EECS 455: Problem Set 9
Submit via Gradescope via link on Canvas

Due: Wednesday, December 1, 2021, 11pm.

1. The (15,11) Hamming code has the following parity check matrix.

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

The received vector is $[000000000000011]$. Determine the most likely transmitted codeword.

2. A communication system uses a (7,4) Hamming code with FSK modulation with noncoherent reception. The channel attenuates the signal by a Rayleigh random variable.

$$f_X(x) = \begin{cases} 2xe^{-x^2}, & x \geq 0 \\ 0, & x < 0 \end{cases}$$

Assume each bit is attenuated independently. Determine the bit error probability with and without coding. Plot the bit error rate (BER) on a log scale as a function of E_b/N_0 in dB on a scale down to 10^{-5} BER. Determine the "coding gain" (reduction in energy possible compared to an uncoded system) at BER of 10^{-5}. Be sure to normalize the energy per coded bit and the energy per information bit appropriately.

3. Consider a communication system that transmits one of 16 signals in 8 dimensions. The signals are based on 8-PSK modulation and correspond to the following sequence of 4 phases where m indicates a phase of $m(2\pi)/8 = m\pi/4$. That is a phase index of 1 corresponds to phase $\pi/4$, a phase index of 2 corresponds to phase $\pi/2$, a phase index of 3 corresponds to $3\pi/4$.

Bits	signals	Phase Index	Bits	signals	Phase Index
0000	s_0	(0,0,0,0)	1000	s_8	(2,5,2,0)
0001	s_1	(0,4,0,0)	1001	s_9	(2,1,2,0)
0010	s_2	(0,2,5,2)	1010	s_{10}	(2,3,7,2)
0011	s_3	(0,6,5,2)	1011	s_{11}	(2,7,7,2)
0100	s_4	(4,0,0,0)	1100	s_{12}	(6,5,2,0)
0101	s_5	(4,4,0,0)	1101	s_{13}	(6,1,2,0)
0110	s_6	(4,2,5,2)	1110	s_{14}	(6,3,7,2)
0111	s_7	(4,6,5,2)	1111	s_{15}	(6,7,7,2)

A signal with "phase index" 0 corresponds to a constellation point $\sqrt{E}\exp(j0)$ where the orthonormal signals are $\sqrt{2/T}\cos(2\pi f_c t)p_T(t)$ and $-\sqrt{2/T}\sin(2\pi f_c t)p_T(t)$. The real waveform is for phase index 0 is $\sqrt{2P}\cos(2\pi f_c t)p_T(t)$. A signal with phase index 1 corresponds to the constellation point $\sqrt{E}\exp(j2\pi/8)$ or a waveform $\sqrt{2P}\cos(2\pi f_c t + 2\pi/8)p_T(t)$ A signal with phase index i corresponds to $\sqrt{E}\exp(j2i\pi/8)$ or a signal $\sqrt{2P}\cos(2\pi f_c t + 2\pi i/8)p_T(t)$.

So signal s_{10} actually corresponds to transmitting a signal of duration $4T$ which is the represented by the vector $\sqrt{E}(\exp(j2\pi 2/8), \exp(j2\pi 3/8), \exp(j2\pi 7/8), \exp(j2\pi 2/8))$. In the first T seconds the signal transmitted is a sinusoid with phase $2\pi/8$. In the second T seconds the signal is a sinusoid with pahse $3\pi/8$ and so on. Four bits are transmitted each with energy $4E$ where E is the energy for each 8-PSK modulation. So the energy per bit is

$$E_b = 4E/4 = E.$$

(a) Describe the optimum receiver for minimizing the probability of choosing the wrong signal.

(b) Simulate the system down to error probabilities of 10^{-3}. That is, generate all the signals in Matlab. Choose one of the 16 at random to transmit. Add noise that is Gaussian, mean 0, variance $N_0/2$ in each dimension. Find the closest signal to the received signal. Decide if there is an error or if the decision is correct. Repeat this many times (e.g. 100,000 times) and count how many errors occurred. Then divide by the number of simulations to get an estimate of the probability of error. Do this for various value of E_b/N_0 and plot the results on a log scale for the vertical (probability of error) and linear scale for the horizontal (E_b/N_0 in dB).

(c) Determine the pairwise distance between all signals (you can use Matlab for this).

(d) Determine the Union Bound on the probaiblity of signal error (choosing the wrong signal).

Plot the simulation results and the union bound of error probability on the same plot with the horizontal axis being E_b/N_0 in dB.

```
clear all;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                                                         %
%               Simulation Parameters                     %
%                                                         %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


ncount=input('Number of errors = ');

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                                                         %
%               Setup the Simulation                      %
%                                                         %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
E=1
Eb=E;                   % Relation between energy per code symbol and bit
s(1,:)  =  sqrt(E)*[ exp(j*2*pi*0/8), exp(j*2*pi*0/8), exp(j*2*pi*0/8), exp(j*2*pi*0/8) ];
s(2,:)  =  sqrt(E)*[ exp(j*2*pi*0/8), exp(j*2*pi*4/8), exp(j*2*pi*0/8), exp(j*2*pi*0/8) ];
s(3,:)  =  sqrt(E)*[ exp(j*2*pi*0/8), exp(j*2*pi*2/8), exp(j*2*pi*5/8), exp(j*2*pi*2/8) ];
s(4,:)  =  sqrt(E)*[ exp(j*2*pi*0/8), exp(j*2*pi*6/8), exp(j*2*pi*5/8), exp(j*2*pi*2/8) ];
s(5,:)  =  sqrt(E)*[ exp(j*2*pi*4/8), exp(j*2*pi*0/8), exp(j*2*pi*0/8), exp(j*2*pi*0/8) ];
s(6,:)  =  sqrt(E)*[ exp(j*2*pi*4/8), exp(j*2*pi*4/8), exp(j*2*pi*0/8), exp(j*2*pi*0/8) ];
s(7,:)  =  sqrt(E)*[ exp(j*2*pi*4/8), exp(j*2*pi*2/8), exp(j*2*pi*5/8), exp(j*2*pi*2/8) ];
s(8,:)  =  sqrt(E)*[ exp(j*2*pi*4/8), exp(j*2*pi*6/8), exp(j*2*pi*5/8), exp(j*2*pi*2/8) ];
s(9,:)  =  sqrt(E)*[ exp(j*2*pi*2/8), exp(j*2*pi*5/8), exp(j*2*pi*2/8), exp(j*2*pi*0/8) ];
s(10,:)  =  sqrt(E)*[ exp(j*2*pi*2/8), exp(j*2*pi*1/8), exp(j*2*pi*2/8), exp(j*2*pi*0/8) ];
s(11,:)  =  sqrt(E)*[ exp(j*2*pi*2/8), exp(j*2*pi*3/8), exp(j*2*pi*7/8), exp(j*2*pi*2/8) ];
s(12,:)  =  sqrt(E)*[ exp(j*2*pi*2/8), exp(j*2*pi*7/8), exp(j*2*pi*7/8), exp(j*2*pi*2/8) ];
s(13,:)  =  sqrt(E)*[ exp(j*2*pi*6/8), exp(j*2*pi*5/8), exp(j*2*pi*2/8), exp(j*2*pi*0/8) ];
s(14,:)  =  sqrt(E)*[ exp(j*2*pi*6/8), exp(j*2*pi*1/8), exp(j*2*pi*2/8), exp(j*2*pi*0/8) ];
s(15,:)  =  sqrt(E)*[ exp(j*2*pi*6/8), exp(j*2*pi*3/8), exp(j*2*pi*7/8), exp(j*2*pi*2/8) ];
s(16,:)  =  sqrt(E)*[ exp(j*2*pi*6/8), exp(j*2*pi*7/8), exp(j*2*pi*7/8), exp(j*2*pi*2/8) ];

%==================================================================================%
%                                                                                  %
%                           Compute the Union Bound First                          %
%                                                                                  %
%==================================================================================%
for m=1:35
EbN0dB(m)=-4+(m-1)/2
```

```
EbN0(m)=10^(EbN0dB(m)/10);
N0=Eb/EbN0(m);            % Noise power
sigma=sqrt(N0/2);

%=========================================================================================%
%                                                                                         %
%   YOUR CODE FOR UNION BOUND GOES HERE                                                   %
%                                                                                         %
%=========================================================================================%

%  pes_ub(m)= .... this is your result for the union bound

end
semilogy(EbN0dB,pes_ub,'r','LineWidth',2)    % PLOT THE RESULTS
hold on

clear EbN0dB
%=========================================================================================%
%                                                                                         %
%                           Now do the simulation                                         %
%                                                                                         %
%=========================================================================================%

for m=1:25
EbN0dB(m)=-4+(m-1)/2
EbN0(m)=10^(EbN0dB(m)/10);
N0=Eb/EbN0(m);            % Noise power
sigma=sqrt(N0/2);


%=========================================================================================%

allsignals=s;
nberrors=0;
nserrors=0;
nbsim=0;
nsim=0;
while(or((nserrors < ncount),(nbsim < 100000)))  % THIS CONDITION DETERMINES HOW MANY SIMULATIONS YOU RUN


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                                                        %
%              Generate data and signals                 %
%                                                        %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

b=round(rand(1,4));
index=(b(4)+2*b(3)+4*b(2)+8*b(1))+1;
strans=allsignals(index,:);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                                                        %
%                  Add Noise                             %
%                                                        %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

noise=sigma*randn(1,4)+j*sigma*randn(1,4);
rcvd=strans+noise;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                                                        %
%           Decode the received signal               %
%                                                        %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%



%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                                                        %
%        PUT YOUR CODE HERE FOR DECODING THE SIGNAL      %
%        AND COUNTING THE NUMBER OF ERRORS               %
%        nberrors is number of bit errors                %
%        nserrors is number of symbol errors             %
%                                                        %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

end

peb(m)=nberrors/nbsim;
pes(m)=nserrors/nsim;
end
semilogy(EbN0dB,pes,'LineWidth',2)
axis([-4 12 0.00000001 1])
grid on
hold on
xlabel('E_b/N_0 (dB)','FontSize',16)
ylabel('P_{e,b}','FontSize',16)
```

4. Consider the following code with $M = 4$ codewords of length $n = 6$. The code is used on a

binary symmetric channel with crossover probability p.

$$\begin{array}{c} 000000 \\ 000111 \\ 111000 \\ 111111 \end{array}$$

(a) Find the minimum distance of this code.

(b) Determine the guaranteed error correcting capability of the code. That is, find the maximum number of errors that the optimum decoder can always correct no matter where the errors are located.

(c) Consider a bounded distance decoder that decodes to a codeword only if the (Hamming) distance between the received vector is less than or equal to the error correcting capability. Find the number of vectors in each decoding region of the bounded distance decoder.

(d) What fraction of the total received vectors are in the decoding region of some codeword for the bounded distance decoder.

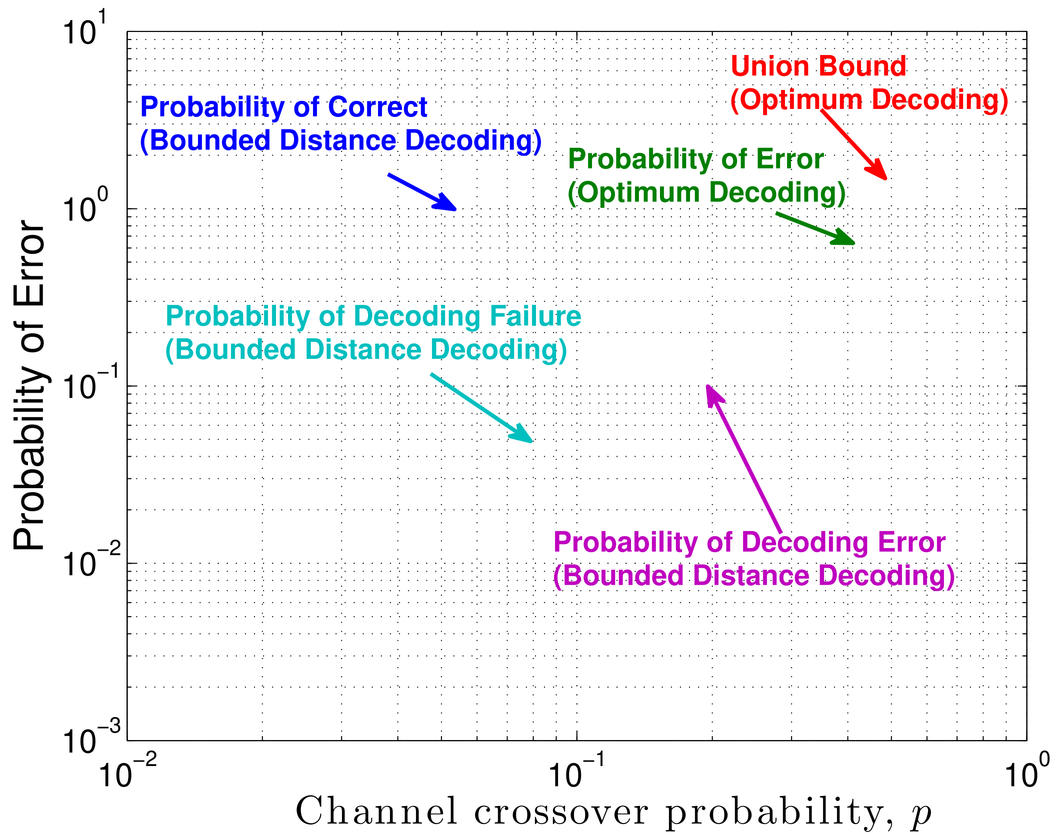
(e) Find a received vector that is not in the decoding region of any codeword for the bounded distance decoder and find the codeword that an optimum decoder would choose for that received vector. This must be a received vector with distance to a codeword greater than the guaranteed error correcting capability.

(f) Find an exact expression for the probability of error for the optimum decoder (the decoder that minimizes the probability of choosing the wrong codeword, not the bounded distance decoder). This is not the union bound.

(g) Find the probability of a received vector being in the decoding region of the codeword (000111) given that the codeword (000000) was transmitted for the bounded distance decoder.

(h) Plot on a log-log scale the probability of error for the optimum decoder, the probability of error for a bounded distance decoder that only attempts to correct one error, the probability of failure for a bounded distance decoder and the probability of correct for the bounded distance decoder. The (log-log) plot should have x scale from 0.01 to 1 and the y scale from 0.001 to 10. The only reason to plot up to 10 on the y scale is to make room for labels and the fact that the union bound will be larger than 1 at some point.

The plot should have 5 curves and look like the plot below (except that all the curves are missing).


Probability of Error
Channel crossover probability, p
Union Bound
(Optimum Decoding)
Probability of Correct
(Bounded Distance Decoding)
Probability of Error
(Optimum Decoding)
Probability of Decoding Failure
(Bounded Distance Decoding)
Probability of Decoding Error
(Bounded Distance Decoding)
10^1
10^0
10^−1
10^−2
10^−3
10^−2
10^−1
10^0