Cyclic Block Codes

Cyclic block codes are an important subclass of linear block codes. A (n,k) linear code that is cyclic has the following property. If the n-tuple $c=[c_0,c_1,c_2,\ldots,c_{n-1}]$ is a code word then any cyclic shift (right or left) of c is also a code word.

The components of c can be treated as the coefficients of a polynomial $c(X)$:

$$c(X)=c_0+c_1X+c_2X^2+\ldots+c_{n-1}X^{n-1}$$

The polynomials representing cyclic code words have an interesting algebraic property. If $c(X)$ is the polynomial representing the code word c and $c^{(i)}(X)$ is the polynomial representing the cyclic right-shift of the code word c by i places then $c^{(i)}(X)$ is the remainder when $X^i c(X)$ is divided by X^n+1:

$$X^i c(X)=q(X)(X¿¿n+1)+c^{(i)}(X)¿$$

Example:

Suppose $c=[1101]$ is a cyclic code word. The cyclic right-shift $c^{(3)}=[1011]$ is also a code word and can be found by dividing $X^3c(X)$ by X^4+1:

$$X^3c(X)=X^3+X^4+X^6$$

$$\begin{array}{r|l} & X^2+1=q(X) \\ \hline X^4+1 & X^6+X^4+X^3 \\ & X^6 \qquad\qquad +X^2 \\ \hline & X^4+X^3+X^2 \\ & X^4 \qquad\qquad +1 \\ \hline & X^3+X^2+1=c^{(3)}(X) \end{array}$$

For every (n,k) cyclic block code there is one and only one code polynomial of degree $n-k$:

$$g(X)=¿1+g_1X+g_2X^2+\ldots+g_{n-k-1}X^{n-k-1}+X^{n-k}$$

Every code polynomial in this (n,k) cyclic code can be expressed in the following form:

$$c(X)=m(X)g(X)$$

$$¿(m¿¿0+m_1X+m_2X^2+\ldots+m_{k-1}X^{k-1})g(X)¿$$

If the coefficients of $m(X)$ are the k message symbols to be encoded then $c(X)$ is the corresponding code polynomial. Encoding can then be achieved by multiplying the message polynomial $m(X)$ by $g(X)$. The polynomial $g(X)$ is called the *generator polynomial*.

Cyclic codes generated in this manner are *not* systematic.

A degree $n-k$ polynomial $g(X)$ is the generator polynomial of a (n,k) cyclic code if and only if $g(X)$ evenly divides X^n+1.

Example:

Suppose we have the (7,4) cyclic block code with generator polynomial $g(X)=1+X+X^3$. Determine the code word for the message word $m=[1011]$.

Form the message polynomial $m(X)$:

$$m(X)=1+X^2+X^3$$

Form the code word polynomial $c(X)$ by multiplying $m(X)$ by the generator polynomial $g(X)$:

$$c(X)=m(X)g(X)$$

$$¿(1+X^2+X^3)(1+X+X^3)$$

$$¿1+X+X^3+X^2+X^3+X^5+X^3+X^4+X^6$$

$$¿1+X+X^2+X^3+X^4+X^5+X^6$$

From $c(X)$ we construct the code word $c=[1111111]$. Since the code is not systematic it is not obvious which symbols are message symbols or parity symbols.

A systematic (n,k) cyclic block code can be generated using the following procedure:

1. Start with a message word polynomial:

$$m(X)=¿\ m_0+m_1X+m_2X^2+\ldots+m_{k-1}X^{k-1}$$

2. Multiply $m(X)$ by $X^p=X^{n-k}$ to get the right-shifted message polynomial:

$$X^{n-k}m(X)=¿\ m_0X^{n-k}+m_1X^{n-k+1}+m_2X^{n-k+2}+\ldots+m_{n-1}X^{n-1}$$

Using Euclid's form for division:

$$X^{n-k}m(X)=q(X)g(X)+p(X)$$

The generator polynomial $g(X)$ has degree $p=n-k$ so the remainder $p(X)$ has a maximum degree of $p-1=n-k-1$:

$$p(X)=¿\ p_0+p_1X+p_2X^2+\ldots+p_{n-k-1}X^{n-k-1}$$

3. Add $p(X)$ to both sides of the equation for the right-shifted message polynomial to create the code word polynomial $c(X)$:

$$c(X)=p(X)+X^{n-k}m(X)=p(X)+q(X)g(X)+p(X)$$

$$¿q(X)g(X)$$

Systematic code words generated by this procedure are evenly divisible by $g(X)$. This useful property is fundamental to the decoding process of systematic cyclic block codes.

Example:

Suppose again the (7,4) cyclic block code with generator polynomial $g(X)=1+X+X^3$. Determine the systematic code word for the message word $m=[1011]$.

1. Form $m(X)=1+X^2+X^3$
2. Multiply $m(X)$ by $X^{7-4}=X^3$. Using Euclid's form for division:

$$X^3m(X)=X^3+X^5+X^6$$

$$¿q(X)g(X)+p(X)$$

$$¿(1+X+X^2+X^3)(1+X+X^3)+1$$

3. Add $p(X)=1$ to both sides of the equation above to create $c(X)$:

$$c(X)=p(X)+X^3m(X)$$

$$¿1+X^3+X^5+X^6$$

From $c(X)$ we construct the code word $c=[1001011]$. The first three symbols are the parity symbols followed by the four message symbols.

Polynomial multiplication is equivalent to the discrete convolution of the polynomial coefficients. (Note the MATLAB functions for polynomial multiplication are called `conv` and `gfconv`.) The polynomial multiplication process for encoding a cyclic code can therefore be represented by the following matrix equation:

$$c = mG$$

where G is the *cyclic code generator matrix*:

$$G = \begin{bmatrix} 1 & g_1 & g_2 & \cdots & g_{n-k-1} & 1 & 0 & 0 & \cdots & 0 \\ 0 & 1 & g_1 & g_2 & \cdots & g_{n-k-1} & 1 & 0 & \cdots & 0 \\ 0 & 0 & 1 & g_1 & g_2 & \cdots & g_{n-k-1} & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & 1 & g_1 & g_2 & \cdots & g_{n-k-1} & 1 \end{bmatrix}$$

In general this matrix is not in systematic form. Subsequent operations involving linear combinations of various rows can form a systematic matrix G'.

BCH Codes

The Bose, Chaudhuri and Hocquenghem (BCH) codes form a large class of powerful error-correcting cyclic block codes. They are a remarkable generalization of the Hamming codes for multiple-error correction.

Binary BCH codes were discovered by Alexis Hocquenghem in 1959 and independently by Raj Bose and D. K. Ray-Chaudhuri in 1960. The cyclic structure of these codes was proved by W. Wesley Peterson in 1960. They were generalized to non-binary codes with $q = p^m$ symbols where p is prime by Daniel Gorenstein and Neal Zierler in 1961.

Among the non-binary q-ary codes the most important subclass are the Reed-Solomon codes. They were discovered by Irving Reed and Gustave Solomon in 1960 at the MIT Lincoln Laboratory independently of the work by Bose, Chaudhuri and Hocquenghem. Development of various Reed-Solomon coding and decoding algorithms took place over the next forty years.

Reed-Solomon codes have many applications:

- Consumer
  - CD, DVD, Blu-ray
  - QR code (two-dimensional barcode)
- Data Transmission
  - DSL
  - WiMAX (IEEE 802.16)
- Digital Television Broadcast
  - DVB
  - ATSC
- Digital Storage
  - RAID 6
- Satellite
  - NASA DSN

Binary Primitive BCH Codes

For any positive integers $m \geq 3$ and $t < 2^{m-1}$ there exists a binary BCH code with the following parameters:

Block Length:	$n = 2^m - 1$
Parity-Check Digits:	$n - k \leq mt$
Minimum Distance:	$d_{min} \geq 2t + 1$

This code can correct any combination of t or fewer errors in a block of n digits. We call this code a t-error-correcting BCH code.

The generator polynomial for this code is specified in terms of its roots from $GF(2^m)$. The generator polynomial $g(X)$ of the t-error-correcting BCH code of length $n = 2^m - 1$ is the polynomial of *lowest degree* over $GF(2)$ that has as roots:

$\alpha, \alpha^2, \alpha^3, \cdots, \alpha^{2t}$

In other words:

$g(\alpha^i)=0 \quad$ for $1 \le i \le 2t$

Minimal polynomials $\phi_i(X)$ over $GF(2^m)$ are the polynomials of lowest degree that have α^i and its conjugates as roots. The generator polynomial $g(X)$ is then the product of all the distinct minimal polynomials (each minimal polynomial appears only once as a factor) that have the roots $\alpha, \alpha^2, \alpha^3, \cdots, \alpha^{2t}$:

$$g(X)=\text{LCM}\left[\phi_1(X), \phi_2(X), \cdots, \phi_{2t}(X)\right]$$

The degree of $g(X)$ which is equal to the number of parity-check digits is at most mt.

There is no simple formula for enumerating the possible $n-k \le mt$. Tables exist that list all the possible binary BCH codes and their generator polynomials. Such codes are usually called *primitive* or *narrow-sense* BCH codes.

Example:

Determine the generator polynomial $g(X)$ for the triple-error correcting BCH(15,5) code.

Here $n=15=2^4-1$ so $m=4$ and we are working over $GF(16)$. Also $k=5$ and $t=3$.
The number of parity digits is $n-k=10$ which is in fact less than or equal to $mt=12$.

The generator polynomial $g(X)$ has $\alpha, \alpha^2, \alpha^3, \alpha^4, \alpha^5$ and α^6 as roots and is given by:

$$g(X)=\text{LCM}\left[\phi_1(X), \phi_2(X), \cdots, \phi_{2t}(X)\right]$$

Note that α, α^2 and α^4 are conjugates as are α^3 and α^6. This means $\phi_1(X)=\phi_2(X)=\phi_4(X)$ and $\phi_3(X)=\phi_6(X)$ so $\phi_2(X), \phi_4(X)$ and $\phi_6(X)$ will not appear in the LCM that defines $g(X)$:

$$g(X)=\phi_1(X)\phi_3(X)\phi_5(X)$$

$$¿\phi_1(X)\phi_3(X)\phi_5(X)$$

$$¿\left(1+X+X^4\right)\left(1+X+X^2+X^3+X^4\right)\left(1+X+X^2\right)$$

$$¿1+X+X^2+X^4+X^5+X^8+X^{10}$$

We can verify this result using MATLAB. The function `gfminpol(i, m)` returns an array of the coefficients of the minimal polynomial $\phi_i(X)$ over $GF(2^m)$ with root α^i in ascending powers of X.

```
>> Phi1 = gfminpol(1, 4)

Phi1 =

     1     1     0     0     1

>> Phi3 = gfminpol(3, 4)

Phi3 =

     1     1     1     1     1

>> Phi5 = gfminpol(5, 4)

Phi5 =

     1     1     1

>> G = gfconv(gfconv(Phi1, Phi3), Phi5)

G =

     1     1     1     0     1     1     0     0     1     0
1
```

These are in fact the coefficients of the generator polynomial of BCH(15,5):

$$g(X)=1+X+X^2+X^4+X^5+X^8+X^{10}$$

The function `bchgenpoly(n, k)` returns a $GF(2)$ array with the coefficients of the generator polynomial $g(X)$ in descending powers of X.

```
>> bchgenpoly(15, 5)

ans = GF(2) array.

Array elements =

   1   0   1   0   0   1   1   0   1   1   1
```

Example:

Systematically encode the message word $m=[10111]$ using the triple-error correcting BCH(15,5) code.

1. Form $m(X)=1+X^2+X^3+X^4$
2. Multiply $m(X)$ by $X^{15-5}=X^{10}$. Using Euclid's form for division:

$$X^{10}m(X)=X^{10}+X^{12}+X^{13}+X^{14}$$

$$¿q(X)g(X)+p(X)$$

$$¿(1+X+X^3+X^4)(1+X+X^2+X^4+X^5+X^8+X^{10})$$
$$+(1+X^4+X^7+X^8)$$

3. Add $p(X)=1+X^4+X^7+X^8$ to both sides of the equation above to create $c(X)$:

$$c(X)=p(X)+X^{10}m(X)$$

$$¿1+X^4+X^7+X^8+X^{10}+X^{12}+X^{13}+X^{14}$$

From $c(X)$ we construct the code word $c=[100010011010111]$. The first ten symbols are parity symbols followed by five message symbols.

Using MATLAB:

```
>> G = [1 1 1 0 1 1 0 0 1 0 1]

G =

     1     1     1     0     1     1     0     0     1     0
1

>> M = [1 0 1 1 1]

M =

     1     0     1     1     1

>> X10M = [zeros(1, 10), M]

X10M =

     0     0     0     0     0     0     0     0     0     0
1  …
     0     1     1     1
```

```
>> [Q, P] = gfdeconv(X10M, G)

Q =

     1     1     0     1     1

P =

     1     0     0     0     1     0     0     1     1

>> P = [P, zeros(1, 10-length(P))]

P =

     1     0     0     0     1     0     0     1     1     0

>> C = [P, M]

C =

     1     0     0     0     1     0     0     1     1     0
1  …
     0     1     1     1
```

This agrees with the earlier result $c=[100010011010111]$.

Reed-Solomon Codes

Primitive Binary BCH codes are based on polynomials over $GF(2)$. Reed-Solomon (RS) codes are non-binary q-ary codes based on polynomials over $GF(q)$. All practical RS codes used in modern digital communication systems set $q=2^m$. The parameters for a t-error-correcting Reed-Solomon code are:

Block Length:	$n=2^m-1$
Parity-Check Symbols:	$n-k=2t$
Minimum Distance:	$d_{min}=2t+1$

As with BCH codes the generator polynomial for a RS code is specified in terms of its roots from $GF(2^m)$. The generator polynomial $g(X)$ of the t-error-correcting Reed-Solomon code of length $n=2^m-1$ is the polynomial of *lowest degree* over $GF(2^m)$ that has as roots:

$\alpha, \alpha^2, \alpha^3, \cdots, \alpha^{2t}$

Since the generator polynomial is now over $GF(2^m)$ the minimal polynomial for α^i is simply $\phi_i(X)=X+\alpha^i$. The generator polynomial is then:

$g(X)=(X+\alpha)(X+\alpha^2)\cdots(X+\alpha^{2t})$
Note that the coefficients of $g(X)$ as well as any message polynomial $m(X)$ or code word polynomial $c(X)$ are over $GF(2^m)$. To represent binary message or code word sequences as elements of $GF(2^m)$ the binary n-tuple forms are used.

Because Reed-Solomon codes correct symbols over $GF(2^m)$ that represent strings of m binary digits they are widely used in burst error environments. All m binary digits of a RS code symbol can be incorrect and yet be fully recovered.

Example:

Determine the generator polynomial $g(X)$ for the single-error correcting RS(7,5) code.

Here $n=7=2^3-1$ so $m=3$ and we are working over $GF(8)$. Also $k=5$ and $t=1$.
The number of parity symbols is $n-k=2$ which is in fact equal to $2t=2$.

The generator polynomial $g(X)$ has α and α^2 as roots and is given by:

$g(X)=(X+\alpha)(X+\alpha^2)$

$$\Rightarrow X^2+(\alpha+\alpha^2)X+\alpha^3$$

$$\Rightarrow \alpha^3+\alpha^4 X+X^2$$

Systematically encode the binary 15-tuple message sequence $m=[100101110111100]$ using the single-error correcting RS(7,5) code.

Convert m to a representation over $GF(8)$:

$m=[1\ \alpha^6\ \alpha^3\ \alpha^5\ 1]$

1. Form $m(X)=1+\alpha^6 X+\alpha^3 X^2+\alpha^5 X^3+X^4$.

2. Multiply $m(X)$ by $X^{7-5}=X^2$. Using Euclid's form for division:

$$X^2 m(X)=X^2+\alpha^6 X^3+\alpha^3 X^4+\alpha^5 X^5+X^6$$

$$\Rightarrow q(X)g(X)+p(X)$$

$$\Rightarrow (\alpha^6+\alpha^2 X+\alpha^4 X^2+X^3+X^4)(\alpha^3+\alpha^4 X+X^2)+(\alpha^2+\alpha^2 X)$$

3. Add $p(X)=\alpha^2+\alpha^2 X$ to both sides of the equation above to create $c(X)$:

$$c(X)=p(X)+X^2 m(X)$$

$$\Rightarrow \alpha^2+\alpha^2 X+X^2+\alpha^6 X^3+\alpha^3 X^4+\alpha^5 X^5+X^6$$

Finally convert $c(X)$ to a binary sequence of seven 3-tuples to obtain c:

$c=[001001100101110111100]$
Verify using MATLAB:

Remember the function `deconv` expects the polynomial coefficients in descending order.

```
>> G = gf([1 6 3], 3)

G = GF(2^3) array. Primitive polynomial = D^3+D+1 (11 decimal)

Array elements =

   1   6   3

>> M = gf([1 7 3 5 1], 3)

M = GF(2^3) array. Primitive polynomial = D^3+D+1 (11 decimal)

Array elements =

   1   7   3   5   1

>> X2M = [M, 0, 0]

X2M = GF(2^3) array. Primitive polynomial = D^3+D+1 (11 decimal)

Array elements =

   1   7   3   5   1   0   0

>> [Q, P] = deconv(X2M, G)

Q = GF(2^3) array. Primitive polynomial = D^3+D+1 (11 decimal)

Array elements =

   1   1   6   4   5

P = GF(2^3) array. Primitive polynomial = D^3+D+1 (11 decimal)

Array elements =

   0   0   0   0   0   4   4

>> P = flip(P)

P = GF(2^3) array. Primitive polynomial = D^3+D+1 (11 decimal)

Array elements =
```

```
   4   4   0   0   0   0   0

>> X2M = flip(X2M)

X2M = GF(2^3) array. Primitive polynomial = D^3+D+1 (11 decimal)

Array elements =

   0   0   1   5   3   7   1

>> C = P + X2M

C = GF(2^3) array. Primitive polynomial = D^3+D+1 (11 decimal)

Array elements =

   4   4   1   5   3   7   1

>> C = double(C.x)

C =

     4     4     1     5     3     7     1

>> C = de2bi(C, 3)

C =

     0     0     1
     0     0     1
     1     0     0
     1     0     1
     1     1     0
     1     1     1
     1     0     0

>> C = reshape(transpose(C), 1, [])

C =

     0     0     1     0     0     1     1     0     0     1
0  …
     1     1     1     0     1     1     1     1     0     0
```

From which $c=[00100110010111011100]$ which agrees with the result above.