Lecture 5
Decoding Binary BCH Codes

In this class, we will introduce different methods for decoding BCH codes.

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

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

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

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

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

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

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

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

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

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

Our task should thus consist of finding XY. Note that

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

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

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

Lecture and scribes by Amin Shokrollahi

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

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

Table 5.1: Table of values of ω^i

Suppose that we have received the word

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

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

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

We therefore have

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

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

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

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

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

5.2. The Newton Relations

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

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

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

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

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

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

Theorem 5.2. *[Newton Relations] Let $H(z) = (-1)^t \sigma_t z^t + (-1)^{t-1} \sigma_{t-1} z^{t-1} + \cdots - \sigma_1 z + 1$, and let $H'(z)$ denote the formal derivative of H with respect to z. Further, let $S(z) = S_1 + S_2 z + S_3 z^2 + \cdots$ be a power series with the S_i as coefficients. Then we have*

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

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

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

which is equivalent to our assertion. □

5.3. Recurrence Relations

Let $S = (S_1, S_2, \ldots, S_{2t})$ be a sequence of length $2t$ over a field $\mathbb{F}$. A polynomial $r(z)$ of degree $\leq t$ is called a *recurrence relation* for the sequence if there is a polynomial $\mu(z)$ of degree less than $\deg(r)$ such that $S(z) \equiv \mu(z)/r(z)$, where $S(z) = S_1 + S_2 z + \cdots + S_{2t} z^{2t-1}$. Note that S may not have a recurrence at all. However, if it does, then it has a minimal recurrence, i.e., one of minimal degree. It turns out that every recurrence is a multiple of a minimal one, and hence, the minimal recurrence is unique up to scalar multiplication. Sums of recurrence relations are also recurrence relations,

Theorem 5.3. *Suppose that $S(z) = S_1 + S_2 z + \cdots + S_{2t} z^{2t-1}$ has a recurrence. Then it has a minimal recurrence $\mu(z)$, and any other recurrence is a multiple of $\mu(z)$. Moreover, if $S(z) \equiv v(z)/a(z)$ for co-prime polynomials $v(z)$ and $a(z)$ with $\deg(u) < \deg(a) \leq t$, then $a(z)$ is a minimal recurrence.*

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

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

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

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

Theorem 5.4. *Any recurrence relation of length $2t$ and degree $\leq t$ for the sequence of syndromes is a multiple of the error locator polynomial.*

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

The previous result implies the following.

Corollary 5.5. *If $a(z)$ is a polynomial of degree $\leq t$ and $v(z)$ is a polynomial of degree $< t$, and if*

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

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

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

5.4. Decoding BCH-Codes using Matrix Equations

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

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

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

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

1. Upon reception of y, check whether it is in the code. If so, then STOP.
2. Compute the syndromes $S_i = y(\omega^i)$, $i = 1, \ldots, 2t$.
3. Find a nonzero solution a of the system (5.4). Declare ERROR if it does not exist.
4. Form the polynomial $a(z)$ and calculate its roots $\omega^{-i_1}, \ldots, \omega^{-i_s}$ in $\{1, \omega, \ldots, \omega^{n-1}\}$.
5. Declare positions $i_1, \ldots, i_s$ as erasures. Use any erasure decoding algorithm for the BCH-code to recover these positions and obtain a codeword c.

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

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

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

Example 5.6. We use the same example as in the first section. Then, recalling (5.2), the system (5.4) becomes

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

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

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

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

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

As soon as the degree of $v_i(z)$ becomes smaller than that of $a_{i-1}(z)$, we have satisfied the assumptions of Corollary 5.5, and we have identified a super-set of the error positions. Erasure decoding, as described in the previous section, will then reveal the correct codeword.

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

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

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

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

Then we have the recurrence relations

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

The algorithm would thus proceed as follows:

1. Initialize a_i, b_i, $i = -1, 0$, as in (5.5).
2. Initialize $v_0 = z^{2t}$ and $v_1 = S(z)$, $i = 0$.
3. While $\deg(v_i) \geq \deg(a_{i-1})$ do
   (a) (Division step) Compute $q_i(z)$ and v_{i+2} from $v_i(z) = q_i(z)v_{i+1}(z) + v_{i+2}$.
   (b) (Update) Set $a_{i+1} = -q_i a_i + a_{i-1}$, and $b_{i+1} = -q_i b_i + b_{i-1}$.
   (c) Set $i = i + 1$.
4. The polynomial $a_{i-1}(z)$ is now a multiple of the error locator polynomial. Calculate its roots $\omega^{-i_1}, \ldots, \omega^{-i_s}$ in $\{1, \omega, \ldots, \omega^{n-1}\}$.
5. Declare positions $i_1, \ldots, i_s$ as erasures. Use any erasure decoding algorithm for the BCH-code to recover these positions and obtain a codeword c.

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

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

Example 5.7. We continue with the running example of Section 1. Setting $v_0(z) = z^4$ and $v_1(z) = S(z) = \omega^9 z^3 + \omega^7 z^2 + \omega^{12} z + \omega^6$, we obtain

$$\begin{array}{rcccccc}
v_0(z) & = & v_1(z) & \cdot & (\omega^6 z + \omega^4) & + & (\omega^5 z^2 + \omega^{11} z + \omega^{10}) \\
 & = & v_1(z) & \cdot & q_0(z) & + & v_2(z) \\
 & & & & & & \\
b_1(z) & = & 1 & & & & \\
a_1(z) & = & q_0(z) & & & & \\
 & & & & & & \\
v_1(z) & = & v_2(z) & \cdot & (\omega^4 z + \omega^7) & + & \omega^3 \\
 & = & v_2(z) & \cdot & q_1(z) & + & v_3(z) \\
 & & & & & & \\
b_2(z) & = & q_1(z) & & & & \\
a_2(z) & = & q_1(z) & \cdot & b_1(z) & + & b_0(z) \\
 & = & (\omega^6 z + \omega^4) & \cdot & (\omega^4 z + \omega^7) & + & 1 \\
 & = & \omega^{10} z^2 + \omega^3 z + \omega^{12}. & & & &
\end{array}$$

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