Introduction to Channel Coding

D. Hemanth Kumar and G V V Sharma*

Problem 1. Type the following code and execute using python2.7

```
import numpy as np
from scipy import special
import matplotlib.pyplot as plt

#Block Length
n = 7

#Systematic Bits
k = 4

#Code Rate
R = float(k)/n

#Number of bits
N = int(2e6)

#Range of SNR
snrlen = 11

#SNR for uncoded system
Eb_N0_dB = np.arange(0,snrlen)

#SNR for coded system
Ec_N0_dB=Eb_N0_dB - 10*np.log10(1/R)

#Parity Matrix
h=np.matrix([[1,0,1],[1,1,1],[1,1,0],[0,1,1]])

#Generator Matrix for Encoding
g = np.column_stack((np.eye(4), h))


#Parity Check Matrix for Decoding
ht = np.row_stack((h,np.eye(3)))
```

*The authors are with the Department of Electrical Engineering, Indian Institute of Technology, Hyderabad 502285 India e-mail: {ee16mtech11008,gadepall}@iith.ac.in.

```
#Codebook
c_vec = np.zeros((2**k,n))

#nErr_hard=np.zeros((1,snrlen))

nErr_soft=np.zeros((1,snrlen))

#Encoder

for kk in range(2**k):
    m_vec=np.matrix(map(int,np.binary_repr(kk,width=k)))
    c_vec[kk,:]=(m_vec*g)%2


for yy in range(len(Eb_N0_dB)):

    #trasmitter
    ip = np.random.randint(2,size=N) #generating 0,1 with equa
       probability

    ip=np.array(ip)
        #hamming coding (7,4)
    ipM=np.matrix(np.reshape(ip,(-1,4)))
    ipC=(ipM*g)%2
    cip=np.reshape(ipC,(1,(N/4)*7))

        #modulation
    s=2*cip-1      #BPSK modulation 0 -> -1; 1 -> 0

        #channel-AWGN


    sigma = np.sqrt((1/2.0)*(10**(-Ec_N0_dB[yy]/10)))
    noise=np.random.normal(0,sigma,(np.shape(cip)))

        #noise addition
    y=s+noise

    y=np.array(y)
       #receiver

    #    Soft decision Hamming decoder
    cipSoftM     = np.reshape(np.real(y),(-1,n))
    c_vec=np.matrix(c_vec)
    corr=cipSoftM*(2*c_vec.T-1)
    idx=corr.argmax(axis=1)
    ipHat_soft=[]
    for i1 in range(np.shape(idx)[0]):
        aa=list(np.binary_repr(idx[i1,0],width=k))
        for j1 in range(k):
```

```
            ipHat_soft.append(int(aa[j1]))
    ipHat_soft =np.array(ipHat_soft)

    #    counting the errors


    nErr_soft[0,yy] = np.count_nonzero(ip-ipHat_soft)

theoryBer = 0.5*special.erfc(np.sqrt(10**(Eb_N0_dB/10.0))) #
    theoretical ber uncoded AWGN

simBer_soft     = nErr_soft/float(N)

plt.plot(Eb_N0_dB,theoryBer,'b',Eb_N0_dB,simBer_soft[0],'g')
plt.legend(['theory-Uncoded','coded-soft'],loc=1)
plt.yscale('log')
plt.ylabel('Bit Error Rate')
plt.xlabel('Eb/N0 in dB')
plt.title('BER for BPSK in AWGN with hamming(7,4) code')
plt.xticks([0,1,2,3,4,5,6,7,8,9,10,11])
plt.grid()


plt.show()
```

Find the size of the matrix.

The following questions are based on the above code.

Problem 2. Find the number of input and output bits.

Problem 3. Find the rate of the code.

Problem 4. Find the relation between the coded and uncoded SNR.

Problem 5. Print the parity matrix P.

Problem 6. Print the *systematic* generator matrix G.

Problem 7. Print the parity check matrix H.

Problem 8. Generate the binary code from a decimal number between 0 to 15.

Problem 9. Generate a 16×4 matrix M whose rows are the 4 bit binary codes for the numbers from 0 to 15.

Problem 10. Find MG.

Problem 11. Generate random information bits 0 and 1 with equal probability.

Problem 12. Generate an $\frac{N}{4} \times 4$ matrix M_T from the above random information bits by stacking set of 4 bits starting from the beginning row wise.

Problem 13. Find $T_M G$

Problem 14. Reshape $T_M G$ to generate a bitstream of size $\frac{7N}{4}$.

Problem 15. Modulate the above bitstream using BPSK. Convert 0 to -1 and 1 to +1.

Problem 16. Add AWGN with appropriate SNR. This will give you the received symbols.

Problem 17. How will you detect your transmitted symbols?