Solutions 2

Andrew Niedens

Assignment 13

Problem 1: Finding Error

The receiver will not find an error, if even number of errors were made. We are not interested in case 0. Biggest contribution comes from term with 2 errors. It is: $\binom{N}{2} * p^2 * (1-p)^{N-2}$

Where N is total message length, and p is error probability.

Problem 2: Finding Error

a) Not to find correct error in 3-repeat code, there has to be 2 or 3 errors in it. Probability that there are exactly 2 errors = (Number of ways we can choose 2 errors) * (probability of 2 errors at given positions). This is equal to $\binom{3}{2} * p^2 * (1-p)$. Probability that there are exactly 3 errors is $\binom{3}{3} * p^3$. Summing those up, we get: $3*p^2*(1-p)+1*p^3 = 3p^2-3p^3+p^3 = 3p^2-2p^3$

Similarly, for 5 errors, we sum up terms with 3, 4 and 5 errors: $\binom{5}{3} * p^3 * (1-p)^2 + \binom{5}{4} * p^4 * (1-p) + \binom{5}{5} * p^5 = 10p^3 * (1-2p+p^2) + 5 * p^4 - 5p^5 + p^5 = 10p^3 - 15p^4 + 6p^5$.

b) p = 0.01. Putting in values, we get: 3-repeat code : $3 * (0.01)^2 - 2(0.01)^3 = 0.000298$. 5-repeat code: $10 * (0.01)^3 - 15 * (0.01)^4 + 6 * (0.01)^5 = 0.0000098506$. We see, that 5-repeat code is bout 30 times more effective.

In Hamming code, we cannot fix the code if there are more then 2 errors: $\binom{7}{2} * p^2 * (1-p)^5$. Other terms neglected.

c) Same thing as above, inserting p = 0.001. For 3-repeat code, we get: 0.000002998. For 5-repeat: $9.985 * (10^{-9})$. Now 5-repeat is 300 times more effective.

Problem 3: Repeating Information

$\binom{2N+1}{N+1} * p^{N+1} * (1-p)^N < 10^{-18}$ For p = 0.0025. We find, that $N = 8$ is enough, or total length 17

Assignment 14

Problem 1: Sending code with non-uniform distribution

X is sent, Y is received

a) $p = \frac{1}{5}$. 3-repeat code. We received 110. $P(X = 000) = \frac{9}{10}$ Maximum probability method: $P(X = 000|Y = 110) = \frac{P(X = 000, Y = 110)}{P(Y = 110)}$ Lets compute both terms.

$P(Y = 110) = P(Y = 110, X = 000) + P(Y = 110, X = 111) = P(X = 000)P(Y = 110|X = 000) + P(X = 111)P(Y = 110|X = 111) = \frac{9}{10} * p^2(1-p) + \frac{1}{10} * p(1-p)^2$

$P(X = 000, Y = 110)$ is the first term in the sum above. Dividing, we get: $P(X = 000|Y = 110) = \frac{9*4}{9*4+4*4} = \frac{36}{52} > 1/2$ So we choose $X = 000$

Maximum likelihood method: $P(Y = 110|X = 000) = p^2(1-p)$ because we made 2 errors. That is equal to $\frac{4}{125}$ $P(Y = 110|X = 111) = p(1-p)^2 = \frac{16}{125}$. The second term is bigger, so we choose $X = 111$, which is different from result we god in part 1(And is wrong, because the distribution was not uniform)

b) Let's to this one a bit differently. The channel is symmetrical, so ML gives us

- $P(Y = 1011|X = 1000) = p^2(1-p)^2$ - two errors
- $P(Y = 1011|X = 0110) = p^3(1-p)$
- $P(Y = 1011|X = 0001) = p^2(1-p)^2$
- $P(Y = 1011|X = 1111) = p(1-p)^3$

And for IO(Ideal Observer), we have to compare same values, but multiplied by $P(X = x_i)$ for respective i

For p $= \frac{1}{3}$ ML gives us $X = 1111$, IO gives $X = 1000$

For p $= \frac{1}{5}$ ML gives us $X = 1111$, IO gives $X = 1111$

ML gives the same result, because the channel is symmetrical and we can calculate the answer by least distance method in both cases.

c) Same method as above, both ML and IO give $X = 0001$

Problem 2: Cyclic codes

a) We get 11 words by cyclic rotations (pay attention, that word does not repeat itself till full cycle) + 1 word for all zeros = 12 words

b)Distance of any cycled code with zeros is 5. Lets denote by x_i the i'th cycle of given string. Now can compare x_0 to all other permutations. We can do only that because $d(x_i, x_j) = d(x_0, x_{j-i})$, when $i < j$. Also we note, that $d(x_0, x_i) = d(x_{11-i}, x_0)$, by cycling $11 - i$ times. Now we check all left options and conclude, that d = 5.

c) $e = \frac{d-1}{2} = 2$

Problem 3: Hamming distance properties

a)$\forall x, y : d(x, y) \geq 0$

Amount of differences cannot be negative

b)$d(x, y) = 0 \iff (x = y)$

If $x = y$, there are no differences, so $d(x, y) = 0$. If $d(x, y) = 0$, there are no differences, so $x = y$

c)$d(x, y) = d(y, x)$

Number of differences is independent of ordering

d)$d(x, z) \leq d(x, y) + d(y, z)$

Take x, y, z and divide into 4 subsections. In section 1, x = y and y = z

1. $x = y \land y = z$ - All three words are same

2. $x = y \wedge y \neq z$ - y and z differ, but x and y don't
3. $x \neq y \wedge y = z$
4. $x \neq y \wedge y \neq z$

Lets see, how each section contributes to the corresponding sum. Section 1 adds 0 to both sides of inequality. Sections 2 and 3 add 1 to both sides, because if x is same, as y and y differs from z, then x must differ from z as well. Section 4 adds 2 to the right side of inequality, but cannot add more than 1 to the left side. We conclude, that the right side cannot be less.

Assignment 15

Problem 1: Asymmetric channel capacity

Lets compute $I(X : Y)$ for this channel. It is equal to $H(Y) - H(Y|X)$. p is P(X = 1)

$H(Y) = -[\frac{p}{2} * \log_2 \frac{p}{2}] + (1 - \frac{p}{2}) * \log_2 (1 - \frac{p}{2})]$

$H(Y|X) = p * H(Y|X = 1) + (1 - p) * H(Y|X = 0)$. The latter equals 0, because Y is 0 with probability 1. The first term is $p * \log_2 2 = p$, because Y is distributed equally

We get: $I(X : Y) = -[\frac{p}{2} * \log_2 \frac{p}{2}] + (1 - \frac{p}{2}) * \log_2 (1 - \frac{p}{2})] - p$ We want that to be maximum. Taking derivatives:

$\frac{\mathrm{d}I(X : Y)}{\mathrm{d}p} = -[\frac{1}{2} * \log_2 \frac{p}{2} + \frac{1}{2} - \frac{1}{2} * \log_2 (1 - \frac{p}{2}) - \frac{1}{2}] - 1 = 0$

$\log_2 (\frac{2 - p}{p}) = 2$

$p = \frac{2}{5}$ and we find corresponding Capacity by plugging p in formula

Problem 2: BSC and BEC channel capacity

$I(X : Y) = H(Y) - H(Y|X)$

$P(Y = 0) = \frac{1}{2} * \alpha + \frac{1}{2} * (1 - \alpha - \beta) = \frac{1 - \beta}{2} = P(Y = 1)$

$P(Y = ?) = \beta$

$H(Y) = -p_i * log_2 p_i$

$H(Y|X) = \frac{1}{2} * H(Y|X = 1) + \frac{1}{2} * H(Y|X = 0) = H(\alpha, \beta, 1 - \alpha - \beta)$

Problem 3: mod 11 channel capacity

X must be uniformly distributed for maximum capacity. Then Y is also equally distributed. $I(X : Y) = H(Y) - H(Y|X)$. First term is 11 equally distributed outcomes, second is 3 equally distributed outcomes. so it is $\log_2 11 - log_2 3$

Problem 4: channel with memory

$I(X_1...X_n : Y_1...Yn) = H(X_1...X_n) - H(X_1...X_n|Y_1...Y_n)$

But $H(X_1...X_n|Y_1...Y_n) = H(Z_1...Z_n|Y_1...Y_n)$, because knowing Y fixes distributions on X and Z. For example, if Y is 0, then X and Z have same value with same probabilities and their entropy is the same. If we take X-es to be independent and uniformly distributed, first term gives us $H(X_1...X_n) = n * H(1/2, 1/2) = n$. The other term is $H(Z_1...Z_n|Y_1...Y_n) \leq H(Z_1...Z_n) \leq H(Z_1) + ... + H(Zn) = n * H(p, 1-p)$

We get: $I(X_1...X_n : Y_1...Yn) \leq n + n * H(p, 1-p) = n * C$

Problem 5: symmetric channel with many symbols

$I(X : Y) = H(Y) - H(Y|X)$

$H(Y|X) = P(X = x_i) * H(Y|X = x_i) = H(p_1...p_n)$ if X is taken uniformly

Similarly, $H(Y) = -p_i * log_2 p_i$. But each p_i is equal to $\frac{1}{n} * \sum_{k=1}^{n} p_k = \frac{1}{n}$, because each column sums to 1. Adding those terms up, we get the desired formula.

Assignment 16

Problem 1: Number of code words

a) We can choose one word from any 2^n words, and the second is fixed. Doing this we counted each pair twice, so we divide by 2 in the end. Answer: 2^{n-1}

b)To detect 1 error, minimal distance must be 2. We can take code of all words with even number of 1-s. Information rate is $R = \frac{\log_2 m}{n} = \frac{n-1}{n}$

Problem 2: Sphere limits

$d-1=2$, $\frac{d-1}{2}=1$

$V(10,2)=S(10,0)+S(10,1)+S(10,2)=\binom{10}{0}+\binom{10}{1}+\binom{10}{2}=1+10+45=56$

$$\frac{2^n}{V(10,2)} \leq A(10,3) \leq \frac{2^n}{V(10,1)}$$

Problem 3: relations between A(n,d)-s

a)Lets look at all words from A(n,d). More than half of them end with 1-s or all end with 0-s. Their minimal distance is d. If we take those words and remove the last bit, we get [n-1,m,d] code. This m is bigger than half A(n,d)

b) By removing last bit, we can get from A(n,d) to [n-1, m, d-1 or d] code which is $\leq A(n-1,d-1)$ from definition, so $A(n,d) \leq A(n-1,d-1)$. On the other hand, we can go from A(n-1, d-1) to [n,m,d] - add xor of all bit to the end of each code from A(n-1, d-1). If their distance was d-1, which is odd, their checksums will differ, giving minimum distance d. Else, their distance is d anyway. We can see, that their checksums differ, by observing next thing: we can get other word by flipping each bit at a time. Each flip changes checksum, and there are odd number of flips, so the checksum is changed odd number of times. Thus, $A(n-1,d-1) \leq A(n,d)$. We can conclude that $A(n,d)=A(n-1,d-1)$

c) Hamming code has [7, 16, 3]. By using result from b and knowing 16 = A(7,3), we can get [8,16,4] code. By using result from a, we can get [7, m, 4] code, where $m \geq 8$. Removing unnecessary words, we get [7,8,4] code.

Assignment 17

Problem 1: Minimal weights and distances

let w be minimal vector weight. Lets also take two vectors, x and y. Let z = x - y. z is in code and will have zeros in places, where $x[i]=y[i]$ This vector's weight is the same as d(x,y), and we can do that for any x and y. It is more or equal to w, since w is minimum of such distances.

Problem 2: Encoding with linear codes

a) Looking at matrix, we can see, that it takes as argument an n = 3 size words and gives us l = 7 length codes. Lets calculate the first chunk.

Code = 1 * [1 0 0 1 2 0] + 0 * [0 1 0 0 1 1] + 2 * [0 0 1 2 0 1] = [1 0 0 1 2 0] + [0 0 2 1 0 2] = [1 0 2 2 2 2]

b) Minimum distance is less than 3, because vectors in matrix all have weight 3. Taking linear combination of any 2 cant give us less weight, because first 3 digits are linearly independent and that gives 2 at least, and so are the last 3 digits. Same with taking all three, so d = 3

Problem 3: Hamming code matrix

Each row of hamming matrix multiplied by code gives us 0 vector (sum mod 2 is the same as xor). If we got two vectors x and y, then $v * (x + y) = v * x + v * y = 0 + 0 = 0$, so the sum is also in the code

We can get generating matrix by having 4 independent vectors. To get them, we put cyclic rotations of [1 0 0 0] in non parity check parts, and fill the parity check bits later. We get standard form by moving those columns left.

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

Problem 4: Matrix standard form

Divide first row by 2 and move identity matrix columns to the left. Don't forget, that 1/2=2 mod(3)

Assignment 18

Problem 1: Hamming code verification matrix

The matrix we generate by listing all words of length 2^n without zeros is our verification matrix. This can be checked by noticing, that taking $\oplus$ of given vector values is the same as adding them modulo 2 and multiplying other elements by 0. We get generating matrix by putting verification matrix in standard form and transposing non-identity part accordingly.

Verification matrix:

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

Generating matrix:

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

Problem 2: Dual codes

a) For hamming code, dual matrix in standard form is the same thing as hamming verification matrix in standard form, but identity part shifted to the left.

b) We know that its ≤ 4 and we have to check weights of all possible words. Turns out, it's really 4

c)No, we can't do that. In given matrix first three coefficients of row vectors are linearly dependent, and by adding them to each other multiplied by scalar they remain dependent. But in the standard form, they form identity matrix and are independent.

Problem 3: Linear codes decoding

a) Every row of verification matrix has even number of ones, and scalar multiplication by all ones gives us 0, so there are no errors and code = 1111111, word = 1111 b) Multiplying matrix by 1101011 gives us 010, so the error is in the 6'th column and the code = 1101001, word = 1101 c)Multiplying matrix by 0101111 gives us 101, so the error is in the second column and the code = 0001111, word = 0001 d)Multiplying matrix by 1111000 gives us 111, so the error is in the 4'th column and the code = 1110000, word = 1110

To fix 2 errors, we need to multiply matrix by all one and two possible errors and learn syndromes. 5-repeat code has $\binom{5}{0}+\binom{5}{1}+\binom{5}{2} = 1+5+5*2 = 16$ syndromes

Assignment 19

Problem 1: Mixed product generating matrix

We need to get words of type $u * (u + v)$, where $*$ is concatenation. Original word length doubles. Note, that G_0 and G_1 must have equal number of rows, but not columns.

$$\begin{bmatrix} G_0 & G_0 \\ 0 & G_1 \end{bmatrix}$$

b)$RM(4,1) = RM(3,1)|RM(3,0)$
$RM(3,0)$ is, by definition,

$$\begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix}$$

$RM(3,1) = RM(2,1)|RM(2,0)$
$RM(2,1) = RM(1,1)|RM(1,0)$
Let's find $RM(2,1)$
$RM(1,1) is:$

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

$RM(1,0) is:$

$$\begin{bmatrix} 1 & 1 \end{bmatrix}$$

Their Mixed Product:

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

Continuing, we find $RM(3,1)$:

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

We find $RM(3,1)$: the same way.

Problem 2: Number of RM codes

For given q, there are $q + 1$ RM codes, because r goes from 0 to q. We sum all numbers from 1 to 6 and get 21 RM codes.

Each code's efficiency is sum of efficiencies of codes we used to make it, divided by 2^q. In base cases, it is either $1/2^q$ for repeat code or 1 for full code.

$d_{min} = 2^{q-r}$. q - r ranges from 0 to 6. Each code can fix $\lfloor (d - 1)/2 \rfloor$ errors.

Problem 3: RM and hamming

We are given $RM(d, d - 2)$. $dist_{min} = 2^{d-(d-2)} = 2^2 = 4$, 1 more than hamming $n = 2^d$, 1 bit more than hamming code $log_2 m = \sum_{a=0}^{d-2} \binom{d}{a} = 2^d - d - 1$, same amount of words. As we have already seen, adding a $\oplus$ on odd length sequence adds one to the minimal distance and code length. Keeping that in mind, those codes have equal parameters

Problem 4: alternative RM

Prove this by induction on q. If q is equal to r, we have to choose all elements to check for subset. When we do that, we write 1 in corresponding position of that basis vector in matrix. This matrix will have 1's on diagonals, so we can make it into identity matrix by adding linear combinations of other rows and that is exactly what we wanted. On the other hand, if r is 0, we have one basis vector with all ones, and that is exactly H(q,0).

Suppose this is true for some q. We want to show, that it is still true for q+1. We have 1 element more that in q, and subset doubles. Lets call it E. Arrange first half of subset from elements without E, and the other half exactly the same but with E element in each subset. By induction, we want to show that we get something of a form:

$$\begin{bmatrix} G_0 & G_0 \\ 0 & G_1 \end{bmatrix}$$

Indeed, G_0 part is the same by assumption. we arrange rows in such a way, so we pick elements with E last. Thus, the other G_0 is the same too, because we don't care about E so far. Lower left is all zeros, because now we are left with subsets containing E, and first half columns of matrix cannot be 1. Now, every element of second half of columns contains E, so what we really are doing is putting H(q-1, r-1) corresponding matrix in lower right corner. Thus we see, that the basis is exactly the same.

Assignment 20

Problem 1: Cyclic or not

a) No, [1100] + [0011] = [1111], which is not in the code.

b) No, [11011]'s cyclic rotation is [11101], which is not in the code.

c) No, [2211]'s cyclic rotation is [1221], which is not in the code

d) Yes, repeat codes are invariant under cyclic rotation and adding repeat codes gives us repeat code

Problem 2: Dividing polynomials

We are dividing vector [3 0 3 4 0 0 5] = p by [1 3 0 0 4] = q. Multiply q by 3 to get rid of biggest power and subtract. We get [1 3 4 3 0 5]. Multiply by 1, subtract again. We get [0 4 3 1 5]. Multiply by 0. Remainder is [4 3 1 5] = [4 3 1 0], because its length is less than q's length.

Problem 3: GCD

Let's use we want GCD([11001100][111001]). We know, that by dividing bigger one by lesser one, we get a remainder r, and our GCD is equal to GCD([111001], r). We divide and find r = 10001. Now we continue dividing, that gives us GCD([10001][1010]), then again, GCD([1010][101]) and then GCD([101][0]), and that is [101], which happens to be the answer.

Problem 4: Minimal cyclic codes

Let's divide $x^8 - 1$ by [11011000]. We get remainder [101]. We know, that we can get that remainder using sums and cyclical rotations of [11011000]. So we check, if that is our generator. We check, and remainder is 0. So [101] is our generator. We can get [11011000] from it, so we get everything we were getting by [11011000] and vice versa. So m is minimal.

Assignment 21

Problem 1: Prime polynomials

When multiplying two polynomials, say, of degree n and m, we get polynomial of degree n+m. If both n and m are less than d/2 for some d, then their sum is less than d and we cannot get d degree polynomial by their multiplication. So at least one of them is of degree more than or equal to d/2. Then the other is of degree less than or equal to d/2. So if we checked all polynomial with degree less than d/2 and our polynomial is not divisible by them, then it must be prime. On the other hand, if polynomial is prime, it is not divisible by anything but itself and zero degree polynomials.

Problem 2: Counting cyclic codes

a) We can choose to include any prime polynomial and get different minimal generating polynomials in each case, so 2^t is an answer. That includes a code with just all zeros.

b) We need to factor $x^7 - 1$. To do that, we list all prime polynomials up to degree 3 and try to divide by them. Easy factorization is $(x-1)(x^6 + x^5 + x^4 + x^3 + x^2 + x + 1)$. Turns out, second term can be factored ad $(x^3 + x + 1)(x^3 + x^2 + 1)$. Then we can take any combination of those polynomials (7 if not counting all zeros), and quotient is checking polynomial.

c) We need to factor $x^4 - 1$ on F_5. After dividing by all degree one prime polynomials, we get 0. So $x^4 - 1 = (x+1)(x+2)(x+3)(x+4)$. We can make $2^4 - 1$ non-zero generators by choosing subsets of these prime divisors. Counting minimal distance in each case is a daunting task, as we have to generate all words and count minimal weight.

d)Again, $x^8 - 1 = (x-1)(x+1)(x^2+1)(x^2+x-1)(x^2-x-1)$, so there are $2^5 - 1$ codes.

Problem 3: Even weights

a)If $g(x) = h(x)(x+1) = h(x) + x * h(x)$ If h(x) has weight w, and none of the terms cancel out (1+1=0 mod 2), then g(x) will have weight $2 * w$, even number. If k terms cancel each other out, then g(x) will have weight $2*w-2*k$, an even number. On the other hand, if g(x) has even number of zeros, we can divide it by x + 1. We get: $g(x) = h(x)(x+1) + r(x)$, where r(x)'s degree is 0 or it is 0 itself. suppose it is a number : 1. Then, as we have already checked, $h(x)(x+1)$ has even weight, $g(x)$ has even weight by assumption. but we cannot add 1 to polynomial with even weight and get another polynomial with even weight - adding 1 changes weight by 1. So r(x) = 0 and g(x) is divisible by (x+1)

If the code contains all 1's, then that word's weight is odd and it is false, that all code words have even weights. Suppose code does not contain all 1's. $x^n - 1 = (x+1)(1+x+...+x^{n-1})$ Generator is divisible by (x+1), otherwise, it can generate $[1+x+...+x^{n-1}]$, which is all 1's. All words are divisible by generator, so they are divisible by $(x+1)$, so they have even weights

Problem 4: Binomials

$\binom{p}{k} = \frac{p!}{k!\,p-k!}$ Since p is prime, it cannot be cancelled out by anything in denominator, we multiply terms less than p.

Suppose $a(x) = a_0 + a_1 * x . a(x)^n = \binom{p}{0} a_0^p + \binom{p}{1} a_0^{p-1} a_1 x + ... + \binom{p}{p} a_1^p x^p$. All terms but first an last cancel out, because they are 0 mod p. We have $a_0^p + a_1^p x^p$. But all numbers in p-1 'th power are 1, so that is equal to $a_0 + a_1 x^p$. We complete the proof by induction on degree of a(x), using the same method.

Assignment 22

Problem 1: Finding inverse

Use Euclidian algorithm. We divide [1 0 0 1 1] by [1 1 0] and get [1 1 1] and remainder 1. [1 0 0 1 1] is 0, and we can move 1 to the left, because 1 = -1 mod 2. So 1 = [1 1 0] * [1 1 1]. $1 + x + x^2$ is the reverse

Problem 2: Finding inverse

For $[1, \alpha, 1]$, call it $g(x)$, to be a generator, $x^5 - 1$ must divide it. Note, that we write polynomials in order of decreasing degree. At first, $[1, 0, 0, 0, 0, 1]$ divided by $g(x)$ gives us 1. We subtract 1 * $g(x)$ from its highest term, and note, that subtraction is same as addition mod 2. we get $[\alpha, 1, 0, 0, 1]$. Next, we divide again, get α. Subtract α * g(x) from highest term and get $[\alpha, \alpha, 0, 1]$. Note, that $\alpha^2 = \beta$. Repeating the process we get quotient α and remainder $[1, \alpha, 1]$. That is exactly our $g(x)$, so last part of quotient is 1. We get : $g(x) * [1, \alpha, \alpha, 1] = x^5 - 1$. $[1, \alpha, \alpha, 1]$ is checking polynomial.

b) Inserting generating polynomial into matrix from, we get:

$$\begin{bmatrix} 1 & \alpha & 1 & 0 & 0 \\ 0 & 1 & \alpha & 1 & 0 \\ 0 & 0 & 1 & \alpha & 1 \end{bmatrix}$$

Multiplying second row by α and subtracting that from first one, we get:

$$\begin{bmatrix} 1 & 0 & \alpha & \alpha & 0 \\ 0 & 1 & \alpha & 1 & 0 \\ 0 & 0 & 1 & \alpha & 1 \end{bmatrix}$$

Multiplying third row by α and subtracting that from first and second one, we get:

$$\begin{bmatrix} 1 & 0 & 0 & 1 & \alpha \\ 0 & 1 & 0 & \alpha & \alpha \\ 0 & 0 & 1 & \alpha & 1 \end{bmatrix}$$

Minimal distance is 3, so the code can fix one error. First, we see, that each row's weight is 3, and that cannot be changed by multiplying by any scalar (no zeros in multiplication group). Taking all three basis vectors leaves us with first three coefficients being non-zero, because there is identity matrix there and weights are 3 again. Taking any two, we note, that any two rows from last two columns are linearly independent, and give us 1 more weight to the 2 weights from identity matrix part.

Problem 3: Primitive elements

We have to check all powers of each α to be different. We can stop at 5, because if the values did not repeat until then, they will never do so. There are $2^4 - 1 = 15$ elements in the group, and 15 is divisible by each such cycle's length. First four powers will always have different values: $1, \alpha, \alpha^2, \alpha^3$

1) Checking $x^4 + x + 1$. $\alpha^4 = \alpha + 1$. $\alpha^5 = \alpha^2 + \alpha$. That is not 1, so it is primitive element

2) Checking $x^4 + x^3 + 1$. $\alpha^4 = \alpha^3 + 1$. $\alpha^5 = \alpha^4 + \alpha = \alpha^3 + 1 + \alpha$. Also primitive

3) Checking $x^4+x^3+x^2+x+1$. $\alpha^4 = \alpha^3+\alpha^2+\alpha+1$. $\alpha^5 = \alpha^4+\alpha^3+\alpha^2+\alpha = 1$. Not primitive.

b) Every element is α to some power. For element to be primitive, it must be co-prime with number elements in a group, that is, 15. This number is $\varphi(15) = \varphi(5) * \varphi(3) = 4 * 2 = 8$

Problem 4: Orthogonal cyclic codes

Let C be a cyclic code word and O be it's orthogonal code. Orthogonal code is linear, because $c * (a * o_0 + b * o_1) = a * c * o_0 + b * c * o1 = 0 + 0 = 0$. For every o there is o_{+1} in code, where +1 denotes cyclical rotation by 1. To see this, we take some code c from C. c * o = 0, by definition. Then we look at c_{+1}. But multiplying c_{+1} by o_{+1} is same as multiplying c by o. So o_{+1} multiplied by any c from C gives us 0, because c was not fixed in the above example. So O contains o_{+1}, by definition.

b)As we show in next problem, if $g(x) * h(x) = c(x)$, then $inv(g(x)) * inv(h(x)) = inv(c(x))$ so $-x^n + 1 = x^n - 1$, when multiplied by $-1 mod p$, is divisible by $inv(h(x))$ and we can take it as a generator. If g(x) is a number, h(x) = 0, so the scalar product of any linear combinations of them is always 0. Also, 0 is the only number we can multiply scalar by to get zero, so we get all orthogonal C. If g(x) is polynomial of degree k, then h(x) is polynomial of degree n - k. Their scalar product is $g_0 p_{n-1} + g_1 p_{n-2} + ... + g_{n-1} p_0$. That is exactly formula of coefficient of x^{n-1} in their product, which is 0. If we multiply g(x) by x, we get: $x * g(x) * h(x) = x^{n+1} - x$. Scalar product is $(x * g(x))_0 p_{n-1} + .. + (x * g(x))_{n-1} p_0$, which is again a coefficient of x^{n-2} of $x^{n+1} - x$ and is zero. We can conclude, that by multiplying by x and taking linear combinations of g(x) or inv(h(x)), their scalar product will always be zero.

Problem 5: Inverse codes

Polynomial multiplication formula: when $g(x) * p(x) = c(x)$, the coefficient at x^d can be computed as $\sum g_i * p_j$ where i + j = d, for some d. We take all such i and j, and define $p_i = 0$ when i out of bounds. After the inverse, coefficients are updated: $g_i = inv(g)_{k-i}$, where k is g's degree. Let's check, that by multiplying $inv(g)$ by $inv(p)$ we get $inv(c)$. c's degree is k + l, where k = $deg(g)$ and l = $deg(p)$. After the inverse, we get: $\sum inv(g)_i * inv(p)_j$. If i = j sum to $k + l - d$ This is coefficient of x^{k+l-d}, but $inv(g)_i = g_{k-i}$, so we are left with $\sum g_{k-i} * p_{l-j}$, where $k - i$ and $l - j$ sum up to $k + l - d$. But that is the same as taking all i and j that sum up to d. So this is coefficient near power of d in the original c(x)