Cyclic codes

Vahid Meghdadi

Reference: Error Correction Coding by Todd K. Moon

February 2008

1 Definitions

Definition 1. A ring $< \mathbf{R}, +, . >$ is a set $\mathbf{R}$ with two binary operation $+$ (addition) and . (multiplication) defined on $\mathbf{R}$ such that:

1. $< \mathbf{R}, +, . >$ is an Abelian [1] (commutative) group
2. The multiplication operation is associative:
$$\forall a, b, c \in \mathbf{R}, (a.b).c = a.(b.c)$$
3. the left and right distributive laws hold:
$$a.(b + c) = a.b + a.c$$
$$(a + b).c = a.c + a.b$$

$\square$

The ring $< \mathbf{R}, +, . >$ is frequently referred to as $\mathbf{R}$.

Example 1. $< \mathbf{Z}_4, +, . >$ is a ring with two operations of addition and multiplication defined as follows:

+	0	1	2	3
0	0	1	2	3
1	1	2	3	0
2	2	3	0	1
3	3	0	1	2

.	0	1	2	3
0	0	0	0	0
1	0	1	2	3
2	0	2	0	2
3	0	3	2	1

$\square$

[1] An abelian group is a set A together with an operation "." that combine any two elements a and b to form another element $c = a.b$. To be abelian, the following requirements must be met: Closure ($a.b \in A$), Commutativity $a.b = b.a$, Associativity ($(a.b).c = a.(b.c)$), Identity element $\exists e \in A, e.a = a.e = a$, Inverse element $\forall a \in A, \exists b \in A, a.b = b.a = e$

Definition 2. Ring of polynomials

Let $\mathbf{R}$ be a ring. A polynomial $f(x)$ of degree n with coefficients in $\mathbf{R}$ is $f(x) = \sum_{i=0}^{n} a_i x^i$. The symbol x is an indeterminate.

Definition 3. The set of all polynomials with the coefficients in a ring $\mathbf{R}$, using the normal operations of addition and multiplication forms a ring called polynomial ring and denoted by $\mathbf{R}[x]$.

Example 2. Let's $\mathbf{R} =< \mathbf{Z}_4, +, . >$ and $\mathbf{R}[x] = \mathbf{Z}_4[x]$ is the polynomial ring defined in $\mathbf{R}$. $1 + x \in \mathbf{R}[x], 3x + 2x^2 \in \mathbf{R}[x]$ then:

$$(1 + x) + (3x + 2x^2) = 1 + 0x + 2x^2 = 1 + 2x^2$$

$$(1 + x).(3x + 2x^2) = 3x + 2x^2 + 3x^2 + 2x^3 = 3x + x^2 + 2x^3$$

$\square$

1.1 Quotient rings

Let's define a subset of all polynomials where addition and multiplication are defined as follows. Let $a(x) \in \mathbf{R}[x]$ and $b(x) \in \mathbf{R}[x]$. $a(x) + b(x)$ is defined as before but $a(x).b(x)$ is defined as $a(x).b(x)$ modulo $(x^n - 1)$. This is represented by $\mathbf{R}[x]/(x^n - 1)$.

Example 3. $\mathbf{Z}_2[x]/(x^3+1) =< \{0, 1, x, x+1, x^2, x^2+x, x^2+1, x^2+x+1\}, +, . >$. Note that in $\mathbf{Z}_2$, $x^n - 1 = x^n + 1$. For example:

$$\begin{aligned}
(x + 1)(x^2 + x) &= (x^3 + x^2 + x^2 + x) \ \texttt{modulo} \ (x^3 + 1) \\
&= (x^3 + x) \ \texttt{modulo} \ x^3 + 1 \\
&= (x^3 + 1) + x + 1 \ \texttt{modulo} \ x^3 + 1 \\
&= x + 1
\end{aligned}$$

$\square$

Note that the set of polynomials containing all possible polynomials with order less than n and with the two operations addition and multiplication forms a ring.

1.2 Ideals in rings

Ideal I in a ring $\mathbf{R}$ is a subset of $\mathbf{R}$ defined with the same two binary operations $+$ and $.$ where:

1. $\mathbf{I}$ forms a group under addition operation
2. For any $a \in I$ and any $r \in R$, $ar \in I$.

Note: Cyclic codes form an ideal in a ring of polynomials.

Definition 4. An ideal I in a ring is said to be **principal** if there exists some $g \in I$ such that every element $a \in I$ can be expressed as a product $a = mg$ for some $m \in \mathbf{R}$. For a principal ideal, such an element g is called the **generator element**. This ideal generated by g is denoted as $< g >$.

Theorem 1. Let I be an ideal in $\mathbf{F}_q[x]/(x^n - 1)$. Then

1. There is a unique monic polynomial $g(x) \in I$ of minimal degree.[2]
2. I is principal with generator $g(x)$.
3. $g(x)$ divides $(x^n - 1)$ in $\mathbf{F}_q[x]$.

Proof: We are showing that among all polynomials in the ideal, there is just one with a minimal degree. Suppose there are two polynomials with minimum degree in I, say $g(x)$ and $f(x)$, and $g(x) \neq f(x)$. Since the ideal forms a group, $h(x) = f(x) - g(x) \in I$ and the degree of $h(x)$ is less than the degree of $g(x)$ and $h(x)$. This is contrary with the minimality of degree of $g(x)$ and $f(x)$.
We next show that I is principal, i.e., every element of I is a multiple of $g(x)$. Suppose there is an $f(x)$ which is not a multiple of $g(x)$. It means that:

$$f(x) = m(x)g(x) + r(x)$$

where $\deg(r) < \deg(g)$. Since $mg \in I$, $r = f - mg$ is also in I and it is in contrary with the minimality of degree of $g(x)$, unless $r = 0$.
To show that $g(x)$ divides $(x^n - 1)$, suppose the contrary. So we can say:

$$x^n - 1 = h(x)g(x) + r(x)$$

where $0 \leq \deg(r) \leq \deg(g)$. But $h(x)g(x) \in I$ and $r(x) = (x^n - 1) - h(x)g(x)$ is the additive inverse of $h(x)g(x)$ and so in I. It is contrary to the degree minimality of $g(x)$.

$\square$

Conclusion:

1. If a monic polynomial $g(x)$ divides $x^n - 1$, it can be used to generate an ideal $I =< g(x) >$.
2. In the ring $\mathbf{F}q[x]/(x^n - 1)$, different ideals can be obtained by selecting different divisors $g(x)$ of $x^n - 1$.

2 Cyclic code definition

Consider a vector $C = (c_0, c_1, ..., c_{n-1}) \in GF_{q^n}$ where GF_q is defined as $< \mathbf{Z}_q, +, . >$.

Definition 5. An (n, k) block code $\mathcal{C}$ is said to be cyclic if it is linear and for every code word $C = (c_0, c_1, ..., c_{n-1}) \in \mathcal{C}$, its right cyclic shift $C' = (c_{n-1}, c_0, c_1, ..., c_{n-2}) \in \mathcal{C}$.

[2]A polynomial is monic if the coefficient of the leading term, the term of highest degree, is equal to 1.

2.1 Polynomial representation of cyclic codes

Let $\mathcal{C}$ be a cyclic code. Each codeword can be represented as a polynomial $c(x)$ defined as:

$$c(x) = \sum_{i=0}^{n-1} c_i x^i \Leftrightarrow (c_0, c_1, ..., c_{n-1}) \in \mathcal{C}$$

Because of the definition, a cyclic shift of every codeword is also a codeword. Let $c(x) \in GF_q[x]/(x^n - 1)$ where GF_q is $< \mathbf{Z}_q, +, . >$.
Property: $xc(x) = (c_{n-1}, c_0, c_1, ..., c_{n-2}) \in \mathcal{C}$
proof

$$\begin{aligned} xc(x) &= c_0 x + c_1 x^2 + ... + c_{n-1} x^n + c_{n-1} - c_{n-1} \text{ modulo } x^n - 1 \\ &= c_{n-1} + c_0 x + c_1 x^2 + ... + c_{n-2} x^{n-1} + c_{n-1}(x^n - 1) \text{ modulo } x^n - 1 \\ &= c_{n-1} + c_0 x + c_1 x^2 + ... + c_{n-2} x^{n-1} \\ &\Leftrightarrow (c_{n-1}, c_0, c_1, ..., c_{n-2}) \end{aligned}$$

□

Conclusion: $x^r c(x) \in \mathcal{C}$.

Since the code is linear, $C_1 + C_2 \in \mathcal{C}, \forall C_1, C_2 \in \mathcal{C}$. On the other side $a(x)c(x), \forall a$ and $\forall c \in \mathcal{C}$ is a linear combination of codewords and will be another codeword. So code polynomials form an ideal in $GF_q[n]/(x^n - 1)$.

2.1.1 Conclusion

- A cyclic code is a linear block code
- To each codeword corresponds a polynomial
- The code polynomials form an ideal in $GF_q[x]/(x^n - 1)$.
- For a cyclic code there is a generator $g(x)$ which is a monic minimal degree polynomial and a divisor of $x^n - 1$, that can generate all of the codewords: $c(x) = m(x)g(x)$

Example 4. We consider a cyclic code of length 15 with binary coefficients ($q = 2$). By multiplication one can verify that

$$x^{15} - 1 = (1 + x)(1 + x + x^2)(1 + x + x^4)(1 + x + x^2 + x^3 + x^4)(1 + x^3 + x^4)$$

So there are polynomials of degrees 1, 2, 4, 4, and 4 to be used as generator. However any product of these polynomials also can be used as generator. For example for a degree 5 we can choose $g(x) = (1 + x)(1 + x + x^4)$. So a cyclic code (15,11) can be generated.

□

Until now, we can generate the space of codeword. Now how many message can be mapped to a codeword? In the following non-systematic and systematic cyclic coding will be discussed.

3 Non-systematic encoding and parity check

Consider a message of size k as $\mathbf{m} = [m_0 m_1 ... m_{k-1}]$. The corresponding polynomial is $m(x) = m_0 + m_1 x + ... + m_{k-1}x^{k-1}$. There are q^k degree of freedom, so q_k distinct polynomials. Suppose a $g(x)$ of order $n-k$. So $m(x)g(x)$ will give q^k distinct polynomials that form our ideal, and then our cyclic code.

Note: if $m_1(x)g(x) = m_2(x)g(x)$, then $m_1(x)$ must be equal to $m_2(x)$.

So in this way, we have a systematic method to map our messages to the codeword by a one-to-one mapping.

$$c(x) = m(x)g(x) = m_0 g(x) + m_1 x g(x) + m_2 x^2 g(x) + ... + m_{k-1}x^{k-1}g(x)$$

This can be expressed using matrix representation:

$$c(x) = [m_0 m_1 ... m_{k-1}] \begin{bmatrix} g(x) \\ xg(x) \\ ... \\ x^{k-1}g(x) \end{bmatrix}$$

The sequence representation will be:

$$\mathbf{c} = [m_0 m_1 ... m_{k-1}] \begin{bmatrix} g_0 & g_1 & \cdots & g_r & & & \\ & g_0 & g_1 & \cdots & g_r & & \\ & & \ddots & \ddots & & \ddots & \\ & & & g_0 & g_1 & \cdots & g_r \end{bmatrix}$$

where $r = n-k$. So the coding operation is

$$\mathbf{c} = \mathbf{mG}$$

Exercise: Let $n = 7$ and $g(x) = 1 + x + x^2 + x^3 + x^4$. Give the generator matrix of the code. Give the codeword for the messages [1 0 0] and [1 1 0]. □

The question is now how to produce the parity check matrix. To answer to this question, suppose that there is a polynomial with degree k such that $h(x)g(x) = x^n - 1$. For a codeword $c(x)$:

$$c(x)h(x) = m(x)g(x)h(x) = m(x)(x^n - 1) = 0$$

Note that the multiplication operation in the ring is defined always modulo $(x^n - 1)$. So we can check easily if a received sequence is a codeword: $r(x)$ is a codeword if and only if $r(x)h(x) \mod (x^n - 1) = 0$. $h(x)$ is called *Parity Check Polynomial.*

The remainder of division $r(x)/h(x)$ is called *Syndrome.* If it is zero, no transmission error is supposed. Suppose an error pattern of $e(x)$ such that $r(x) = c(x) + e(x)$, then:

$$h(x)r(x) = h(x)c(x) + h(x)e(x) = m(x)(x^n - 1) + h(x)e(x) = h(x)e(x)$$

The important conclusion is the syndrome is independent of the codeword and depends only to the error pattern. So one way to correct the error is to stock all the possible syndromes with corresponding error patterns in a table. Then we compare our calculated syndrome with this table in order to identify the most probable error pattern. The corrected codeword will be $r(x) - e(x)$.

Now we are to construct the parity check matrix from $h(x)$. It can be shown that the parity check matrix has the following form:

$$\mathbf{H} = \begin{bmatrix} h_k & h_{k-1} & \cdots & h_0 & & & & \\ & h_k & h_{k-1} & \cdots & h_0 & & & \\ & & \ddots & \ddots & & \ddots & & \\ & & & h_k & h_{k-1} & \cdots & h_0 \end{bmatrix}$$

Exercise: For the previous exercise give the parity check polynomial and matrix. □

4 Systematic cyclic codes

It is not complicated to modify our mapping to obtain systematic cyclic code. First we construct the polynomial $x^{n-k}m(x)$ of degree n. Then we divide it by $g(x)$ and calculate the remainder:

$$x^{n-k}m(x) = q(x)g(x) + d(x)$$

Now we form the codeword as $c(x) = x^{n-k}m(x) - d(x) = q(x)g(x)$. Since it is a multiple of $g(x)$, it must be a codeword. The vector representation of the codeword is:

$$\mathbf{c} = [-d_0, -d_1, ..., -d_{n-k-1}, m_0, m_1, ..., m_{k-1}]$$

Exercise: For the previous exercise compute the codeword for the message $[1, 0, 1]$.

5 Circuit realization

One of the reasons that the cyclic codes are very popular is that the circuit implementing these codes are very simple. In this section some circuits for some polynomial operations are presented. Suppose that $a(x)$ is a polynomial defined in GF_q and the block "D" in the figures is a storage element that hold one symbol in the field $\mathbf{F}$. All of these elements in a circuit is clocked simultaneously.

5.1 Polynomial Multiplication

Let $a(x) = a_0 + a_1x + a_2x^2 + ... + a_kx^k$ and $h(k) = h_0 + h_1x + h_2x^2 + ... + h_rx^r$. The product $b(x) = a(x)h(x)$ is to be calculated. The circuit presented in Figure 1 is used to this polynomial multiplication. It is the same as a convolution operation which happens when two polynomials are multiplied. At the first all the registers is initialized to zero. After k clock cycles all the $a(x)$ coefficients are entered. Then, the input is set to zero and there are r more clock cycles where the rest of $b(x)$ coefficients are getting out.

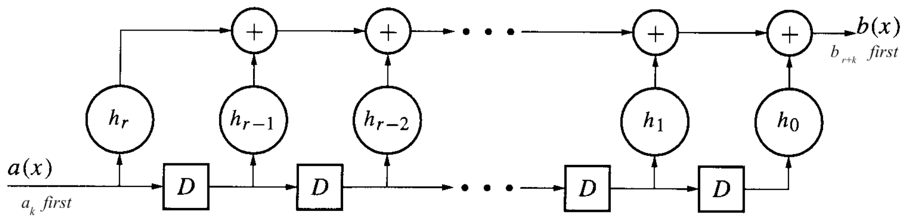

Figure 1: Circuit for multiplying of two polynomials

It is possible not to put all the additions in a direct chain and use a pipeline like architecture, which is in fact the transposed circuit of the first one (Figure 2). So this configuration is more suitable for high speed applications.

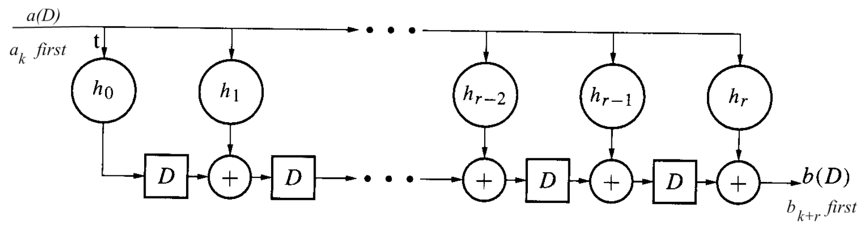

Figure 2: Circuit for high speed multiplying of two polynomials

5.2 Polynomial division

Now we are giving a circuit to compute not only the quotient of a polynomial division but also the remainder. So in the operation $d(x) = g(x)q(x) + r(x)$, $d(x)$ and $(g(x)$ are known and the circuit will give $q(x)$ and $r(x)$. Suppose $d(x) = d_0 + d_1x + ... + d_nx^n$ and $g(x) = g_0 + g_1x + ... + g_px^p$. The last coefficient of $g(x)$ i.e. g_p is not zero. The degree of $q(x)$ is at most $n - p$ and the degree of $r(x)$ is at most $p - 1$. The circuit is shown in Figure 3.

The algorithm is as follows:

1. All registers are cleared.

2. During p clocks, the p first coefficients of $d(x)$ are clocked in.

3. During $n - p$ new clock cycles, the rest of coefficients of $d(x)$ are shifted in while the quotient $q(x)$ is getting out from the output (last coefficient first).

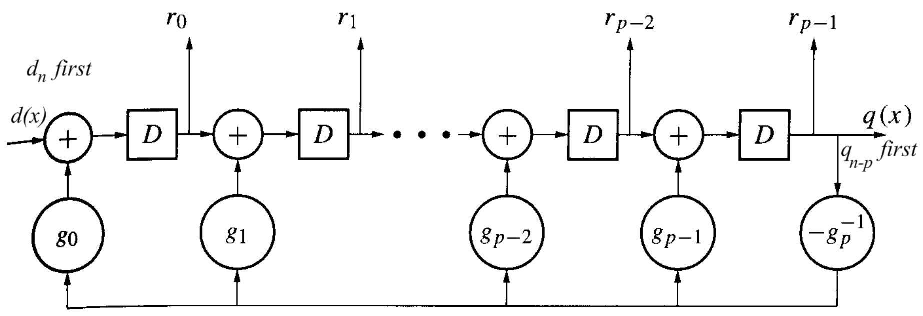

Figure 3: Circuit to perform polynomial division

4. The contents of registers is the remainder of the division.

5.3 Simultaneous polynomial division and multiplication

Figure 4 shows a circuit that computes

$$b(x) = a(x)\frac{h(x)}{g(x)}$$

where

$$\frac{h(x)}{g(x)} = \frac{h_0 + h_1x + \ldots + h_rx^r}{g_0 + g_1x + \ldots + g_rx^r}$$

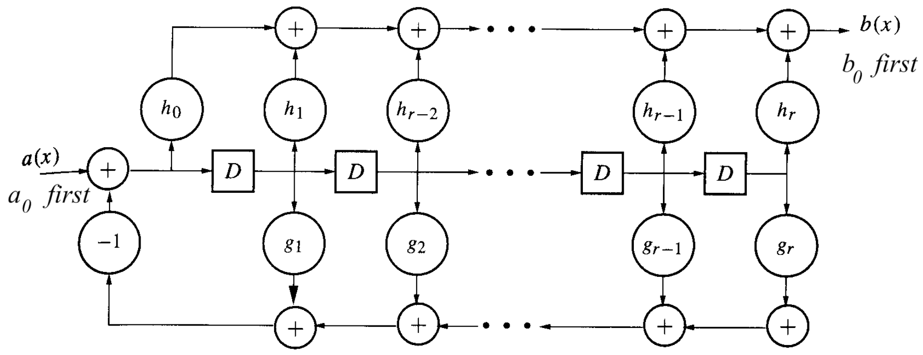

Figure 4: Circuit to perform $a(x)\frac{h(x)}{g(x)}$

This circuit is used in the implementation of recursive convolutional codes.

5.4 Cyclic encoder circuit

Using the same approach, the circuit of cycling encoder is not too complicated to guess. The Figure 5 presents the circuit. The message is $m(x) = m_0 + m_1x +$

$... + m_{k-1}x^{k-1}$. First with the gate open (allowing the signal to pass through) and the switch in position A, the message is shifted in with k clocks (m_{k-1} first) while at the same time it is also fed to the output (systematic part of the codeword). Then the gate is closed (its output is zero) and the switch is moved to the position B. There are then $n - k$ more clocks where the party symbols are shifted out following the systematic symbols.

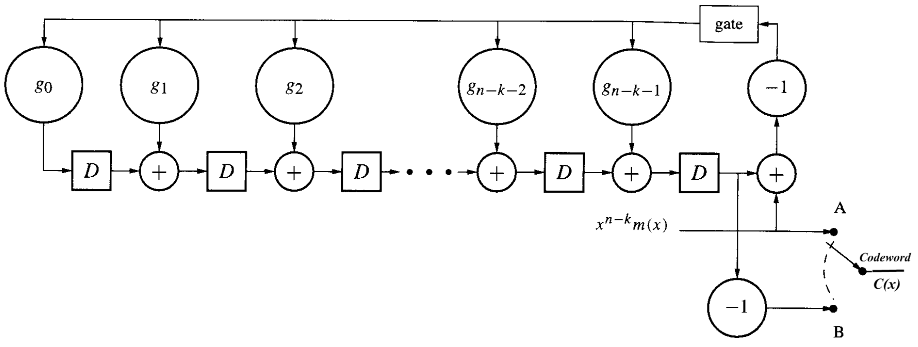

Figure 5: Circuit to perform systematic cyclic coding

5.5 Decoding circuit

The decoding circuit is the same as Figure 3 where the received sequence is divided by generator polynomial and the quotient is through away. The remainder however is the syndrome. If just the error detection is desired, if the syndrom is all zero, no transmission error is declared. If an error correction is required, the syndrom is used to find the error position.

6 Binary BCH codes

The Bose, Chaudhuri, and Hocquenghem (BCH) codes form a class of binary cyclic codes. In fact they are a generalization of Hamming code for multiple error correction. The inventors produced a table that gives the generator polynomial for different packet size and error correction power of the code. The table 1 shows these codes for small packets. For example for the code $(7, 4)$ the $g(x)$ is represented by octal number $13 = (1011)_8$. This means that the $x(x) = x^3 + 0x^2 + x + 1$.

7 Bit error rate performance

In this section we present the improvement of bit error rate performance using cyclic codes. First we consider an uncoded BPSK system. The constellation for

n	k	t	g(x)
7	4	1	13
15	11	1	23
	7	2	721
	5	3	2467
31	26	1	45
	21	2	3551
	16	3	107657
	11	5	5423325
	6	7	313365047
63	57	1	103
	51	2	12471
	45	3	1701317
	39	4	166623567
	36	5	1033500423
	30	6	157464165547
	24	7	17323260404441
	18	10	1363026512351725
	16	11	6331141367235453
	10	13	472622305527250155
	7	15	5231045543503271737

Table 1: BCH code generator polynomial

this code is presented in Figure 6. The bit error probability for BPSK signaling is:

$$P_b = Q\left(\sqrt{\frac{d_{min}^2}{2N_0}}\right) = Q\left(\frac{d_{min}}{\sqrt{2N_0}}\right) = Q\left(\sqrt{2\gamma_b}\right)$$

where:

$$\gamma_b := \frac{E_b}{N_0} = \frac{A^2/2B}{2N_0 B} = \frac{A^2}{4N_0}$$

and $Q(x)$ is defined as follows:

$$Q(x) = \frac{1}{\sqrt{2\pi}} \int_x^\infty e^{-\frac{x^2}{2}} dx$$

This BER is drawn on Figure 8. Now we use a BCH (7,4) channel coding. So each packet of 4 bits is converted to a sequence of 7 bits. Clearly, with the same constellation we have the same bit error rate over the coded bits. After correction, since some errors will be corrected, the information bit error rate will be less than the coded bit error rate. However, we consum more energy per information bit. In order hat the comparison be fair, we reduce the transmission power to have the same energy per information bit. The coding rate is $k/n = 4/7$. So we reduce the power per coded bit to the same amount. This is equivalent to use the BPSK constellation for coded bits represented in Figure 7.

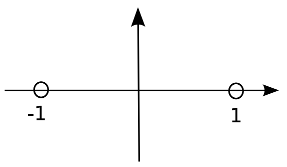

Figure 6: BPSK constellation

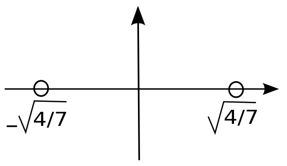

Figure 7: BPSK constellation for coded bits

The coded bit error probability is also drawn in Figure 8. It is clear that the coding scheme should not only compensate for this degradation, but also give some gain. The Figure 8 shows the results for (7,4), (15,11) and (31,26) BCH codes. In the following a matlab program is given to evaluate the BER performance with Monte-carlo simulation.

```
clear all;
n=15;k=7; block_size=2000;
EbN0_dB=[0 2 4 6 8];
EbN0=10.^(EbN0_dB/10);
noise_var=0.5./((k/n).*EbN0);
h=modem.pskmod(2);g=modem.pskdemod(2);
indice=0;
for var=noise_var
  err_sum=0;iter=0;
  while(iter<100 & err_sum < 50)
    bits_mat= randint(block_size,k);
    code_mat=bchenc(gf(bits_mat),n,k);
    rx=modulate(h,double(code_mat.x)) + ...
                           sqrt(var)*randn(block_size,n);
    rx_decide=demodulate(g,rx);
    bits_mat_hat = bchdec(gf(rx_decide),n,k);
    err_sum = err_sum + sum(sum(xor(bits_mat,bits_mat_hat.x)))
    iter=iter+1
  end
```

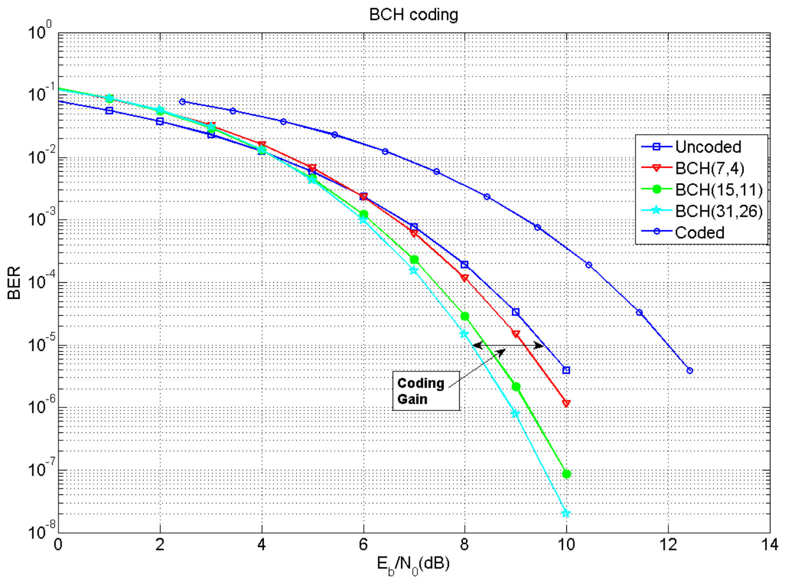

Figure 8: BER curves for BCH coding

```
  indice=indice+1;
  BER(indice)=err_sum/(block_size*k*iter);
end
semilogy(EbN0_dB,BER);grid;
title('BCH code ');
xlabel('E_b/N_0(dB)');
ylabel('BER');
```