M836 A First Course in Coding Theory

Ryan May

18/10/2018

Unit 1

Abelian Group is any commutative group i.e $(a * b) * c = a * (b * c)$

Order of Groups $\mathbb{Z}_{18}$ is a cyclic group of order 18. The possible orders of subgroups are any d—n. That is from $\mathbb{Z}_{18} 1, 2, 3, 6, 9, 18$.
The subgroups are a closed group via addition. For example $0 + 0 = 0$, this is a closed subgroup of $\mathbb{Z}_{18}$ with order 1.

Subgroup of order 1: $\{0\}$

Subgroup of order 2: $\{0, 9\}$

Subgroup of order 3: $\{0, 6, 9\}$

Subgroup of order 6: $\{0, 3, 6, 9, 12, 15\}$

Subgroup of order 9: $\{0, 2, 4, 6, 8, 10, 12, 14, 16\}$

Subgroup of order 18: $\{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17\} = \mathbb{Z}_{18}$

Cosets of subgroups are closed subgroups of the same order. For example:
Cosets of the subgroup of order 3 are $\{0, 6, 12\}, \{1, 7, 13\}, \{2, 8, 14\}, \{3, 9, 15\}, \{4, 10, 16\}, \{5, 11, 17\}$

Code Notation either $[n, k, d]$ or $[n, q^k, d]$ where n is the code length, k is the dimension and d is the Hamming distance.
The code length is how many bits there are within the code e.g. 0110101, n=7
The dimension k refers to how many codewords for the basis of the block of code. The basis being the smallest amount of codewords that can be used to make up any other codeword within the block. the Hamming distance d is the minimum number of bits that differ between the codewords e.g 01000 and 10000 differ by 2 bits therefore $d = 2$
q^k which refers to the number of codewords. q is the number of elements, so a binary code has 2 elements $(0, 1)$ thus $q = 2$. A ternary code $q = 3$ has elements $(0, 1, 2)$.
A $[n, k, d]$ is a linear code and is also a $[n, q^k, d]$ code, but a $[n, q^k, d]$ code may not be a $[n, k, d]$ code nor linear.
$A_q(n, d)$ is the code with the largest value M such that there exists a q-ary (n, M, d)-code. $A_q(n, 1) = q^n$ and $A_q(n, n) = q$

Distance and Weight If x and $y \in (F_2)^n$ then $d(x, y) = w(x, y)$

$$d(x, y) = w(x) + w(y) - 2w(x \cap y)$$

that is the weight of x + y - twice the weights that are the same in x and y i.e. $11100 \cap 00111 = 00100$

Binomial coefficients

$$\binom{n}{m} = \frac{n!}{m!(n-m)!}$$

Sphere packing bound or Hamming bound A q-ary $(n, M, 2t+1)$-code satisfies

$$M\left\{\binom{n}{0} + \binom{n}{1}(q-1) + ... + \binom{n}{t}(q-1)^t\right\} \leq q^n$$

For binary codes this is

$$M\left\{1 + \binom{n}{1} + ... + \binom{n}{t}\right\} \leq 2^n$$

Perfect Code is a code that achieves the sphere packing bound, that is every vector $(f_q)^n$ is at distance $\leq t$ from exactly one codeword.. e.g binary repetition codes and Hadamard codes.

Unit 2

ring A set of elements with + and · satisfying the field properties is called a ring

Galois field or finite field is a field (set on which addition, subtraction, multiplication, and division are defined) that contains a finite number of elements. The most common examples of finite fields are given by the integers mod p when p is a prime number.
$GF(q)$ is a Galois field where q is a prime power, we regard $(F_q)^n$ as the vector space $V(n,q)$. If $F_q = \{a, b\}$ so $q = 2$ then $(F_q)^3 = \{aaa, aab, aba, abb, baa, bab, bba, bbb\}$

ISBN code ISBN numbers take on the form

$$\sum_{i=1}^{10} ix_i \equiv 0 \pmod{11}$$

If we were to have a transposition error where 2 digits are swapped from the i and $i+1$ place, then if

$$\sum_{i=1}^{10} ix_i \equiv e \pmod{11}$$

where e is the error, then the values for i and $i+1$ namely a and b give $a-b \equiv c \pmod{11}$

Unit 3

linear Code is defined as:

$$\mathbf{u} + \mathbf{v} \in C \text{ for all } \mathbf{u} \text{ and } \mathbf{v} \text{ in C}$$

$$a\mathbf{u} \in C \text{ for all } \mathbf{u} \text{ in C}, a \text{ in GF(q)}$$

so if you can create a generating matrix for the set of codes, then the code is linear. $\{0000, 1100, 0110, 0011, 1100\}$ has no matrix that can generate only these codes, so therefore is not linear.

Rate of a code

$$R = k/n$$

Hamming distance A Hamming distance is the distance from one code to another. The minimum Hamming distance is the smallest distance from any code to another.
With a Hamming distance of d a code is a $d-1$ detecting code, and a $\left[\frac{d-1}{2}\right]$correcting code.

equivalent codes If a generator (or parity check) matrix can be rearranged to obtain another generator (or parity check matrix) following these rules, then they are equivalent
Permutation (rearrangement) of the rows
multiplication of a row not by 0
addition of a scalar multiple of 1 row to another
permutation of the columns
multiplication of any column by a non zero scalar

Generator Matrix is a matrix from which every codeword can be created. The standard form of a Generator is:

$$G = [I_k|A] \text{ where } I_k \text{ is the Identity Matrix}$$

Parity Check Matrix is such that $\mathbf{x}H^T = 0$ if and only if $\mathbf{x} \in C$. It takes the form:

$$H = [-A^T|I_{n-k}]$$

A^T can be obtained by drawing a diagonal line from top left of A and then reflecting the values around the line. eg:

$$\begin{bmatrix} \mathbf{1} & 1 & 1 & 0 \\ 1 & \mathbf{1} & 0 & 1 \\ 1 & 0 & \mathbf{1} & 1 \end{bmatrix} \rightarrow \begin{bmatrix} \mathbf{1} & 1 & 1 \\ 1 & \mathbf{1} & 0 \\ 1 & 0 & \mathbf{1} \\ 0 & 1 & 1 \end{bmatrix}$$

Unit 4

Standard (Slepian) Arrays is an array that contains all F_q^n codes in the format $q^{n-k} \times q^k$ (columns x rows):

$codewords \rightarrow$	000	011	101	110
$cosets \rightarrow$	001	010	100	111
	$\uparrow$			
	coset	*leaders*		

The codewords are found using the codeword generator G. This forms the code C
The table is filled by adding $C + 001$, $C + 010$ etc to the codewords. The coset leaders are the codes with minimum weight. If 2 codes have equal minimum weight then pick 1 (remember that in the case above the coset leader contains 010 and 100 thus these are not separate coset leaders). Once complete every possible permutation should be within the array
If codeword x is transmitted and y is received then $x = y - e$ where e is the coset leader e.g from above 011=010-001

Error probability

$$P_{corr} = \sum_{i=0}^{n} \alpha_i p^i (1-p)^{n-i}$$

where p =probability, α = the number of coset leaders with weight i. e.g for a [4,2]-code the coset eladers are 0000, 1000, 0100 and 0010. giving

$$P_{corr} = p^0(1-p)^4 + 3p^1(1-p)^3 = (1-p)^4 + 3p(1-p)^3$$

$P_{err} = 1 - P_{corr}$

Error detection

$$P_{undetec} = \sum_{i=1}^{n} A_i p^i (1-p)^{n-i}$$

where p =probability, A = the number of codewords of C with weight i. e.g for a

$$P_{retrans} = 1 - (1-p)^n - P_{undetec}$$

Dual Code A dual code $C^\perp$ is orthogonal to C, that is $u \cdot v = 0$ eg $(1001) \cdot (1101) = 0$ The parity check matrix of C is the generator matrix of $C^\perp$ and is a linear $[n, n-k]$-code If $C = C^\perp$ then C is self-dual

syndrome decoding For any vector $\mathbf{y} \in V(n,q)$ the $1 \times (n-k)$ row vector

$$S(\mathbf{y}) = \mathbf{y}H^T$$

is called the syndrome of $\mathbf{y}$
If $H = \begin{pmatrix} 1010 \\ 1101 \end{pmatrix}$
Then $S(0000) = 00$,
S(1000)=$(1 \cdot 1 + 0 \cdot 0 + 0 \cdot 1 + 0 \cdot 0)(1 \cdot 1 + 0 \cdot 1 + 0 \cdot 0 + 0 \cdot 1) = (11)$ If two codewords have the same syndrome, they are in the same coset

If we have the syndromes of the coset leaders say:

coset leader	syndrome
0000	00
1000	11
0100	01
0010	10

then if we receive the codeword 1111, working out its syndrome, say S(1111)=01, then we know the syndrome leader is 0100 and hence the error, thus the transmitted codeword was 1111-0100=1011

Incomplete decoding If we have $d(C) = 2t+1$ or $2t+2$ we can only guarantee the correction of $\leq t$ errors, and detect some cases of more than t errors. we arrange a slepian array in order of increasing weight and divide the array into a top part compromising those cosets whose leaders have weights $\leq t$ and a bottom part compromising of the remaining cosets. If a received codeword is in the top part we can correct it otherwise we can seek retransmission

Unit 5 - Hamming Codes

Binary Hamming Codes A Hamming code is most conveniently defined by specifying its partity matrix
Let r be a positive integer and let H be an $r \times (2^r - 1)$ (rows x columns) matrix whose columns are distinct non-zero vectors of $V(r, 2)$. Then the code having H as its parity check matrix is called a binary Hamming code, denoted as $Ham(r, 2)$

The binary Hamming code $Ham(r, 2)$ for $r \geq 2$ is a $[2^r - 1, 2^r - 1 - r]$-code with minimum distance 3 and is a perfect code.

It has $2^r = n + 1$ coset leaders, that is all vectors $V(n, 2)$ of weight $le1$
Arranging the columns of H in order of increasing binary numbers then to decode we just need to calculate the syndrome of the received vector $S(\mathbf{y}) = \mathbf{y}H^T$. If the syndrome is not 0 then this gives the binary representation of the error position, assuming that 1 error occurred. e.g.

$$H = \begin{pmatrix} 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 \end{pmatrix}$$

with $\mathbf{y} = 1101011$ then $S(\mathbf{y}) = 110$ indicating the error is at the 6th position thus $\mathbf{x} = 1101011 - 0000010 = 1101001$

Extended binary Hamming Codes denoted as $H\hat{a}m(r, 2)$ is the code $Ham(r, 2)$ + a parity check. The minimum distance is increased to 4, and is still linear thus the code is a $[2^r, 2^r - 1 - r, 4]$-code. This is inferior for complete decoding like above as it has an extra digit, but is better for incomplete decoding as it can correct 1 error and detect 2 errors. The parity check matrix $\bar{H}$ can be obtained from h by adding an extra column of 0s, then an extra row of 1s to give

$$\bar{H} = \begin{bmatrix} & & & 0 \\ & & & 0 \\ & H & & : \\ & & & 0 \\ 1 & 1 & 1 & 1 \end{bmatrix}$$

For decoding suppose $S(\mathbf{y}) = (s_1, s_2, s_3, s_4)$ then
if $s_4 = 0$ and $s_1 = s_2 = s_3 = 0$ assume no errors
if $s_4 = 0$ and $(s_1, s_2, s_3) \neq 0$ assume 2 errors have occurred and seek retransmission
if $s_4 = 1$ and $s_1 = s_2 = s_3 = 0$ assume single error in the last place
if $s_4 = 1$ and $(s_1, s_2, s_3) \neq 0$ assume 1 error in the jth place

q-ary Hamming Codes Fundamental theorem: Suppose C is a linear $[n,k]$-code over $GF(q)$ with parity check H then the minimum distance of C is d iff any $d-1$ columns are linearly independent but some d columns are linearly dependent

A hamming code $Ham(r,q)$is a $[\frac{q^r-1}{q-1}, \frac{q^r-1}{q-1}-r, 3]$-code. For it to be distance 3 this means that H has 2 linearly independent columns and 3 linearly dependent columns
$Ham(r,q)$ is a perfect single-error-correcting code

Remember that r is the column size in a parity check, and q is the modulus of the code. For example a 7-ary $(8,6)$ Hamming code is a Ham$(2,7)$ code with distance 3.
If there are any d number of columns in a parity check that are linearly dependent but $d-1$ number of columns within d that are linearly independent then the distance is d.
Therefore we have a parity check matrix for Ham$(2,7)$ as:

$$\begin{pmatrix} 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 2 & 3 & 4 & 5 & 6 \end{pmatrix}$$

where c_1, c_2, c_3 are linearly dependent and c_1, c_2 are linerly independent, thus $d=3$.
Using the parity matrix to find the error syndrome $S(\mathbf{y}) = H\mathbf{y}^T$. If $S(\mathbf{y}) = 0$ then the codeword is correct otherwise $S(\mathbf{y}) = b\mathbf{H}_j^T$

Using the above parity matrix with codeword 10521360 we get $S(\mathbf{y}) = (3\ 6) = 3(1\ 2)$ giving $j=4, b=3$ therefore the forth column is incorrect and then subtract b from this column to get the correct codeword thus $2-3=6\ mod(7)$ giving the correct codeword 105**6**1360

shortening a code Suppose C is a (n,M,d)-code. If we delete the digit k from each of the codewords we get the code C' which is $[n-1,k,d']$ code in which d' is generally the same as d but in some cases may be greater. If all the codewords have the digit $k=0$ that has been deleted then we are left with an $[n-1,k-1,d']$-code

Suppose we have the codewords C where

$$G = \begin{pmatrix} 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 \end{pmatrix}$$

the codewords are:

$$\begin{array}{ccccc} 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 \\ 1 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 \end{array}$$

we can shorten the code by removing, say the 1st digit from each codeword. If the digit is 1, then the whole codeword is removed, thus giving the new codewords of C'

$$\begin{array}{cccc} 0 & 0 & 0 & 0 \end{array}$$

$$\begin{array}{cccc} 1 & 1 & 0 & 1 \\ 1 & 1 & 0 & 0 \end{array}$$

$$\begin{array}{cccc} 0 & 0 & 0 & 1 \end{array}$$

gives the same codewords as above notice that by deleting either 1st,2nd or 3rd digits gives the same $d = 1$
removing the last coordinate gives the codewords 0000,1100,1010,0110 and a minimum distance of 2
Removing the 4th digit gives the codewords 0000,1100,1010,0111,0110,1011,1101,0001
With the number of codewords $=q^k$ we see that deleting the digits 1,2,3 give an [4,2,1]-code, deleting the 5th digit gives a [4,2,2]-code and deleting the 4th digit gives an [4,3,1]-code

Simplex Code

$$Sim(r, q) = Ham(r, q)^{\perp}$$

That is the simplex code is a dual of the Hamming code.
$Sim(r, q)$ is a $\left[\frac{(q^r-1)}{(q-1)}, r, q^{r-1}\right]$-code.
The Hamming distance between any two distinct codewords is q^{r-1}, this means that each non-zero codeword has the same weight. Thus the Simplex code is known as a constant weight code.

Unit 6 BCH codes- not in exam

Unit 7- Reed Muller

Suppose $\mathbf{u} = 010$ and $\mathbf{v} = 110$ then $\mathbf{u}|\mathbf{v} = 010110$
Reed-Muller codes take on the form $(a|a + b)$ for all $\mathbf{a} \in A$ and $\mathbf{b} \in B$. This is

denoted as $A * B$

If A and B have parameters (n, M_A, d_A) and (n, M_B, d_B) then $A * B = C$ where C has parameters $(2n, M_A M_B, d_C)$ where:
$d_C = \min\{2d_A, d_B\}$
If working with linear code [n,k,d] rather than the plotkin code(n,M,d), then $M_A = q^{k_A}$ and $M_B = q^{k_B}$,giving $M_A M_B = q^{k_A} q^{k_B}$ so it has dimension $k_C = k_A + k_B$ which is the same as M_C

Reed Muller codes are in the form of $RM(r, m)$ in which

$$RM(r, m) = RM(r, m-1) * RM(r-1, m-1)$$

where

- $RM(0, m)$ is the binary linear $[2^m, 1, 2^m]$ repetition code 00...0 and 11...1 each of length 2^m and

- $RM(m, m)$ is the binary linear $[2^m, 2^m, 1]$-code corresponding to the whole of $V(2^m, 2)$ i.e all binary vectors of length 2^m

For example, find the codewords for $RM(1, 2)$ and $RM(1, 3)$:

$$RM(1, 2) = RM(1, 1) * RM(0, 1)$$

$$RM(1, 1) = [2, 2, 1] = 00, 01, 10, 11 \text{ and } RM(0, 1) = [2, 1, 2] = 00, 11$$

With $(a|a + b)$ where $a = RM(1, 1)$ and $b = RM(0, 1)$ we have:

0000	0011
0101	0110
1010	1001
1111	1100

For $RM(1, 3)$ we have $RM(1, 3) = RM(1, 2) * RM(0, 2)$ where $RM(0, 2) = [4, 1, 4] = \{0000, 1111\}$. Therefore we get the codewords:

00000000	00001111
01010101	01011010
10101010	10100101
11111111	11110000
00110011	00111100
01100110	01101001
10011001	10010110
11001100	11000011

Above we can see that a $RM(1,2)$ code consists of a $[2,2,1] * [2,1,2] = [2 \cdot 2, 2+1, \min(2 \cdot 1, 2)] = [4,3,2]$.
For a $RM(2,3)$ code we have $RM(2,3) = RM(2,2) * RM(1,2)$ in which $RM(2,2) = [4,4,1]$.
Therefore the parameters of an $RM(2,3)$ code are $[4,4,1] * [4,3,2] = [2 \cdot 4, 4+3, \min(2 \cdot 1, 2)] = [8,7,2]$code. This has $2^7 = 128$ codewords of all even weight vectors in $V(8,2)$

RM Matrix The matrix $G(r,m)$ where $r \leq m$ is as follows

1. $G(0,m) = (11...1)$ (2^m bits)

2. $G(m,m) = \begin{pmatrix} G(m-1,m) \\ 00...01 \end{pmatrix}$

3. For$0 < r < m$
   $G(r,m) = \begin{pmatrix} G(r,m-1) & G(r,m-1) \\ \mathbf{0} & G(r-1,m-1) \end{pmatrix}$

$RM(r,m)$ is a binary $[n,k,d]$-code $= [2^m, \sum_{i=0}^{r} \binom{m}{i}, 2^{m-r}] - code$

Truth Table

To find the formula for a truth table select each formula that the function equals 1. Then or them together using the fact that for each bit $v = 1, \bar{v} = 0,$ or $= \vee$ and the fact that $\bar{x} = (1+x),\ x \vee y = x + y + xy,\ xx = x,\ x + x = 0$
Ensure you write out each bracketed part fully before doing the 'or' parts
Should be in the format as:

$$f(v_1, v_2, v_3) = ((v_1 \bar{v}_2 \bar{v}_3) \vee (\bar{v}_1 v_2 v_3)) \vee (v_1 v_2 v_3)$$

The truth table has the form:

v_1	v_2	v_3	$f(v_1, v_2, v_3)$
0	0	0	
1	0	0	
0	1	0	
1	1	0	
0	0	1	
1	0	1	
0	1	1	
1	1	1	

Writing the table this way allows us to transpose it. For $m = 4$ we get

1	:	1111111111111111
v_1	:	0101010101010101
v_2	:	0011001100110011
v_3	:	0000111100001111
v_4	:	0000000011111111
v_1v_2	:	..
v_1v_3	:	..
v_1v_4	:	..
v_2v_3	:	..
v_2v_4	:	..
v_3v_4	:	..
$v_1v_2v_3$	:	..
$v_1v_2v_4$	:	..
$v_1v_3v_4$	:	..
$v_2v_3v_4$	:	..
$v_1v_2v_3v_4$	:	..

Now the order r is as follows

$order(r)$	$rows$
0	1
1	$1-5$
2	$1-11$
3	$1-15$
4	$1-16$

Thus the generator matrix for $RM(1,4)$ is

$$\begin{pmatrix} 1111111111111111 \\ 0101010101010101 \\ 0011001100110011 \\ 0000111100001111 \\ 0000000011111111 \end{pmatrix}$$

Decoding RM(r,m)

To recode, we notice that to encode a message we use $\mathbf{aG} = \mathbf{x}$, where $\mathbf{a} = a_0a_1a_2a_3a_4a_{1,2}a_{1,3}a_{1,4}a_{2,3}a_{3,4}$ in the case for $RM(2,4)$.
This will give $\mathbf{aG} = \mathbf{x} = x_0x_1....x_{15}$ (16 bits) For x_0 we take the 1st column of

the table and add the corresponding bits that are equal to one

			x_0.....x_{15}
a_0	1	:	1111111111111111
a_1	v_1	:	0101010101010101
a_2	v_2	:	0011001100110011
.	v_3	:	0000111100001111
.	v_4	:	0000000011111111
$a_{1,2}$	v_1v_2	:	0001000100010001
.	v_1v_3	:	0000010100000101
.	v_1v_4	:	0000000001010101
.	v_2v_3	:	0000001100000011
.	v_2v_4	:	0000000000110011
.	v_3v_4	:	0000000000001111
$a_{1,2,3}$	$v_1v_2v_3$	:	0000000100000001
.	$v_1v_2v_4$	:	0000000000010001
.	$v_1v_3v_4$	:	0000000000000101
.	$v_2v_3v_4$	:	0000000000000011
.	$v_1v_2v_3v_4$	:	0000000000000001

so for the values within the table we see that

$$
\begin{aligned}
x_0 &= a_0 \\
x_1 &= a_0 + a_1 \\
x_2 &= a_0 + a_2 \\
x_3 &= a_0 + a_1 + a_2 + a_{1,2} \\
x_4 &= a_0 + a_3 \\
x_5 &= a_0 + a_1 + a_3 + a_{1,3} \\
x_6 &= a_0 + a_2 + a_3 + a_{2,3} \\
x_7 &= a_0 + a_1 + a_2 + a_3 + a_{1,2} + a_{1,3} + a_{2,3} \\
x_8 &= a_0 + a_4 \\
x_9 &= a_0 + a_1 + a_4 + a_{1,4} \\
x_{10} &= a_0 + a_2 + a_{2,4} \\
x_{11} &= a_0 + a_1 + a_2 + a_4 + a_{1,2} + a_{1,4} + a_{2,4} \\
x_{12} &= a_0 + a_3 + a_4 + a_{3,4} \\
x_{13} &= a_0 + a_1 + a_3 + a_4 + a_{1,3} + a_{1,4} + a_{3,4} \\
x_{14} &= a_0 + a_2 + a_3 + a_4 + a_{2,3} + a_{2,4} + a_{3,4} \\
x_{15} &= a_0 + a_1 + a_2 + a_3 + a_4 + a_{1,2} + a_{1,3} + a_{1,4} + a_{2,3} + a_{2,4} + a_{3,4}
\end{aligned}
$$

To find the values for $a_{i,j}$ we need to rearrange each of the above, going from highest value to lowest remembering that for $\pmod 2$ $+ = -$ and $a_i + a_i = 0$.

For instance:

$$\begin{aligned} a_{1,3} =& x_5 - a_3 + a_1 - a_0 \\ =& x_5 + a_3 + a_1 + a_0 \\ =& x_5 + (x_4 + a_0) + a_1 + a_0 \\ =& x_5 + x_4 + a_1 \\ =& x_5 + x_4 + x_1 + a_0 \\ =& x_5 + x_4 + x_1 + x_0 \end{aligned}$$

$$\begin{aligned} a_{1,3} =& x_7 + a_{2,3} + a_{1,2} + a_3 + a_2 + a_1 + a_0 \\ =& x_7 + (x_6 + a_3 + a_2 + a_0) + (x_3 + a_2 + a_1 + a_0) + a_3 + a_2 + a_1 + a_0 \\ =& x_7 + x_6 + x_3 + a_2 + a_0 \\ =& x_7 + x_6 + x_3 + x_2 \end{aligned}$$

...and so on

To decode a message $M_e = x0...x_n$, we first note what each x_i is and then plug these numbers into $a_{i,j}$. We can then use majority vote to find the values of $a_{i,j}$. For instance:-

$$a_{1,2} = x_0 + x_1 + x_2 + x_3 = 0 + 1 + 0 + 0 = 1$$

$$a_{1,2} = x_4 + x_5 + x_6 + x_7 = 1 + 1 + 0 + 0 = 0$$

$$a_{1,2} = x_8 + x_9 + x_10 + x_{11} = 1 + 0 + 0 + 1 = 0$$

$$a_{1,2} = x_{12} + x_{13} + x_{14} + x_{15} = 0 + 1 + 1 + 0 = 0$$

Thus $a_{1,2} = 1, 0, 0, 0 = 0$ majority vote. Plugging these values in to x'_1 to x'_{15} gives us a set of formulas we can transpose to find a_1 to a_4, once again using majority vote.

$$\begin{aligned} x'_0 &= a_0 = x_0 = 0 \\ x'_1 &= a_0 + a_1 = x_1 = 1 \\ x'_2 &= a_0 + a_2 = x_2 = 0 \\ x'_3 &= a_0 + a_1 + a_2 = x_2 + a_{1,2} = 0 + 0 = 0 \\ &\text{etc} \end{aligned}$$

transposing

$$a_1 = x'_1 + x'_0 = 1 + 0$$

$$a_1 = x'_2 + x'_3 = 0 + 0$$

etc
Now we know $x'_0..x_n$ and $a_1...a_4$ we can plug these values in to find the 16 values

for a_0

$$x'_0 = a_0 = x_0 = 0$$
$$x'_1 = a_0 + a_1 = 1$$
$$x'_2 = a_0 + a_2 = 0$$
$$x'_3 = a_0 + a_1 + a_2 = 0$$

If the majority vote is say 0 and $x_6 = 1$ we know that the value x_6 is incorrect and can be corrected (following normal rules on number of identifications and corrections)

Unit 8 - Cyclic Codes

A cyclic code must be linear.
If there codeword $x_0, x_1, ..., x_n - 1, x_n$ in the set of codewords then $x_1, ...x_n, x_0$ must also be a codeword within the set. A code that is not a cyclic code can be equivalent to a cyclic code, for instance

$$0000, 1122, 2211$$

is not cyclic, but changing the 2nd and 3rd values around gives

$$0000, 1212, 2121$$

which is cyclic, so therefore the code is equivalent to a cyclic code.

A repetition code, as long as its linear, will be cyclic

A binary even-weight code will be cyclic as the code will contain every even weight codewords and if you cycle a codeword it will be another even weight codeword, which of course will be within the code

A ternary code $\{\mathbf{x} \in V(n,3) | w(\mathbf{x}) \equiv 0 \pmod 3\}$ can contain $\mathbf{x} = 1110...0$ and $\mathbf{y} = 2210...0$ but $\mathbf{x} + \mathbf{y} = 0020...0$ in which $w(\mathbf{x} + \mathbf{y}) = 1$ so therefore is not linear we denote $F[x]$ as the set of polynomials in x with coefficients in F, If $f(x) = f_0 + f_1 x + ... + f_m x^m$ where $f_m \neq 0$ then m is called the degree of $f(x)$ Also note that $\deg(f(x)g(x)) = \deg f(x) + \deg g(x)$

If we calculate $(x+1)^2$ in $F[x]/(x^2 + x + 1)$ we have $(x+1)^2 = x^2 + 2x + 1 \equiv x \pmod{x^2 + x + 1}$

$$|F_q[x]/f(x)| = q^n$$

Reducibility A polynomial $f(x)$ in $F[x]$ is said to be reducible if $f(x) = a(x)b(x)$. If not $f(x)$ is said to be irreducible.

1. A polynomial $f(x)$ has linear factor $x - a$ iff $f(a) = 0$

2. $f(x)$ of degree 2 or 3 is irreducible iff $f(a) \neq 0$ for all a in F

3. over any field $x^n - 1 = (x - 1)(x^{n-1} + x^{n-2} + ... + x + 1)$

For (1), take $x^4 + 2x^3 + 3x + 1$ in $\mathbb{Z}_5$, that is $\mathbb{Z}_5 = \{0, 1, 2, 3, 4\}$. Check to see if substuting x with the set of number equates to 0.
$f(0) = 0 + 0 + 0 + 1 \neq 0$ but $f(3) = 3^4 + 2 \cdot 3^3 + 3 \cdot 3 + 1 = 145 \equiv 0 \pmod 5$ thus we have a root $a = 3$ giving a linear factor of $(x - 3)$. If i[t doesnt have a root, this does not mean its not irreducible!

For (2), we do the same above but, if the degree is 2 or 3, then this does mean it is irreducible.

The ring $F[x]/f(x)$ is a field iff $f(x)$ is irreducible in $F[x]$

for $F[x]/(x^n - 1)$ we write it as R_n

For a vector

$$a(x) = a_0 + a_1x + ... + a_{n-1}x^n$$

multiplying by x gives

$$xa(x) = a_{n-1} + a_0x + a_1x^2 + ...a_{n-2}x^n - 1$$

therefore shifting the vector once to the right

A code in R_n is cyclic iff C satisfies

1. $a(x), b(x) \in C \Rightarrow a(x) + b(x) \in C$

2. $a(x) \in C, r(x) \in R_n \Rightarrow r(x)a(x) \in C$

Example:

Consider $C =< 1 + x^2 >$ in R_3 with $F = GF(2)$. we have

$$\begin{aligned} R_3 &= 0 \\ &= 1 \\ &= x \\ &= 1 + x \\ &= x^2 \\ &= 1 + x^2 \\ &= x + x^2 \\ &= 1 + x + x^2 \end{aligned}$$

multiplying by C gives

$$\begin{aligned} (1 + x^2) \cdot 0 &= 0 \\ (1 + x^2) \cdot 1 &= 1 + x^2 \\ (1 + x^2) \cdot x &= x + x^3 = 1 + x \\ (1 + x^2) \cdot (1 + x) &= 1 + x + x^2 + x^3 = x + x^2 \\ (1 + x^2) \cdot x^2 &= x^2 + x^4 = x + x^2 \\ (1 + x^2) \cdot (1 + x^2) &= 1 + 2x^2 + x^4 = 1 + x \\ (1 + x^2) \cdot (x + x^2) &= x + x^2 + x^3 + x^4 = 1 + x \\ (1 + x^2) \cdot (1 + xx^2) &= 1 + x + 2x^2 + x^3 + x^4 = 0 \end{aligned}$$

where $x^3 = 1$ and $x^4 = x$ giving the codewords $\{0, 1 + x, 1 + x^2, x + x^2\}$ Thus C is the code $\{000, 110, 101, 011\}$

Constructing Codewords Suppose $f(x) = g(x)h(x)$ where $g(x)$ is of the least degree, then $g(x)$ the generator polynomial. For instance $x^3 - 1 = (x + 1)(x^2 + x + 1)$

Generator Polynomial	Code in R_3	Corresponding Code Code in $V(3, 2)$
1	all of R_3	all of $V(3, 2)$
$x + 1$	$\{0, 1 + x, 1 + x^2, x + x^2\}$	$\{000, 110, 101, 011\}$
$x^2 + x + 1$	$\{0, 1 + x + x^2\}$	$\{000, 111\}$
$x^3 - 1 = 0$	$\{0\}$	$\{000\}$

Suppose $g(x) = g_0 + g_1x + ... + g_rx^r$ then $dim(C) = n - r$ and

$$G = \begin{pmatrix} g_0 & g_1 & ... & g_r & 0 & 0 & ... & 0 \\ 0 & g_0 & g_1 & ... & g_r & 0 & ... & 0 \\ ... & & & & & & & \\ 0 & ... & 0 & 0 & g_0 & g_1 & ... & g_r \end{pmatrix}$$

For $x^4 = (x-1)(x+1)(x^2+1)$ the generator polynomials are:
$1, (x-1), (x+1), (x^2+1), (x-1)(x+1), (x-1)(x^2+1), (x+1)(x^2+1), x^4-1 = 0$

Check polynomial and parity matrix With $f(x) = g(x)h(x)$, $h(x)$ is known as the check polynomial

Suppose$C$ is a cyclic code in R_n then an element of $c(x)$ is a codeword of C iff $c(x)h(x) = 0$ As $g(x)$ has $dim(n-k)$ $h(x)$ therefore has $dim(k)$
With $h(x) = h_0 + h_1x + ... + h_kx^k$

$$G = \begin{pmatrix} h_k & h_{k-1} & ... & h_0 & 0 & 0 & ... & 0 \\ 0 & h_k & h_{k-1} & ... & h_0 & 0 & ... & 0 \\ ... & & & & & & & \\ 0 & ... & 0 & 0 & h_k & h_{k-1} & ... & h_0 \end{pmatrix}$$

NOTICE THAT THE VALUES ARE THE OTHER WAY AROUND WHEN CREATING THE PARITY MATRIX COMPARED TO THE GENERATOR MATRIX

$C^\perp$ is a cyclic code generated by $\bar{h}(x) = h_k + h_{k-1} + ... + h_0x^k$ that is, the first line of the parity matrix with the lagging 0s removed.

Unit 9 - Codes and Latin Squares

A Latin square of order q is a $q \times q$ array whose entries are from a set F_q. So $F_3 = \{1, 2, 3\}$ and a Latin square of order 3 is:

$$\begin{array}{ccc} 1 & 2 & 3 \\ 2 & 3 & 1 \\ 3 & 1 & 2 \end{array}$$

Mutually orthogonal Latin squares (MOLS) are when 2 latin squares can be combined to give ordered distinct pairs in each entry e.g

$$A = \begin{array}{ccc} 1 & 2 & 3 \\ 2 & 3 & 1 \\ 3 & 1 & 2 \end{array} \quad B = \begin{array}{ccc} 1 & 2 & 3 \\ 3 & 1 & 2 \\ 2 & 3 & 1 \end{array}$$

then A and B gives

$$\begin{array}{ccc} (1,1) & (2,2) & (3,3) \\ (2,3) & (3,1) & (1,2) \\ (3,2) & (1,3) & (2,1) \end{array}$$

10.5 $\quad A_q(4,3) \leq q^2$ for all q^2

where $A_q(n,d)$ is the largest value of M such that there exists a q-ary (n, M, d)-code, where M is the number of codewords

10.7 There exists a q-ary $(4, q^2, 3)$-code if and only if there exists a pair of MOLS of order q

10.8 If q is a prime power and $q \neq 2$ then there exists a pair of MOLS of order q.

10.14 $A_q(4,3) = q^2$ for all $q \neq 2, 6$

If there exists a pair of MOLS of order m and a pair of MOLS of order n then there exists a pair of MOLS of order mn
If m is order 3 with the MOLS $A_1, A_2$and n is order 4 of the MOLS B_2, B_2 then

we have the MOLS

$C_1 =$

$A_{1(0,0)}B_{1(0,0)}$ $A_{1(0,0)}B_{1(1,0)}$ $A_{1(0,0)}B_{1(2,0)}$ $A_{1(0,0)}B_{1(3,0)}$	$A_{1(0,0)}B_{1(0,1)}$ ...	$A_{1(0,1)}B_{1(0,0)}$ ...	$A_{1(0,2)}B_{1(0,0)}$...
$A_{1(1,0)}B_{1(0,0)}$ $A_{1(1,0)}B_{1(1,0)}$ $A_{1(1,0)}B_{1(2,0)}$ $A_{1(1,0)}B_{1(3,0)}$			
$A_{1(2,0)}B_{1(0,0)}$ $A_{1(2,0)}B_{1(1,0)}$ $A_{1(2,0)}B_{1(2,0)}$ $A_{1(2,0)}B_{1(3,0)}$			

Giving a

4×4	4×4	4×4
4×4	4×4	4×4
4×4	4×4	4×4

matrix. C_2 is constructed the same way with A_2B_2

Constructing a pair of MOLS

Creating a pair of MOLS or order 3 means that we are in GF(3), that is 0,1,2
We use the following equations:

$$a_{ij} = \lambda_i + \mu\lambda_j$$

$$a_{ij} = \lambda_i + \nu\lambda_j$$

where i,j is the row and column in the MOL respectively. Therefore $\lambda_0 = 0, \lambda_1 = 1$ etc. Thus $\lambda_i = i$ and $\lambda_j = j$
μ and ν are any values within $GF(2) \neq 0$, thus we can say $\mu = 1$ and $\nu = 2$. This gives us the equations:

$$a_{ij} = i + j$$

$$a_{ij} = i + 2j$$

With $A = a_{ij}$ and $B = b_{ij} \pmod 3$ we get

$A =$

			j	
0	1	2		
0	1	2	0	
1	2	0	1	i
2	0	1	2	

$$B = \begin{array}{ccc|c|c} & j & & & \\ 0 & 1 & 2 & & \\ \hline 0 & 1 & 2 & 0 & \\ 1 & 0 & 2 & 2 & i \\ 2 & 1 & 0 & 1 & \end{array}$$

where AB are MOL

row complete A Latin square of order q is said to be row-complete if every ordered pair of distinct symbols of the Latin square occurs in adjacent position precisely once. Like wise for Column complete. e.g

$$\begin{pmatrix} 0 & 1 & 2 & 3 \\ 2 & 0 & 3 & 1 \\ 3 & 2 & 1 & 0 \\ 1 & 3 & 0 & 2 \end{pmatrix}$$

is row complete, i.e (01) only exists in the 1st row and so on, but it is not column complete as (02) exists in the 1st column and 4th column.

If q is even then a complete Latin square can be constructed by setting $q = 2s$ Let 0th row and 0th column be:

$$0, 1, 2s-1, 2, 2s-2, 3 ... s-1, s+1, s$$

The other entries are defined as

$$a_{i,j} = a_{i,0} + a_{o,j} \pmod 2$$

This is a q-ary $(4, q^2 - q, 3)$-code.

Unit 10 - Codes and Block Designs

v	points, number of elements of X
b	number of blocks
r	number of blocks containing a given point
k	number of points in a block
λ	number of blocks containing any 2 (or more generally t) distinct points

a Block design is of $t-(v, k, \lambda)$, in which t-element subset of V occurs precisely λ times. k is the number of elements within a block and v is the number of elements in V

Thus a 2-(6,3,2) block design will have 6 points ABCDEF arranged in block of 3-element subsets of V in which $t = 2$ elements will occur $\lambda = 2$ times in the set.
ABF, ACF, BDF, CEF, DEF, ABE, ACD, ADE, BCD, BCE
A design in where blocks arent repeated (like the above) are known as *simple designs*

For instance CE occurs twice, in blocks 4 and 10.

A $t-(v,k,\lambda)$ can also be written as $S_\lambda(t,k,v)$

A $t-(v,k,\lambda)$ exists only if $\binom{k}{t}$ divides $\lambda\binom{v}{t}$

r is replication number, that is the number of blocks a given element is within. e.g. the above system has the point A in 5 blocks so $r=5$

$$r = \lambda\binom{v-1}{t-1} / \binom{k-1}{t-1}$$

Admissibility conditions
Necessary conditions for the existence of a $t-(v,k,\lambda)$ block design are that $\binom{k-i}{t-i}$ divides $\lambda\binom{v-i}{t-i}$ for $i=0,1,2,...,t-1$

A block design with $t=2$ and $k=3$ is called a *triple system*. If $\lambda=1$ it is a *Steiner triple system* which is denoted as $S(2,3,v)$ where $v \equiv 1$ or $3 \pmod 6$

$$bk = vr$$

An incidence matrix can be formed from the blocks where the rows are the points, and the columns the blocks in which 1 denotes that the point is within that block, a 0 otherwise.

$$\begin{pmatrix} 1 & 1 & 0 \\ 0 & 1 & 1 \\ 1 & 0 & 1 \end{pmatrix}$$

this denoted that the blocks are AC AB BC

Balanced design

A balanced block design is when $t=2$. Any $t-(v,k,\lambda)$ design with $t \geq 2$ can be regard as $2-(v,k,\mu)$ design and thus a balanced design, where

$$\mu = \lambda\binom{v-2}{t-2} / \binom{k-2}{t-2}$$

Within this course a balanced block design is referred to as a (b,v,r,k,λ)-design In a (b,v,r,k,λ)-design

$$\lambda(v-1) = r(k-1)$$

Fisher's inequality states that $b \geq v$

symmetric design

In a (b, v, r, k, λ)-design, $\lambda(v-1) = r(k-1)$

In a symmetric design $b = v$, which therefore means that $r = k$, thus we have the parameters (v, v, k, k, λ). This is known as a (v, k, λ)-symmetric design.

In a (v, k, λ)-symmetric design, any pair of blocks intersects in precisely λ points

The code C constructed from a Steiner system $S(t, k, v)$ is a constant weight binary $(v, b, 2(k-t+1)$-code

e.g. With a binary (9,12,4)-code of weight 3. This is a Steiner code

A Steiner code has $\lambda = 1$,$k = 3$ and $t = 2$, also using the equation supplied $r = 4$

so we have a $v \times b$ matrix in which each point occurs 4 times, each set of 2 points occur only once, and there are 3 points in each block (column).

If I denote the blocks as numbers and points a letters, we can see that the 1st block can be made with ABC, the 2nd DEF, 3rd GHI.

Now A,B,and C have been used together once, so they cannot be used in another block together, and likewise with D,E,F and G,H,I

$$\begin{pmatrix} 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 \end{pmatrix}$$

A (v, k, λ)-symmetric design is a constant weight $(v, v, 2(k-\lambda))$-code

Creating a design

Fpor GF(q) create a set of q-1 MOLS as per unit 9, name these $A_1, A_2, ...A_{q-1}$
Create a $q \times q$ array named R where $r_{ij} = \lambda_i$ that is each point within the row

has the same value λ_i. Set this to A_q
Create a $q \times q$ array named C (not to be confused with C for code) where $c_{ij} = \lambda_i$ that is each point within the column has the same value λ_i. Set this to A_{q+1}
Create a $q \times q$ array N where the values are in some order $\{0, 1, ...q^2\}$ An example of this is, for GF(3)

$$A_1 = \begin{matrix} 0 & 1 & 2 \\ 1 & 2 & 0 \\ 2 & 0 & 1 \end{matrix} \quad A_2 = \begin{matrix} 0 & 1 & 2 \\ 2 & 0 & 1 \\ 1 & 2 & 0 \end{matrix}$$

$$R = A_3 = \begin{matrix} 0 & 0 & 0 \\ 1 & 1 & 1 \\ 2 & 2 & 2 \end{matrix} \quad C = A_4 = \begin{matrix} 0 & 1 & 2 \\ 0 & 1 & 2 \\ 0 & 1 & 2 \end{matrix}$$

$$N = \begin{matrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{matrix}$$

From here we can create partial blocks, denoted as $(i, j)^{th}$ partial blocks, where i is the Block A_i and j is the value within that block.
So for an $(1, 0)^th$ partial block we take the Block A_1 and look at the placement of all the 0's. We translate these positions onto N to get the values for that partial block, which will be $\{1, 6, 8\}$ Thus all the values will be

$(1, 0)$ 168	$(1, 1)$ 249	$(1, 2)$ 357
$(2, 0)$ 159	$(2, 1)$ 267	$(2, 2)$ 348
$(3, 0)$ 123	$(3, 1)$ 456	$(3, 2)$ 789
$(4, 0)$ 147	$(4, 1)$ 258	$(4, 2)$ 369

Adding a letter to each partial block where $i_1 = a...i_4 = d$ to produce

$(1, 0)$ 168a	$(1, 1)$ 249a	$(1, 2)$ 357a
$(2, 0)$ 159b	$(2, 1)$ 267b	$(2, 2)$ 348b
$(3, 0)$ 123c	$(3, 1)$ 456c	$(3, 2)$ 789c
$(4, 0)$ 147d	$(4, 1)$ 258d	$(4, 2)$ 369d

and creating a 2nd block consisting of $q^2+1, ...q^2+q+1$, and replacing the letters with these values respectively to create a projective plane of order 3 so the second block for the example will be $\{10, 11, 12, 13\}$ will will replace a, b, c, d. Hence creating the blocks of a projective plane of order 3, i.e. a (13,4,1)-symmetric design

Properites of a symmetric design

Matrix A has $k = r$ 1s in each row and each column.
Every pair of distinct rows contain 1s in corresponding positions λ times. The same is true for every pair of distinct columns.

A^T is also an incidence matrix of (v, k, λ)-symmetric design, called the dual design. Although A and A^T have the same parameters they may not be isomorphic.
The columns of A^T (= rows of A) form a constant weight binary $(v, v, 2(k-\lambda)$-code.

A **complimentary block**, denoted as C', is where you replace the 1s with 0s and vice versa.
there will still be v points and v blocks but each block will contain $v-k$ points.
Each point will occur in $v-r = v-k$ of these complimentary blocks.
The complimentary rows/columns, each pair will occur in $v-2r+\lambda) = v-2k+\lambda$ blocks. This is a $(v, v-k, v-2k+\lambda)$-symmetric design called the complimentary design.
It may be used to construct a constant weight binary $(v, v, 2(k-\lambda))$-code.
The parameters are the same as for the code obtained from the original symmetric design, however the weight of each codeword in the original is k whereas in the complimentary code it is $v-k$

Let C be the columns of an incidence matrix of (v, k, λ) symmetry design.
Let C' be the columns of an incidence matrix of (v, k, λ) complimentary design
Note this can also be done with rows of both C and C'
Construct a further code C^{ext} by taking all the codewords of C and C' plus the all 0s and all 1s codewords to give $2v+2$ codewords of length v.
The minimum distance of the different types of codewords are as follows

	All zeros	All ones	C	C'
All zeros	–	v	k	$v-k$
All ones		–	$v-k$	k
C			$2(k-\lambda)$	$v-2(k-\lambda)$
C'				$2(k-\lambda)$

Unit 11 - Perfect Codes and Bounds

$A_q(n,d)$ denotes the maximum number of codewords in any q-ary code of length n and minimum distance d. For instance,

$$A_q(n,1) = q^n$$

$$A_q(n,n) = q$$

$$A_2(n,d) = A_2(n-1,d-1)$$

see page 14-16 of H for more, plus Unit 1 Note 1.15

Singleton Bound

$$A_q(n,d) \leq q^{n-d+1}$$

If there exists a set of $n-2$ MOLS of order q, then

$$A_q(n,n-1) = q^2$$

By theorem 10.20 on page 123, a q-ary $(n, q^2, n-1)$-code is equivalent to a set of $n-2$ MOLS of order q
If q is a prime power and $n \leq q+1$ then

$$_q(n,n-1) = q^2$$

The largest number of codewords in q-ary linear code is denoted by $B_q(n,d)$ in which

$$b_q(n,d) \leq A_q(n,d)$$

Also $B_q(n,d)$ must be a power of q. We have:

n	$d=3$	$d=4$	$d=5$	$d=6$	$d=7$
5	4	2	2	–	–
6	8	4	2	2	–
7	16	8	2	2	2
8	16	16	4	2	2

The Gilbert-Varshamov bound provides a lower bound for when q is a prime power, and hence also for $a_q(n,d)$ in such cases.
If q is a prime power then there exists a q-ary $[n,k,d]$-code provided

$$\sum_{i=0}^{d-2}(q-1)^i\binom{n-1}{i} < q^{n-k}$$

Thus

$$q^k < \frac{q^n}{\sum_{i=0}^{d-2}(q-1)^i\binom{n-1}{i}}$$

Consequently, for q a prime power we have

$$A_q(n,d) \geq B_q(n,d) \geq q^{k_1}$$

where k_1 is the largest integer k satisfying the equation above this one

example Qs

Using a 2-(11,5,2) symmetric block design create a binary (11,24,5) code. Hence show that $A_2(11,5) = 24$ The 11 blocks listed on page 27 are 1,3,4,5,9....2,3,4,8,11 These can be put in a block table like so

	blocks		
	1	...	11
1	1	...	0
2	0	...	1
3	1	...	1
4	1	...	1
5	1	...	0
6	0	...	0
7	0	...	0
8	0	...	1
9	1	...	0
10	0	...	0
11	0	....	1

We can take the codewords to be either the columns or rows, plus these codewords with the 1 & 0s interchanged, plus the 0...0 and 1..1 codewords to create the 24 codewords needed.
Note that Lemma 2.6 shows that if $\mathbf{x}$ and $\mathbf{y} \in (F_2)^n$ then

$$d(\mathbf{x},\mathbf{y}) = w(\mathbf{x}) + w(\mathbf{y}) - 2w(\mathbf{x} \cap \mathbf{y})$$

where $\mathbf{x} \cap \mathbf{y}$) means the 1s in the same position in x and y e.g $11100 \cap 00111 = 00100$ Thus taking he 2 smallest weights 00000000000 and 10111000100

$$d(\mathbf{x},\mathbf{y}) = 5 + 0 - 2 \cdot 0 = 5$$

We know the minimum size of M is 24 as it is a (11,24,5)-code thus $A_2 \geq 24$ but Plotkins bound shows that $A_2(2d+1,d) \leq 4d+4 = 24$ thus $A_2(11,5) = 24$

Unit 12 - Weight Enumerators

If C is a linear $[n, k]$-code then it's weight enumerator is

$$\begin{aligned} W_C(z) &= \sum_{i=0}^{n} A_i z^i \\ &= A_0 + A_1 z + ... + A_n z^n \end{aligned}$$

where A_i denotes the number of codewords of weight i Another way of writing this is

$$W_C(z) = \sum_{x \in C} Z^{w(\mathbf{x})}$$

E.g. Let C be the even weight binary code of length 3. that is 000,011,101,110. Its dual code is 000,111 thus

$$\begin{aligned} W_C(z) &= 1 + 3z^2 \\ W_{C^\perp}(z) &= 1 + z^3 \end{aligned}$$

Macwilliams Identity If C is a linear $[n, k]$-code over GF(q) then

$$W_{C^\perp}(z) = \frac{1}{q^k}[1 + (q-1)z]^n W_C\left(\frac{1-z}{1+(q-1)z}\right)$$

e.g if $W_C(z) = 1 + 4z^4 + 16z^5 + 4z^6$ where $n = 6$, $k = 2$ then

$$\begin{aligned} W_{C^\perp}(z) &= \frac{1}{5^2}[1 + (5-1)z]^6 W_C\left(\frac{1-z}{1+(5-1)z}\right) \\ &= \frac{1}{25}[1+4z]^6 W_C\left(\frac{1-z}{1+4z}\right) \\ &= \frac{1}{25}[1+4z]^6 \left[1 + 4\left(\frac{1-z}{1+4z}\right)^4 + 15\left(\frac{1-z}{1+4z}\right)^5 + 4\left(\frac{1-z}{1+4z}\right)^6\right] \end{aligned}$$

If we have $W_{C^\perp}$ and want to find W_C then $(C^\perp)^\perp = C$

Probability of undetected errors

$$P_{undetec}(C) = (1-p)^n \left[W_C\left(\frac{p}{1-p}\right) - 1\right]$$

We can also use the following formula

$$P_{undetec}(C) = \sum_{i=1}^{n} A_i p^i (1-p)^{n-i} (q-1)^{-i}$$

example Let $W_C(z) = 1 + 4z^2 + 64z^3 + 144z^4 + 248z^5 + 164z^6$ where $q = 5$ and $n = 6$ we get

$$P_{undetec}(C) = \sum_{i=1}^{6} A_i p^i (1-p)^{6-i} 4^{-i}$$

where $A_1 = 0$, $A_2 = 4$,..etc NOTE that we do not count A_0 as the summation starts at 1.
Also note that each summation tends to 0 as $i \to \infty$, thus

$P_{undetec} \approx A_i p^i (1-p)^{n-i} (q-1)^{-i}$ where i is the smallest non-zero value

Unit 13

Interleaving

Interleaving is a method for improving burst-correcting properties of a code. It spreads the burst error over several codes. For instance, by using **block interleaving**, if we to interleave to a depth of 3 we would arrange the codewords that are to be transmitted into groups of 3, then with the 1st group we take the 1st digit of each codeword, then add the 2nd digit of each codeword and so on until all the digits are used, to create a new codeword.
E.g codewords of C which has $d = 3$ and thus can correct up to 1 error

$$c_1 = 1000011$$
$$c_2 = 1100110$$
$$c_3 = 0010110$$

thus the transmission codeword is 110010001000011100 Should a single burst of length 3 happens, then this will now only affect 1 bit of each codeword, which can be corrected providing there is no other error.

To decode a message we remove the effects of interleaving (vector v_1 is the 1st bit of each block) and then finding the closest codeword in C. Suppose C is an l burst error correcting binary linear code. If C is interleaved to depth s then each codeword can correct all burst of length sl provided only 1 burst error and no other errors on that codeword.

f-frame delay interleaving also known as convolution interleaving, works by having the codeword written in a column then offsetting each row within the block by f amount more than the previous row
For instance:
if we have 5 codes of length 3 a, b, c, d, e in which $a = a_1a_2a_3$ and so on then we can arrange them in a block such as

$$\begin{array}{ccccccccc} a_1 & b_1 & c_1 & d_1 & e_10 & 0 & 0 & 0 & 0 \\ 0 & 0 & a_2 & b_2 & c_2 & d_2 & e_2 & 0 & 0 \\ 0 & 0 & 0 & 0 & a_3 & b_3 & c_3 & d_3 & e_3 \end{array}$$

will give the transmission codeword $a_1 00\ b_1 00\ c_1 a_2 00$ .... This particular codeword has $f = 2\ s = 3$

Suppose that C is an l burst error correcting binary linear $[n, k, d]-$code. If C is f-frame delay interleaved then all bursts at most $l(fn+1)$ will be corrected provided that each codeword is affected by only one burst and no other errors.

1 cross-interleaving

Interleaving a code may be combined with a second code to improve performance. Suppose we have 2 codes C_1 and C_2. First we encode the messages using G_1, then we interleave the codewords to a depth k_2 to produce a vector. This vector is then encoded k_2 bits at a time to produce the codewords of C_2. Suppose we have 4 messages ($k_1 = 4$) of length 4. We use these on G_1 to get 4 codewords.

$$\begin{aligned} c_1 =& m_1 G_1 \\ c_2 =& m_2 G_1 \\ c_3 =& m_3 G_1 \\ c_4 =& m_4 G_1 \end{aligned}$$

We now create a vector using block interleaving to a depth of $k_1 = 4$ We can now use these new message (n_i) vectors much in the same way as above but on G_2 to give

$$\begin{aligned} d_1 =& n_1 G_2 \\ d_2 =& n_2 G_2 \\ d_3 =& n_3 G_2 \\ d_4 =& n_4 G_2 \\ d_5 =& n_5 G_2 \\ d_6 =& n_6 G_2 \\ d_7 =& n_7 G_2 \\ d_8 =& n_8 G_2 \end{aligned}$$

Using block interleaving to a depth of what is asked for (say 2 or 3) produces the vector that can be transmitted. Depth 2 will give $d_1 d_2, d_3 d_4$.... A depth a 3 will give $d_1 d_2 d_3, d_4 ..., d_7 d_8 \cdot$ in which $\cdot$ is the place holder for d_9 which hasnt been given

Theorem 13.5 Suppose that the binary linear $[n_1, k_1, d_1]$-code C_1 and the binary linear $[n_2, k_2, d_2]$-code C_2 are cross-interleaved to depth $s \leq (d_1 - 1)/2$ Then all bursts of length at most $s(d_2 - 1)$ will be corrected provided that each codeword is affected by at most one burst error and no other errors.
If we can be sure that a burst error affects at most one group of s codewords then the inequality can be relaxed to $s \leq d_1 - 1$. One way of doing this is to separate the groups by a number of zeros.

To decode a message we first find the vectors d_i and see which group is incorrect by checking with G_2. Once found, we can then create the large vector, noticing that with G_2 in standard form the 1st 4 bits equate to n_i. With the incorrect group we denote each bit as (?).
For instance, the following vector is received

010 101 010 001 101 110 110
110 010 101 111 000 100 011
101 100 010 001 111 110 000

Usncrambling with G_2 (not shown) gives

$$\begin{array}{lll} v_1 = 0100111 & v_4 = 1011010 & v_7 = 1100110 \\ v_2 = 1010011 & v_5 = 1101001 & v_8 = 0010110 \\ v_3 = 0101100 & v_6 = 0011001 & v_9 = 1001100 \end{array}$$

Say that $v_1 - v_3$ were incorrect and knowing that G_2 is in standard form and therefore the message n_i is the 1st 4 bits of each codeword d_i we get

???? ???? ???? 1011 1101 0011 1100 0010 1001

Thus giving the original message

???110101
???010100
???101010
???111001

Using inspection on G_1 (not shown) will give the 1st 3 values to be

010110101
101010100
010101010
001111001

And from these and G_1 we can work out the messages using $m_i = c_i G_1^T$ to give

0101 1010 0101 0011

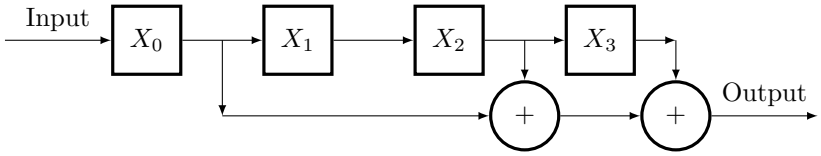
Remember if only one burst is expected and you have incomplete parts of a signal then if there is another burst error, then these cannot have a burst error and must be correct. For instance Ending up with the vector ????????????????????01101010... and applying it to Repetition code $G_1 = [11111]$ we have

?????0 ?????1 ?????1 ?????0 1... 0... 1... 0...

As the ? denotes a burst error, then the 1... and 0... must be error free thus giving 1111 and 0000 respectively.

Shift Register

Shift Registers A Feedback shift register is a pictorial of $g(x)$ in which we have a flipflop node, denoted as a recatngular box in which contains a 0 or 1, and a XOR node which either produces a 0 if the 2 inputs are both 0 or 1, and a 1 otherwise.



Above is a pictorial of $g(x) = 1+x^2+x^3$ as these are the values that are summed to the output.

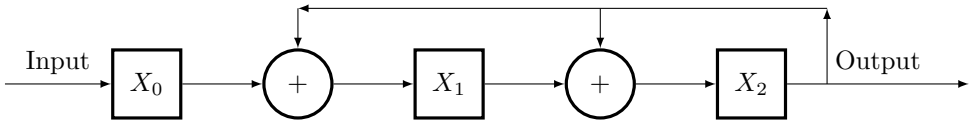
With $g(x)$ being the generator we can also input a polynomial $q(x)$ to give $a(x) = g(x)q(x)$
For instance if we had $g(x) = x + x^2$ and $q(x) = 1 + x + x^2$ then we can create an I-O table to show the steps

Time	*Input*	$X_0X_1X_2$	*Output* $= X_1 + X_2$
-1	$-$	000	$-$
0	1	100	0
1	1	110	1
2	1	111	0
3	0	011	0
4	0	001	1

As above you can see that the input will go straight into X_0 whilst everything else is shifted along. The output is XOR of $X_1 + X_2$. The input is the values of $q(x)$ ensuring that we stop only when there is only a 1 left in the last flipflop.
If we now read the values of the time and output we see there at time 1 there is an output of 1, likewise for time 4. This equate to $a(x) = x + x^4$.
If we work out $a(x) = g(x)q(x) = (x+x^2)(1+x+x^2) = x+x^2+x^3+x^2+x^3+x^4 \equiv x + x^4 \pmod 2$.

If $g(x) = g_0 + g_1(x) + ... + g_r x^r$ is a generator polynomial for a binary cyclic [n,n-r]-code C, and if $q(x) = q_0 + q_1 x + ... + q_{n-r} x^{n-r}$ represents a message then the corresponding codeword is $a(x) = q(x)g(x)$.

Th 12.14 asserts that if C is a cyclic [n,k]-code with generator polynomial $g(x)$ then there exists a check polynomial $h(x)$ of degree k such that $g(x)h(x) = x^n - 1$. Furthermore a polynomial $c(x)$ of R_n is a codeword of C iff $c(x)h(x)$ is a multiple of $x^n - 1$. To compute $h(x)$ we must divide $x^n - 1$ by $g(x)$ and to check $c(c)$ is a codeword of C we must divide $c(x)h(x)$ by $x^n - 1$. To do this we use a Feedback Shift Register



Above is the feedback shift register for $g(x) = x + x^2 + x^3$

We can also create an IO table for this. For example $(1 + x^4 + x^5)/(x + x^2 + x^3)$

$Time$	$Input$	$X_0X_1 + d_tX2 + d_t$	$Output = d_t$
-1	$-$	000	$-$
0	1	100	0
1	1	110	0
2	0	011	0
3	0	010	1
4	0	001	0
5	1	111	1

As seen above, the divisor is the feedback shift register, the numerator is the message. This time we start with the highest order working down. Once the step has a 1 in its output it will change the $Xi + d_t$ values within that row. The number of steps is the same as the highest order of the numerator.
The output is the quotient and the last line (time=5 in this case) is the remainder.
So for the above we have the quotient $x^2 + 1$ and remainder $x^2 + x + 1$. Thus $(1 + x^4 + x^5) = (x + x^2 + x^3)(x^2 + 1) + (x^2 + x + 1)$. If we were to do the polynomial division $\pmod 2$ we would get the same answer.

Unit 14 - Convolutional Codes

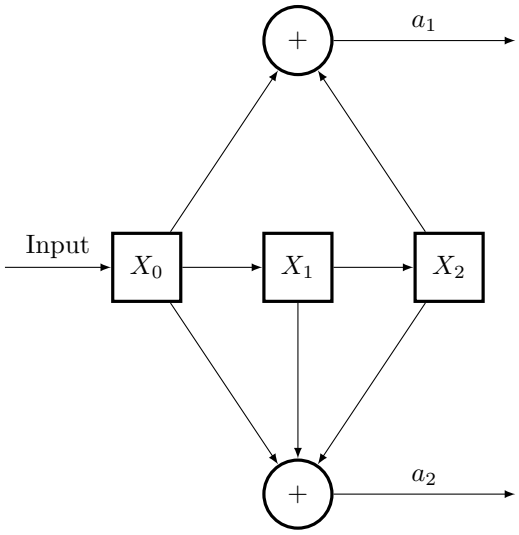
These are suited for continuous streams of message bits, such as digital broadcasting. f-frame is a type of convolutional code, whereas the others are known as block codes.
A finite vector can be represented as a polynomial, as we have been doing.

Extending this idea we can represent an infinite vector by the power series

$$a(x) = a_0 + a_1 x + a_2 x^2 + ... = \sum_{i=0}^{\infty} x^i$$

With GF(2) $a_i = 0$ or 1.

A power series is said to be invertible if there exists $b(x)$ such that $a(x)b(x) = 1$
In fact, this is only the case if $a_0 = b_0 = 1$



Above is a shift register with 1 input and two outputs, it can be regarded as 2 separate shift registers, where $g_1(x) = 1 + x^2$ and $g_2(x) = 1 + x + x^2$. The highest degree of these polynomials iss $m = 2$. This is a $(2, 1, 2)$-code

A convolution code has the form (n, k, m) where n is the number of output bits, k is the number of input bits, and m is the highest degree of the polynomials.
The rate of such code is k/n

With the above shift register, encoding a message $q = 101000...$ gives the fol-

lowing IO table.

Time	*Input*	$X_0\ X_1\ X_2$	*Output* $a_1\ a_2$
-1	$-$	0 0 0	$-\ -$
0	1	1 0 0	1 1
1	0	0 1 0	0 1
2	1	1 0 1	0 0
3	0	0 1 0	0 1
4	0	0 0 1	1 1
5	0	0 0 0	0 0

Interleaving the outputs a_1 and a_2 we obtain the encoded message

$$11\ 01\ 00\ 01\ 11\ 00...$$

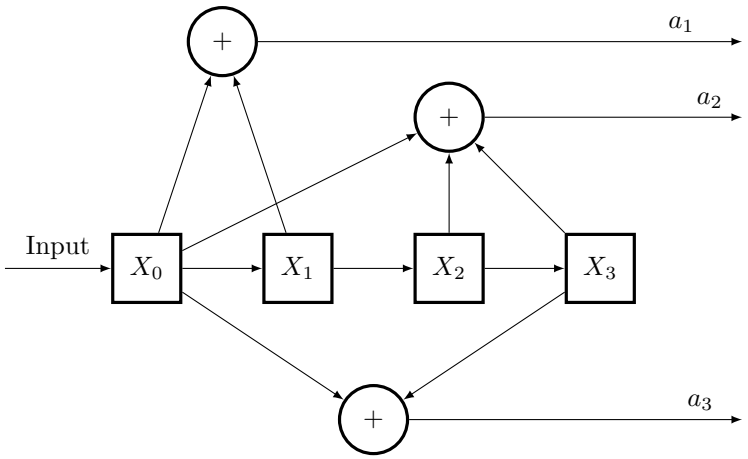
Alternatively this can be expressed as $a_1(x) = q(x)g_1(x) = 1 + x^4$ and $a_2(x) = 1 + x + x^3 + x^4$.
Interleaving the two output corresponds to forming

$$\begin{aligned} a(x) =& a_1(x^2) + xa_2(x^2) \\ =& (1 + x^8) + x(1 + x^2 + x^6 + x^8) \\ =& 1 + x + x^3 + x^7 + x^8 + x^9 \\ =& 10\ 01\ 00\ 01\ 11... \end{aligned}$$

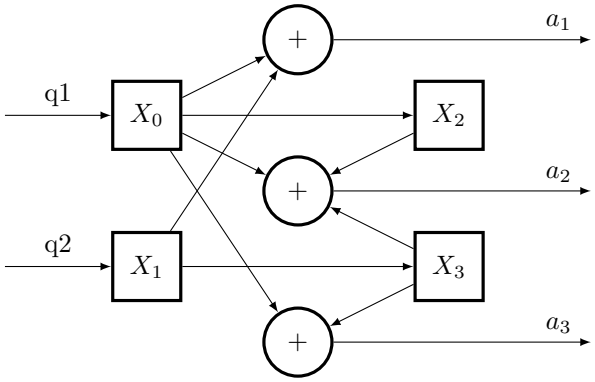
Another example: use the convolution code C with generator polynomials $g_1(x) = 1 + x^2, g_2(x) = 1 + x^3, g_3(x) = 1 + x + x^3$ to encode the message $1 + x^2 + x^3$
First of all, with $n = 3$ lets find $a(x)$ remembering that with $a_i(x^3)$ we only cube any x^y where $y \geq 1$ (so not 1), and that if $a(x) = x^2$ then $a(x^3) = a(x)^3 = (x^2)^3 = x^{2\cdot 3=6}$

$$\begin{aligned} a(x) &= q(x^3)[a_1(x^3) + xa_2(x^3) + x^2a_3(x^3)] \\ &= q(x^3)[(1 + x^6) + x(1 + x^9) + x^2(1 + x^3 + x^9)] \\ &= q(x^3)[1 + x^5 + x + x^{10} + x^2 + x^6 + x^{11}] \\ &= q(x^3)[1 + x + x^2 + x^5 + x^6 + x^{10} + x^{11}] \\ &= (1 + x^6 + x^9)[1 + x + x^2 + x^5 + x^6 + x^{10} + x^{11}] \\ &= 1 + x + x^2 + x^5 + x^6 + x^{10} + x^{11} + x^6 + x^7 + x^8 + x^{11} + x^{12} + x^{16} + x^{17} + x^9 + x^{10} + x^{11} + x^{14} + x^{15} + \\ &= 1 + x + x^2 + x^5 + x^7 + x^8 + x^9 + x^{11} + x^{12} + x^{14} + x^{15} + x^{16} + x^{17} + x^{19} + x^{20} \\ &= 111\ 001\ 011\ 101\ 101\ 111\ 011... \end{aligned}$$

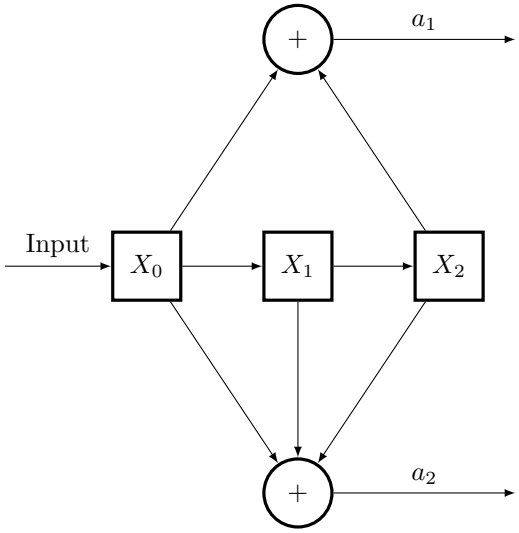
Above we have been talking about an $(n, 1, m)$-code. Below is a $(3, 2, 3)$-code



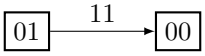
The above has one input, so for $k =$ we need to move 2 bits at a time. Noticing that this means that with bits ab a will move to X_2, b will move to X_1, likewise when the next to bits along a will move to X_4, b will move to X_3. so all the odd bits go to X_2, X_4, all the even bits go to X_1, X_3, this we can separate this shift register to look like



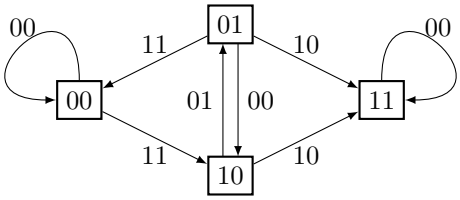
State diagrams Going back to this state register



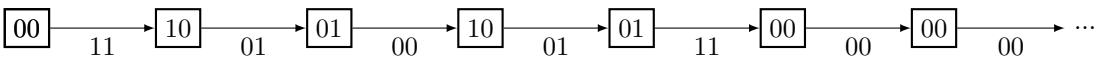
we can create a state diagram to represent the contents for what is in X_0 and X_1, know as the state of the encoder. If we have 01 ($X_0 = 0, X_1 = 1$) and we input a 0 to the shift register, then ($x_0 = 0, X_1 = 0, X_2 = 1$) and thus $a_1 = 1, a_2 = 1$ This can be represented as



Filling out the whole state diagram gives



In the general case of $(n, 1, m)$ convolutional code there will be 2^m states giving 2^m nodes on the diagram corresponding to 2^m possible values of $X_0, X_1, ...X_{m-1}$ Each node will have two edges directed outward and two edges directed inwards, with loops counting once in each direction. Each edge will be marked with an n-tuple giving the corresponding output. The state diagrams for two different $(n, 1, m)$ convolutional codes will differ only in respect of these edge labels.
If $q = 1010000$ is fed into the shift register above we can read off the corresponding directed walk as below (starting at 00...0)



Hence the encode message is 11 01 00 01 11 00 00 ...

We may also reverse this procedure. Given a codeword we can locate the di-

rected walk which gives rise to it and consequently obtain the original message, starting at 00 node. Once all nodes found, take the 1st digit of each node to get q.

Catastrophic Codes

Theorem 14.1 Suppose that $g_1(x)$ and $g_2(x)$ are polynomials over GF(2) with no common factor of degree at least 1. i.e $GCD(g_1, g_2) = 1$ Then there exists polynomials $p_1(x)$ and $p_2(x)$ such that $p_1(x)g_1(x) + p_2(x)g_2(x) = 1$

Above is the theorem for a non catastrophic code. A convolutional code that is prone to catastrophic error propagation, i.e. a situation in which a finite number of channel errors causes an infinite number of decoder errors. Any given convolutional code is or is not a catastrophic code.

Suppose that $g_1(x), g_2(x), ..., g_n(x)$ are polynomials over GF(2) with GCD=1, then there exist polynomials $p_1(x), p_2(x), ..., p_n(x)$ such that $\sum_{i=1}^{n} p_i(x)g_i(x) = 1$

An $(n, 1, m)$ convolution code C is catastrophic iff its state diagram contains a cycle, other than the loop on the zero state, all of whose edge labels are 0s.

Crytography

First some terms
Plaintext - original message P
Ciphertext - enciphered message C
plaintext to cyphertext is called encrypting
cyphertext to plaintext is called decrypting

Simplest encryption is $C \equiv P + k \pmod{26}$ where the letters are denoted as numbers 00-25, k is any number $0 \leq k \leq 25$ Decryption is $p \equiv C - k \pmod{26}$

Next is '$C \equiv aP + k \pmod{26}$ where a is relatively prime to 26, tht is $a = 1, 3, 5, 7, 9, 11, 15, 17, 19, 21, 23, 25$ Decryption by $P = a^{-1}(C - k) \pmod{26}$

enciphering using a permutation of letters Knowing the frequency of letters in texts and use of the english language we can guess at short ciphered words to get a bigger picture of the other words. There is no system for the ciphering. A one letter word would normally be 'a' or 'I'. E and T are the most common letters in texts.

Block Cipher Systems This is using a matrix to encode a block of letters. where $C \equiv AP \pmod{26}$ where A is a matrix.
$P \equiv A^{-1}C \pmod{26}$ to decode. e.g

$$A = \begin{pmatrix} 2 & 11 \\ 19 & 23 \end{pmatrix}$$

to encode GROUP Splitting GROUP into blocks of 2 to get GR OU PZ where Z is used to fill in the gaps. thus we get

$$\begin{pmatrix} 2 & 11 \\ 19 & 23 \end{pmatrix} \begin{pmatrix} 6 \\ 17 \end{pmatrix} = \begin{pmatrix} 2 \cdot 6 + 11 \cdot 17 \\ 19 \cdot 6 + 23 \cdot 17 \end{pmatrix} = \begin{pmatrix} 199 \\ 505 \end{pmatrix} = \begin{pmatrix} 17 \\ 11 \end{pmatrix} \pmod{26}$$

giving the 1st block GR encoded as RL. Carry on with the rest of the blocks to get the answer RLOYTC
To decipher

$$A = \begin{pmatrix} a & b \\ c & d \end{pmatrix} \qquad A^{-1} = (ad - bc)^{-1} \begin{pmatrix} d & -b \\ -c & a \end{pmatrix}$$

Playfair cypher Matrix is made up of all letters apart from I or J, pick a keyword which has no letters repeated. put the unused letters in order afterwards
eg. Keyword=CIPHER

C	I	P	H	E
R	A	B	D	F
G	K	L	M	N
O	Q	S	T	U
V	W	X	Y	Z

pick a plaintext word such as CRYPTOGRAPHY break into blocks of 2.
If the 2 letters are in the same row, the cipher letter is the letter to the right (wrapped around, so right of the 5th letter is the 1st letter)
if the letters are in the same column then the encrypted letter is one position down
if in different rows and columns, draw a box around them and the encryted letter is the opposite corner letter in the same row.

so we have CR YP TO GR AP HY CR in the same column so we get RG
YP are in different row/column. so Y goes to X and P goes to H giving XP and so on to give RG XP UQ OG BI DH

Exponential Ciphers Let p be a large prime number. Choose e, the enciphering key so that it is relatively prime to $p - 1$. Replace each letter by a 2 digit number ie A=00, B=01....

Split the resulting sequence of digits into blocks of $2n$ (corresponding t n consecutive Zs) where n is the largest possible integer $< p$. so with Z=25 for the alphabet, we have 25,2525,252525 to get $25 < p < 2525, n = 1$ $2525 < p < 252525, n = 2$ etc
Encipher each block using $C \equiv P^e \pmod{p}$

To decipher we need d in which $d = e^{-1} \pmod{p-1}$. Giving $de \equiv 1 \pmod{p-1}$ giving

$$C^d \equiv (P^e)^d \equiv P^{k(p-1)+1} \equiv (P^{p-1})^k \equiv P \pmod{p}$$

for some positive integer k

Example: With $p = 53$ and $e = 21$ encrypt DOG=031406. $n = 1$ as $< 25p < 2525$, so we have blocks 03 14 06.
Note that with powers of 2 $X^{21} = X^{16} + X^4 + X^1$ where the powers are base 2 i.e $2^0, 2^1, 2^2$... so with $e = 21 = 16 + 4 + 1$ we create a table for powers of P up to 16, (mod 53)

$P =$	03	14	06
$P^2 =$	09	37	36
$P^4 =$	28	44	24
$P^8 =$	42	28	46
P^{16}	15	42	49

So $P^e = P^{16} \cdot P^4 \cdot P$
Giving D=$15 \cdot 28 \cdot 03 \equiv 41 \pmod{53}$
O=$42 \cdot 44 \cdot 14 \equiv 08 \pmod{53}$
G=$49 \cdot 24 \cdot 06 \equiv 07 \pmod{53}$
so the cyphertext is 41 08 07.

To decipher we need to find $ed \equiv 1 \pmod{p-1}$ that is $21d \equiv 1 \pmod{52}$ by inspection, or by using Euclids algorithm $d = 5$. With ciphertext 22 25 20 and $d = 5 = 4 + 1$ we have the table

$C =$	22	25	20
$C^2 =$	07	42	29
$C^4 =$	49	15	46

and we see that $22^5 = 22^4 \cdot 22^1 = 49 \cdot 22 \equiv 18 \pmod{53}$=S. Carrying on we get SET

RSA This uses the same as above but we use (mod m) where $m = pq$ where p and q are primes. We choose e to be relatively prime to $\phi(m) = (p-1)(q-1)$. The values m and e are made public but as p and q are kept secret there is no way of guessing these values for large integers of m

Knapsack Problem A knapsack of capacity S is packed with items $a_1, a_2, ...a_n$ which collectively has a total capacity larger than S. With a_1 being the largest

capacity item, and the others decreasing in value, we can use the greedy algorithm, in which we add the items in order, until the knapsack cannot hold any other item. e.g. if we have 2,5,6,7,15. $S = 19$ we have 15+2=17. This is not optimal as 7+6+5=18

We are only interested in the case where the knapsack can be filled with no space left, that is

$$\sum_{i=1}^{n} a_i x_i = S$$

where x_i is 0 or 1. There are 2^n possibilities for the vector $(x_1, x_2..x_n)$ and it is not feasible to check every possibility. On the other hand, if $a_1, a_2..a_n$ is super-increasing the solution is almost trivial. A super-increasing sequence is such that

$$\sum_{i=1}^{j-1} a_i < a_j$$

for $j = 2, 3, .., n$. i.e. each integer is greater than the sum of all previous integers in the sequence. e.g 1,3,5,10,25,48.

To solve the knapsack problem for a super-increasing sequence we simply apply the greedy algorithm.

$$x_n = \begin{cases} 0, & \text{if } S < a_n. \\ 1, & \text{if } S \geq a_n. \end{cases}$$

then successively put

$$x_j = \begin{cases} 0, & \text{if } S - \sum_{i=j+1}^{n} a_i x_i < a_j. \\ 1, & \text{if } S - \sum_{i=j+1}^{n} a_i x_i \geq a_j. \end{cases}$$

Finally compute the remainder

$$R = S - S - \sum_{i=1}^{n} a_i x_i$$

If $R = 0$ then a solution is obtained and it is unique.

For a public-key encryption a person chooses there own super-increasing sequence $a_1..a_n$ together with a modulus $m > 2a_n > \sum_{i=1}^{n} a_i$ and an integer e which is relatively prime to m.
For each a_i calculate $b_i \in Z_m$ by

$$b_i \equiv ea_i \pmod{m}$$

In general the sequence b_i will not be super-increasing.. This sequence together with m forms the public key.
When a plaintext message is to be sent the letter is converted into a number,

then a 5 digit binary number, eg N=13=01101. The resulting sequence is then split into blocks of n zeros and ones, filling out the last block if necessary. If $P = (x_1, x_2, ..., x_n)$ is such a block then $C \in Z_m$ is calculated by

$$C \equiv \sum_{i=1}^{n} b_i x_i \pmod{m}$$

The sequence of such sums form the ciphertext.

To break the cipher it is necessary to solve a knapsack problem for each block, where the set of integers $\{b_1, b_2, ..., b_n\}$ and the sum is some number congruent to C mod m. However the legitimate recipient knows e and hence can compute $d = e^{-1}$ solving $de \equiv 1 \pmod{m}$ Then

$$dC \equiv \sum_{i=1}^{n} a_i x_i \pmod{m}$$

giving a knapsack problem with a super-increasing sequence which can easily be solved to obtain $P = (x_1, x_2, ...x_n)$ and from here recovering the plaintext message.

Example:
super-increasing sequence 1,4,6,15,37,66 (mod 133) and multiplier 32.

The public key is

$$\begin{aligned}
1 \cdot 32 &= 32 \equiv 32 \pmod{133} \\
4 \cdot 32 &= 128 \equiv 128 \pmod{133} \\
8 \cdot 32 &= 256 \equiv 123 \pmod{133} \\
15 \cdot 32 &= 480 \equiv 81 \pmod{133} \\
37 \cdot 32 &= 1184 \equiv 120 \pmod{133} \\
66 \cdot 32 &= 2112 \equiv 117 \pmod{133}
\end{aligned}$$

so the public key is 32,128,1223,81,120,117

Encoding RING is 17,08,13,06 which equates to 10001,01000,01101,00110. Since the public key is length 6 we split these into blocks of the same length,with 1s to fill the last block 100010 100001 101001 101111 This gives

$$\begin{aligned}
100010 &= 32 + 120 \equiv 19 \pmod{133} \\
100001 &= 32 + 117 \equiv 16 \pmod{133} \\
101001 &= 32 + 123 \equiv 6 \pmod{133} \\
101111 &= 32 + 123 + 81 + 120 + 117 \equiv 74 \pmod{133}
\end{aligned}$$

Thus the ciphertext is 19,16,6,74.

To decipher the received ciphertext we first need to calculate the multiplicative inverse of the multiplier 32 (mod 133) Using Euclids algorithm:

$$\begin{aligned} 133 &= 32(4) + 5 \\ 32 &= 5(6) + 2 \\ 5 &= 2(2) + 1 \\ 2 &= 1(2) + 0 \end{aligned}$$

Giving

$$\begin{aligned} 1 &= 5 + 2(-2) \\ 2 &= 32 + 5(-6) \\ 5 &= 133 + 32(-4) \end{aligned}$$

$$\begin{aligned} 1 &= 5 + (-2)(32 + 5(-6)) \\ &= 5 + (-2)32 + 5(12) \\ &= (-2)32 + 5(13) \\ &= (-2)32 + (13)(133 + 32(-4)) \\ &= (-2)32 + (13)133 + (-52)32 \\ &= (-54)32 + (13)133 \end{aligned}$$

Thus we have $32 \cdot -54 \equiv 1 \cdot 133$ so the inverse is $133 - 54 = 79$. With a ciphertext 110,4,110,74

$$\begin{aligned} 110 \cdot 79 = 8690 &\equiv 45 \pmod{133} \\ 4 \cdot 79 = 316 &\equiv 50 \pmod{133} \\ 110 \cdot 79 = 8690 &\equiv 45 \pmod{133} \\ 74 \cdot 79 = 5846 &\equiv 127 \pmod{133} \end{aligned}$$

Solving the knapsack problem with the original super-increasing sequence 1,4,8,15,37,66

$$\begin{aligned} 45 &= 37 + 8 \\ 50 &= 37 + 8 + 4 + 1 \\ 45 &= 37 + 8 \\ 127 &= 66 + 37 + 8 + 1 \end{aligned}$$

Expressing these as binary values we have 001010 111010 001010 101111. splitting these into blocks of length 5, discarding any trailing 1s or 0s 00101 01110 10001 01010 ~~1111~~ converting these to numbers 05,14,17,10. Which represents plaintext FORK.