Codes and Fields

Part 3
The BCH Family of Codes

Oliver Pretzel

Chapter 3.1

BCH-Codes as subcodes of Hamming Codes

IDEA. Extend the Hamming check matrix to reduce the dimension of the code and increase the number of errors it can correct.

Recall that in Chapter 1.3 Section 5, we showed that extending a check matrix by rows that are linear combinations of the original rows does not change the code. So we will have to find a non-linear extensior. Nevertheless, it should be algebraic, so that we have a chance of constructing an efficient decoder to find multiple errors.

1. EXAMPLE. Consider the columns of the Hamming check-matrix H_k as representing the non-zero elements of a finite field of order 2^k. To use GF(16), we should start with H_4.

We shall feel free to permute these columns to produce a nice order, as this only means permuting the bits of a code word. For our present purposes the best way to arrange the elements is as powers of a primitive element α, say 2, starting at the right. The check matrix H_4 is:

$$\begin{bmatrix} 12 & 6 & 3 & 13 & 10 & 5 & 14 & 7 & 15 & 11 & 9 & 8 & 4 & 2 & 1 \end{bmatrix},$$

which is the same as:

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

Now we wish to add a further four rows to H_4 to double its decoding capability. It seems reasonable that if we need four rows for the first error we shall need another four for the second, although this may not always be strictly true. Of course we interpret these new rows as elements of GF(16) again. So we get a matrix with columns of the form:

$$\begin{pmatrix} \alpha^i \\ T(\alpha^i) \end{pmatrix},$$

where T is some function which must not be linear. Try taking powers for T. Squaring will not do because $(\beta+\gamma)^2 = \beta^2+\gamma^2$, which is linear. But cubing does work. We hopefully define the double error BCH-code BCH(4,2) to have the check matrix $H_{4,2}$ with columns:

$$\begin{pmatrix} \alpha^i \\ \alpha^{3i} \end{pmatrix}.$$

Thus we get as our check matrix the matrix $H_{4,2}$:

$$\begin{bmatrix} 12 & 6 & 3 & 13 & 10 & 5 & 14 & 7 & 15 & 11 & 9 & 8 & 4 & 2 & 1 \\ 3 & 5 & 15 & 8 & 1 & 3 & 5 & 15 & 8 & 1 & 3 & 5 & 15 & 8 & 1 \end{bmatrix},$$

or in binary:

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

2. How can we prove that this will work? First, notice that when we multiply a vector by $H_{4,2}$, we can save time and effort by multiplying in GF(16) and then expanding the result in binary afterwards. Also remember that squaring was linear and so we can extend the matrix to one which we shall call $V_{4,2}$ without changing the code it checks for. This matrix, $V_{4,2}$ is not used in practice for BCH-codes, but it is theoretically valuable and it *is* used for Reed-Solomon codes. It has the columns:

$$\begin{matrix} \alpha^i \\ \alpha^{2i} \\ \alpha^{3i} \\ \alpha^{4i} \end{matrix}.$$

PROPOSITION. (a) *$V_{4,2}$ is a check matrix for the same code as $H_{4,2}$.*

(b) *No two error distinct patterns of weight ≤ 2 produce the same syndromes with $V_{4,2}$. Hence the code can correct double errors.*

Proof. (a) The block lengths of the two codes are both 15. For reasons that will appear below we number the entries of the words from 14 to 0 in descending order. Let $w = (b_{14}, b_{13}, ..., b_1, b_0)$ be a word. Then w is a code word iff $wH_{4,2} = \underline{0}$. This can be rewritten as two equations (using α instead of 2 for clarity):

$$\sum b_i \alpha^i = 0,$$

and

$$\sum b_i \alpha^{3i} = 0.$$

Now as $b_i = 0, 1$, it follows that $b_i = b_i^2$. So if $\sum b_i \alpha^i = 0$, then also $\sum b_i \alpha^{2i} = (\sum b_i \alpha^i)^2 = 0$, and $\sum b_i \alpha^{4i} = (\sum b_i \alpha^{2i})^2 = (\sum b_i \alpha^i)^4 = 0$. Thus $wH_{4,2} = 0$ implies $wV_{4,2} = 0$. The converse is obvious because $V_{4,2}$ contains $H_{4,2}$.

(b) Two syndromes are the same iff their sum is 0. The sum of two error patterns of weight ≤ 2 is an error pattern of weight ≤ 4. So we must show no non-zero error pattern of weight ≤ 4 produces a zero syndrome. We use $V_{4,2}$. Suppose the errors occur among the columns i, j, k, l (if there are fewer than 4 errors, we add in a column with a zero error). So let the errors be e_i, e_j, e_k, and e_l. Then the syndrome is:

$$\begin{aligned} &e_i\alpha^i + e_j\alpha^j + e_k\alpha^k + e_l\alpha^l, \quad e_i\alpha^{2i} + e_j\alpha^{2j} + e_k\alpha^{2k} + e_l\alpha^{2l}, \\ &e_i\alpha^{3i} + e_j\alpha^{3j} + e_k\alpha^{3k} + e_l\alpha^{3l}, \quad e_i\alpha^{4i} + e_j\alpha^{4j} + e_k\alpha^{4k} + e_l\alpha^{4l} \\ &= 0, 0, 0, 0. \end{aligned}$$

We rewrite this as a matrix equation:

$$\begin{pmatrix} \alpha^i & \alpha^j & \alpha^k & \alpha^l \\ \alpha^{2i} & \alpha^{2j} & \alpha^{2k} & \alpha^{2l} \\ \alpha^{3i} & \alpha^{3j} & \alpha^{3k} & \alpha^{3l} \\ \alpha^{4i} & \alpha^{4j} & \alpha^{4k} & \alpha^{4l} \end{pmatrix} \begin{pmatrix} e_i \\ e_j \\ e_k \\ e_l \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}.$$

But the matrix in this equation is a Vandermonde matrix with determinant

$$\alpha^{i+j+k+l}(\alpha^j - \alpha^i)(\alpha^k - \alpha^i)(\alpha^k - \alpha^j)(\alpha^l - \alpha^i)(\alpha^l - \alpha^j)(\alpha^l - \alpha^k),$$

(see Exercise Sheet 6). It is non-zero because all its factors are. Hence it follows that

$$e_i = e_j = e_k = e_l = 0. \qquad \square$$

3. This idea can clearly be extended to produce codes of block length $2^k - 1$ correcting any reasonable number t of errors per block.

DEFINITION. The t–error correcting BCH-code over the field of order 2^k based on the primitive element α, BCH(k, t) has as its check matrix the matrix $H_{k,t}$ with columns

$$\begin{matrix} \alpha^i \\ \alpha^{3i} \\ \vdots \\ \alpha^{(2t-1)i} \end{matrix}.$$

we take the index i in descending order from $2^k - 1$ to 0. The code also has the extended (Vandermonde type) check matrix $V_{k,t}$ with columns

$$\begin{matrix} \alpha^i \\ \alpha^{2i} \\ \alpha^{3i} \\ \vdots \\ \alpha^{(2t-1)i} \\ \alpha^{2ti} \end{matrix}$$

Obviously we must have $2t < 2^k$ for this definition to be reasonable.

There are now three major questions:

QUESTION 1. What is the dimension of the code?

If it is too small then the code will be hopelessly inefficient, for example if the dimension is 1 then the code has become the $2^k - 1$ repetition code.

QUESTION 2. What is the minimum distance of the code?

If the code is to correct t errors it must be at least $2t + 1$.

QUESTION 3. How can we correct errors in received words efficiently?

Just knowing that the code is capable of correcting t errors without a practical correcting algorithm is of little use.

The following theorem gives answers to Questions 1 and 2. The answer to Question 1 is poor and we shall improve it in the next chapter. The answer to Question 2 is not perfect, but it is very hard to improve, and although there are theorems giving better values, there are no corresponding algorithms.

THEOREM. *BCH(k,t) has block length $n = 2^k - 1$ and binary dimension $\geq n - kt$.*

It can correct all error patterns of weight $\leq t$, and so has minimum distance $\geq 2t + 1$.

Proof. $n = 2k - 1$, because that is then number of distinct powers of α.

The rank of $H_{k,t}$ is at most equal to the number of rows in binary. This is kt. The Rank and Nullity Theorem of linear algebra states that for any matrix A the dimension of the space of solutions of the equation $vA = 0$ is

$$\text{the number of columns of } A - \text{the rank of } A.$$

In our case this gives the dimension of the code as $m - \text{rank}(H_{k,t})$.

The argument for the error correction parallels that in the example. Two error patterns of weight $\leq t$ should only produce the same syndrome if they are equal. The sum is an error pattern of weight $\leq 2t$. We must show that this only produces a zero syndrome when the error pattern itself is 0. Let the error pattern be in

positions $i(1),...,i(2t)$. We shall denote the power $\alpha^{i(k)}$ by α_k, and the error value 0 or 1 in the corresponding position by e_k. Then we have a system of equations:

$$\begin{pmatrix} \alpha_1 & \alpha_2 & \dots & \alpha_{2t} \\ \alpha_1^2 & \alpha_2^2 & \dots & \alpha_{2t}^2 \\ & & \dots & \\ \alpha_1^{2t} & \alpha_2^{2t} & \dots & \alpha_{2t}^{2t} \end{pmatrix} \begin{pmatrix} e_1 \\ e_2 \\ \cdot \\ e_{2t} \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ \cdot \\ 0 \end{pmatrix}.$$

The matrix of this system is a Vandermonde matrix with non-zero determinant (see Exercise Sheet 6. Thus the only solution is $(e_1, \dots, e_{2t}) = (0, ..., 0)$. □

Chapter 3.2

BCH-codes as Cyclic Codes

The precise definition of a cyclic code will be deferred until the end of the chapter. It will then become evident that the properties of BCH-codes established in this section, qualify them as such codes.

1. The generator polynomial

In the last chapter we showed that checking whether a binary word $(u_m, \ldots, u_0)$ was a codeword of BCH(k,t), where $m = 2^k - 2$, is equivalent to verifying the equations

$$\sum_i u_i \alpha^{ij} = 0,$$

for certain powers α^j of a primitive element α. This makes it natural to identify codewords with binary polynomials $u(x)$ of deg $< m$. So that the equations can be rewritten

$$u(\alpha^j) = 0,$$

where $u(x) = u_m x^m + \cdots + u_1 x + u_0$.

PROPOSITION. *Let $g(x)$ be the product of the distinct minimal polynomials of $\alpha, \alpha^3, \ldots, \alpha^{2t}$ (each polynomial is taken only once, even if it occurs as minimal polynomial several times). Then a polynomial $u(x)$ of degree $\leq m$ represents a codeword of* BCH(k,t) *iff $g(x) \mid u(x)$.*

Proof. If $g(x)$ divides $u(x)$, then $u(\alpha^j) = 0$ for $j = 1, 3, \ldots, 2t-1$, as that is true of $g(x)$. Hence $u(x)$ represents a codeword. Conversely if $u(\alpha^j) = 0$ for $j = 1, 3, \ldots, 2t-1$, then each of the irreducible factors of $g(x)$ divides $u(x)$. It follows from the next lemma that their product $g(x)$ also does so. □

LEMMA. *If $a(x)$ and $b(x)$ are polynomials with $(a(x), b(x)) = 1$ and $a(x) \mid c(x)$ and $b(x) \mid c(x)$, then $a(x)b(x) | c(x)$.*

Proof. By Euclid's Algorithm, there exist polynomials $s(x)$ and $t(x)$ such that $a(x)s(x) + b(x)t(x) = 1$. Hence $s(x)a(x)c(x) + t(x)b(x)c(x) = c(x)$. Now $b(x)$ divides $c(x)$. So $a(x)b(x)$ divides the first summand and $a(x)$ divides $c(x)$ so $a(x)b(x)$ divides the second summand. □

DEFINITION. The polynomial $g(x)$ is called the *generator polynomial* of BCH(k,t).

COROLLARY. *The dimension of* BCH(k,t) *is* $2^k - \deg g(x) - 1$, *where* $g(x)$ *is its generator polynomial.*

Proof. We can obtain all the codewords by multiplying $g(x)$ by all polynomials of degree $2^k - \deg g(x) - 2$. These correspond to message words of length $2^k - \deg g(x) - 1$. □

EXAMPLE. Consider BCH$(4,3)$ based on the primitive element $\alpha = 2$ in GF(16). Its generator polynomial is

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

Hence the dimension of the code is 5, even though Theorem 3.1.4 gives a bound of 15 - 12 = 3.

We could obtain all codewords by multiplying $g(x)$ by all binary polynomials of degree ≤ 4 as in the corollary, but this does not give 'systematic' encoding, where the message bits form the first five bits of the codeword. The Example Sheet shows how one can do that.

2. The check polynomial

We know from the Little Fermat Theorem that $Q(x) = x^{q-1} - 1$ has all the non-zero elements of the field of order q as its roots. So it must be a multiple of the generator polynomial $g(x)$ of BCH(k,t), where $q = 2^k$.

NOTE. $Q(x)$ is not a codeword of BCH(k,t). Why?

Let us write $Q(x) = g(x)h(x)$. Then $u(x)$ is a codeword iff $g(x) \mid u(x)$ which is true iff $Q(x)$ divides $u(x)h(x)$. That is easy to check as it holds iff the coefficients of $u(x)h(x)$ repeat after q steps.

DEFINITION. The polynomial $h(x)$ is called the check polynomial of BCH(k,t).

EXAMPLE. For BCH(4, 3) the check polynomial $h(x)$ is:

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

Consider the codeword:

$$(0\ 0\ 0\ 0\ 1 \quad 1\ 1\ 0\ 1\ 1 \quad 0\ 0\ 1\ 0\ 1).$$

Multiplying by $h(x)$ we get:

		0	0	0	0	1	1	1	0	1	1	0	0	1	0	1
+		0	0	1	1	1	0	1	1	0	0	1	0	1	0	0
+		1	1	1	0	1	1	0	0	1	0	1	0	0	0	0
+	1	1	1	0	1	1	0	0	1	0	1	0	0	0	0	0
$Q(x) =$	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1.

3. Polynomial Codes

DEFINITION. We can always consider code words as polynomials of appropriate length but this does not necessarily benefit us much. We shall therefore call a code of block length n a *polynomial code* if, when this is done, the code consists of all the multiples of degree $< n$ of a single polynomial $g(x)$, the *generator polynomial.*

We define a *left shift* of a code word $(a, b, \ldots, z)$ to be the word $(b, \ldots, z, a)$. Notice that the entries 'wrap around'.

PROPOSITION. (a) *Polynomial codes are linear.*

(b) *A linear code is a polynomial code iff for every codeword beginning with 0 the left shift is also a code word.*

Proof. (a) The sum of two multiples of $g(x)$ is a multiple of $g(x)$. A scalar multiple of a multiple of $g(x)$ is a multiple of $g(x)$.

(b) In polynomial terms this corresponds to multiplying by x, so it is true for polynomial codes.

Suppose conversely that a code C has this property. Let $f(x)$ be the polynomial corresponding to a code word. Any multiple of $f(x)$ can be obtained by multiplying by powers of x, multiplying by scalars, and adding. By assumption, provided the degree never goes above $n - 1$, this will always produce a codeword.

Let $g(x)$ be a non-zero codeword of smallest possible degree. Let $f(x)$ be any codeword. Then $f(x) = q(x)g(x) - r(x)$, where $r(x) = 0$, or $\deg(r(x)) < \deg(g(x))$. By the above $q(x)g(x)$ is a codeword. Hence $r(x)$ is a codeword (C is linear). As $g(x)$ has minimal degree among the non-zero codewords $r(x) = 0$. So $g(x)$ is a generator for C and C is a polynomial code. □

4. Cyclic codes

DEFINITION. If the generator polynomial $g(x)$ of a polynomial code of block length n divides $x^n - 1$, then the code is called *cyclic*. In that case the polynomial $h(x)$ such that $h(x)g(x) = x^n - 1$ is called the *check polynomial*.

For a cyclic code it is possible to check whether a word is a codeword by multiplying it by $h(x)$ to see whether the result is a multiple of $x^n - 1$. That is easily checked, because the result will be of the form $f(x)x^n - f(x)$, where $f(x)$ is a polynomial of degree $\leq n - \deg(g(x))$.

PROPOSITION. *A linear code is cyclic iff for any codeword u the left shift of u is also a codeword.*

Remark. That explains the name 'cyclic'.

Proof. Suppose the code is cyclic, with generator polynomial $g(x)$, Let u be a codeword. If u starts with 0, then its left shift is a codeword by Proposition 3. So assume u starts with a non-zero symbol a. If $f(x)$ is the polynomial corresponding to u, then the polynomial corresponding to its left shift is

$$xf(x) - ax^n + a,$$

where the first term produces the shift proper and the two last terms produce the 'wrap around'. By assumption, $g(x)$ divides $xf(x)$ and also $-ax^n + a$, so it divides the polynomial of the left shift.

Conversely suppose every left shifted codeword is a codeword. Then the code is a polynomial code with generator, say, $g(x)$ with highest coefficient $a \neq 0$. We must show $g(x)$ divides $x^n - 1$. Let the degree of $g(x)$ be s. Consider the polynomial

$$f(x) = x^{n-s-1}g(x).$$

This corresponds to a codeword u. As before the left shift of u corresponds to

$$xf(x) - ax^n + a.$$

Now this is, by assumption, a codeword and so divisible by $g(x)$. Also $xf(x)$ is divisible by $g(x)$ as $f(x)$ is. Hence $g(x)$ divides $ax^n - a$, and since we can divide by non-zero constants, it divides $x^n - 1$. □

The Code BCH(4,3)

BCH(4,3) as a Linear Code

The check matrix of BCH(4,3) is $H = H_{4,3}$:

$$\begin{bmatrix} 12 & 6 & 3 & 13 & 10 & 5 & 14 & 7 & 15 & 11 & 9 & 8 & 4 & 2 & 1 \\ 3 & 5 & 15 & 8 & 1 & 3 & 5 & 15 & 8 & 1 & 3 & 5 & 15 & 8 & 1 \\ 10 & 11 & 1 & 10 & 11 & 1 & 10 & 11 & 1 & 10 & 11 & 1 & 10 & 11 & 1 \end{bmatrix}.$$

The code is actually binary. So here is H in binary:

$$\left[\begin{array}{ccccccccccccccc} 1 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 & 0 & 1 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \\ \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 & 1 & 1 & 1 & 1 & 0 & 1 & 1 & 1 & 1 & 0 & 1 \\ \\ 1 & 1 & 0 & 1 & 1 & 0 & 1 & 1 & 0 & 1 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 1 & 1 & 0 & 1 & 1 & 0 & 1 & 1 & 0 & 1 & 1 & 0 \\ 0 & 1 & 1 & 0 & 1 & 1 & 0 & 1 & 1 & 0 & 1 & 1 & 0 & 1 & 1 \end{array}\right].$$

It is easy to see directly that several columns of H are superfluous. It is not so easy to find a generator matrix.

BCH(4,3) as a Polynomial Code

The generator polynomial $g(x)$ is:

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

This gives a generator matrix $G = G_{4,3}$. Here is $G_{4,3}^{\mathsf{T}}$:

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

The matrix is obtained by writing down the coefficients of $g(x)x^j$ for $j = 4, \ldots, 0$. Note that neither the generator nor the check matrix are systematic.

MULTIPLICATIVE ENCODING. Multiply a message word by $g(x)$. Example to encode 1 0 1 1 1:

```
                1 1 1 0 1 1 0 0 1 0 1  (g(x))
                      1 0 1 1 1
-------------------------------------
          1 1 1 0 1 1 0 0 1 0 1
        1 1 1 0 1 1 0 0 1 0 1
      1 1 1 0 1 1 0 0 1 0 1
1 1 1 0 1 1 0 0 1 0 1
-------------------------------------
1 1 0 0 0 1 0 0 1 1 0 1 0 1 1
```

SYSTEMATIC ENCODING. Take a message word $m(x)$. Multiply it by x^{10} (ie. shift to left ten steps), divide the result by $g(x)$ and subtract off the remainder to get a polynomial starting with $m(x)x^{10}$ and divisible by $g(x)$.For instance message: 1 1 0 0 0:

```
1 1 1 0 1 1 0 0 1 0 1)1 1 0 0 0 0 0 0 0 0 0 0 0 0 0
                       1 1 1 0 1 1 0 0 1 0 1
                     ---------------------------
                     0 0 1 0 1 1 0 0 1 0 1 0 0
                         1 1 1 0 1 1 0 0 1 0 1
                       -----------------------
                       0 1 0 1 1 1 1 0 0 0 1 0
                           1 1 1 0 1 1 0 0 1 0 1
                         -------------------------
                         0 1 0 1 0 0 0 0 1 1 1 0
                             1 1 1 0 1 1 0 0 1 0 1
                           -----------------------
                           0 1 0 0 1 1 0 1 0 1 1
```

Codeword: 1 1 0 0 0 1 0 0 1 1 0 1 0 1 1.

The check polynomial is:

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

CHECKING. To use the check polynomial to check a word $u(x)$ multiply the polynomial by $h(x)$ and see if the coefficients repeat from x^{15}.

We shall use this test on the codeword we constructed.

```
                  1 1 0 0 0 1 0 0 1 1 0 1 0 1 1
                1 1 0 0 0 1 0 0 1 1 0 1 0 1 1 0 0
              1 1 0 0 0 1 0 0 1 1 0 1 0 1 1 0 0 0 0
            1 1 0 0 0 1 0 0 1 1 0 1 0 1 1 0 0 0 0 0
-------------------------------------------------------
1 0 1 1 1 0 0 0 0 0 0 0 0 0 0 0 1 0 1 1 1
```

This shows the repeat pattern confirming it is a codeword. Note that the repeated word is the message word if multiplicative encoding was used.

Below is a table of the codewords of BCH(4,3). Choose a few codewords. Check their validity as above. Check also that their sums are codewords. How can you see directly that this code has minimum distance 7?

Codewords of BCH(4,3)

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0,
0 0 0 0 1 1 1 0 1 1 0 0 1 0 1,
0 0 0 1 0 0 1 1 0 1 0 1 1 1 1,
0 0 0 1 1 1 0 1 1 0 0 1 0 1 0;

0 0 1 0 0 1 1 0 1 0 1 1 1 1 0,
0 0 1 0 1 0 0 0 0 1 1 1 0 1 1,
0 0 1 1 0 1 0 1 1 1 1 0 0 0 1,
0 0 1 1 1 0 1 1 0 0 1 0 1 0 0;

0 1 0 0 0 0 1 1 1 0 1 1 0 0 1,
0 1 0 0 1 1 0 1 0 1 1 1 1 0 0,
0 1 0 1 0 0 0 0 1 1 1 0 1 1 0,
0 1 0 1 1 1 1 0 0 0 1 0 0 1 1;

0 1 1 0 0 1 0 1 0 0 0 0 1 1 1,
0 1 1 0 1 0 1 1 1 1 0 0 0 1 0,
0 1 1 1 0 1 1 0 0 1 0 1 0 0 0,
0 1 1 1 1 0 0 0 1 0 0 1 1 0 1;

1 0 0 0 0 1 1 1 0 1 1 0 0 1 0,
1 0 0 0 1 0 0 1 1 0 1 0 1 1 1,
1 0 0 1 0 1 0 0 0 0 1 1 1 0 1,
1 0 0 1 1 0 1 0 1 1 1 1 0 0 0;

1 0 1 0 0 0 0 1 1 1 0 1 1 0 0,
1 0 1 0 1 1 1 1 0 0 0 1 0 0 1,
1 0 1 1 0 0 1 0 1 0 0 0 0 1 1,
1 0 1 1 1 1 0 0 0 1 0 0 1 1 0;

1 1 0 0 0 1 0 0 1 1 0 1 0 1 1,
1 1 0 0 1 0 1 0 0 0 0 1 1 1 0,
1 1 0 1 0 1 1 1 1 0 0 0 1 0 0,
1 1 0 1 1 0 0 1 0 1 0 0 0 0 1;

1 1 1 0 0 0 1 0 0 1 1 0 1 0 1,
1 1 1 0 1 1 0 0 1 0 1 0 0 0 0,
1 1 1 1 0 0 0 1 0 0 1 1 0 1 0,
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1.

Chapter 3.3

Decoding BCH-Codes

Decoding Hamming Codes is easy, but to use the same method we would have to look up $\binom{q-1}{t}$ combinations of rows for a BCH(k,t) with $q = 2^k - 1$. With Reed-Solomon codes, the difficulty is compounded by the fact that the error values in each place can be any of $q-1$ values. In this chapter we present an algorithm based on Euclid's algorithm due to Sugiyama, Kasahara, Hirasawa and Namekawa (1975). It is theoretically equivalent to the fastest practical algorithm: the Berlekamp-Massey algorithm.

1. Set-up for BCH(k,t) based on primitive element $\alpha \in F = GF(q)$ $(q = 2^k,\ m = q-2)$.

Transmitted Codeword	:	$c(x) = c_m x^m + \cdots + c_0$
Received Word	:	$d(x) = d_m x^m + \cdots + d_0$
Error Word	:	$d(x) - c(x) = e(x) = e_m x^m + \cdots + e_0$
Error Positions	:	$M = \{i : e_i \neq 0\},$
	:	$s = \lvert M \rvert \leq t$
Error Locator Polynomial	:	$l(z) = \prod_{i \in M} (1 - \alpha^i z)$
This has roots	:	$\{\alpha^{-i} : i \in M\}$
Error Evaluator Polynomial	:	$w(z) = \sum_{i \in M} e_i \alpha^i \prod_{j \in M \setminus i} (1 - \alpha^j z)$

We shall show that the value of the error e_i, which is 1 for BCH-codes, but may be any field element for Reed-Solomon codes can be calculated from $l(z)$ and $w(z)$

Syndromes : $d(\alpha^i) = e(\alpha^i) = S_i$ for $i = 1, \ldots, 2t$

Note that for BCH-Codes the even syndromes S_{2j} can be calculated by squaring S_j.

Syndrome Polynomial : $s(z) = \sum_{i=1}^{2t} S_i z^{i-1}$

Fundamental Problem : Given $s(z)$ find $l(z)$ and $w(z)$.

Examples for BCH(4,3) are calculated on a separate sheet.

2. PROPOSITION Basic Properties of $l(z)$ and $w(z)$.

(a) $\deg(l) = s \leq t$, $deg(w) < s$.

(b) $l(0) = 1$, $l(\alpha^i) = 0 \Rightarrow w(\alpha^i) \neq 0$, *hence* $(l(z), w(z)) = 1$.

(c) $l(z)s(z) - w(z)$ *is divisible by* z^{2t} *in* $F[x]$.

Remark. (c) states that the first $2t$ coefficients of $l(z)s(z)$ and $w(z)$ are the same. It is called the fundamental property of l and w and is the reason they are chosen as they are.

Proof.

(a) □

(b) l splits into linear factors with non-zero roots. The value of w at a root α^{-i} of l is

$$e_i\alpha^i \prod_{j\in M\setminus i} (1 - \alpha^{j-i}) \neq 0.$$

Thus they have no common factors. □

(c) $(1 - \alpha^i z)\sum_{j=1}^{2t} \alpha^{ij} z^{j-1} = \alpha^i - \alpha^{i(2t+1)} z^{2t} \equiv \alpha^i \pmod{z^2 t}$.
We write this more suggestively as a formula for a geometric progression:

$$\sum_{j=1}^{2t} \alpha^{ij} z^{j-1} \equiv \frac{\alpha^i}{(1 - \alpha^i z)}$$

Hence

$$\begin{aligned}
l(z)s(z) &= \prod_{i\in M}(1 - \alpha^i z)\sum_{j=1}^{2t} S_j z^{j-1} \\
&= \prod_{i\in M}(1 - \alpha^i z)\sum_{j=1}^{2t}\sum_{l\in M} e_l\alpha^{lj} z^{j-1} \\
&= \prod_{i\in M}(1 - \alpha^i z)\sum_{l\in M}\sum_{j=1}^{2t} e_l\alpha^{lj} z^{j-1} \\
&= \prod_{i\in M}(1 - \alpha^i z)\sum_{l\in M} e_l\frac{\alpha^l}{(1 - \alpha^l z)} \\
&= \sum_{l\in M} e_l\alpha^l \prod_{i\in M\setminus l}(1 - \alpha^i z) \\
&= w(z)
\end{aligned}$$

□

3. PROPOSITION Uniqueness of $l(z)$ and $w(z)$. *If $l^\circ(z)$ and $w^\circ(z)$ satisfy conditions (a) and (c) of Proposition 2, thenn there exists a polynomial $k(z)$ such that $l^\circ(z) = k(z)l(z)$ and $w^\circ(z) = k(z)w(z)$.*

Proof. z^{2t} divides $ls - w$ and $l^\circ s - w^\circ$. Multiplying the first of these expression by l° and the second by l and subtracting we find that z^{2t} divides $l^\circ w - lw^\circ$. But the degree of $l^\circ w < 2e \leq 2t$ and the same holds for lw°. Hence

$$l^\circ(z)w(z) = l(z)w^\circ(z).$$

Thus $l(z)$ divides $l^\circ(z)w(z)$. It follows from the lemma below that $l(z)$ divides $l^\circ(z)$, say $l^\circ(z) = l(z)k(z)$. Then $k(z)l(z)w(z) = l(z)w^\circ(z)$ so $w^\circ(z) = w(z)k(z)$. □

LEMMA. *If $(f(x), g(x)) = 1$ and $f(x) \mid g(x)h(x)$, then $f(x) \mid h(x)$.*

Proof. By Euclid's Algorithm there are polynomials $s(x)$, $t(x)$ such that $s(x)f(x) + t(x)g(x) = 1$. Thus $h(x) = s(x)f(x)h(x) + t(x)g(x)h(x)$. But by assumption $f(x)$ divides both summands of the RHS. So it divides the LHS. □

COROLLARIES. (a) *If l° and w° also satisfy $(l^\circ, w^\circ) = 1$ they differ from l and w by a constant K. Hence l and w are uniquely determined by the properties 2(a), (b) and (c).*

(b) *If there are e errors, $(1 \leq e \leq t)$, then* $\deg(s(z)) \geq t$. For otherwise $l = 1$ and $w = S$, but $l = 1 \Rightarrow$ no errors. (Why?) □

4. ALGORITHM for decoding BCH(k, t).

It is assumed that $\leq t$ errors occurred in $R(x)$. If that assumption is false the algorithm may decode incorrectly or it may fail giving an error diagnosis. The modes of failure are discussed later.

An Example of the algorithm at work is given on a separate page.

Step 1. Calculate $S_i = d(\alpha^i)$ for $i = 1, 3, \ldots, 2t - 1$.

Calculate $S_{2j} = S_j^2$ for $i = 1, 2, ..., t$.

Put $s(z) = \sum_{i=1}^{2t} S_i z^{i-1}$.

If $s(z) = 0$, no errors.

Step 2. Apply Euclid's Algorithm to $a(z) = z^{2t}$ and $b(z) = s(z)$.

Finish at the first stage where $r_j(z)$ has degree $< t$.

If $r_j = 0$, there are more than t errors: STOP.

Step 3. Put $l^\circ(z) = v_j(z)$. Find the roots of $l^\circ(z)$: $\beta_1, \ldots, \beta_s$.

Step 4. If $\beta_i = \alpha^{p(i)}$, then the errors occurred at the places $2^k - p(i) - 1$, $i = 1 \ldots s$, counting from the right starting at 0.

Remark. An extended form of this algorithm will be used for Reed-Solomon codes.

The next proposition is a proof that the algorithm works.

PROPOSITION. *Assume $1 \leq s \leq t$ errors occurred. Then*
(a) *Step 2 of the algorithm will end with a non-zero r_j,*
(b) $\deg(v_j(z)) \leq t$,
(c) z^{2t} *divides* $v_j(z)s(z) - r_j(z)$.
(d) $v_j = Kl$ *for some non-zero constant* K.

Remark. From Proposition 3 and statements (b) and (c) above, it follows that not only $v_j = l^\circ = Kl$, but also $r_j = w^\circ = Kw$. We reserve this fact for later use with RS-Codes. Statement (d) implies that l° has the same roots as l.

Proof. (a) From Proposition 2(c) $(z^{2t}, s(z))$ divides $w(z)$, so its degree is $< t$.

(b) From Exercise Sheet 6 we have: $v_j r_{j-1} - v_{j-1} r_j = \pm z^{2t}$ and $\deg(v_{j-1}) < \deg(v_j)$. Thus $\deg(v_{j-1} r_j) < \deg(v_j r_{j-1})$. Hence $\deg(v_j r_{j-1}) = 2t$ and $\deg r_{j-1} \geq t$.

(c) From Euclid's Algorithm $v_j s(z) + u_j z^{2t} = r_j$.

(d) From (c) and Proposition 3 it follows that $v_j(x) = k(x)l(x)$ and $r_j = k(x)w(x)$. If we knew that r_j and v_j were relatively prime the statement would follow directly, but we do not know that yet. So we shall show directly that v_j divides l.

From Proposition 2(c) there exists a polynomial $u(z)$ such that $l(z)s(z) + u(z)z^{2t} = w(z)$, While from statement (c) which we have just proved, $v_j(z)s(z) + u_j(z)z^{2t} = r_j(z)$. Eliminating terms involving $s(z)$ we obtain

$$(u(z)v_j(z) - l(z)u_j(z))z^{2t} = w(z)v_j(z) - r_j(z)l(z) = w(z)k(z)l(z) - k(z)w(z)l(z) = 0.$$

Hence $(u(z)v_j(z) - l(z)u_j(z)) = 0$.

Therefore v_j divides $l(z)u_j(z)$, but from Exercise Sheet 6 $(u_j, v_j) = 1$. Hence by Lemma3 v_j divides l and K is a non-zero constant. □

5. Failure Modes

The algorithm may fail in the presence of more than t errors (it may also decode incorrectly in that case, but that cannot be detected). There are three basic failure modes.

MODE A. $z^t \mid s(z)$ or $\deg(s(z)) < t$. In this case there is no row in Euclid's algorithm satisfying the requirements.

Example. 101101101101101 has $s(z) = 11z^4$.

The failure mode where $\deg(s(z)) < t$ never occurs for BCH Codes, but it can occur with Reed-Solomon Codes.

MODE B. The algorithm terminates but produces a faulty error locator l:

1. 0 is a root of l;
2. l does not have 0 as a root, but it has a multiple root;
3. l does not split into linear factors;

Examples. B1: 101110000000000, $l^\circ = 6z^3 + 10z^2$.
B2 cannot occur.
B3: 110001100011000, $l = 1 + 9z^3$.

MODE C. The algorithm produces valid error locator, but the error evaluator produces an error value $\neq 1$. This type of error would not be detected by the algorithm as it stands, but it would be picked up by an extended algorithm that calculates the error value. However, it can be shown that this type of error cannot occur, unless there is calculation error.

Horner's Scheme

This is known to electrical engineers as Goerzel's algorithm. It was published by Horner in 1819, but was already known to Newton. It is a method for calculating the value of a polynomial $f(x)$ and its derivatives for a particular value $x = u$. We shall only go as far as the first derivative. The method works over any field, and in decoding BCH and RS codes we shall use it in the appropriate finite field. But for the sake of clarity the examples we give here will use ordinary numbers.

We will use as our example $f(x) = 2x^4 + x^3 - x + 1$, and take $u = -1$. Start by writing the coefficients of the polynomial (including zeros) in descending order in a row.

a_4	a_3	a_2	a_1	a_0
2	1	0	-1	1

To the left of the table write u. Copy the highest coefficient a_n as the first entry b_n of the second row. For the later entries put $b_k = a_k + ub_{k+1}$. The last entry b_0 is the value $f(u)$.

	a_4	a_3	a_2	a_1	a_0	
u	$b_4 = a_4$	$b_3 = a_3 + ub_4$	$b_2 = a_2 + ub_3$	$b_1 = a_1 + ub_2$	$b_0 = a_0 + ub_1$	
	2	1	0	-1	1	
-1	2	-1	1	-2	3	$f(-1) = 3.$

To find the first derivative repeat the procedure starting with row b, but ending at column one. Thus $c_n = b_n$, $c_k = b_k + uc_{k+1}$ for $k = n - 1$ down to 1. $c_1 = f'(u)$.

	a_4	a_3	a_2	a_1	a_0	
u	$b_4 = a_4$	$b_3 = a_3 + ub_4$	$b_2 = a_2 + ub_3$	$b_1 = a_1 + ub_2$	$b_0 = a_0 + ub_1$	
	$c_4 = b_4$	$c_3 = b_3 + uc_4$	$c_2 = b_2 + uc_3$	$c_1 = b_1 + uc_2$		
	2	1	0	-1	1	
-1	2	-1	1	-2	3	$f(-1) = 3.$
	2	-3	4	-6		$f'(-1) = -6.$

For those who would like a proof, here is a sketch:

Proof. First notice that the first calculation can be rewritten as $f(x) = a_n x^n + \cdots + a_0 = (x - u)(b_n x^{n-1} + \cdots + b_1) + b_0$. So b_0 is the value $f(u)$, and differentiating and at $x = u$, we get $f'(x) = b_n u^{n-1} + \cdots + b_1$. Hence the same calculation for $b_n, \ldots b_1$ will give $f'(u)$. □

Decoding BCH(4,3)

Syndromes,Error Locators and Evaluators

Suppose transmitted word is c: 1 1 0 1 1 0 0 1 0 1 0 0 0 0 1,
and received word is d: 1 1 0 0 0 0 0 1 0 1 0 0 0 0 1.
Error positions
¿ : 10, 11 (count from right staring at 0).

Error locator $l(z)$: $(1-2^{10}z)(1-2^{11}z) =$
$(1+10z)(1+13z) = 1+7z+15z^2$.
Error evaluator $w(z)$: $10(1+13z)+13(1+10z) = 7$.

Syndromes calculated by Horner's scheme:

	1	1	0	0	0	0	0	1	0	1	0	0	0	0	1	
$2^1=2$	1	3	6	12	1	2	4	9	11	14	5	10	13	3		|7
$2^3=8$	1	9	7	10	6	2	9	6	2	8	15	5	3	1		|9
$2^5=11$	1	10	1	11	10	1	11	11	10	0	0	0	0	0		|1

[Direct calculation is also possible (and in this case more efficient)].

Even syndromes found by squaring: $S_2 = 7^2 = 12$, $S_4 = 12^2 = 6$, $S_6 = 9^2 = 14$.
Syndrome Polynomial $s(z)$: $14z^5 + z^4 + 6z^3 + 9z^2 + 12z + 7$.
$l(z)s(z) = 7 + 2z^6 + 12z^7 \equiv w(z) \pmod{z^6}$.

Error Locator and Evaluator via Euclid's Algorithm

We omit the S column as it is only needed for calculation checks, and the constant terms in the R column as they are always 0.)

Q	R							V		
-	1	0	0	0	0	0	0	0	0	0
-		14	1	6	9	12	7	0	0	1
7 0		7	11	13	15	12	0	0	7	0
0 12			7	14	7	10	15	0	7	12
2 0			4	8	4	11	7	14	1	1
0 10							5	14	5	4.

The table incorporates a method of doing long division internally. We first match the highest coefficient z^6 by a multiple of $s(z)$. The Q entry gives the multiple and shift required (7 shifted one slot). Subtracting gives the third row of the table. We now match the highest coefficient of this row by a multiple of $s(z)$ (again showing shift and number in the Q column: 12 unshifted) and subtract. That gives the fourth row. As this has degree less than $s(z)$ we underline it. It corresponds to the next step in the table.

Next we apply the same process to the division of $s(z)$ by the fourth row. The 5th row is obtained by matching the highest coefficient of $s(z)$ by a multiple of the fourth row and subtracting. The sixth by matching the highest coefficient of the 5th by a multiple of the fourth. This has degree less than 3, so it is the final row of the table.

Correction of received word

Thus $l^\circ = 14z^2 + 5z + 4,$
and $w^\circ = 5.$
To normalize, make lowest coefficient of $l = 1$ by multiplying by 6 .
This gives $l = 15z^2 + 7z + 1,$
and $w(z) = 7.$

Search for roots of l (or equally well l°).
We show only the successful rows.

	15 7 1
9	15 13 0
11	15 10 0.

Roots are $9 = 2^4$ and $11 = 2^5$.
Error positions are $15 - 4 = 11$ and $15 - 5 = 10$.
(Check error values $E_{11} = w(9)/(1 + 9/11) = 7.9/13 = 1,$
$E_{10} = w(11)/(1 + 11/9) = 7.11/3 = 1.$ OK)
Transmitted word: 1 1 0 1 1 0 0 1 0 1 0 0 0 0 1.
Message (if systematic coding used): 1 1 0 1 1.

Chapter 3.4

Reed-Solomon Codes and Burst Error Correction

When you check through the previous two sections you will notice that in dealing with BCH-Codes all calculations take place in GF(16). We only check in the end that the answers lie in $\mathbb{B}$.

Why not drop that condition?

Then we obtain a multiple error correcting code with an alphabet of elements of GF(16). These correspond to blocks of 4 bits or bytes. Now any collection of errors affecting a single byte is treated as a single error. This gives the code good burst error correcting capabilities.

1. DEFINITION. The Reed-Solomon Code RS(k,t), based on a primitive element α of GF(q) (where $q = 2^k$), consists of all words w of length $q-1$ with entries in GF(q), such that $V_{k,t}w = 0$, where $V_{k,t}$ is the Vandermonde type check matrix for BCH(k,t).

Remark. We cannot use $H_{k,t}$ because squaring affects the coefficients of a polynomial with entries in GF(q), so that $f(\alpha) = 0$ no longer implies $f(\alpha^2) = 0$. For BCH we could use the fact that the coefficients there are all 1 or 0 and $1^2 = 1$.

PROPOSITION. *As a code over* GF(q), RS(k,t) *is a cyclic (and hence polynomial) code with generator polynomial* $g(x) = (x-\alpha)(x-\alpha^2)...(x-\alpha^{2t})$. *It has block length* $n = q-1$ *and dimension* $m = n-2t$. *Its minimum distance is* $\geq 2t+1$.

Remark. It can be proved that the minimum distance is exactly $2t+1$ but we shall not do that here.

Proof. With $V_{k,t}$ arranged as for BCH, a word u with entries in GF(q) lies in RS(k,t) iff for the corresponding polynomial $f_u(x)$, $f_u(\alpha) = f_u(\alpha^2) = \cdots = f_u(\alpha^{2t}) = 0$. That is the case iff $g(x) \mid f_u(x)$.

The block length is the number of rows of $V_{k,t} = q - 1$.

The dimension is obtained by subtracting the degree of $g(x)$ from n.

The argument that RS(k,t) can correct t errors is precisely the same as that for BCH(k,t) in the Theorem 3.1.3. We need only observe that the assumption that $e_1, ..., e_{2t}$ should be 0 or 1 is never used in that proof. □

Note that $g(x)$ is not binary. Although all codewords of BCH(k,t) are codewords of RS(k,t), RS has many more codewords which do not consist of $q-1$ zeros and ones. The generator polynomial $g(x)$ cannot distinguish between these different codewords. That for BCH does.

RS(4,3) is treated on a separate example sheet. Its parameters over GF(16) are $n = 15$, $m = 9$, $d \geq 7$.

2. RS(k,t) as a binary code, burst error correction

As a binary code RS has block length $n = k(q-1)$ (eg. 60 for RS(4,3)). Its dimension is $n - 2kt$ (eg. 36 for RS(4,3)) the minimum distance is, however unchanged, as a single bit error in a group defining an element of GF(q) contributes the full unit distance over GF(q). However, errors occurring close together will affect either the same group or neighbouring groups. So for such 'burst errors' we do obtain good decoding capabilities.

DEFINITION. A *symbol* is an element of GF(q). In dealing with RS(k,t) considered as a binary code, we shall identify symbols with the sequences of bits representing them.

A set of binary errors in a word is called a *burst*. The length of the burst is the number of binary positions between the first and last error (inclusive).

Remark. Note that not every position of a burst need contain an error. In practice, it is assumed that many, but not necessarily all, places contain an error.

PROPOSITION. RS(k,t) *can correct a burst of length* $k(t-1)+1$.

EXAMPLE. RS(4,3) can correct a burst of length 9.

Proof. Such a burst cannot affect more than t symbols. □

3. Decoding RS(k,t)

Any decoding algorithm for BCH(k,t) can be adapted to RS(k,t), considered as a code over GF(q). Here is the adaptation of the algorithm we gave for BCH(k,t).

The setup is the same as for BCH, note only that all syndromes must be calculated: it is not true any more that S_2 is S_1^2. The calculations proceed as for BCH but having located the errors, we must also determine their values. In the last chapter we gave a formula for e_i, but there is a computationally advantageous one (due to Forney) which we use here.

ALGORITHM for decoding RS(k,t).

Write the code words and received word as words with entries in GF(q). Then consider them as polynomials as with BCH. We assume that no more than t symbol errors have occurred.

An example of the algorithm is given on a separate page.

Step 1. Calculate $S_i = d(\alpha^i)$ for $i = 1,\dots,2t$.

Put $s(z) = \sum_{i=1}^{2t} S_i z^{i-1}$.

If $s(z) = 0$, no errors.

Step 2. Apply Euclid's Algorithm to $a(z) = z^{2t}$ and $b(z) = s(z)$.

Finish at the first stage where $r_j(z)$ has degree $< t$.

If $r_j = 0$, there are more than t errors: STOP, otherwise put $w^\circ(z) = r_j(z)$.

Step 3. Put $l^\circ(z) = t_j(z)$. Find the roots of $l^\circ(z)$: $\beta_1,\dots,\beta_s$.

Step 4. If $\beta_i = \alpha^{p(i)}$, then the errors occurred at the places $2^k - p(i) - 1$, $i = 1 \dots s$, counting from the right starting at 0.

Step 5. Calculate the error values $e_{p(i)} = w^\circ(\beta_i)/l^{\circ\prime}(\beta_i)$, where $l^{\circ\prime}$ denotes the derivative of l°.

Remark. If we use Horner's scheme to find $l(\alpha^{-i}) = 0$, the same calculation can be continued to obtain the derivative $l'(\alpha^{-i})$.

PROPOSITION. (a) *The algorithm correctly locates the symbols in error.*

(b) *It calculates the error values correctly.*

Proof. (a) The proof is identical to that in Proposition 3.3.4.

(b) From the formula for w, $w(\alpha^{-i}) = e_i\alpha^i \prod_{j\in M\setminus i}(1-\alpha^{j-i})$ (all other terms of the sum are 0). From the formula for l, $l'(\alpha^{-i}) = -\alpha^i \prod_{j\in M\setminus i}(1-\alpha^{j-i})$ (l' is a sum, and again all other terms are 0).

As l° and w° differ from l and w only by multiplication by the same constant it makes no difference if we use them instead. □

FAILURE MODES. Modes A and B still occur. Mode C no longer occurs, as any error value is legal.

4. Burst Error Performance

We compare the performance of RS(4,3) and RS(4,4) with codes obtained by interleaving 4 words of BCH(4,2) and BCH(4,3).

The Codes have word length 60 and dimensions as follows:
RS(4,3): 36; RS(4,4): 28; 4-BCH(4,2): 28; 4-BCH(4,3): 20.

The maximum length of a burst they can correct is calculated by considering for RS at what length the burst will certainly only cover t symbols as above, and for the interleaved codes what length will certainly not contain more than t bits from one pure code word word, which gives kt. The values are as follows:
RS(4,3): 9; RS(4,4): 13; 4-BCH(4,2): 8; 4-BCH(4,3): 12.

CONCLUSION. The RS-Code that corrects burst of length 1 greater than the corresponding interleaved BCH-Code has significantly better rate.

The Code RS(4,3)

RS(4,3) as a linear code

Check matrix $V_{k,t}$:

$$\begin{bmatrix} 12 & 6 & 3 & 13 & 10 & 5 & 14 & 7 & 15 & 11 & 9 & 8 & 4 & 2 & 1 \\ 6 & 13 & 5 & 7 & 11 & 8 & 2 & 12 & 3 & 10 & 14 & 15 & 9 & 4 & 1 \\ 3 & 5 & 15 & 8 & 1 & 3 & 5 & 15 & 8 & 1 & 3 & 5 & 15 & 8 & 1 \\ 13 & 7 & 8 & 12 & 10 & 15 & 4 & 6 & 5 & 11 & 2 & 3 & 14 & 9 & 1 \\ 10 & 11 & 1 & 10 & 11 & 1 & 10 & 11 & 1 & 10 & 11 & 1 & 10 & 11 & 1 \\ 5 & 8 & 3 & 15 & 1 & 5 & 8 & 3 & 15 & 1 & 5 & 8 & 3 & 15 & 1 \end{bmatrix}.$$

Check that (14 3 8 14 3 8 14 3 8 9 9 14 3 13 6) is a codeword.
In binary this is:

1110 0011 1000 1110 0011 1000 1110 0011 1000 1001 1001 1110 0011 1101 0110
Burst of length 11: 1011 1000 111.

This produces the word (14 3 8 14 3 8 5 11 6 9 9 14 3 13 6).
Error locations are in symbols 6,7,8 counting from the right starting from 0.
Error values are 14, 8, 11, again counted from the right.

RS(4,3) as a Polynomial Code

Generator polynomial g(x):

$$(x-2)(x-4)(x-8)(x-9)(x-11)(x-13) = x^6 + 3x^5 + x^4 + 4x^3 + 7x^2 + 13x + 15.$$

Check polynomial h(x):

$$(x-1)(x-3)(x-5)(x-6)(x-7)(x-10)(x-12)(x-14)(x-15).$$

Multiply this out and check that $g(x)h(x) = x^{15} - 1$.

Syndromes, Error Locators and Evaluators

Given the burst error above the error locator is:

$$(1-15z)(1-7z)(1-14z) = 15z^3 + 11z^2 + 6z + 1.$$

The error evaluator is:

$$14 \cdot 15(1-7z)(1-14z) + 8 \cdot 7(1-15z)(1-14z) + 11 \cdot 14(1-15z)(1-7z) = 4z^2.$$

The syndromes are calculated by Horner's Scheme:

	14	3	8	14	3	8	5	11	6	9	9	14	3	13	6
2	14	6	4	6	15	15	2	15	1	11	6	2	7	3	|0
4	14	9	7	11	4	1	1	15	8	2	1	10	0	13	|0
8	14	14	5	13	15	13	9	12	2	0	9	9	4	6	|4
9	14	0	8	9	13	9	11	14	5	15	3	12	11	8	|1
11	14	5	4	9	6	0	5	7	5	5	5	2	12	4	|1
15	14	15	11	3	11	5	4	5	7	15	10	12	8	8	|3

Note that $S_4 \neq S_2^2$ because we are not dealing with a binary polynomial. Syndrome polynomial: $3z^5 + z^4 + z^3 + 4z^2$.

Euclid's Algorithm for Decoding

Q		R							V			
		1	0	0	0	0	0	0	0	0	0	0
			3	1	1	4	0	0	0	0	0	1
8	0		8	8	11	0	0	0			8	0
	15			7	4	14	0	0			8	15
11	0			6	7	4	0	0		14	13	1
	15				9	8	0	0		14	8	2
8	0				11	14	0	0	13	15	1	15
	2					7	0	0	13	10	8	11

Hence $l^\circ = 13z^3 + 10z^2 + 8z + 11 = 11l$ and $w^\circ = 7z^2 = 11w$.

Correcting the error

For the further calculations we can use l and w or l° and w°, but we must not mix them. We shall use l° and w°. A good check to see if you have followed the calculations is to repeat them with l and w.

Horner's scheme to find the roots of l and the value of $l^{\circ\prime}$ there. We omit the unsuccessful runs.

	13	10	8	11
5	13	1	13	|0
	13	10	4	
7	13	2	6	|0
	13	10	2	
14	13	3	3	|0
	13	10	11.	

Error Locations: Roots of l are $5 = 2^9 = 2^{-6}$, $14 = 2^8 = 2^{-7}$,and $7 = 2^7 = 2^{-8}$. Thus error locations are symbols 6,7 and 8.

Error Values:

$$\begin{aligned} e_6 &= 7 \cdot 5 \cdot 5 \cdot /4 &&= 14 = 1110 \\ e_7 &= 7 \cdot 14 \cdot 14/11 &&= 8 = 1000 \\ e_8 &= 7 \cdot 7 \cdot 7/2 &&= 11 = 1011. \end{aligned}$$

Correction

$$\begin{aligned} c &= 14\ 3\ 8\ 14\ 3\ 8\ 5 + 11\ 11 + 8\ 6 + 14\ 9\ 9\ 14\ 3\ 13\ 6 \\ &= 14\ 3\ 8\ 14\ 3\ 8\ \ 14\ \ \ \ 3\ \ \ 8\ \ \ 9\ 9\ 14\ 3\ 13\ 6. \end{aligned}$$

Coding Theory
Exercise Sheet 7

1. Calculate the generator and check polynomials for BCH(4,3) based on the primitive element 4.

2. Show that the polynomial

$$x^{10} + x^8 + x^5 + x^4 + x^2 + x + 1$$

generates a cyclic binary code of block length 15. Find its check polynomial. Find the code word corresponding to the message 11011 using systematic encoding. Determine whether 11001 01000 01110 is a code word.

3. Suppose that BCH(4,3) (based on the primitive element 2) is used to transmit a message and the error pattern of a received word is

10000 10000 10000.

Calculate the syndrome, error locator and evaluator polynomials and check the validity of the fundamental equation.

4. The BCH(4,3) Code based on a primitive element 2 of the field GF(16) is used to transmit a message. One received word is

01100 01110 01010.

Assuming that no more than three errors occurred, find the transmitted codeword.

5. As above the following word is received under BCH(4,3):

00001 10011 10010.

Show that at least 4 errors occured.

2001

Coding Theory
Exercise Sheet 8

1. Encode the message word 9 8 7 6 5 4 3 2 1using RS(4,3) and systematic encoding.

2. Verify that

   111000111000111000111000111000111000100110011110001111010110

   is a code word of RS(4,3). A burst of length 6 affects some of bits 10–15, counting from the left, starting with 1. What is the error locator polynomial of the resulting word? What would it be if a burst affected bits 10–15, counting from the right, starting with 0?

3. Using RS(4,3) correct the received word

   1 2 4 8 5 8 4 2 1 8 14 2 10 12 4.

4. The RS(4,3) Code based on the primitive element 2 of the field GF(16) is used to store messages as sequences of bytes. One stored word is read as

   2 13 3 1 12 7 11 12 7 8 5 4 11 14 7.

   Assuming that the word has been distorted by a single error burst, show that the length of the burst was at least 10 bits.