Classical Channel Coding, II
Hamming Codes, Cyclic Codes, BCH Codes, RS Codes

John MacLaren Walsh, Ph.D.

ECET 602, Winter Quarter, 2013

1 References

- *Error Control Coding*, 2nd Ed. Shu Lin and Daniel J. Costello, Jr. Pearson Prentice Hall, 2004.
- *Error Control Systems for Digital Communication and Storage*, Prentice Hall, S. B. Wicker, 1995.

2 Example of Syndrome Decoder – Hamming Codes

Recall that a (n, k) Hamming code is a perfect code with a $m \times 2^m - 1$ parity check matrix whose columns are all $2^m - 1$ of the non-zero binary words of length m. This yields an especially simple syndrome decoder look up table for

$$\arg \min_{\mathbf{e}|\mathbf{e}\mathbf{H}^T = \mathbf{r}\mathbf{H}^T} \mathrm{wt}(\mathbf{e}) \tag{1}$$

In particular, one selects the error vector to be the jth column of the identity matrix, where j is the column of the parity check matrix which is equal to the computed syndome $\mathbf{s} = \mathbf{r}\mathbf{H}^T$.

3 Cyclic Codes

Cyclic codes are a subclass of linear block codes over $GF(q)$ which have the property that every *cyclic shift* of a codeword is itself also a codeword. The ith cyclic shift of a vector $\mathbf{v} = (v_0, v_1, \ldots, v_{n-1})$ is the vector $\mathbf{v}^{(i)} = (v_{n-i}, v_{n-i+1}, \ldots, v_{n-1}, v_0, v_1, \ldots, v_{n-i-1})$. If we think of the elements of the vector as coefficients in the polynomial $v(x) = v_0 + v_1 x + \cdots + v_{n-1}x^{n-1}$ we observe that

$$\begin{aligned} x^i v(x) &= v_0 x^i + v_1 x^{i+1} + \cdots + v_{n-i-1}x^{n-1} + v_{n-i}x^n + \cdots + v_{n+i-1}x^{n+i-1} \\ &= v_{n-i} + v_{n-i+1}x + \cdots + v_{n-1}x^{i-1} + v_0 x^i + \cdots + v_{n-i-1}x^{n-1} \\ &\quad + v_{n-i}(x^n - 1) + v_{n-i+1}x(x^n - 1) + \cdots + v_{n+i-1}x^{i-1}(x^n - 1) \end{aligned}$$

Hence,

$$x^i v(x) \mathrm{mod}(x^n - 1) = v_{n-i} + v_{n-i+1}x + \cdots + v_{n-1}x^{i-1} + v_0 x^i + v_1 x^{i+1} + \cdots + v_{n-i-1}x^{n-1} = v^{(i)}(x) \tag{2}$$

That is, the ith cyclic shift can alternatively be thought of as the operation $v^{(i)}(x) = x^i v(x) \mathrm{mod}(x^n - 1)$.

Let $g(x) = g_0 + g_1 x + g_2 x^2 + \cdots + g_{n-k-1}x^{n-k-1} + x^{n-k}$ be the minimum degree monic (and hence non-zero) codeword of a cyclic code, called the *generator polynomial* for the code. Due to the fact that the code is cyclic and $g(x)$ is degree $n-k$, $x^i g(x) \mod x^n - 1 = x^i g(x)$ is a codeword for $i \in \{0, 1, \ldots, k-1\}$. This, together with the linearity of the code, then implies that *any* product $w(x)g(x)$, where $w(x) = w_0 + w_1 x + \cdots + w_{k-1}x^{k-1} \in GF(q)[x]$ is any degree $k-1$ or less polynomial, must be a codeword of the code.

The generator polynomial $g(x)$ is aptly named because *every* codeword in the cyclic code can be represented as $c(x) = w(x)g(x)$, where $w(x) = w_0 + w_1 x + \cdots + w_{k-1}x^{k-1}$ is some polynomial in $GF(q)[x]$. (Indeed, if there were a codeword $c(x)$ that weren't a multiple of $g(x)$, then we could write $c(x) = q(x)g(x) + r(x)$ with $r(x) = c(x) \mod g(x)$ a polynomial of degree less than the degree of $g(x)$. But then by the linearity of the code and the fact that any $q(x)g(x)$ is a codeword, $r(x) = c(x) - q(x)g(x)$ must be

a codeword. This is a contradiction, since now $r(x)$ is a codeword of degree less than $g(x)$, violating its minimum degree non-zero property.)

Summarizing, $c(x)$ is a codeword of the cyclic code if and only if $c(x) = w(x)g(x)$ for some $w(x) = w_0 + w_1 x + \cdots + w_{k-1}x^{k-1} \in GF(q)[x]$

Additionally, the generator polynomial must divide (i.e. with remainder zero) $x^n - 1$. This is because $g^{(k)}(x) = x^k g(x) \mod x^n - 1$ is a cyclic shift of $g(x)$ and hence must be a codeword. But since it is a codeword, it must be expressible as $g^{(k)}(x) = a(x)g(x)$. Observing then that $x^k g(x) = x^n - 1 + g^{(k)}(x)$, we see that $x^n - 1 = x^k g(x) - g^{(k)}(x) = (x^k - a(x))g(x)$, so $g(x)$ divides $x^n - 1$ as desired.

Given any degree $n - k$ monic polynomial $g(x)$ that divides $x^n - 1$, one can generate the associated cyclic code. (Not all such $g(x)$ give good codes, and we will investigate constructions of them that provide good codes.)

3.1 Systematic Encoding for Cyclic Codes

If a systematic form of the code is alternatively desired, one can polynomial long divide $x^{n-k}u(x)$ by $g(x)$ to get

$$x^{n-k}u(x) = q(x)g(x) + d(x), \quad d(x) = d_0 + d_1 x + \cdots + d_{n-k-1}x^{n-k-1} = u(x) \mod g(x) \tag{3}$$

Rearranging this equation, we observe that $x^{n-k}u(x) + d(x) = q(x)g(x)$, so that $x^{n-k}u(x) + d(x)$ is a codeword, which additionally takes the form (due to the involved polynomial degrees)

$$x^{n-k}u(x) - d(x) = -d_0 - d_1 x - \cdots - d_{n-k-1}x^{n-k-1} + u_0 x^{n-k} + u_1 x^{n-k+1} + \cdots + u_{k-1}x^{n-1} \tag{4}$$

which, when written in the vector form $(-d_0, -d_1, \ldots, -d_{n-k-1}, u_0, u_1, \ldots, u_{k-1})$, reveals that it this is a systematic representation of the codeword.

3.2 Error Detection for Cyclic Codes

Forming the received polynomial $r(x) = r_0 + r_1 x + \cdots + r_{n-1}x^{n-1}$, the syndrome polynomial for the cyclic code is the remainder upon division by the generator polynomial.

$$s(x) = r(x) \mod g(x) \tag{5}$$

Note that as $g(x)$ is of degree $n-k$ the remainder $s(x)$ will be of degree at most $n-k-1$, as the syndrome should be. Also, it is clear to see that the syndrome $s(x)$ is zero if and only if $r(x)$ is a multiple of $g(x)$, and hence a codeword. Hence, an error is detected if the syndrome $s(x)$ is non-zero.

Alternatively, one can check for errors using the parity check polynomial $h(x)$, which is determined by exploiting the fact that the generator polynomial of a cyclic code $g(x)$ must divide $x^n - 1$, so that an $h(x)$ must exist such that $x^n - 1 = h(x)g(x)$. Because any codeword can be expressed as $c(x) = w(x)g(x)$ for some $w(x)$, if we multiply a codeword by $h(x)$ we get

$$h(x)c(x) = w(x)\underbrace{h(x)g(x)}_{x^n-1} \quad \Longrightarrow \quad h(x)c(x) \bmod (x^n - 1) = 0 \tag{6}$$

Hence, for the parity check polynomial (defined through $h(x)g(x) = x^n - 1$) we have that $r(x)$ is a codeword if and only if

$$r(x)h(x) \mod (x^n - 1) = 0 \tag{7}$$

this provides an alternate way of computing a syndrome.

We discussed that both forms of syndrome computation, as well as the systematic and non-systematic encoders of cyclic codes admit an "easy" hardware implementation in terms of shift registers and multiply adders. Additionally, we noted that cyclic codes have additional structure that allow their syndrome lookup tables for error correction to be simplified beyond (i.e. smaller than) those required for a generic linear block code.

3.3 Cyclic Redundancy Check Codes (CRC)

One of the most commonly encountered variants of cyclic codes are cyclic redundancy check (CRC) codes which are widely used for error detection in computer communications systems. CRC codes are obtained by *shortening* cyclic codes, which means that a collection of systematic positions (in this case, those of highest order) set equal to zero, and then simply not transmitted. The receiver, knowing they were set to zero, can input zeros at these locations and decode as usual for the original code. This creates a $(n-j, k-j)$ code of a lower rate than the original (n, k) cyclic code, and the resulting shortened code has error detecting power at least as good as the original cyclic code. Once the zeros have been re-inserted at the receiver at the proper systematic positions, syndrome computation (and hence error detection, since this amounts to verifying that the syndrome is the zero vector) can take place with the normal cyclic code's hardware.

4 BCH Codes

Thus far our discussion of cyclic codes has not dealt with their relative performance, or how to select the generator polynomial of the cyclic code so that it will give good error detection or correction performance. Primitive BCH codes, named after their discoverers Bose, Ray-Chaudhuri, and Hocquenghem are a class of cyclic codes designed to have the ability to detect any error pattern of weight $2t$ or less and correct any error pattern of t or fewer errors. BCH codes select the generator polynomial $g(x) \in GF(q)[x]$ (i.e. with coefficients in $GF(q)$) to have $2t$ successive powers of a primitive element of $GF(q^m)$ among its roots $\{\alpha, \alpha^2, \ldots, \alpha^{2t}\}$. In particular, the generator polynomial $g(x)$ is selected as the least common polynomial multiple of the minimal polynomials $\{\phi_i(x)\}$ for these roots α^i, $i \in \{1, \ldots, 2t\}$.

$$g(x) = \text{LCM}(\phi_1(x), \ldots, \phi_{2t}(x)) \tag{8}$$

Here, the minimal polynomial in $GF(q)[x]$ of an element β in $GF(q^m)$ is the monic polynomial of smallest degree in $GF(q)[x]$ which has β among its roots. The minimal polynomial for β has as its roots the *conjugates* of β, which are the powers $\{\beta, \beta^q, \ldots, \beta^{q^j}, \ldots\}$. Because the field is finite, at some point $\beta^{q^e} = \beta$ (note we must have $e \leq m$ with equality when β is a primitive element of $GF(q^m)$), and hence we can represent the minimal polynomial of β as

$$\phi(x) = \prod_{j=0}^{e-1} (x - \beta^{q^j}) \tag{9}$$

which (despite the fact that the roots are in $GF(q^m)$) will have coefficients in $GF(q)$ and be of degree less than or equal to m. *To allow recognition of elements in $GF(q)$, the arithmetic in $GF(q^m)$ should be constructed using a degree m primitive polynomial for $GF(q^m)$ that is in $GF(q)[x]$, so that elements in $GF(q^m)$ are represented in terms of degree $m-1$ or less polynomials over $GF(q)$.*

4.1 Error Detection and Syndrome Computation for BCH codes

Because a primitive element of $GF(q^m)$ is among the roots of $g(x)$, the first n for which $g(x)$ divides $x^n - 1$ is $n = q^m - 1$. Hence a primitive BCH code has block length $n = q^m - 1$. Since each of the $2t$ minimal polynomials have degree less than or equal to m, the degree of $g(x)$, $n - k \leq 2mt$.

Noting that $g(\alpha^i) = 0$ (as it was determined by requiring these to be roots), and because the code is cyclic, any codeword polynomial can be represented as $c(x) = w(x)g(x)$ for some $w(x)$, the ith position in the syndrome for a BCH code can be calculated as

$$s_i = r(\alpha^i) = r_0 + r_1\alpha^i + r_2(\alpha^i)^2 + \cdots + r_{n-1}(\alpha^i)^{n-1}, \quad i \in \{1, \ldots, 2t\} \tag{10}$$

Hence the parity check matrix takes the form

$$\mathbf{H} = \begin{bmatrix} 1 & \alpha & \alpha^2 & \cdots & \alpha^{n-1} \\ 1 & \alpha^2 & (\alpha^2)^2 & \cdots & (\alpha^2)^{n-1} \\ \vdots & & & & \vdots \\ 1 & \alpha^{2t} & (\alpha^{2t})^2 & \cdots & (\alpha^{2t})^{n-1} \end{bmatrix} \tag{11}$$

The minimum distance $d_{min} \geq 2t+1$ because no $2t$ or fewer columns of $\mathbf{H}$ are linearly dependent, which is proven by showing that any $2t$ columns of $\mathbf{H}$ have a determinant which is a non-zero multiple of a Vandermonde matrix's determinant, which must be non-zero (pp. 203). This fact is known as the BCH bound.

4.2 RS Codes

Reed-Solomon codes are a class of primitive q-ary (non-binary) BCH codes with the following characteristics. They are primitive BCH codes, and the element α is a primitive element of $GF(q)$ (that is they take $m = 1$ in the construction above). This gives the generator polynomial

$$g(x) = (x-\alpha)(x-\alpha^2)(x-\alpha^3)\cdots(x-\alpha^{2t}) \tag{12}$$

They have a block length $n = q-1$, $n-k = 2t$ parity check symbols (i.e. dimension $k = q-1-2t$) and minimum distance $d_{min} = 2t+1$. Because $d_{min} = n-k+1$, these codes are said to be *maximum distance separable*.

4.3 Decoding BCH & RS Codes – Berlekamp's Algorithm

The received word polynomial $r(x) = c(x) + e(x)$ is the sum of the codeword polynomial $c(x)$ and the error polynomial $e(x)$. If the errors are located at positions $j_1, \ldots, j_v$ and have error values $e_{j_1}, \ldots, e_{j_v}$, then the error polynomial can be written as

$$e(x) = e_{j_1}x^{j_1} + e_{j_2}x^{j_2} + \cdots + e_{j_v}x^{j_v} \tag{13}$$

Recall that by construction $c(x) = w(x)g(x)$ so $c(\alpha^i) = 0$ for each $i \in \{1, \ldots, 2t\}$. Then we see that the syndrome is equal to

$$s_i = e(\alpha^i) = e_{j_1}(\alpha^i)^{j_1} + e_{j_2}(\alpha^i)^{j_2} + \cdots + e_{j_v}(\alpha^i)^{j_v} \tag{14}$$

Swapping exponents, we observe that finding the error locations can be equivalently posed as finding the *error location numbers* $\beta_k := \alpha^{j_k}$ and error values $\delta_k = e_{j_k}$, $k \in \{1, \ldots, v\}$, which are related to the syndrome values through

$$\begin{aligned}
S_1 &= \delta_1\beta_1 + \delta_2\beta_2 + \cdots + \delta_v\beta_v && (15)\\
S_2 &= \delta_1\beta_1^2 + \delta_2\beta_2^2 + \cdots + \delta_v\beta_v^2 && (16)\\
&\vdots && (17)\\
S_{2t} &= \delta_1\beta_1^{2t} + \delta_2\beta_2^{2t} + \cdots + \delta_v\beta_v^{2t} && (18)
\end{aligned}$$

In order to find the error location numbers, we determine the *error locator polynomial* $\sigma(x)$ which has roots at $\{\beta_1^{-1}, \ldots, \beta_v^{-1}\}$:

$$\sigma(x) = \prod_{k=1}^{v}(1-\beta_k x) \tag{19}$$

Representing this polynomial in its expanded form $\sigma(x) = \sigma_0 + \sigma_1 x + \cdots + \sigma_v x^v$, it turns out that the coefficients of this polynomial are related to the syndrome values through the *generalized Newton's identities*

$$\begin{aligned}
S_{v+1} + \sigma_1 S_v + \sigma_2 S_{v-1} + \cdots + \sigma_v S_1 &= 0 && (20)\\
S_{v+2} + \sigma_1 S_{v+1} + \sigma_2 S_v + \cdots + \sigma_v S_2 &= 0 && (21)\\
&\vdots && (22)\\
S_{2t} + \sigma_1 S_{2t-1} + \sigma_2 S_{2t-2} + \cdots + \sigma_v S_{2t-v} &= 0 && (23)
\end{aligned}$$

These identities can be derived by defining the infinite sequence

$$S_j = \sum_{k=1}^{v}\delta_k(\beta_k)^j \tag{24}$$

for all $j > 2t$ (note that for $j \in \{1, \ldots, 2t\}$ the S_js are the syndrome values), allowing us to define a power series syndrome polynomial

$$\tilde{S}(x) = \sum_{j=1}^{\infty} S_j x^{j-1} = \sum_{j=1}^{\infty} \left(\sum_{k=1}^{v} \delta_k \beta_k^j \right) x^{j-1} = \sum_{k=1}^{v} \delta_k \beta_k \left(\sum_{j=1}^{\infty} (\beta_k x)^{j-1} \right) = \sum_{k=1}^{v} \frac{\delta_k \beta_k}{1 - \beta_k x} \tag{25}$$

This shows that if we multiply $\tilde{S}(x)$ by $\sigma(x) = \prod_k (1 - \beta_k x)$, we should get

$$\sigma(x)\tilde{S}(x) = \sum_{k=1} \delta_k \beta_k \prod_{\substack{j=1 \\ j \neq k}}^{v} (1 - \beta_j x) \tag{26}$$

which is a polynomial of degree $v - 1$ or less! Since this product must have degree $v - 1$ or less, the coefficients on x^k for $k \geq v$ must be zero, and the left hand side of the generalized Newton's identities (20-23) are these coefficients for $k \in \{v, \ldots, 2t\}$. This proves the generalized Newton's identities.

Berlekamp's algorithm finds the error locator polynomial $\sigma(x)$ by progressively solving a larger and larger subsystem of the generalized Newton's identities. At the μth step in the progression, the polynomial $\sigma^{(\mu)} = \sigma_0 + \sigma_1 x + \cdots + \sigma_{\ell_\mu} x^{\ell_\mu}$ of minimum degree ℓ_μ is determined that satisfies

$$\begin{bmatrix} S_{\ell_\mu+1} & S_{\ell_\mu} & \cdots & S_1 \\ S_{\ell_\mu+2} & S_{\ell_\mu+1} & \cdots & S_2 \\ & & \vdots & \\ S_\mu & S_{\mu-1} & \cdots & S_{\mu-\ell_\mu} \end{bmatrix} \begin{bmatrix} 1 \\ \sigma_1^{(\mu)} \\ \vdots \\ \sigma_{\ell_\mu}^{(\mu)} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix} \tag{27}$$

Next, the discrepancy d_μ is calculated

$$d_\mu = S_{\mu+1} + \sigma_1^{(\mu)} S_\mu + \cdots + \sigma_{\ell_\mu}^{(\mu)} S_{\mu+1-\ell_\mu} \tag{28}$$

If $d_\mu = 0$ then we can take $\sigma^{(\mu+1)}(x) = \sigma^{(\mu)}(x)$ (as the error locator polynomial from the previous iteration also solves this equation). Otherwise, we set

$$\sigma^{(\mu+1)}(x) = \sigma^{(\mu)}(x) - d_\mu d_\rho^{-1} X^{\mu-\rho} \sigma^{(\rho)}(x) \tag{29}$$

Where $\rho < \mu$ is selected so that $d_\rho \neq 0$ and $\rho - \ell_\rho$ is largest. The procedure is initialized at $\mu = 0$, with $\sigma^{(-1)}(x) = 1$ and $d_{-1} = 1$.

Once the error locator polynomial is found, one can find the error locations find finding those js such that $\sigma(\alpha^{-j}) = 0$ (this can be done by a search through j).

From the error locator polynomial $\sigma(x)$, the error locations $j_1, \ldots, j_v$, and the polynomial

$$Z_0(x) = \sigma(x)S(x) = S_1 + (S_2 + \sigma_1 S_1)X + (S_3 + \sigma_1 S_2 + \sigma_2 S_1)X^2 + \cdots + (S_{v+1} + \sigma_1 S_v + \cdots \sigma_{v-1} S_1)x^{v-1} \tag{30}$$

we can calculate the error values, with e.g. an equation derived by Forney, as

$$e_{j_k} = \frac{-Z_0(\alpha^{-j_k})}{\sigma'(\alpha^{-j_k})} \tag{31}$$

where $\sigma'(x) = \frac{d}{dx}\sigma(x)$.

This can be seen to be true by observing via (26) that

$$Z_0(\beta_l^{-1}) = \sum_{k=1} \delta_k \beta_k \prod_{\substack{j=1 \\ j \neq k}}^{v} (1 - \beta_j \beta_l^{-1}) = \delta_l \beta_l \prod_{\substack{j=1 \\ j \neq l}}^{v} (1 - \beta_j \beta_l^{-1}) \tag{32}$$

While taking the derivative of the error locator shows

$$\sigma'(\beta_l^{-1}) = -\sum_{k=1}^{v} \beta_k \prod_{\substack{j=1 \\ j \neq k}}^{v} (1 - \beta_j \beta_l^{-1}) = \beta_l \prod_{\substack{j=1 \\ j \neq l}}^{v} (1 - \beta_j \beta_l^{-1}) \tag{33}$$