EECS 455: Solutions to Problem Set 9

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.

   Solution: Since the syndrome is easily seen to be $[0011]^T$ the position where the error occurred is the 11th position. That is the error vector is $[000000000010000]$. So the transmitted codeword most likely is $[000000000010011]$.

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.

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

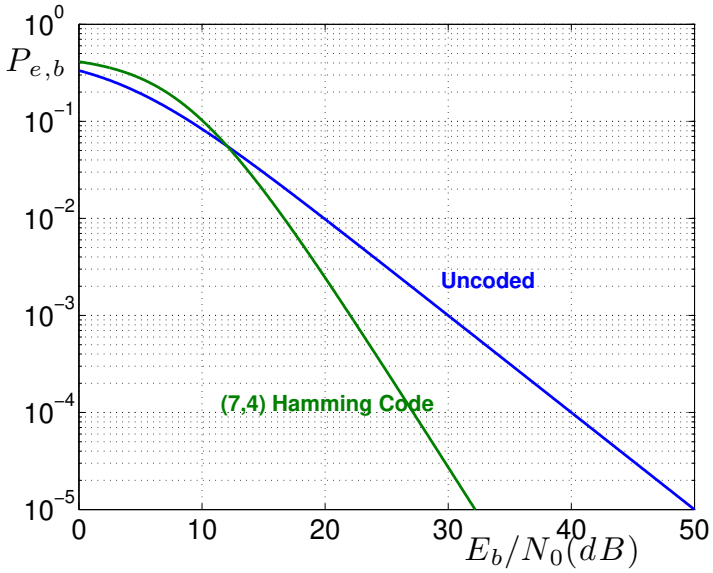
   Determine the bit error probability with and without coding. Plot the bit error rate (BER) on a log scale as a function of E_b/N_0 in dB on a scale down to 10^{-5} BER. Determine the "coding gain" (reduction in energy possible compared to an uncoded system) at BER of 10^{-5}. Be sure to normalize the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

Solution: The bit error probability without coding is

$$p = \frac{1}{2+E/N_0}$$

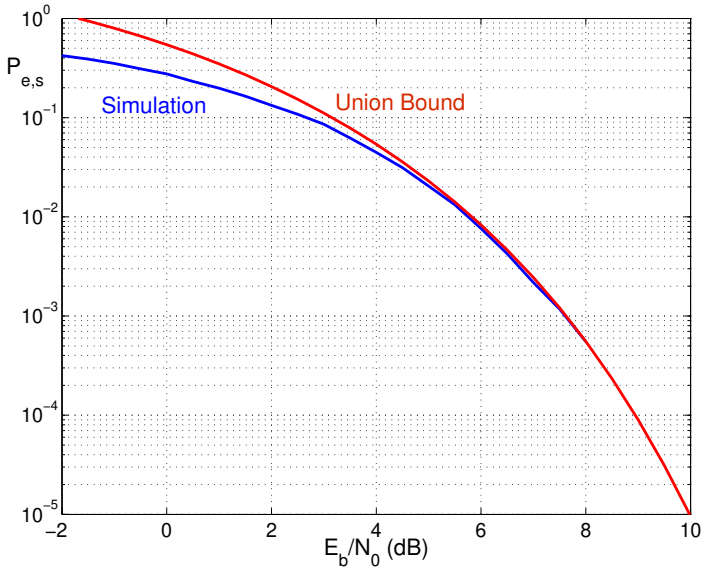
where E is the energy used per bit. If we do not do any coding then $E = E_b$. For the case of the (7,4) Hamming code the energy per bit is $E_b = 7E/4$ or $E = 4E_b/7$. The error probability is

$$P_b = 9p^2(1-p)^5 + 19p^3(1-p)^4 + 16p^4(1-p)^3 + 12p^5(1-p)^2 + 7p^6(1-p) + p^7$$



The difference in signal-to-noise ratio at error probability of 10^{-5} is 17.8dB =50-32.2.

3. The following code finds the union bound and simulates the performance. The plot of error probability is given below.




```
%  This program simulates a 16 signals in 8 dimensions.
```

```
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) ];

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);


%==========================================================================================%
%                                                                                          %
%                           Compute the Union Bound First                                  %
%                                                                                          %
%==========================================================================================%
pes_ub(m)=0
for i=1:16
pe(i)=0;
for l=1:16
d(i,l)=sqrt(sum((abs(s(i,:)-s(l,:))).^2));

if (l ~= i) pe(i)=pe(i)+q(d(i,l)/(2*sigma)); end
end
pes_ub(m)=pes_ub(m)+pe(i)/16
end
end
semilogy(EbN0dB,pes_ub,'r','LineWidth',2)
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)))


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                                                         %
%              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              %
%                                                           %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

[corr,l]=max(real(rcvd*(allsignals')));
lhat=l-1;
nserrors=nserrors+(l~=index);
bhat(4)=mod(lhat,2);
lhat=floor(lhat/2);
bhat(3)=mod(lhat,2);
lhat=floor(lhat/2);
bhat(2)=mod(lhat,2);
lhat=floor(lhat/2);
bhat(1)=mod(lhat,2);
nberrors=nberrors+sum(abs(sign(bhat-b)));
nsim=nsim+1;
nbsim=nbsim+4;
if (mod(nbsim,5000)==0)
peb(m)=nberrors/nbsim;
pes(m)=nserrors/nsim;
format('long')
[EbN0dB(m),nberrors,nbsim,peb(m),nserrors,pes(m)]
        save s129data
end;
end

peb(m)=nberrors/nbsim;
pes(m)=nserrors/nsim;
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)
save s129data
end
```

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.

Solution: The minimum distance is 3.

(b) Determine the guranteed 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.

Solution: The code can correct any pattern of one error.

(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.

Solution: The decoding region of a bounded distance decoder contains the codeword and all 6 single error patterns. Thus there are seven vectors in each decoding region.

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

Solution: The fraction of vectors in the decoding region of some codeword is 28/64.

(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. This must be a received vector with distance to a codeword greater than the guaranteed error correcting capability.

Solution: The vector 010010 is distance 2 to the codeword 000000, distance 3 to codeword 000111, distance 3 to codeword 111000 and distance 4 to codeword 111111 so it is not in the decoding region of any codeword for a bounded distance decoder.

(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).

Solution: The optimum decoder will decoded the first three bits and the second three bits separately. The probability of error for the first three bits is the probability of more than one error in the first three bits. Alternatively the decoder will not make an error if the number of errors is 1 or fewer in the first three coded bits AND then number of errors is 1 or fewer in the last three coded bits. The probability of more than 1 error in 3 bits is.

$$P_{e,2} = P_{e,1} = 3p^2(1-p) + p^3.$$

Thus

$$\begin{aligned} P_e &= 1-(1-P_{e,1})(1-P_{e,2}) \\ &= P_{e,1} + P_{e,2} - P_{e,1}P_{e,2} \\ &= 6p^2(1-p) + 2p^3 - (3p^2(1-p)+p^3)^2 \end{aligned}$$

Alternatively there are 6 patterns of 2 errors that will cause the received vector to be closer to one of the wrong codewords (the second codeword or the third codeword). Either 2 errors in the first three bits or 2 errors in the last three bits. There are 3 ways for each of those to happen. There are 20 ways for 3 errors to cause the received vector to be closer to a wrong codeword. Three errors all in the last 3 bits (one way), or 3 errors all in the first three bits (one way) or 2 errors in the first three and 1 in the last (9 ways) and 1 in the first three and 2 in the last three (9 ways). There are 15 ways to have 4 errors cause the received vector to be closer to one of the wrong codewords. One error in the first three positions and three errors in last 3 positions (3 ways), 2 errors in the first three positions and 2 errors in last 3 positions (9 ways), three errors in first three positions and one error in last 3 positions (3 ways). There are 6 ways for 5 errors to cause the received vector to be closer to a wrong codeword. Any pattern of 5 errors will make the received vector closer to the last codeword (111111). Similarly there is one way for 6 errors to occur. Thus

$$P_e = 6p^2(1-p)^4 + 20p^3(1-p)^3 + 15p^4(1-p)^2 + 6p^5(1-p)^6 + p^6.$$

These two expressions are the same.

(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.

Solution: The possible received vectors that are in the bounded distance decoding region of (000111) are

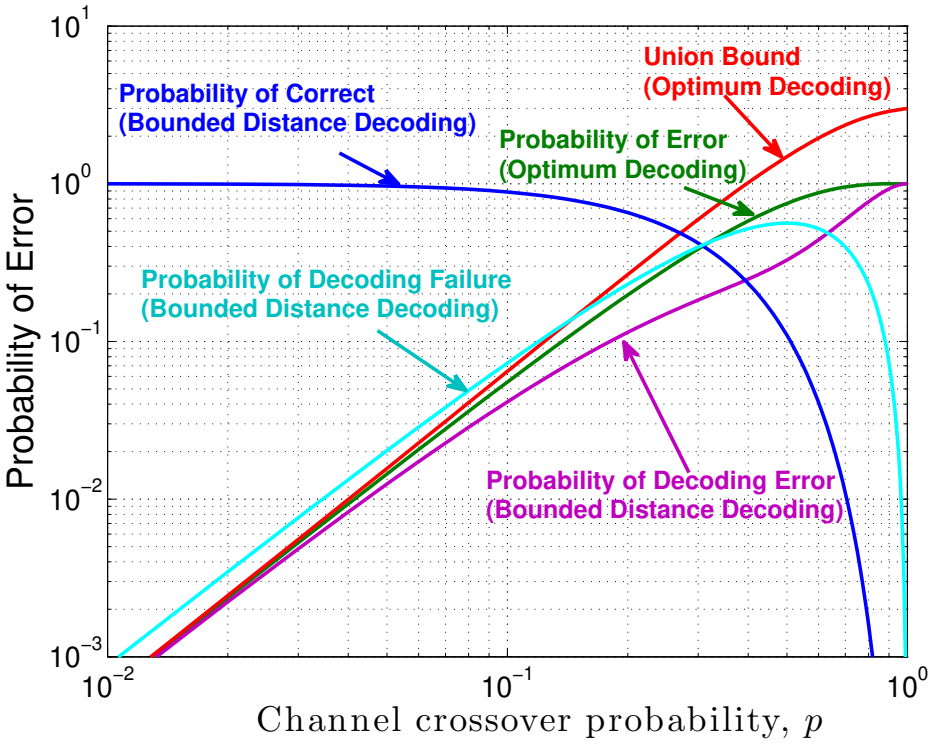
100111
010111
001111
000011
000101
000110
000111

There are three vectors that correspond to 2 errors, three vectors that correspond to 3 errors and 3 vectors that correspond to 4 errors. Thus the probability of decoding to the codeword (000111) with bounded distance decoding is

$$P_{e,bd} = 3p^2(1-p)^4 + p^3(1-p)^3 + 3p^4(1-p)^2$$

(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 1.

Solution:


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)
Probability of Error
Channel crossover probability, p
10^{1}
10^{0}
10^{-1}
10^{-2}
10^{-3}
10^{-2}
10^{-1}
10^{0}