Lab 2

Digital Communications

PCM

ID	الاسم
19016377	محمد سامي محمد عبدالعزيز
19015660	رشاد السيد ابراهيم احمد
19015536	تسنيم عبد الصمد

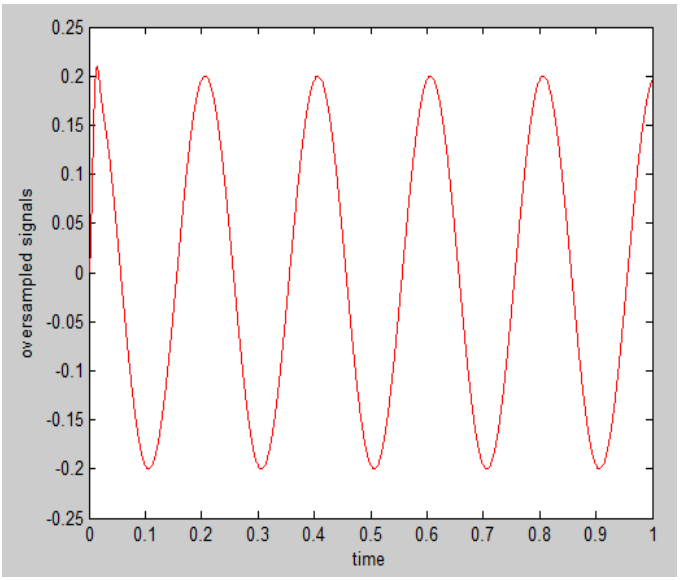
1-Sampling and Reconstruction

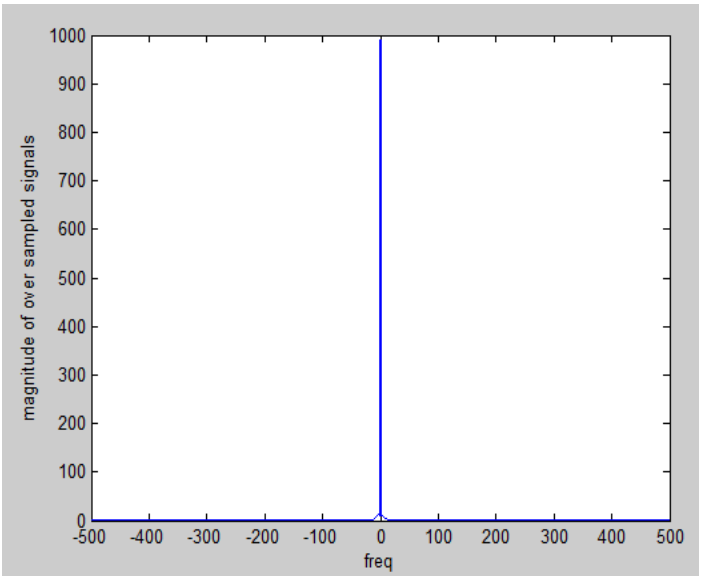
```
clear all;clc;
%% __1  construction for over sampling    "fs>Fn"
t=0:0.001:1;            % time vector   fs=1000(sampling freq)
y=2*cos(2*pi*5*t);      %the signal
[B,A] = butter(3,1000/100000,'low' );       % butter fly filter "LB"
zero_added_signal=zeros(1,length(y)*10);    %to change sample freq
for i=1:length(y)
zero_added_signal(i*10)=y(i);
end
zero_added_signal(1:9)=[];
% Adding zeros enhances the signal display and
%don't change the spectrum,it changes sampling freq. only
t=linspace(0,1,length(zero_added_signal));
filtered_signal = filter(B,A,zero_added_signal);
figure
plot(t,filtered_signal,'r' )
xlabel('time')
ylabel('oversampled signals')
s=fft(filtered_signal);
s=fftshift(s);
fs=1000; % f=5 FN=10 over samping fs>FN so assume fs=100 after up sampling fs=1000
freq=linspace(-fs/2,fs/2,length(s));
figure
plot(freq,abs(s))
xlabel('freq')
ylabel('magnitude of over sampled signals')
```

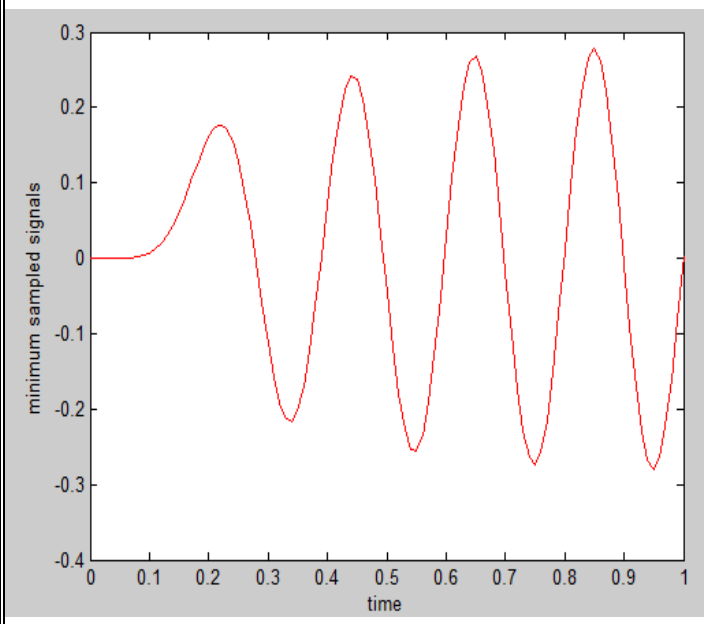
```
%% __2  construction for minimum sampling   "fs=Fn"

t=0:0.1:1;   % 0.1 because f=5 ,FN=10 at ""critcal niquect"" fs=FN=10 ,t=0:1/fs:1
y=2*cos(2*pi*5*t);
[B,A] = butter(10,0.1,'low' );
zero_added_signal=zeros(1,length(y)*10);
for i=1:length(y)
zero_added_signal(i*10)=y(i);
end
zero_added_signal(1:9)=[];
%
t=linspace(0,1,length(zero_added_signal));
filtered_signal = filter(B,A,zero_added_signal);
figure
plot(t,filtered_signal,'r' )
xlabel('time')
ylabel('minimum sampled signals')
%
s=fft(filtered_signal);
s=fftshift(s);
fs=100; % f=5 FN=10 minumum samping fs=FN so fs=10 after up sampling fs=100
freq=linspace(-fs/2,fs/2,length(s));
figure
plot(freq,abs(s))
xlabel('freq')
ylabel('magnitude of minimum sampled signals')
```

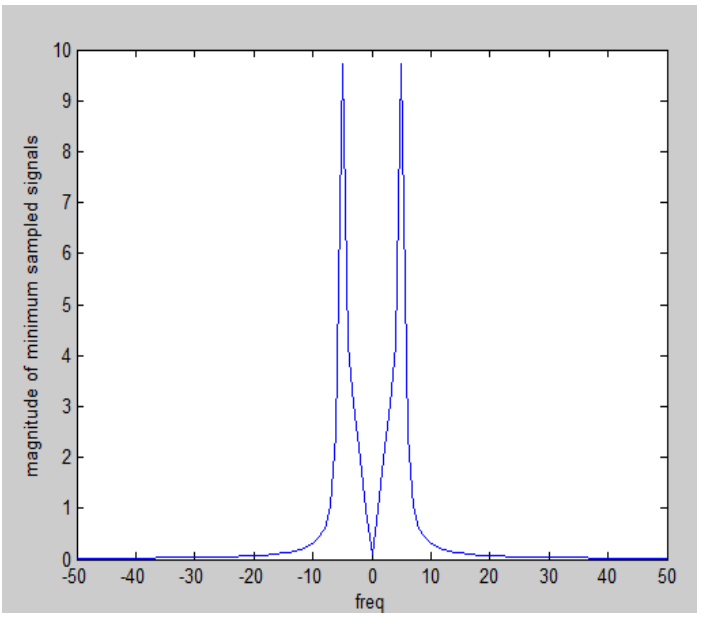
```
%% __3  construction for undersampling sampling  "fs<Fn"

t=0:0.2:1;  %fs=5 <Fn'10'
y=2*cos(2*pi*5*t);
[B,A] = butter(10,0.2,'low' );
zero_added_signal=zeros(1,length(y)*10);
for i=1:length(y)
zero_added_signal(i*10)=y(i);
end
zero_added_signal(1:9)=[];
%
t=linspace(0,1,length(zero_added_signal));
filtered_signal = filter(B,A,zero_added_signal);
figure
plot(t,filtered_signal,'r' )
xlabel('time')
ylabel('undersampling  signals')
%
s=fft(filtered_signal);
s=fftshift(s);
fs=50;  %  f=5 FN=10 under samping fs<FN so fs=5 after up sampling fs=50
freq=linspace(-fs/2,fs/2,length(s));
figure
plot(freq,abs(s))
xlabel('freq')
ylabel('magnitude of  undersampling  signals')
```

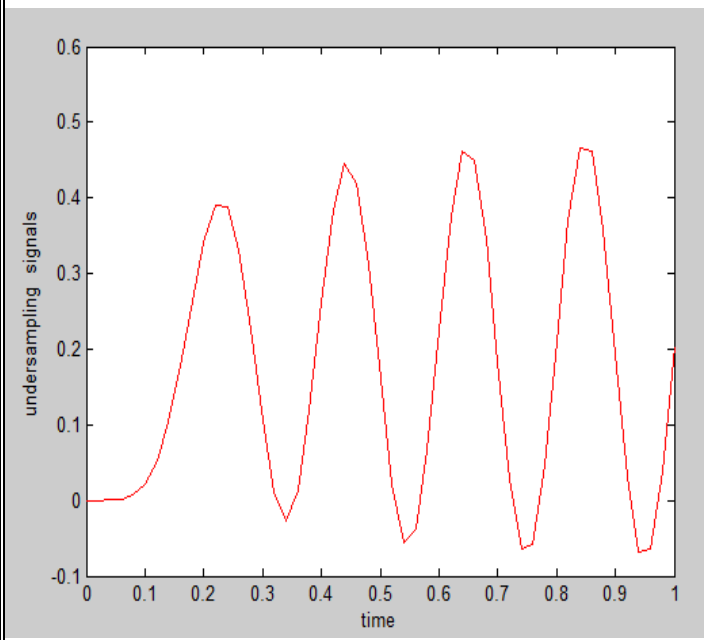
Result

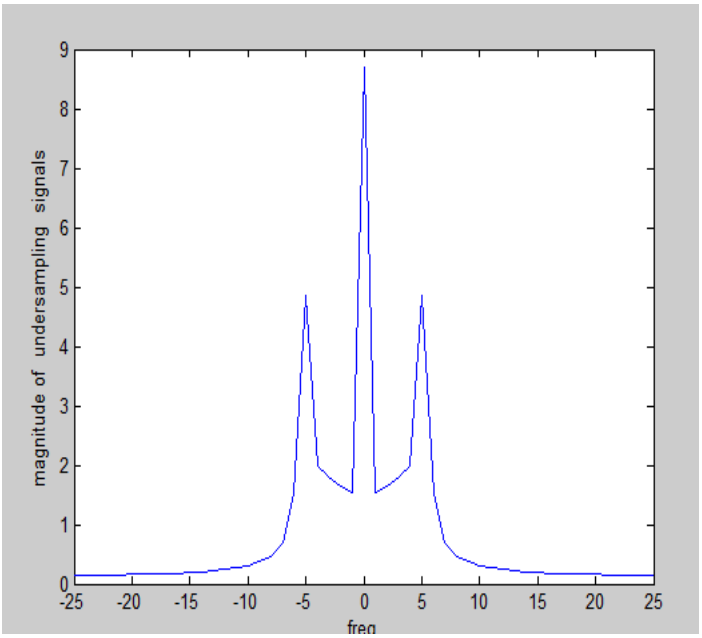

oversampled signals
time


magnitude of over sampled signals
freq


minimum sampled signals
time


magnitude of minimum sampled signals
freq


undersampling signals
time

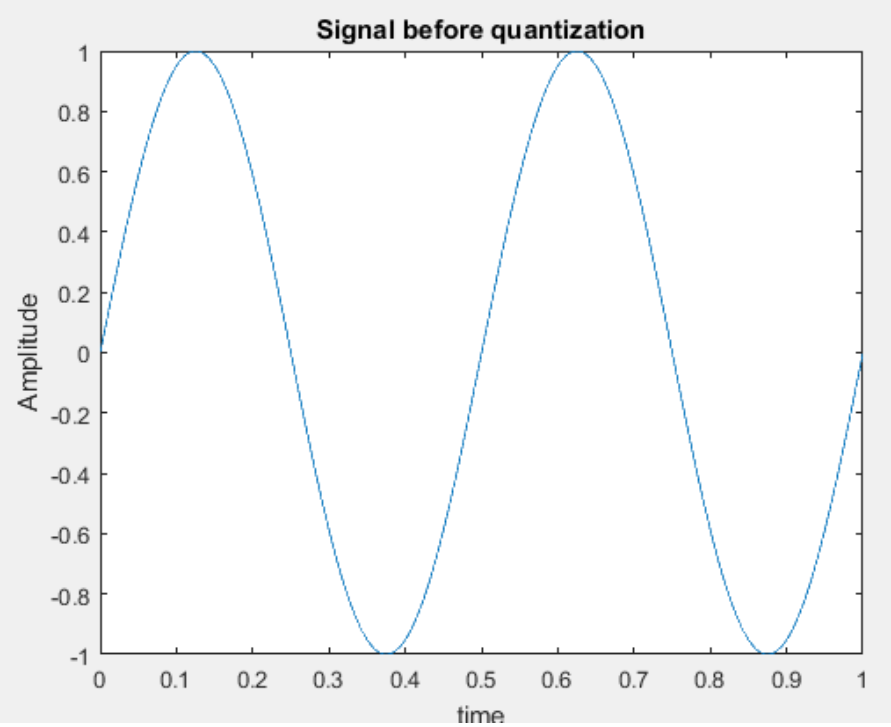

magnitude of undersampling signals
freq

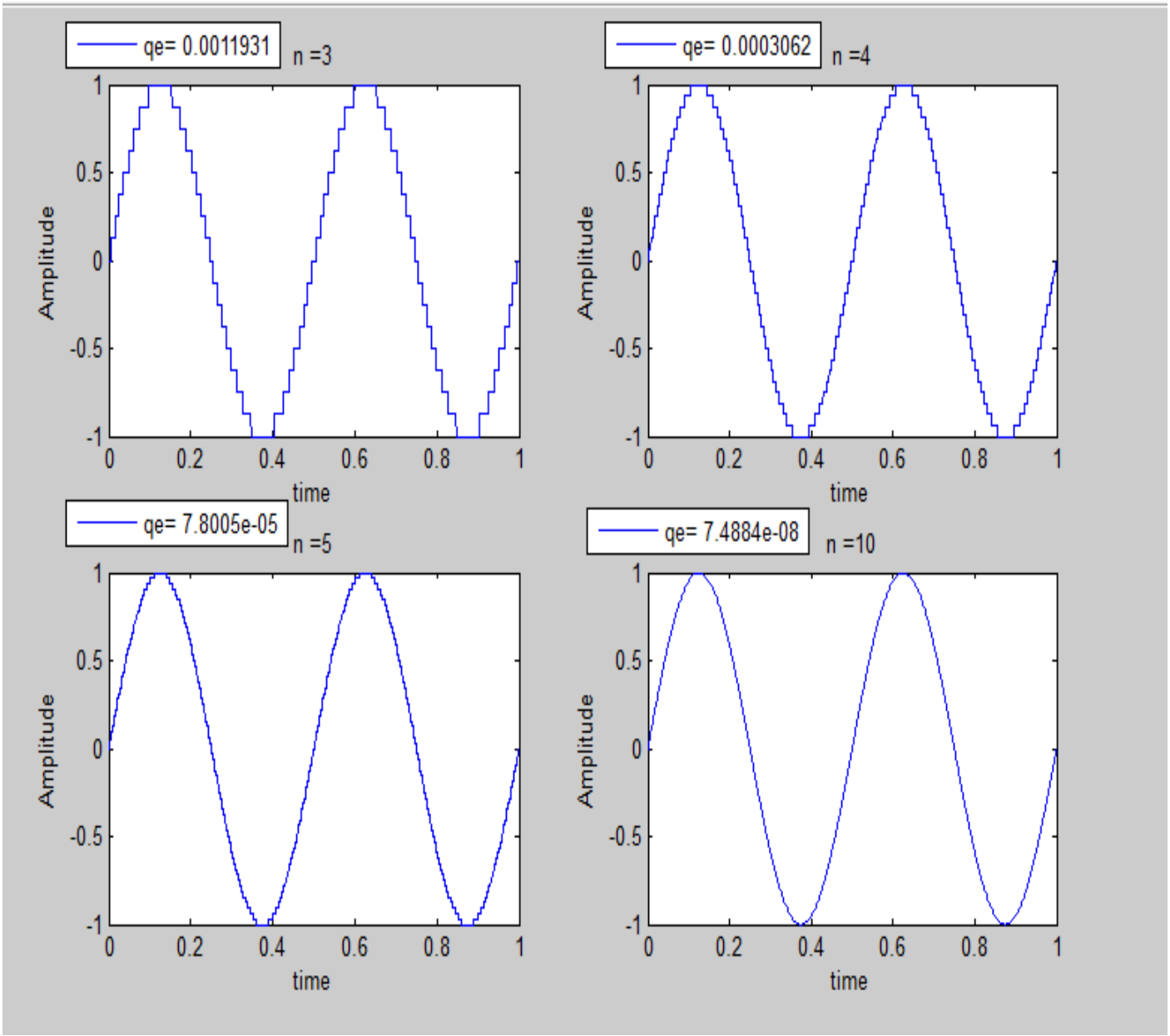
2- quantization

```
clear all;clc;
%% _ quantize by fi function
A=1;         %amplitude
f=2;         %frequancy
fs=4000;     %sampling freq
t=0:1/fs:1; %time vector
y= A*sin(2*pi*f*t); %the signal

figure
plot(t,y)
ylabel('Amplitude');
xlabel('time');
title('Signal before quantization');
%num of bits represents int value and fraction and last bit in the sign bit
n=[3,4,5,10];
m=2.*n+1;    %quantization num
figure
for i=1:length(n)
   yq= double(fi(y,1,m(i),n(i)));   %signal quantized by fi function
   Pe = sum((yq - y).^2)./length(y);            %quantization error
   subplot(2,2,i)
   plot(t,yq,'b')
   ylabel('Amplitude');
   xlabel('time');
   legend(['qe= ',num2str(Pe)])
   title(['n =',num2str(n(i))])
end
```

```
   %% encoding
   n=input('choose which n you want to encode with ( n[3,4,5,10] ): ');
   m=2*n+1;
   yq_e=double(fi(y,1,m,n));
   encoded_signal = zeros(length(y),m);
  for i=1:length(yq_e)
   if yq_e(i)< 0 % for negtive numbers
      encoded_signal(i,1) = 1;
      x = abs(yq_e(i));
   elseif yq_e(i)> 0
      encoded_signal(i,1) = 0;
      x = yq_e(i);
    end

   % number bits for integer part
   integer = n;
   encoded_signal(i,2:4) = fix(rem(x*pow2(-(n-1):0),2));

   % number bits for  fraction
   fraction = n;
   encoded_signal(i,5:end) = fix(rem(x*pow2(1:n),2));

   end
```

Result


Signal before quantization
Amplitude
time

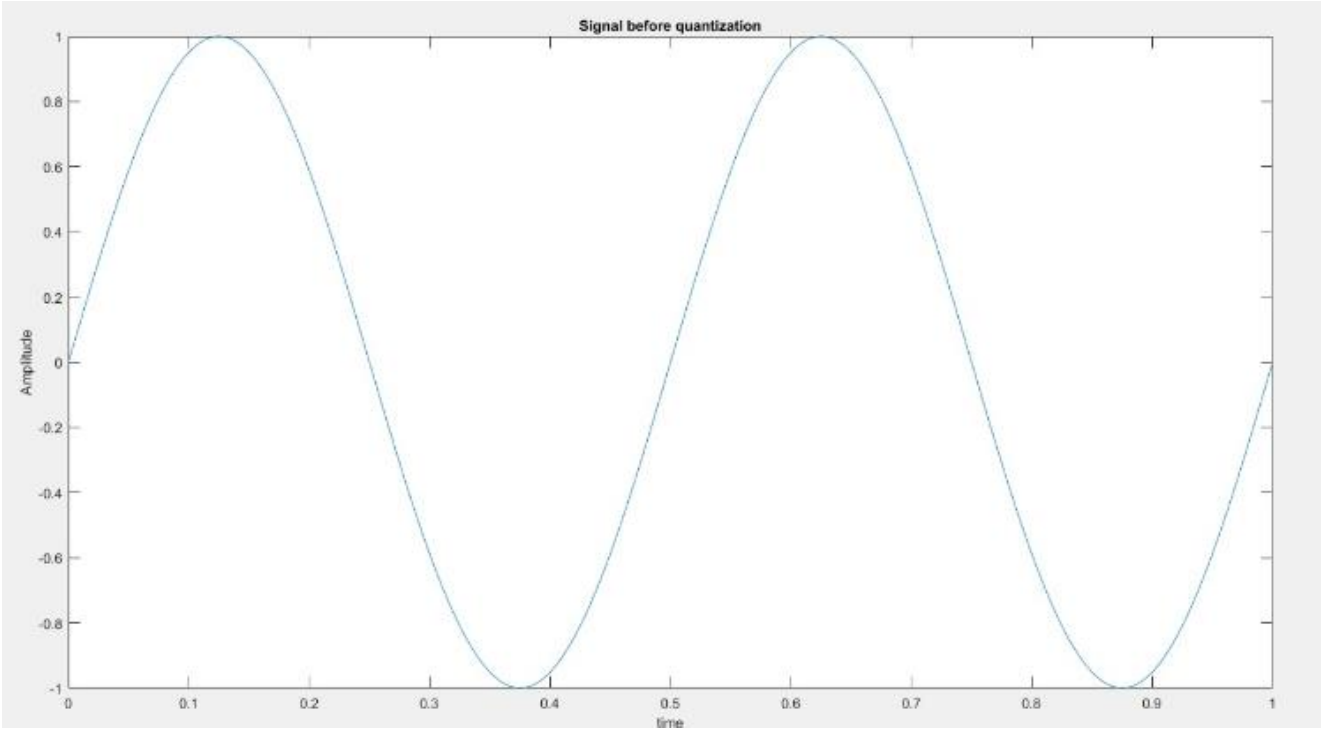

qe= 0.0011931
n =3
qe= 0.0003062
n =4
qe= 7.8005e-05
n =5
qe= 7.4884e-08
n =10
Amplitude
time

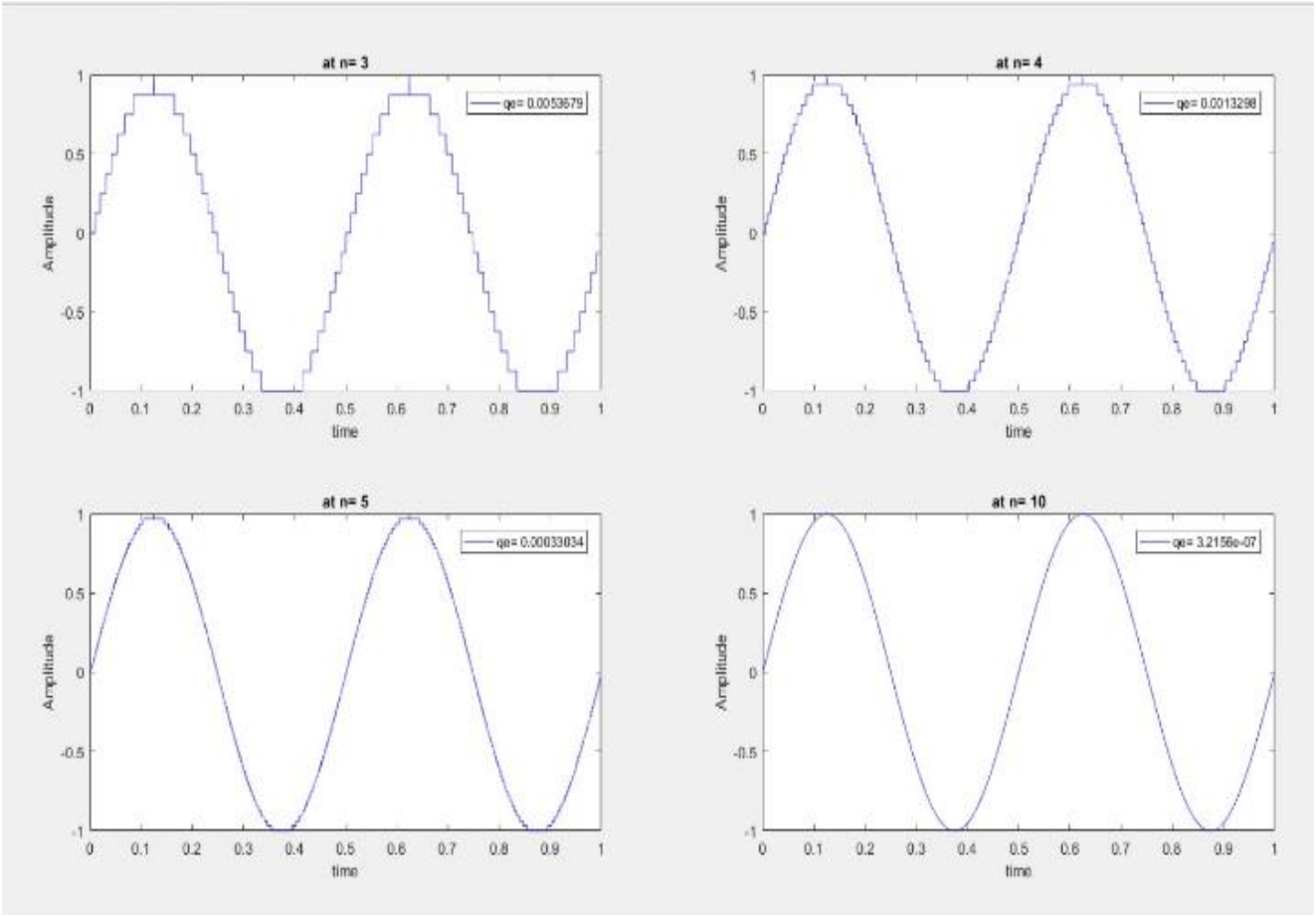
3- Quantize

```
%% _ quantize by quantiz function
A=1;        %amplitude
f=2;        %frequancy
fs=4000;    %sampling freq
t=0:1/fs:1; %time vector
y= A*sin(2*pi*f*t); %the signal

figure
plot(t,y)
ylabel('Amplitude');
xlabel('time');
title('Signal before quantization');

figure
n=[3,4,5,10];%num of bits represents int value and fraction and last bit in the sign bit
m=2.*n+1;    %quantization num
for i=1:length(n)
   q =  quantizer( [m(i) n(i)] , 'fixed');
   yq=quantize(q, y);              %signal quantized by quantiz function
   Pe = sum((yq - y).^2)./length(y);%quantization error
   subplot(2,2,i)
   plot(t,yq,'b')
   legend(['qe= ',num2str(Pe)])
   ylabel('Amplitude');
   xlabel('time');
   legend(['qe= ',num2str(Pe)])
   title(['at n= ',num2str(n(i))])
end
```

```
%% encoding
n=input('choose which n you want to encode with ( n[3,4,5,10] ): ');
m=2*n+1;
quantizer( [m n] , 'fixed');
yq_e=quantiz(q , y);
encoded_signal = zeros(length(y),m);
for i=1:length(yq_e)
 if yq_e(i)< 0 % for negtive numbers
    encoded_signal(i,1) = 1;
    x = abs(yq_e(i));
 elseif yq_e(i)> 0
    encoded_signal(i,1) = 0;
    x = yq_e(i);
  end

% number bits for integer part
integer = n;
encoded_signal(i,2:4) = fix(rem(x*pow2(-(n-1):0),2));

% number bits for  fraction
fraction = n;
encoded_signal(i,5:end) = fix(rem(x*pow2(1:n),2));

end
```

Result


Signal before quantization
Amplitude
time


at n= 3
qe= 0.0053679
at n= 4
qe= 0.0013298
at n= 5
qe= 0.00033034
at n= 10
qe= 3.2156e-07
Amplitude
time

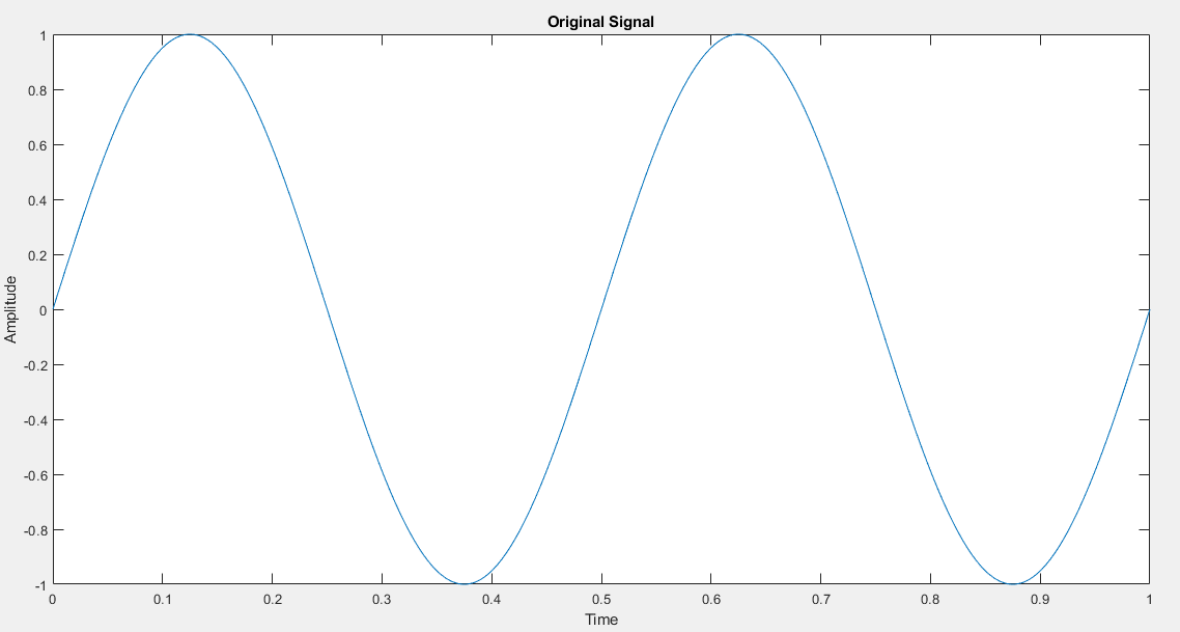
4-Non-Uniform Quantization

```
clear all;close all; clc;

Fs = 4000;
F = 2;
t = 0:1/Fs:1;
y = sin(2*pi*F*t);
%%%%Plot
figure(1);
plot(t,y);
title('Original Signal')
xlabel('Time'); ylabel('Amplitude')

% Non-Uniform Quantization/Companding
n = [3, 4, 5, 10];
m = 2.*n+1;          %Quantization Levels
law_param_mu = 255;
law_param_A = 87.6;
%mu/compressor
for i=1:length(n)
    compressed_mu = compand(y,255,max(y),'mu/compressed');
    quantized_mu =double(fi(compressed_mu,1,m(i),n(i)));
    mse_mu = sum((quantized_mu - y).^2) / length(y);
    figure(1);
    subplot(2,2,i)
    plot(t,quantized_mu)
    xlabel(' time')
    ylabel(' quantization mu samples')
    legend(['mse_mu= ',num2str((mse_mu))])
    title(['n = ',num2str(n(i))])
end
```

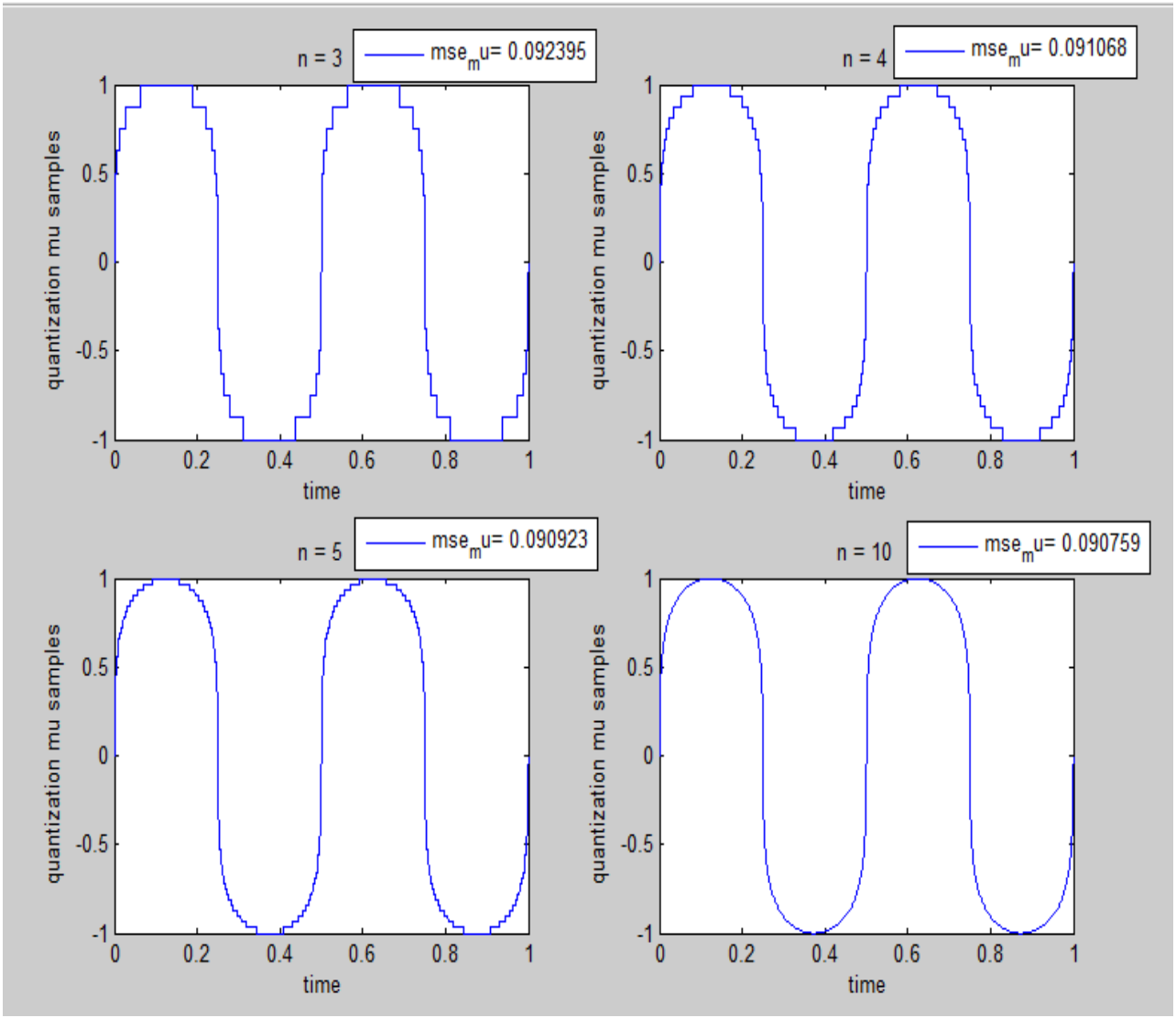
```
% A-law
for i = 1:length(n)
    compressed_A = compand(y,law_param_A,max(y),'A/compressor');
    quantized_A = double(fi(compressed_A,1 ,m(i) ,n(i)));
    MSQE_A_law =sum((quantized_A-y).^2)/length(y);
    figure(3); subplot(2,2,i);
    plot(t,quantized_A);
    xlabel('Time'); ylabel('A-law Quantized Signal');
    title(['n=', num2str(n(i))])
    legend(['MSQE=', num2str((MSQE_A_law))])
end
```

```
%% encoding
n=input('choose which n you want to encode with ( n[3,4,5,10] ): ');
yq=input('choose mu pr A : ','s');
if(yq == 'A' )
    yq_e= quantized_A;
elseif (yq == 'mu')
    yq_e= quantized_mu;
end
m=2*n+1;
encoded_signal = zeros(length(y),m);

for i=1:length(yq_e)
 if yq_e(i)< 0 % for negtive numbers
    encoded_signal(i,1) = 1;
    x = abs(yq_e(i));
 elseif yq_e(i)> 0
    encoded_signal(i,1) = 0;
    x = yq_e(i);
  end

% number bits for integer part
integer = n;
encoded_signal(i,2:4) = fix(rem(x*pow2(-(n-1):0),2));

% number bits for  fraction
fraction = n;
encoded_signal(i,5:end) = fix(rem(x*pow2(1:n),2));

end
```

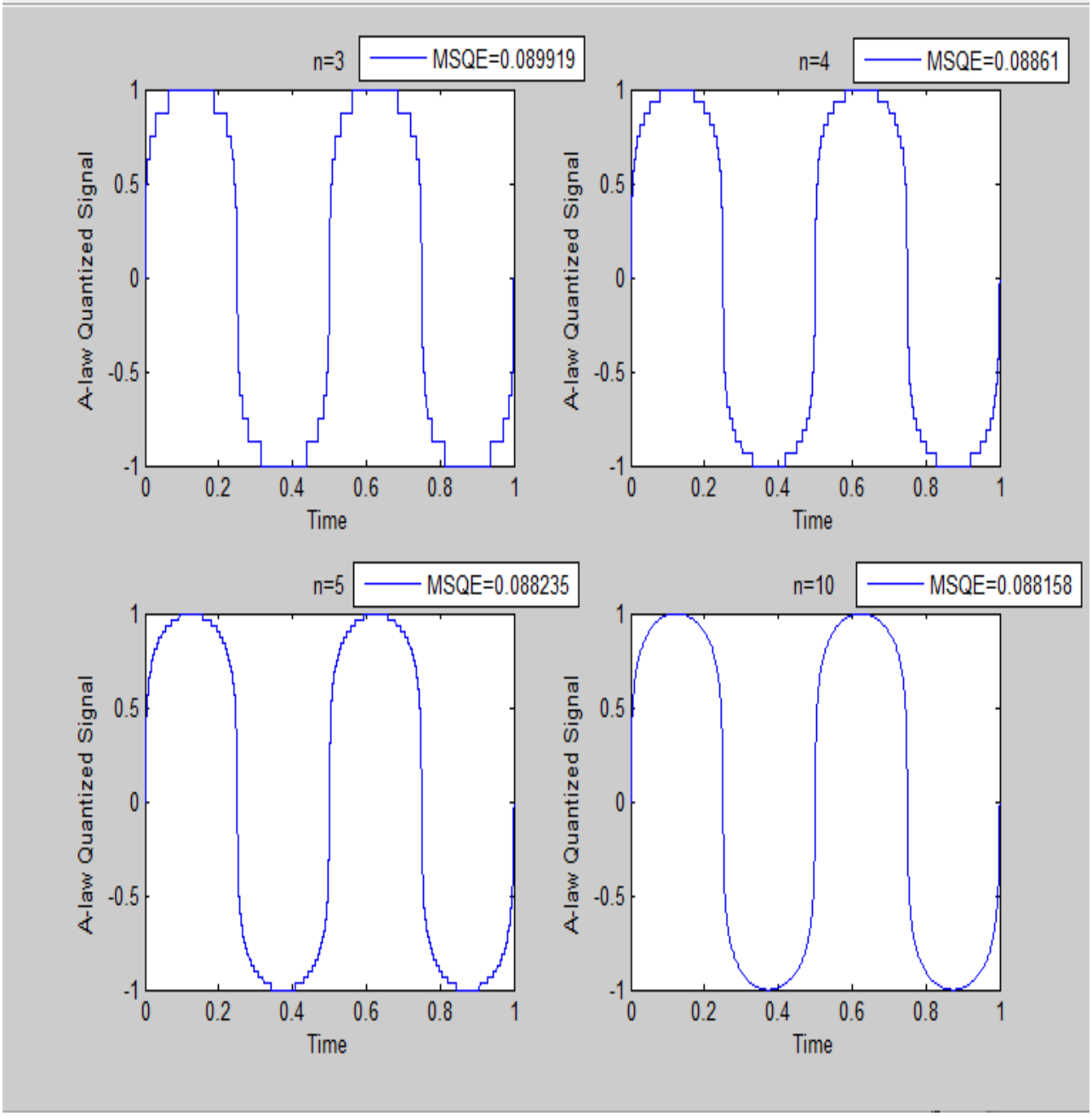
Results:



Mu law



A-law



Comment:

- As shown in the above figures, the higher the number of bits (n), the lower MSQE (mean square quantization error).