ERROR-CORRECTING CODES – WEEK 3

Evaluation of the syndromes of received messages provides the error-vectors directly – so correction to the transmitted codewords is immediate. But it is not practicable to maintain an enormous table (e.g. 2^32 entries) of syndromes and corresponding errors in most systems. Instead, codes with more structure than simple linearity are used, enabling calculation of errors from syndromes, rather than crude table look-up. Most used are cyclic codes.

Cyclic Codes

Vectors are envisaged as the coefficients of polynomials in x over GF(q). A **Cyclic Code is an ideal** in the ring of polynomials modulo (x^n – 1). A codeword c(x) has the form c[n-1],c[n-2]......c[1],c[0]   and is manipulated as c[n=1]x^(n-1)+c[n-2]x^(n-2)+ . . . . +c[1] x+c[0]. An ideal is an additive sub-group with the extra property that the product of any member of the ideal with any member of the ring produces another member of the ideal. Therefore a left-rotation of a codeword c(x) (multiplication by x) produces another codeword c'(x) = x.c(x) mod(x^n – 1).

Theorem: **A cyclic code has a generating polynomial g(x) which divides all codewords, and also (x^n – 1)**. Proof: Let g(x) be the codeword of least degree – conventionally (n-k) – and let c(x) be any other codeword . Then c(x) = q(x).g(x) + r(x), where q(x) is the quotient polynomial on dividing c(x) by g(x), and r(x) is the remainder of degree less than g(x). But by the definition of an ideal since c(x) and g(x) are members so must be r(x). But this contradicts the supposition that g(x) has least degree. Therefore r(x) is null and g(x) divides c(x). *Students to prove g(x) divides (x^n – 1).*

From the theorem, an (n,k) cyclic code could be constructed by taking user messages m(x) of degree (k-1) and multiplying them by the generating polynomial. But it is more convenient to maintain **systematic form by multiplying m(x) by x^(n-k) and appending – r(x) to it, where r(x) is the remainder on dividing m(x).x^(n-k) by g(x). This representation means that the jth row of the generator matrix of the code corresponds to [x^(n-j),-r[n-j](x)] where r[n-j](x) is the remainder on dividing by g(x).**

Students to form the systematic generator matrices for the cyclic codes which have (x^3 + x + 1) and (x^3 + x^2 +1) as generator polynomials.

The roots of g(x) and n.

Since g(x) divides all c(x) a root α of g(x) is also one of c(x). Therefore the vector of coefficients of c(x) from x^(n-1) to x^0, namely c[n-1] to c[0], is orthogonal to the vector α^(n-1) to α^0, which must lie in the null space spanned bt the Null Matrix H. *Students to show how this is so with the code defined by g(x) = x^3 + x + 1.*

The generating polynomial must divide (x^n – 1), therefore its roots are also some of those of x^n – 1, which is the same as saying that the order of a root must divide n. If g(x) is a single irreducible polynomial of degree m then n = q^m – 1 (2^m –1 over GF(2)). Thus is

$g(x) = x^3 + x + 1$ we have n=7. If g(x) is composite, then n = LCM(order of roots of the irreducible factors of g(x)).

Exercise: What are the factors of $x^7 + 1$ over GF(2)? Of $x^{15} + 1$?

Error-detection with cyclic codes

A received vector is in error if it is not a codeword. To test if it is a codeword divide it by g(x) and check that the remainder is zero.

Students to verify that this division by g(x) gives the syndrome, as previously defined.

Equivalently, if we denote the received, corrupted codeword by $c'(x) = m'(x).x^{(n-k)} - r'(x)$, we calculate $r''(x)$ = remainder on dividing $c'(x).x^{(n-k)}$ by g(x), and check if $r''(x) = r'(x)$.

Generating and checking cyclic codewords involves polynomial division and evaluation of remainders. This can be done using **Feedback Shift Registers (FBSR)**, (see Annexe 3A) or of course by software. The method is usually used with shortened cyclic codes. **A shortened cyclic code does not have the full length, n, as implied by g(x); instead, the most left-hand symbols are taken to be all zero** and the calculation of the remainder starts with the first non-zero symbol.

For a real error to go undetected it must convert the original codeword into another one, and so the error pattern must itself be that of a codeword**. The weight distribution of a code gives W[j], the number of codewords of weight = j.** For example in a (15,7) binary code with W[1]=W[15]=1, W[2]=W14]=W[3]=W[13]=W[4]=W[12]=W[5]=W[11]=0, W[5] =W[10]=18, W[6]=W[9]=30, and W[7]=W[8]=15 there are only 15 7-bit codewords out of 15C7 = 6435 possible 7-bit error patterns. Thus 6420/6435 = 99.77% of all 7-bit error patterns will be detected.

Exercise: What is the weight distribution of the (15,5) binary code with $g(x)=x^{10}+x^{8}+x^{5}+x^{4}+x^{2}+x+1$?

Error-Correction using Cyclic Codes (Kasami's Method)

Provided that all j<= t errors fall within a range of (n-k) bits, this method works. Suppose r <= j error bits occur in the checksum (n-k bits) part of the received vector, and (j-r) in the message (k bits) part. These latter contribute at least (d – (j-r)) bits to the syndrome/remainder (***Why?***), and will at most cancel out the other r bits, thus leaving (d-j) >= (t+1) bits set. But if all j error bits fall within the checksum part then the syndrome/remainder will have <= t bits. So the method is: Look at the syndrome, if it has <= t bits then this is the error pattern. If it hasn't, rotate the received vector one bit and recalculate the syndrome and see if it has <= t bits. If it hasn't rotate and repeat until a syndrome with <= t bits is found. Then correct the rotated received vector, and rotate backwards to get to the original position.

Example with (15,7) distance-5 code with generating polynomial $x^8 + x^7 + x^6 + x^4 + 1$ (111010001). We suppose the received vector is (111110010000001). The sequence of syndromes is: 11011100, 01101001, 11010010, 01110101, 11101010, 00000101. We add this back into the received vector rotated five times 001000000111111 to get 001000000111010, and rotate back to get the corrected vector 110100010000001.

Exercise: Correct 010001000000100 from same code.

One can easily construct non-binary cyclic codes.

Exercise: Construct a cyclic code over GF(3) with generating polynomial ($x^2 + x + 2$). What is its length? What is its distance?

BCH Codes

We can fix the distance for a cyclic code if we make it a BCH code, with the generating polynomial composed of a product of m minimum polynomials $p[i](x)$ with consecutive roots. A minimum polynomial is irreducible and has lowest degree. We start with a root α and its minimum polynomial $p[1](x)$. In a binary system (characteristic = 2) α^2, α^4, α^8 etc are also roots of $p[1](x)$. *Why?* So the next consecutive root is α^3, α^6 etc and its $p[2](x)$. Then α^5, α^{10} etc and $p[3](x)$, giving $g(x) = \text{Product } (1 \text{ to } m)\ p[i](x)$, with at least 2m consecutive roots. The degree of $g(x) = (n-k) <= m.r$ where r is the degree of $p[1](x)$. And subsequent $p[i](x)$ all have degrees $<=r$ (*Why?*).

It can be shown (see Annexe 3B) that the distance d>=2m+1, therefore t >= (n-k)/r. Essentially this is because the rows of the Null Matrix are of the form $[(\alpha^j)^k]$ where α^j is the first root for that jth row and minimum polynomial and k is the column number from $(2^m - 1)$ to 0. Remember the distance is the minimum number of linearly independent columns of H.

Example: $p[1](x) = x^4 + x + 1$ roots $\alpha, \alpha^2, \alpha^4, \alpha^8$; $p[2](x) = x^4 + x^3 + x^2 + x + 1$ roots $\alpha^3, \alpha^6, \alpha^{12}, \alpha^9$; $p[3](x) = x^2 + x + 1$ roots α^5, α^{10}; so the product $g(x)$ has degree = 10 with six consecutive roots ($\alpha \ldots \alpha^6$), and gives a (15,5) code with d=7,

One can readily find minimum polynomials of powers α^j of a basic α, but tables exist which help (see Annexe 3C)

Exercise: Construct a (31,6) code with $p[1](x) = x^5 + x^2 + 1$. What is its distance?

Error-correction with BCH Codes

We designate the location of the <=t correctable errors at $\alpha^{j1}, \alpha^{j2}, \ldots \alpha^{jk}, \ldots \alpha^{jt}$ by error locators = $X[k]$. They correspond to the first row of H and give rise to syndrome $S[1] = \text{Sum } X[k]$, for k= 1 to t. Because $(\alpha^i)^{jk} = (\alpha^{jk})^i = X[k]^i$ syndromes from all the relevant rows may be written as $S[i] = \text{Sum(over k)}\ X[k]^i$, for i = 1 to m, the number of consecutive roots.

Suppose the error locators X[k] are roots of $f(x) = x^t + f1.x^{(t-1)} + f2.x^{(t-2)} + \ldots ft$, then:

$X[k]^t + f1.X[k]^{(t-1)} + f2.X[k]^{(t-2)} + .. ft = 0$. Multiply by $X[k]^j$ and sum j= 1 to t, and get:

$S[t+j] + f1.S[t+j-1] + f2.S[t+j-2] \ \ldots + ft.S[j] = 0$ for j = 1 to t. These are t linear equations in t unknowns (f1 to ft) which may be solved because the 2t syndromes are known.

Having found the coefficients f1 to ft, we may find the roots of f(x), which are the error locators, and then make the corrections in those locations. Summarising:

1) **Find the m=2t syndromes corresponding to the consecutive roots.**
2) **Solve the linear system of t equations to find the coefficients of f(x)**
3) **Find the roots of f(x), the error locators.**
4) **From the X[k] identify the columns of H where errors occurred and correct them.**

Example: We solve the previous (Kasami) problem with the full method. The generating polynomial is the product of x^4+x+1 and $x^4+x^3+x^2+x+1$ giving four consecutive roots. For the calculations we need representation of the powers α^i, for i=0 to 14. To the base $(\alpha^3,\alpha^2,\alpha,1)$ we have i:xxxx thus 0:0001, 1:0010, 2:0100, 3:1000, 4:0011, 5:0110, 6:1100, 7:1011, 8:0101, 9:1010, 10:0111, 11:1110, 12:1111, 13:1101, 14:1001, 15:0001.

From (010001000000100) we find $S1= \alpha^{13} +\alpha^9 +\alpha^2$ = 1101+1010+0100 = 0011 = α^4. $S2=S1^2=\alpha^8$. $S3=\alpha^6 +\alpha^{12} +\alpha^9$ = 1100+1111+1919=1001= α^{14}. $S4=S2^2=\alpha$.

To find the f(x) whose roots are the error locators we solve:

$S3 +f1.S2 + f2.S1 = 0$
$S4 +f1.S3 + f2.S2 = 0$ and get $f1= \alpha^4$, $f2= \alpha$

So the roots of $f(x) = x^2 +\alpha^4.x +\alpha =0$ are x=1 and $x=\alpha$, and these are the error locators. Therefore the error vector is 000000000000011 and the corrected codeword is 010001000000111 (which is obviously correct since it is g(x) rotated somewhat).