Chapter 8

Cyclic Codes

Among the first codes used practically were the cyclic codes which were generated using shift registers. It was quickly noticed by Prange that the class of cyclic codes has a rich algebraic structure, the first indication that algebra would be a valuable tool in code design.

The linear code C of length n is a *cyclic code* if it is invariant under a cyclic shift: cyclic code

$$\mathbf{c} = (c_0, c_1, c_2 \ldots, c_{n-2}, c_{n-1}) \in C$$

if and only if

$$\tilde{\mathbf{c}} = (c_{n-1}, c_0, c_1 \ldots, c_{n-3}, c_{n-2}) \in C\,.$$

As C is invariant under this single right cyclic shift, by iteration it is invariant under any number of right cyclic shifts. As a single left cyclic shift is the same as $n-1$ right cyclic shifts, C is also invariant under a single left cyclic shift and hence all left cyclic shifts. Therefore the linear code C is cyclic precisely when it is invariant under all cyclic shifts.

There are some obvious examples of cyclic codes. The $\mathbf{0}$-code is certainly cyclic as is F^n. Less trivially, repetition codes are cyclic. The binary parity check code is also cyclic, and this goes over to the sum-0 codes over any field.

Notice that this shift invariance criterion does not depend at all upon the code being linear. It is possible to define nonlinear cyclic codes, but that is rarely done. The history of cyclic codes as shift register codes and the mathematical structure theory of cyclic codes both suggest the study of cyclic invariance in the context of linear codes.

8.1 Basics

It is convenient to think of cyclic codes as consisting of polynomials as well as codewords. With every word

$$\mathbf{a} = (a_0, a_1, \ldots, a_i, \ldots, a_{n-2}, a_{n-1}) \in F^n$$

we associate the polynomial of degree less than n

$$a(x) = a_0 + a_1 x + \cdots + a_i x^i + \cdots + a_{n-1} x^{n-1} \in F[x]_n \,.$$

(We see here why in this chapter we index coordinates from 0 to $n-1$.) If $\mathbf{c}$ is a codeword of the code C, then we call $c(x)$ the associated *code polynomial*.

code polynomial

With this convention, the shifted codeword $\tilde{\mathbf{c}}$ has associated code polynomial

$$\tilde{c}(x) = c_{n-1} + c_0 x + c_1 x^2 + \cdots + c_i x^{i+1} + \cdots + c_{n-2} x^{n-1} \,.$$

Thus $\tilde{c}(x)$ is almost equal to the product polynomial $xc(x)$. More precisely,

$$\tilde{c}(x) = xc(x) - c_{n-1}(x^n - 1) \,.$$

Therefore $\tilde{c}(x)$ also has degree less than n and is equal to the remainder when $xc(x)$ is divided by $x^n - 1$. In particular

$$\tilde{c}(x) = xc(x) \pmod{x^n - 1} \,.$$

That is, $\tilde{c}(x)$ and $xc(x)$ are equal in the ring of polynomials $F[x] \pmod{x^n - 1}$, where arithmetic is done modulo the polynomial $x^n - 1$.

If $c(x)$ is the code polynomial associated with some codeword $\mathbf{c}$ of C, then we will allow ourselves to abuse notation by writing

$$c(x) \in C \,.$$

Indeed, if $f(x)$ is any polynomial of $F[x]$ whose remainder, upon division by $x^n - 1$, belongs to C then we may write

$$f(x) \in C \pmod{x^n - 1} \,.$$

With these notational conventions in mind, we see that our definition of the cyclic code C has the pleasing polynomial form

$$c(x) \in C \pmod{x^n - 1} \text{if and only if} xc(x) \in C \pmod{x^n - 1} \,.$$

Since additional shifts do not take us out of the cyclic code C, we have

$$x^i c(x) \in C \pmod{x^n - 1} \,,$$

for all i. By linearity, for any $a_i \in F$,

$$a_i x^i c(x) \in C \pmod{x^n - 1}$$

and indeed

$$\sum_{i=0}^{d} a_i x^i \, c(x) \in C \pmod{x^n - 1} \,,$$

That is, for every polynomial $a(x) = \sum_{i=0}^{d} a_i x^i \in F[x]$, the product $a(x)c(x)$ (or more properly $a(x)c(x) \pmod{x^n - 1}$) still belongs to C. This observation, due to Prange, opened the way for the application of algebra to cyclic codes.

(8.1.1) THEOREM. *Let $C \neq \{\mathbf{0}\}$ be a cyclic code of length n over F.*

(1) *Let $g(x)$ be a monic code polynomial of minimal degree in C. Then $g(x)$ is uniquely determined in C, and*

$$C = \{\, q(x)g(x) \,|\, q(x) \in F[x]_{n-r} \,\},$$

where $r = \deg(g(x))$. In particular, C has dimension $n - r$.

(2) *The polynomial $g(x)$ divides $x^n - 1$ in $F[x]$.*

PROOF. As $C \neq \{\mathbf{0}\}$, it contains nonzero code polynomials, each of which has a unique monic scalar multiple. Thus there is a monic polynomial $g(x)$ in C of minimal degree. Let this degree be r, unique even if $g(x)$ is not.

By the remarks preceding the theorem, the set of polynomials

$$C_0 = \{\, q(x)g(x) \,|\, q(x) \in F[x]_{n-r} \,\}$$

is certainly contained in C, since it is composed of those multiples of the code polynomial $g(x)$ with the additional property of having degree less than n. Under addition and scalar multiplication C_0 is an F-vector space of dimension $n - r$. The polynomial $g(x)$ is the unique monic polynomial of degree r in C_0.

To prove (1), we must show that every code polynomial $c(x)$ is an $F[x]$-multiple of $g(x)$ and so is in the set C_0. By the Division Algorithm A.2.5 we have

$$c(x) = q(x)g(x) + r(x)\,,$$

for some $q(x), r(x) \in F[x]$ with $\deg(r(x)) < r = \deg(g(x))$. Therefore

$$r(x) = c(x) - q(x)g(x)\,.$$

By definition $c(x) \in C$ and $q(x)g(x)$ is in C_0 (as $c(x)$ has degree less than n). Thus by linearity, the righthand side of this equation is in C, hence the remainder term $r(x)$ is in C. If $r(x)$ was nonzero, then it would have a monic scalar multiple belonging to C and of smaller degree than r. But this would contradict the original choice of $g(x)$. Therefore $r(x) = 0$ and $c(x) = q(x)g(x)$, as desired.

Next let

$$x^n - 1 = h(x)g(x) + s(x)\,,$$

for some $s(x)$ of degree less than $\deg(g(x))$. Then, as before,

$$s(x) = (-h(x))g(x) \pmod{x^n - 1}$$

belongs to C. Again, if $s(x)$ is not zero, then it has a monic scalar multiple belonging to C and of smaller degree than that of $g(x)$, a contradiction. Thus $s(x) = 0$ and $g(x)h(x) = x^n - 1$, as in (2). □

The polynomial $g(x)$ is called the *generator polynomial* for the code C. generator polynomial

The polynomial $h(x) \in F[x]$ determined by

$$g(x)h(x) = x^n - 1$$

check polynomial

is the *check polynomial* of C.

Under some circumstances it is convenient to consider $x^n - 1$ to be the generator polynomial of the cyclic code $\mathbf{0}$ of length n. Then by the theorem, there is a one-to-one correspondence between cyclic codes of length n and monic divisors of $x^n - 1$ in $F[x]$.

EXAMPLE. Consider length 7 binary cyclic codes. We have the factorization into irreducible polynomials

$$x^7 - 1 = (x-1)(x^3 + x + 1)(x^3 + x^2 + 1)\,.$$

Since we are looking at binary codes, all the minus signs can be replaced by plus signs:

$$x^7 + 1 = (x+1)(x^3 + x + 1)(x^3 + x^2 + 1)\,.$$

As there are 3 irreducible factors, there are $2^3 = 8$ cyclic codes (including $\mathbf{0}$ and $\mathbb{F}_2^7$). The 8 generator polynomials are:

(i) $1 = 1$
(ii) $x + 1 = x + 1$
(iii) $x^3 + x + 1 = x^3 + x + 1$
(iv) $x^3 + x^2 + 1 = x^3 + x^2 + 1$
(v) $(x+1)(x^3 + x + 1) = x^4 + x^3 + x^2 + 1$
(vi) $(x+1)(x^3 + x^2 + 1) = x^4 + x^2 + x + 1$
(vii) $(x^3 + x + 1)(x^3 + x^2 + 1) = x^6 + x^5 + x^4 + x^3 + x^2 + x + 1$
$(viii)$ $(x+1)(x^3 + x + 1)(x^3 + x^2 + 1) = x^7 + 1$

Here in (i) the polynomial 1 generates all $\mathbb{F}_2^7$. In (ii) we find the parity check code and in (vii) the repetition code. As mentioned before, in $(viii)$ we view the $\mathbf{0}$-code as being generated by $x^7 + 1$.

The polynomials of (iii) and (iv) have degree 3 and so generate $[7, 4]$ codes, which we shall later see are Hamming codes. The $[7, 3]$ codes of (v) and (vi) are the duals of the Hamming codes.

(8.1.2) PROBLEM. *How many cyclic codes of length* 8 *over* $\mathbb{F}_3$ *are there? Give a generator polynomial for each such code.*

(8.1.3) PROBLEM. *Prove that there is no cyclic code that is (equivalent to) an* $[8, 4]$ *extended binary Hamming code.*

(8.1.4) PROBLEM. *Let cyclic code* C *have generator polynomial* $g(x)$*. Prove that* C *is contained in the sum-*0 *code if and only if* $g(1) = 0$*.*

(8.1.5) PROBLEM. *Let* C *be a cyclic code. Let* C_- *be the code resulting from shortening* C *at a single position, and let* C^- *be the code resulting from puncturing* C *at a single position.*
(a) Give all C *for which* C_- *is cyclic.*
(b) Give all C *for which* C^- *is cyclic.*

The check polynomial earns its name by the following

(8.1.6) PROPOSITION. *If C is the cyclic code of length n with check polynomial $h(x)$, then*

$$C = \{\, c(x) \in F[x]_n \mid c(x)h(x) = 0 \pmod{x^n - 1} \,\}\,.$$

PROOF. The containment in one direction is easy. Indeed if $c(x) \in C$, then by Theorem 8.1.1 there is a $q(x)$ with $c(x) = q(x)g(x)$. But then

$$c(x)h(x) = q(x)g(x)h(x) = q(x)(x^n - 1) = 0 \pmod{x^n - 1}\,.$$

Now consider an arbitrary polynomial $c(x) \in F[x]_n$ with

$$c(x)h(x) = p(x)(x^n - 1), \text{say.}$$

Then

$$\begin{aligned} c(x)h(x) &= p(x)(x^n - 1) \\ &= p(x)g(x)h(x)\,, \end{aligned}$$

hence

$$(c(x) - p(x)g(x))h(x) = 0\,.$$

As $g(x)h(x) = x^n - 1$, we do not have $h(x) = 0$. Therefore

$$\begin{aligned} c(x) - p(x)g(x) &= 0 \\ \text{and } c(x) &= p(x)g(x)\,, \end{aligned}$$

as desired. □

If we are in possession of a generator polynomial $g(x) = \sum_{j=0}^{r} g_j x^j$ for the cyclic code C, then we can easily construct a generator matrix for C. Consider

$$G = \begin{bmatrix} g_0 & g_1 & \cdots & \cdots & \cdots & \cdots & g_{r-1} & g_r & 0 & 0 & \ldots & 0 \\ 0 & g_0 & g_1 & \cdots & \cdots & \cdots & \cdots & g_{r-1} & g_r & 0 & \ldots & 0 \\ \vdots & \vdots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \vdots \\ \vdots & \vdots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \vdots \\ 0 & 0 & \ldots & 0 & g_0 & g_1 & \cdots & \cdots & \cdots & \cdots & g_{r-1} & g_r \end{bmatrix}$$

The matrix G has n columns and $k = n - r$ rows; so the first row, row $\mathbf{g}_0$, finishes with a string of 0's of length $k - 1$. Each successive row is the cyclic shift of the previous row: $\mathbf{g}_i = \tilde{\mathbf{g}}_{i-1}$, for $i = 1, \ldots, k-1$. As $g(x)h(x) = x^n - 1$, we have

$$g_0 h_0 = g(0)h(0) = 0^n - 1 \neq 0\,.$$

In particular $g_0 \neq 0$ (and $h_0 \neq 0$). Therefore G is in echelon form (although likely not reduced). In particular the $k = \dim(C)$ rows of G are linearly independent. Clearly the rows of G belong to C, so G is indeed a generator matrix for C, sometimes called the *cyclic generator matrix* of C.

cyclic generator matrix

For instance, if C is a $[7,4]$ binary cyclic code with generator polynomial $1+x+x^3$, then the cyclic generator matrix is

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

cyclic encoding

Given the cyclic generator matrix G, *cyclic encoding* is the process of encoding the message k-tuple $\mathbf{m} = (m_0, \ldots, m_{k-1})$ into the codeword $\mathbf{c} = \mathbf{m}G$.

message polynomial

At the polynomial level, this corresponds to encoding the *message polynomial* $m(x) = \sum_{i=0}^{k-1} m_i x^i$ into the code polynomial $c(x) = m(x)g(x)$.

Since the cyclic generator G is in echelon form, the first k coordinate positions form an information set. Therefore cyclic C has a standard generator matrix, although the cyclic generator matrix is almost never standard (or even systematic).

(8.1.7) PROBLEM. (*a*) *Describe all situations in which the cyclic generator matrix for a cyclic code is the standard generator matrix.*

(*b*) *Describe all situations in which the cyclic generator matrix for a cyclic code is systematic.*

We next present for cyclic C a linear encoding method corresponding to the standard generator matrix. Namely

$$\mathbf{m} = (m_0, \ldots, m_{k-1}) \mapsto \mathbf{c} = (m_0, \ldots, m_{k-1}, -s_0, -s_1, \ldots, -s_{r-1}),$$

where $s(x) = \sum_{j=0}^{r-1} s_j x^j$ is the remainder upon dividing $x^r m(x)$ by $g(x)$. That is,

$$x^r m(x) = q(x)g(x) + s(x),$$

with $\deg(s(x)) < \deg(g(x)) = r$. To see that this is the correct standard encoding, first note that

$$x^r m(x) - s(x) = q(x)g(x) = b(x) \in C$$

with corresponding codeword

$$\mathbf{b} = (-s_0, -s_1, \ldots, -s_{r-1}, m_0, \ldots, m_{k-1}).$$

As this is a codeword of cyclic C, every cyclic shift of it is also a codeword. In particular the $\mathbf{c}$ given above is found after k right shifts. Thus $\mathbf{c}$ is a codeword of C. Since C is systematic on the first k positions, this codeword is the only one with $\mathbf{m}$ on those positions and so is the result of standard encoding. To construct the standard generator matrix itself, we encode the k different k-tuple messages $(0, 0, \ldots, 0, 1, 0, \ldots, 0)$ of weight 1 corresponding to message polynomials x^i, for $0 \le i \le k-1$. These are the rows of the standard generator matrix.

When we try this for the $[7,4]$ binary cyclic code with generator x^3+x+1 (so $r = 7-4 = 3$), we find, for instance,

$$x^3 x^2 = (x^2+1)(x^3+x+1) + (x^2+x+1)$$

so that the third row of the standard generator matrix, corresponding to message polynomial x^2, is

$$(m_0, m_1, m_2, m_3, -s_0, -s_1, -s_2) = (0, 0, 1, 0, 1, 1, 1)\,.$$

Proceeding in this way, we find that the standard generator matrix is

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

By Problem 4.1.9, C is a Hamming code (although this can also be checked easily by hand).

This process of *systematic encoding* for cyclic codes is important in practice, since a machine can be transmitting the information symbols from $\mathbf{m}$ during the time it is calculating the check symbols s_j. systematic encoding

(8.1.8) PROBLEM. (*a*) *Find the cyclic and standard generator matrices for the* $[7, 4]$ *binary cyclic code* D *with generator polynomial* $x^3 + x^2 + 1$.

(*b*) *Find the cyclic and standard generator matrices for the* $[15, 11]$ *binary cyclic code* E *with generator polynomial* $x^4 + x + 1$.

(*c*) *Prove that* D *and* E *are Hamming codes.*

A code equivalent to a cyclic code need not be cyclic itself. For instance, there are 30 distinct binary $[7, 4]$ Hamming codes; but, as we saw in the example above, only two of them are cyclic.

One permutation does take cyclic codes to cyclic codes. The *reverse code* $C^{[-1]}$ of a cyclic code C, gotten by reversing each codeword, is still cyclic. We have reverse code

$$(c_0, c_1, \ldots, c_i, \ldots, , c_{n-1}) \in C \quad \Longleftrightarrow \quad (c_{n-1}, \ldots, c_{n-1-i}, \ldots, c_1, c_0) \in C^{[-1]}\,.$$

In polynomial notation, this becomes

$$c(x) \in C \quad \Longleftrightarrow \quad x^{n-1} c(x^{-1}) \in C^{[-1]}\,.$$

For the polynomial $p(x)$ of degree d, we let its *reciprocal polynomial* be given by reciprocal polynomial

$$p^{[-1]}(x) = \sum_{i=0}^{d} p_{d-i} x^i = x^d p(x^{-1})\,.$$

The roots of the reciprocal polynomial are the reciprocals of the nonzero roots of the original polynomial.

(8.1.9) LEMMA. *If* $g(x)$ *generates cyclic* C*, then* $g_0^{-1} g^{[-1]}(x)$ *generates* $C^{[-1]}$*, the reverse code of* C.

PROOF. Starting from the cyclic generator matrix for C, we reverse all the rows and then write them from bottom to top. The result is

$$\begin{bmatrix} g_r & g_{r-1} & \cdots & \cdots & \cdots & \cdots & g_1 & g_0 & 0 & 0 & \ldots & 0 \\ 0 & g_r & g_{r-1} & \cdots & \cdots & \cdots & \cdots & g_1 & g_0 & 0 & \ldots & 0 \\ \vdots & \vdots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \vdots \\ \vdots & \vdots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \vdots \\ 0 & 0 & \ldots & 0 & g_r & g_{r-1} & \cdots & \cdots & \cdots & \cdots & g_1 & g_0 \end{bmatrix}.$$

The rows of this matrix certainly belong to $C^{[-1]}$. As before, they are linearly independent since $g_0 \neq 0$. Therefore we have a generator matrix for $C^{[-1]}$. Its first row visibly corresponds to a nonzero code polynomial of degree less than r, which is seen to be $g^{[-1]}(x)$. By Theorem 8.1.1 the monic scalar multiple $g_0^{-1}g^{[-1]}(x)$ is the generator polynomial. (In fact, we have a scalar multiple of the cyclic generator matrix for $C^{[-1]}$.) □

It is easy to see that the dual of a cyclic code C is again a cyclic code. Proposition 8.1.6 suggests that the dual is associated with the check polynomial of C.

Let the cyclic code C of length n have generator polynomial $g(x)$ of degree r and check polynomial $h(x)$ of degree $k = n - r = \dim C$. As $h(x)$ is a divisor of $x^n - 1$, it is the generator polynomial for a cyclic code D of length n and dimension $n - k = n - (n - r) = r$. We have

$$C = \{\, q(x)g(x) \,|\, q(x) \in F[x]_k \,\}$$

and

$$D = \{\, p(x)h(x) \,|\, p(x) \in F[x]_r \,\}.$$

Let $c(x) = q(x)g(x) \in C$, so that $\deg(q(x)) \leq k - 1$; and let $d(x) = p(x)h(x) \in D$, so that $\deg(p(x)) \leq r - 1$. Consider

$$\begin{aligned} c(x)d(x) &= q(x)g(x)p(x)h(x) \\ &= q(x)p(x)(x^n - 1) \\ &= s(x)(x^n - 1) \\ &= s(x)x^n - s(x), \end{aligned}$$

where $s(x) = q(x)p(x)$ with

$$\deg(s(x)) \leq (k-1) + (r-1) = r + k - 2 = n - 2 < n - 1.$$

Therefore the coefficient of x^{n-1} in $c(x)d(x)$ is 0. If $c(x) = \sum_{i=0}^{n-1} c_i x^i$ and $d(x) = \sum_{j=0}^{n-1} d_j x^j$, then in general the coefficient of x^m in $c(x)d(x)$ is $\sum_{i+j=m} c_i d_j$. In

particular, the two determinations of the coefficient of x^{n-1} in $c(x)d(x)$ give

$$\begin{aligned} 0 &= \sum_{i+j=n-1} c_i d_j \\ &= \sum_{i=0}^{n-1} c_i d_{n-1-i} \\ &= c_0 d_{n-1} + c_1 d_{n-2} + \cdots + c_i d_{n-i} + \cdots + c_{n-1} d_0 \\ &= \mathbf{c} \cdot \mathbf{d}^* \,. \end{aligned}$$

where

$$\mathbf{c} = (c_0, c_1, \ldots, c_i, \ldots, c_{n-1}) \text{and} \mathbf{d}^* = (d_{n-1}, d_{n-2}, \ldots, d_{n-i}, \ldots, d_0) \,.$$

That is, each codeword $\mathbf{c}$ of C has dot product 0 with the reverse of each codeword $\mathbf{d}$ of D.

Therefore $C^\perp$ contains $D^{[-1]}$. Also

$$\dim(C^\perp) = n - \dim(C) = n - k = r = n - \deg(h^{[-1]}(x)) = \dim(D^{[-1]}) \,,$$

so from Lemma 8.1.9 we conclude

(8.1.10) THEOREM. *If C is the cyclic code of length n with check polynomial $h(x)$, then $C^\perp$ is cyclic with generator polynomial $h_0^{-1} h^{[-1]}(x)$.* □

8.2 Cyclic GRS codes and Reed-Solomon codes

For α a primitive n^{th} root of unity in the field F, set

$$\begin{aligned} \boldsymbol{\alpha}^{(a)} &= ((\alpha^0)^a, \ldots, (\alpha^j)^a, \ldots, (\alpha^{n-1})^a) \\ &= ((\alpha^a)^0, \ldots, (\alpha^a)^j, \ldots, (\alpha^a)^{n-1}) \,. \end{aligned}$$

In particular, $\boldsymbol{\alpha} = \boldsymbol{\alpha}^{(1)}$ and $\boldsymbol{\alpha}^{(0)} = \mathbf{1}$, the all 1-vector.

The basic observation is that

$$\begin{aligned} \tilde{\boldsymbol{\alpha}}^{(a)} &= ((\alpha^{n-1})^a, (\alpha^0)^a, \ldots, (\alpha^j)^a, \ldots, (\alpha^{n-2})^a) \\ &= \alpha^{-a}((\alpha^0)^a, (\alpha^1)^a, \ldots, (\alpha^j)^a, \ldots, (\alpha^{n-1})^a) \\ &= \alpha^{-a} \boldsymbol{\alpha}^{(a)} \,. \end{aligned}$$

Thus a cyclic shift of $\boldsymbol{\alpha}^{(a)}$ is always a scalar multiple of $\boldsymbol{\alpha}^{(a)}$.

(8.2.1) PROPOSITION. *$\mathrm{GRS}_{n,k}(\boldsymbol{\alpha}, \boldsymbol{\alpha}^{(a)})$ is cyclic.*

PROOF. For $0 \le i \le k-1$ and $0 \le j \le n-1$, the (i,j)-entry of the canonical generator matix is

$$\begin{aligned} v_j \alpha_j^i &= (\alpha^j)^a (\alpha^j)^i \\ &= \alpha^{ja} \alpha^{ji} = (\alpha^j)^{a+i} \,. \end{aligned}$$

Therefore the canonical generator matrix has as rows the k codewords $\boldsymbol{\alpha}^{(a+i)}$, for $i = 0, \ldots, k-1$. We have seen above that shifting any of these only gives scalar multiples, so the code itself is invariant under shifting. □

A cyclic code $\mathrm{GRS}_{n,k}(\boldsymbol{\alpha}, \boldsymbol{\alpha}^{(a)})$ as in Proposition 8.2.1 is a *Reed-Solomon code*. It is said to be *primitive* if $n = |F| - 1$ and of *narrow-sense* if $a = 0$ (so that $\mathbf{v} = \boldsymbol{\alpha}^{(a)} = \mathbf{1}$).

Reed-Solomon code
primitive
narrow-sense

(8.2.2) Lemma. *If $\alpha^n = 1$ and $\boldsymbol{\alpha} = (\alpha^0, \ldots, \alpha^{n-1})$, then*

$$\mathrm{GRS}_{n,k}(\boldsymbol{\alpha}, \boldsymbol{\alpha}^{(a)})^{\perp} = \mathrm{GRS}_{n,n-k}(\boldsymbol{\alpha}, \boldsymbol{\alpha}^{(1-a)})\,.$$

Proof. By Theorem 5.1.6

$$\mathrm{GRS}_{n,k}(\boldsymbol{\alpha}, \boldsymbol{\alpha}^{(a)})^{\perp} = \mathrm{GRS}_{n,n-k}(\boldsymbol{\alpha}, \mathbf{u})\,,$$

where, for $0 \le j \le n-1$ and $\mathbf{v} = \boldsymbol{\alpha}^{(a)}$, we have $u_j = v_j^{-1} L_j(\alpha^j)^{-1}$. By Problem 5.1.5(c), $L_j(\alpha^j) = n(\alpha^j)^{-1}$ $(\neq 0)$. Thus

$$\begin{aligned} u_j &= ((\alpha^j)^a)^{-1}(n(\alpha^j)^{-1})^{-1} \\ &= n^{-1}\alpha^{-ja}\alpha^j \\ &= n^{-1}(\alpha^j)^{1-a} \end{aligned}$$

Therefore $\mathbf{u} = n^{-1}\boldsymbol{\alpha}^{(1-a)}$, so by Problem 5.1.3(a)

$$\begin{aligned} \mathrm{GRS}_{n,k}(\boldsymbol{\alpha}, \boldsymbol{\alpha}^{(a)})^{\perp} &= \mathrm{GRS}_{n,n-k}(\boldsymbol{\alpha}, n^{-1}\boldsymbol{\alpha}^{(1-a)}) \\ &= \mathrm{GRS}_{n,n-k}(\boldsymbol{\alpha}, \boldsymbol{\alpha}^{(1-a)}) \end{aligned}$$

as desired. □

(8.2.3) Theorem. *An $[n, k]$ Reed-Solomon code over F is a cyclic code with generator polynomial*

$$\prod_{j=1}^{t}(x - \alpha^{j+b})$$

where $t = n - k$, the integer b is a fixed constant, and α is a primitive n^{th} root of unity in F. This Reed-Solomon code is primitive if $n = |F| - 1$ and narrow-sense if $b = 0$.

Proof. Let $C = \mathrm{GRS}_{n,k}(\boldsymbol{\alpha}, \boldsymbol{\alpha}^{(a)})$. The rows of the canonical generator matrix of the dual code $C^{\perp}$ are, by Lemma 8.2.2 and a previous calculation, the vectors $\boldsymbol{\alpha}^{(j-a)}$, for $1 \le j \le t$. Therefore, for $\mathbf{c} = (c_0, \ldots, c_i, \ldots, c_{n-1})$ and

$c(x) = \sum_{i=0}^{n-1} c_i x^i$,

$$\begin{aligned}
\mathbf{c} \in C \quad &\iff \quad \mathbf{c} \cdot \boldsymbol{\alpha}^{(j-a)} = 0, \quad 1 \le j \le t \\
&\iff \quad \sum_{i=0}^{n-1} c_i (\alpha^i)^{j-a} = 0, \quad 1 \le j \le t \\
&\iff \quad \sum_{i=0}^{n-1} c_i (\alpha^{j-a})^i = 0, \quad 1 \le j \le t \\
&\iff \quad c(\alpha^{j-a}) = 0, \quad 1 \le j \le t\,.
\end{aligned}$$

Thus, writing cyclic C in terms of polynomials, we have by Lemma A.2.8

$$\begin{aligned}
c(x) \in C \quad &\iff \quad c(\alpha^{j-a}) = 0, \quad 1 \le j \le t \\
&\iff \quad \prod_{j=1}^{t} (x - \alpha^{j+b}) \text{divides} c(x)\,,
\end{aligned}$$

for $b = -a$. As $\prod_{j=1}^{t}(x - \alpha^{j+b})$ is monic and has degree $t = n - k$, it is the generator polynomial of C by Theorem 8.1.1.

Also α is a primitive element of F when $n = |F| - 1$; and C is narrow-sense when $a = 0$, that is, when $b = -a = 0$. □

In most places, the statement of Theorem 8.2.3 is taken as the definition of a Reed-Solomon code. It is then proven that such a code is MDS with $\mathrm{d}_{\min} = t + 1 = n - k + 1$. Our development is somewhat closer to the original presentation of Reed and Solomon from 1960.

(8.2.4) PROBLEM. *Prove that* $\mathrm{EGRS}_{q+1,k}(\boldsymbol{\beta}, \boldsymbol{\gamma}; \mathbf{w})$*, where* $|F| = q$*, is monomially equivalent to a cyclic code when* q *is even and to a negacyclic code when* q *is odd. Here a code* C *is* negacyclic *provided*

negacyclic

$$(c_0, c_1, c_2 \ldots, c_{n-2}, c_{n-1}) \in C$$

if and only if

$$(-c_{n-1}, c_0, c_1 \ldots, c_{n-3}, c_{n-2}) \in C\,.$$

*(*HINT: *See Theorem 6.3.4.)*

8.3 Cylic alternant codes and BCH codes

Let $K \le F$ be fields. Starting with the Reed-Solomon code $\mathrm{GRS}_{n,k}(\boldsymbol{\alpha}, \boldsymbol{\alpha}^{(a)})$ over F, the cyclic, alternant code $C = K^n \cap \mathrm{GRS}_{n,k}(\boldsymbol{\alpha}, \boldsymbol{\alpha}^{(a)})$ is called a *BCH code of designed distance* $t + 1$, where $t = n - k$. C is *primitive* if $n = |F| - 1$ and *narrow-sense* if $a = 0$ (that is to say, $\mathbf{v} = \mathbf{1}$).

(8.3.1) THEOREM. *A BCH code C of length n and designed distance $t+1$ over K is a cyclic code composed of all those code polynomials $c(x) \in K[x]$ of degree less than n satisfying*

$$c(\alpha^{b+1}) = c(\alpha^{b+2}) = c(\alpha^{b+3}) = \cdots = c(\alpha^{b+t}) = 0\,,$$

where b is a fixed integer and α is a primitive n^{th} root of unity in the field $F \geq K$. The code is primitive if $n = |F| - 1$ and is narrow-sense if $b = 0$.

The code C is linear and cyclic with generator polynomial

$$\operatorname{lcm}_{1 \leq j \leq t} \{m_{\alpha^{j+b},K}(x)\}\,.$$

It has minimum distance at least $t+1$ and dimension at least $n - mt$, where $m = \dim_K F$.

PROOF. The first paragraph is an immediate consequence of Theorem 8.2.3 and the definitions. As C is the alternant code $K^n \cap \mathrm{GRS}_{n,k}(\boldsymbol{\alpha}, \boldsymbol{\alpha}^{(a)})$, it is by Theorem 7.5.1 linear of minimum distance at least $n - k + 1 = t + 1$ and dimension at least $n - m(n-k) = n - mt$. The form taken by the generator polynomial follows from the first paragraph and Lemma A.3.19 of the Appendix. □

As with Reed-Solomon codes, the first paragraph of this theorem consists of the usual definition of a BCH code. Indeed, that is essentially the original definition as given by Bose and Ray-Chaudhuri (1960) and Hocquenghem (1959). (The codes were then given the somewhat inaccurate acronym as name.) It then must be proven that the designed distance of a BCH code gives a lower bound for the actual minimum distance. In many places Reed-Solomon codes are defined as those BCH codes in which the fields F and K are the same. Historically, the two classes of codes were discovered independently and the connections only noticed later.

Sometimes one takes a different view of Theorem 8.3.1, viewing it instead as a general bound on cyclic codes in terms of root patterns for the generator polynomial.

(8.3.2) COROLLARY. (BCH BOUND.) *Let C be a cyclic code of length n over K with generator polynomial $g(x)$. Suppose that $g(\alpha^{j+b}) = 0$, for some fixed b and $1 \leq j \leq t$, where α is a primitive n^{th} root of unity in the field $F \geq K$. Then $\mathrm{d}_{\min}(C) \geq t + 1$.*

PROOF. In this case, C is a subcode of a BCH code with designed distance $t+1$. □

This corollary admits many generalizations, the general form of which states that a certain pattern of roots for the generator polynomial of a cyclic code implies a lower bound for the minimum distance.

(8.3.3) PROBLEM. *Assume that the cyclic code C has generator polynomial $g(x)$ with $g(1) \neq 0$. Prove that $(x-1)g(x)$ is the generator polynomial of the sum-0 subcode of C (those codewords of C whose coordinate entries sum to 0).*

The last sentence in the theorem gives us two lower bounds for BCH codes, one for the minimum distance (the BCH bound) and one for the dimension. As we prefer large distance and dimension, we would hope to find situations in which one or both of these bounds are not met exactly. For any cyclic code, the generator polynomial has degree equal to the redundancy of the code. In Theorem 8.3.1 that degree/redundancy is bounded above by mt. This bound will be met exactly if and only if each of the minimal polynomials $m_{\alpha^{j+b},K}(x)$ has the maximum possible degree m and, additionally, all of these polynomials, for $1 \leq j \leq t$, are distinct. This sounds an unlikely event but can, in fact, happen. Conversely we often can make our choices so as to guarantee that the degree of the generator is dramatically less than this maximum. We shall see below that the two bounds of the theorem are independent and can be either met or beaten, depending upon the specific circumstances. (Both bounds are tight for Reed-Solomon codes, but there are other cases as well where this happens.)

(8.3.4) COROLLARY. (1) *A binary, narrow-sense, primitive BCH code of designed distance 2 is a cyclic Hamming code.*

(2) *A binary, narrow-sense, primitive BCH code of designed distance 3 is a cyclic Hamming code.*

PROOF. Let $n = 2^m - 1$ and $K = \mathbb{F}_2 \leq \mathbb{F}_{2^m} = F$. Let α be a primitive element in $\mathbb{F}_{2^m}$ (so it has order n). Then the associated designed distance 2 code C_2 has generator polynomial

$$m(x) = m_\alpha(x) = m_{\alpha,\mathbb{F}_2}(x)$$

of degree m, the minimal polynomial of α over the prime subfield $\mathbb{F}_2$. The corresponding designed distance 3 code C_3 has generator polynomial

$$\operatorname{lcm}\{m_\alpha(x), m_{\alpha^2}(x)\}\,.$$

From Theorem A.3.20 we learn that $m_{\alpha^2}(x) = m_\alpha(x)$. Therefore this lcm is again equal to $m(x)$, and C_2 and C_3 both have generator polynomial $m(x)$ of degree m. Thus $C_2 = C_3$ has dimension $n - m = 2^m - 1 - m$ and minimum distance at least 3. It is therefore a Hamming code by Problem 4.1.3 or Problem 4.1.9. (Alternatively C_2 is, by Lemma 8.2.2, equal to the alternant code $\mathbb{F}_2^n \cap \mathrm{GRS}_{n,1}(\boldsymbol{\alpha}, \boldsymbol{\alpha})^\perp$, which we have already identified as a Hamming code in Section 7.5.) □

From this corollary we learn that it is possible to find BCH codes with inequality in the distance bound (BCH bound) and equality in the dimension bound of Theorem 8.3.1 ($\mathrm{d}_{\min}(C_2) = 3 > 1 + 1$ and $\dim(C_2) = n - m \cdot 1$) and also BCH codes with equality in the distance bound and inequality in the dimension bound ($\mathrm{d}_{\min}(C_3) = 3 = 2 + 1$ and $\dim(C_3) = n - m > n - m \cdot 2$).

As in the corollary, narrow-sense codes frequently have better parameters than those that are not. For instance, in the situation of the corollary, the designed distance 2 code with $b = -1$ has generator polynomial $m_{\alpha^{1-1},\mathbb{F}_2}(x) = x - 1$. This code is the parity check code with $\mathrm{d}_{\min}$ indeed equal to 2 and dimension $n - 1 (> n - m)$. When $n = 15$ (so that $m = 4$), the designed distance 2 code with $b = 2$ has generator polynomial

$$m_{\alpha^{1+2},\mathbb{F}_2}(x) = x^4 + x^3 + x^2 + x + 1 = (x^5 - 1)/(x - 1)\,,$$

since $(\alpha^3)^5 = \alpha^{15} = 1$. Therefore this code meets both bounds exactly, having dimension $11 = 15 - 4$ and minimum distance 2, as it contains the code polynomial $x^5 - 1$.

Consider next the binary, narrow-sense, primitive BCH code with length 15 and designed distance 5, defined using as primitive element α a root of the primitive polynomial x^4+x+1. The generator polynomial is, by Theorem 8.3.1,

$$g(x) = \mathrm{lcm}_{1\le j\le 4}\left\{m_{\alpha^j}(x)\right\}.$$

By definition $m_\alpha(x) = x^4 + x + 1$, and we found $m_{\alpha^3}(x) = x^4 + x^3 + x^2 + x + 1$ above. By Theorem A.3.20 of the Appendix,

$$m_\alpha(x) = m_{\alpha^2}(x) = m_{\alpha^4}(x)\,,$$

therefore

$$\begin{aligned} g(x) &= m_\alpha(x)m_{\alpha^3}(x) = (x^4 + x + 1)(x^4 + x^3 + x^2 + x + 1) \\ &= x^8 + x^7 + x^6 + x^4 + 1\,. \end{aligned}$$

In particular, the code has dimension $15-8 = 7$, whereas the bound of Theorem 8.3.1 is useless, claiming only that the dimension is at least $15 - 4 \cdot 4 = -1$. Furthermore $g(x)$ itself has weight 5, so in this case the designed distance 5 code has minimum distance exactly 5. (Although the generator polynomial always has relatively low weight, in general it will not have the minimum weight. Still it is often worth checking.) We see again here the advantage of looking at narrow-sense codes. By Theorem A.3.20, whenever α^i is a root of $m(x)$, then α^{2i} is as well (in the binary case). In particular, the binary, narrow-sense, designed distance $2d$ code, given by roots α^j, for $1 \le j \le 2d - 1$, is also equal to the designed distance $2d + 1$ code, given by roots α^j, for $1 \le j \le 2d$, since α^d a root implies α^{2d} is as well. (We saw a particular case of this in Corollary 8.3.4.) Similar but weaker statements can be made for nonbinary BCH codes by appealing to Theorem A.3.20 or the more general Problem A.3.21.

We also see that Theorem A.3.20 and Problem A.3.21 can be used effectively to calculate the parameters and generator polynomials of BCH codes. Consider next a binary, narrow-sense, primitive BCH code C of length 31 with designed distance 8. The previous paragraph already tells us that C is also the corresponding designed distance 9 code, but more is true. We have generator polynomial

$$\begin{aligned} g(x) &= \mathrm{lcm}_{1\le j\le 8}\left\{m_{\alpha^j}(x)\right\} \\ &= m_\alpha(x)m_{\alpha^3}(x)m_{\alpha^5}(x)m_{\alpha^7}(x)\,, \end{aligned}$$

where α is an arbitrary but fixed primitive 31^{st} root of unity in $\mathbb{F}_{32}$. By Theorem A.3.20

$$\begin{aligned} m_{\alpha}(x) &= (x-\alpha)(x-\alpha^2)(x-\alpha^4)(x-\alpha^8)(x-\alpha^{15})\,; \\ m_{\alpha^3}(x) &= (x-\alpha^3)(x-\alpha^6)(x-\alpha^{12})(x-\alpha^{24})(x-\alpha^{17})\,; \\ m_{\alpha^5}(x) &= (x-\alpha^5)(x-\alpha^{10})(x-\alpha^{20})(x-\alpha^{9})(x-\alpha^{18})\,; \\ m_{\alpha^7}(x) &= (x-\alpha^7)(x-\alpha^{14})(x-\alpha^{28})(x-\alpha^{25})(x-\alpha^{19})\,. \end{aligned}$$

Therefore C has dimension $31-4\cdot 5=11$. We also discover that we have gotten the roots α^9 and α^{10} 'for free', so that the designed distance 8(9) BCH code is actually equal to the designed distance 11 code (so in this case, neither of the bounds of Theorem 8.3.1 hold with equality). It is worth noting that we can calculate this dimension and improved BCH bound without explicitly finding the generator polynomial. The calculations are valid no matter which primitive element α we choose. Examples below find explicit generator polynomials, using similar calculations based upon Theorem A.3.20.

The good fortune seen in the previous paragraph can often be dramatic. Berlekamp has noted that the binary, narrow-sense, primitive BCH code of length $2^{12}-1$ and designed distance 768 is equal to the corresponding designed distance 819 code. On the other hand, there are many situations where the BCH bound still does not give the true minimum distance. Roos, Van Lint, and Wilson have noted that the binary length 21 code with generator polynomial

$$m_{\beta}(x)m_{\beta^3}(x)m_{\beta^7}(x)m_{\beta^9}(x)\,,$$

which has as roots β^j (β a 21^{st} root of unity) for

$$j\in\{1,2,3,4,6,7,8,9,11,12,14,15,16,18\}\,,$$

has true minimum distance 8, whereas the BCH bound only guarantees distance at least 5.

EXAMPLES. (*i*) Let α be a root of the primitive polynomial $x^4+x^3+1\in\mathbb{F}_2[x]$. What is the generator of the binary, narrow-sense, primitive BCH code C_1 of length 15 and designed distance 5?

The code is composed of all polynomials $c(x)\in\mathbb{F}_2[x]$ that have each of $\alpha^1,\alpha^2,\alpha^3,\alpha^4$ as a root. Therefore $c(x)$ is divisible by

$$m_{\alpha}(x)=(x-\alpha^1)(x-\alpha^2)(x-\alpha^4)(x-\alpha^8)=x^4+x^3+1$$

and also by

$$m_{\alpha^3}(x)=(x-\alpha^3)(x-\alpha^6)(x-\alpha^{12})(x-\alpha^9)=x^4+x^3+x^2+x+1.$$

So C_1 has generator

$$g_1(x)=(x^4+x^3+1)(x^4+x^3+x^2+x+1)=x^8+x^4+x^2+x+1.$$

As $g_1(x)$ has degree 8, the code has dimension $15 - 8 = 7$. (Here again the generator has weight 5, so the minimal distance of this code equals 5.)

(*ii*) Let α be a root of the primitive polynomial $x^5 + x^2 + 1 \in \mathbb{F}_2[x]$. What is the generator of the binary, narrow-sense, primitive BCH code C_2 of length 31 and designed distance 5?

Again the code is composed of all polynomials $c(x) \in \mathbb{F}_2[x]$ that have each of $\alpha^1, \alpha^2, \alpha^3, \alpha^4$ as a root. Therefore $c(x)$ is divisible by

$$m_\alpha(x) = (x - \alpha^1)(x - \alpha^2)(x - \alpha^4)(x - \alpha^8)(x - \alpha^{16}) = x^5 + x^2 + 1$$

and also

$$m_{\alpha^3}(x) = (x-\alpha^3)(x-\alpha^6)(x-\alpha^{12})(x-\alpha^{24})(x-\alpha^{17}) = x^5+x^4+x^3+x^2+1.$$

So C_2 has generator

$$g_2(x) = (x^5+x^2+1)(x^5+x^4+x^3+x^2+1) = x^{10}+x^9+x^8+x^6+x^5+x^3+1.$$

As $g_2(x)$ has degree 10, the code has dimension $31 - 10 = 21$. (But notice that here the weight of the generator is larger than 5.)

(*iii*) Maintaining the notation of Example (*ii*), find the generator of the BCH code C_3 of length 31 with designed distance 7.

The code polynomials $c(x)$ must satisfy

$$c(\alpha^1) = c(\alpha^2) = c(\alpha^3) = c(\alpha^4) = c(\alpha^5) = c(\alpha^6) = 0.$$

In particular $c(x)$ must be a multiple of $g_2(x)$, calculated in the previous example. But $c(x)$ must also be a multiple of

$$m_{\alpha^5}(x) = (x-\alpha^5)(x-\alpha^{10})(x-\alpha^{20})(x-\alpha^9)(x-\alpha^{18}) = x^5+x^4+x^2+x+1.$$

(This calculation is done in detail in section A.3.3 of the Appendix on algebra.) Thus the generator $g_3(x)$ for C_3 is

$$g_2(x)(x^5+x^4+x^2+x+1) = x^{15}+x^{11}+x^{10}+x^9+x^8+x^7+x^5+x^3+x^2+x+1.$$

This code has dimension $31 - 15 = 16$.

(*iv*) Let β be a root of the irreducible but imprimitive polynomial $x^3 + 2x + 2 \in \mathbb{F}_3[x]$ so that β is a 13^{th} root of unity. We can, using β, find the generator polynomial of the ternary, narrow-sense BCH code D_1 of length 13 with designed distance 4.

The code polynomials must have as roots β, β^2, and β^3. Thus they must be multiples of

$$m_\beta(x) = m_{\beta^3}(x) = (x - \beta)(x - \beta^3)(x - \beta^9) = x^3 + 2x + 2$$

and of

$$m_{\beta^2}(x) = (x - \beta^2)(x - \beta^6)(x - \beta^5) = x^3 + x^2 + x + 2.$$

Therefore D_1 has generator

$$g_4(x) = (x^3 + 2x + 2)(x^3 + x^2 + x + 2) = x^6 + x^5 + x^2 + 1.$$

In particular the code has dimension $13 - 6 = 7$. Also its generator has weight 4, so its minimal distance is equal to its designed distance 4.

(8.3.5) PROBLEM. *Give the generator polynomial of the ternary, narrow-sense BCH code D_2 of length 13 with designed distance 5, using β of Example (iv) above as a primitive 13^{th} root of unity. What is the dimension of D_2?*

(8.3.6) PROBLEM. *Give the generator polynomial of the ternary, narrow-sense, primitive, BCH code D_3 of length 26 with designed distance 4, using as primitive element γ a root of the polynomial $x^3 + 2x + 1 \in \mathbb{F}_3[x]$. What is the dimension of D_3?*

(8.3.7) PROBLEM. *(a) What is the dimension of a binary, narrow-sense, primitive BCH code of length 63 and designed distance 17.*

(b) Does this code contain any codewords of weight 17? Explain your answer.

(8.3.8) PROBLEM. *Prove that a narrow-sense, primitive BCH code of length 24 over $\mathbb{F}_5$ with designed distance 3 has minimum distance 3 and dimension $20 = 24 - 2(3-1)$.*

8.4 Cyclic Hamming codes and their relatives

Cyclic binary Hamming codes and codes related to them are of particular interest.

(8.4.1) THEOREM. *For every m, there is a cyclic, binary Hamming code of redundancy m. Indeed any primitive polynomial of degree m in $\mathbb{F}_2[x]$ generates a cyclic Hamming code of redundancy m.*

PROOF. This is essentially equivalent to Corollary 8.3.4 (in view of Theorems A.3.8 and A.3.10 on the general structure and existence of finite fields). □

(8.4.2) THEOREM. *The polynomial $g(x) \in \mathbb{F}_2[x]$ generates a cyclic, binary Hamming code if and only if it is primitive.*

PROOF. In Theorem 8.4.1 we have seen that a binary primitive polynomial generates a cyclic Hamming code.

Now let C be a binary, cyclic Hamming code of length $2^m - 1 = n$. Let $g(x) = \prod_{i=1}^{r} g_i(x)$, where the $g_i(x)$ are distinct irreducible polynomials of degree m_i, so that $\sum_{i=1}^{r} m_i = m = \deg g(x)$. Then $g_i(x)$ divides $x^{n_i} - 1$ with $n_i = 2^{m_i} - 1$, hence $g(x)$ divides $x^{n_0} - 1$ where $n_0 = \prod_{i=1}^{r} n_i$. Now

$$n + 1 = 2^m - 1 + 1 = \prod_{i=1}^{r} 2^{m_i} = \prod_{i=1}^{r} (n_i + 1)\,.$$

If $r \neq 1$, then $n > n_0$ and $x^{n_0} - 1$ is a codeword of weight 2 in C, which is not the case. Therefore $g(x) = g_1(x)$ is irreducible and divides $x^n - 1$. Indeed $g(x)$ is primitive, as otherwise again there would be a code polynomial $x^p - 1$ of weight 2. □

(8.4.3) PROBLEM. *Prove that there exists a cyclic Hamming code of redundancy m and length $(q^m-1)/(q-1)$ over $\mathbb{F}_q$ if $\gcd((q^m-1)/(q-1), q-1) = 1$. (*HINT: *For construction try as before to find such a code as a subfield subcode of a Reed-Solomon code.)*

8.4.1 Even subcodes and error detection

(8.4.4) LEMMA. *Let $F = \mathbb{F}_{2^m}$ $(m > 1)$, and let $p(x)$ be a primitive polynomial of degree m in $\mathbb{F}_2[x]$. The polynomial $g(x) = (x+1)p(x)$ generates the even subcode E composed of all codewords of even weight in the Hamming code with generator $p(x)$. In particular, E has minimum weight* 4.

PROOF. The generator polynomial for E is a multiple of the generator polynomial $p(x)$ for the Hamming code, and so E is contained in the Hamming code. For any

$$c(x) = a(x)q(x) = a(x)(x+1)p(x) \in E\,,$$

we have

$$c(1) = a(1)(1+1)p(1) = 0\,.$$

Therefore E is contained in the even subcode of the Hamming code. As the codes have the same dimension, they are equal. The Hamming code has minimum weight 3, so E has minimum weight 4. □

CRC codes

The even cyclic Hamming subcodes like E have often been used for detecting errors in computer applications. In that context, they are often called *CRC codes* (for 'cyclic redundancy checking'). We devote some time to detection issues for general linear and cyclic codes.

We recall that error detection is the particularly simple type of error control in which a received word is decoded to itself if it is a codeword and otherwise a decoding default is declared. (See Problems 2.2.2 and 2.2.3.) For a linear code, this can be gauged by whether or not the received word has syndrome $\mathbf{0}$.

(8.4.5) LEMMA. *Let C be a linear code.*

(1) *C detects any error pattern that is not a codeword.*

(2) *C detects any nonzero error pattern whose nonzero entries are restricted to the complement of an information set.*

PROOF. An error pattern that is not a codeword has nonzero syndrome as does any word in its coset.

If a codeword is 0 on an information set, then it is the $\mathbf{0}$ codeword. Thus any nonzero word that is 0 on an information set is not a codeword. □

(8.4.6) LEMMA. *A cyclic code of redundancy r detects all nonzero bursts of length at most r.*

PROOF. By Lemma 8.4.5, we must show that a codeword that is a burst of length r or less must be $\mathbf{0}$. Let $\mathbf{c}$ be such a codeword. Then it has a cyclic

shift that represents a nonzero code polynomial of degree less than r. But by Theorem 8.1.1, the generator polynomial is a nonzero polynomial of minimal degree and that degree is r. Therefore $\mathbf{c} = \mathbf{0}$, as desired. □

The same argument shows that 'wrap around' burst errors, whose nonzero errors occur in a burst at the front of the word and a burst at the end, are also detected provided the combined length of the two bursts is at most r.

(8.4.7) PROBLEM. *If C is a cyclic code of redundancy r, prove that the only bursts of length $r+1$ that are codewords (and so are not detectable error patterns) are shifts of scalar multiples of the generator polynomial.*

(8.4.8) PROBLEM. *Starting with a cyclic code C of redundancy r, shorten C in its last s coordinates (or first s coordinates) by choosing all codewords that are 0 in those positions and then deleting those positions.*

*Prove that the resulting code D still can be used to detect all bursts of length at most r. (*REMARK. *The code D will no longer be cyclic and can not be relied upon for detecting burst errors that 'wrap around'.)*

(8.4.9) PROBLEM. *Have an existentialist discussion (or write such an essay) as to whether or not linear codes should be said to detect the $\mathbf{0}$ error pattern.*

Now we return to the CRC code E of Lemma 8.4.4, the even subcode of a binary cyclic Hamming code. E has redundancy $r = 1 + m$, where m is the redundancy of the Hamming code. Thus E can be used to detect:

(i) all odd weight errors,
(ii) all weight 2 errors,
(iii) most weight 4 errors,
(iv) all nonzero burst errors of length at most r,
(v) most burst errors of length $r + 1$.

Here C detects 'most' weight 4 errors because (at least for reasonably large r) the codewords of weight 4 form only a small fraction of the total number of words of weight 4. (See Problem 7.3.7; the total number of words of weight 4 is quartic in $n = 2^{r-1} - 1$, while the number of codewords of weight 4 is cubic in n.) The only bursts of length $r + 1$ that are codewords are the n shifts of the generator polynomial $g(x)$. (See Problem 8.4.7.) So we see that the various most likely error patterns are all detected.

EXAMPLES. (*i*) *CRC*-12 of length $2047 = 2^{11} - 1$ with generator polynomial

$$(x+1)(x^{11}+x^2+1) = x^{12}+x^{11}+x^3+x^2+x+1\,.$$

(*ii*) *CRC-ANSI* of length $32767 = 2^{15} - 1$ with generator polynomial

$$(x+1)(x^{15}+x+1) = x^{16}+x^{15}+x^2+1\,.$$

(*iii*) *CRC-CCITT* of length $32767 = 2^{15} - 1$ with generator polynomial

$$(x+1)(x^{15}+x^{14}+x^{13}+x^{12}+x^4+x^3+x^2+x+1) = x^{16}+x^{12}+x^5+1\,.$$

The last two examples have generator polynomials of the minimum weight 4. This is advantageous since the linear feedback circuitry required to implement encoding and decoding is simpler for generator polynomials of small weight.

As in Problem 8.4.8 the detection properties (i)-(v) are not lost by shortening E, so various shortened versions of even subcodes of binary cyclic Hamming codes are also used as CRC codes. If the code is shortened in its last s positions, then 'cyclic' encoding is still available, encoding the message polynomial $a(x)$ of degree less than $k-s$ (the dimension of the shortened code) into the code polynomial $a(x)g(x)$ of degree less than $r+k-s=n-s$ (the length of the shortened code).

8.4.2 Simplex codes and pseudo-noise sequences

As there are cyclic, binary Hamming codes of every redundancy m, there are also cyclic, binary dual Hamming codes of every dimension m. Recall that these codes are called simplex codes (or shortened first order Reed-Muller codes). They were studied previously in Section 4.3. By Theorem 8.4.2 they are precisely those cyclic, binary codes whose check polynomials are primitive.

(8.4.10) THEOREM. *Let C be a cyclic simplex code of dimension m and length $n=2^m-1$. Then C is composed of the codeword $\mathbf{0}$ plus the n distinct cyclic shifts of any nonzero codeword.*

PROOF. Let C have generator polynomial $g(x)$ and primitive check polynomial $h(x)$, so that $g(x)h(x)=x^n-1$. Since $|C|=2^m=n+1$, we need only prove that the n cyclic shifts of $g(x)$ are distinct. Suppose $g(x)=x^jg(x) \pmod{x^n-1}$, for some $0<j\leq n$. Thus

$$\begin{aligned}(x^j-1)g(x) &= q(x)(x^n-1)\\ (x^j-1)g(x) &= q(x)g(x)h(x)\\ x^j-1 &= q(x)h(x)\,.\end{aligned}$$

As $h(x)$ is primitive, we must have $j\geq n$ hence $j=n$. Therefore the n shifts $x^jg(x) \pmod{x^n-1}$, for $0\leq j<n$, are all distinct, as desired. □

(8.4.11) COROLLARY. *Let $\mathbf{0}\neq\mathbf{c}\in C$, a cyclic simplex code of dimension m. Then, for every nonzero m-tuple $\mathbf{m}$, there is exactly one set of m consecutive coordinate entries in $\mathbf{c}$ (including those that wrap around) that is equal to $\mathbf{m}$.*

PROOF. As C is cyclic, its first m coordinate positions form an information set. Every $\mathbf{m}$ occurs in these positions in exactly one codeword $\mathbf{b}$. By the theorem, $\mathbf{b}$ is a cyclic shift of $\mathbf{c}$ when $\mathbf{m}$ is one of the 2^m-1 nonzero m-tuples. The nonzero codeword $\mathbf{c}$ has only $n=2^m-1$ sets of m consecutive positions. Therefore nonzero $\mathbf{m}$ occurs exactly once among the sets of m consecutive positions in $\mathbf{c}$. □

The property described in the corollary can be thought of as a randomness property. If we were to flip an unbiased coin any number of times, then no

particular combination of m consecutive heads/tails would be expected to occur more often than any other. We will call a binary sequence of length $2^m - 1$ in which each nonzero m-tuple occurs exactly once in consecutive positions a *pseudo-noise sequence* or *PN-sequence*, for short. (Here and below, when we speak of consecutive positions, we allow these positions to wrap around from the end of the word to the front.) We call it a sequence rather than word because, when we repeat it any number of times, we get a sequence of 0's and 1's whose statistical properties mimic, in part, those of a random sequence, that is, those of noise. The length $n = 2^m - 1$ is then the *period* of the sequence.

pseudo-noise sequence

PN-sequence

period

With these definitions in hand, the corollary can be restated as

(8.4.12) COROLLARY. *A nonzero codeword from a cyclic simplex code of dimension m is a PN-sequence of period $2^m - 1$.* □

There are other consequences of the *PN* definition that are similar to properties of random sequences. A *run* is a maximal set of consecutive entries consisting entirely of 0's or entirely of 1's. The length of a run is the number of its entries. In a random sequence, one would expect, for a fixed length, the same number of runs of 0's as 1's and that runs of length p would be twice as likely as runs of length $p + 1$.

run

(8.4.13) PROPOSITION. *Let $\mathbf{s}$ be a PN-sequence of period $2^m - 1$.*

(1) (Run balance) *There are exactly 2^{m-p-2} runs of 0's of length p ($\leq m-2$) and exactly 2^{m-p-2} runs of 1's of length p ($\leq m-2$). The sequence $\mathbf{s}$ contains exactly 2^{m-1} runs.*

(2) (General balance) *If $\mathbf{p}$ is a nonzero p-tuple with $p \leq m$, then $\mathbf{p}$ occurs in consecutive positions of $\mathbf{s}$ exactly 2^{m-p} times. If $\mathbf{p}$ is a p-tuple of 0's, then it occurs in consecutive positions exactly $2^{m-p} - 1$ times. In particular, $\mathbf{s}$ has weight 2^{m-1}.*

PROOF. For (2), the p-tuple $\mathbf{p}$ is the initial segment of 2^{m-p} distinct m-tuples. If $\mathbf{p}$ is nonzero, then each of these m-tuples occurs within $\mathbf{s}$. If $\mathbf{p} = \mathbf{0}$, then the m-tuple $\mathbf{0}$ is the only completion of $\mathbf{p}$ that does not occur within $\mathbf{s}$. In particular, the 1-tuple 1 occurs exactly 2^{m-1} times, completing (2).

A run $aa\cdots aa$ of length p ($\leq m-2$) corresponds to a $(p+2)$-tuple $baa\cdots aab$ with $\{a, b\} = \{0, 1\}$ (which is never $\mathbf{0}$). Therefore by (2), the number of runs $aa\cdots aa$ of length p is $2^{m-(p+2)} = 2^{m-p-2}$.

If $m = 1$, then certainly there is only one run. For $m \geq 2$, a transition between two runs occurs precisely when we encounter either 01 or 10. By (2) there are 2^{m-2} of each. Therefore the number of runs, being equal to the number of transitions, is $2^{m-2} + 2^{m-2} = 2^{m-1}$. □

Although pseudo-noise and pseudo-random sequences have been studied a great deal, there is no consensus about the terminology or definitions. In some places there is no distinction made between *PN*-sequences in general and those special ones coming from simplex codes (so that Corollary 8.4.12 becomes the

definition). We will call the nonzero codewords of cyclic simplex codes *m-sequences*. (This is an abbreviation for *maximal length feedback shift register sequences*.) m-sequences

It might seem more natural to consider sequences of length 2^m in which every m-tuple occurs. Such sequences are called *DeBruijn cycles*. The two concepts are in fact equivalent. If in a *PN*-sequence $\mathbf{s}$ of period $2^m - 1$ we locate the unique run of $m-1$ 0's, then we can construct a DeBruijn cycle by inserting one further 0 into the run. Conversely, if we delete a 0 from the unique run of m 0's in a DeBruijn cycle, we are left with a *PN*-sequence. Given the connection with simplex codes, we prefer the present formulation. DeBruijn cycles

(8.4.14) PROBLEM. *Prove that every PN-sequence of length* 7 *is an m-sequence.* (REMARK. *Up to cycling, there are* 16 *PN-sequences of length* 15*, only* 2 *of which are m-sequences.)*

An important property of m-sequences is not shared by all *PN*-sequences.

(8.4.15) LEMMA. (Shift-and-add property) *If* $\mathbf{s}$ *is an m-sequence and* $\mathbf{s}'$ *is a cyclic shift of* $\mathbf{s}$*, then* $\mathbf{s}+\mathbf{s}'$ *is also a cyclic shift of* $\mathbf{s}$ *(or* $\mathbf{0}$*). In particular nonzero* $\mathbf{s}+\mathbf{s}'$ *is itself an m-sequence.*

PROOF. This is a direct consequence of Theorem 8.4.10. □

(8.4.16) PROBLEM. *Prove that a PN-sequence possessing the shift-and-add property must be an m-sequence.*

One last property is more striking if we change to the $\pm$-version of the simplex code, as described in Section 4.3. With each binary sequence $\mathbf{s}$ we associate the ± 1-sequence $\mathbf{s}^*$ by replacing each 0 with the real number 1 and each 1 with the real number -1. If $\mathbf{s}$ is an m-sequence, then we abuse terminology by also referring to the associated sequence $\mathbf{s}^*$ as an m-sequence.

(8.4.17) PROPOSITION. (Perfect autocorrelation)
If $(s_0, \ldots, s_{n-1}) = \mathbf{s}^* \in \{\pm 1\}^n \subset \mathbb{R}^n$ *is an m-sequence with* $n = 2^m - 1$*, then*

$$\begin{array}{rcll} \sum_{i=0}^{n-1} s_i s_{i+p} & = & n & \text{for } p = 0 \\ & = & -1 & \text{for } 0 < p < n\,, \end{array}$$

where indices are all read modulo n*.*

PROOF. The summation is the dot product of $\mathbf{s}^*$ with a cyclic shift of itself. The associated binary vectors are also cyclic shifts and are either equal (when $p = 0$) or at Hamming distance 2^{m-1} by Proposition 8.4.13(2) and Lemma 8.4.15. By Lemma 4.3.4 the dot product is $n\,(= 2^m - 1)$ when $p = 0$ and otherwise $(2^m - 1) - 2 \cdot 2^{m-1} = -1$. (In fact this proposition is nearly equivalent to Lemma 4.3.5.) □

The function $a(p) = \sum_{i=0}^{n-1} s_i s_{i+p}$, for $0 \le p < n$, of the proposition is the *autocorrelation function* for $\mathbf{s}^*$. As n is odd, the sequences could never be autocorrelation function

orthogonal; so the proposition says, in a sense, that $\mathbf{s}^*$ and its shifts are as close to being orthogonal as possible. This is why the autocorrelation function is called perfect.

Thus the ± 1 m-sequences are very unlike nontrivial shifts of themselves. For this reason, they are at times used for synchronization of signals. They are also used for modulation of distinct signals in multiple user situations. This is an example of spread spectrum communication. The idea is that multiplication by a *PN*-sequence will make a coherent signal look noise-like (taking its usual spiked frequency spectrum and spreading it out toward the flat spectrum of noise).

For such applications, it is often helpful to have not just one sequence with good autocorrelation properties but large families of them with good crosscorrelation properties. The constructions of such families may start from nice m-sequences. Their investigation is of on-going interest.

Pseudo-random binary sequences are also important for cryptography. In that context m-sequences are bad, since the shift-and-add property implies that they have low computational complexity.