EXP 5:

MPSK and MQAM

clc; close all;

N=input (‘Enter number of bits to be grouped: ‘); M=2^N;

x=[0 : M-1]; k=1; OFF=0; z=pskmod (x, M) ; scatterplot (z, k, OFF, 'r+'); title (‘M-ary PSK’) y=qammod(x, M);

scatterplot(y, k, OFF, ‘b\* ‘); title (‘M-QAM’)

EXP 6:

Random Process

clc; close all; load count.dat c3= count(:,3) bin\_counts= hist(c3); N= max(bin\_counts); mu3= mean(c3) sigma3= std(c3) hist(c3); hold on

plot([mu3 mu3],[0,N],'r','Linewidth',2)

X = repmat(mu3+(1:2)\*sigma3,2,1); Y = repmat([0;N],1,2); plot(X,Y,'g','Linewidth',2) legend('Data', 'Mean','Stds') hold off figure; c2= count(:,2) bin\_counts= hist(c2); N= max(bin\_counts); mu2= mean(c2) sigma2= std(c2) hist(c2); hold on

plot([mu2 mu2],[0,N],'r','Linewidth',2)

X = repmat(mu2+(1:2)\*sigma2,2,1); Y = repmat([0;N],1,2); plot(X,Y,'g','Linewidth',2) legend('Data', 'Mean','Stds') hold off figure; c1= count(:,1) bin\_counts= hist(c1); N= max(bin\_counts); mu1= mean(c1) sigma1= std(c1) hist(c1); hold on

plot([mu1 mu1],[0,N],'r','Linewidth',2)

X = repmat(mu1+(1:2)\*sigma1,2,1); Y = repmat([0;N],1,2); plot(X,Y,'g','Linewidth',2) legend('Data', 'Mean','Stds') hold off

Meantotal= mean(mean(count)); disp('overall mean=');

Meantotal

EXP 7:

BPSK

clc; close all;

data bits=1000000; % no. of bits assumed b = (randn (1, data\_bits) > .5); %random 0's and 1's s=2\*b-1; %conversion of data into bipolar format for BPSK modulation

SNRdB=0:9; %Assumed SNR in dB

for (k=1:length (SNRdB)) %BER (error/bit) calculation for different SNR y=s+awgn (s, SNRdB (k) ) ; error=0; for (c=1:1:data bits)

if (y (c)>0&&s (c) ==-1) || (y (c) <0&&s (c) ==1) %logic according to BPSK error=error+1; end end

BER (k) =error/data\_bits; %Calculate error/bit end

figure (1); %plot start

semilogy (SNRdB, BER, 'r', 'linewidth', 2); grid on;

hold on;

SNR=10.^ (SNRdB/10); % conversion of SNR to Linear value BER\_thBPSK= (1/2) \*erfc (sqrt (SNR) ) ; semilogy (SNRdB, BER\_thBPSK, 'k', 'linewidth',2); BER\_thQPSK=erfc (sqrt (SNR) ) ;

semilogy (SNRdB, BER\_thQPSK, 'b', 'linewidth', 2); legend ('PR-SNR', 'BPSK', 'QPSK' )

EXP 8:

Source Coding techniq

clc; clear all; close all;

n=input('No of symbols'); x=length(n);

p=input('Enter the probabilities');

[p,L]=sort(p,'descend'); [d,a]=huffmandict(L,p); disp([L;p]');

disp('probability codeword'); for j=1:x code=d{j,2}; fprintf('%f\t',L(j)); fprintf('%f\t',p(j)); disp([code]); end; h=sum(-p.\*log2(p)); eff=(h/a)\*100; red=(1-(h/a))\*100; disp('entropy'); disp(h); disp('average length'); disp(a); disp('efficiency'); disp(eff); disp('redundancy');

disp(red);

EXP 9:

Linear Block Code

clc; clear all; close all;

n=input('enter the codeword length in LBC (n)'); k=input('enter the no of message bits in LBC'); p=input('enter the parity check matrix'); g=[eye(k),p]; disp('Genertor matrix'); disp(g);

%d=input('enter the combination of message bits'); d=dec2bin(0:2^k-1); c=d\*g; c=rem(c,2); disp('all codewods'); disp(c); for i=1:2^k wt=0; for j=1:n if(c(i,j)==1) wt=wt+1; end

end disp(wt); Hw(i,1)=wt; end y=cat(2,c,Hw);

disp('code vector with hamming weight'); disp(y); dmin=sort(Hw(2,1)); for i=2:2^k if(dmin>Hw(i,1)) dmin=hw(i,1); end end disp('dmin'); disp(dmin); td=dmin-1;

disp('td'); disp(td); tc=(dmin-1)/2; disp('tc'); disp(tc); pt=transpose(p); disp('pt'); disp(pt); H=[pt,eye(n-k)]; disp('parity check matrix'); disp(H); ht=transpose(H);

disp('transpose of parity check matrix'); disp(ht); e=eye(n); s=e\*ht; disp(cat(2,e,s));

r=input('enter the received codeword'); synd=r\*ht; synd=rem(synd,2); disp(synd); for i=1:1:size(ht) if(ht(i,1:n-k)==synd) r(i)=1-r(i); disp('error location'); disp(i); end end

disp('corrected codeword'); disp(r);

EXP 10:

Cyclic codes

clc; clear all;

n=input('Enter the length of codeword : '); k=input('Enter the length of message : '); gen\_coff=input('Enter the generator coefficient : '); m=input('Enter the message : '); y2=[1]; a=zeros(1,n-k); z1=cat(2,y2,a); x=conv(z1,m); x1=abs(rem(x,2)); [q,r]=deconv(x1,gen\_coff); r1=abs(rem(r,2)); codeword=xor(x1,r1)

rec=input('Enter the received codeword : ');

[q,r]=deconv(rec,gen\_coff); syn=abs(rem(r,2));

if syn==0 disp('no error'); else disp('error'); end if syn==0 disp('no need of correction') else y2=zeros(1,n); e=eye(n); for i=1:n

[x2,y2(i,:)]=deconv(e(i,:),gen\_coff); end

z=abs(rem(y2,2))

for i=1:n if syn==z(i,:) break end end corrected=xor(rec,e(i,:)) end