Practical-5

Case 1:

```
m = 0;

sd = 1;

x = -6:0.1:6;

y = normpdf(x, m, sd);

figure(1)

subplot(2, 3, 1);

plot(x, y);

axis([-6 6 0 0.5]);

title('CASE-1, mean=0, sd=1');

xlabel('x values --->');

ylabel('pdf --->');

grid on;
```

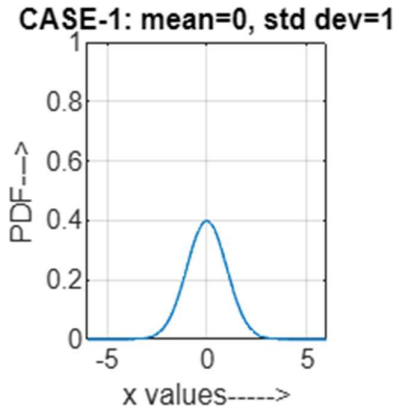
Case 2:

```
m = -1;

sd = 1;

x = -6:0.1:6;

y = normpdf(x, m, sd);

figure(1)

subplot(2, 3, 2);

plot(x, y);

axis([-6 6 0 0.5]);

title('CASE-2, mean=-1, sd=1');

xlabel('x values --->');

ylabel('pdf --->');

grid on;
```

Case 3:

```
m = 1;

sd = 1;

x = -6:0.1:6;

y = normpdf(x, m, sd);

figure(1)

subplot(2, 3, 3);

plot(x, y);

axis([-6 6 0 0.5]);

title('CASE-3, mean=1, sd=1');

xlabel('x values --->');

ylabel('pdf --->');

grid on;
```

Outputs:

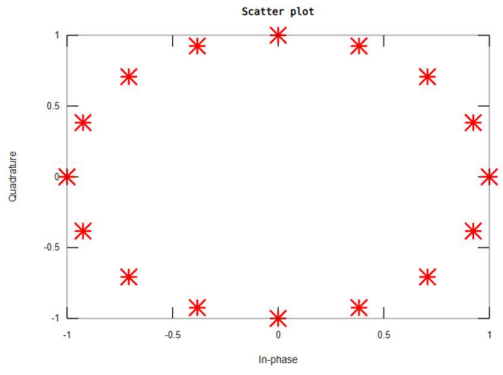


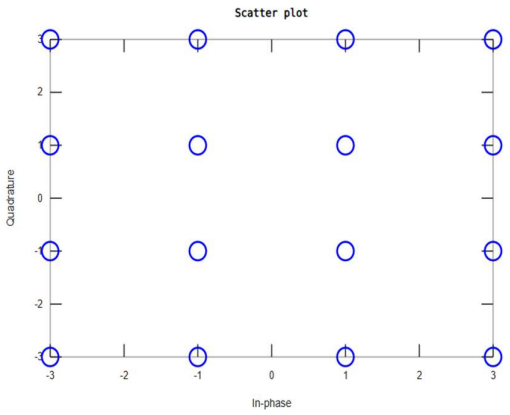
Practical-6

```
clc;
clear all;
M=input('Number_Symbols=');
x=0:M-1;
N=1;
OFF=0;
z=pskmod(x,M);
figure(1)
scatterplot(z,N,OFF,"r*");
N=1;
OFF=0;
y=qammod(x,M);
figure(2)
scatterplot(y,N,OFF,"bo");
```

OUTPUT:

```
Number_Symbols=> 16
```

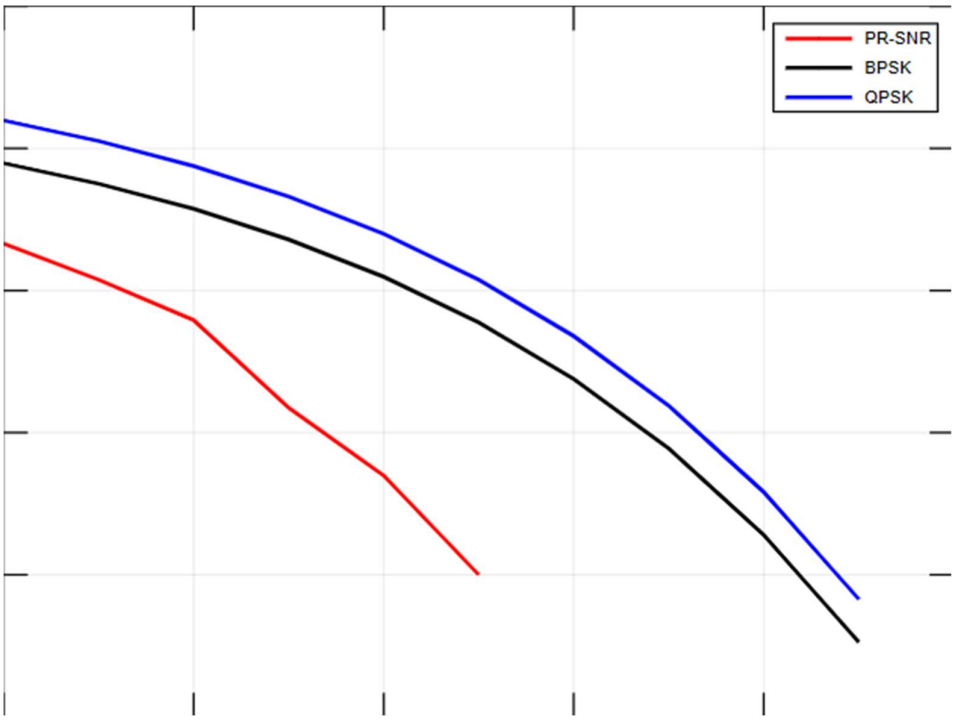




Practical-7

```
clc;
close all;
data_bits=10000;
b=(randn(1,data_bits) > 5);
s=2*b-1;
SNRdB=0:9;
for(k=1:length(SNRdB))
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)
error=error+1;
end
end
BER(k)=error/data_bits;
end
figure(1);
semilogy(SNRdB, BER, 'r', 'Linewidth', 2);
grid on;
hold on;
SNR=10.^(SNRdB/10);
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:



Practical-9

```
clc;
clear all;
close all;
n=input('Enter the value of n: ');
k=input('Enter the value of k: ');
m=n-k;
G=cyclpoly(n,k,'max')
poly2sym(G)
d1=[1 0 0 0];
poly2sym(d1)
c1=poly2sym(d1)*poly2sym(G)
d2=[0 1 0 0];
poly2sym(d2)
c2=poly2sym(d2)*poly2sym(G)
d3=[0 0 1 0];
poly2sym(d3)
c3=poly2sym(d3)*poly2sym(G)
d4=[0 0 0 1];
poly2sym(d4)
c4=poly2sym(d4)*poly2sym(G)
s=[c1;c2;c3;c4]
d=[d1;d2;d3;d4]
c=d*s
parmat=hammgen(m)
trt=syndtable(parmat)
recd=[0 1 0 1 0 0 0]
syndrome=rem(recd*parmat',2)
syndrome_de=bi2de(syndrome,'left-msb')
disp([syndrome,'left-msb'])
```

```
disp(['Syndrome=',num2str(syndrome_de),'(decimal)',num2str(syndrome_de),(binary)'])
Error=trt(1+syndrome_de,:)
correctedcode= rem(Error+recd,2)
recd=[1 1 0 1 1 0 1]
syndrome=rem(recd*parmat',2)
syndrome_de=bi2de(syndrome,'left-msb');
disp(['Syndrome=',num2str(syndrome_de), '(decimal)',
num2str(syndrome_de),'(binary)'])
Error=trt(1+syndrome_de,:)
correctedcode=rem(Error+recd,2)
```

```
Enter the value of n: > 5
Enter the value of k: > 4
G =

   1   1

Symbolic pkg v3.1.1: Python communication link active, SymPy v1.9.
ans = (sym) x + 1
ans = (sym)

   3
  x

c1 = (sym)

   3
  x ·(x + 1)

ans = (sym)

   2
  x

c2 = (sym)

   2
  x ·(x + 1)

ans = (sym) x
!!! OUT OF TIME !!!
```

Practical-8

```
clc;
clear all;
k=input('Enter the length of msg word:');
n=input('Enter the length of codeward:');
p=input('Enter the parity matrix:'); G=[eye(k);p];
m=input('Enter the length of msg word:');
H=[p' eye(n-k)]
dtable=syndtable (H)
R=input('Enter the received code word'); %S=R*H'
S_B=rem(R*H',2)
S_D=bi2de(S_B, 'left-msb')
if(S_D==0)
disp('The received codeward is valid:')
else
disp('The corrected codeward is invalid:')
E=dtable(S_D+1,:)
%CC=R+E
disp('The corrected codeward is:')
cc=rem (R+E,2)
msg=cc(1:k)
end
```

```
Enter the length of msg word:> 3
Enter the length of codward:> 6
Enter the parity matrix:> [1 1 1;1 1 0;1 0 1]
Enter the length of msg word:> [1 1 1]
H =

   1   1   1   1   0   0
   1   1   0   0   1   0
   1   0   1   0   0   1

dtable =

   0   0   0   0   0   0
   0   0   0   0   0   1
   0   0   0   0   1   0
   1   0   0   1   0   0
   0   0   0   1   0   0
   0   0   1   0   0   0
   0   1   0   0   0   0
   1   0   0   0   0   0

Enter the received code word> [1 0 1 1 0 0]
S_B =

   1   1   0

S_D = 6
The corrected codeward is invalid:
E =

   0   1   0   0   0   0

The corrected codeward is:
cc =

   1   1   1   1   0   0

msg =

   1   1   1
```

Experiment 10-Huffman Coding

```
clc;
 clear all;
 close all;
 code_length=0;
 x=input('Enter number of symbols: ');
 for m=1:x
 symbols(m)=input('Enter the symbol number: ');
 p(m)=input('Enter the probability: ');
 end
 Hx=0
 for m=1:x
 [dict,avglen]=huffmandict(symbols,p)
 hcode=huffmanenco(m,dict)
 dsig = huffmandeco(hcode,dict)
 code_length=length(hcode)
 Hx=Hx+(p(m)*(-log(p(m)))/(log(2)));
 end
 display(Hx);
 Efficiency=(Hx/avglen)*100
 Disp(Efficiency)
```

OUTPUT→

```
Enter number of symbols: 6
Enter the symbol number: 1
Enter the probability: 0.3
Enter the symbol number: 2
Enter the probability: 0.25
Enter the symbol number: 3
Enter the probability: 0.2
```

Enter the symbol number: 4

Enter the probability: 0.12

Enter the symbol number: 5

Enter the probability: 0.05

Enter the symbol number: 6

Enter the probability: 0.08

p =

0.3000 0.2500 0.2000 0.1200 0.0800 0.0500

0.3000 0.2500 0.2000 0.1200 0.0800 0.0500

Hx =

0

dict =

6×2 cell array

{[1]} {[0 0]}

{[2]} {[0 1]}

{[3]} {[1 1]}

{[4]} {[1 0 1]}

{[5]} {[1 0 0 0]}

{[6]} {[1 0 0 1]}

avglen =

2.3800

Hx =

2.3601

Efficiency =

99.1659

Efficiency= 99.165859