THEORY OF ERROR CORRECTING CODES MT461/MT5461

MARK WILDON

These notes cover the part of the syllabus for MT461/MT5461 that is not part of MT361. Further installments will be issued as they are ready. All handouts and problem sheets will be put on the MT361 Moodle page, marked **MSc/MSci**.

I would very much appreciate being told of any corrections or possible improvements to these notes.

You are warmly encouraged to ask questions in lectures, and to talk to me after lectures and in my office hours. I am also happy to answer questions about the lectures or problem sheets by email. My email address is `mark.wildon@rhul.ac.uk`.

Lecture times: Monday 3pm (MFLEC), Tuesday 3pm (BLT2) and Thursday 10am (BLT2).

Extra lecture for MT461/MT5461: Thursday noon (ABLT3).

Office hours in McCrea 240: Tuesday 11am, Thursday 2pm and Friday 11am.

Date: Second term 2012/13.

OVERVIEW

The extra content on the syllabus for MT5461 is on Reed–Solomon codes and cyclic codes over finite fields. These codes are examples of the linear codes that will be covered in Part C of the main lectures.

We will first look at the original definition of Reed–Solomon codes, and see one efficient decoding algorithm. Then in the second half of the term we will look at cyclic codes in general, and make the connection with Reed–Solomon codes. We will end by defining the important family of BCH codes.

1. REVISION OF FIELDS AND POLYNOMIALS

Most good codes make use of the algebraic structure of finite fields and polynomial rings. For example, the Reed–Solomon code used on compact discs has as its alphabet the finite field of order 2^8. This is a convenient choice because it means that each symbol fits exactly into a single 8-bit byte on a computer.

This section should give enough background for most of the course, but if you have not seen finite fields of prime power order then you will have to take one or two results on trust towards the end.[1] Proofs in this section are non-examinable. The examination will only require finite fields of prime order.

FIELDS. A field is a set in which one can add, subtract and multiply any two elements, and also divide by non-zero elements. Examples of familiar infinite fields are the rational numbers $\mathbf{Q}$ and the real numbers $\mathbf{R}$. If p is a prime, then the set $\mathbf{F}_p = \{0, 1, \ldots, p-1\}$, with addition and multiplication defined modulo p is a finite field: see Theorem 1.3.

Definition 1.1. A *field* is a set of elements $\mathbf{F}$ with two operations, $+$ (addition) and $\times$ (multiplication), and two special elements $0, 1 \in \mathbf{F}$ such that $0 \neq 1$ and

(1) $a + b = b + a$ for all $a, b \in \mathbf{F}$;
(2) $0 + a = a + 0 = a$ for all $a \in \mathbf{F}$;
(3) for all $a \in \mathbf{F}$ there exists $b \in \mathbf{F}$ such that $a + b = 0$;
(4) $a + (b + c) = (a + b) + c$ for all $a, b, c \in \mathbf{F}$;

[1] It is interesting that Golay, who was the first to publish the family of codes that generalize the Hamming code of length 7, knew only about fields of prime order when he wrote his first two important papers. See T. M. Thompson, *From error correcting codes through sphere packing to simple groups*, Mathematical Association of America, 1983, page 43.

(5) $a \times b = b \times a$ for all $a, b \in \mathbf{F}$;
(6) $1 \times a = a \times 1 = a$ for all $a \in \mathbf{F}$;
(7) for all non-zero $a \in \mathbf{F}$ there exists $b \in \mathbf{F}$ such that $a \times b = 1$;
(8) $a \times (b \times c) = (a \times b) \times c$ for all $a, b, c \in \mathbf{F}$;
(9) $a \times (b + c) = a \times b + a \times c$ for all $a, b, c \in \mathbf{F}$.

If $\mathbf{F}$ is finite, then we define its *order* to be its number of elements.

It may be helpful to note that (1)–(4) imply that $\mathbf{F}$ is an abelian group under addition, and that (5)–(8) imply that $(\mathbf{F}\backslash\{0\}, \times)$ is an abelian group under multiplication. The final axiom (9) is the *distributive law* relating addition and multiplication.

It is usual to write $-a$ for the element b in (4); we call $-a$ the *additive inverse* of a. We write a^{-1} for the element b in (8); we call a^{-1} the *multiplicative inverse* of a. We usually write ab rather than $a \times b$.

Exercise: Show, from the field axioms, that if $x \in \mathbf{F}$, then x has a unique additive inverse, and that if $x \neq 0$ then x has a unique multiplicative inverse. Show also that if $\mathbf{F}$ is a field then $a \times 0 = 0$ for all $a \in \mathbf{F}$.

Exercise: Show from the field axioms that if $\mathbf{F}$ is a field and a, $b \in \mathbf{F}$ are such that $ab = 0$, then either $a = 0$ or $b = 0$.

We will use the second exercise above many times in the following sections.

Theorem 1.2. *Let p be a prime. The set $\mathbf{F}_p = \{0, 1, \ldots, p-1\}$ with addition and multiplication defined modulo p is a finite field of order p.*

In fact there is a unique (up to a suitable notion of isomorphism) finite field of any given prime-power order. The smallest field not of prime order is the finite field of order 4.

Example 1.3. The addition and multiplication tables for the finite field $\mathbf{F}_4 = \{0, 1, \alpha, 1+\alpha\}$ of order 4 are shown below.

$+$	0	1	α	$1+\alpha$
0	0	1	α	$1+\alpha$
1	1	0	$1+\alpha$	α
α	α	$1+\alpha$	0	1
$1+\alpha$	$1+\alpha$	α	1	0

$\times$	1	α	$1+\alpha$
1	1	α	$1+\alpha$
α	α	$1+\alpha$	1
$1+\alpha$	$1+\alpha$	1	α

Probably the most important thing to realise is that $\mathbf{F}_4$ **is not the integers modulo** 4. Indeed, in $\mathbf{Z}_4 = \{0,1,2,3\}$ we have $2 \times 2 = 0$, but if $a \in \mathbf{F}_4$ and $a \neq 0$ then $a \times a \neq 0$, as can be seen from the multiplication table. (Alternatively this follows from the second exercise above.)

POLYNOMIALS. Let $\mathbf{F}$ be a field. Let $\mathbf{F}[x]$ denote the set of all polynomials

$$f(x) = a_0 + a_1 x + a_2 x^2 + \cdots + a_m x^m$$

where $m \in \mathbf{N}_0$ and $a_0, a_1, a_2, \ldots, a_m \in \mathbf{F}$.

Definition 1.4. If $f(x) = a_0 + a_1 x + a_2 x^2 + \cdots + a_m x^m$ where $a_m \neq 0$, then we say that m is the *degree* of the polynomial $f(x)$, and write $\deg f = m$. We define the degree of the *zero polynomial* $f(x) = 0$ to be -1.

Note that a polynomial is a non-zero constant if and only if it has degree 0. The degree of the zero polynomial is not entirely standardized: you might also see it defined to be $-\infty$, or left undefined. For this reason we will phrase some results, such as Lemma 1.5 below, so that the whole issue is avoided.

Polynomials are added and multiplied in the natural way.

Lemma 1.5 (Division algorithm)**.** *Let* $\mathbf{F}$ *be a field, let* $f(x) \in \mathbf{F}[x]$ *be a non-zero polynomial and let* $g(x) \in \mathbf{F}[x]$*. There exist polynomials* $s(x), r(x) \in \mathbf{F}[x]$ *such that*

$$g(x) = s(x)f(x) + r(x)$$

and either $r(x) = 0$ *or* $\deg r(x) < \deg f(x)$*.*

We say that $s(x)$ is the *quotient* and $r(x)$ is the *remainder* when $g(x)$ is divided by $f(x)$. Lemma 1.5 will not be proved in lectures. The important thing is that you can find the quotient and remainder in practice.

Exercise: Let $f(x) = x^3 + x + 1 \in \mathbf{F}_2[x]$, let $g(x) = x^5 + x^2 + x \in \mathbf{F}_2[x]$. Find the quotient and remainder when $g(x)$ is divided by $f(x)$.

For Reed–Solomon codes we shall need the following standard properties of polynomials.

Lemma 1.6. *Let* $\mathbf{F}$ *be a field.*

(i) *If* $f(x) \in \mathbf{F}[x]$ *has* $a \in \mathbf{F}$ *as a root, i.e.* $f(a) = 0$*, then there is a polynomial* $g(x) \in \mathbf{F}[x]$ *such that* $f(x) = (x - a)g(x)$*.*

(ii) *If* $f(x) \in \mathbf{F}[x]$ *has degree* $m \in \mathbf{N}_0$ *then* $f(x)$ *has at most* m *distinct roots in* $\mathbf{F}$*.*

(iii) *Suppose that* f*,* $g \in \mathbf{F}[x]$ *are non-zero polynomials such that* $\deg f$*,* $\deg g < k$*. If there exist distinct* $c_1, \ldots, c_k \in \mathbf{F}$ *such that* $f(c_i) = g(c_i)$ *for each* $i \in \{1, \ldots, k\}$ *then* $f(x) = g(x)$*.*

We also need a result on polynomial interpolation that has a surprisingly quick direct proof.

Lemma 1.7 (Polynomial interpolation)**.** *Let* $\mathbf{F}$ *be a field. Let*

$$c_1, c_2, \ldots, c_k \in \mathbf{F}$$

be distinct and let $y_1, y_2, \ldots, y_k \in \mathbf{F}$*. The unique polynomial* $f(x) \in \mathbf{F}[x]$ *of degree* $< k$ *such that* $f(c_i) = y_i$ *for all* i *is*

$$f(x) = \sum_{i=1}^{k} y_i \frac{\prod_{j \neq i}(x - c_j)}{\prod_{j \neq i}(c_i - c_j)}.$$

Finally, the following result will turn out to control the number of errors a large class of cyclic codes can detect and correct. A short proof is given below for interest and logical completeness.

Theorem 1.8. *Let* $\mathbf{F}$ *be a field and let* $c_1, \ldots, c_k \in \mathbf{F}$ *be distinct. Then the columns of the* $k \times k$ *matrix*

$$\begin{pmatrix} c_1 & c_2 & \ldots & c_k \\ c_1^2 & c_2^2 & \ldots & c_k^2 \\ \vdots & \vdots & \ddots & \vdots \\ c_1^k & c_2^k & \ldots & c_k^k \end{pmatrix}$$

are linearly independent.

Proof. Let M be the matrix. The columns of M are linearly independent if and only if the rows of M are linearly independent, since either condition is equivalent to M having full rank k. Let r_i denote row i of M and suppose that there exist $t_1, t_2, \ldots, t_k \in F$ such that

$$t_1 r_1 + t_2 r_2 + \cdots + t_k r_k = 0.$$

Looking at position j on either side of this equation we get

$$t_1 c_j + t_2 c_j^2 + \cdots + t_k c_j^k = 0.$$

This holds for each $j \in \{1, \ldots, k\}$, so the polynomial

$$f(x) = t_1 x + t_2 x^2 + \cdots + t_k x^k$$

has roots at $c_1, c_2, \ldots, c_k$, and also, at 0. But a non-zero polynomial of degree at most k has at most k roots, by Lemma 1.6(ii). Hence $f(x) = 0$ and $t_1 = t_2 = \cdots = t_k = 0$. □

2. Definition and basic properties of Reed–Solomon codes

In this section we give the original definition of Reed–Solomon codes over finite fields.[2] We will work over the fields $\mathbf{F}_p$ of prime degree given by Theorem 1.2, but it is easy to see that the definition and all the results extend to a general finite field; this will be assumed in some examples.

Definition 2.1. Let p be a prime and let $k, n \in \mathbf{N}$ be such that $k \leq n \leq p$. Let

$$a_1, a_2, \ldots, a_n$$

be distinct elements of $\mathbf{F}_p$. For each polynomial $f(x) \in \mathbf{F}_p[x]$ we define a word $u(f) \in \mathbf{F}_p^n$ by

$$u(f) = (f(a_1), f(a_2), \ldots, f(a_n)).$$

The *Reed–Solomon code* associated to the parameters p, n, k and the field elements $a_1, a_2, \ldots, a_n$ is the length n code over $\mathbf{F}_p$ with codewords

$$\{u(f) : f \in \mathbf{F}_p[x],\ \deg f \leq k-1\}.$$

It is worth bearing in mind Remark 1.7(3) from the main lectures: the parameters p, n, k and the field elements $a_1, a_2, \ldots, a_n$ are part of the specification of a Reed–Solomon code, and should be assumed to be known to everyone. We will write $RS_{p,n,k}$ for the code defined in Definition 2.1.

For instance, the Reed–Solomon codes used on compact discs have alphabet the finite field $\mathbf{F}_{2^8}$. One has parameters $n = 32$ and $k = 28$, the other $n = 28$ and $k = 24$.

[2] See I. S. Reed and G. Solomon, *Polynomial codes over certain finite fields*, SIAM 8 (1960) 300–304.

Example 2.2. Let $p = 5$ and let $k = 2$.

(1) If $n = 3$ and we take $a_1 = 0$, $a_2 = 1$ and $a_3 = 2$, then the associated Reed–Solomon code has a codeword

$$(f(0), f(1), f(2))$$

for each $f(x) \in \mathbf{F}_p[x]$ of degree ≤ 1. If $f(x) = bx + c$ then

$$u(f) = (c, b + c, 2b + c)$$

so the full set of codewords is

$$\{(c, b + c, 2b + c) : b, c \in \mathbf{F}_5\}.$$

(2) If $n = 4$ and we take a_1, a_2, a_3 as before, and $a_4 = 3$ then we get an extension of the code in (1). The set of codewords is

$$\{(c, b + c, 2b + c, 3b + c) : b, c \in \mathbf{F}_5\}.$$

This code has the same size as the previous code, but longer length, so one might expect it to have better error-detecting and error-correcting properties.

The next exercise will be subsumed by more general results proved using the theory developed in Part A of the main course. But it is very instructive to find a direct proof. (The following exercise is the final part of Question 9 on Sheet 2.)

Exercise: Let C be the Reed–Solomon code in Example 2.2(2). Show that if $u \in C$ is sent down a noisy channel, and v is received such that $d(u, v) \leq 2$ then either $v = u$ or $v \notin C$. Can the receiver guarantee to detect if three errors occur?

Possibly you have realised that Hamming distances between codewords in a code controls how many errors the code can detect and correct. The next lemma gives a lower bound on these distances for the Reed–Solomon code.

Lemma 2.3. *If $f, g \in \mathbf{F}_p[x]$ are distinct polynomials of degree $\leq k - 1$ then*

$$d(u(f), u(g)) \geq n - k + 1.$$

An immediate corollary of Lemma 2.3 is that if f, g are distinct polynomials of degree $\leq k - 1$ then $u(f) \neq u(g)$. A short counting argument now proves the next lemma.

Lemma 2.4. *The Reed–Solomon code $RS_{p,n,k}$ has size p^k.*

By Lemma 2.3, the minimum distance (as defined in Definition 4.1 of the main notes) of $RS_{p,n,k}$ is at least $n-k+1$. To prove the next theorem we will use polynomial interpolation to find two codewords in $RS_{p,n,k}$ at distance $n-k+1$.

Theorem 2.5. *The minimum distance of $RS_{p,n,k}$ is $n-k+1$.*

The Singleton Bound (to be proved in Part B of the main course) states that a code of length n and minimum distance d over an alphabet of size p has at most p^{n-d+1} codewords. By Lemma 2.3 and Theorem 2.5, the Reed–Solomon codes meet this bound, and so have the largest possible size for their length and minimum distance.

Corollary 2.6. *Let p be a prime. If $k, e \in \mathbf{N}$ are such that $k+2e \leq p$ then the Reed–Solomon code $RS_{p,k+2e,k}$ is e-error correcting.*

We now discuss encoding and decoding for Reed–Solomon codes. The code $RS_{p,n,k}$ has size p^k so it can encode p^k different messages. Given any $(b_1, b_2, \ldots, b_k) \in \mathbf{F}_p^k$, we can use polynomial interpolation to find a polynomial $f(x) \in \mathbf{F}_p[x]$ of degree $< k$ such that $f(a_i) = b_i$ for $1 \leq i \leq k$, and so

$$u(f) = (b_1, b_2, \ldots, b_k, \ldots).$$

Hence if we agree to identify messages with $\mathbf{F}_p^k$, then we can use polynomial interpolation to define a suitable encoder. (One advantage of this encoder is that if a message is received without any errors, then the message can be read off from the first k positions.)

Decoding is a much trickier issue. We would like to use nearest neighbour decoding, but the naïve algorithm where we search through the entire code looking for the nearest codeword is completely impractical once p and k are large.

For example, the larger Reed–Solomon codes used on compact discs has $k = 28$ and alphabet $\mathbf{F}_{2^8}$, so has size $(2^8)^{28} = 2^{224}$. Since a cipher is often considered cryptographically secure if the only attacks require a hunt through 2^{128} different keys, the naïve algorithm has no chance.

The original decoder proposed by Reed and Solomon used polynomial interpolation to find all codewords that agreed with the received word in at least k positions. The nearest one would then be taken as the sent codeword. (So the decoder implements nearest neighbour decoding.)

Example 2.7. Suppose we use the Reed–Solomon code with $p = 5$, $n = 4$ and $k = 2$ evaluating at $a_1 = 0$, $a_2 = 1$, $a_3 = 2$, $a_4 = 3$, as in Example 2.2(2). By Corollary 2.6, this code is 1-error correcting. Suppose we receive $v = (4, 0, 3, 0)$.

Given any two positions i and j, it follows from Lemma 1.7 that there is a unique polynomial g of degree < 2 such that $g(a_i) = v_i$ and $g(a_j) = v_j$.

The table below shows the interpolating polynomials for each pair of positions and the corresponding codewords. For example, to find $f(x)$ such that $f(0) = 4$ and $f(2) = 3$, we use Lemma 1.7 and get

$$f(x) = 4\frac{x-2}{0-2} + 3\frac{x-0}{2-0} = 3(x-2) - x = 2x + 4.$$

Conditions on f	Solution	Codeword $u(f)$
$f(0) = 4, f(1) = 0$	$f(x) = 4 + x$	$(4, 0, 1, 2)$
$f(0) = 4, f(2) = 3$	$f(x) = 4 + 2x$	$(4, 1, 3, 0)$
$f(1) = 0, f(2) = 3$	$f(x) = 2 + 3x$	$(2, 0, 3, 1)$
$f(0) = 4, f(3) = 0$	$f(x) = 4 + 2x$	$(4, 1, 3, 0)$
$f(1) = 0, f(3) = 0$	$f(x) = 0$	$(0, 0, 0, 0)$
$f(2) = 3, f(3) = 0$	$f(x) = 4 + 2x$	$(4, 1, 3, 0)$

In practice, we would stop as soon as we found the codeword $(4, 1, 3, 0)$ since $d(4130, 4030) = 1$, and by Question 6 on Sheet 2, there is at most one codeword within distance 1 of any given word.

There are $\binom{n}{k}$ choices of k positions from n. For example, $\binom{32}{28} = 35960$. So while a big improvement over the naïve algorithm, polynomial interpolation is still impractical in cases of interest.

3. Efficient decoding of Reed–Solomon codes

In this section we shall see an efficient algorithm for decoding Reed–Solomon codes invented by Berlekamp and Welch in 1983.[3] As usual we work with the Reed–Solomon code $RS_{p,n,k}$ where p is prime and n, $k \in \mathbf{N}$, and polynomials are evaluated at $a_1, a_2, \ldots, a_n$. Assume that $n = k + 2e$, so by Corollary 2.6 the code is e-error correcting.

[3] L. Welch and E. R. Berlekamp, Error correction for algebraic block codes, U.S. Patent 4 633 470 (1983).

Definition 3.1 (Key Equation)**.** The *Key Equation* for the received word $(v_1, v_2, \ldots, v_n)$ is

$$Q(a_i) = v_i E(a_i)$$

where $Q(x)$, $E(x) \in \mathbf{F}_p[x]$ are non-zero polynomials such that

- $\deg Q(x) \leq k + e - 1$
- $\deg E(x) \leq e$.

The polynomial $E(x)$ is called the *error locator polynomial*.

Here is a small observation that helps to motivate the Key Equation.

Example 3.2. Suppose that $u(f)$ is sent and that a single error occurs in position j. Then $f(a_i) = v_i$ at all $i \neq j$. If we put in an extra term $x - a_j$ to 'hide' the error in position j, then the equation

$$f(x)(x - a_j) = v_i(x - a_j)$$

holds when we replace x with any a_i. So $Q(x) = f(x)(x - a_j)$, $E(x) = x - a_j$ is a solution to the Key Equation. Note that $f(x) = Q(x)/E(x)$ and the root of $E(x)$ tell us which position of v is in error.

The main theorem on the Key Equation is as follows.

Theorem 3.3. *Suppose that $u(f)$ is sent and that errors occur in positions $j_1, j_2, \ldots, j_t$ where $t \leq e$. Let v be the received word. Then $Q(x)$, $E(x)$ solve the Key Equation if and only if*

(i) *$E(x) = (x - a_{j_1})(x - a_{j_2}) \ldots (x - a_{j_t})s(x)$ for some polynomial $s(x)$ such that $\deg s(x) \leq e - t$, and*
(ii) *$Q(x) = E(x)f(x)$.*

Theorem 3.3 characterises all solutions to the Key Equation when at most e errors occur. However it does not help us to solve it in practice. For this we proceed by solving linear equations. Unlike the naïve approach in §2, we only have to solve one system, not $\binom{n}{e}$ separate systems! [**misprinted as** $\binom{n}{k}$ in original version]

Lemma 3.4. *Suppose that the word $(v_1, \ldots, v_n)$ is received. The polynomials*

$$Q(x) = Q_0 + Q_1 x + \cdots + Q_{k+e-1} x^{k+e-1}$$
$$E(x) = E_0 + E_1 x + \cdots + E_e x^e$$

in $\mathbf{F}_p[x]$ satisfy the Key Equation if and only if

$$Q_0 + a_i Q_1 + a_i^2 Q_2 + \cdots + a_i^{k+e-1} Q_{k+e-1} = v_i(E_0 + a_i E_1 + a_i^2 E_2 + \cdots + a_i^e E_e)$$

for each $i \in \{1, \ldots, n\}$. An equivalent condition is that

$$\begin{pmatrix} 1 & a_1 & a_1^2 & \cdots & a_1^{k+e-1} & -v_1 & -v_1 a_1 & \cdots & -v_1 a_1^e \\ 1 & a_2 & a_2^2 & \cdots & a_2^{k+e-1} & -v_2 & -v_2 a_2 & \cdots & -v_2 a_2^e \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & a_n & a_n^2 & \cdots & a_n^{k+e-1} & -v_n & -v_n a_n & \cdots & -v_n a_n^e \end{pmatrix} \begin{pmatrix} Q_0 \\ Q_1 \\ \vdots \\ Q_{k+e-1} \\ E_0 \\ E_1 \\ \vdots \\ E_e \end{pmatrix} = 0$$

The matrix above has n rows and $k + 2e + 1 = n + 1$ columns. So we can solve the Key Equation by solving an $n \times (n + 1)$ system of linear equations. Notice that this system always has a non-zero solution.

Example 3.5. We shall use the code of Example 2.2(2) and Example 2.7. Let $p = 5$, let $k = 2$, let $e = 1$ (so $n = 4$) and let $a_1 = 0$, $a_2 = 1$, $a_3 = 2$, $a_4 = 3$. With these parameters, the Key Equation for the polynomials $Q(x) = Q_0 + Q_1 x + Q_2 x^2$ and $E(x) = E_0 + E_1 x$ is

$$\begin{pmatrix} 1 & 0 & 0 & 4v_1 & 0 \\ 1 & 1 & 1 & 4v_2 & 4v_2 \\ 1 & 2 & 4 & 4v_3 & 3v_3 \\ 1 & 3 & 4 & 4v_4 & 2v_4 \end{pmatrix} \begin{pmatrix} Q_0 \\ Q_1 \\ Q_2 \\ E_0 \\ E_1 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}.$$

(1) Suppose we receive the word 4130. (This is the codeword for $f(x) = 4 + 2x$.) Then $v_1 = 4$, $v_2 = 1$, $v_3 = 3$, $v_4 = 0$ and we must solve

$$\begin{pmatrix} 1 & 0 & 0 & 1 & 0 \\ 1 & 1 & 1 & 4 & 4 \\ 1 & 2 & 4 & 2 & 4 \\ 1 & 3 & 4 & 0 & 0 \end{pmatrix} \begin{pmatrix} Q_0 \\ Q_1 \\ Q_2 \\ E_0 \\ E_1 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}.$$

The kernel is two dimensional, spanned by the vectors

$$(0, 4, 2, 0, 1)^t, \quad (4, 2, 0, 1, 0)^t.$$

The first vector gives $Q(x) = 4x + 2x^2$ and $E(x) = x$, so we decode using $f(x) = Q(x)/E(x) = 4 + 2x$ to get $u(f) = 4130$. (The second vector gives the same answer even more quickly.)

(2) Suppose we receive the word 4030. Then $v_1 = 4$, $v_2 = 0$, $v_3 = 3$, $v_4 = 0$ and we must solve

$$\begin{pmatrix} 1 & 0 & 0 & 1 & 0 \\ 1 & 1 & 1 & 0 & 0 \\ 1 & 2 & 4 & 2 & 4 \\ 1 & 3 & 4 & 0 & 0 \end{pmatrix} \begin{pmatrix} Q_0 \\ Q_1 \\ Q_2 \\ E_0 \\ E_1 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}.$$

The kernel is one dimensional spanned by $(1, 2, 2, 4, 1)^t$. So we take $Q(x) = 1 + 2x + 2x^2$ and $E(x) = 4 + x$. Polynomial division gives

$$Q(x)/E(x) = 2x + 4$$

so we decode using $f(x) = 2x + 4$ to get $u(f) = 4130$.

(3) Finally suppose we receive 4020. Then the kernel is one dimensional, spanned by $(4, 3, 3, 1, 0)^t$. So we take $Q(x) = 4 + 3x + 3x^2$ and $E(x) = 1$, but $Q(x)/E(x)$ does not have degree ≤ 1, so we are unable to decode. Since the Key Equation method always works when $\leq e$ errors occur, we know that ≥ 2 errors have occurred, but we are unable to correct them.

When more than e errors occur it can also happen that the received word is decoded incorrectly. For example, this would happen in the setup of Example 3.4 if we received 4000. Another possibility is that $E(x)$ does not divide $Q(x)$: in this case we detect an error but are unable to correct it.

Final remarks: (1) When preparing this section I used §4 of these notes: `http://math.berkeley.edu/~mhaiman/math55/reed-solomon.pdf`.

The next section in Haiman's notes explains a refinement to the Key Equation method that reduces the problem to solving an $e \times e$ system of equations and then doing a one-off polynomial interpolation. Since e is usually much less than n (for the codes used on compact discs, $n = 28$ and $n = 32$ and $e = 4$ in both cases) this is a further big improvement.

(2) A MATHEMATICA notebook for solving the Key Equation is available on Moodle. Unless you really want to do it by hand, I suggest you use it (or another computer algebra program) to do the computational questions on problem sheets.

Part 2: Cyclic codes

4. Cyclic codes

In the second part of the course we will study cyclic codes. In this section we will introduce cyclic codes and see some examples and a fast way to encode messages for a general cyclic code. We will then show that a special class of Reed–Solomon codes are cyclic, and, if time permits, see a fast decoder for these codes that uses their cyclic structure. The course ends with BCH-codes; these are the binary cyclic codes most used in practice.

Cyclic codes are a special type of linear code. In Part C of the main course we only consider linear codes over the binary alphabet, but all the results extend to a general finite field. Here we will work over a finite field $\mathbf{F}_p$ of prime order.

Definition 4.1. Let p be prime. A code C over $\mathbf{F}_p$ is *linear* if

(i) for all $u \in C$ and $a \in \mathbf{F}_p$ we have $au \in C$;
(ii) for all $u, w \in C$ we have $u + w \in C$.

Here au is the word defined by $(au)_i = au_i$ and $u + v$ is defined by $(u + w)_i = u_i + w_i$. Equivalently, a code C over $\mathbf{F}_p$ is linear if C is a vector subspace of $\mathbf{F}_p^n$.

Exercise: Show that any Reed–Solomon code is linear.

Definition 4.2. Let p be a prime. A code C over $\mathbf{F}_p$ is said to be *cyclic* if C is linear and

$$(u_0, u_1, \ldots, u_{n-1}) \implies (u_{n-1}, u_0, \ldots, u_{n-2}) \in C.$$

We say that $(u_{n-1}, u_0, \ldots, u_{n-2})$ is the *cyclic shift* of $(u_0, u_1, \ldots, u_{n-1})$.

The reason for numbering positions from 0 will be seen shortly. Note that we can apply the cyclic shift several times over, so a cyclic code is closed under arbitrarily many cyclic shifts.

Example 4.3.

(1) Let p be prime and let $n \in \mathbf{N}$. The repetition code of length n over $\mathbf{F}_p$ is cyclic.

(2) Let C be the binary parity check code of length n consisting of all binary words of length n with evenly many 1s. We may define C using addition in $\mathbf{F}_2$ by

$$C = \{(u_0, \ldots, u_{n-1}) : u_i \in \mathbf{F}_2, u_0 + \cdots + u_{n-1} = 0\}.$$

Then C is a cyclic code.

(3) Let D be the binary code with codewords

$$\{000000, 110110, 011011, 101101\}.$$

Exercise: check that D is linear. The shift map acts on D by fixing 000000 and cyclically permuting the other three codewords. Hence D is cyclic.

There is a very helpful correspondence between codewords in a cyclic code and polynomials. To see how this might work, consider the code D in Example 4.3(3) above. If we associate the polynomial

$$u_0 + u_1x + u_2x^2 + u_3x^3 + u_4x^4 + u_5x^5$$

to the codeword $(u_0, u_1, u_2, u_3, u_4, u_5)$ then we have

$$\begin{aligned} 000000 &\longleftrightarrow 0 \\ 110110 &\longleftrightarrow 1 + x + x^3 + x^4 \\ 011011 &\longleftrightarrow x + x^2 + x^4 + x^5 \\ 101101 &\longleftrightarrow 1 + x^2 + x^3 + x^5. \end{aligned}$$

When we multiply $1 + x + x^3 + x^4$ by x we get $x + x^2 + x^4 + x^5$, which is the polynomial corresponding to 011011. So far so good! But when we multiply $x + x^2 + x^4 + x^5$ we get $x^2 + x^3 + x^5 + x^6$, which is not the polynomial we choose to correspond to 101101. Somehow we need to make $x^2 + x^3 + x^5 + x^6$ correspond to 101101, as well.

Definition 4.4. Let $n \in \mathbf{N}$ and let p be prime. Let $f(x) \in \mathbf{F}_p[x]$. We say that $f(x)$ *corresponds* to $(u_0, u_1, \ldots, u_{n-1}) \in \mathbf{F}_p^n$ if, when $f(x)$ is divided by $x^n - 1$, the remainder is

$$u_0 + u_1x + u_2x^2 + \ldots + u_{n-1}x^{n-1}.$$

Note, in particular, that if $f(x) = u_0 + u_1x + u_2x^2 + \cdots + u_{n-1}x^{n-1}$ then, when we divide $f(x)$ by $x^n - 1$, the quotient is 0 and the remainder is $f(x)$. So $f(x)$ corresponds to the word $(u_0, u_1, \ldots, u_{n-1}) \in \mathbf{F}_p^n$.

Exercise: Check that, when $n = 6$, the polynomial $x^2 + x^3 + x^5 + x^6$ corresponds to $(1, 0, 1, 1, 0, 1)$. Find the word corresponding to the polynomial $x(1 + x^2 + x^3 + x^5)$.

The following remark should be helpful to people who know about quotient rings, but should be skipped by everyone else.

Remark 4.5. Definition 4.4 defines a map from $\mathbf{F}_p[x]$ to words in $\mathbf{F}_p^n$. Two polynomials have the same image if and only if they have the same remainder on division by $x^n - 1$. So the polynomials that map to the word $(u_0, u_1, \ldots, u_{n-1})$ are exactly the elements of the coset

$$u_0 + u_1 x + \cdots + u_{n-1} x^{n-1} + \langle x^n - 1 \rangle$$

in the quotient ring $\mathbf{F}_p[x]/\langle x^n - 1\rangle$. Thus Definition 4.4 defines a bijection $\mathbf{F}_p[x]/\langle x^n - 1\rangle \longleftrightarrow \mathbf{F}_p^n$. The rest of this section is summarized in this footnote.[4]

Definition 4.4 was made so that the following lemma would hold.

Lemma 4.6. *Let p be a prime and let C be a cyclic code over $\mathbf{F}_p$. Let $u \in C$.*

(i) *Suppose that $f(x) \in \mathbf{F}_p[x]$ corresponds to u. Then $xf(x)$ corresponds to the cyclic shift of u, namely $(u_{n-1}, u_0, \ldots, u_{n-2})$.*
(ii) *If $s(x) \in \mathbf{F}_p[x]$ then $s(x)f(x)$ corresponds to a codeword in C.*

It will often be useful to think of a cyclic code as a set of polynomials of degree $< n$, by choosing the obvious polynomial $u_0 + u_1 x + \cdots + u_{n-1} x^{n-1}$ to correspond to the codeword $(u_0, u_1, \ldots, u_{n-1})$.

Definition 4.7. Let p be a prime. Let C be a cyclic code of length n over the finite field $\mathbf{F}_p$. Think of C as a set of polynomials of degree $< n$. A *generator polynomial* for C is a polynomial $g(x) \in \mathbf{F}_p[x]$ of degree $k < n$ such that $g(x)$ divides $x^n - 1$ and

$$C = \{f(x)g(x) : f(x) \in \mathbf{F}_p[x], \deg f(x) \leq n - 1 - k\}.$$

Example 4.8. Let

$$C = \{0, 1 + x + x^3 + x^4, x + x^2 + x^4 + x^5, 1 + x^2 + x^3 + x^5$$

be the polynomial version of the code in Example 4.2(3). We shall show that $g(x) = 1 + x + x^3 + x^4$ is a generator polynomial for C.

[4] Lemma 4.6 says that multiplication by x in $\mathbf{F}_p[x]/\langle x^n - 1\rangle$ corresponds to cyclic shift in $\mathbf{F}_p^n$. Theorem 4.9 and Theorem 4.10 follow from the fact that I is an ideal in $\mathbf{F}_p[x]/\langle x^n - 1\rangle$ if and only if I is principal and I is generated by an element of the form $g(x) + \langle x^n - 1\rangle$ where $g(x)$ divides $x^n - 1$.

Exercise: Consider the code over $\mathbf{F}_3$

$$\{(a, b, c, a, b, c) : a, b, c \in \mathbf{F}_3\}.$$

Thought of as a set of polynomials, this code becomes

$$C = \{a + bx + cx^2 + ax^3 + bx^4 + cx^5 : a, b, c \in \mathbf{F}_3\}.$$

Find a generator polynomial for C.

Theorem 4.9. *Let p be prime and let $C \subseteq \mathbf{F}_p[x]$ be a cyclic code of length n, represented by polynomials of degree $< n$. Then C has a generator polynomial.*

The next theorem tells us that we can construct a cyclic code over $\mathbf{F}_p$ of length n using any proper divisor of $x^n - 1$ in $\mathbf{F}_p[x]$ as a generator polynomial. Alternatively, if we know a generator polynomial then it gives us information about the code.

Theorem 4.10. *Let p be a prime, let $n \in \mathbf{N}$ and let $g(x) \in \mathbf{F}_p[x]$ be a divisor of $x^n - 1$. If $g(x)$ has degree $r < n$ then*

$$\{g(x), xg(x), \ldots, x^{n-r-1}g(x)\}.$$

is a basis for the cyclic code C with generator polynomial $g(x)$.

In particular, Theorem 4.10 implies that a cyclic code of length n with a generator polynomial of degree r over the finite field $\mathbf{F}_p$ has dimension $n - r$ and size p^{n-r}.

The following example shows how to construct all cyclic codes of a given length over the finite field $\mathbf{F}_p$, provided we have to hand the factorization of $x^n - 1$ in $\mathbf{F}_p$. (Finding these factorizations requires finite field theory beyond the scope of this course: they will be provided if required in an exam question.)

Example 4.11. In $\mathbf{F}_2[x]$ we have

$$x^6 - 1 = (1 + x)^2(1 + x + x^2)^2$$

where the factors $1 + x$ and $1 + x + x^2$ are irreducible, i.e. they cannot be written as products of polynomials of smaller degree.[5] The polynomial

[5] For $1 + x$ this is clear; if $1 + x + x^2$ factorized as the product of two polynomials of degree 1, it would have a root in $\mathbf{F}_2$. But $1 + 0 + 0^2 = 1 + 1 + 1^2 = 1$.

divisors of $x^6 - 1$ are therefore 1, $1 + x$, $1 + x + x^2$ and

$$\begin{aligned}(1+x)^2 &= 1 + x^2,\\ (1+x+x^2)^2 &= 1 + x^2 + x^4,\\ (1+x)(1+x+x^2) &= 1 + x^3,\\ (1+x)^2(1+x+x^2) &= 1 + x + x^3 + x^4,\\ (1+x)(1+x+x^2)^2 &= 1 + x + x^2 + x^3 + x^4 + x^5.\end{aligned}$$

For example, the code with generator polynomial $1 + x$ is the binary parity check code of length 6 (as defined in Example 2.9 of the main course). The code with generator polynomial $(1 + x)^2(1 + x + x^2)$ was seen in Example 4.3(3).

Theorem 4.12. *Let p be prime and let C be a cyclic code of length n over $\mathbf{F}_p$ with generator polynomial $g(x) \in \mathbf{F}_p[x]$ of degree r. If $g(x) = a_0 + a_1x + \cdots + a_rx^r$ then the $(n - r) \times n$ matrix*

$$G = \begin{pmatrix} a_0 & a_1 & a_2 & \dots & a_r & 0 & \dots & 0 \\ 0 & a_0 & a_1 & \dots & a_{r-1} & a_r & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & a_0 & \dots & \dots & a_{r-1} & a_r \end{pmatrix}$$

is a generator matrix for C.

Let $k = n - r$. Encoding using this generator matrix, we encode the binary word $(b_0, b_1, \dots, b_{k-1})$ of length $k - 1$ as

$$(b_0, b_1, \dots, b_{k-1})G$$

As a polynomial, this codeword is

$$\begin{aligned}&b_0g(x) + b_1xg(x) + \cdots + b_{k-1}x^{k-1}g(x)\\ &\qquad = (b_0 + b_1x + \cdots + b_{k-1}x^{k-1})g(x).\end{aligned}$$

So we can encode messages in a cyclic code by polynomial multiplication. This can be performed even more quickly than matrix multiplication.

We end this section with a small result showing that most cyclic codes are 1-error correcting.

Theorem 4.13. *Let C be a cyclic binary code of length n with generator polynomial $g(x) = a_0 + a_1x + \cdots + a_rx^r \in \mathbf{F}_2[x]$ of degree $r \geq 1$. Assume that $a_0 \neq 0$. If $x^s + 1$ is not divisible by $g(x)$ for any $i \in \{2, 3, \dots, n - 1\}$ then C has minimum distance at least 3.*

In fact it is unnecessary to assume that $a_0 \neq 0$. If $a_0 = 0$ then $g(x)$ is divisible by x. Since $g(x)$ divides $x^n - 1$, it follows that x divides $x^n - 1$, which is impossible.

Exercise: There is a refinement of the previous theorem: show that it suffices to check those $x^s + 1$ such that r divides s. [*Hint:* if $g(x)$ divides $x^s + 1$ then, since $g(x)$ divides $x^n + 1$, $g(x)$ divides the greatest common divisor of $x^n + 1$ and $x^s + 1$. Show that this greatest common divisor is $x^t + 1$ for some t dividing n.]

5. Reed–Solomon codes as cyclic codes

We now show that a special class of Reed–Solomon codes (as defined in Definition 2.1) are cyclic codes. For this we need one further general result. As usual, we state it for the finite fields $\mathbf{F}_p$, but it holds for a general finite field.

Lemma 5.1. *Let p be a prime. There exists an element $a \in \mathbf{F}_p$ such that the non-zero elements of $\mathbf{F}_p$ are exactly a^j for $0 \leq j \leq p - 2$.*

For example, in $\mathbf{F}_7$, one can take $a = 3$, since

$$3^0 = 1,\ 3^1 = 3,\ 3^2 = 2,\ 3^3 = 6,\ 3^4 = 4,\ 3^5 = 5.$$

Such an element a is said to be a *primitive root*.

Throughout this section, fix a prime p and let $k < p$. We suppose that $p - 1 - k$ is even and let $p - 1 - k = 2e$. (This agrees with notation in §3.) Let $a \in \mathbf{F}_p$ be a primitive root and let $RS_{p,p-1,k}$ be the Reed–Solomon code where polynomials are evaluated at $a_i = a^{i-1}$ for $i \in \{1, \ldots, p - 1\}$.

Lemma 5.2. *The code $RS_{p,p-1,k}$ is cyclic.*

It is now very natural to ask for a generator polynomial for $RS_{p,p-1,k}$. For this we shall first find a parity check matrix using the generator matrix found on Sheet 8, Question 4(a).

Theorem 5.3. *A parity check matrix for the cyclic Reed–Solomon code $RS_{p,p-1,k}$ is*

$$H = \begin{pmatrix} 1 & a & a^2 & \cdots & a^{p-2} \\ 1 & a^2 & a^4 & \cdots & a^{2(p-2)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & a^{2e} & a^{4e} & \cdots & a^{2e(p-2)} \end{pmatrix}.$$

Corollary 5.4. *The cyclic Reed–Solomon code $RS_{p,p-1,k}$ has generator polynomial*

$$g(x) = (x - a)(x - a^2) \dots (x - a^{2e}).$$

See Sheet 9 for the example $RS_{5,4,2}$ where polynomials are evaluated at $1, 2, 4, 3 \in \mathbf{F}_5$.

Remark 5.5. Theorem 5.3 can be used to give an alternative proof that the minimum distance of the cyclic Reed–Solomon code $RS_{p,p-1,k}$ is at least $p - k = 2e$. By Theorem 14.7 in the main notes, the minimum distance of $RS_{p,p-1,k}$ is equal to the minimum r such that there are r linearly dependent of the columns of the parity check matrix H. But by Theorem 1.8, any $2e$ columns of H are linearly independent.

An easier corollary of Theorem 5.3 is that the syndrome vH^{tr} of a received word $v \in \mathbf{F}_p^n$ corresponding to the polynomial $k(x)$ is equal to

$$\big(k(a), \dots, k(a^{2e})\big).$$

So one way to decode received words to codewords in $RS_{p,p-1,k}$ would be to compute syndromes using polynomial evaluation; then a table, as in page 52 of the main notes, could be used to perform syndrome decoding.

A much faster and more elegant decoding algorithm for cyclic Reed–Solomon codes was invented by Berlekamp and Welsh in 1971. As for the Key Equation decoder in §3, the algorithm implements nearest neighbour decoding, on the assumption that at most e errors have occurred.

Theorem 5.6. *Let $v \in \mathbf{F}_p^n$ be a received word with syndrome*

$$vH^{tr} = (S_1, \dots, S_{2e}).$$

Suppose that at most e errors occurred. Then given any $A(x), B(x) \in \mathbf{F}_p[x]$ with $\deg A(x) \leq e - 1$ and $\deg B(x) \leq e$ such that

$$B(x) = (S_1 + S_2 x + \dots + S_{2e} x^{2e-1}) A(x)$$

where we work modulo x^{2e}, so all powers x^{2e} and higher in this equation are regarded as 0, the sent codeword can be determined.

Provided at most e errors occurred, such polynomials $A(x)$, $B(x)$ always exist. They can be found by linear algebra, in a similar way to Lemma 3.4, or using the Berlekamp–Massey algorithm (this is its original application).

The final part of this section is based on the excellent account of the Berlekamp–Welsh decoder in Chapter 11 of [2]. Hill describes a decoder that can be used on a general (not necessarily cyclic) Reed–Solomon code; he also gives some refinements that further speed up decoding.

6. A BRIEF LOOK AT BCH CODES

This section may be considered non-examinable, and is included for interest only. Some knowledge of finite fields of prime power order is needed: let $\mathbf{F}_2^r$ denote the finite field of order r.

HAMMING CODES AS CYCLIC CODES. We start by giving a more algebraic way to construct the Hamming code of length 7. This construction generalizes to Hamming codes of any length. (See Exercise 7.20 in [4] in the recommended reading list.) Let $a \in \mathbf{F}_8$ be a primitive element, so $a^7 - 1$ and every non-zero element of $\mathbf{F}_8$ is equal to some power of a. Let

$$C = \{f(x) \in \mathbf{F}_2[x] : \deg f < 7, f(a) = 0\}.$$

Exercise: show from this definition that C is a cyclic code. (We have defined C as a set of polynomials, so the cyclic shift of $f(x) \in C$ should be defined to be the remainder when $xf(x)$ is divided by $x^7 - 1$.)

An equivalent definition of C is

$$C = \{(b_0, b_1, \ldots, b_6) \in \mathbf{F}_2^7 : b_0 + b_1 a + \cdots + b_6 a^6 = 0\}$$

Lemma 6.1. *The code C is equivalent to the Hamming $[7, 4, 3]$-code by a permutation of its positions.*

Proof. We may define the finite field $\mathbf{F}_8$ by $\mathbf{F}_8 = \mathbf{F}_2(a)$ where a is a root of the irreducible primitive polynomial

$$g(x) = x^3 + x + 1 \in \mathbf{F}_2[x].$$

If $f(x) \in \mathbf{F}_2[x]$ is a polynomial such that $f(a) = 0$ then, since $g(x)$ is the minimum polynomial of a, $g(x)$ divides $f(x)$. Hence

$$C = \{f(x) \in \mathbf{F}_2[x] : \deg f < 7, f(x) \text{ is divisible by } g(x)\}.$$

Thus C has generator polynomial $g(x)$. Hence, by Theorem 4.12, C has generator matrix

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

The required equivalence can be found by putting this generator matrix into standard form (G' below), and comparing with the generator matrix in Example 14.6 with columns permuted so that it is also in standard form (G'' below):

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

The matrices G' and G'' become the same, up to the order of the rows, if we swap columns 3 and 4 *and* columns 5 and 6 in G'. Hence C is equivalent to the Hamming code by a shuffle of positions.[6] □

BINARY BCH CODES. BCH codes (named after Bose and Ray-Chaudhuri) are a generalization of cyclic Hamming codes in which the polynomials corresponding to codewords are required to have roots at several different powers of the primitive element $a \in \mathbf{F}_2^r$.

BCH-codes are the most commonly used binary codes. For example, the BCH [15,5,7]-code is used to encode the format-string in the QR-codes in Example 1.11 of the main notes.

Definition 6.2. Let $t \in \mathbf{N}$ be given and let $2^r - 1 > 2t + 1$. Let $a \in \mathbf{F}_{2^r}$ be a primitive root. The *BCH-code* with *design distance* $2t + 1$ and length $n = 2^r - 1$ is the binary cyclic code C defined by

$$C = \{f(x) \in \mathbf{F}_2[x] : \deg f < n, f(a) = f(a^2) = \ldots = f(a^{2t+1}) = 0\}.$$

Equivalently, we may define

$$C = \{(u_0, u_1, \ldots, u_{n-1}) \in \mathbf{F}_2^n : (u_0, u_1, \ldots, u_{n-1})K^{tr} = 0\}$$

where

$$K = \begin{pmatrix} 1 & a & a^2 & \ldots & a^{n-1} \\ 1 & a^2 & a^4 & \ldots & a^{2(n-1)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & a^{2t} & a^{2(2t)} & \ldots & a^{2t(n-1)} \end{pmatrix}$$

Note that while K behaves like a parity matrix for C, and looks very like the matrix in Theorem 5.3, it is not a parity check matrix in the

[6] For the record, the permutation of columns needed to make C equal to the code in Example 14.6 is: $1 \mapsto 3$, $2 \mapsto 5$, $3 \mapsto 7$, $4 \mapsto 6$, $5 \mapsto 2$, $6 \mapsto 1$, $7 \mapsto 4$. For a better approach, which turns out to give a different equivalence, see Question 10 on Sheet 9. (The group of all permutations of the columns that leaves the Hamming $[7, 4, 3]$-code invariant is the simple group $\mathrm{PSL}(2, \mathbf{F}_7)$ of order 168.)

strict sense of Definition 14.1 in the main notes, because the entries of K are not in $\mathbf{F}_2$.

Example 6.3. Let r, $t \in \mathbf{N}$ and let C be the BCH-code of length $2^r - 1$ with design distance $2t + 1$, defined with respect to the primitive root $a \in \mathbf{F}_{2^r}$. Think of C as a set of polynomials of degree $< 2^r - 1$.

(1) If $r = 3$ and $t = 1$ then we may take a to be a root of $x^3 + x + 1$. Since a^2 is also a root of this polynomial, we have $f(a) = f(a^2) = 0$ if and only if $x^3 + x + 1$ divides $f(x)$. Hence C has generator polynomial $x^3 + x + 1$ and, by Lemma 6.1, C is equivalent to the Hamming $[7, 4, 3]$-code.

(2) If $r = 3$ and $t = 2$ then, taking a as in (1), we have

$$C = \{f(x) \in \mathbf{F}_2[x] : f(a) = f(a^3) = 0, \deg f \leq 6.\}$$

The minimum polynomials of a, a^2 and a^4 are $x^3 + x + 1$ and the minimum polynomial of a^3 is $x^3 + x^2 + 1$. Hence, C has generator polynomial

$$(x^3 + x + 1)(x^3 + x^2 + 1) = x^6 + x^5 + x^4 + x^3 + x^2 + x + 1.$$

In this case we merely succeed in giving a complicated construction of the repetition code. Note that the minimum distance of this code is 7, which is strictly more than the design distance 5.

(3) Let $r = 4$ and let $t = 2$. If a is a root of $x^4 + x + 1$; then a is a primitive root in $\mathbf{F}_2^4$. The minimum polynomial of a^3 is $x^4 + x^3 + x^2 + x + 1$ so the BCH code for these parameters is cyclic with generator polynomial $x^8 + x^7 + x^6 + x^4 + 1$. By Theorem 4.10 the code is

$$C = \{f(x)(x^8 + x^7 + x^6 + x^4 + 1) : f(x) \in \mathbf{F}_2[x], \deg f \leq 6\}$$

and so $\dim C = 15 - 8 = 7$. Since the generator polynomial has weight 5, the code contains words of weight 5. Hence C has minimum distance at most 5. It therefore follows from Theorem 6.4 below that C is a 2-error binary $[15, 7, 5]$-code.

We now give the general result on the size and minimum distance of binary BCH-codes.

Theorem 6.4. *Let C be the binary BCH code of length $2^r - 1$ and design distance $2t + 1$ defined with respect to the primitive root $a \in \mathbf{F}_{2^r}$. Then C is a cyclic code of dimension $\geq n - rt$ and minimum distance $\geq 2t + 1$.*

Outline proof. The product of the minimum polynomials of a^i for $1 \leq i \leq 2t+1$ is a generator polynomial for C. Each minimum polynomial has degree $\leq r$, since $a \in \mathbf{F}_2^r$. Moreover, if $f \in \mathbf{F}_2[x]$ then $f(a) = 0$ if and only if $f(a^2) = 0$. Hence the minimum polynomial of a^{2i} is the same as the minimum polynomial of a^i, and so we need only consider odd powers of a. It follows that g has degree $\leq rt$ and so, by Theorem 4.12, the dimension of C is at least $n - rt$.

To show that the minimum distance of C is at least $2t+1$ it suffices, by the same argument used in Remark 5.5, to show that no $2t$ columns of the matrix K following Definition 5.2 are linearly dependent. Again this follows from Theorem 1.8.

□

FINAL REMARKS ON BCH-CODES.

(1) More generally, BCH-codes can be defined over an arbitrary finite field $\mathbf{F}$ using a primitive element a in any field extension of $\mathbf{F}$. Over $\mathbf{F}_p$, taking a to be a primitive root in $\mathbf{F}_p$, the code with design distance $2t+1$ is the cyclic Reed–Solomon code $RS_{p,p-1,p-1-2t}$. (By Theorem 2.5 the minimum distance of this code is $2t+1$, so in these cases the design distance is exactly the minimum distance.)

(2) Different choice of the primitive root a will give different BCH-codes, but it turns out they have the same length and minimum distance: see Theorem 8.1.9 in [4] in the recommended reading list.

(3) BCH-codes can be decoded using the Berlekamp–Welsh decoder see in §5 for cyclic Reed–Solomon codes. See §8.2 of [4] for details.

We end with a final example that uses several different codes seen in this course.

Example 6.5. The Hamming code of length 15 is a $[15, 11, 3]$-code. Generalizing Example 14.6 in the main notes, it can be defined to be the binary code whose parity check matrix has all non-zero words of length 4 as its columns. An equivalent code is the binary BCH-code of length 15 and design distance 3.

The Hadamard codes constructed in Question 2 of Sheet 6 are linear: taking a linear Hadamard $(16, 32, 8)$ and puncturing it in its final position gives a $[15, 5, 7]$-code.

Working with linear codes of length 15 we have the following table of good codes.

1-error correcting (Hamming)	size $2^{11} = 2048$	[15,11,3]
2-error correcting (BCH)	size $2^7 = 128$	[15,7,5]
3-error corr. (punctured Hadamard)	size $2^5 = 32$	[15,5,7]

By the exercise below, the Hamming and punctured Hadamard codes are optimal. The BCH code has the largest possible size of a *linear* binary code of length 15 and minimum distance 5, but there is a larger non-linear code, of size 256. (See Theorem 7.4.5 in Van Lint, *Introduction to coding theory*, Springer 1982.)

Exercise: Show from the Hamming Packing Bound that $A_2(15, 3) = 2^{11}$. Use Theorem 11.6 and the Plotkin bound in Corollary 9.7 in the main notes to show that $A_2(15, 7) = 2^5$.