Chapter 4

Cyclic Block Codes

Spring 2009

Cyclic Block Codes

A cyclic code is characterized as a linear block code $B(n, k, d)$ with the additional property that for each code word

$$G = (g_{n-1}, g_{n-2}, \ldots, g_2, g_1, g_0)$$

All the cyclic shift versions of $b$ is also valid code words (i.e)

$$G^{(1)} = (g_{n-2}, g_{n-3}, \ldots, g_1, g_0, g_{n-1})$$

$$G^{(2)} = (g_{n-3}, g_{n-4}, \ldots, g_0, g_{n-1}, g_{n-2})$$

$$\vdots$$

$$G^{(n-2)} = (g_1, g_0, g_{n-1}, \ldots, g_{n-2})$$

$$G^{(n-1)} = (g_0, g_{n-1}, g_{n-2}, \ldots, g_2, g_1)$$

All are valid codewords and of course all the linear combinations between them.
Cyclic Block Codes

Example if we define a generator codeword as $G = 0001011$ then all the possible cyclic codewords are:

1. All zeros $000000$
2. $G = 0001011$
3. $G^{(1)} = 0010110$
4. $G^{(2)} = 0101100$
5. $G^{(3)} = 1011000$
6. $G^{(4)} = 0110001$
7. $G^{(5)} = 1100010$
8. $G^{(6)} = 1000101$
9. $C = G + G^{(1)} = 0011101$
10. $C^{(1)} = 0111010$
11. $C^{(2)} = 1110100$
12. $C^{(3)} = 1101001$
13. $C^{(4)} = 1010011$

Encoding by convolution

- This property can be formulated by using polynomial. Assume we can represent the code word $G$ as following:
  
  $$g(X) = g_0 + g_1X + g_2X^2 + \cdots + g_{n-2}X^{n-2} + g_{n-1}X^{n-1}$$

- For our Example $G = 0001011$ or $g(X) = 1 + X + X^2$

- If we take the generator polynomials and its first $k-1$ left shift, we find that we have $k$ linearly independent sequence.

These can form the generator matrix for the cyclic code, then

$$G = \begin{bmatrix}
1 & 0 & 1 & 1 & 0 & 0 & 0 \\
0 & 1 & 0 & 1 & 1 & 0 & 0 \\
0 & 0 & 1 & 0 & 1 & 1 & 0 \\
0 & 0 & 0 & 1 & 0 & 1 & 1
\end{bmatrix}$$
Encoding by convolution

- Going back to the general notations

\[ g(X) = g_0 + g_1X + g_2X^2 + \cdots + g_{n-k}X^{n-k-1} + b_{n-k}X^{n-k} \]

So that

\[
G = \begin{pmatrix}
g_0 & g_1 & \cdots & g_{n-k} & 0 & \cdots & 0 & 0 & \cdots & 0 & 0 \\
0 & g_0 & g_1 & \cdots & g_{n-k} & 0 & 0 & 0 & \cdots & 0 & 0 \\
\vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\
0 & 0 & 0 & 0 & 0 & 0 & g_0 & g_1 & \cdots & g_{n-k} & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & g_0 & g_1 & \cdots & g_{n-k} \\
\end{pmatrix}
\]

Encoding by convolution

- Then the generator matrix can be written as

\[
G = \begin{pmatrix}
X^{k-1}g(X) \\
\vdots \\
xg(X) \\
g(X)
\end{pmatrix}
\]

assume the information bits are given by

\[ I = (i_{k-1}, i_{k-2}, \cdots, i_2, i_1, i_0) \]

Then

\[ b = IG = _0g(X) + i_1Xg(X) + i_2X^2g(X) + \cdots + i_{k-1}X^{k-3}g(X) \]
Encoding by convolution

- But as information vector \( i \) can be written as polynomial also
  \[
i(X) = i_0 + i_1 X + i_2 X^2 + \ldots + i_{k-2} X^{k-2} + i_{k-1} X^{k-1}
  \]
- Then
  \[
c = (i_0 + i_1 X + i_2 X^2 + \ldots + i_{k-1} X^{k-1}) g(X)
  \]
- Or
  \[
c(X) = i(X) g(X)
  \]
- This forms non-systematic way for cyclic coding. But it proves that \( c(X) \) divides \( g(X) \) and \( g(X) \) is of order \((n-k)\).

Cyclic Block Codes

- The cyclic property can be formulated by using polynomial representation,
  \[
c(X) = c_0 + c_1 X + c_2 X^2 + \ldots + c_{n-2} X^{n-2} + c_{n-1} X^{n-1}
  \]
If we cycle shift \( g \) by one bit. This generates
\[
c^{(1)}(X) = c_{n-1} + c_0 X + c_1 X^2 + \ldots + c_{n-3} X^{n-2} + c_{n-2} X^{n-1}
\]
But as
\[
X. i(X) = c_0 X + c_1 X^2 + \ldots + c_{n-3} X^{n-2} + c_{n-2} X^{n-1} + c_{n-1} X^n
\]
\[
X. i(X) = c_0 X + c_1 X^2 + \ldots + c_{n-3} X^{n-2} + c_{n-2} X^{n-1} + c_{n-1} X^n + c_{n-1} + c_{n-1}
\]
\[
Xc(X) = c_{n-1} (1 + X^n) + c^{(1)}(X)
\]
Or
\[
c^{(1)}(X) = Xc(X) \mod (X^n + 1)
**Cyclic Block Codes**

- By extension we can write
  \[ c^{(n)}(X) = X^n c(X) \mod (X^n + 1) \]

- Example if \( c = 1101 \) by using previous equation find \( c^{(3)} \)
  \[
  c(X) = 1 + X^2 + X^3 \\
  X^3 c(X) = X^3 + X^5 + X^6
  
  \]

- Then
  \[ c^{(3)}(X) = X^3 c(X) \mod (X^4 + 1) = X^4 + X^3 + X \]

- \( i^{(3)} \) is given by 1110

---

**Cyclic Block Codes**

- But as generator polynomial is valid code word
  \[
  Xg(X) = g_{n-1}(1 + X^n) + g^{(0)}(X)
  \]

  Or
  \[
  X^n + 1 = g^{(0)}(X) + Xg(X)
  
  But as \( g(X) = i(X)g(X) \). This leads
  \[
  X^n + 1 = i(X)g(X) + Xg(X) \\
  X^n + 1 = \left[ i(X) + X \right] g(X)
  
  Which proves that the generator polynomial \( g(X) \) should be a factor of \( X^n + 1 \)
**Generator Polynomial**

- The generator sequence/polynomial for cyclic code must have the following properties:
  1. Generator polynomial is a factor of $X^n + 1$
  2. Degree of the generator polynomial is $n - k$

So if we know the generator polynomial we can find:

1. Number of the parity bits which is the order of $g(X)$
2. The code length which can be calculated from minimum value of $n$ where $X^n + 1$ divides $g(X)$.

For our example $n - k = 3$, $X^7 + 1$ divides $g(X)$ then $n = 7$.

**Systematic block code**

- In systematic encoding the message is part from the code
- Or $C = (i_{k-1}, i_{k-2}, \ldots, i_2, i_1, p_{n-k-1}, \ldots, p_1, p_0)$

In polynomial format:

$c(X) = p_0 + p_1 X + \cdots + p_{n-k-1} X^{n-k-1} + i_0 X^{n-k} + i_1 X^{n-k+1} + \cdots + i_{k-1} X^{n-1}$

$c(X) = p(X) + X^{n-k} i(X)$

- Or $X^{n-k} i(X) = c(X) + p(X)$

- But as $c(X)$ divides $g(X)$ and $p(X)$ of order $n - k - 1$ then $p(X) = X^{n-k} i(X) \mod g(x)$
**Systematic block code**

**Example:**
The (7,4) Hamming code is defined by the generator polynomial

\[ g(X) = 1 + X + X^3 \]

Encode the following message \( i = 1001 \)

\[ i(X) = 1 + X^2 \]

The **nonsystematic** encoding will lead to

\[ c(X) = g(X)i(X) = 1 + X + X^4 + X^6 \]

While the **systematic** coding will give

\[ b(X) = X^{n-k}i(X) \text{mod } g(X) + X^{n-k}i(X) = \\
= X + X^3 + X^4 + X^6 \]

---

**Time Domain Encoding**

Another method of doing long division is by using shift register as shown

\[ X^{n-k}u(X) \]
Example:
For our example

To see how this circuit work let us consider the encoding of \( u = 0110 \)
We start the shift register by initial value of 000
The remainder is content of the shift register when the final bit is shifted

The following table gives the shift register after each shift

<table>
<thead>
<tr>
<th>Input</th>
<th>Register Content</th>
</tr>
</thead>
<tbody>
<tr>
<td>-</td>
<td>000</td>
</tr>
<tr>
<td>0</td>
<td>000</td>
</tr>
<tr>
<td>1</td>
<td>001</td>
</tr>
<tr>
<td>1</td>
<td>011</td>
</tr>
<tr>
<td>0</td>
<td>110</td>
</tr>
<tr>
<td>0</td>
<td>111</td>
</tr>
<tr>
<td>0</td>
<td>101</td>
</tr>
<tr>
<td>0</td>
<td>001</td>
</tr>
</tbody>
</table>

The codeword is \( 0110001 \)
A possible criticism of the circuit of previous circuit is that after the information has been entered a further $n - k$ shifts are required before the syndrome is formed.

This can be overcome by the following configuration

The following table gives the shift register after each shift

<table>
<thead>
<tr>
<th>Input</th>
<th>Register Content</th>
</tr>
</thead>
<tbody>
<tr>
<td>-</td>
<td>000</td>
</tr>
<tr>
<td>0</td>
<td>000</td>
</tr>
<tr>
<td>1</td>
<td>011</td>
</tr>
<tr>
<td>1</td>
<td>101</td>
</tr>
<tr>
<td>0</td>
<td>001</td>
</tr>
</tbody>
</table>

The codeword is 0110001
**SYNDROME OF A CYCLIC CODE**

It is fairly easy to show that if we divide an error sequence by the generator and take the remainder, the result is the syndrome.

**Proof**

To do this, consider the received sequence \( r(X) \), as consisting of the sum of the code sequence \( b(X) \) and an error pattern \( e(X) \):

\[
r(X) = b(X) + e(X)
\]

The syndrome equal the remainder resulting from dividing \( r(X) \) by \( g(X) \), or

\[
r(X) = q(X)g(X) + s(X)
\]

But as \( b(X) = u(X)g(X) \) this leads to

\[
e(X) = [q(X) + i(X)]g(X) + S(X)
\]

The same as block codes all possible syndrome can be associated to error patterns. \( s(X) \equiv e(X) \mod g(X) \)

But as

\[
e(X) = a(X)g(X) + s(X)
\]

then

\[
e^{01}(X) = Xa(X) + e_{n-1}(X^n + 1)
\]

\[
e^{01}(X) = Xa(X)g(x) + e_{n-1}(X^n + 1) + Xs(X)
\]

Which means that \( e^{01}(X) \mod g(X) \) equal \( Xs(X) \mod g(X) \)

Which means cycle shift in the error pattern is equivalent to cyclic shift of the syndrome in the shift register.
The same as block codes all possible syndrome can be associated to error patterns. \( s(X) = e(X) \mod g(X) \)

For our example this gives the following table:

<table>
<thead>
<tr>
<th>Error position</th>
<th>( e )</th>
<th>( S )</th>
</tr>
</thead>
<tbody>
<tr>
<td>6</td>
<td>1000000</td>
<td>101</td>
</tr>
<tr>
<td>5</td>
<td>0100000</td>
<td>111</td>
</tr>
<tr>
<td>4</td>
<td>0010000</td>
<td>110</td>
</tr>
<tr>
<td>3</td>
<td>0001000</td>
<td>011</td>
</tr>
<tr>
<td>2</td>
<td>0000100</td>
<td>100</td>
</tr>
<tr>
<td>1</td>
<td>0000010</td>
<td>010</td>
</tr>
<tr>
<td>0</td>
<td>0000001</td>
<td>001</td>
</tr>
</tbody>
</table>

But if we start from 001 and shift the register with feedback, the result is:

<table>
<thead>
<tr>
<th>Error position</th>
<th>( S )</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>001</td>
</tr>
<tr>
<td>1</td>
<td>010</td>
</tr>
<tr>
<td>2</td>
<td>100</td>
</tr>
<tr>
<td>3</td>
<td>011</td>
</tr>
<tr>
<td>4</td>
<td>110</td>
</tr>
<tr>
<td>5</td>
<td>111</td>
</tr>
<tr>
<td>6</td>
<td>101</td>
</tr>
</tbody>
</table>

The relation with the previous table is apparent.
Let us look at some examples of this principle in operation. Let the received sequence equal 0010011.

There are two ways to solve for the syndrome:
1) $S(X) = R(x) \mod g(X) = X^2 + 1$ or 101

The codeword corresponding to the received information 0010 is calculated 0010110.

The syndrome can then be calculated by $110 + 011 = 101$.

In both cases this means the error pattern is 1000000.

The original word is 1010.

If we repeat the same for 1110011.

There are two ways to solve for the syndrome:
1) $S(X) = R(x) \mod g(X) = X^2 + 1$ or 111

The codeword corresponding to the received information 1110 is calculated 1110100. Or $s = 111$.

The error pattern 0100000.

If 111 is shifted in the register we get 101, which means the error is shifted one bit to the left.
**SYNDROME OF A CYCLIC CODE**

- Double error correction can be done by the same method
  \[ g(X) = X^5 + X^4 + X^2 + X + 1 \]

- If an encoder previously discussed is used to generate the syndrome for single error sequence. It gives

<table>
<thead>
<tr>
<th>Error position</th>
<th>Syndrome</th>
<th>Error position</th>
<th>Syndrome</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0010110</td>
<td>8</td>
<td>0000010</td>
</tr>
<tr>
<td>1</td>
<td>0010101</td>
<td>9</td>
<td>0001000</td>
</tr>
<tr>
<td>2</td>
<td>0101100</td>
<td>10</td>
<td>0000100</td>
</tr>
<tr>
<td>3</td>
<td>1011000</td>
<td>11</td>
<td>0010000</td>
</tr>
<tr>
<td>4</td>
<td>0101011</td>
<td>12</td>
<td>0100000</td>
</tr>
<tr>
<td>5</td>
<td>1010111</td>
<td>13</td>
<td>0100000</td>
</tr>
<tr>
<td>6</td>
<td>1001101</td>
<td>14</td>
<td>1000000</td>
</tr>
</tbody>
</table>

- The syndromes to look for are those resulting from an error in bit 14, either on its own or in combination with one other bit. This gives rise to the list shown in

<table>
<thead>
<tr>
<th>Error position</th>
<th>Syndrome</th>
<th>Error position</th>
<th>Syndrome</th>
</tr>
</thead>
<tbody>
<tr>
<td>14,0</td>
<td>10010111</td>
<td>14,8</td>
<td>10000010</td>
</tr>
<tr>
<td>14,1</td>
<td>10101110</td>
<td>14,9</td>
<td>10001000</td>
</tr>
<tr>
<td>14,2</td>
<td>11011100</td>
<td>14,10</td>
<td>10010000</td>
</tr>
<tr>
<td>14,3</td>
<td>00111000</td>
<td>14,11</td>
<td>10100000</td>
</tr>
<tr>
<td>14,4</td>
<td>11100111</td>
<td>14,12</td>
<td>10100000</td>
</tr>
<tr>
<td>14,5</td>
<td>01001110</td>
<td>14,13</td>
<td>11000000</td>
</tr>
<tr>
<td>14,6</td>
<td>00001011</td>
<td>14</td>
<td>10000000</td>
</tr>
<tr>
<td>14,7</td>
<td>10000001</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Now suppose that the errors are in positions 12 and 5. By adding the syndromes of those single errors we get a syndrome value 11101110 as being computed by the encoder.

It is not on the syndrome list
So shift once 11001011 and once again 10000001.
Which in the list 14 - 2shift = 12 the error position

We therefore correct bit 12 and invert the leftmost bit of the syndrome to leave 00000001. A further seven shifts, making nine in all, will produce the pattern 10000000 indicating another correctable error in bit 5.

Suppose the errors are in bits 12 and 10. The syndrome calculated by using the encoder will be 00101000.

Two shifts will bring this to 10100000. We correct bit 12 and invert the first bit of the syndrome, leaving 00100000.

Two further shifts produce the syndrome 10000000 indicating an error in bit 10.
Shorted Cyclic Code

Suppose we have a \((n, k)\) cyclic code shortened to \((n - i, k - i)\).

We receive a sequence \(r(X)\) and wish to compute the syndrome of \(X^j r(X)\), where \(j\) is the sum of \(i\) (number of bits removed) and \(n - k\) (the usual amount by which the syndrome is pre-shifted).

If \(s_1(X)\) is the syndrome of \(r(X)\) and \(s_2(X)\) is the syndrome of \(X^j\), then the required syndrome is \(s_1(X)s_2(X) \mod g(X)\).

We therefore multiply the received sequence by \(s_2(X) \mod g(X)\) by feeding it into the appropriate points of the shift registers.

Consider, for example, the \((15, 11)\) code generated by \(X^4 + X + 1\), shortened to \((12, 8)\). First we compute \(X^7 \mod g(X)\), which is found to be \(X^3 + X + 1\).

Now we arrange the feeding of the received sequence into the shift registers as shown, such that there is a feed into the \(X^3\), \(X\) and \(1\) registers.

If a sequence 1000000000000 is fed into this arrangement, the syndrome is 1000.

If the first transmitted bit is in error, that fact will therefore be indicated immediately. Any other syndrome will indicate a need to shift until 1000 is obtained or the error is found to be uncorrectable.
Example
Consider (15,11) code generated by $X^{15} + X + 1$, shortened to (12,8)
First we compute $X^7 \mod g(X)$ which is $X^3 + X + 1$
Now we feedback the received sequence as shown

Shorted Cyclic Code

If a sequence 100000000000 is fed into the arrangement the resultant contents of the shift register are

<table>
<thead>
<tr>
<th>Input</th>
<th>Register Content</th>
<th>Input</th>
<th>Register Content</th>
</tr>
</thead>
<tbody>
<tr>
<td>-</td>
<td>0000</td>
<td>0</td>
<td>1101</td>
</tr>
<tr>
<td>1</td>
<td>1011</td>
<td>0</td>
<td>1001</td>
</tr>
<tr>
<td>0</td>
<td>0101</td>
<td>0</td>
<td>0001</td>
</tr>
<tr>
<td>0</td>
<td>1010</td>
<td>0</td>
<td>0010</td>
</tr>
<tr>
<td>0</td>
<td>0111</td>
<td>0</td>
<td>0100</td>
</tr>
<tr>
<td>0</td>
<td>1110</td>
<td>0</td>
<td>1000</td>
</tr>
<tr>
<td>0</td>
<td>1111</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
EXPURGATED CYCLIC CODES

- Expurgation is the conversion of information bit into parity bit and keeping the length the same.

- If a cyclic code has an odd value of the minimum distance, multiplying the generator polynomial by \( X + 1 \) has the effect of expurgating the code and increasing \( d_{\text{min}} \) by 1.

- For example
  
  \[
  g(X) = X^3 + X + 1
  
  g(X)(X + 1) = X^4 + X^3 + X + 1
  \]

- Any codeword of the new code consist of code word of the original code multiplied by \( X + 1 \) (shifted left and added to itself).

- For example the sequence 100101 from the original codeword become
  
  1000101 + 0001011 = 1001110
BCH Codes

Many of the most important block codes for random-error correction fall into the family of BCH codes, named after their discoverers Bose, Chaudhuri and Hocquenghem.

BCH codes include Hamming codes as a special case.

The construction of a t-error correcting binary BCH code starts with an appropriate choice of length:

\[ n = 2^m - 1 \text{ (} m \text{ is integer } \geq 3) \]

It is given that \( n - k \leq mt \) and \( d_{\min} \geq 2t + 1 \)

Cyclic Codes for Burst Error Correction

Many of
Fire codes

Fire codes are cyclic codes that can correct single burst errors with syndrome that can be split into two components for faster decoding.

The form of the generator polynomial for Fire code which is capable of correcting burst of length up to \( l \) is:

\[
g(X) = (X^{2^{l-1} + 1}) h(X)
\]

Where \( h(X) \) is irreducible polynomial of length \( m \geq l \) which is not a factor of \( X^{2^{l-1} + 1} \).

I.E the order \( p \) of \( h(X) \) is not factor of \( 2^l - 1 \). the length of the code will be the lowest common multiple of \( p \) and \( 2^l - 1 \).

An example is \( h(X) = X^4 + X + 1 \) is not a factor of \( X^7 + 1 \) then

\[
g(X) = (X^7 + 1)(X^4 + X + 1)
\]

This generator polynomial generates \((105,94)\) Fire code which can correct burst of length \( l \) or less.