ECE 409 - ERROR CORRECTING CODES - INVESTIGATION 22
INTRODUCTION TO BLOCK CODES - PART II

SUMMER 2004 **A.P. FELZER**

To do "well" on this investigation you must not only get the right answers but must also do neat, complete and concise writeups that make obvious what each problem is, how you're solving the problem and what your answer is. You also need to include drawings of all circuits as well as appropriate graphs and tables.

In the last Investigation we introduced block codes of size (n,k) and rate k/n. We showed that when a code has a distance d it can detect d-1 errors. Now if the receiver detects an error it can request retransmission (ARQ). Or it can try to correct the error itself. We call this Forward Error Correction (FEC). The main objectives of this Investigation are to introduce FEC and some properties of codes that make FEC easier to implement.

1. The objective of this first problem is to introduce how **Forward Error Correction (FEC)** works. Suppose we have a block code of size (6,3) with the following valid codewords

Data	Codeword
000	000000
100	100110
010	010011
110	110101
001	001101
101	101011
011	011110
111	111000

   a. Calculate the distances between at least five pairs of codewords
   b. How many distances would you have to calculate to verify that $d_{\min} = 3$
   c. How do we know that the received signal $R = 101111$ is not a valid codeword
   d. Which valid codeword is the one most likely to be the real value of R. Explain

2. Generalizing on the result of Problem (1) we have that if we receive an invalid codeword then the valid codeword that was most likely sent is the one with the closest valid code - the one for which there would be the fewest number of errors. Make use of this observation to find how many errors can be found and corrected in a code of distance d
   a. When d is odd
   b. When d is even

3. From the previous two problems it's pretty straightforward to see how an invalid code can be corrected. The problem is that in general we have to check every received codeword with every valid code to find the closest one. How many checks would have to be done in this brute force approach if the size of the code is (30, 20)

4. From Problem (3) we know that the brute force method of decoding a code can be very difficult if not impractical. But if we put restrictions on our codes to keep them "simple" then the encoding and decoding can be greatly simplified. The first restriction we impose on our codes is that they be **systematic**. A systematic code is one in which the data is easily recognizable. This is usually done in an (n,k) code by making the first k bits equal to the data and the last n-k bits equal to the error correction. **Memorize** this definition. Then find a 3-bit code for the following data 00, 01, 10, 11 that is
   a. Systematic - put your result in a Table
   b. Not systematic - put your result in a table

5. Before we specify the next "restriction" on our codes we need to define what we mean by the **sum of two codewords**. When we add two codewords A and B we add each a_i and b_i (mod 2) as follows

$$(a_i + b_i) \text{ (mod 2)} = \text{remainder after dividing } a_i + b_i \text{ by 2}$$

as indicated in the following table of sums (mod 2)

	0	1
0	0	1
1	1	0

Memorize how codewords are added. Then

a. Make use of the table above to show that $(a_i + b_i)(\text{mod } 2) = a_i \quad b_i$

b. Find the following sum of codewords A+B = 100110101+ 010100110

6. The second restriction we impose on our codes to keep them "simple" is **linearity**. A code is linear if the sum of any two valid codewords is also a valid codeword. **Memorize** this definition. Then

a. Do at least several calculations to show that the code in Problem (1) is linear

b. Show that linear codes must contain the zero codeword 00 . . . 0. Hint - what happens when we add a codeword to itself

7. Show that the even parity code of size (2, 1) is linear

8. Generalize on Problem (7) to show that all even parity codes of size (n, n-1) are linear. Consider the cases of an odd number of 1's and an even number of 1's

9. Are odd parity codes of size (n, n-1) linear. How do you know

10. The objective of this and the next two problems is to show an easy way to find the distance of a linear code. But first we must define what we mean by the **weight** of a codeword. The weight of a codeword is simply the number of ones.

a. What is the weight of A=10011010

b. Find a codeword of size (8,5) of weight 3

11. Given a code

a. Do some examples to illustrate the fact that the distance between two codewords is the weight of their sum

b. Explain why the distance between two codewords is the weight of their sum. **Memorize** this result.

12. Show that the distance d of a linear code is equal to the weight of the codeword with the smallest nonzero weight - which is equal to the distance between that codeword and the zero codeword. Hint - consider distances of all pairs

13. Make use of the result in Problem (12) to find the distance d of the linear code in Problem (1)