Source Encoding

```
clc; clear all; close all;
m=input('Enter the
no. of symbols : ');
z=[]; h=0; l=0;
display('Enter the
symbol
probabilities');
for i=1:m
p(i)=input('');
end
p=sort(p,'descend');
fprintf('\n  Prob  in
descending order');
display(p);
F(1)=0;  for
j=2:(m+1);
 F(j)=F(j-1)+p(j-1);
End
 fprintf('\n F -
Matrix'); display(F);
for i=1:m
n(i)=ceil(log2(1/p(i)));
 end
fprintf('\n
Codeword length
matrix');
display(n);   for
i=1:m  int=F(i);  for
j=1:n(i)  frac=int*2;
c=floor(frac);
frac=frac-c; z=[z c];
int=frac;  end
fprintf('Codeword
%d',i);
 display(z);
z=[];  end   for
i=1:m
x=p(i)*n(i);
l=l+x;
 x=p(i)*log2(1/p(i));
h=h+x;
end
disp(['Avg.
Codeword Length, L =
',num2str(l),'
bits/symbol']);
disp(['Entropy H(X) =
',num2str(h),'
bits/symbol']);
eff=100*h/l;
disp(['Efficiency =
',num2str(eff),'%']);
rdc=100-eff;
disp(['Redundancy =
',num2str(rdc),'%'])
```

Linear block codes

```
clear all; close all; clc;
P=input('Enter the parity
matrix:\n'); disp("The value of k is")
k=size(P,1) temp=size(P,2);
disp("The value of n is")
 n=k+temp
Disp("*********ENCODER*******")
I=eye(k);
disp("The generator matrix is")
G=[I P]
a=dec2bin(0:+1:2^k-1);
C=a*G;  for i=1:2^k
for j=1:n
 if(rem(C(i,j),2)==0)
C(i,j)=0;  else
C(i,j)=1; end
 End  end
disp("The codewords are as
follows:\n")
disp("****DECODER*************")
I=eye(temp);
 disp("The parity check matrix is")
Pt=P.';
H=[Pt I]
R=input('Enter    the    recieved
codeword in matrix format:\n'); C
Ht=H.';
 disp("The syndrome matrix
is") S=R*Ht;  for i=1:temp
if(rem(S(i),2)==0)  S(i)=0;  else
S(i)=1;  end
 end S   for
i=1:n
if(Ht(i,:)==S)
E(i)=1;  else
E(i)=0;
end  end
disp("The error matrix is") E
 for i=1:n
 CC(i)=xor(E(i),R(i));
End
 disp("The corrected codeword
is")CC
```

PCM

```
fm=5;
 fs=1000*fm;
 t=0:1/fs:1;
 m=3.5;
x=m*sin(2*pi*fm*t)
figure(1);
plot(t,x);
 xlabel('Time');
ylabel('Amplitude');
title('Message
Signal');
 for i= 1:length(x);
 if x(i)>0.5 &&
x(i)<=1.5 xq(i)=1;
 e=[1 0 0];
elseif x(i)>1.5 &&
x(i)<=2.5 xq(i)=2;
 e=[1 0 1];
 elseif x(i)>2.5 &&
x(i)<=3.5 xq(i)=3;
e=[1 1 0];
 elseif x(i)>-3.5 &&
x(i)<=-2.5
xq(i)=-3;
e=[0 0 0];
 elseif x(i)>-2.5 &&
x(i)<=-1.5 xq(i)=-2;
 e=[0 0 1];
 elseif x(i)>-1.5 &&
x(i)<=-0.5 xq(i)=-1;
 e=[0 1 0 ];
 elseif x(i)>-0.5 &&
x(i)<=0.5 xq(i)=0;
 e=[0 1 1 ];
 end
 end
 figure(2);
 plot(t,xq);
 title('Quantized
Signal') figure(3);
plot(x,x-xq);
  title('Error Signal')
```

PSD OF SPECTRAL DENSITY

```
clc clear all close all
Tb=1;
 f=0:0.0001*Tb:5;
 x=f*Tb;
P1=(0.25*Tb*(sinc(x).^2)+0.25*(dirac(f)));
figure(1)
plot(f,P1,'r')
 xlabel('fTb ')
 ylabel('Power Spectral Density ')
title('PSD')
P=1*Tb*(sinc(x).*sinc(x));
 figure(2)
 plot(f,P,'r')
xlabel('fTb *')
 ylabel('Power Spectral Density ')
title('PSD for Polar Signal')
P3=1*Tb*(sinc(x/2)).^2.*(sin(pi*x)).^2;
figure(4)
 plot(f,P3,'r')
 xlabel('fTb ')
ylabel('Power Spectral Density')
title('PSD for Bipolar Signal')
```

GENERATION OF MOD WAVES

```
clc
close all;
clear all;
clc;
f1=5;
f2=10;
x=[1 0 1 0 1 1 0 1];
nx=size(x,2);
i=1;
while (i<(nx+1));
 t=i:0.001:i+1;
 if (x(i)==1)
 ASK=sin(2*pi*f1*t);
 FSK=sin(2*pi*f1*t);
 PSK=sin(2*pi*f1*t);
 else
 ASK=0;
 FSK=sin(2*pi*f2*t);
 PSK=sin(2*pi*f1*t+pi);
 end
 %Amplitude Shift Keying
 subplot (3,1,1);
 plot(t,ASK);
 hold on;
 grid on;
 axis([1,10,-1,1]);
 title('Amplitude Shift Keying');
%Frequency Shift Keying
 subplot (3,1,2);
 plot(t,FSK);
 hold on;
 grid on;
 axis([1,10,-1,1]);
 title('Frequency Shift Keying');
  %Phase Shift Keying
 subplot (3,1,3);
 plot(t,PSK)
 hold on;
 grid on;
 axis([1,10,-1,1]);
 title('Phase Shift Keying');
 i=i+1;
end
```