DC

Experiment no. 5:-

% Program for Simulation of Performance of M-aryPSK & 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')
```

Experiment no. 6:-

% Program to find various statistical parameters of a random process.

```
clc;
clear all;
close all;
load count.dat;
for i = 1:3
figure;
```

```
mu(i)= mean(count(:,i));
sigma(i)= std(count(:,i));
hist(count(:,i));
title(sprintf('historgram of column %d',i));
end
MeanTotal= mean(mean(count));
disp('Mean for individual column of "Count"
Dataset=');
mu
disp('Standard Deviation Mean for individual column
of "Count" Dataset=');
sigma
disp('Overall Mean=');
MeanTotal
```

Experiment no. 7:-

% Program for Simulation study of Performance of BPSK/QPSK receiver in Presence of noise

```
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')
```

Experiment no. 8:-

% Program to Implement the algorithm of generation of Variable Length

%Source coding using Huffman Coding Algorithm.

```
clc;
```

```
clear all;
close all;
n=input('Enter symbols');
x=length(n);
p=input('Enter their probabilities');
[p,I]=sort(p,'descend');
[d,L]=huffmandict(I,p);
disp('probability codeword');
for j=1:x
code=d{j,2};
fprintf('%d\t',I(j));
fprintf('%f\t',p(j));
disp([code]);
end;
H=sum(-p.*log2(p));
eff=(H/L)*100;
red=(1-(H/L))*100;
disp('entropy');
disp(H);
disp('average length');
disp(L);
disp('efficiency');
disp(eff);
disp('redundancy');
disp(red);
```

Experiment no. 9:-

```
% Program for Simulation study LBC
clc;
clear all;
close all;
n=input('enter the codeword length in LBC (n)');//6
k=input('enter the number of message bits in LBC
(m)');(d0,d1,d2)//3
p=input('enter the parity check matrix');(as first
are identity matrix)
g=[eye(k),p];(let k 3
disp('Genertor matrix');
disp(g);
d=dec2bin(0:2^k-1);(0 to 7 binary)
c=d*g;( data*generator .atrix(
c=rem(c,2);((to display in tabular format)
disp('all codewods');
disp(c);
for i=1:2^k(1:8 times)
wt=0;
for j=1:n
if(c(i,j)==1)
wt=wt+1;
(No of 1s count in a row called hamming)
```

```
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));(minimum value of hamming wait)
for i=2:2^k
if(dmin>Hw(i,1))
dmin=hw(i,1);
end
end
disp('dmin');
disp(dmin);
td=dmin-1;( error detection capablity of coad)
disp('td');
disp(td);
tc=(dmin-1)/2;
disp('tc');(error correction capablity)
disp(tc);
//Till this encoding
pt=transpose(p);
disp('pt');
disp(pt);
H=[pt,eye(n-k)];//n-k is identity matrix
disp('parity check matrix');
disp(H);
ht=transpose(H);
disp('transpose of parity check matrix');
disp(ht);
e=eye(n);
s=e*ht;// syndrom
disp(cat(2,e,s));//concat function
r=input('enter the received codeword');// do wrong
one
synd=r*ht;//if non zero then error exited
synd=rem(synd,2);//in matrix form.
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);
```

Experiment no. 10:-

```
% Program for Simulation study of Cyclic code

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));

disp('S(x) for received code is:');disp(syn);

if syn==0

  disp('No error in received code');

  disp('No need of correction')

else

  disp('Error in received code');
  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

  correctedCode=xor(rec,e(i,:))

end
```