Compute the value of Erlang B, given the value of N and Rho.

```
function B=erlangb(n,rho)
% Sanity check- make sure that n is a positive integer.  n=1;rho=4;
if ((floor(n) ~= n) || (n < 1))  warning('n is not
a positive integer');
B=NaN;
return;  end;
% Sanity check- make sure that rho >= 0.0.
if (rho < 0.0)
warning('rho is negative!');
B=NaN;
return;  end;
% Start the recursion with B=1.
B=1;
% Run the recursion.  for
k=1:n,
%B=((rho.^n)/factorial(n))./(sum((rho.^k)/factorial(k)));
%B(0,rho)=1;
%B(n,rho)=(rho*B(n-1,rho)/n)/(1+rho*B(n-1,rho)/n);
B=((rho*B)/k)/(1+rho*B/k);
%disp(B);
end;
```

OUTPUT:

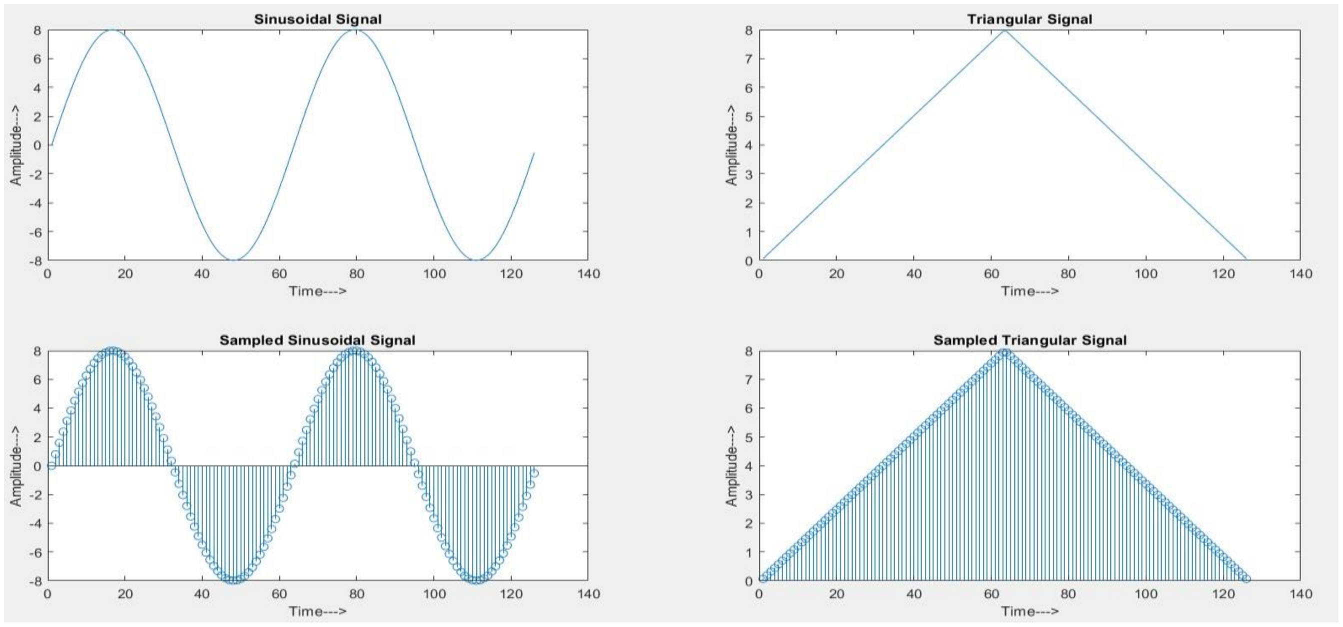
S.No	N	Rho	Blocking probability
1	10	5	0.0184
2	20	5	2.6412e-07
3	10	10	0.2146
4	8	3.627	0.02
5	10	20	0.5380

From the above table we see that N is inversely proportional to blocking probability and rho is directly proportional to blocking probability, which is as predicted from theoretical analysis.

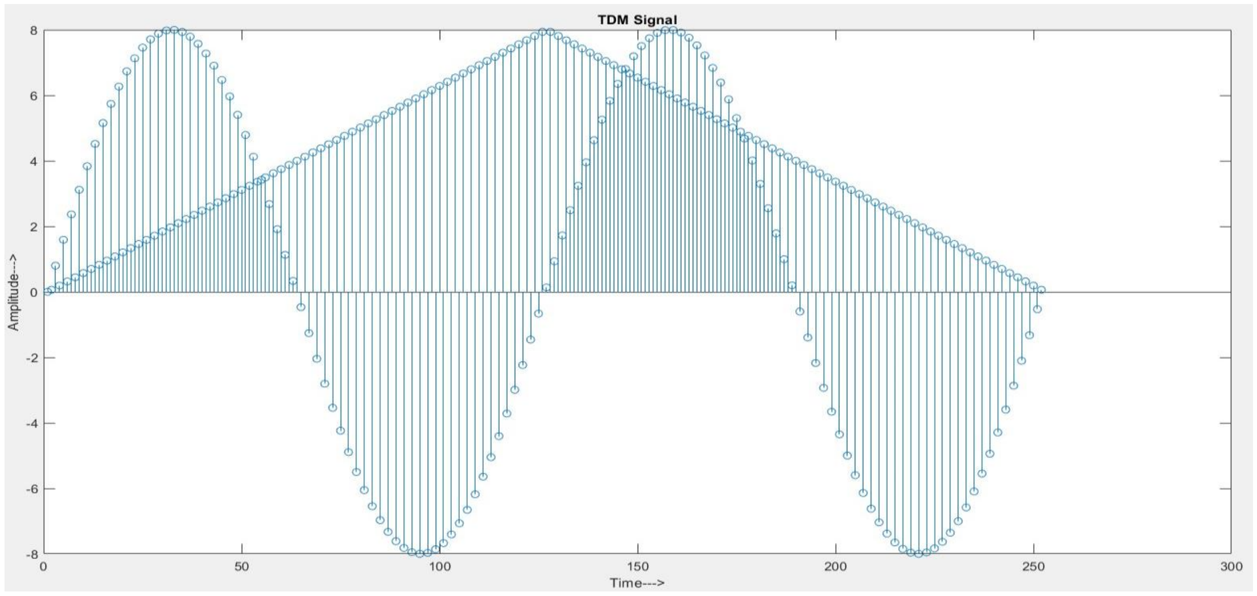
Time Division Multiplexing

```
% Signal generation  x=0:.5:4*pi; % siganal taken
upto 4pi  sig1=8*sin(x); % generate 1st sinusoidal
signal  l=length(sig1);
sig2=8*triang(l); % Generate 2nd traingular Sigal
% Display of Both Signal
subplot(2,2,1);  plot(sig1);
title('Sinusoidal Signal');
ylabel('Amplitude--->');
xlabel('Time--->');
subplot(2,2,2);  plot(sig2);
title('Triangular Signal');
ylabel('Amplitude--->');
xlabel('Time--->');
% Display of Both Sampled Signal
subplot(2,2,3);  stem(sig1);
title('Sampled Sinusoidal Signal');
ylabel('Amplitude--->');  xlabel('Time--
->');  subplot(2,2,4);  stem(sig2);
title('Sampled Triangular Signal');
ylabel('Amplitude--->');  xlabel('Time--
->');  l1=length(sig1);  l2=length(sig2);
for i=1:l1
sig(1,i)=sig1(i); % Making Both row vector to a matrix
sig(2,i)=sig2(i);  end
% TDM of both quantize signal
tdmsig=reshape(sig,1,2*l1);  %
Display of TDM Signal  figure
stem(tdmsig);  title('TDM Signal');
ylabel('Amplitude--->');
xlabel('Time--->');
% Demultiplexing of TDM Signal  demux=reshape(tdmsig,2,l1);
for i=1:l1
sig3(i)=demux(1,i); % Converting The matrix into row vectors
sig4(i)=demux(2,i);  end
% display of demultiplexed signal
figure  subplot(2,1,1)
plot(sig3);
title('Recovered Sinusoidal Signal');
ylabel('Amplitude--->');  xlabel('Time---
>');  subplot(2,1,2)  plot(sig4);
```

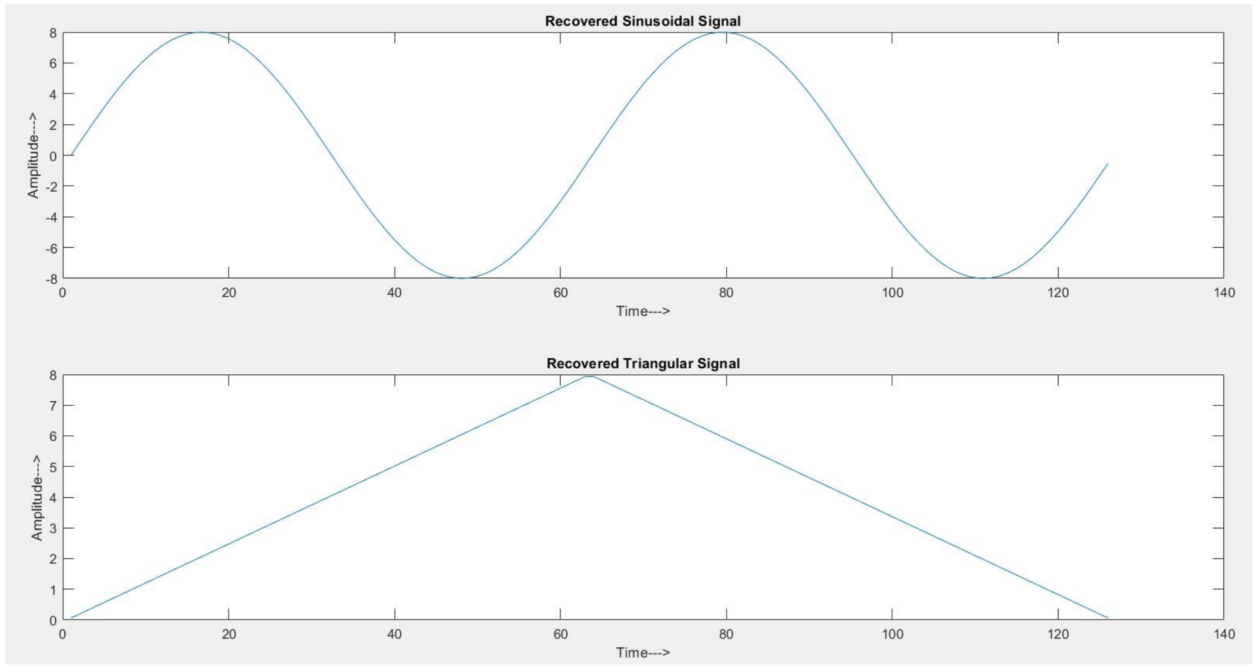
```
title('Recovered Triangular Signal');  ylabel('Amplitude--->');
xlabel('Time--->');
```



The above figure shows sinusoidal and triangular wave with continuous and discrete plot.



This figure shows that the triangular wave and sinusoidal wave are multiplexed as a single wave. The values of sinusoidal signal and triangular signal are plotted alternately.



Once the two signals are multiplexed. They are recovered and the recovered signal are identical to the original individual signals.

Calculation of Engset for, m customers connected with n servers.

```
function E=engset(m,n,rho)
% Sanity check- make sure that m and n are positive integers.  if ((floor(m)
~= m) || (m < 1))
warning('m is not a positive integer');
E=NaN;
return  end
if ((floor(n) ~= n) || (n < 0))
warning('n is not a nonnegative integer');
E=NaN;
return  end
% Sanity check- make sure that rho >= 0.0.
if (rho < 0.0)
warning('rho is negative!');
E=NaN;
return  end;
% Special case. If we have fewer customers than servers, than the  %
blocking probability is 0.  if (m<=n)  E=0;  return  end
% Start the recursion with B=1.
E=1;
% Run the recursion.
for k=1:n,
E=(rho*(m-k+1)*E)/(k+rho*(m-k+1)*E);  end;
```

OUTPUT:

S.No	M	N	Rho	Blocking probability
1	20	5	10	0.969
2	20	10	10	0.9106
3	20	5	5	0.9385
4	30	10	10	0.9527
5	30	10	4.8	0.9023

From the above different values of M, N, and Rho, we get different values of blocking probability. We see that by increasing number of sources and keeping others constant, the blocking probability increases. By increasing N only, the
blocking probability decreases and by decreasing Rho only, the blocking probability decreases.

The function is used to compute the value of Erlang C. This value uses the value of Erlang B along with, the number of servers N, and rho.

```
function C=erlangc(n,rho)
% Sanity check- make sure that n is a positive integer.
%n=2;rho=2;
if ((floor(n) ~= n) || (n < 1))  warning('n is not
a positive integer');
C=NaN;
return;  end;
% Sanity check- make sure that rho >= 0.0.
if (rho < 0.0)
warning('rho is negative!');
C=NaN;
return;  end;
% Calculate the Erlang B probability and then convert to Erlang C.
B=erlangb(n,rho);
C=n*B/(n-rho*(1-B));
```

OUTPUT:

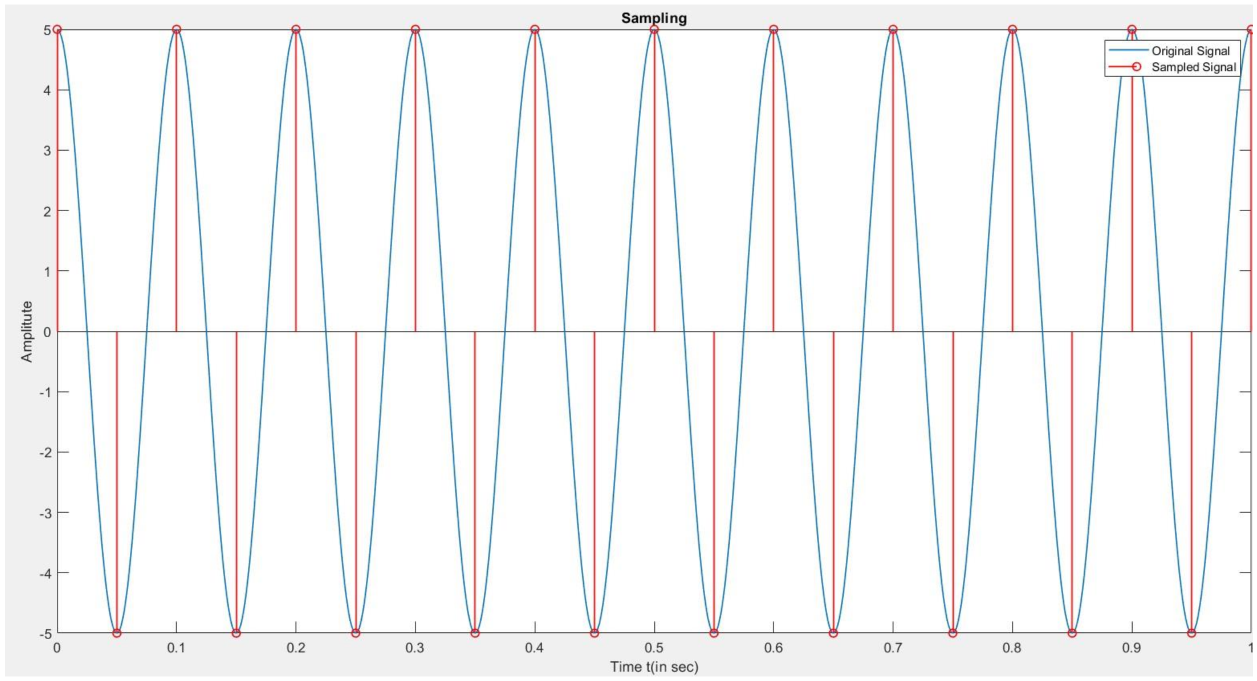
S.No	N	Rho	Blocking probability
1	10	5	0.0361
2	20	5	3.5216e-07
3	10	10	1
4	18	7	3.79523-04
5	20	18	0.5508

From the table we see that when N increases blocking probability decreases and when rho increases blocking probability increases.

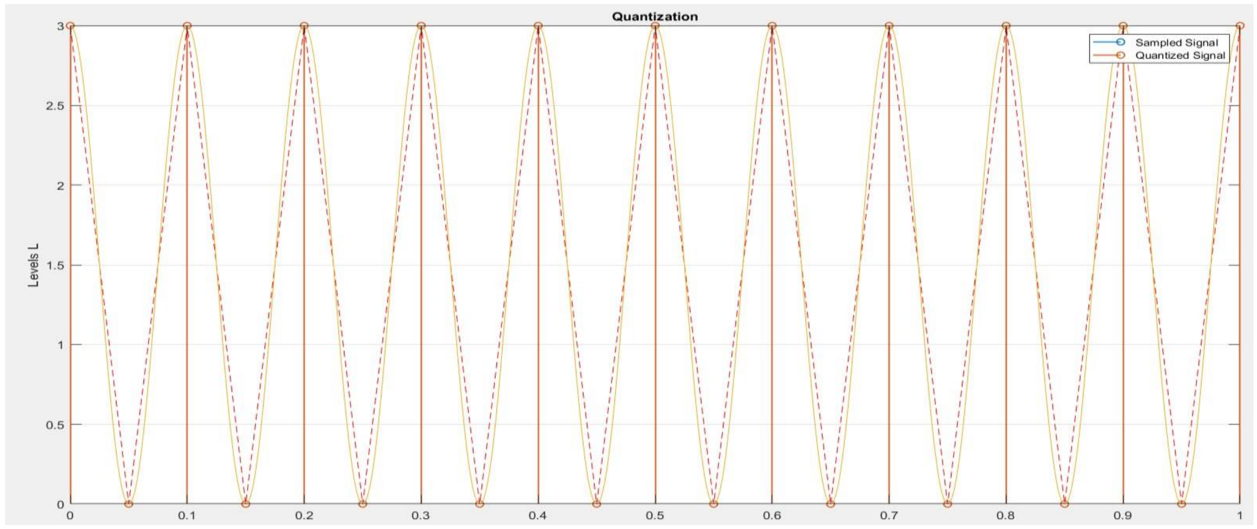
Pulse Code Modulation

```
function [y Bitrate MSE Stepsize QNoise]=pcm(A,fm,fs,n)
%A=amplitute of cosine signal
%fm=frequency of cosine signal
%fs=sampling frequency  %n=
number of bits per sample
t=0:1/(100*fm):1;
x=A*cos(2*pi*fm*t);  %---
Sampling-----  ts=0:1/fs:1;
xs=A*cos(2*pi*fm*ts);  %xs
Sampled signal  %--Quantization---
x1=xs+A;  x1=x1/(2*A);  L=(-
1+2^n); % Levels  x1=L*x1;
xq=round(x1); r=xq/L; r=2*A*r;
r=r-A;
%r quantized signal
%----Encoding---
y=[];  for
i=1:length(xq)
d=dec2bin(xq(i),n);
y=[y double(d)-48];
end  %Calculations
MSE=sum((xs-r).^2)/length(x);
Bitrate=n*fs;
Stepsize=2*A/L;
QNoise=((Stepsize)^2)/12;
figure(1)  plot(t,x,'linewidth',1)
title('Sampling')  ylabel('Amplitute')
xlabel('Time t(in sec)')  hold on
stem(ts,xs,'r','linewidth',1)  hold off
legend('Original Signal','Sampled Signal');
figure(2)  stem(ts,x1,'linewidth',1)
title('Quantization')  ylabel('Levels L')  hold on
stem(ts,xq,'linewidth',1)  plot(ts,xq,'--r')
plot(t,(x+A)*L/(2*A))
grid  hold
off
legend('Sampled Signal','Quantized Signal');  figure(3)
stairs([y y(length(y))],'linewidth',1)
title('Encoding')  ylabel('Binary Signal')
xlabel('bits')  axis([0 length(y) -1 2])
grid
```

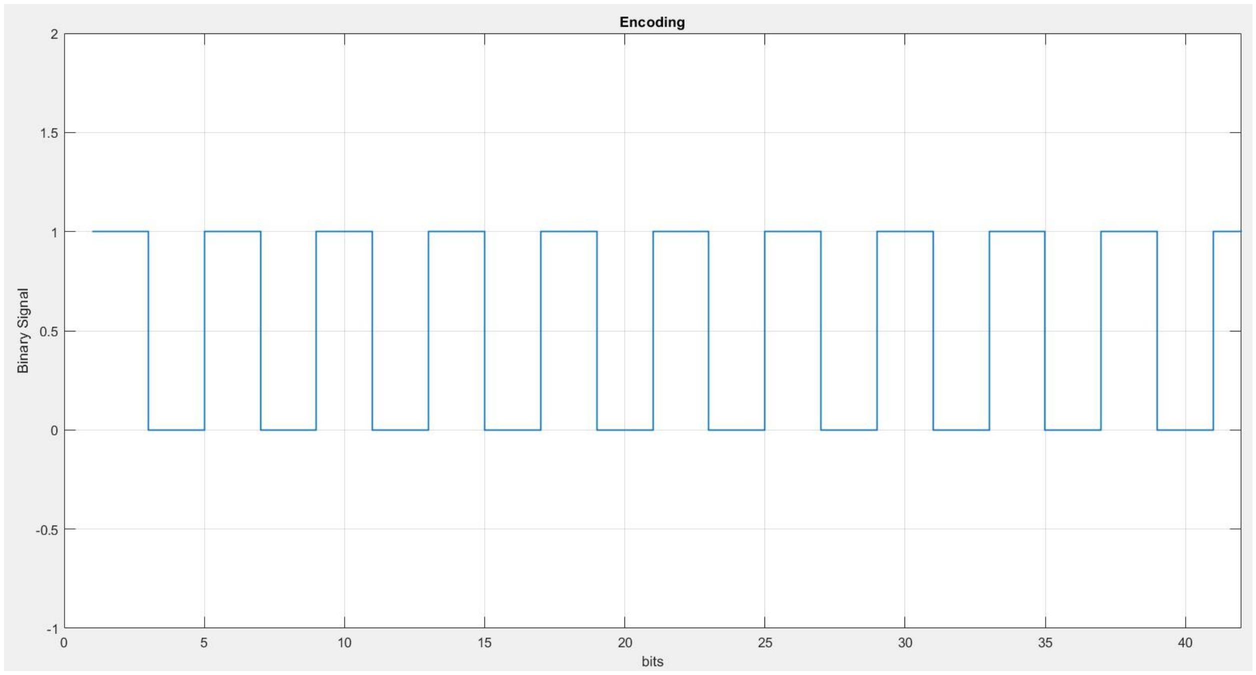
OUTPUT:



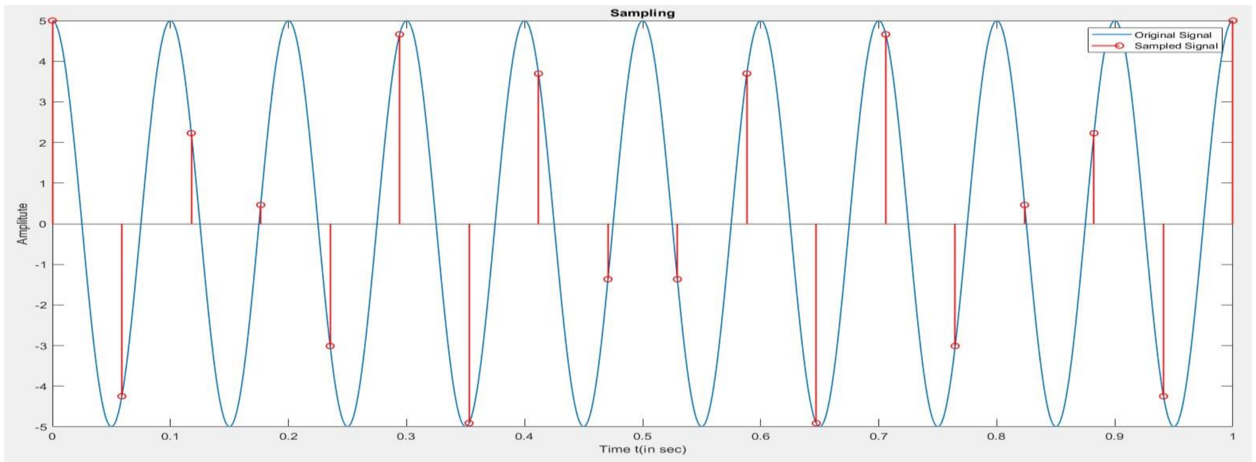
The blue curve shows the cosine wave and the discrete red signals shows the sampled values. The cosine frequency is 10 while the sampling frequency is 20, which means for every cosine wave, two samples are taken.

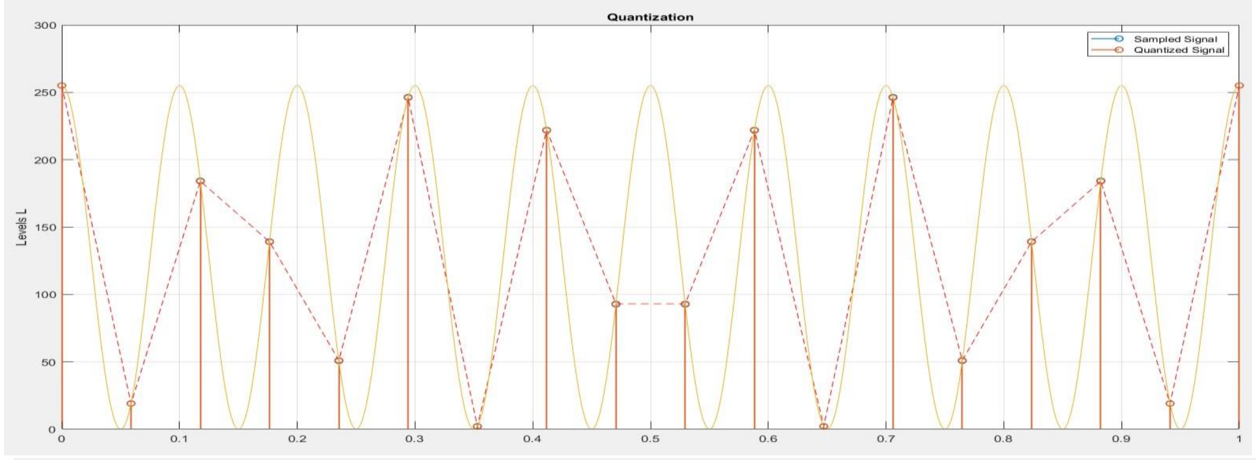


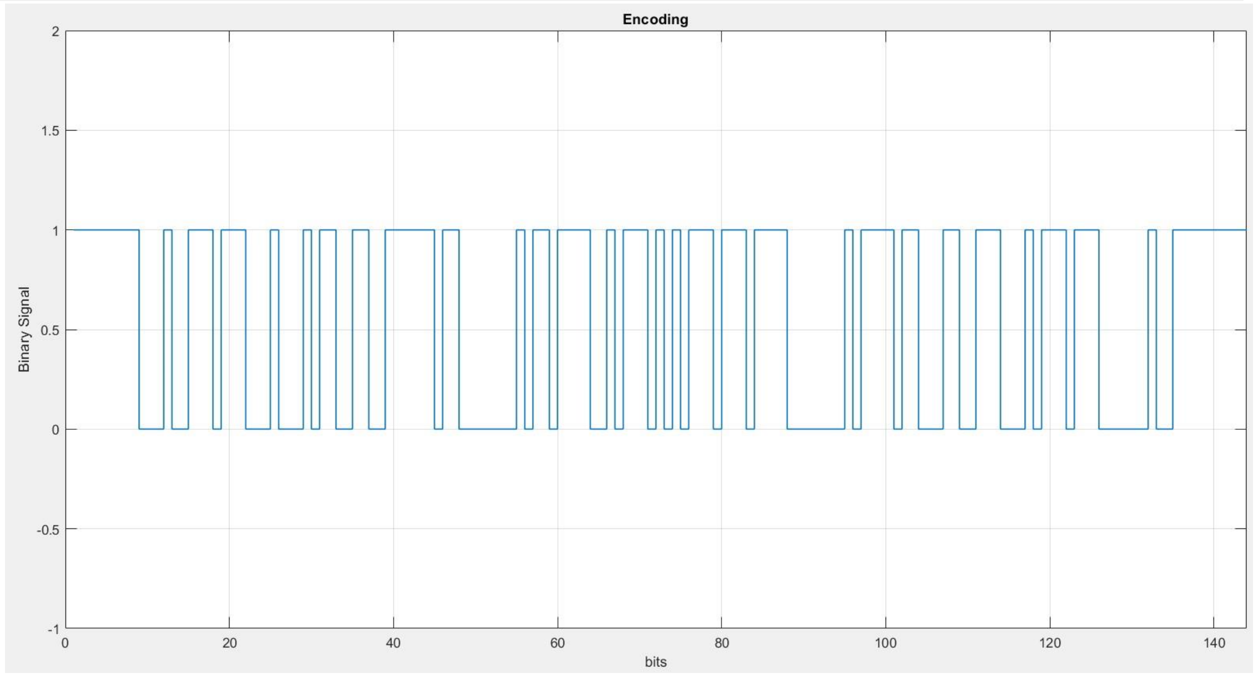
This figure shows quantization. Here for this Nyquist sampling rate, there are only two quantization levels.



The final part in PCM is encoding. The quantized signals are converted to digital signals. Here, as there are only two distinct levels, this is binary signals. Now for cosine frequency of 10 and sampling frequency of 17 and 8 quantization levels, the three levels in PCM is shown as follows:


Sampling
Original Signal
Sampled Signal
Amplitude
Time t(in sec)


Quantization
Sampled Signal
Quantized Signal
Levels L


Encoding
Binary Signal
bits