16.36 Communication Systems Engineering
Spring 2009

MASSACHUSETTS INSTITUTE OF TECHNOLOGY
Department of Aeronautics and Astronautics

16.36: Comm. Sys. Engineering
Problem Set No. 7

Problem 1: Text problem 9.1

Problem 2: Text problem 9.8

Problem 3: Text problem 9.27

Problem 4:

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

The generator matrix for a (6,3) code is given above.

A) find the minimum distance for the code
B) Find the parity check matrix for the code
C) What codeword would you use to encode 111?
D) Suppose you receive 111111, how would you decode it?

Problem 5: Matlab Exercise

In this exercise, you will create a linear block code encoder and decoder function, test it for antipodal signaling with your PAM modulator and demodulator, and compare the error performance to not encoding.

A) You will create two separate functions: `encode` and `decode`
   a. Implement the (5,2) code that was presented in the textbook in examples 9.5.1, 9.5.2, 9.5.3, and 9.5.7.
   b. The decoder should implement hard decision decoding using syndrome decoding, as presented class and shown in example 9.5.7.
   c. For each of your functions, the input and output should be a bit string. This will allow you to more easily integrate your new functions with preexisting ones.
   d. You may find the following Matlab functions useful (for more information, look at the Matlab help files)
      i. `bin2dec`: will take a binary string and convert it to decimal: `'0'`→`0`, `'10'`→`2`
      ii. `dec2bin`: will take a decimal value and create a bitstring. This function has also can take as an input the minimum number of bits to output. `dec2bin(1,2)='01'`
      iii. `mod`: modulo operator (% in C or java). All the arithmetic that is being preformed for coding is modulo-2: 1+1=0. An example use of the function is `mod(4,2)=0` and `mod(3,2)=1`. You can also apply the function to an entire array.
      iv. `xor`: bit-wise xor, also denoted by ⊕, is a modulo-2 addition or subtraction operator, with no carry. `xor(1,1)=0` and `xor(1,0)=1`.

B) If you construct the input and outputs of your encoder and decoder properly, you will not need to make any changes to any of your existing functions.

C) Run the same test as Test 1 from the last homework, but this time encode and decode the data
   a. Add noise with σ ranging from 0.4 to 1.0, increasing in 0.1 increments. Plot your observed error rate, as well as the plot of the theoretical bit error rate (using the same Q function, `qfunc`, as the last homework) of unencoded data. Run this simulation for $d = 2$. Plot the simulated encoded error rate and the theoretical single-bit error rate on a single plot and label each line.
   b. You should be able to reuse almost all of your code from the previous homework.

D) Use the following inputs for your tests
   a. Carrier frequency, fc, of 1 Hz.
   b. Sampling frequency, fs, of 4 Hz.
   c. Symbol rate, Rs, of 1.
   d. Bit string size of $n = 5000$, pre-encoded.
   e. Antipodal signaling: $M = 2$.
   f. Use even bit distribution, $p=0.5$, for `rand_bitstring`.

E) Please produce as output the plotted error rates. Make sure to comment all of your code and clearly label plots.