x)$, by induction. Since it does not divide $g_0(x)$, it must divide $h_\ell(x)$. But

$$f_m(x) = h_\ell(x)g_{m-\ell}(x),$$

which shows that $p(x)$ divides $f_m(x)$, a contradiction. This shows that $f(x)$ is irreducible. □

10.6. Examples

10.6.1. Elliptic Curves

Consider the polynomial $x^2 + x - y^3 \in \mathbb{F}_2[x, y]$. Eisenstein's criterion shows that it is irreducible: consider the polynomial $p(x) = x$. It divides $x^2 + x$, but x^2 does not divide $x^2 + x$. It divides the coefficients of y and y^2 of the polynomial (both these coefficients are zero), but it does not divide the coefficient of y^3 which is 1. Moreover, the polynomial is primitive.

What kind of a code do we obtain from this? The number of zeros of this polynomial is 2: the zeros consist of the points $(0, 0)$, $(1, 0)$, so this will hardly give us a reasonable code over $\mathbb{F}_2$.

However, if we change the field to $\mathbb{F}_4$, something remarkable happens. Suppose that $\mathbb{F}_4 = \{0, 1, \omega, \omega^2\}$, where $\omega^2 + \omega + 1 = 0$. Then the zeros of f consist of the 8 points

$$\begin{array}{ccc} (0,0) & (1,0) & \\ (\omega,1) & (\omega,\omega) & (\omega,\omega^2) \\ (\omega^2,1) & (\omega^2,\omega) & (\omega^2,\omega^2) \end{array}$$

This gives us $[8, 3(m-1), 8-3(m-1)]_4$-codes for $m \geq 2$. In other words, we obtain an $[8, 3, 5]_4$-code and an $[8, 6, 2]_4$-code.

10.6.2. Hermitian Curves

This example is a generalization of the last one. Consider the field $\mathbb{F}_{q^2}$, and the polynomial

$$f(x, y) = x^q + x - y^{q+1}.$$

The zeroset of this polynomial is called a *Hermitian Curve*. In the same way as above, we can show that $f(x, y)$ is irreducible.

How many zeros does f have? We use the fact that the mapping $x \mapsto x^q + x$ is the *trace* of $\mathbb{F}_{q^2}$ over $\mathbb{F}_q$, and that the mapping $y \mapsto y^{q+1}$ is the *norm*, which means that it maps $\mathbb{F}_{q^2}$ to $\mathbb{F}_q$. This implies that for any $y \in \mathbb{F}_{q^2}$, there are exactly q values of x such that $x^q + x = y^{q+1}$: given y, the value of y^{q+1}, which is the norm of y, is in $\mathbb{F}_q$, and hence, since the trace map is surjective, there are exactly q values for x such that $x^q + x = y^{q+1}$.

The degree of f is $q+1$, so putting things together, we see that for any $m \geq q$ and $(m-1)(q+1) < q^3$ we obtain a code with parameters

$$\left[q^3, (m-1)(q+1) - \frac{q(q-1)}{2} + 1, \geq q^3 - (m-1)(q+1)\right]_{q^2}.$$

Note that a Reed-Solomon code would yield codes of length at most q^2, so the length of the codes we have constructed is considerably larger than that of Reed-Solomon codes, while for large q the relative minimum distance and the rate of these codes behaves almost like those of an MDS-code. (The rate is roughly $m/q^2 + O(1/q)$ and the relative minimum distance is roughly $1 - m/q^2$.)

Lecture 11

AG Codes II

In the last lecture we saw how to construct AG codes defined as evaluation of bivariate polynomials on a curve in the plane $\mathbb{F}_q^2$. The block length of such codes does not exceed q^2, so for constructing longer codes, we need to generalize the construction. This is going to be the topic of this lecture.

11.1. Irreducible Curves and Bézout's Theorem

For what follows, the following informal definition of a curve is sufficient: a curve in $\mathbb{F}_q^m$ is the common zeroset of $m-1$ nonconstant polynomials $f_1(x_1,x_2), f_2(x_2,x_3), \ldots, f_{m-1}(x_{m-1},x_m) \in \mathbb{F}_q[x_1,\ldots,x_m]$. The *degree* of this curve is defined as the product of the total degrees of $f_1, f_2, \ldots, f_{m-1}$. The curve is called *irreducible* if all the f_i are irreducible polynomials.

Crucial for the construction of AG codes in this more general setting is the following theorem of Bézout:

Theorem 11.1 (Bézout's Theorem)**.** *Suppose that the curve given by*

$$\begin{aligned} f_1(x_1,x_2) &= 0 \\ f_2(x_2,x_3) &= 0 \\ \vdots &\quad \vdots \\ f_{t-1}(x_{t-1},x_t) &= 0 \end{aligned}$$

is irreducible, and that $g \in \mathbb{F}_q[x_1,\ldots,x_t]$ is a polynomial that cannot be expressed in the form $h_1f_1+\cdots+h_{t-1}f_{t-1}$ for $h_1,\ldots,h_{t-1} \in \mathbb{F}_q[x_1,\ldots,x_t]$. Then the number of intersection points of the zeroset of g and the curve is at most $\deg(g)\deg(f_1)\cdots\deg(f_{t-1})$.

A proof of a more general version of this theorem can be found, e.g., in Chapter 1 of "Algebraic Geometry" by Hartshorne, or in Chapter 8 of "Algebraic Complexity Theory" by Bürgisser et al.

It remains to be show how to test whether a polynomial $g \in \mathbb{F}_q[x_1,\ldots,x_m]$ has the property that it cannot be expressed in the form $h_1f_1+\cdots+h_{m-1}f_{m-1}$ for $h_1,\ldots,h_{m-1} \in \mathbb{F}_q[x_1,\ldots,x_m]$. As in the last section, we can assume w.l.o.g. that each f_i is of the form $ax_{i+1}^{d_{i+1}}+h_i$, wherein $\deg_{x_{i+1}}(h_i) < d_{i+1}$, and d_{i+1} is the total degree of f_i. Consider the vector space

$$\Gamma := \bigoplus_{i_2=0}^{d_2-1}\bigoplus_{i_3=0}^{d_3-1}\cdots\bigoplus_{i_t=0}^{d_t-1}\mathbb{F}_q[x_1]x_2^{i_2}\cdots x_t^{i_t}.$$

If $g \in \Gamma$ and it is nonzero, then it cannot be represented as $h_1f_1+\cdots h_{t-1}f_{t-1}$, since the local degrees of g in $x_2,\ldots,x_t$ is smaller than the maximum of the local degrees of the f_j in these variables. Moreover, any $g \in \mathbb{F}_q[x_1,\ldots,x_t]$ has the property that there exist $h_1,\ldots,h_{t-1}$ such that $g+h_1f_1+\cdots+h_{t-1}f_{t-1} \in \Gamma$.

For $m > (t-1)q$, let

$$\Gamma_{<m} := \bigoplus_{i_2=0}^{d_2-1} \bigoplus_{i_3=0}^{d_3-1} \cdots \bigoplus_{i_t=0}^{d_t-1} \mathbb{F}_q[x_1]_{m-(i_2+\cdots+i_t)} x_2^{i_2} \cdots x_t^{i_t}.$$

Then any nonzero $g \in \Gamma_{<m}$ satisfies the property of Bézout's Theorem. Moreover, the dimension of $\Gamma_{<m}$ is

$$\sum_{i_2=0}^{d_2-1} \cdots \sum_{i_t=0}^{d_t-1} m - (i_2 + \cdots + i_t) = (m-1)D - D\frac{d_2 + \cdots + d_t - t - 1}{2},$$

where $D = d_2 \cdots d_t$.

Suppose now that $g \in \Gamma_{<m}$, that X has n points, and that $d_2 \cdots d_t(m-1) < n$, and consider the evaluation code via $(\Gamma_{<m}, X; \mathbb{F}_q)$. By Bézout's theorem, we obtain a code with parameters

$$\left[n, (m-1)D - D\frac{\sum_2^t d_t - t - 1}{2}, n - (m-1)D \right]_q. \tag{11.1}$$

11.2. Example: Hermitian Towers

Consider the curve X in $\mathbb{F}_{q^2}^3$ given by

$$\begin{aligned} y^{q+1} &= x^q - x \\ x^{q+1} &= z^q - z. \end{aligned}$$

The number of points of this curve, i.e., the number of $(x, y, z) \in \mathbb{F}_{q^2}^3$ satisfying the above equations, is q^4: For each of the q^2 possible values for y, there are q values for x, and for each of these x-values there are q values for z. Each of the equations in the curve is irreducible (by Eisenstein's Criterion) and hence the curve is irreducible. Applying the machinery of the last section with $d_2 = d_3 = q+1$, we obtain a code with parameters

$$\left[q^4, (q+1)^2 (m-q), q^4 - (q+1)^2(m-1)\right]_{q^2}.$$

In general, let us look at the Hermitian tower

$$\begin{aligned} x_2^{q+1} &= x_1^q - x_1 \\ x_3^{q+1} &= x_2^q - x_2 \\ &\vdots \quad\quad \vdots \\ x_t^{q+1} &= x_{t-1}^q - x_{t-1}. \end{aligned}$$

This tower defines a set X of q^{t+1} points in $\mathbb{F}_{q^2}^t$. The curve is irreducible, since at every step the polynomial used is irreducible (use Eisenstein's Criterion). If $m > (t-1)q$ and $(m-1)(q+1)^{t-1} < q^{t+1}$, we obtain a code with parameters

$$\left[q^{t+1}, (q+1)^{t-1}\left(m - q\frac{t-1}{2}\right), q^{t+1} - (m-1)(q+1)^{t-1}\right]_{q^2}.$$

Asymptotically, when we let t go to infinity, we obtain codes for which the relative minimum distance δ and the rate R satisfy the following relation

$$\begin{aligned} \delta + R &\geq 1 - \frac{(q+1)^{t-1}}{q^{t+1}}\left(\frac{q(t-1)}{2} - 1\right) \\ &= 1 - \left(1 + \frac{1}{q}\right)^{t-1} \frac{1}{q^2}\left(\frac{q(t-1)}{2} - 1\right). \end{aligned}$$

Unfortunately, the right hand side of the above inequality converges to $-\infty$, so that we do not get obtain asymptotically good codes (or at least, we cannot prove that we do).

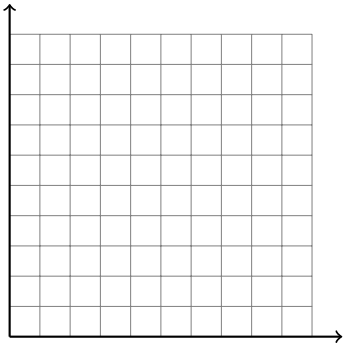


Figure 11.1: Construction of codes beyond the GV bound. The line corresponds to the equation $x + y = 1 - \gamma_q$, and the curve corresponds to the GV bound.

11.3. Codes Beyond the GV-Bound

Fundamentally, a construction like the one described above is not sufficient to find asymptotically good codes, though it is a first step in this direction. Let us see, why: Suppose that for any of the qx possibilities for x_1, we can find d_2 possibilities for x_2, d_3 possibilities for x_3, and finally d_t possibilities for x_t. Altogether, this would give rise to $qd_1d_2 \cdots d_t$ points, i.e., in 11.1, we have $n \leq qd_1d_2 \cdots d_t$. Referring to (11.1), let

$$g := \frac{d_2 + \cdots + d_t - t - 1}{2}.$$

Then, we obtain a code with parameters $[qD, (m-1)D - gD, qD - (m-1)D]_q$. Asymptotically, this gives a code for which the rate R and the relative distance δ satisfy $R + \delta \geq 1 - g/q$. But, g goes to infinity as the tower grows, so this does not give asymptotically good codes.

The presentation we have given here is a very restricted view of AG codes. In fact, the actual construction of such codes uses a different approach for which we would have to develop some parts of the language of algebraic geometry, which we are not going to do here.

Given a curve X defined over $\mathbb{F}_q$, we can associate to X an invariant, called the (geometric) *genus* of the curve. In the case of a plane algebraic curve with no singularities defined by a bivariate polynomial of degree d, this quantity is simply $(d-1)(d-2)/2$, a number we encountered in the last lecture.

We can introduce the notion of irreducibility of a curve: for a curve defined by polynomial equations in many variables, irreducibility coincides with the notion of the ideal generated by the polynomials to be prime. Moreover, we can associate with arbitrarily defined irreducible curves an analogue of the vector space $\Gamma_{<m}$, and the notion of *degree*. The dimension of $\Gamma_{<m}$, the degree d, and the genus g of nonsingular curves are intimately related: if m is large enough, then the dimension of $\Gamma_{<m}$ is $(m-1)d - g + 1$. This is often called the *Theorem of Riemann*. Putting things together, this would give us a code with parameters

$$[n, (m-1)d - g + 1, n - (m-1)d]_q,$$

where n is the number of points of the curve over $\mathbb{F}_q$. The rate and the relative distance of the code satisfy

$$R + \delta \geq 1 - \frac{g-1}{n}.$$

Let us define γ_q as the lower limit, taken over all curves for which the number of points goes to infinity, of the quantity g/n. Then we know that, asymptotically, the points (R, δ) with $R + \delta = 1 - \gamma_q$ are achievable in the sense that there are sequences of codes of rate R and relative distance at least δ for which the block length goes to infinity. If the point (R, δ) lies above the GV bound, then we have shown the existence of codes beyond the GV bound. (Technically, we need also to show that the rate R is achievable for the code, since for the codes we have built the dimension is of the form $(m-1)d - g + 1$, but if $d = o(g)$, then this is will be OK.)

When does the line $1 - \gamma_q - x$ intersect the GV curve

$$f(x) = 1 - x\log_q(q-1) + x\log_q(x) + (1-x)\log_q(1-x)?$$

For this, we calculate the derivative of $f(x)$ which is

$$f'(x) = \log_q\left(\frac{x}{1-x}\right) - \log_q(q-1).$$

This value is -1 if $x = x_0 = (q-1)/(2q-1)$. The equation of the tangent line to the GV bound at x_0 is

$$x + y = x_0 + f(x_0).$$

It turns out that if $1 - \gamma_q > x_0 + f(x_0)$, then we obtain codes beyond the GV bound.

Tsfasman, Vladut, and Zink were the first to show that if q is a square, then $\gamma_q \leq \frac{1}{\sqrt{q}-1}$. They used a special class of curves, called "modular curves" to show this result. Using this result, they showed that if q is a square and

$$x_0 + f(x_0) \leq 1 - \frac{1}{\sqrt{q}-1},$$

then their codes are above the GV bound. The situation is depicted in Figure 11.1: Between the intersection points of the line $x + y = 1 - \gamma_q$ and the GV bound we obtain codes that are genuinely above the GV bound. Solving the above equation, it turns out that q has to be larger than or equal to 49.

Theorem 11.2 (Tsfasman, Vladut, and Zink)**.** *If q is a square larger than or equal to 49, then one can construct AG codes over $\mathbb{F}_q$ that are beyond the GV bound.*

Later, Drinfeld and Vladut showed that $\gamma_q \geq \frac{1}{\sqrt{q}-1}$, thereby showing the optimality of the curves constructed by Tsfasman et al.

A description of codes derived from such curves has been vastly simplified by Garcia and Stichtenoth. They have provided an explicit tower of "function fields" and calculated the genus and and the number of points explicitly. The tower is given by

$$\begin{array}{rcl} z_2^q + z_2 & = & x_1^{q+1} \\ z_3^q + z_3 & = & x_2^{q+1} \\ \vdots & & \vdots \end{array}$$

where at each step, $x_n = z_n / x_{n-1}$.

Lecture 12
Codes and graphs

12.1. Cycle Codes

Let $G = (V, E)$ be an undirected connected graph. If an edge e connects a vertex v to some other vertex, then we say that e is *incident* to v, and denote it by $e \dashv v$. If $e = (u, v)$, then we call u and v *adjacent*.

The graph *induced* by a subset E' of the edges is the graph (V', E') where V' is the set of vertices incident to at least one edge in E'. A *spanning tree* T on G is a collection of edges such that the graph induced by these edges is a tree, and such that all vertices in V are incident to at least one edge in T.

A *cycle* in G is a collection of edges in G such that in the graph induced by them all vertices have even degree. A cycle is *primitive* if it is a closed path in which all vertices incident to an edge in the set have degree 2. The *girth* of G is the length of the smallest cycle in G.

The *cycle code* associated to G, called $C(G)$, is a code of block length $N = |E|$, which is constructed as follows. Index the coordinates of $\mathbb{F}_2^n$ by the edges of G (after fixing an ordering of the edges). Then $C(G)$ equals the set of all $(x_e \mid e \in E)$ such that for all $v \in V$ we have $\sum_{e \dashv v} x_e = 0$.

The cycle code associated to G has very interesting properties, which we will explore in this section.

Theorem 12.1. *$C(G)$ is an $[n, k, d]_q$-code where $k \geq n - |V|$, and d is the girth of G. Moreover, the codewords in $C(G)$ are characteristic functions of cycles in G.*

Proof. Since every vertex imposes a linear constraint, the assertion on the dimension follows.

The codewods in $C(G)$ are rivially the characteristic functions of cycles: if we put a one on the edges participating in a cycle, then all vertices are incident to an even number of edges carrying a one, and hence the sum of the edge labels of edges incident to the vertex is zero, so that the characteristic function of the cycle is in $C(G)$. Conversely, if we have a codeword in $C(G)$ and take all the edges corresponding to the nonzero positions of the codeword, then all vertices incident to at least one edge in the set have even degree.

It follows that the minimum distance of $C(G)$ is the length of the smallest cycle in G, i.e., it is equal to the girth of G. □

The true minimum distance of a cycle code is larger. To see this, note that

$$\sum_{v \in V} \sum_{e \dashv v} x_e = 0,$$

since in this sum every x_e appears twice, one for each endpoint of e. If $w \in V$, then this means that

$$\sum_{w \neq v \in V} \sum_{e \dashv v} x_e = \sum_{e' \dashv w} x_{e'}, \tag{12.1}$$

so that the constraint posed by w is dependent on the other constraints. This means that the dimension of $C(G)$ is at least $n - |V| + 1$. We will now prove that the dimension is equal to this number, thereby also deriving a linear time algorithm for encoding cycle codes.

Theorem 12.2. *Let T be a spanning tree on G, and $E' = E \setminus T$. Then the positions in E' are information positions of $C(G)$ and there is a simple linear time algorithm which calculates the values on the edges in T given the values on the edges in E'.*

Proof. We will develop an algorithm that maintains at every point a set S of edges the values of which have been calculated up to that point in the algorithm. Moreover, at any point in the algorithm, the set $V \setminus S$ will be a tree T'. At the beginning, $S = E'$, since we will put values on the edges of E' according to the information positions, and $T' = T$.

The leaves of T' are incident to exactly one edge in T'. Therefore, the values of these edges can be calculated using the constraints at the leaves: each value is the sum of the values of the edges in S incident to the leaf. We will add these edges into S, thereby increasing the set S. It is clear that the remaining edges (the ones not in S) still form a tree, because we have separated only the leaves from the tree in the previous round.

It remains to be shown that the labelings on the edges produce a valid codeword. Note that all the vertices of V, except for the root w of T, are forced to satisfy the constraints of the cycle code. Using (12.1), we see that the constraint at the root w has to be satisfied as well.

The algorithm provided above is clearly linear time, since for every vertex v (except the root of T) we are performing $\deg(v) - 1$ operations, where $\deg(v)$ is the degree of v. □

It can be shown that if $G_1, G_2, \ldots$ is a sequence of graphs for which the rates of $C(G_i)$ converges to some R, $0 < R < 1$, then the minimum distance of these codes will be at most $c \log(|E_i|)$ for some positive constant c, where E_i is the set of edges of G_i. We will leave this as an exercise to the reader. Instead, we will prove this for d-regular graphs, where $d > 2$.

Theorem 12.3. *Let $G_1, G_2, \ldots$ be a sequence of connected d-regular graphs with $d > 2$ for which the number of vertices goes to infinity Then there is some universal constant $c > 0$ such that the minimum distance of $C(G_i)$ is at most $c \log_{d-1}(|E_i|)$, where E_i is the set of edges of G_i.*

Proof. The equivalent assertion is to show that the girth of G_i is at most $c \log_{d-1}(|E_i|)$. Let g denote the girth of G_i, and let v be a vertex of G_i. The *neighborhood* of a subset W of vertices of G_i, denoted $\Gamma(W)$, is the set of neighbors of the vertices in W. The *i-th iterated neighborhood* of W is $\Gamma(\Gamma(\cdots \Gamma(W))) =: \Gamma^{(i)}(W)$, where the iteration goes for i steps. Thus, the first iterated neighborhood of W is just $\Gamma(W)$.

Since g is the girth of the graph, the $(g/2 - 1)$-th iterated neighborhood of $\{v\}$ is a tree: otherwise, if a node w appears twice in the iterated neighborhood, there will be two paths of length $< g/2$ from v to w, which gives rise to a cycle of length $< g$ in G_i.

The number of vertices in the k-th iterated neighborhood, under the assumption that this neighborhood is a tree, is equal to

$$|\Gamma^{(k)}(\{v\})| = \frac{d}{d-2}(d-1)^k - \frac{2}{d-2}.$$

Since the nodes in the $(g/2 - 1)$-th iterated neighborhood of $\{v\}$ are distinct, we have

$$\frac{d}{d-2}(d-1)^{g/2-1} - \frac{2}{d-2} \leq |V_i|,$$

where V_i is the number of vertices of G_i. As a result

$$g = O(\log_{d-1} |V_i|) = O\left(\log_{d-1} \frac{2|E_i|}{d}\right) = O(\log_{d-1} |E_i|),$$

and we are done. □

12.2. Strengthening Cycle Codes

Why is the minimum distance of cycle codes with a constant rate small? The reason is that constraints in a valid codeword can be adjacent to as few as two edges, giving rise to cycles being valid codewords. How can we fix this?

There are vaious ways to do so:

1. Consider more general constraints at the nodes. This gives rise to expander code.
2. Consider hyperedges instead of edges. This gives rise to LDPC codes.
3. Use decoding algorithms that allow for decoding errors at the expense of decoding larger fractions of errors. This gives rise to iterative decoding.

12.2.1. LDPC Codes

A possibility to strengthen cycle codes is through the use of LDPC codes.**LDPC code or graph code? Note that LDPC codes are low-density parity-check codes, but the hypergraph is not necessarily low-density.** In this case, a graph is replaced by a hypergraph $H = (V, E)$ wherein V is a finite set of vertices, and E is a set of subsets of V. (A graph is thus a special type of a hypergraph, one in which the set E consists of subsets of V with two elements, called *edges*.) In analogy to the case of a graph, the elements of E are called *hyperedges*. A vertex v is said to be *adjacent* to a hyperedge $e \in E$ if $v \in e$.

The code associated with the hypergraph is a subspace of $F_2^{|E|}$, and it consists of binary labeling of the hyperedges such that for each of the vertices the sum of the labels on the hyperedges adjacent to the node is zero. This is a direct generalization of cycle codes to hypergraphs.

The code described here is an LDPC code. We associate a bipartite graph B to the hypergraph H in the following way: The left (variable) nodes of B correspond to the hyperedges, and the right (check) nodes of B correspond to the vertices. There is an edge between a vertex v and a hyperedge e iff $v \in e$. It is easily seen that the graph code associated with this bipartite graph is the same as the code associated with the hypergraph.

12.2.2. Expander Codes

Let G be a d-regular graph with vertex set V and edge set E. Further, let C be a binary code with block length d, rate rd, and minimum distance εd.

First, we define an ordering on the set E of edges, so that $E = (e_1, \ldots, e_{|E|})$. For every vertex v we define an ordering of its d adjacent edges, which we denote by $e_{v,1}, \ldots, e_{v,d}$. We now consider the code $\mathcal{C}(G, C)$ defined as follows. The codewords in $\mathcal{C}(G, C)$ are all binary vectors $(x(e_1), \ldots, x(e_{|E|}))$ such that for all $v \in V$ we have $(x(e_{v,1}), \cdots, x(e_{v,d})) \in C$. In this terminology a cycle code is a code of the form $\mathcal{C}(G, P)$, where P is the $[d, d-1, 2]_2$ parity code.

What are the parameters of $\mathcal{C}(G, C)$? Let $n := |V|$. At this point, we can give a partial answer to this question.

Theorem 12.4. *The code $\mathcal{C}(G, C)$ is a $[dn/2, k]_2$-code, where $k \geq dn(2r-1)/2$.*

Proof. Every vertex in G has degree d, and every edge is adjacent to 2 vertices, so G has $dn/2$ edges.

As for the dimension, each vertex in G imposes $d(1-r)$ linear constraints on the codewords. In total, this is equal to $dn(1-r)$ constraints on $dn/2$ edges, so that the dimension of $\mathcal{C}(G, C)$ is at least $dn/2 - dn(1-r) = dn(2r-1)/2$. □

What about the minimum distance of $\mathcal{C}(G, C)$? An answer to this can be given using expansion properties of G Let A denote the adjacency matrix of G. It is an $n \times n$ matrix which is symmetric and in which all rows and all columns sum up to d. We denote by L the matrix A/d. Then L is doubly stochastic, i.e., the sum of each row and each column of L is 1.

Since L is symmetric, all its eigenvalues are real. Let $\lambda_{n-1} \leq \lambda_{n-2} \leq \cdots \leq \lambda_1 \leq \lambda_0$ denote the n eigenvalues of L, sorted in descending order. Then we have

Proposition 12.5. *The following assertions hold for the eigenvalues of L:*

(1) *The largest eigenvalue of L is 1, i.e., $\lambda_0 = 1$.*

(2) $|\lambda_{n-1}| \leq 1$.

(3) $\lambda_{n-1} = -1$ *if and only if G is bipartite.*

(4) $\lambda_1 = 1$ *if and only if G is not connected.*

(5) *If G is bipartite, then the eigenvalues are symmetric around 0.*

We shall need also a theorem of Alon and Chung, which bounds the number of edges in subgraphs of expanders. More precisely, let $X \subseteq E$ be a subset of the edges of G. The graph *induced* by X is the graph formed by the endpoints of the edges in X.

Theorem 12.6. *Let G be a d-regular graph with second largest eigenvalue λ. Let $X \subseteq E$ with $|X| = xdn/2$, $n := |V|$. Then the number of vertices in the graph induced by $|X|$ is at least γn, where γ is the unique solution of*

$$x = \gamma^2 + \lambda\gamma(1-\gamma).$$

Proof. Let S be a subset of the vertices and let $E(S)$ denote the set of edges in E with both endpoints in S, i.e., $E(S) = E \cap S \times S$. Further, let $e(S) := |E(S)|$. We will prove that if $|S| = \gamma n$, then

$$|e(S)| \leq \frac{dn}{2}\left(\gamma^2 + \lambda\gamma(1-\gamma)\right). \tag{12.2}$$

It can easily be seen that this is equivalent to the assertion of the theorem: if W is the set of vertices in the graph induced by X, then $X \subseteq E(W)$, so that $|X| \leq e(W)$. If $|W| = \mu n$, then $x = |X|/(dn/2) \leq \mu^2(1-\lambda) + \lambda\mu$, and since $t^2(1-\lambda) + \lambda t$ is an increasing function of t, $\mu \geq \gamma$.

To prove (12.2), let L denote the normalized adjacency matrix of G, so that λ is the second largest eigenvalue of L. Moroever, let f be a vector that is $1/|S|$ on S and $-1/|S|$ on $\overline{S}$. Since f is orthogonal to $\mathbf{1}$, we have

$$|\langle Lf, f\rangle| \leq \lambda\langle f, f\rangle. \tag{12.3}$$

We index the coordinates of f by the vertices in G. Then we have

$$\begin{aligned}
\langle Lf, f\rangle &= \frac{2}{d}\sum_{(u,v)\in E} f_u f_v \\
&= \frac{1}{d}\sum_{(u,v)\in E}\left[f_u^2 + f_v^2 - (f_u^2 - 2f_uf_v + f_v^2)\right] \\
&= \frac{1}{d}\sum_{(u,v)\in E}(f_u^2 + f_v^2) - \frac{1}{d}\sum_{(u,v)\in E}(f_u - f_v)^2 \\
&= \sum_{v\in V} f_v^2 - \frac{1}{d}\sum_{(u,v)\in E}(f_u - f_v)^2.
\end{aligned}$$

Let $E(S,\overline{S})$ be the set of edges with one endpoint in S and another in $\overline{S}$, and let $e(S,\overline{S}) := |E(S,\overline{S})|$. Then we have

$$\sum_{v\in V} f_v^2 = \frac{1}{|S|} + \frac{1}{n-|S|}, \quad \sum_{(u,v)\in E}(f_u - f_v)^2 = e(S,\overline{S})\left(\frac{1}{|S|} + \frac{1}{n-|S|}\right)^2.$$

Combining this with (12.3), we obtain

$$|\langle Lf, f\rangle| = \left|1 - \frac{e(S,\overline{S})}{d}\left(\frac{1}{|S|} + \frac{1}{n-|S|}\right)\right| \leq \lambda.$$

Since $|S| = \gamma n$, we have

$$|e(S,\overline{S}) - d\gamma(1-\gamma)n| \leq d\lambda\gamma(1-\gamma)n.$$

All the γn vertices of S have d adjacent edges. Some of these edges are in $E(S)$, while the others are in $E(S,\overline{S})$. Therefore, $d\gamma n = 2e(S) + e(S,\overline{S})$. Therefore, we have

$$\begin{aligned}
\left|\frac{1}{2}d\gamma^2 n - e(S)\right| &= \left|\frac{1}{2}d\gamma n - e(S) - \frac{1}{2}d\gamma(1-\gamma)n\right| \\
&= \frac{1}{2}|e(S,\overline{S}) - d\gamma(1-\gamma)n| \\
&\leq \frac{1}{2}d\lambda\gamma(1-\gamma)n.
\end{aligned}$$

Therefore, $e(S) \leq dn(\gamma^2 + \lambda\gamma(1-\gamma))/2$, and we are done. □

Theorem 12.7. *Suppose that G is a d-regular graph with second largest eigenvalue λ. Further, let C be a $[d, rd, \varepsilon d]_2$-code. Then the minimum distance of $\mathcal{C}(G,C)$ is at least $\delta dn/2$, where*

$$\delta := \varepsilon\frac{\varepsilon - \lambda}{1-\lambda}.$$

Proof. Suppose that $\mathcal{C}(G,C)$ contains a nonzero codeword of weight less than $\delta dn/2$, and let X denote the set of edges that carry a nonzero label in this codeword (i.e., the codeword is the characteristic function of the set X). By

assumption, $|X| < \delta dn/2$, so that $\mu := |X|/(dn/2) < \delta$. By Theorem 12.6 the graph induced by X has at least γn nodes, where $\gamma^2(1-\lambda)+\lambda\gamma = \mu$. The function $\gamma^2(1-\lambda)+\lambda\gamma$ increases monotonically in γ, and since

$$\delta = \left(\frac{\varepsilon-\lambda}{1-\lambda}\right)^2(1-\lambda)+\lambda\frac{\varepsilon-\lambda}{1-\lambda},$$

we know that $\gamma < (\varepsilon-\lambda)/(1-\lambda)$.

Since the graph induced by X has at least γn vertices, the average degree of each vertex in this graph is at most

$$\begin{aligned}\frac{2\frac{dn}{2}\mu}{\gamma n} &= \frac{d\mu}{\gamma}\\ &= d\left(\gamma(1-\lambda)+\lambda\right)\\ &< d\left(\frac{\varepsilon-\lambda}{1-\lambda}(1-\lambda)+\lambda\right)\\ &= d\varepsilon.\end{aligned}$$

This shows that some vertex in the graph induced by X has degree $< d\varepsilon$, and larger than 0. But this is a contradiction, since in the induced graph the number of edges at each vertex should be at least $d\varepsilon$, since these edges give rise to a nonzero codeword in C. □

Bibliography

[Bré09] Pierre Brémaud. *Initiation aux probabilités et aux chaînes de Markov*. Springer-Verlag, Berlin, revised edition, 2009.

[CT06] Thomas M. Cover and Joy A. Thomas. *Elements of information theory*. Wiley-Interscience [John Wiley & Sons], Hoboken, NJ, second edition, 2006.

[Dem97] Michel Demazure. *Cours d'algèbre*. Nouvelle Bibliothèque Mathématique [New Mathematics Library], 1. Cassini, Paris, 1997. Primalité. Divisibilité. Codes. [Primality. Divisibility. Codes].

[MS77] F. J. MacWilliams and N. J. A. Sloane. *The theory of error-correcting codes.* North-Holland Publishing Co., Amsterdam, 1977. North-Holland Mathematical Library, Vol. 16.

[Pre98] Oliver Pretzel. *Codes and algebraic curves*, volume 8 of *Oxford Lecture Series in Mathematics and its Applications*. The Clarendon Press Oxford University Press, New York, 1998.

[Ste99] Serguei A. Stepanov. *Codes on algebraic curves*. Kluwer Academic/Plenum Publishers, New York, 1999.

[SW49] Claude E. Shannon and Warren Weaver. *The Mathematical Theory of Communication*. The University of Illinois Press, Urbana, Ill., 1949.

[vL99] J. H. van Lint. *Introduction to coding theory*, volume 86 of *Graduate Texts in Mathematics*. Springer-Verlag, Berlin, third edition, 1999.

[vzGG03] Joachim von zur Gathen and Jürgen Gerhard. *Modern computer algebra*. Cambridge University Press, Cambridge, second edition, 2003.

[WB94] Stephen B. Wicker and Vijay K. Bhargava. *Reed-Solomon Codes and Their Applications*. IEEE Press, Piscataway NJ, 1994.