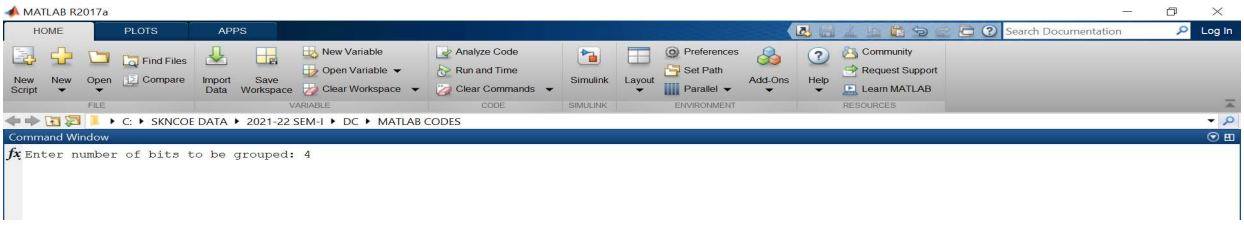
**SAMPLE PRINTOUT:**

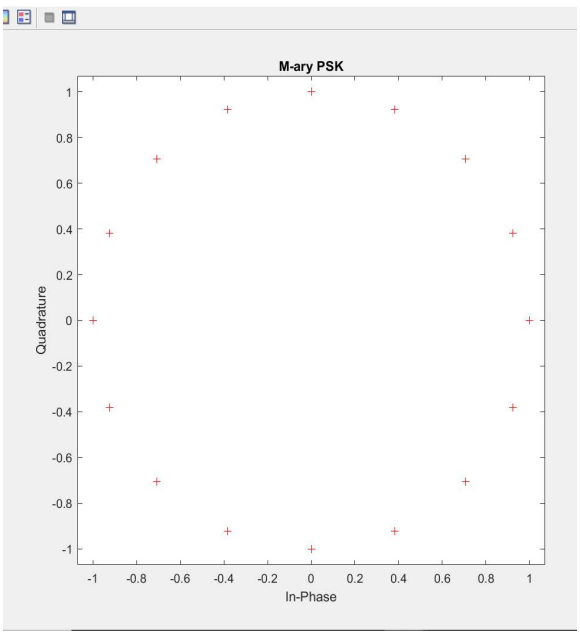
**Title: Simulation Study of Performance of MPSK and MQAM**
**Name:**
**Roll No:_____ Division: TE- Batch:_____**

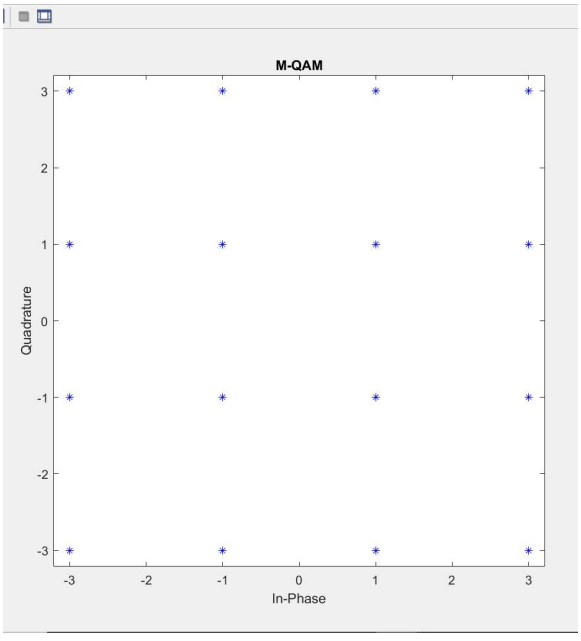
**CODE:**

```
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')
```

**OUTPUT:**



**Title:** Simulation study of random processes. Find various statistical parameters of the random process.

**Program:**

```
clc;
clear all;
close all;
load count.dat;
for i = 1:3
    bin_counts(i,:)= hist(count(:,i));
    mu(i)= mean(count(:,i));
    sigma(i)= std(count(:,i));
    hist(count(:,i));
    figure;
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
```

**Output:**

```
Mean for individual column of "Count" Dataset=

mu =

   32.0000   46.5417   65.5833

Standard Deviation Mean for individual column of "Count"
Dataset=

sigma =

   25.3703   41.4057   68.0281

Overall Mean=

MeanTotal =

   48.0417
```

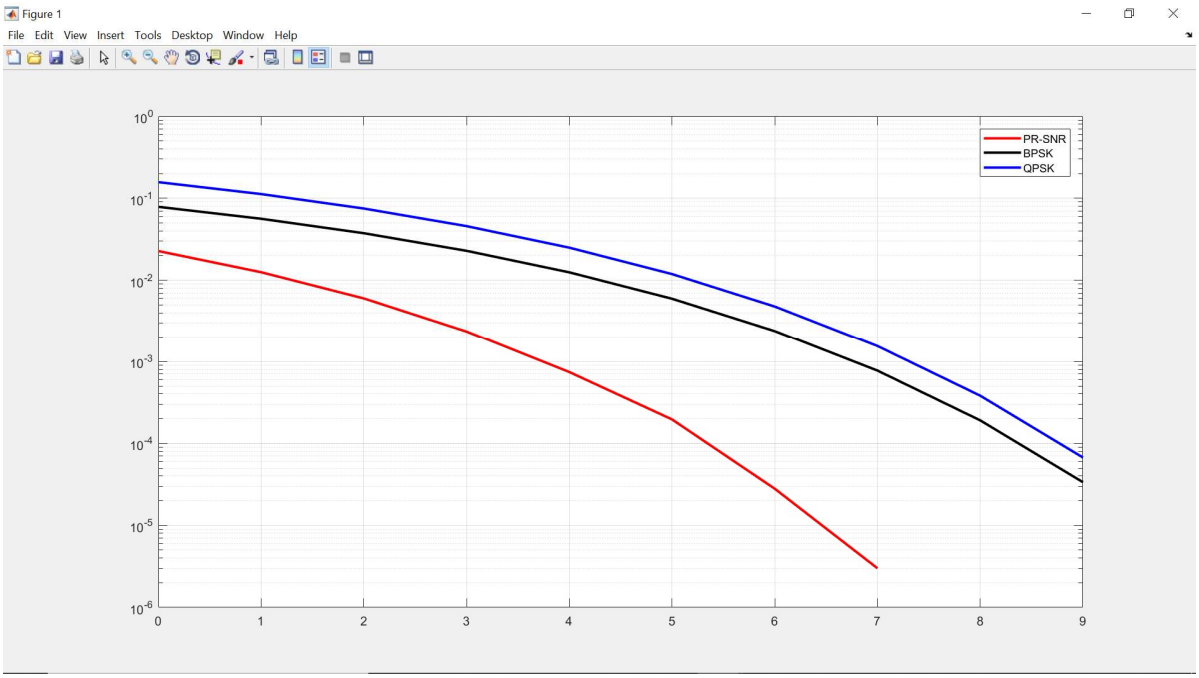
**SAMPLE PRINTOUT:**

**Title: Simulation Study of Performance Evaluation of BPSK**
**Name:**
**Roll No:_____ Division: TE- Batch:_____**

**CODE:**

```
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')
```

**OUTPUT:**



```
**************************************************************************
Name:_____Course: TE Div-

Batch:       Roll No.:

Title: Simulation study of Huffman Source Coding technique.
**************************************************************************
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);
```

## Program Output

No of symbols[1 2 3 4]

Enter the probabilities[0.5 0.1 0.2 0.2]

```
   1.0000    0.5000
   3.0000    0.2000
   4.0000    0.2000
   2.0000    0.1000

probability codeword
1.000000      0.500000        0

3.000000      0.200000        1    0    0

4.000000      0.200000        1    1
```

2.000000 0.100000 1 0 1

entropy
1.7610

average length
1.8000

efficiency
97.8313

redundancy
2.1687

**********
Name:_____Course: TE Div-

Batch:  Roll No.:

Title: Simulation study of Linear Block codes.
**********

```
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);
```

## Program Output

enter the codeword length in LBC (n)6
enter the no of message bits in LBC3
enter the parity check matrix[1 0 1; 1 1 0; 0 0 1]
Genertor matrix

| | | | | | |
|---|---|---|---|---|---|
| 1 | 0 | 0 | 1 | 0 | 1 |
| 0 | 1 | 0 | 1 | 1 | 0 |
| 0 | 0 | 1 | 0 | 0 | 1 |

all codewods

| | | | | | |
|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 | 0 | 1 |
| 0 | 1 | 0 | 1 | 1 | 0 |
| 0 | 1 | 1 | 1 | 1 | 1 |
| 1 | 0 | 0 | 1 | 0 | 1 |
| 1 | 0 | 1 | 1 | 0 | 0 |
| 1 | 1 | 0 | 0 | 1 | 1 |
| 1 | 1 | 1 | 0 | 1 | 0 |

0
2
3
5
3

3
4
4

code vector with hamming weight

| | | | | | | |
|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 | 0 | 1 | 2 |
| 0 | 1 | 0 | 1 | 1 | 0 | 3 |
| 0 | 1 | 1 | 1 | 1 | 1 | 5 |
| 1 | 0 | 0 | 1 | 0 | 1 | 3 |
| 1 | 0 | 1 | 1 | 0 | 0 | 3 |
| 1 | 1 | 0 | 0 | 1 | 1 | 4 |
| 1 | 1 | 1 | 0 | 1 | 0 | 4 |

dmin

2

td

1

tc

0.5000

pt

| | | |
|---|---|---|
| 1 | 1 | 0 |
| 0 | 1 | 0 |
| 1 | 0 | 1 |

parity check matrix

| | | | | | |
|---|---|---|---|---|---|
| 1 | 1 | 0 | 1 | 0 | 0 |
| 0 | 1 | 0 | 0 | 1 | 0 |
| 1 | 0 | 1 | 0 | 0 | 1 |

transpose of parity check matrix

| | | |
|---|---|---|
| 1 | 0 | 1 |
| 1 | 1 | 0 |
| 0 | 0 | 1 |
| 1 | 0 | 0 |
| 0 | 1 | 0 |
| 0 | 0 | 1 |

```
    1    0    0    0    0    0    1    0    1
    0    1    0    0    0    0    1    1    0
    0    0    1    0    0    0    0    0    1
    0    0    0    1    0    0    1    0    0
    0    0    0    0    1    0    0    1    0
    0    0    0    0    0    1    0    0    1

enter the received codeword[0 0 1 0 1 1]
    0    1    0
error location
    5
corrected codeword
    0    0    1    0    0    1
```

```
**************************************************************************
Name:_____Course: TE Div-

Batch:       Roll No.:

Title: Simulation study of 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
```

## Program Output

Enter the length of message : 4
Enter the generator coefficient : [1 0 1  1]
Enter the message : [1  1 0 0]

codeword =

| | | | | | | |
|---|---|---|---|---|---|---|
| 1 | 1 | 0 | 0 | 0 | 1 | 0 |

Enter the received codeword : [1 1 0 0 1 1 0]
error

z =

| | | | | | | |
|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 1 | 0 | 1 |
| 0 | 0 | 0 | 0 | 1 | 1 | 1 |
| 0 | 0 | 0 | 0 | 1 | 1 | 0 |
| 0 | 0 | 0 | 0 | 0 | 1 | 1 |
| 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 1 |

corrected =

| | | | | | | |
|---|---|---|---|---|---|---|
| 1 | 1 | 0 | 0 | 0 | 1 | 0 |