Math 342: Algebra and Coding Theory[1]

Fall 2023/24

[1] Taught by Prof. Kalle Karu

These are my class notes, further research and a possibly random collection of facts from this course.

Introduction to error-correcting codes

Error correcting codes are used to correct errors when messages are transmitted through a noisy channel. Some terminology,

- A **code** is a set of codewords
- A **codeword** is a sequence of symbols chosen from a set $F_q = \{\lambda_1, \lambda_2, \cdots, \lambda_q\}$
- A **q-ary** code is a given set of sequences of symbols chosen from F_q
- The set F_q is called the **alphabet** and is often taken to be the set $Z_q = \{0, 1, 2, \cdots, q-1\}$
- A code in which each codeword is a sequence consisting of a fixed number n of symbols is called a **block code** of length n.
- Let $(F_q)^n$ denote the set of all ordered n-tuples $a = a_1a_2a_3\cdots$ where each $a_i \in F_q$. The order of the set $(F_q)^n$ is q^n. A q-ary code of length n is just a subset of $(F_q)^n$.

To explore the idea of a codeword being "closer" to another, we introduce the ***hamming distance***.

$$d(a_1a_2a_3\cdots, b_1b_2b_3\cdots) = \#\text{ of places the two codewords differ}$$

The hamming distance is a legitimate distance function, or metric, since it satisfies the following three properties:

- $d(x, y) = 0 \leftrightarrow x = y$
- $d(x, y) = d(y, x)$ for all $x, y \in (F_q)^n$
- $d(x, y) \leq d(x, z) + d(z, y)$ for all $x, y, z \in (F_q)^n$

Any set S with a distance function is a **metric space**.

An important parameter of a code $\mathcal{C}$, giving a measure of how good it is at error correcting, is the minimum distance, denoted $d(\mathcal{C})$, which is defined as the following

$$d(\mathcal{C}) = min\{d(w_i, w_j) \mid w_i, w_j \in \mathcal{C}, w_i \neq w_j\}$$

Theorem 1.1. If $d(\mathcal{C}) = d$ then

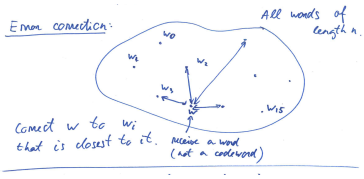

Figure 1: The main idea behind error correction

- $\mathcal{C}$ can detect upto $d-1$ errors (since it needs atleast d errors to reach a new codeword)
- $\mathcal{C}$ can correct $< \frac{d}{2}$ errors

or, equivalently

- A code C can detect upto s errors in any codeword if $d(C) \geq s+1$
- A code C can correct upto t errors in any codeword if $d(C) \geq 2t+1$

Parameters of a code

A q-ary (n, M, d)-code has parameters

- q = size of the alphabet
- $n =$ lenght of codewords
- $M =$ number of codewords
- $d =$ minimum distance between codewords

The main coding theory problem

Now, we ask ourselves - what makes a good code? We would assume that a good code will have the following properties:

- large $M \rightarrow$ gives many codewords enabling wide variety of messages
- small $n \rightarrow$ give us a small length allowing fast transmission
- large $d \rightarrow$ correct many errors

These conflicting aims are referred to as the *main coding theroy problem*. The usual version of this problem is to find the largest q-ary code of a given length and given minimum distance.

We denote by $A_q(n, d)$ the largest value of M such that there exists q-ary (n, M, d)-code. This problem is easily solved for $d = 1$ and $d = n$ for all q.

Theorem 2.2. (i) $A_q(n, 1) = q^n$ (ii) $A_q(n, n) = q$

If we fix $q = 2$, we get *binary* codes. The table to the right specifies the value of $A_2(n, d)$ for values of n, d. [2]

Theorem 2.3. Suppose d is odd. Then a binary (n, M, d)-code exists if and only if a binary $(n+1, M, d+1)$-code exists.

[2] Various values of $A_2(n, d)$

n/d	1	2	3	4	5
1	2^1	-	-	-	-
2	2^2	2^1	-	-	-
3	2^3	2^2	2^1	-	-
4	2^4	2^3	2^2	2^1	-
5	2^5	2^4	2^3	2^2	2^1

The proof for this theorem requires the following definitions and lemmas.

Definition 2.1. The **weight** of a binary word W is the number of 1s in W.

Lemma 2.1. If $x, y \in (F_2)^n \implies d(x,y) = w(x+y) = w(x) + w(y) - 2w(x \cap y)$[3]

Proof. ($\implies$) Suppose C is a (n, M, d)-code and d is odd. Let $\hat{C}$ be the code of length $n+1$ obtained from C by extending every codeword in C according to the rule[4]

$$x = x_1x_2x_{3n} \to \hat{x} = \begin{cases} x_1x_2 \dots x_n0 & \text{if } w(x) \text{ is even} \\ x_1x_2 \dots x_n1 & \text{if } w(y) \text{ is odd} \end{cases}$$

Since $w(\hat{x})$ is even for $\hat{x} \in \hat{C}$, it follows from lemma above that $d(\hat{x}, \hat{y})$ is also even for any $\hat{x}, \hat{y} \in \hat{C}$. Since $\hat{C}$ is an extension of C, it must be that

$$d \leq d(\hat{C}) \leq d+1$$

Since d is odd, it must be that $d(\hat{C}) = d+1$.

($\impliedby$) Suppose D is any $(n+1, M, d+1)$-code where d is odd. Choose codewords $x, y \in D$ such that $d(x,y) = d+1$ and find a position where x, y both differ. Remove this position all codewords in D. We are left with a (n, M, d)-code. □

[3] For $x, y \in (F_q)^n$ the operations $+$ and $\cap$ are defined as follows:

- $x + y = (x_1 + y_1, x_2 + y_2, \dots)$
- $x \cap y = (x_1 \cdot y_1, x_2 \cdot y_2, \dots)$

where $x_i + y_i$ and $x_i \cdot y_i$ are performed modulo q

[4] This construction of $\hat{C}$ from C is called *adding an overall partiy check*.

Equivalence of codes

Let S_1, S_2 be two distinct metric spaces. Then we say that $f : S_1 \to S_2$ is an **isometry** if it preserves distances.

We say that two codes are **equivalent** if we can get from one to other through a sequence of *elementary operations*. These elementary operations can be one of:

1. Permute codewords (rows)
2. Permute columns
3. In one column, permute symbols (e.g $1 \to 0$ in column 2)

Equivalent codes are isometric. That means there exists a bijection $f : C_1 \to C_2$ preserving distance.

$$d(w_1, w_2) = d(f(w_1), f(w_2))$$

Sphere packing

We now introduce the notion of a sphere in the set $(F_q)^n$.

Definition 2.2. For any vector u in $(F_q)^n$ and any integer $r \geq 0$, the *sphere* of radius r and centre u, denoted by $S(u, r)$ is the set

$$\{v \in (F_q)^n \mid d(u, v) \leq r\}$$

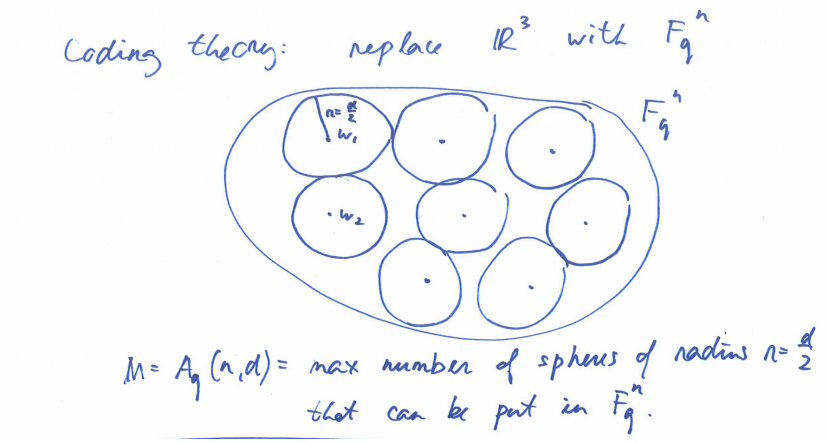

Figure 2: Visualizing codewords in $(F_q)^n$

Recall that a code C can correct t errors if $d(C) \geq 2t + 1$. Visualized, this means that the spheres of radius t centered at the codewords of C are disjoint. Therefore, if t or fewer errors occurs, then the received vector may be different from the centre of the sphere, but it cannot escape the sphere and will be drawn back by nearest neighbour decoding.

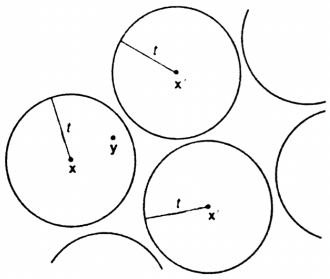

Figure 3: Visualization of C

Now recall the main coding theory problem of determining the largest value of M such that there exists a q-ary (n, M, d)-code. Sometimes, it isn't possible to determine the exact value for M. In which case, we need to figure out bounds on M. Sphere packing gives one such bound. Let $N_{q,r}$ denote the number of points in a sphere of radius in $(F_q)^n$. If we fit M non-overlapping spheres into $(F_q)^n$, we get

$$M \cdot N_{q,r} \leq q^n \text{ (total number of points in } (F_q)^n)$$

And therefore,

$$M \leq \frac{q^n}{N_{q,r}}$$

Lemma 2.2. A sphere of radius r in $(F_q)^n$ and $0 \leq r \leq n$ contains exactly

$$\binom{n}{0} + \binom{n}{1} \cdot (q-1) + \binom{n}{2} \cdot (q-1)^2 + \cdots + \binom{n}{r} \cdot (q-1)^r$$

vectors.

Proof. Let u be any fixed vector in $(F_q)^n$. Consider how many vectors v have distance exactly m from u, for some $m \leq n$. The m positions

in which v is to differ from u can be chosen in $\binom{n}{m}$ ways. In each of these m positions, the entry of v can be chosen in $q-1$ ways to differ from the corresponding entry in u. Hence, the number of vectors at distance exactly m from u is given by $\binom{n}{m} \cdot (q-1)^m$ and so the total number of vectors in $S(u,r)$ is given by

$$\binom{n}{m} + \binom{n}{1} \cdot (q-1) + \binom{n}{2} \cdot (q-1)^2 + \cdots + \binom{n}{r} \cdot (q-1)^r$$

□

Theorem 2.4. A q-ary $(n, M, 2t+1)$-code satisfies

$$M \cdot \left[\binom{n}{0} + \binom{n}{1} \cdot (q-1) + \binom{n}{2} \cdot (q-1)^2 + \cdots + \binom{n}{t} \cdot (q-1)^t \right] \leq q^n$$

The radius for a sphere in a code is given by $r = (d(C)-1)/2$ (truncated)

Definition 2.3. A ***perfect code*** is one that achieves the sphere packing bound with equality.

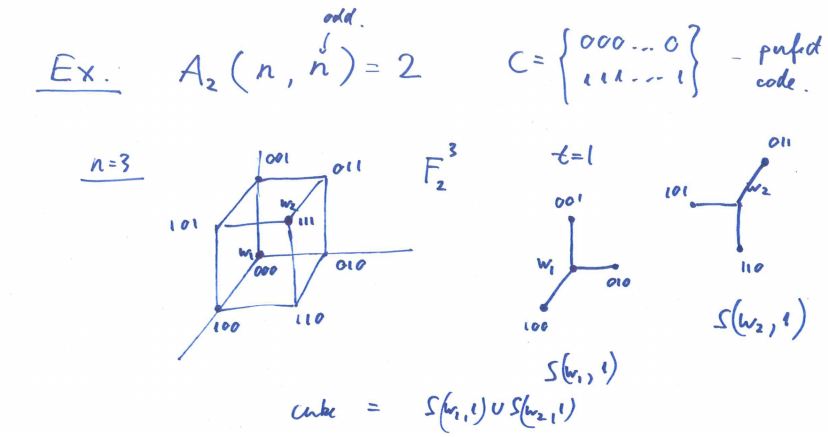

Figure 4: Examples of perfect codes

An introduction to finite fields

To make error-correcting codes easier to analyse, we need to impose a algebraic structure[5] onto them.

Definition 3.4. A ***field*** F is a set of elements with two operations[6] $+$ (called addition) and $\cdot$ (called multiplication) satisfying the following properties:

(i) F is closed under $+$ and $\cdot$

(ii) Commutative laws hold - i.e $a+b = b+a, a \cdot b = b \cdot a$

[5] An algebraic structure consists of non-empty set A, a collection of operations on A and finite set of identites, known as axioms, that these operations must obey.

[6] well, you can say a field has 4 operations, but division and substraction are just multiplication and addition

(iii) Associative laws hold - $(a+b)+c = a+(b+c), a\cdot(b\cdot c) = (a\cdot b)\cdot c$

(iv) Distributive law - $a\cdot(b+c) = a\cdot b + a\cdot c$

(v) Identity elements 0 and 1 must exist in F

(vi) $a+0=a$

(vii) $a\cdot 1 = a$

(viii) There must exists an additive inverse $(-a)$ in F such that $a+(-a)=0$

(ix) For any $a \neq 0$, there exists a multiplicative inverse (a^{-1}) such that $a\cdot a^{-1}=1$

Lemma 3.3. Any field F has the following properties:

(i) $a\cdot 0 = 0$ for all $a \in F$

(ii) $ab = 0 \implies a = 0$ or $b = 0$

Definition 3.5. Any set of elements with $+$ and $\cdot$ satisfying the properties (i) to $(viii)$ but not necessarily (ix) is called a ***ring***.

Examples of fields include $\mathbb{R}, \mathbb{C}$. Examples of rings include $\mathbb{Z}$. Note that every field is a ring.

Definition 3.6. A *finite field* has a finite number of elements in it, this number being called the *order* of the field.

Theorem 3.5. [7] There exists a field of order q if and only if q is a prime power (i.e $q = p^h$ where p is a prime number and h is a positive integer).Furthermore, if q is a prime power, then there is only one field of that order.

[7] Proved by Evariste Galois (1811-32)

Definition 3.7. A field of order q is often called a *Galois field* and is denoted by $GF(q)$.

In this course, we consider only *prime fields*, those of order a prime number p. We shall see that if p is prime, then $GF(p)$ is just the set $\{0,1,2,\ldots,p-1\}$ with arithmetic carried out modulo p.

But first, a review of modular arithmetic.

Modular Arithmetic

Consider $a, b \neq 0 \in \mathbb{Z}$

$$\frac{a}{b} = \underbrace{q}_{\text{quotient}} + \frac{\overbrace{r}^{\text{remainder}}}{b}$$

where $b > 0, r \in \{0,1,2,...,b-1\}$

Definition 3.8. We say that b divides a, $b \mid a$, if

$$\frac{a}{b} = q \leftrightarrow a = bq$$

Equivalently, we say that a divides b if $b/a \in \mathbb{Z}$. We say that the **divisions** of a is the set of all $b > 0$ such that $b \mid a$

Definition 3.9. A number p is **prime** if its divisions are $\{1, p\}$ only.

Theorem 3.6. Consider $a, b \neq 0$, $gcd(a, b) = gcd(a, b - q \cdot a)$ for any $q \in \mathbb{Z}$

Example 3.1.

$$\begin{aligned} gcd(24, 90) &= gcd(24, 90 - 3 \cdot 24) \\ &= gcd(24, 18) \\ &= gcd(24 - 18, 18) \\ &= gcd(6, 18) \\ &= gcd(6, 18 - 3 \cdot 6) \\ &= gcd(6, 0) \end{aligned}$$

This is the **Euclidean** algorithm for finding $gcd(a, b)$

Theorem 3.7. [8] Let $d = gcd(m, n)$. Then we can write

$$d = a \cdot m + b \cdot n$$

for some $a, b \in \mathbb{Z}$

Proof. Use Euclidean algorithm □

[8] This is known as Bézout's identity

Definition 3.10. $m, n \in \mathbb{Z}$ are relatively prime if $gcd(m, n) = 1$.

Definition 3.11. We say that a is congruent to b modulo m if $a = b + qm$ for some q. We denote this by $a \equiv b \pmod{m}$.

Definition 3.12. $\mathbb{Z}_m = \{0, 1, 2, \ldots, m-1\}$ with operations $+$ and $\cdot$ defined as

- $a + b =$ principal remainder of $\frac{a+b}{m}$
- $a \cdot b =$ principal remainder of $\frac{ab}{m}$

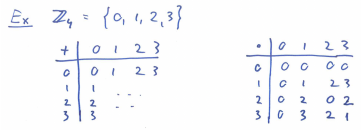

Figure 5: Addition and multiplication in $\mathbb{Z}_4$

Theorem 3.8. If $a \equiv a' \pmod{m}$ and $b \equiv b' \pmod{m}$ then we have that

- $a + b \equiv a' + b' \pmod{m}$
- $ab \equiv a' \cdot b' \pmod{m}$

Definition 3.13. Inverse of y $(1/y)$ in mod n is some $x \in \mathbb{Z}_n$ such that $y \cdot x \equiv 1$ (mod n).

Theorem 3.9. $1/b$ exists in $\mathbb{Z}_m \iff gcd(b,m) = 1$

Proof. @TODO □

Example 3.2. [9] Find $1/14 \ (mod\ 33)$

First note that $gcd(14,33) = 1$, therefore, the inverse of 14 exists in $(mod\ 33)$.

$$\begin{aligned} 1/14 \Leftrightarrow 1 &= c \cdot 14 && \text{in } \mathbb{Z}_{33} \\ 1 &= c \cdot 14 + q \cdot 33 && \text{in } \mathbb{Z} \end{aligned}$$

By the euclidean algorithm, we get $c = -7, q = 3$

$$\frac{1}{14} \equiv -7 \equiv -7 + 33 = 26 \ \ (mod\ 33)$$

[9] https://math.stackexchange.com/questions/586595/finding-modular-of-a-fraction

Theorem 3.10. $\mathbb{Z}_m$ is a ring. But for special m, $\mathbb{Z}_m$ is a field. More precisely,

$$\mathbb{Z}_m \text{ is a field} \iff m \text{ is prime}$$

Proof. @TODO □

Next we briefly we discuss two special properties of fields.

- **Cancellation Property**: If $a \cdot b = a \cdot c$ and $a \neq c$ then $b = c$. This is true because the multiplicative inverse of a exists.

  Example 3.3. (in $\mathbb{Z}_4$) $2 \cdot 2 = 2 \cdot 0$ but $2 \neq 0$

- **Zero Divisions**: If $a \cdot b = 0$ and $a, b \neq 0$ then a, b are zero divisions. There are no zero divisions in a field. To see why, consider

$$\begin{aligned} ab &= 0 \\ \frac{ab}{a} &= \frac{0}{a} && \text{multiplicative inverse of } a \text{ exists} \\ b &= 0 && \text{therefore it cannot be that } b \neq 0 \end{aligned}$$

A remark on notation: $\mathbb{Z}_p = \mathbb{F}_p = GF(p)$ where p is prime. $\mathbb{F}_p$ represents a finite field of p elements.

There is exactly one field with q elements in it. We know all the finite fields (take q to be any prime power). The equality $\mathbb{F}_q = GF(q)$ always holds.

The ISBN Code

The international standard book number.

- The first digit indicates the language
- The next two digits indicate the publishers
- The next six digits are assigned uniquely to every book
- The final digit is chosen to make the whole 10-digit number $x_1 x_2 \ldots x_1 0$ satisfy

$$\sum_{i=1}^{10} i x_i \equiv 0 \ (mod\ 11)$$

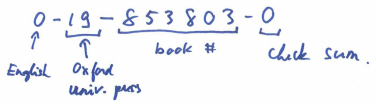

Figure 6: The ISBN code

Therefore, the ISBN code is a $q = 11$ code with $n = 10$. All codewords satisfy

$$1 \cdot x_1 + 2 \cdot x_2 + \cdots + 10 \cdot x_{10} = 0 \quad (in\ \mathbb{Z}_{11})$$

ISBN code is designed to

1. Detect 1 error
2. Detect any error created by the transposition of two digits [10]

[10] this works because of the weight assigned to every digit in the sum

The error detection scheme is simply to calculate the sum talked about above and check whether this sum $Y \equiv 0 \pmod{11}$

ISBN codes cannot be used to correct errors unless we know that just one digit is in error.

Vector Spaces over Finite Fields

For this section (and most of remainder of the course) we let our q-ary code have the alphabet $GF(q)$ (for prime q). The set $GF(q)^n$ forms a **vector space**.

Definition 5.14. Vector space $\mathbb{Z}_p^n = \mathbb{F}_p^n = V(n, p)$

Example 5.4. (in $\mathbb{Z}_5^3$) $\vec{x} = (1, 2, 2, 0, 1)$

We define two operations over a vector space.

1. **Addition of vectors**:

$$(x_1, x_2, \ldots, x_n) + (y_1, y_2, \ldots, y_n) = (x_1 + y_1, x_2 + y_2, \ldots, x_n + y_n)$$

2. **Scalar multiplication**:

$$c \cdot (x_1, \ldots, x_n) = (cx_1, \ldots, cx_n)$$

for all $c \in \mathbb{Z}_p$.

Definition 5.15. A basis for $\mathbb{Z}_p^n$ is a set of n vectors $\{\vec{v_1}, \ldots, \vec{v_n}\}$ such that any vector $\vec{w} \in \mathbb{Z}_p^n$ can be expressed as

$$\vec{w} = c_1\vec{v_1} + \cdots + c_n\vec{v_n}$$

for unique $c_1, \ldots, c_n \in \mathbb{Z}_p$

Recall from linear algebra that $\vec{v_1}, \ldots, \vec{v_2}$ is a basis of $\mathbb{Z}_p^n$ iff

- It spans that space $\mathbb{Z}_p^n$
- It is linearly independent

Definition 5.16. A subset $W \subseteq \mathbb{Z}_p^n$ is a subspace if

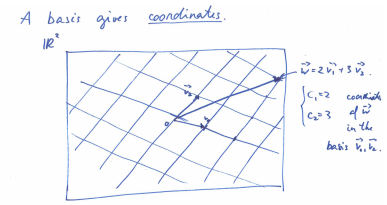

Figure 7: You can think of basis as defining the coordinate lines for a vector space

1. $\vec{0} \in W$
2. W is closed under addition and scalar multiplication

$$\vec{w}_1, \vec{w}_2 \in W, c \in \mathbb{Z}_p \iff \vec{w}_1 + \vec{w}_2 \in W$$
$$\iff c \cdot w_1, c \cdot w_2 \in W$$

A subspace W is itself a vector space.

Theorem 5.11. Solutions to a homogenous linear solution form a subspace. That is, the set of all solutions to

$$c_1 \cdot x_1 + \cdots + c_n \cdot x_n = 0$$

for all $c_1, \ldots, c_n \in \mathbb{Z}_p$ form a subsapce of $\mathbb{Z}_p^n$.

Proof. From linear algebra. □

Definition 5.17. The dimension of a subspace, denoted by $dim(W)$, is the number of vectors in its basis.

Remark: Notice that, over $\mathbb{Z}_p$, we have that

$$c \cdot \vec{v} = \underbrace{\vec{v} + \ldots \vec{v}}_{c \text{ times}}$$

Therefore, closed under addition $\implies$ closed under scalar multiplication. This is **not true** over any other field

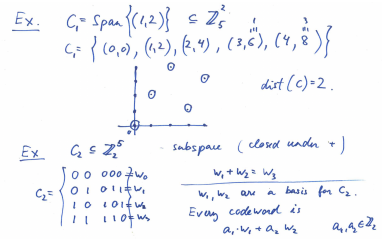

Figure 8: Examples of linear codes

Linear Codes

Linear codes are given by the alphabet $\mathbb{F}_q (= \mathbb{Z}_q)$. The code is a subspace $C \subseteq \mathbb{F}_q^n$

Notation: We have encountered the notation (n, M, d) where

- n is the length of the codewords
- M is the number of codewords in our code
- d is the minimum distance of our code

For linear code, we introduce the following notation

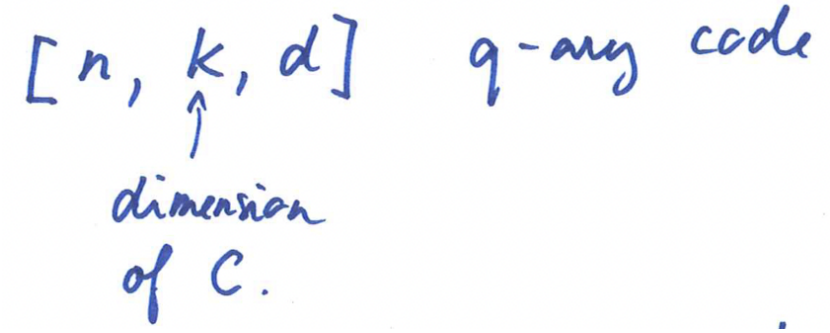

Figure 9: Notation for linear codes

If our code C has $dim(C) = k$, then it has $M = q^k$ codewords. To see why, notice that we have k basis vectors, and so k coefficients. Since we are in $\mathbb{Z}_q$ space, we have q choices for each of these coefficients. This gives us q^k possible vectors.

Definition 6.18. The weight of a codeword is the number of non-zero entries.

Theorem 6.12. Let C be a linear code. Then we have that

$$dist(C) = min\ weight(\vec{w})$$

for all $\vec{w} \in C$ such that $\vec{w} \neq \vec{0}$.

Note: To find minimum weight, you need to consider all codewords of C, not just its basis vectors.

Continuing our discussion of notation, earlier the only way to define a code was to either use set-builder notation or list all the codewords. With linear codes, we just list a basis for the code C. We package the basis vectors in a **generator matrix**.

Definition 6.19. A $k \times n$ matrix whose rows form a basis of a linear $[n, k]$-code is called a **generator matrix**.

$$\begin{bmatrix} \cdots & \vec{v}_1 & \cdots \\ \cdots & \vec{v}_2 & \cdots \\ & \vdots & \\ \cdots & \vec{v}_k & \cdots \end{bmatrix}$$

Standard form of the generator matrix

The standard form of the generator matrix is

$$\left[I_k \,\middle|\, A \right]$$

where I_k is the $k \times k$ identity matrix and A is a $k \times (n-k)$ matrix.

The following operations are allowed (and are guaranteed) to transform any generator matrix G into the standard form

1. Permutation of rows
2. Multiplication of row by a non-zero scalar
3. Addition of scalar multiple of one row to another
4. Permutation of the columns
5. Multiplication of any column by a non-zero scalar

For any given matrix G, first transform the matrix to reduced row-echelon form and then permute the columns to get to standard form.

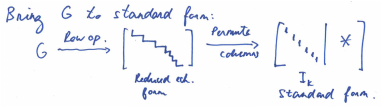

Figure 10: Getting to the standard form

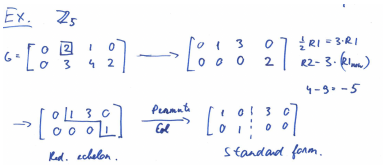

Figure 11: An example of transforming G to the standard form

Encoding and Decoding with a Linear Code

Every vector in a $[n,k]-$code can be written as

$$a_1\vec{v_1} + \cdots + a_k\vec{v_k}$$

To encode linear codes, we use the codeword

$$(a_1, a_2, \ldots, a_k)$$

By encoding the coefficients, we can obtain the actual codeword by

$$[a_1, a_2, \ldots, a_k] \cdot G = \sum_{i=1}^{k} a_i r_i$$

where r_i is a row of the generator matrix, i.e a basis vector.

Decode Decode Decode

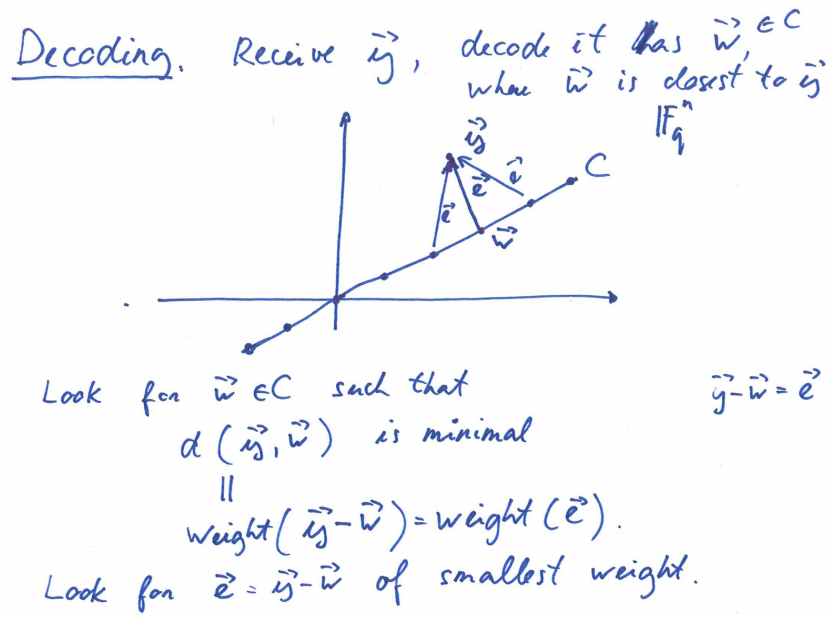

Figure 12: Decoding: a visualization

Suppose we recieve a vector $\vec{y} = y_1 \ldots y_n$. Let the original vector that was sent be $\vec{x} = x_1 \ldots x_n$. Then we define the error vector to be

$$\vec{e} = \vec{y} - \vec{x} = e_1 \ldots e_n$$

The decoding mechanism must decide from y which codeword x was transmitted, or, which error e has occurred.

Definition 7.20. Suppose that C is an $[n,k]$-code over $GF(q)$ and that a is any vector in $V(n,q)$. Then the set $a+C$ is defined by

$$a + C = \{a + x \,|\, x \in C\}$$

and is called a coset of C.

Theorem 7.13. (Lagrange) Suppose C is an $[n,k]$-code over $GF(q)$. Then

1. every vector of $V(n,q)$ is in some coset of C
2. every coset contains exactly q^k vectors
3. two cosets are either disjoint or coincide (partial overlap is impossible)

The idea is very similar to nearest-neighbour decoding. It is summarized below:

- Find the coset $\vec{y} + C$.
- Find the vector in this coset $\vec{e}$ with the smallest weight.
- Decode $\vec{y}$ as $\vec{w} = \vec{y} - \vec{e}$

Lemma 7.4. Suppose that $a + C$ is a coset of C and that $b \in a + C$, then we have that $a + C = b + C$.

Proof. @TODO □

Note that while cosets are either exactly equal or disjoint, it is always true that every memeber of $V(n,q)$ lies in some coset of C. So how many cosets are there? Well, we know that each coset has q^k members and that the parent space has q^n members. This means that there must be q^{n-k} **distinct** cosets.

Definition 7.21. The vector having minimum weight is called the coset leader. If there are two or more candidates for coset leader, anyone can be picked.

Definition 7.22. A (Slepian) Standard Array for a $[n,k]-$code is a $q^{n-k} \times q^k$ array of all the vectors in $V(n,q)$. The first row lists all members of C, with $00\ldots0$ vector at the leftmost position in the row. The other rows are the cosets $a_i + C$, with the coset leader at the leftmost position.

An intuition: We know that, for linear codes, $d(x,y) = w(x-y)$. For a vector in any coset, the distance between it and its corresponding vector in C is given by the coset leader. In other words, the minimum distance between $y \in a_i + C$ and $w \in C$ is $w(a_i)$. In this way, we find our closest neighbour.

Example 6.5 Let C be the binary $[4,2]$-code with generator matrix

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

Figure 13: An example code

Example 6.6 A standard array for the code of Example 6.5 is

codewords	→ 0000	1011	0101	1110
	1000	0011	1101	0110
	0100	1111	0001	1010
	0010	1001	0111	1100
	↑			
	coset leaders			

Figure 14: Its standard array

Now, this simplifies decoding greatly. We no longer need to compare each digit to find the closest neighbour. But matrices can be get pretty big too. It can be tedious to find $\vec{y}$ in a large enough matrix. We tackle this problem next.

Dual Code, Parity-Check Matrix and Syndrome Decoding

The parity-check matrix, just like the generator matrix, provides a way for us to specify a linear code.

Definition 8.23. The *inner product* of two vectors $u \cdot v$ where $u = u_1u_2 \dots u_n$ and $v = v_1v_2 \dots v_n$ in F_q^n is the scalar defined by

$$u \cdot v = u_1v_1 + u_2v_2 + \cdots + u_nv_n$$

If $u \cdot v = 0$, then we say that u, v are **orthogonal**.

Definition 8.24. We define the **dual code** of $C \subseteq F_q^n$ to be

$$C^\perp = \{\vec{x} \in F_q^n \mid \vec{x} \cdot \vec{w} = 0 \text{ for all } \vec{w} \in C\}$$

Theorem 8.14. Let $C \subseteq F_q^n$. Then $dim(C^\perp) = n - dim(C)$.

Proof. @TODO □

Definition 8.25. A *parity check matrix* H for an $[n,k]-$code C is a generator matrix of $C^\perp$.

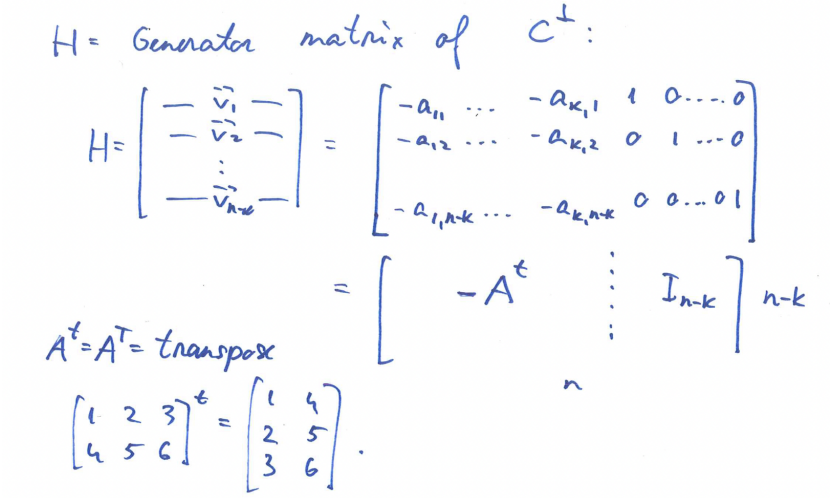

Figure 15: Computing the parity check matrix given the generator matrix for C

The parity check matrix satisfies $GH^T = 0$ (0 here refers to the all-zero matrix). This essentially means that every row of G is orthogonal to every row of the parity check matrix, which in turn implies that every vector in C is orthogonal to every vector in $C^\perp$, consistent with our definition above.

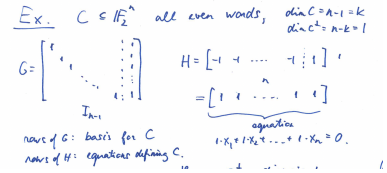

Figure 16: Examples

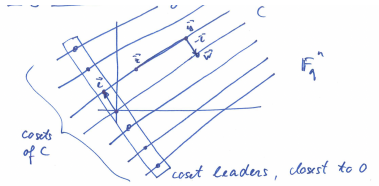

Figure 17: Visualizing cosets

Syndrome Decoding

Now, we covered decoding linear codes in the previous section using the standard array. However, we realized that finding the vector $\vec{y}$ in this table can be really, really slow when the tables get big. There is also the concern of memory - do we really need to have a table with every vector in our space (q^n entries) to decode a linear code? No.

Now we know that

$$\vec{x} \in C \iff H \cdot \vec{x} = \vec{0} \iff \vec{x} \cdot H^t = 0$$

Theorem 8.15. $\vec{y}, \vec{z}$ lie in the same coset $\iff \vec{y} \cdot H^t = \vec{z} \cdot H^t$

Proof.

$$\begin{aligned} \vec{y}, \vec{z} \text{ in the same coset} &\iff (\vec{y} - \vec{z}) \in C \\ &\iff (\vec{y} - \vec{z}) \cdot H^t = 0 \\ &\iff \vec{y} \cdot H^t = \vec{z} \cdot H^t \end{aligned}$$

□

Definition 8.26. The **syndrome** of $\vec{y}$ is $y \cdot H^t \in F_q^{n-k}$

Now we can use the syndrome of $\vec{y}$ to quickly find the coset that it belongs to.

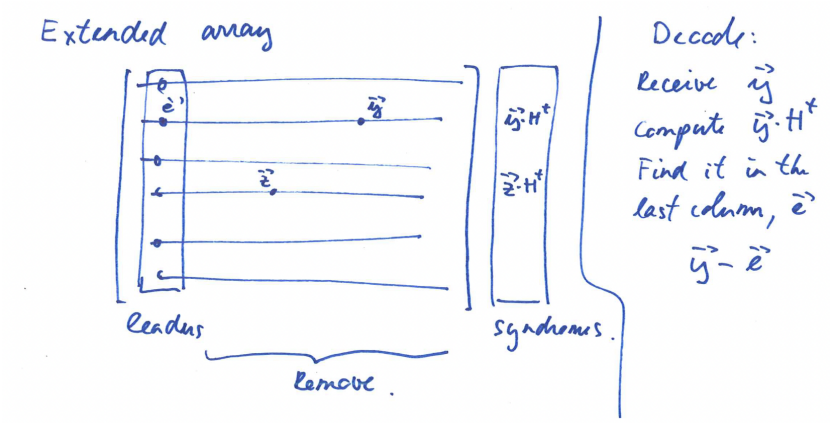

Figure 18: The extended array

Hamming Codes

The *Hamming codes* are a special family of single-error-correcting codes that are easy to encode and decode. They are linear codes and can be defined over any finite field $GF(q)$ but we restrict our study in this section binary Hamming codes.

Definition 9.27. Let r be a positive integer and let H be a $r \times 2^r - 1$ matrix whose columns are the distinct non-zero vectors in F_2^r. The code having H as its parity-check matrix is called a binary Hamming code and is denoted by $Ham(r, 2)$.

Notice that $Ham(r, 2)$ has length $n = 2^r - 1$ and dimension $k = n - r$. Also, since the columns of H may be taken in any order, the code $Ham(r, 2)$ gives one of a number of equivalent codes.

Theorem 9.16. The distance of $Ham(r, 2)$ is 3.

Proof. @TODO □

Theorem 9.17. The binary Hamming code is perfect.

Proof. Check sphere packing bound. □

Decoding with a binary Hamming code

Since $Ham(r, 2)$ is a perfect, single-error-correcting code (i.e distance 3), the coset leaders are precisely the $2^r (= n + 1)$ vectors of F_2^r of weight ≤ 1. This has a very important implication. Notice that the syndrome for the vector $(0 \cdots 10 \cdots 0)$ (i.e the vector with 1 in the jth position) is $(0 \cdots 10 \cdots 0)H^t$ which is just the transpose of the jth column of H. Hence, if the columns of H are arranged in increasing order, the syndrome of $\vec{y} \in Ham(r, 2)$ is just the error position.

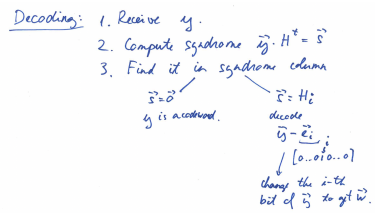

Figure 19: Decoding using the Hamming code

Non-binary Hamming Codes

Theorem 9.18. Let $H_1, \ldots, H_n$ be the columns of H. Then $d(C)$ is such that

1. There exists d linearly dependent columns of H.
2. Any $d - 1$ columns of H are linearly independent.

Proof. Suppose $\vec{w}$ is a codeword of weight l, i.e $\vec{w} = (0, \ldots, c_1, \ldots, c_2, \ldots, c_n)$. Then we have that

$$\vec{w} \cdot H^t = \vec{0} \iff c_1 H_1 + c_2 H_2 + \cdots + c_n H_n = \vec{0}$$

If there existed $d - 1$ linearly dependent columns, then we would arrive at a contradiction since that would imply the existence of a vector $\vec{y}$ that is of weight $< d(C)$ and is in C. □

And so, for a linear code C, $d(C)$ represents the minimum number of columns of the corresponding parity check matrix that are linearly dependent $\iff$ any $d - 1$ columns are linearly independent.

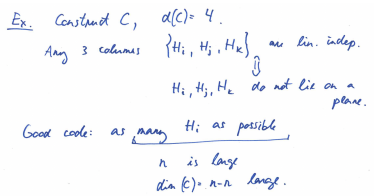

Figure 20: Example

q-ary Hamming Codes

With the theorem we just covered, it becomes easy to define the H matrix for a $q-ary$ Hamming code. We know that we need every pair of columns in our H matrix to be linearly independent. Given this condition, we now try to construct a code with the largest possible n (number of columns = length of our codeword).

Notice that any vector in F_q^n has $q-1$ scalar mutliples. There are q^r-1 non-zero vectors, and so we get $\frac{q^r-1}{q-1}$ equivalence classes. And so, if we choose one vector from each of these equivalence classes, we get the largest possible set of vectors such that any two are linearly independent. This code is called a $q-ary$ Hamming code and is denoted by $Ham(r,q)$.

Theorem 9.19. A $q-ary$ Hamming code $Ham(r,q)$ is perfect.

Proof. □

The decoding scheme of q-ary Hamming codes is similar to binary Hamming codes.

Cyclic Codes

Definition 9.28. A code C is cyclic when

1. it is linear
2. any cyclic shift of a codeword is also a codeword i.e whenever $a_0a_1\dots a_n$ is in C, then so is $a_na_0a_1\dots a_{n-1}$ and so on

Just like we used linear algebra to impose a structure on linear codes, we use polynomials to represent and study cyclic codes. We will see why this is useful in the following section.

Let C be a cyclic code. Define a codeword $w=(a_0,a_1,\dots,a_{n-1})$ in the C as

$$a_0+a_1x+\cdots+a_{n-1}x^{n-1}$$

a polynomial in $F_q[x]$. If we multiply this polynomial by x we get

$$xa_0+a_1x^2+\cdots+a_{n-1}x^n$$

Notice that this polynomial modulo x^n-1 is

$$xa_0+a_1x^2+\cdots+a_{n-1}$$

which, as a codeword, is $(a_{n-1},a_0,\dots a_{n-2})$ a cyclic shift! Just as $\mathbb{Z}$ is a (infinite) ring while $\mathbb{Z}_m$ is a finite ring, $F_q[x]$ is the ring of polynomials while $F_q[x]/x^n-1$ is a finite ring of polynomials. Now we can refine our definition of cyclic codes.

Definition 9.29. $F_q[x]$ is the ring of polynomials with coefficients belonging to the field $F_q[x]$. The ring $F_q[x]/f(x)$ consists of all the polynomial of degree less than $f(x)$. Notice that, if the degree of $f(x) = n$, then there are q^n such polynomials.

Definition 9.30. A cyclice code C is

$$C \subseteq \frac{F_q[x]}{x^n - 1}$$

Definition 9.31. A polynomial of degree m is called monic when the x^m has coefficient $a_m = 1$.

Addition, multiplication, scalar multiplication and division (remember long division for polynomials?) are carried out as usual for $F_q[x]$, but remember that now all coefficients are in the ring F_q.

Definition 9.32. The greatest common divisor of $f(x)$ and $g(x)$ is defined as the monic polynomial $h(x)$ of max degree that divides both f, g. Notice that such a monic polynomial always exists since F_q is a field, hence the inverse of a_m (coeffecient of the x^m term) has an inverse.

Remark 9.1. $\mathbb{Z}$, $F_q[x]$ are known as Euclidean rings because the Euclidean algorithm works in these spaces.

We cover the general procedure of finding the gcd of two polynomials in the ring $F_q[x]$. Say you have two polynomials $f, g \in F_q[x]$. Say $deg(f) < deg(g)$.

1. Divide f by g using long division. Note the remainder r and the quotient q.
2. By the Euclidean algorithm, we have
$$gcd(f, g) = gcd(f - q \cdot g, g) = gcd(r, g)$$
3. Repeat this procedure until you get the expression of the form
$$gcd(c, k) = c$$
where c is some constant and k is some polynomial.

Recall how we found inverses of elements in the field F_q using the Euclidean algorithm. Now that we know how to perform the Euclidean algorithm in the ring $F_q[x]$, we can use that to find inverses in it... wait, its a ring, inverses are not guranteed to exist. Hmm, does a field over polynomials exist? We cover this next.

Theorem 9.20. α is a root of a polynomial $f(x) \iff (x - \alpha)$ divides $f(x) \iff f(\alpha) = 0$.

Theorem 9.21. If $deg(f) = n$ then f can have at most n roots.

Proof. Because you can only have n factors of the form $(x - \alpha)$, since its degree n. □

Next is probably my favorite result of working in fields.

Remark 9.2. If we are working in F_q (i.e the field $GF(q)$), then we have that

$$(a + b)^p = a^p + b^p$$

Isn't this beautiful?

Alright, now back to defining the *field* of polynomials. Notice that when considering the ring of integers $\mathbb{Z}_k$ where $k \in \mathbb{Z}$, we only got fields when k was a prime number. With polynomials you get a symmetric result.

Definition 9.33. We say that the ring of polynomials $F_q[x]/g(x)$ is a field when $g(x)$ is **irreducible**. Notice that $F_q[x]/g(x)$ and $F_q[x]/f(x)$ where f is reducible and of the same degree as g have the same cardinality - they both contain all polynomials of degree less than f (or, equivalently, g)

Irreduciblility has a very natural definition.

Definition 9.34. We say that a polynomial g is irreducible when it is not a product of smaller degree polynommials. Notice that, if f is irreducible, then so is $c \cdot f$ for $c \neq 0$. Therefore, it is convention to state irreducible polynomials as monic (this is always possible in $F_q[x]$ by the field properties of F_q).

Example 9.5. Consider the polynomial $x^3 + x + 1$ in $\mathbb{Z}_2[x]$. Is it reducible? Well, if it were to be reducible, there would be one linear factor and one quadratic factor or 3 linear factors. Since we are in $\mathbb{Z}_2$, we check if $0, 1$ are roots.

$$0 + 0 + 1 = 1$$
$$1 + 1 + 1 \equiv 1$$

Hence, this polynomial has no linear factors and, by our analysis, is irreducible.

Recall the fundamental theorem of arithmetic - just like every integer can be stated as a product of primes, every polynomial can be stated as a product of irreducible polynomials.

Remark 9.3. Any two fields with the same order are isomorphic.

Definition 9.35. Every *field* has a primitive element.

Alright, having a rough idea of polynomial rings and fields, let's see how we can use them to make it easier to work with cyclic codes.

Definition 9.36. Just as linear codes are subspaces of F_q^n, all cyclic codes are subspaces of the space R_n where

$$R_n = F_q[x]/(x^n - 1)$$

Notice that the length our codewords is n. We have

$$R_n = F_q[x]/(x^n - 1) = \{a_0 + a_1 x + \cdots + a_{n-1}x^{n-1}\}$$

where each $a_i \in F_q$.

Also notice that R_n is isomorphic to F_q^n.

Theorem 9.22. A code C in R_n is cyclic if and only if C satisfies the following two conditions

1. $a(x), b(x) \in C \implies a(x) + b(x) \in C$ (closed under addition)
2. $a(x) \in C$ and $r(x) \in R_n \implies r(x)a(x) \in C$ (closed under multiplication, but only for elements in R_n[11])

Consider the set $\langle f(x)\rangle$ defined as

$$\langle f(x)\rangle = \{r(x)f(x) | r(x) \in R_n\}$$

The set $\langle f(x)\rangle$ is a cyclic code! $f(x)$ is called the **generating polynomial** of this code. **Any** cyclic code can be generated by some polynomial.[12]

Theorem 9.23. Let C be a non-zero cyclic code in R_n. Then we have that:

1. there exists a unique monic polynomial $g(x)$ of smallest degree in C
2. $C = \langle g(x)\rangle$
3. $g(x)$ is a factor of $x^n - 1$

Theorem 9.24. In a cyclic code, the monic polynomial of least degree ($g(x)$ given above) is called the generator polynomial.

Notice that by the theroems above, we can find all cyclic codes of arbitrary length.

Example 9.6. Let's say that you want to find all the cyclic codes of length 3 in $F_2[x]$. To do so, we need to find as many distinct generator polynomials as we can. Notice that, since the generator polynomial divides $x^3 - 1$ and is irreducible, all we need to do is find the factorization of $x^3 - 1$ in terms of irreducibles. The number of distinct factors is the maximum number of cyclic codes of length 3.

[11] In ring theory term, we are saying that cyclic codes are precisely the ideals of the ring R_n

[12] In ring theory terminology, this says that every ideal of R_n is a principal ideal

Theorem 9.25. Suppose C is a cyclic code with generator polynomial

$$g(x) = g_0 + g_1 x + \cdots + g_r x^r$$

of degree r. Then, we can define the generator matrix of C to be

$$G = \begin{bmatrix} g_0 & g_1 & g_2 & \cdots & & g_r & 0 & 0 & \cdots & 0 \\ 0 & g_0 & g_1 & g_2 & \cdots & & g_r & 0 & \cdots & 0 \\ 0 & 0 & g_0 & g_1 & g_2 & \cdots & & g_r & & \vdots \\ \vdots & \vdots & & \ddots & \ddots & \ddots & \ddots & & \ddots & 0 \\ 0 & 0 & \cdots & 0 & g_0 & g_1 & g_2 & \cdots & & g_r \end{bmatrix}$$

Figure 21: Generator matrix for a cyclic code with generator polynomial $g(x)$