**convolution**

%convolution code;

clear all;

close all;

msg=input("Enter input msg bits:-");

l=input("Enter the constrain length:-");

n=input("Enter the no. of output:-");

for j=1:n

g (j)=input("Enter the generator polynomial in octal:-");

end

trellis=poly2trellis(l,g);

disp(trellis);

[isok,status]=istrellis(trellis);

if(isok==0);

disp("Terris is not valid");

end

code=convenc(msg,trellis);

disp(code);

rcode=input("Enter the recieved code");

tablen=3;

decoded=vitdec(rcode,trellis,tablen,"trunc","Hard");

disp(decoded)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

 command keys

Enter input msg bits:-[1 0 0 1]  
Enter the constrain length:-3  
Enter the no. of output:-2  
Enter the generator polynomial in octal:-5  
Enter the generator polynomial in octal:-6  
Enter the recieved code [1 0 0 1 1 0 1 1]

-------------------------------------------------------------------------------------------------------------------

**Huffman**

clear all;close all;clc;

n=input('Enter number of symbols generated by source');

p=input('Enter probability of symbols generated by source');

E=-1\*sum(p.\*log2(p));

disp('Entropy H(X):')

disp(E);

s=1:n;

[dict,avglen]=huffmandict(s,p);

disp('Huffman Dictionary');

disp(dict);

disp('Huffman bit length');

disp(avglen);

% temp=dict;

% for i=1:length(temp)

% temp{i,2}=num2str(temp{i,2});

% end

% disp(temp);

sig=input('Enter transmitted symbol');

disp('Huffman codes for given messages');

sc=huffmanenco(sig,dict);

disp(sc);

disp('Recieved bit string');

mr=sc;

mo=huffmandeco(mr,dict);

disp(mo);

flag=isequal(sig,mo);

if flag==1

disp('Message recovered correctly');

disp('');

eff=(E/avglen)\*100;

disp('Efficiency');

disp(eff);

red=100-eff;

disp('Redudancy');

disp(red);

else

disp('Error in decoding');

end

---------------------

command windos

Enter number of symbols generated by source  6  
Enter probability of symbols generated by source [0.4,0.19,0.16,0.15,0.08,0.02]  
Enter transmitted symbol[1 2 3 4 5 6 ]

----------------------------------------------------------------------------------------------------------------------------

**Entropy**

close all;

clc;

clear all;

px(1)=input('enter probability of x1 = ');

px(2)=1-px(1);

py=[0 0];

p=input('Enter conditional probability P(y1/x1)= p = ');

pybx= [p 1-p ; 1-p p]

for i=1:2

for j=1:2

pxy(i,j)=(px(i)\*pybx(i,j));

end

end

py=[0 0];

for i=1:2

for j=1:2

py(i)= py(i)+pxy(j,i);

end

end

hx=0;

for i=1:2

h=px(i)\*(log(1/px(i))/log(2));

hx=hx+h;

end

hy=0;

for i=1:2

y=py(i)\*(log(1/py(i))/log(2));

hy=hy+y;

end

Hxy=0;

for i=1:2

for j=1:2

if pxy(i,j)==0

hxy=0;

else

hxy=-pxy(i,j)\*log2(pxy(i,j));

Hxy=Hxy+hxy;

end

end

end

Hygx=Hxy-hx;

Ixy=hy-Hygx;

Hxgy=Hxy-hy;

if pybx(1,1)~=1

c=1+[p\*log2(p)+(1-p)\*log2(1-p)]

elseif p==1

c=log2(2);

end

z= [' Input probabilities are = ',num2str(px)];

disp(z)

disp(' Channel conditional probability matrix is = ');

pybx

disp(' Channel joint probability matrix is =');

pxy

z= [' Output probabilities P(y1) and P(y2) are = ',num2str(py)];

disp(z)

z= [' Entropy of source H(X) = ',num2str(hx), ' bits/msg' ];

disp(z)

z= [' Entropy of destination H(Y) = ',num2str(hy), 'bits/msg' ];

disp(z)

z= [' Conditional entropy H(Y/X) = ',num2str(Hygx), 'bits/msg' ];

disp(z)

z= ['Joint entropy H(X,Y) = ',num2str(Hxy), 'bits/msg' ];

disp(z)

z= [' Conditional probability H(X/Y) = ',num2str(Hxgy), 'bits/msg' ];

disp(z)

z= [' Mutual information I(X,Y) = ',num2str(Ixy), 'bits' ];

disp(z)

z= [' The Channel capacity of channel is C = ',num2str(c), 'bits/sec' ];

disp(z)

-----------------------

command window

enter probability of x1 = 3/4  
Enter conditional probability P(y1/x1)= p = 2/3

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**MQAM1**

% clc;

close all; clear all;

M = 16;

%Generate random data symbols.

data = randi([0 M-1],1000,1);

%Modulate the data symbols.

txSig = qammod(data,M);

%Pass the signal through a noisy channel.

scatterplot(txSig)

title('Transmited signal');

title('Transmitter 64-QAM constellation');

for snr=1:5:30

rxSig = awgn(txSig,snr);

%Plot the constellation diagram.

scatterplot(rxSig);

title(['Recieved signal with SNR=',num2str(snr),]);

end;

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**MQAM2**

close all; clear all; clc;

len = 10000; % Number of symbols

M =[4 16 64]; % Size of alphabet

k=1

for m=1:3

% Modulate using both PSK and PAM,

% to compare the two methods.

msg = randi([0 M(m)-1],len,1); % Original signal

txpsk = qammod(msg,M(m));

% Create a scatter plot of the received signals.

scatterplot(txpsk);

i=1;

for snr=1:2:50

rxpsk= awgn(txpsk,snr,'measured');

%x=scatterplot(rxpsk);

recovpsk = qamdemod(rxpsk,M(m));

errors=0;

c=xor(msg,recovpsk);

errors=nnz(c);

BER(k,i)=errors/length(msg);i=i+1;

end;

k=k+1;

end

figure

i=0:2:48;

semilogy(i,BER(1,:),'r',i,BER(2,:),'b',i,BER(3,:),'-\*');

legend('4QAM','16QAM','64QAM');

title('BER vs SNR');

ylabel('BER');

xlabel('SNR (dB)');

grid on

hold on

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**MPSK1**

clc;

close all; clear all;

M = 16;

%Generate random data symbols.

data = randi([0 M-1],1000,1);

%Modulate the data symbols.

txSig = pskmod(data,M,pi/M);

%Pass the signal through a noisy channel.

scatterplot(txSig)

%title('Transmitter 64-QAM constellation');

for snr=1:5:30

rxSig = awgn(txSig,snr);

%Plot the constellation diagram.

scatterplot(rxSig);

title(['Recieved signal with SNR=',num2str(snr),]);

end

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**MPSK2**

close all; clear all; clc;

len = 10000; % Number of symbols

M =[4 16 64]; % Size of alphabet

k=1

for m=1:3

% Modulate using both PSK and PAM,

% to compare the two methods.

msg = randi([0 M(m)-1],len,1); % Original signal

txpsk = pskmod(msg,M(m));

% Create a scatter plot of the received signals.

scatterplot(txpsk);

i=1;

for snr=1:2:50

rxpsk= awgn(txpsk,snr,'measured');

%x=scatterplot(rxpsk);

recovpsk = pskdemod(rxpsk,M(m));

errors=0;

c=xor(msg,recovpsk);

errors=nnz(c);

BER(k,i)=errors/length(msg);

i=i+1;

end;

k=k+1;

end

figure

i=0:2:48;

semilogy(i,BER(1,:),'r',i,BER(2,:),'b',i,BER(3,:),'-\*');

legend('4PSK','16PSK','64PSK');

title('BER vs SNR');

ylabel('BER');

xlabel('SNR (dB)');

grid on

hold on