Noise in Communication System

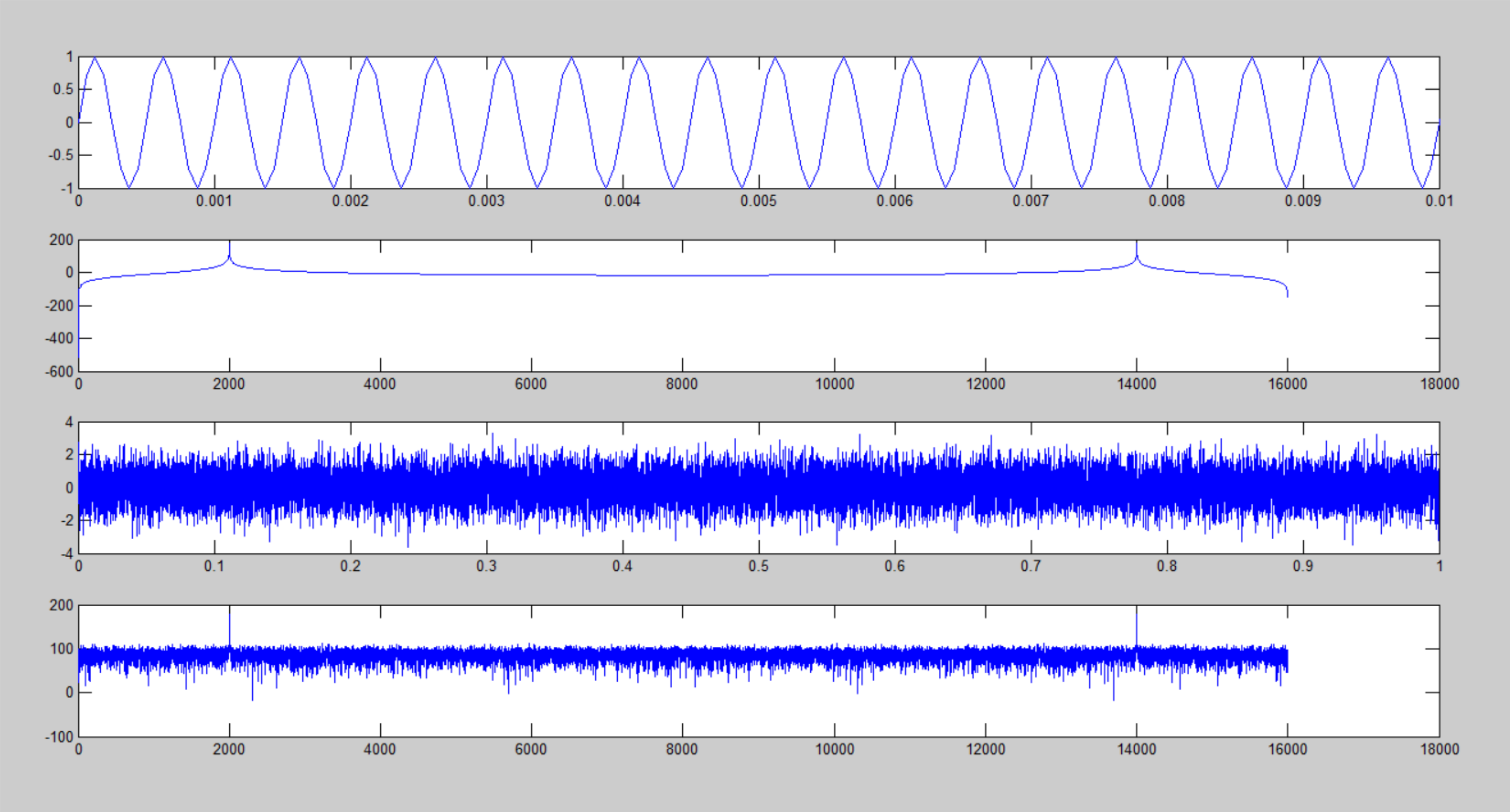
1. Addition of Noise
2. Calculation of snr

Simple noise addition

```
load chirp.mat;
sound(y, Fs);
e=awgn(y,0.8); %additive white Gaussian noise of snr 0.8db
sound(e)
```

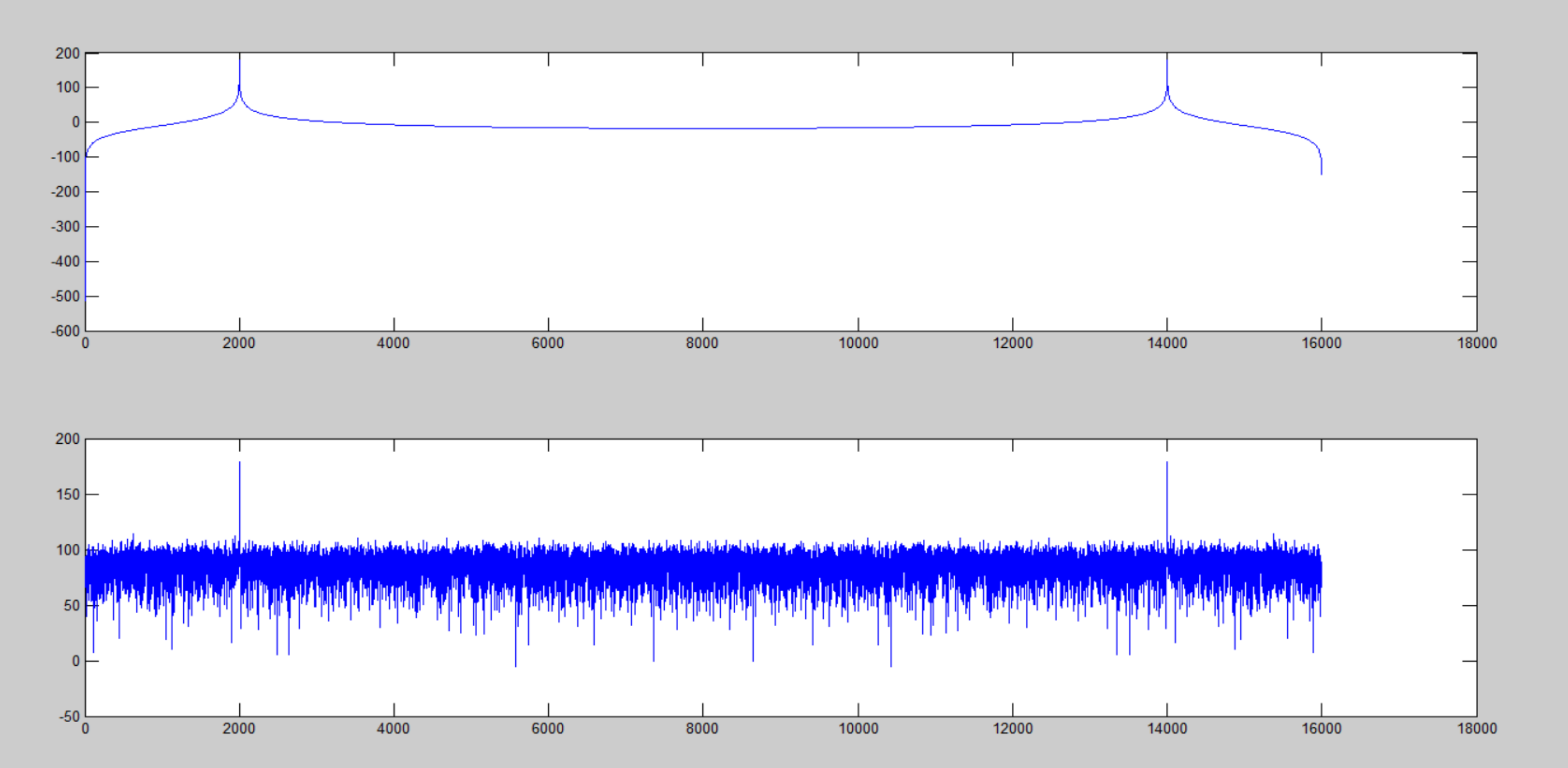
Fourier analysis of a signal

```
clear all
Fs=16000;%sampling frequency
f=2000;%signal frequency
n=[0:1/Fs:1];%sampled with sampling time 1/Fs

x=sin(2*pi*f*n);
subplot(411)
plot(n,x);axis([0 0.01 -1 1]) %original signal x
%sound(x,Fs);
subplot(412);
plot(20*log(abs(fft(x))));%PSD of signal x
y=awgn(x,3); %adds white Gaussian noise to X.  The SNR is in 3dB.
%sound(y,Fs)
subplot(413)%Signal after noise
plot(n,y)
subplot(414)
plot(20*log(abs(fft(y))));%PSD of signal noisy signal y
```

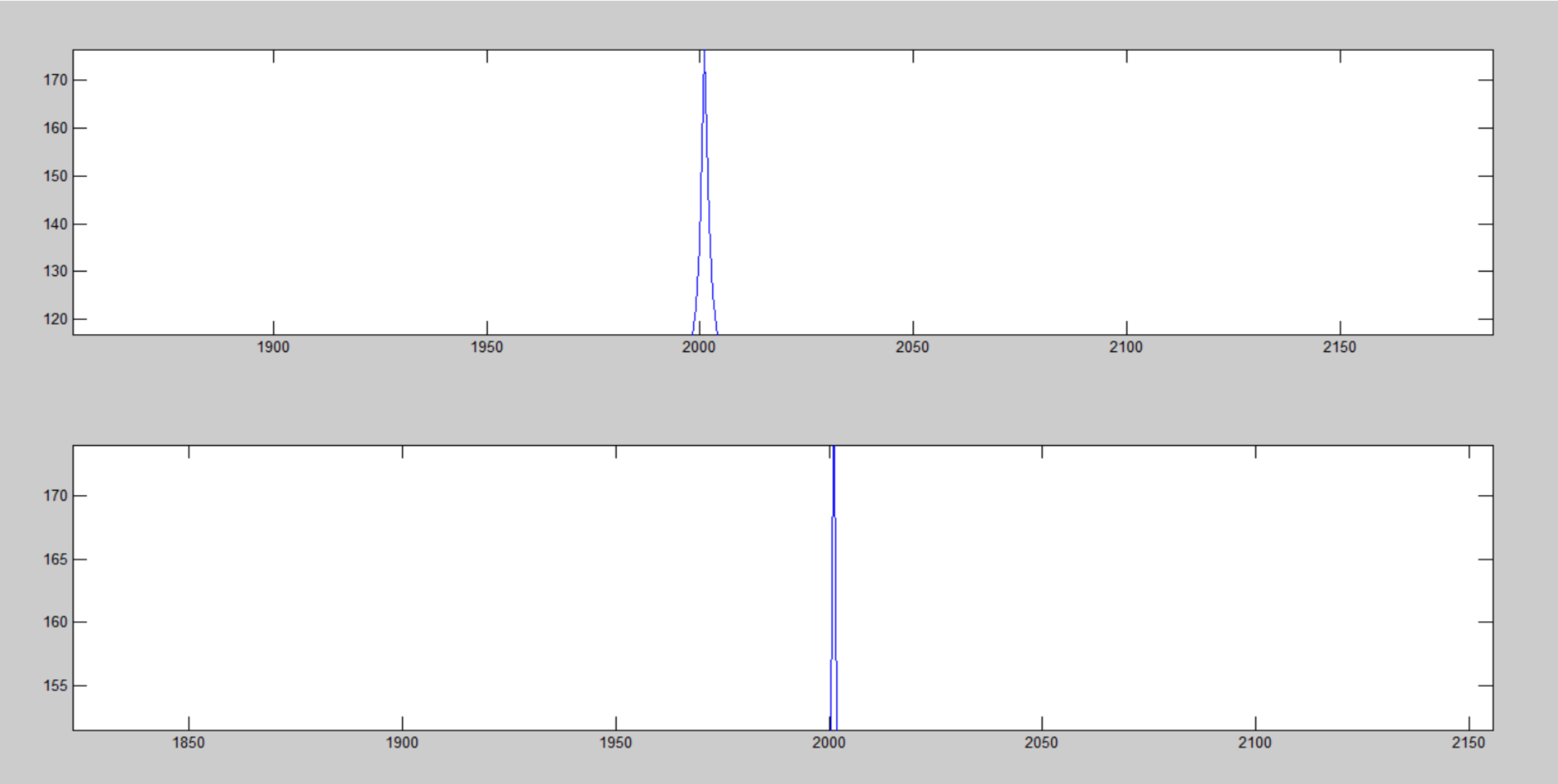


1
0.5
0
-0.5
-1
0
0.001
0.002
0.003
0.004
0.005
0.006
0.007
0.008
0.009
0.01
200
0
-200
-400
-600
0
2000
4000
6000
8000
10000
12000
14000
16000
18000
4
2
0
-2
-4
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
200
100
0
-100
0
2000
4000
6000
8000
10000
12000
14000
16000
18000

The Power Spectral Density



The peak is found at natural frequency(use zoom at the first peak in the figure)



- By knowing the natural frequency of a signal one can extract the signal by using filter.

snr calculation

```
Fi = 2500; %signal frequency
Fs = 48e3; %sampling frequency
N = 1024;  %number of samples
nsig=0.001*randn(1,N); %noise signal with…
Normally distributed pseudorandom numbers.
x0=sin(2*pi*Fi/Fs*(1:N)); %original signal
x = x0 + nsig; %addition of noise
SNR1=snr(x0,Fs) %output snr in dB of the signal
before noise
SNR = snr(x,Fs) %output snr in dB of the signal
after noise
```

Results…….

Command Window

```
>> snr_calc

SNR1 =

  281.6380


SNR =

   58.0260

>> snr_calc

SNR1 =

  281.6380


SNR =

   57.0962

>> snr_calc

SNR1 =

  281.6380


SNR =

   56.8055
```

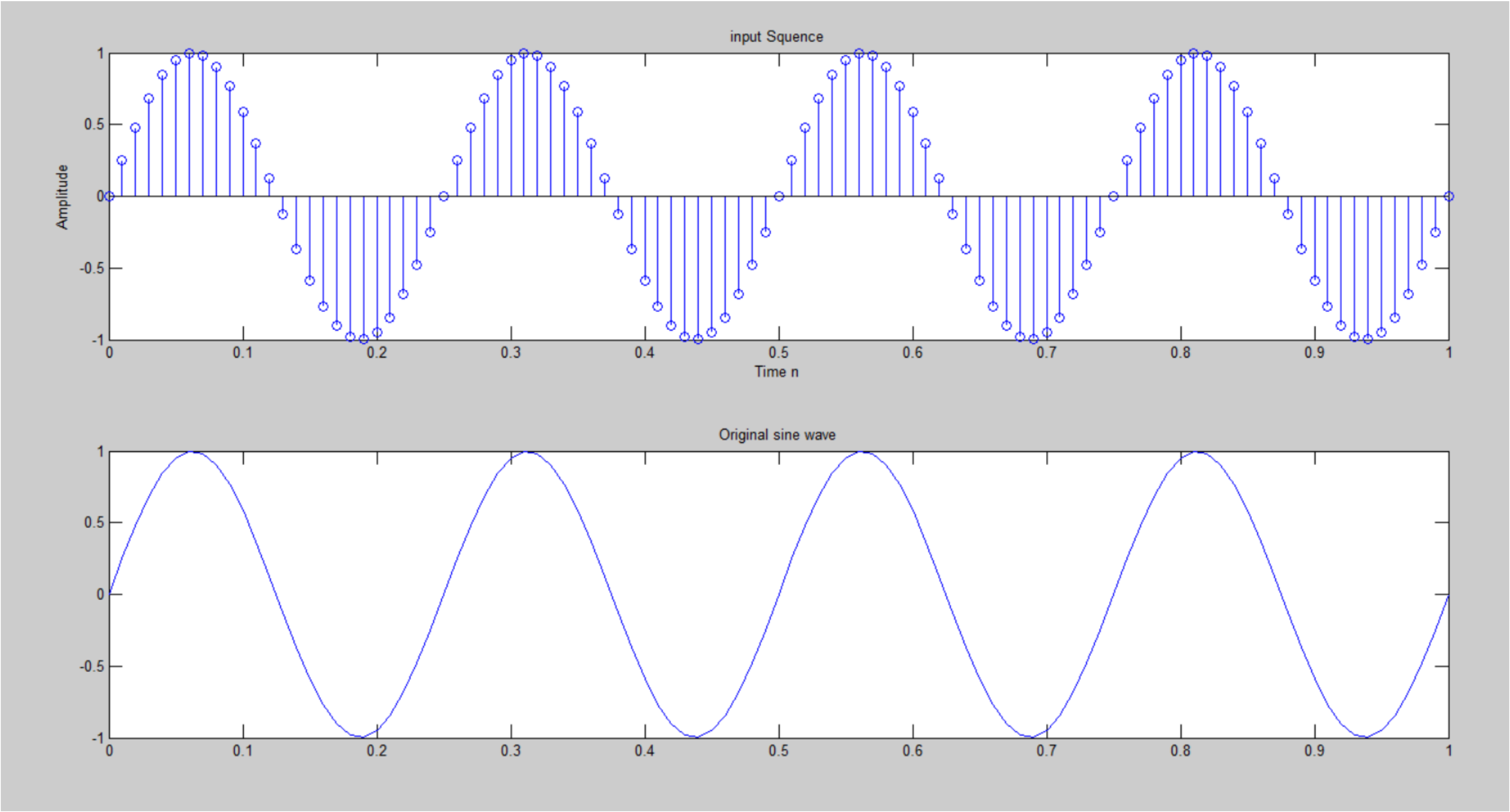
Pulse Code Modulation

1. Sampling
2. Quantization
3. Encoding

Sampling a signal

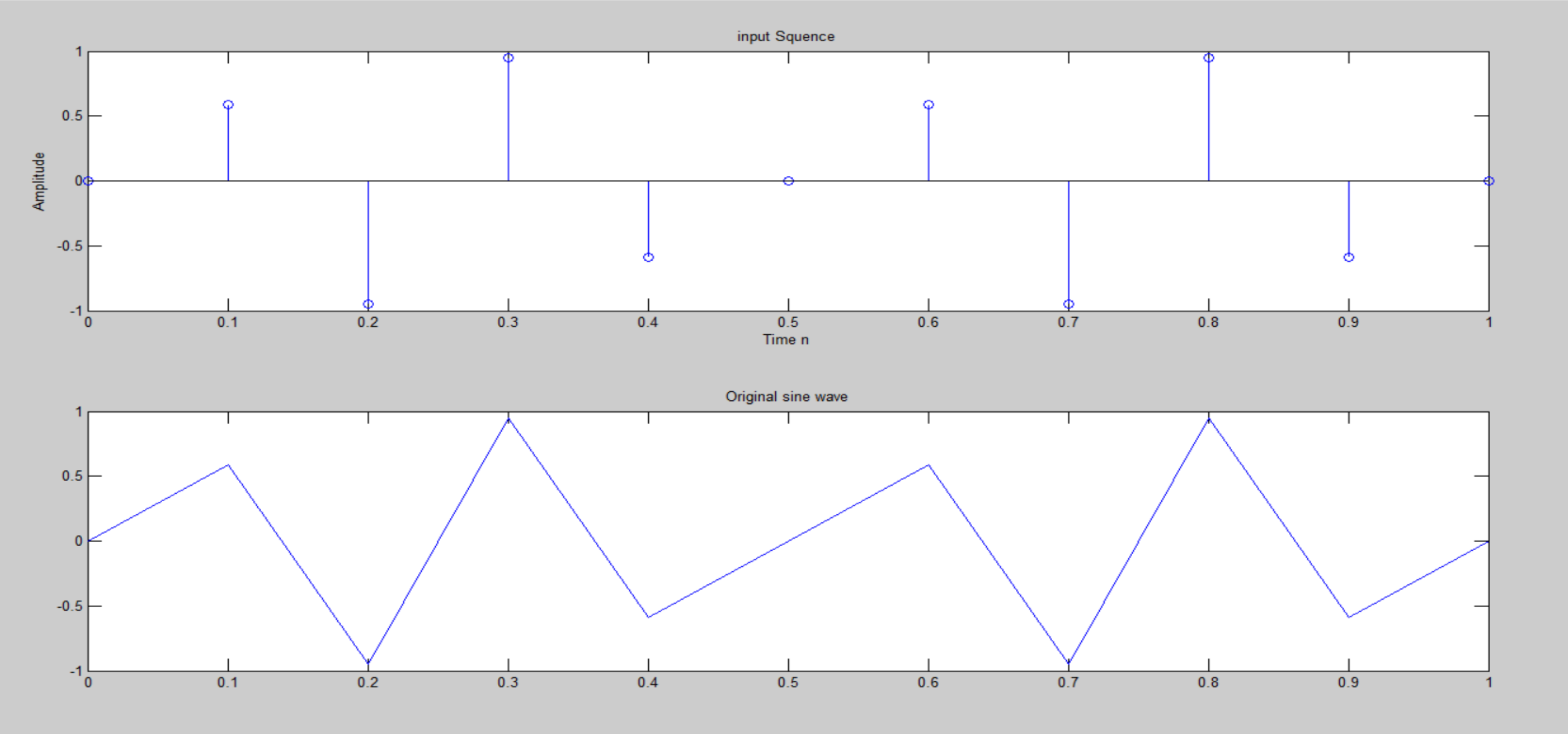
```
T=input ('Input time of the Sinusoidal signal= ');
fi=input('Input Frequency of the Sinusoidal signal= ');
n =0:0.01:T; % sampling time 0.01s
X=sin(2*pi*fi*n);
subplot(211)
stem(n,X);
title('input Squence');
xlabel('Time n');
ylabel('Amplitude');
subplot(212)
plot(n,X)
title('Original sine wave')
```

Input time of the Sinusoidal signal= 1
Input Frequency of the Sinusoidal signal= 4



If the sampling time is not small enough(i.e. sampling frequency is not high enough)

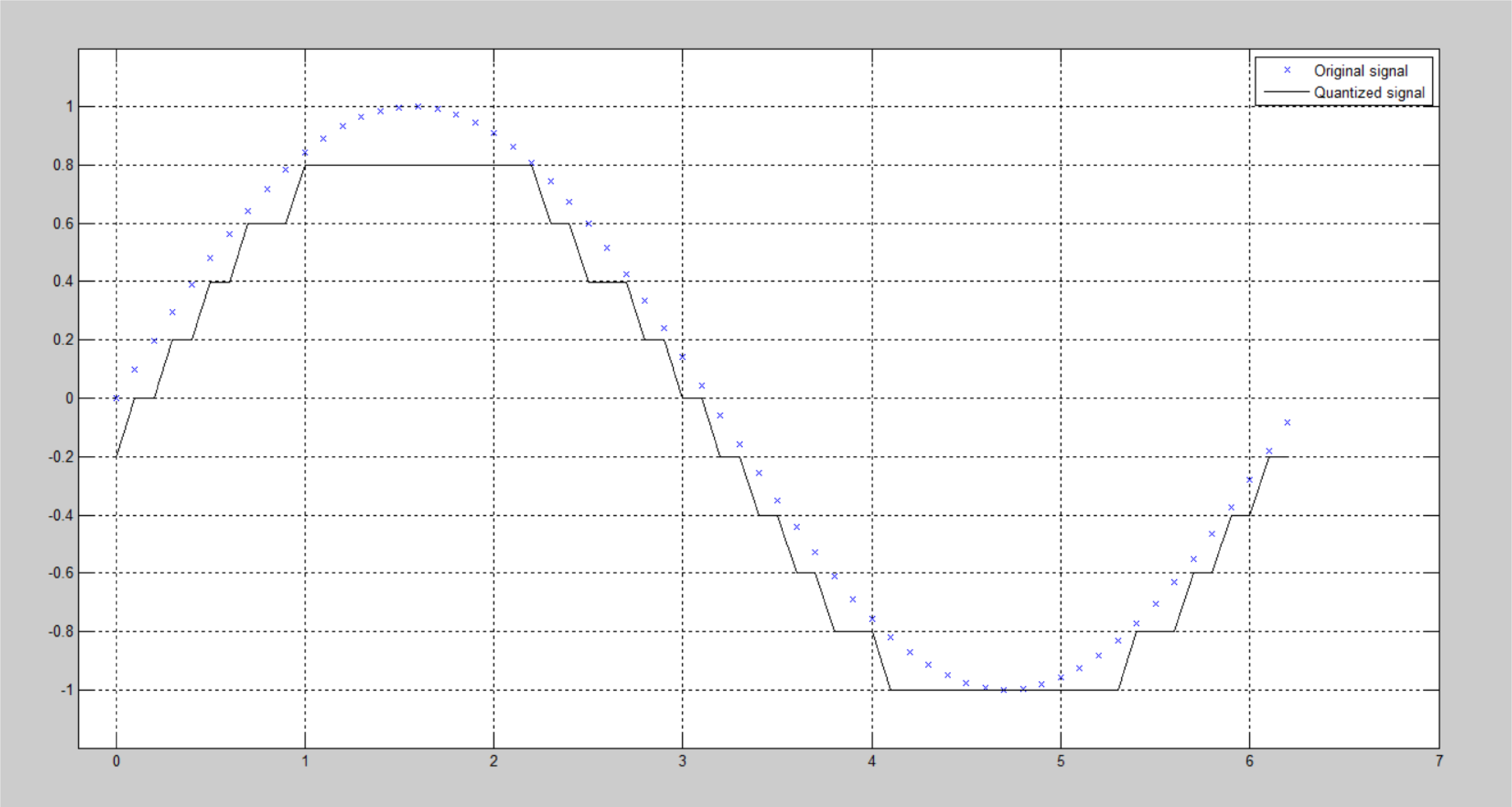
```
n =0:0.1:T; % sampling time 0.1s
```



Quantization

MATLAB Program……….

```
t = [0:.1:2*pi]; % Times at which to sample the sine function
sig = sin(t); % Original signal, a sine wave
partition = [-1:.2:1]; % Length 11, 2mp/L=10, index=1.....10
codebook = [-1.2:.2:1]; % Length 12, one entry for each interval
must be one more than the length of partiton
[index,quants] = quantiz(sig,partition,codebook); % Quantize.
plot(t,sig,'x',t,quants,'k'); grid on
legend('Original signal','Quantized signal');
axis([-.2 7 -1.2 1.2]) %Change the axis limits so that the x-axis
ranges from -0.2 to 7 and the y-axis ranges from -1.2 to 1.2.
```


Original signal
Quantized signal
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-1
0
1
2
3
4
5
6
7

In command window

- Type sig…………..press enter
- Type index……….press enter
- Type quants……..press enter

sig

Command Window

```
sig =

  Columns 1 through 8

         0    0.0998    0.1987    0.2955    0.3894    0.4794    0.5646    0.6442

  Columns 9 through 16

    0.7174    0.7833    0.8415    0.8912    0.9320    0.9636    0.9854    0.9975

  Columns 17 through 24

    0.9996    0.9917    0.9738    0.9463    0.9093    0.8632    0.8085    0.7457

  Columns 25 through 32

    0.6755    0.5985    0.5155    0.4274    0.3350    0.2392    0.1411    0.0416
```

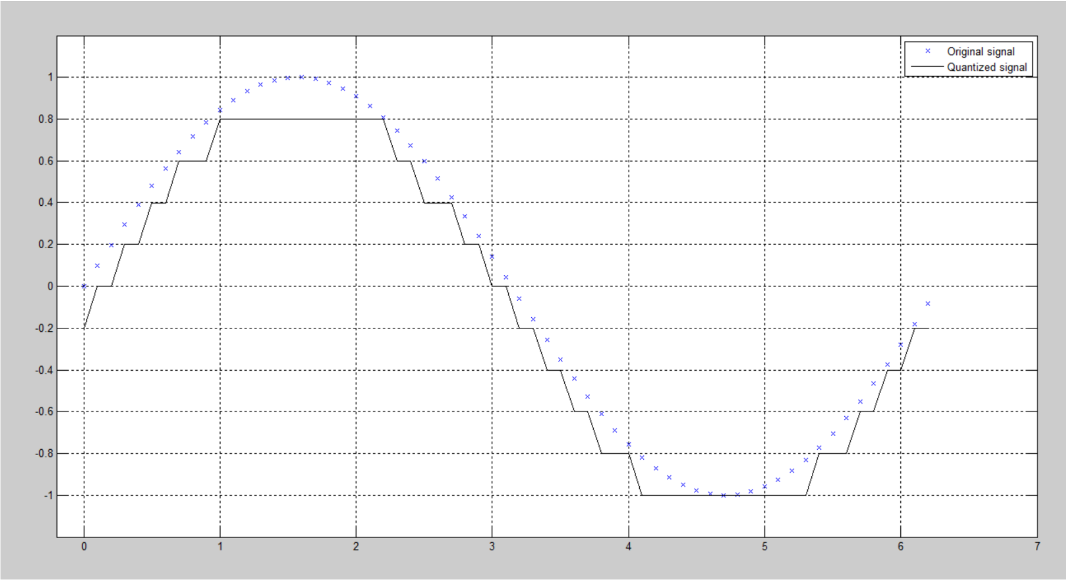
index

```
>> index

index =

  Columns 1 through 19

     5     6     6     7     7     8     8     9     9     9    10    10    10    10    10    10    10    10    10

  Columns 20 through 38

    10    10    10    10     9     9     8     8     8     7     7     6     6     5     5     4     4     3     3

  Columns 39 through 57

     2     2     2     1     1     1     1     1     1     1     1     1     1     1     1     1     2     2     2

  Columns 58 through 63

     3     3     4     4     5     5
```

quants

```
>> quants

quants =

  Columns 1 through 11

   -0.2000         0         0    0.2000    0.2000    0.4000    0.4000    0.6000    0.6000    0.6000    0.8000

  Columns 12 through 22

    0.8000    0.8000    0.8000    0.8000    0.8000    0.8000    0.8000    0.8000    0.8000    0.8000    0.8000

  Columns 23 through 33

    0.8000    0.6000    0.6000    0.4000    0.4000    0.4000    0.2000    0.2000         0         0   -0.2000

  Columns 34 through 44

   -0.2000   -0.4000   -0.4000   -0.6000   -0.6000   -0.8000   -0.8000   -0.8000   -1.0000   -1.0000   -1.0000
```

- The x signal in the figure shows the sampling rate which is equal to the length of time period t.
- In command window type t and enter. Or type `length(t)`. You will find 63 point. So the number of x will be 63. Also you can see in workspace.



Partition and Codebook

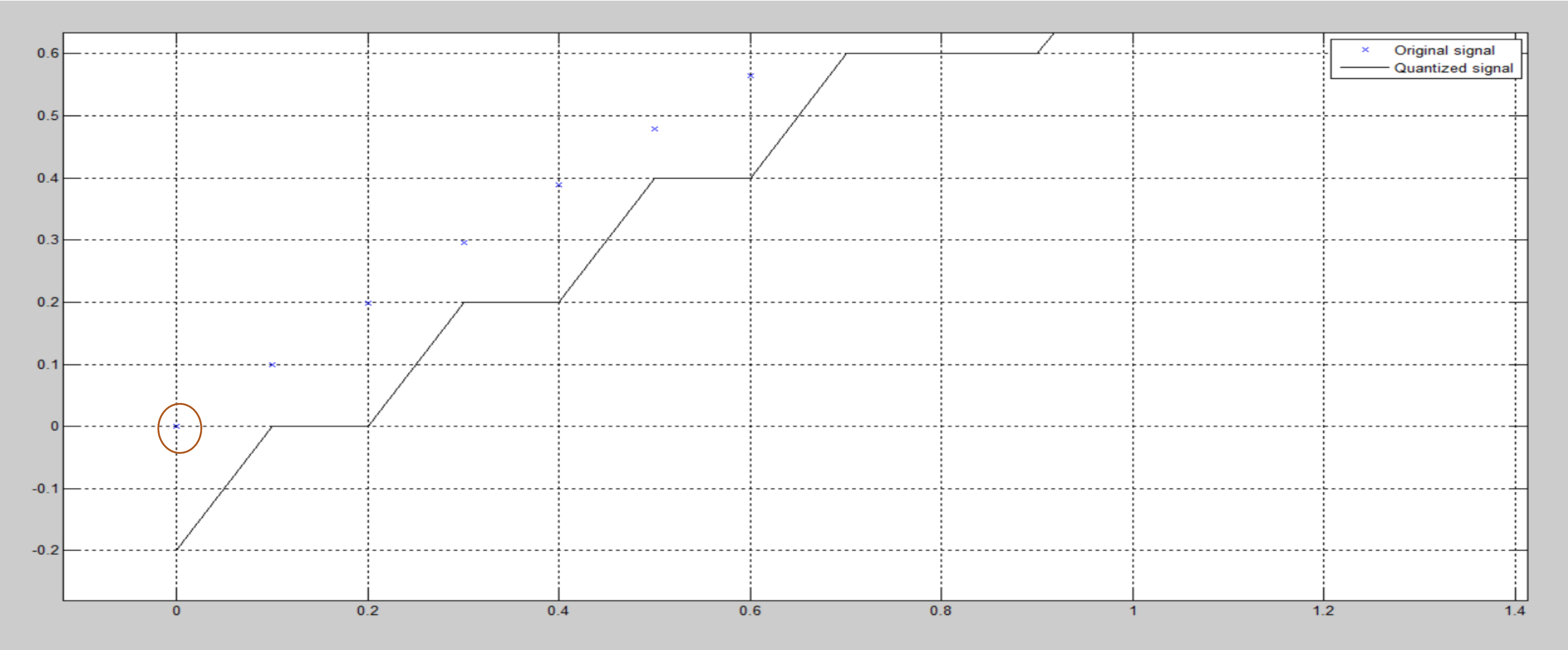
```
t = [0:.1:2*pi]; % Times at which to sample the sine function
sig = sin(t); % Original signal, a sine wave
partition = [-1:.2:1]; % Length 11, 2mp/L=10, index=1.....10
codebook = [-1.2:.2:1]; % Length 12, one entry for each interval must be
one more than the length of partiton
[index,quants] = quantiz(sig,partition,codebook); % Quantize.
plot(t,sig,'x',t,quants,'k'); grid on
legend('Original signal','Quantized signal');
axis([-.2 7 -1.2 1.2]) %Change the axis limits so that the x-axis ranges
from -0.2 to 7 and the y-axis ranges from -1.2 to 1.2.
```

Type *index* in command window

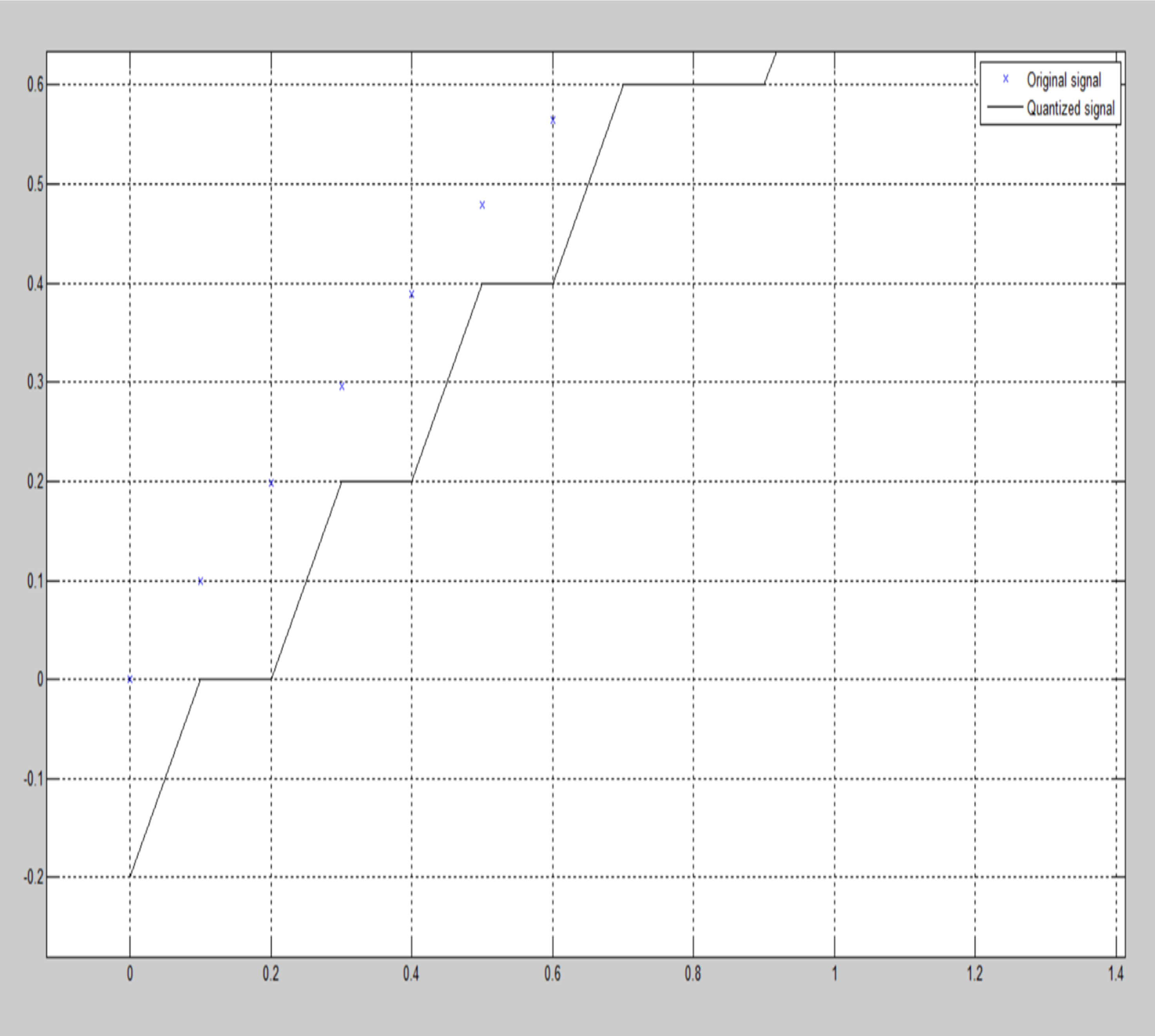
```
>> index

index =

  Columns 1 through 13

     5     6     6     7     7     8     8     9     9     9    10    10    10

  Columns 14 through 26

    10    10    10    10    10    10    10    10    10    10     9     9     8

  Columns 27 through 39

     8     8     7     7     6     6     5     5     4     4     3     3     2

  Columns 40 through 52

     2     2     1     1     1     1     1     1     1     1     1     1     1

  Columns 53 through 63

     1     1     2     2     2     3     3     4     4     5     5
```

- Index defines which value to be taken in which portion of the partition.
- Number/Length of index will be equal to length of t. In this case 63

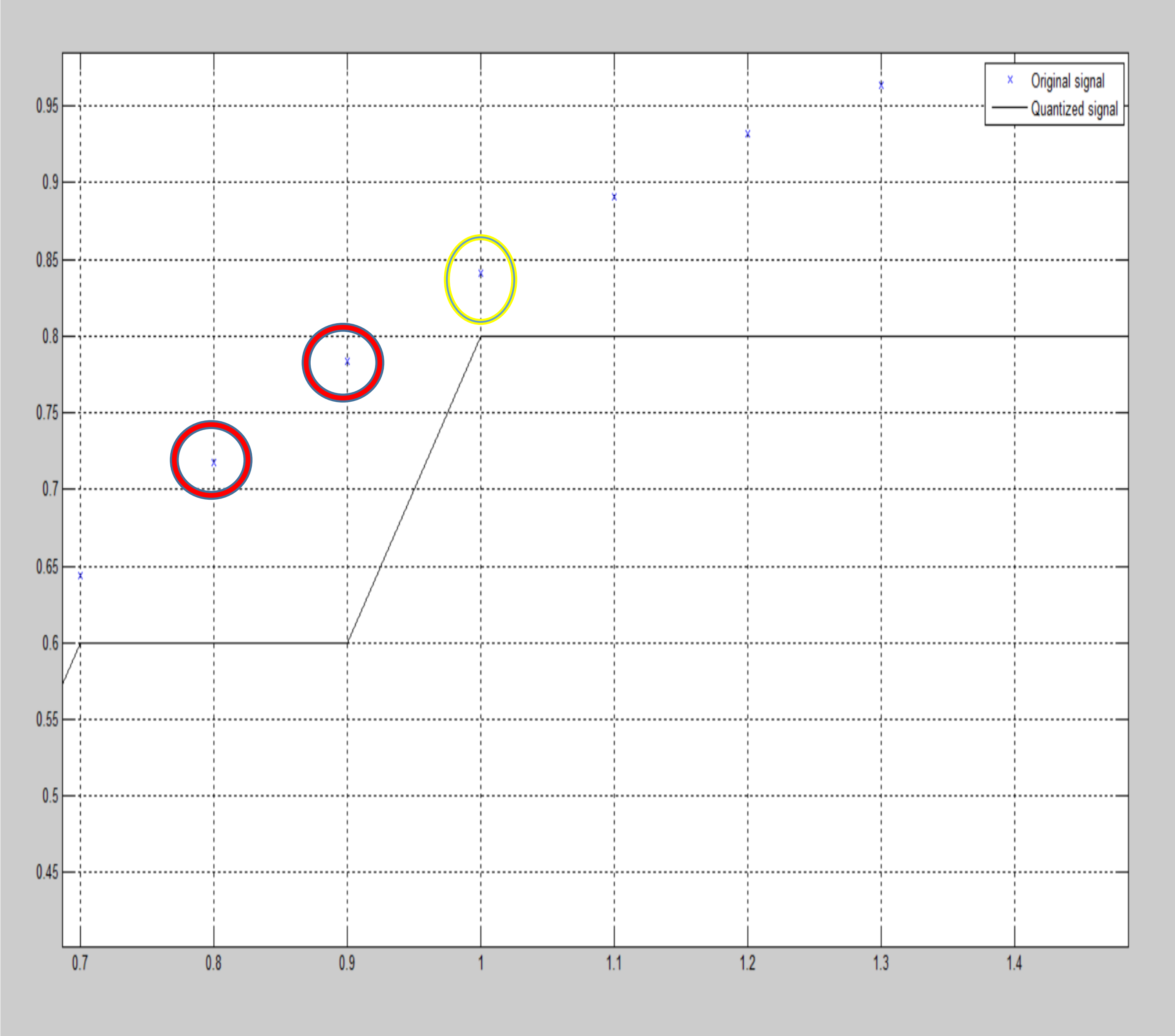
Consider the first sample



- Its in 0…..in the 5th index
- That's why when you typed index the first position was 5
- The next two is in 6th index.
- And so on goes the index.
- The quantized signal changes its position when index changes.



- As the step size is 0.6 to 0.8 these two (marked red) are within range so the quantized signal changed for the next one after 0.8



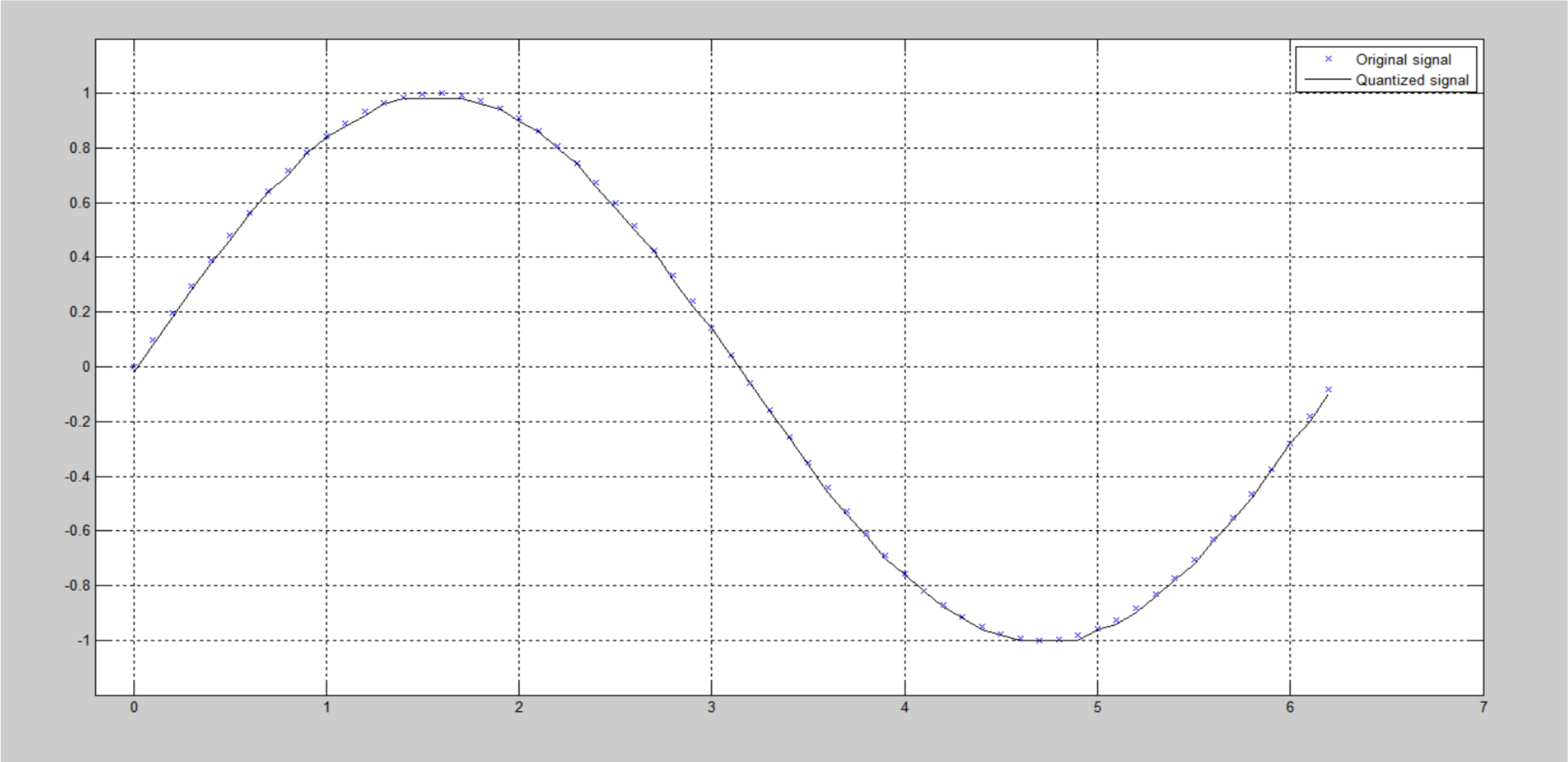
Encode

- To encode the quantization to binary digits type…………………
  `dec2bin(index)`

```
>> dec2bin(index)

ans =

0101
0110
0110
0111
0111
1000
1000
1001
1001
1001
1010
1010
1010
1010
1010
1010
1010
1010
1010
1010
```

To remove quantization noise.....increase levels(partition)

```
t = [0:.1:2*pi]; % Times at which to sample the sine function
sig = sin(t); % Original signal, a sine wave
partition = [-1:.02:1];
codebook = [-1.02:.02:1];
[index,quants] = quantiz(sig,partition,codebook); % Quantize.
plot(t,sig,'x',t,quants,'k'); grid on
legend('Original signal','Quantized signal');
axis([-.2 7 -1.2 1.2])
```



Original signal
Quantized signal
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-1
0
1
2
3
4
5
6
7

Lab Report 1: Analysis of Noise Signals using MATLAB

1. Find out the natural frequency of a noisy signal using fast Fourier transform function.
2. Find snr of a signal before and after addition of noise

Lab Report 2: Realization of sampling, quantization and encoding of PCM by MATLAB

1. Sample a sinusoidal signal
2. Quantize
3. Encode (show the encoded binary bits)
4. Show a quantized sinusoidal signal by decreasing quantization noise from the previous signal. (Show that quantization noise can be decreased with the cost of increased number of bits)