Concordia University
Engineering and Computer Science

LABORATORY REPORT

ELEC 342 Discrete Time Signals and Systems

FALL 2023

Course:	ELEC 342	**Lab Section:**	UN-X
Experiment No.:	3	**Date Performed:**	2023 – 10 –24 (YYYY – MM – DD)
Experiment Title:	Functions in MATLAB and the Sampling Theorem		
Name:	BAYAN ALSALEM	**ID No.:**	40105034

I certify that this submission is my original work and meets the Faculty's Expectations of Originality

Signature: [illegible] **Date:** 2023 – 10 – 31
YYYY – MM – DD

Table of Contents

Objectives

The main objective is to utilize the built-in MATLAB functions to analyze discrete properties of a signal such as the sampling rate of the Fourier transform. Moreover, there are many other built-in functionalities that are implemented such as floor, function, etc. Lastly, the second part of the lab explores the exportation of original and distorted signals from wav files.

Theory

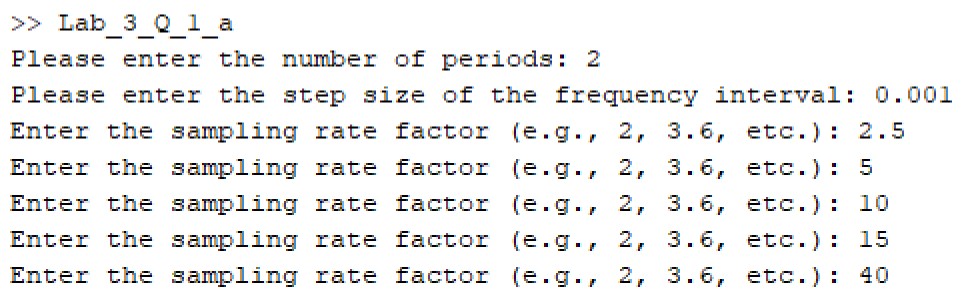
Sampling refers to the process of converting a continuous-time signal into a discrete-time signal by selecting specific points in time at which the signal is measured or sampled. This process is essential for various applications, including digital signal processing and data acquisition. Sampling theorem is a key part of discrete signal analysis where increasing the number of samples can produce a closer representation in data to the original signal. The higher the number of samples, the greater the accuracy of analysis. The rate at which a continuous-time signal is sampled is referred to as the "sampling rate" or "sampling frequency." It is typically denoted as "Fs" and is measured in Hertz (Hz). A higher sampling rate captures more details of the original signal but requires more data storage and processing.

Tasks, Results, Discussion and Questions

Question 1

(a) This question will investigate the <u>effect of varying the sampling rate</u> on the transform of the sampled signal. Write a MATLAB script which will prompt the user to specify the number of periods and the step size over which the transform is to be computed. This will define the `w` array as `[ -n *pi: step_size : n*pi]`
where n is the number of periods. The array `w` should be declared to be global.
Re-write your code used in Lab #2 as a function which receives a signal `x(n)` and returns the Discrete Time Fourier Transform of the signal. Make use of a function to compute the value of `N` for a given value of sampling rate.
The program is to ask the user to input the value of the sampling rate (in terms of the number of times of the Nyquist rate, i.e. 2 times, 3.6 times, etc). The program then plots in one figure

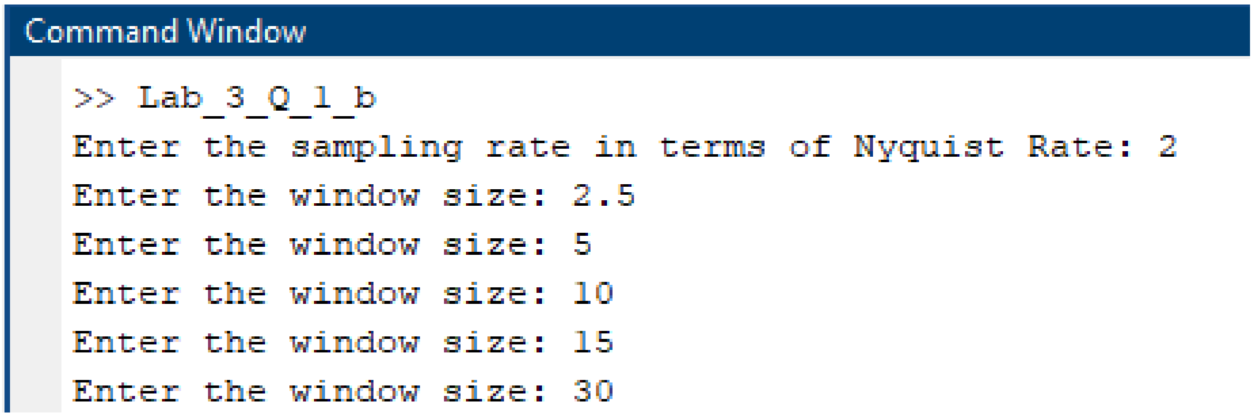
window the sampled signal (using stem) and the Fourier transform of the sampled signal. This is to be repeated 5 times.



```
>> Lab_3_Q_1_a
Please enter the number of periods: 2
Please enter the step size of the frequency interval: 0.001
Enter the sampling rate factor (e.g., 2, 3.6, etc.): 2.5
Enter the sampling rate factor (e.g., 2, 3.6, etc.): 5
Enter the sampling rate factor (e.g., 2, 3.6, etc.): 10
Enter the sampling rate factor (e.g., 2, 3.6, etc.): 15
Enter the sampling rate factor (e.g., 2, 3.6, etc.): 40
```

Please refer to the MATLAB code in figure 1 and the 5 plots showing the effect of varying the sampling rate in figures 2, 3, 4, 5, and 6.

(b) To investigate the effect of changing the window size (i.e. how many samples of the signal are obtained at some fixed sampling rate) we will rewrite the code in part (a) to ask the user to input the sampling rate (in terms of the Nyquist rate). This sampling rate will be kept constant and the program is to ask the user to enter the window size (in terms of the number periods of the signal). The sampled signal (over the total number of periods as defined by the value of the window size) and its Fourier transform is to be plotted in one figure window. This is to be repeated 5 times with a different value for the window size in each loop iteration.



```
>> Lab_3_Q_1_b
Enter the sampling rate in terms of Nyquist Rate: 2
Enter the window size: 2.5
Enter the window size: 5
Enter the window size: 10
Enter the window size: 15
Enter the window size: 30
```

Please refer to the MATLAB code in figure 7 and the 5 plots showing the effect of varying the window size in figures 8, 9, 10, 11, and 12.

Question 2

Repeat Question 1(a) using polar plots for the transform instead of rectangular plots. Use the following for the input signal

$$x[n] = 0.5sin\left(\frac{2\pi}{N}n\right) + 0.33sin\left(\frac{4\pi}{N}n\right)$$

For this question, the code for Question 1 part a) was reused but with a different input signal x[n] of 0.5*sin(2*pi/N * n) + 0.33* sin(4*pi/N*n). Furthermore, the signal was plotted using polar plot functionalities for the Fourier transform. This is practical in our case as the input signal is periodic. The polar plot function is special as it creates a plot with polar coordinates through the inputs of angle theta (in rad) and radius rho. The corresponding code is in Figure 13 and the 5 result plots are shown in Figures 14, 15, 16, 17 and 18.

Question 3

To measure the difference between two signals we use mean square error (MSE) as a metric. The MSE for given signals $xx[kk]$ and $yy[kk]$ with length of L is defined as:

$$MSE = \frac{1}{L}\sum_{k=1}^{L}(x[k] - y[k])^2$$

Tasks:

(a) Load two files Original.wav and Distorted.wav to your program and plot them in time domain.

This part involved having to analyze a distorted signal from a system and determining how to remove the distortion to make it as close to the original signal as possible. An audio clip from the song All Around the World by ATC was provided in both its original and distorted form as .wav files. To do this, the 2 files were imported into MATLAB. Their waveforms in time domain were plotted.

Please refer to figure 19 and 20.

(b) Use MSE to compare the original signal and the distorted.

Mean square error (MSE) is commonly used as a benchmark to measure the difference between two given signals. For this question, there are two signals: an original one and a distorted one. The audio files are loaded using the audioread function (replacing wavread as it is outdated) which takes in the sampling frequency of the audio signals. MSE resulted in 0.0250, which can be seen in the command window in Figure 21.

(c) Design a system that recovers the original signal from the distorted signal and save it to a file called Recovered.wav. Which domain did you choose for the design: frequency domain or time domain? Why?

A system was designed in order to recover the original signal from a distorted signal using a low-pass FIR filter with a cutoff frequency of 4 kHz. The filter was passed in the time-domain to remove distortion from the distorted signal. The filter function was utilized to interact the defined coefficients with the distorted signal to be restored. Then, the function audiowrite was utilized to write an audio signal to a file called **Recovered.wav**. Please refer to the MATLAB code of this part in Figure 21.

(d) Compute the MSE between the recovered signal and the original signal. Does your system improve the MSE?

The MSE was computed again comparing the original and the recovered signals in which the result was 0.0600 which differed from the previous result ans the MES has been increased. The higher MSE indicates that there is more distortion or difference between the original signal and the processed signal. Please refer to the MATLAB code of this part in Figure 21.

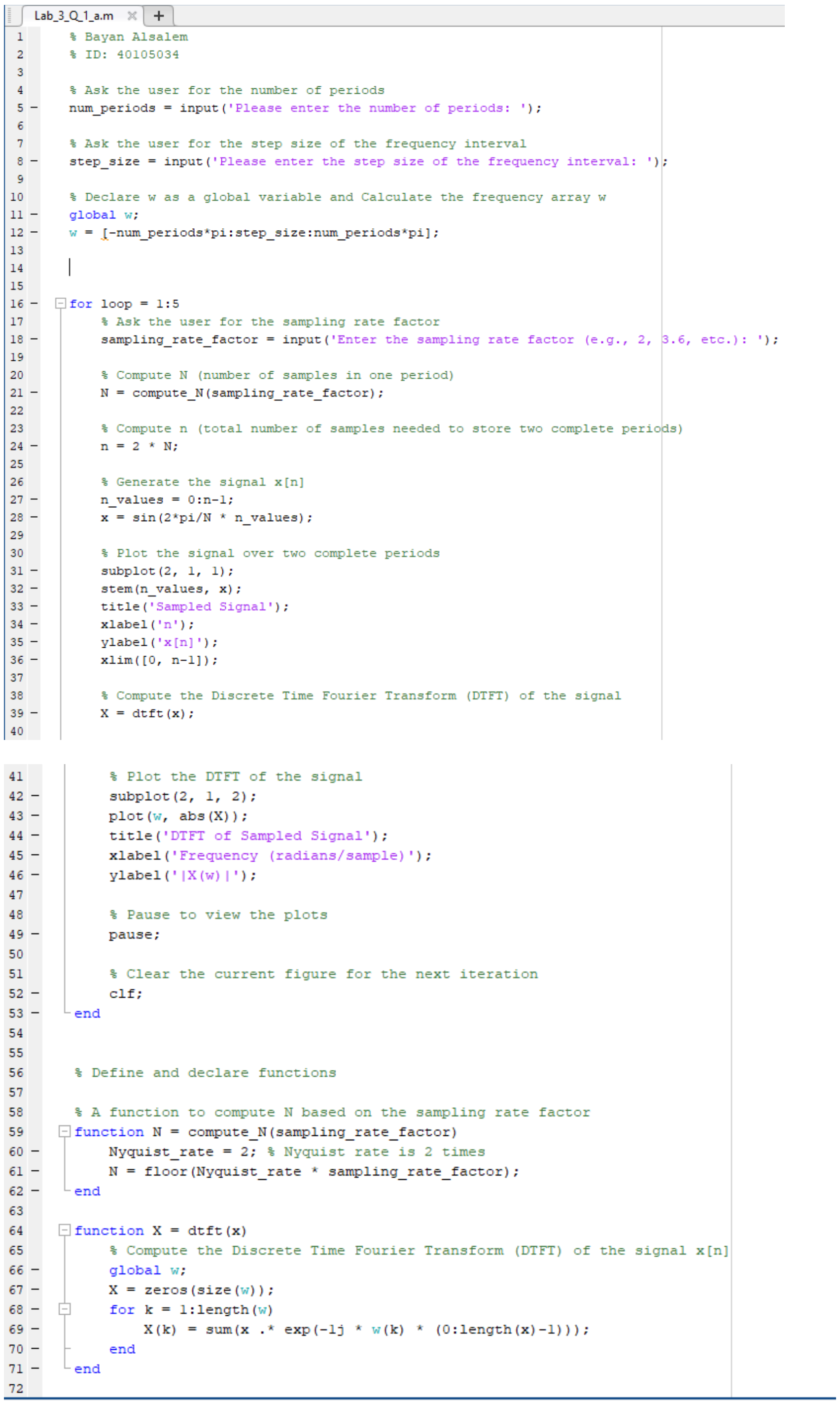
(e) By playing and listening to recovered signal, does your system improve the quality of the sound?

By playing and listening to the recovered signal, the system seems to have removed the distortion pretty well. However, in terms of the quality of sound, it seemed to have decreased the quality of the sound and rather outputted an audio sound appearing to be muffled and less clear as the original.

Conclusion

To conclude, the objective of the experiment was aimed to utilize various built-in MATLAB functions to analyze the discrete properties of a signal, with a particular focus on the sampling rate of the Fourier transform. Additionally, other built-in functionalities such as floor and function were implemented to achieve this goal to further analyze concepts such as window size. Finally, the second part of the lab explored the process of exporting original and distorted signals from WAV files, which was essential to the overall analysis of the signal properties. A recovered signal was created by removing the distortion of the distorted signal and appeared to have resulted in an acceptable output.

Appendix



Lab_3_Q_1_a.m

```matlab
% Bayan Alsalem
% ID: 40105034

% Ask the user for the number of periods
num_periods = input('Please enter the number of periods: ');

% Ask the user for the step size of the frequency interval
step_size = input('Please enter the step size of the frequency interval: ');

% Declare w as a global variable and Calculate the frequency array w
global w;
w = [-num_periods*pi:step_size:num_periods*pi];



for loop = 1:5
    % Ask the user for the sampling rate factor
    sampling_rate_factor = input('Enter the sampling rate factor (e.g., 2, 3.6, etc.): ');

    % Compute N (number of samples in one period)
    N = compute_N(sampling_rate_factor);

    % Compute n (total number of samples needed to store two complete periods)
    n = 2 * N;

    % Generate the signal x[n]
    n_values = 0:n-1;
    x = sin(2*pi/N * n_values);

    % Plot the signal over two complete periods
    subplot(2, 1, 1);
    stem(n_values, x);
    title('Sampled Signal');
    xlabel('n');
    ylabel('x[n]');
    xlim([0, n-1]);

    % Compute the Discrete Time Fourier Transform (DTFT) of the signal
    X = dtft(x);

    % Plot the DTFT of the signal
    subplot(2, 1, 2);
    plot(w, abs(X));
    title('DTFT of Sampled Signal');
    xlabel('Frequency (radians/sample)');
    ylabel('|X(w)|');

    % Pause to view the plots
    pause;

    % Clear the current figure for the next iteration
    clf;
end


% Define and declare functions

% A function to compute N based on the sampling rate factor
function N = compute_N(sampling_rate_factor)
    Nyquist_rate = 2; % Nyquist rate is 2 times
    N = floor(Nyquist_rate * sampling_rate_factor);
end

function X = dtft(x)
    % Compute the Discrete Time Fourier Transform (DTFT) of the signal x[n]
    global w;
    X = zeros(size(w));
    for k = 1:length(w)
        X(k) = sum(x .* exp(-1j * w(k) * (0:length(x)-1)));
    end
end
```

Figure 1: MATLAB code for Question 1 Part a

```
>> Lab_3_Q_1_a
Please enter the number of periods: 2
Please enter the step size of the frequency interval: 0.001
Enter the sampling rate factor (e.g., 2, 3.6, etc.): 2.5
```

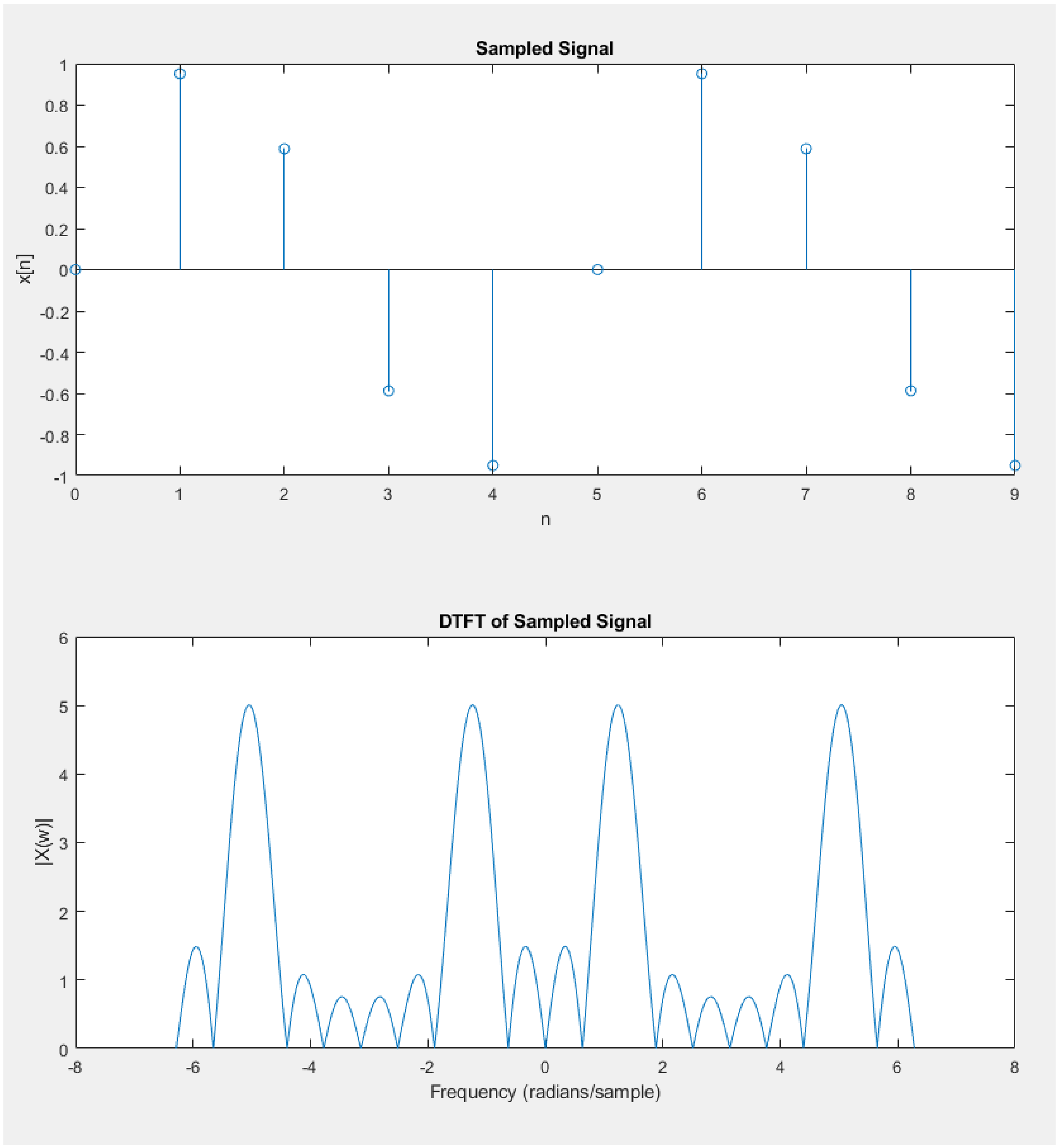


Figure 2: Sampling rate: 2.5

```
Enter the sampling rate factor (e.g., 2, 3.6, etc.): 5
```

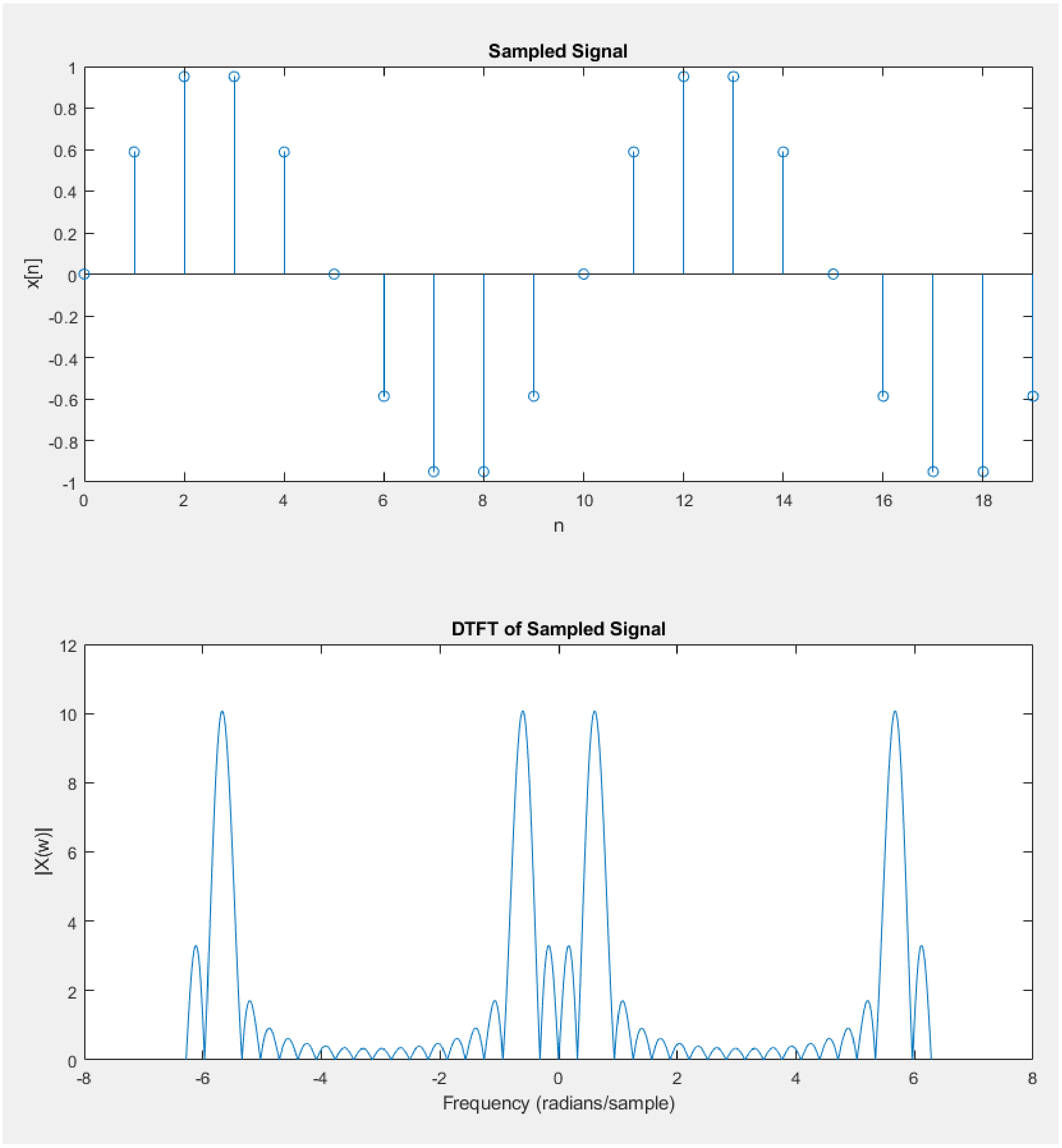


Figure 3: Sampling rate: 5

```
Enter the sampling rate factor (e.g., 2, 3.6, etc.): 10
```

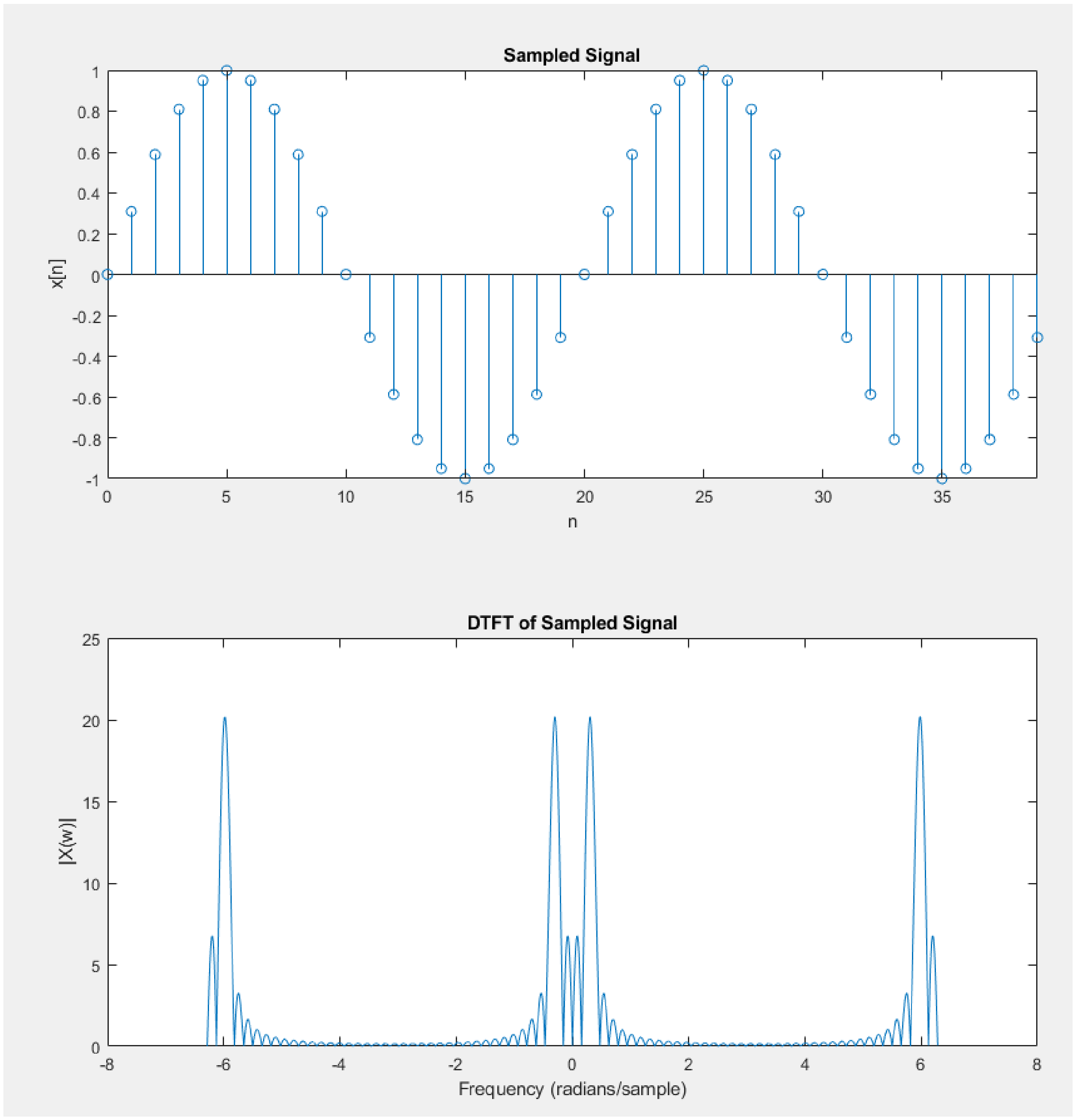


Figure 4: Sampling rate: 10

```
Enter the sampling rate factor (e.g., 2, 3.6, etc.): 15
```

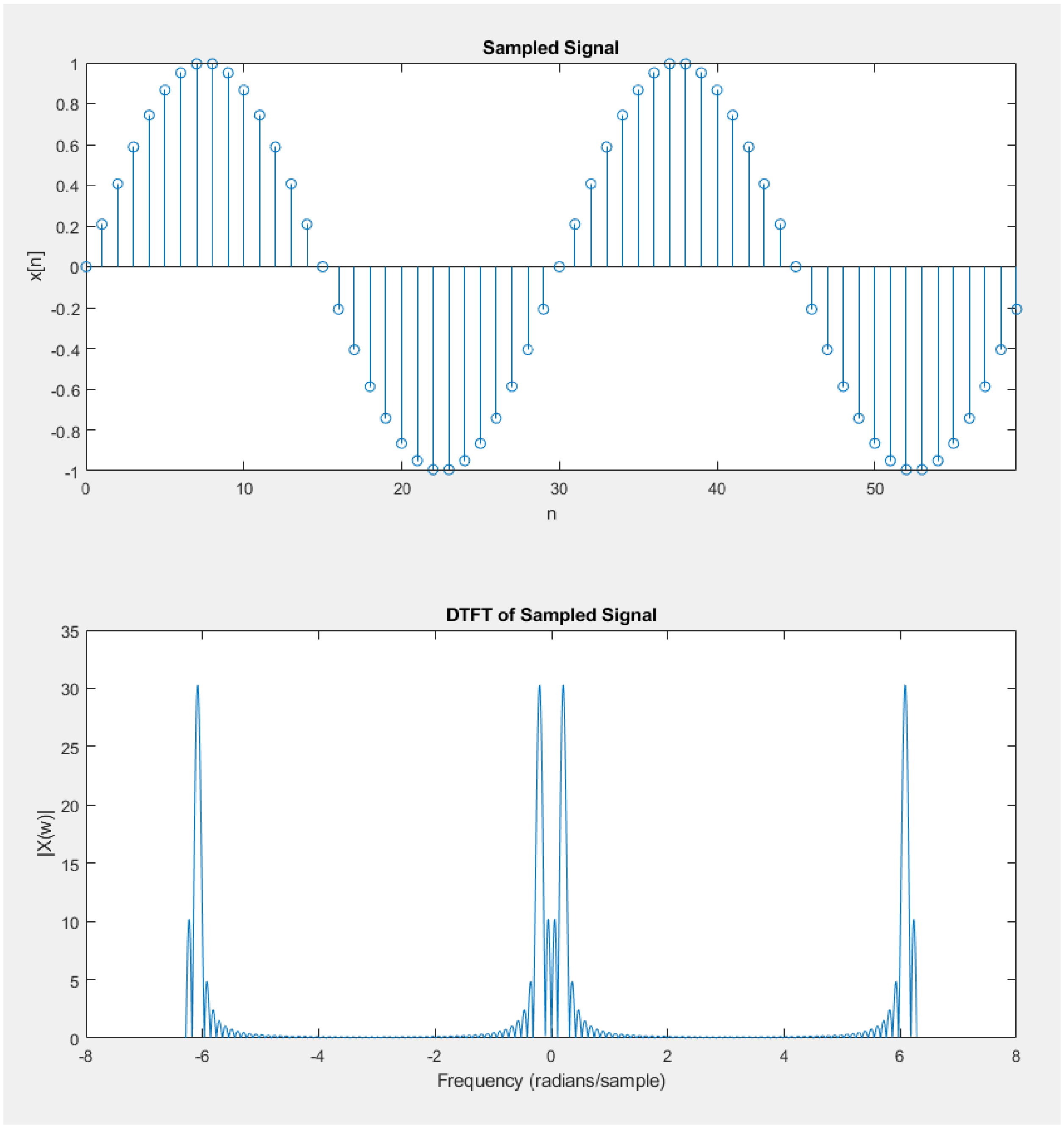


Figure 5: Sampling rate: 15

```
Enter the sampling rate factor (e.g., 2, 3.6, etc.): 40
```

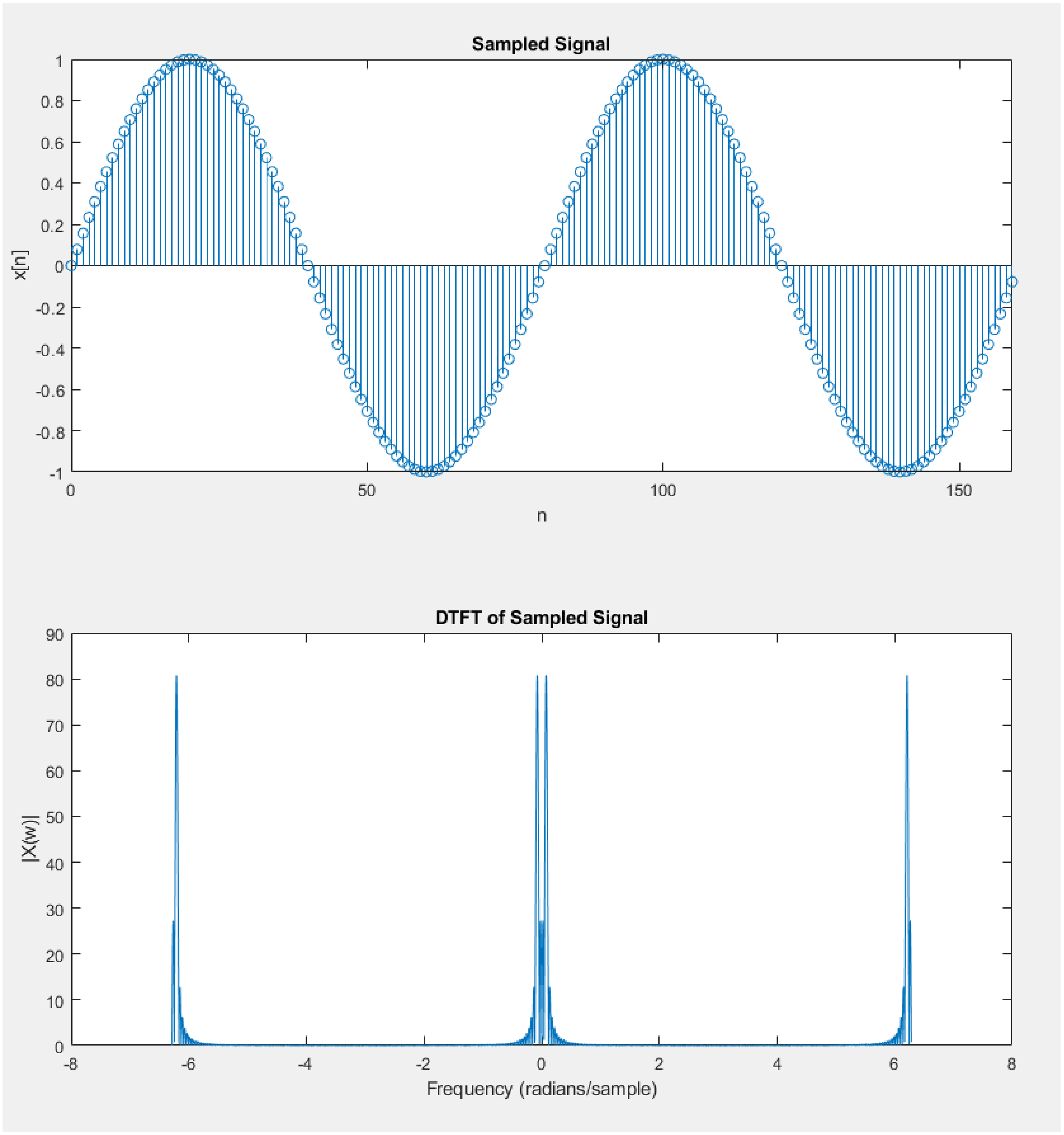
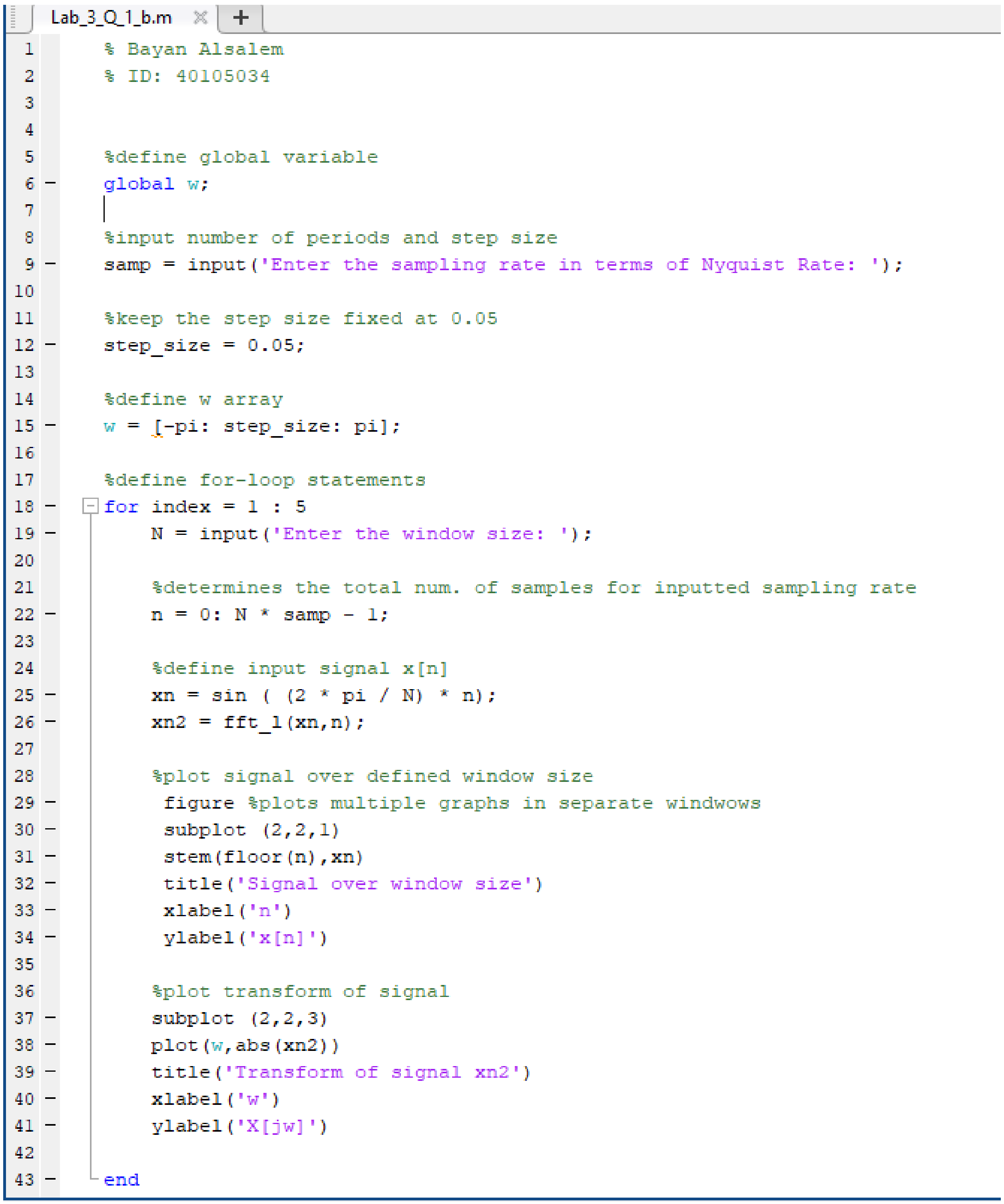
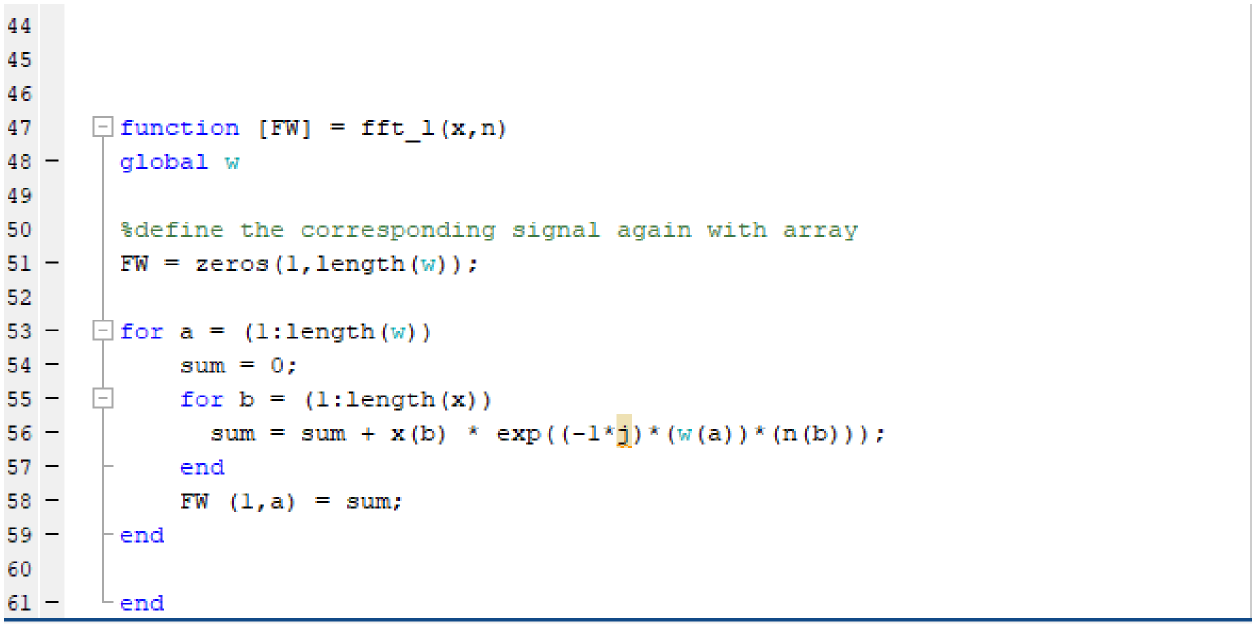


Figure 6: Sampling rate: 40



Lab_3_Q_1_b.m

```
% Bayan Alsalem
% ID: 40105034


%define global variable
global w;

%input number of periods and step size
samp = input('Enter the sampling rate in terms of Nyquist Rate: ');

%keep the step size fixed at 0.05
step_size = 0.05;

%define w array
w = [-pi: step_size: pi];

%define for-loop statements
for index = 1 : 5
    N = input('Enter the window size: ');

    %determines the total num. of samples for inputted sampling rate
    n = 0: N * samp - 1;

    %define input signal x[n]
    xn = sin ( (2 * pi / N) * n);
    xn2 = fft_1(xn,n);

    %plot signal over defined window size
     figure %plots multiple graphs in separate windwows
     subplot (2,2,1)
     stem(floor(n),xn)
     title('Signal over window size')
     xlabel('n')
     ylabel('x[n]')

    %plot transform of signal
    subplot (2,2,3)
    plot(w,abs(xn2))
    title('Transform of signal xn2')
    xlabel('w')
    ylabel('X[jw]')

end
```



```
function [FW] = fft_1(x,n)
global w

%define the corresponding signal again with array
FW = zeros(1,length(w));

for a = (1:length(w))
    sum = 0;
    for b = (1:length(x))
      sum = sum + x(b)  * exp((-1*j)*(w(a))*(n(b)));
    end
    FW (1,a) = sum;
end

end
```

Figure 7 MATLAB code for question 1 part b

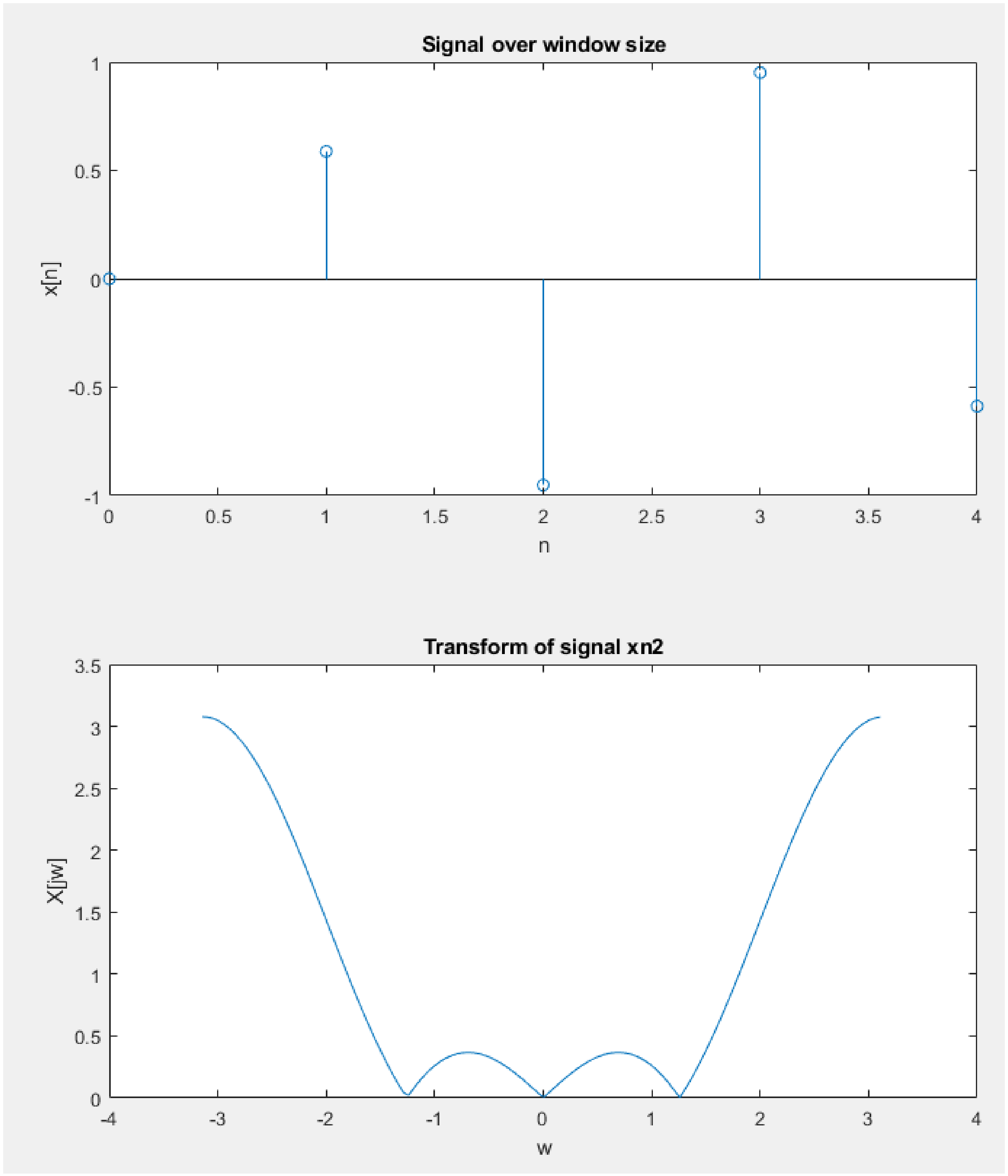


Figure 8 window size: 2.5

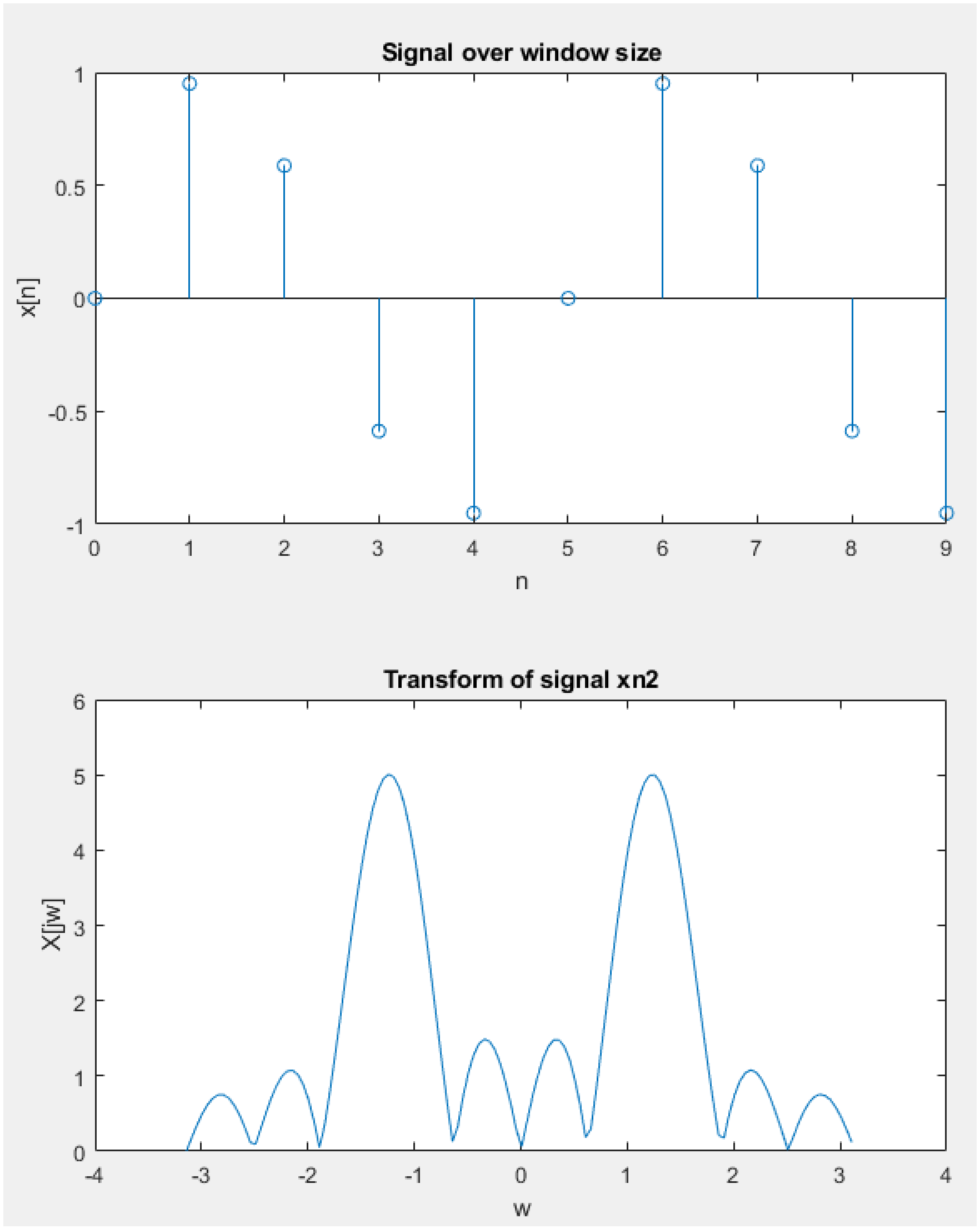


Figure 9 window size: 5

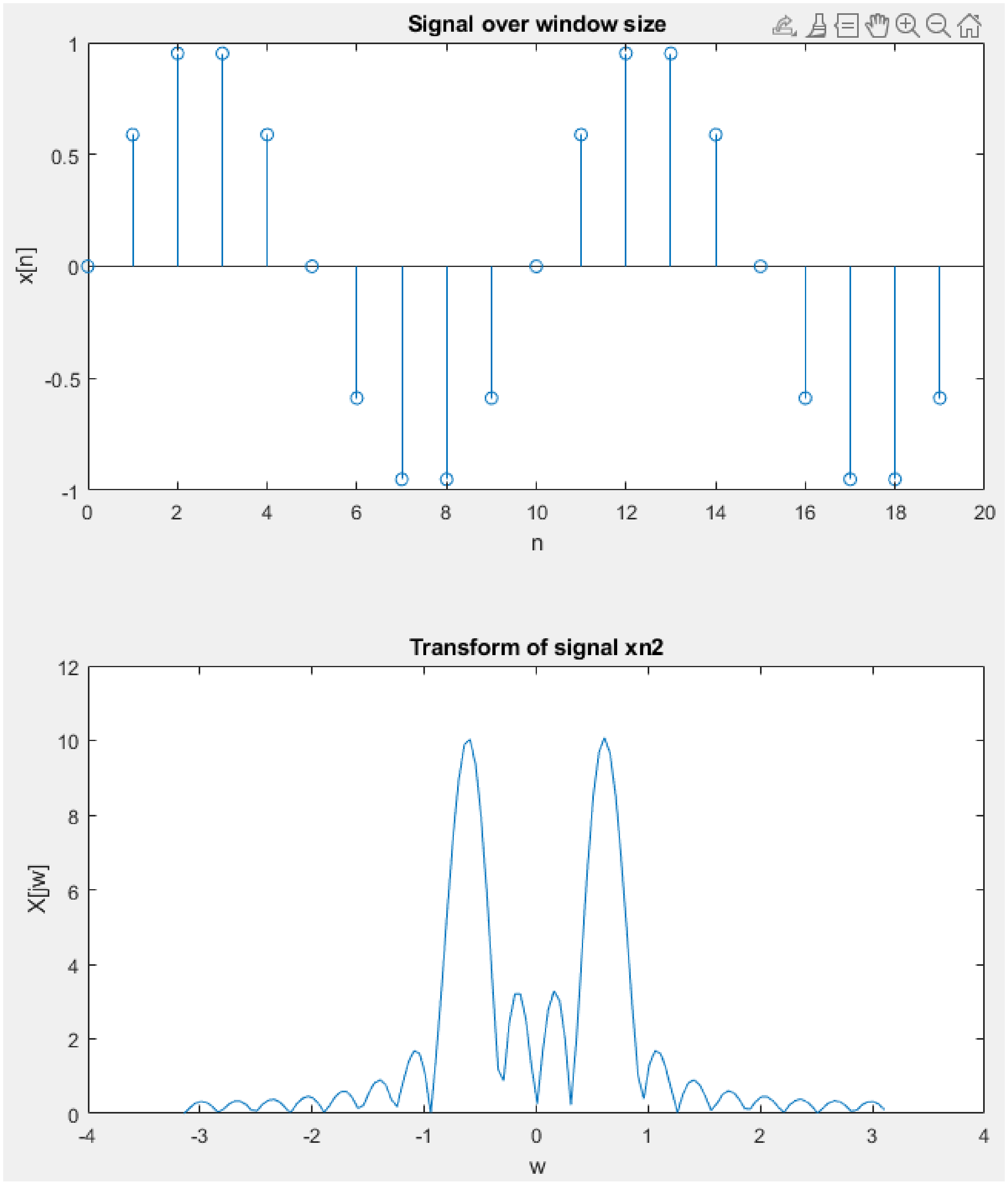


Figure 10 window size:10

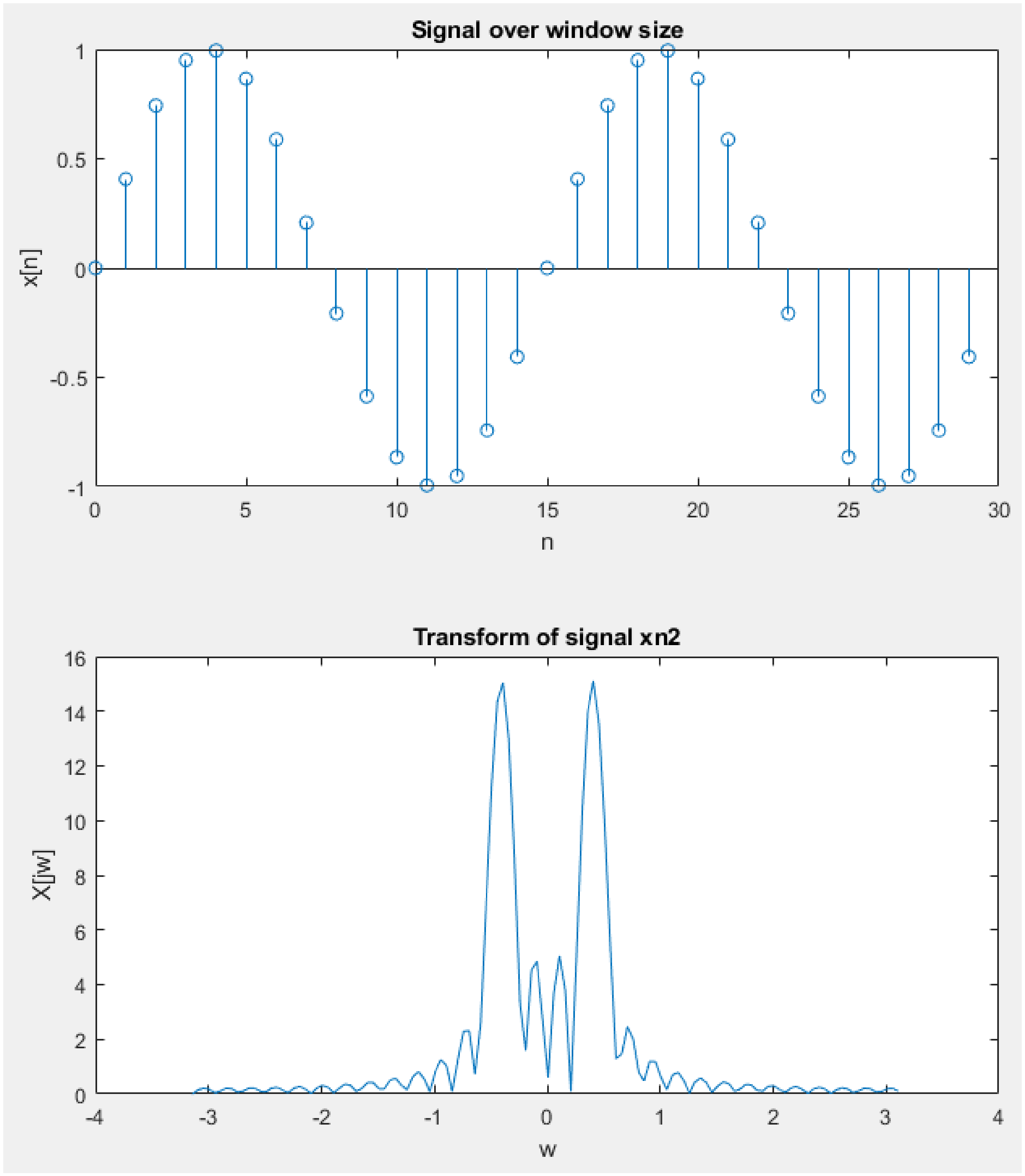


Figure 11 window size: 15

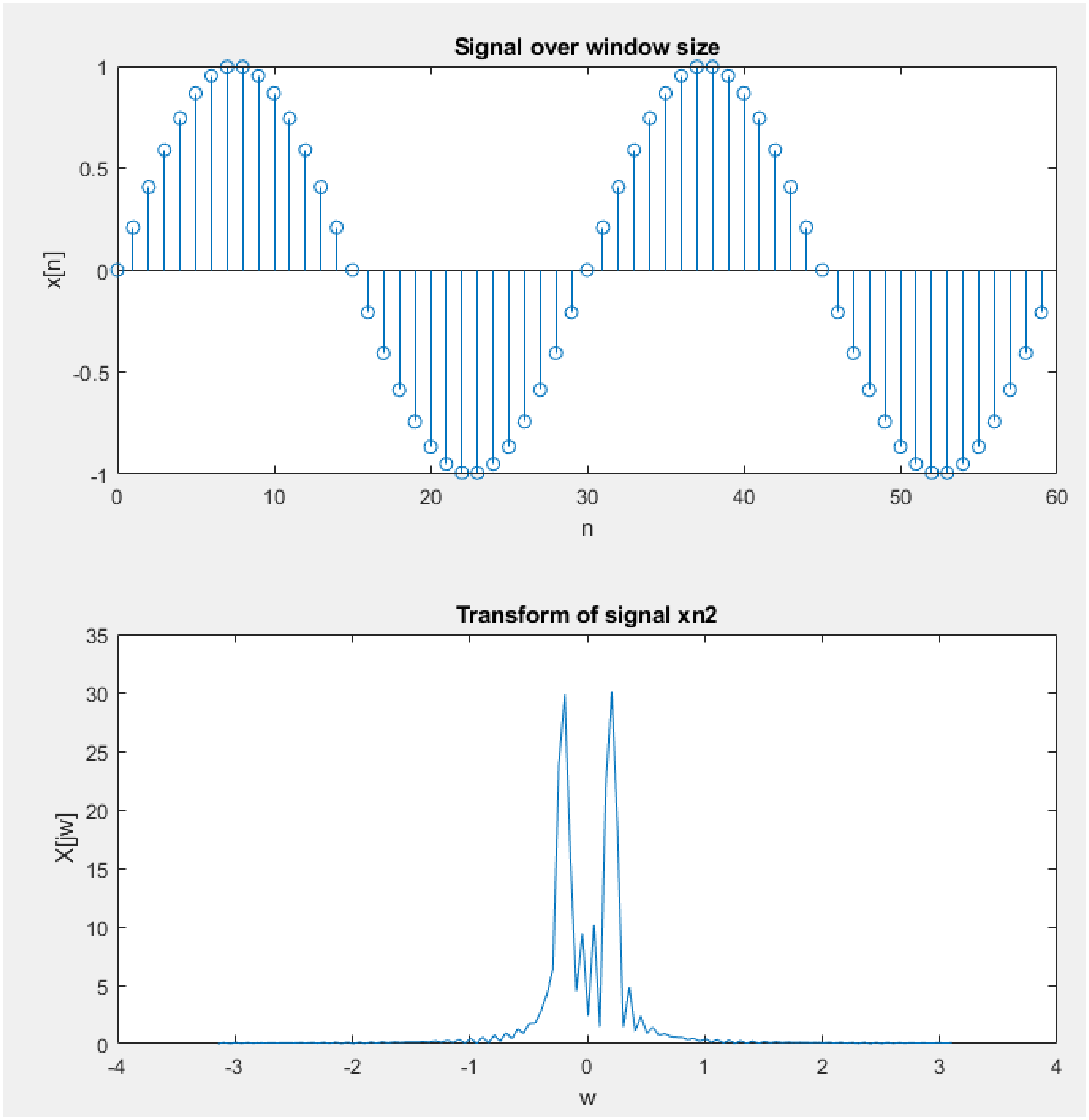
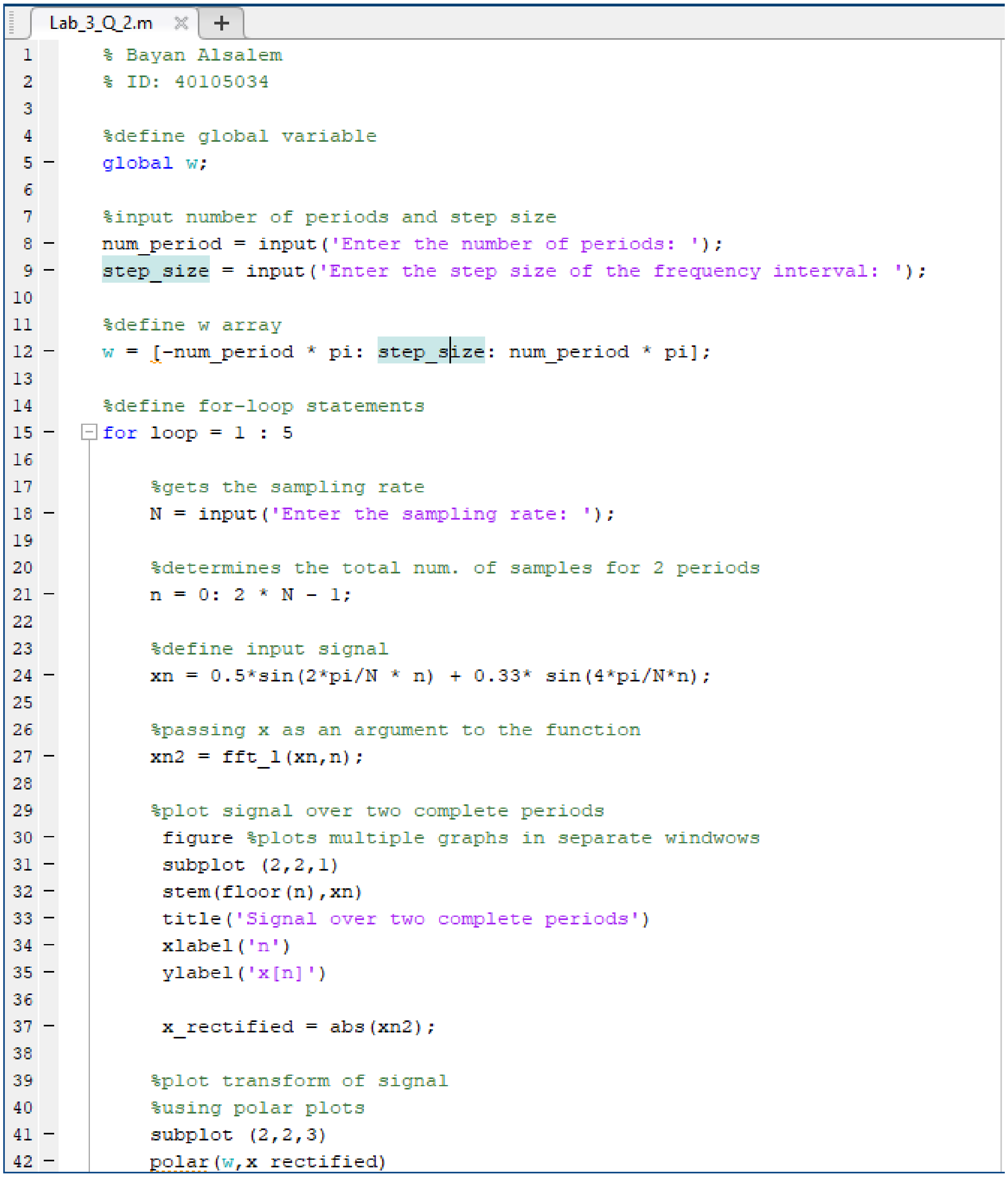
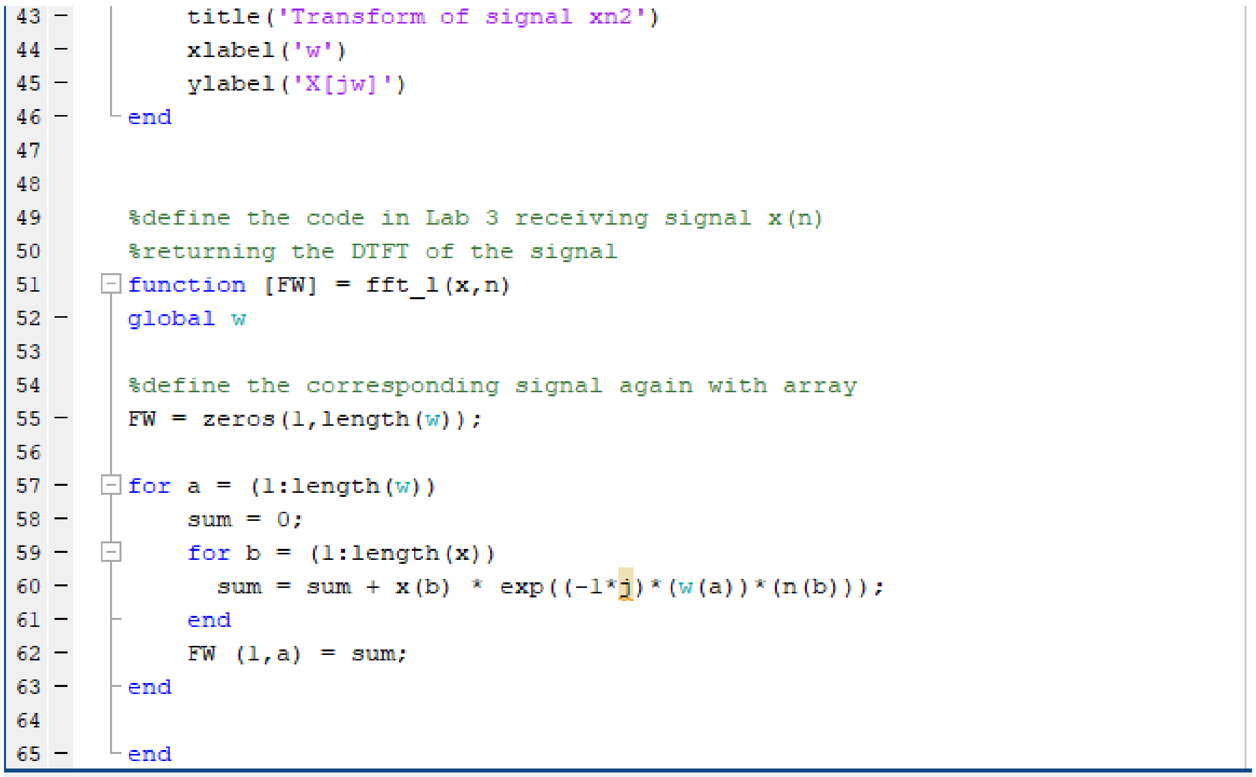


Figure 12 window size: 30

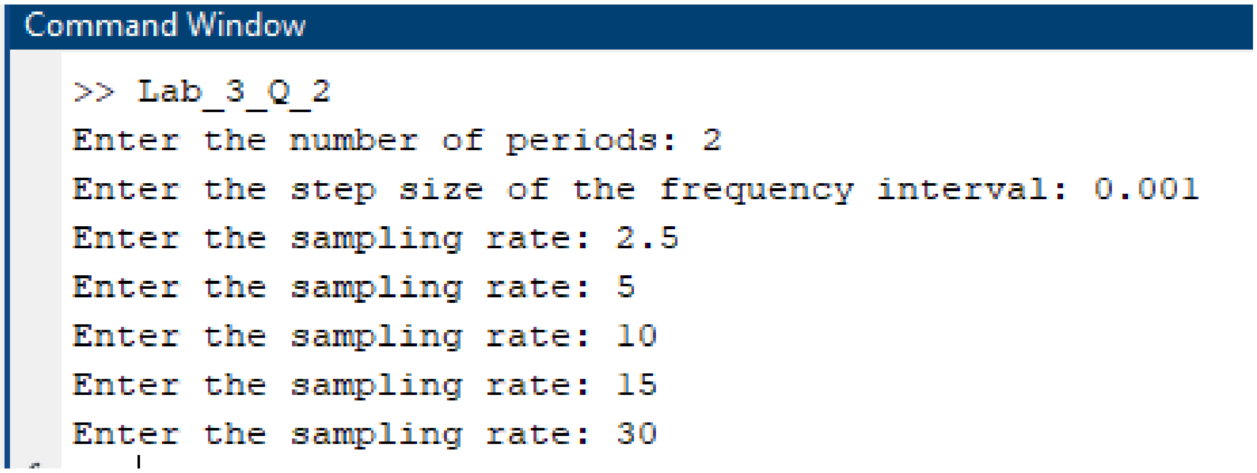


Lab_3_Q_2.m

```
% Bayan Alsalem
% ID: 40105034

%define global variable
global w;

%input number of periods and step size
num_period = input('Enter the number of periods: ');
step_size = input('Enter the step size of the frequency interval: ');

%define w array
w = [-num_period * pi: step_size: num_period * pi];

%define for-loop statements
for loop = 1 : 5

    %gets the sampling rate
    N = input('Enter the sampling rate: ');

    %determines the total num. of samples for 2 periods
    n = 0: 2 * N - 1;

    %define input signal
    xn = 0.5*sin(2*pi/N * n) + 0.33* sin(4*pi/N*n);

    %passing x as an argument to the function
    xn2 = fft_1(xn,n);

    %plot signal over two complete periods
     figure %plots multiple graphs in separate windwows
     subplot (2,2,1)
     stem(floor(n),xn)
     title('Signal over two complete periods')
     xlabel('n')
     ylabel('x[n]')

     x_rectified = abs(xn2);

    %plot transform of signal
    %using polar plots
    subplot (2,2,3)
    polar(w,x_rectified)
```



```
    title('Transform of signal xn2')
    xlabel('w')
    ylabel('X[jw]')
end


%define the code in Lab 3 receiving signal x(n)
%returning the DTFT of the signal
function [FW] = fft_1(x,n)
global w

%define the corresponding signal again with array
FW = zeros(1,length(w));

for a = (1:length(w))
    sum = 0;
    for b = (1:length(x))
      sum = sum + x(b)  * exp((-1*j)*(w(a))*(n(b)));
    end
    FW (1,a) = sum;
end

end
```



```
>> Lab_3_Q_2
Enter the number of periods: 2
Enter the step size of the frequency interval: 0.001
Enter the sampling rate: 2.5
Enter the sampling rate: 5
Enter the sampling rate: 10
Enter the sampling rate: 15
Enter the sampling rate: 30
```

Figure 13 MATLAB code and command window for Question 2

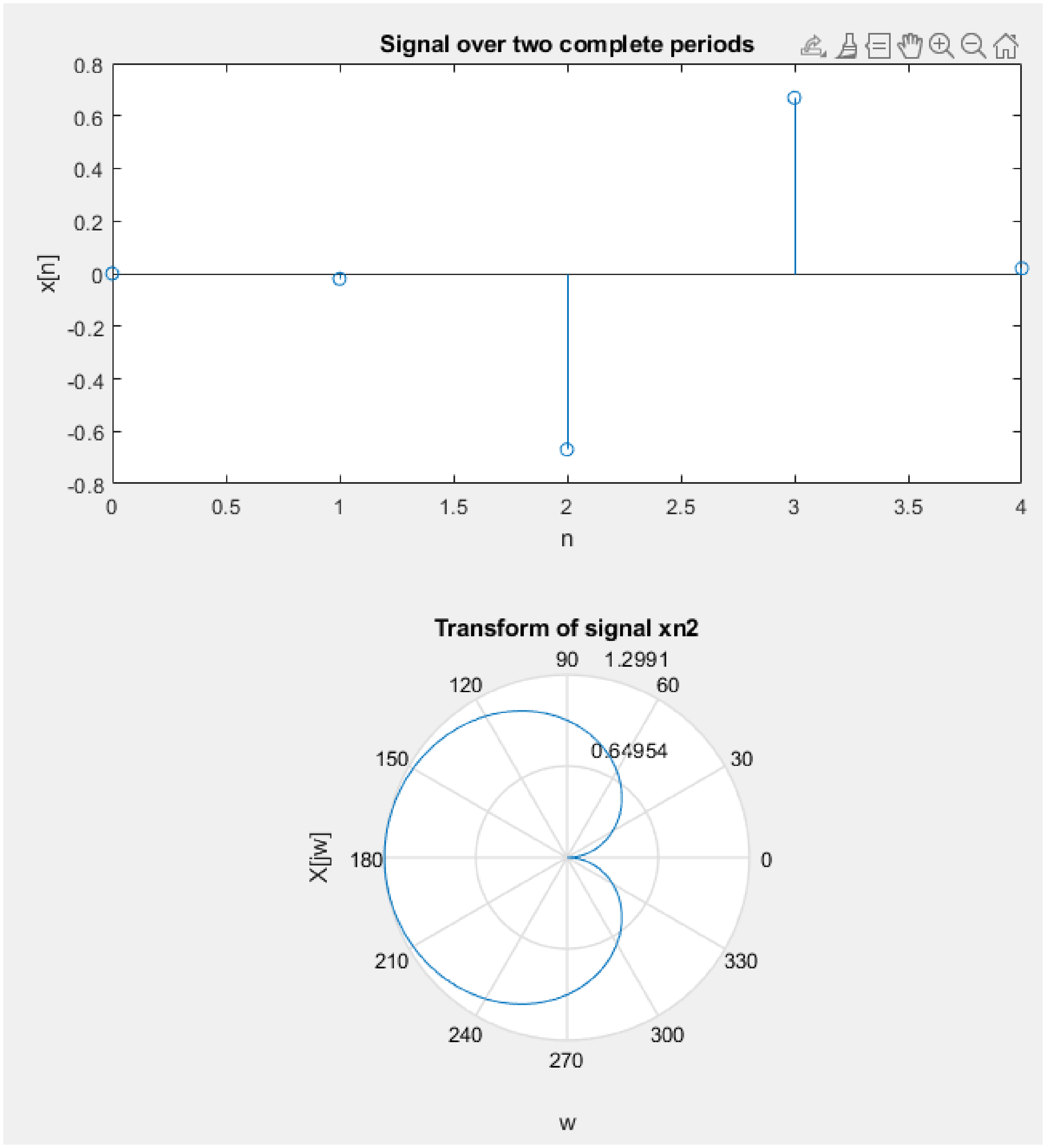

Signal over two complete periods
x[n]
n
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
0
0.5
1
1.5
2
2.5
3
3.5
4
Transform of signal xn2
X[jw]
w
90
1.2991
60
120
0.64954
30
150
180
0
210
330
240
300
270

Figure 14

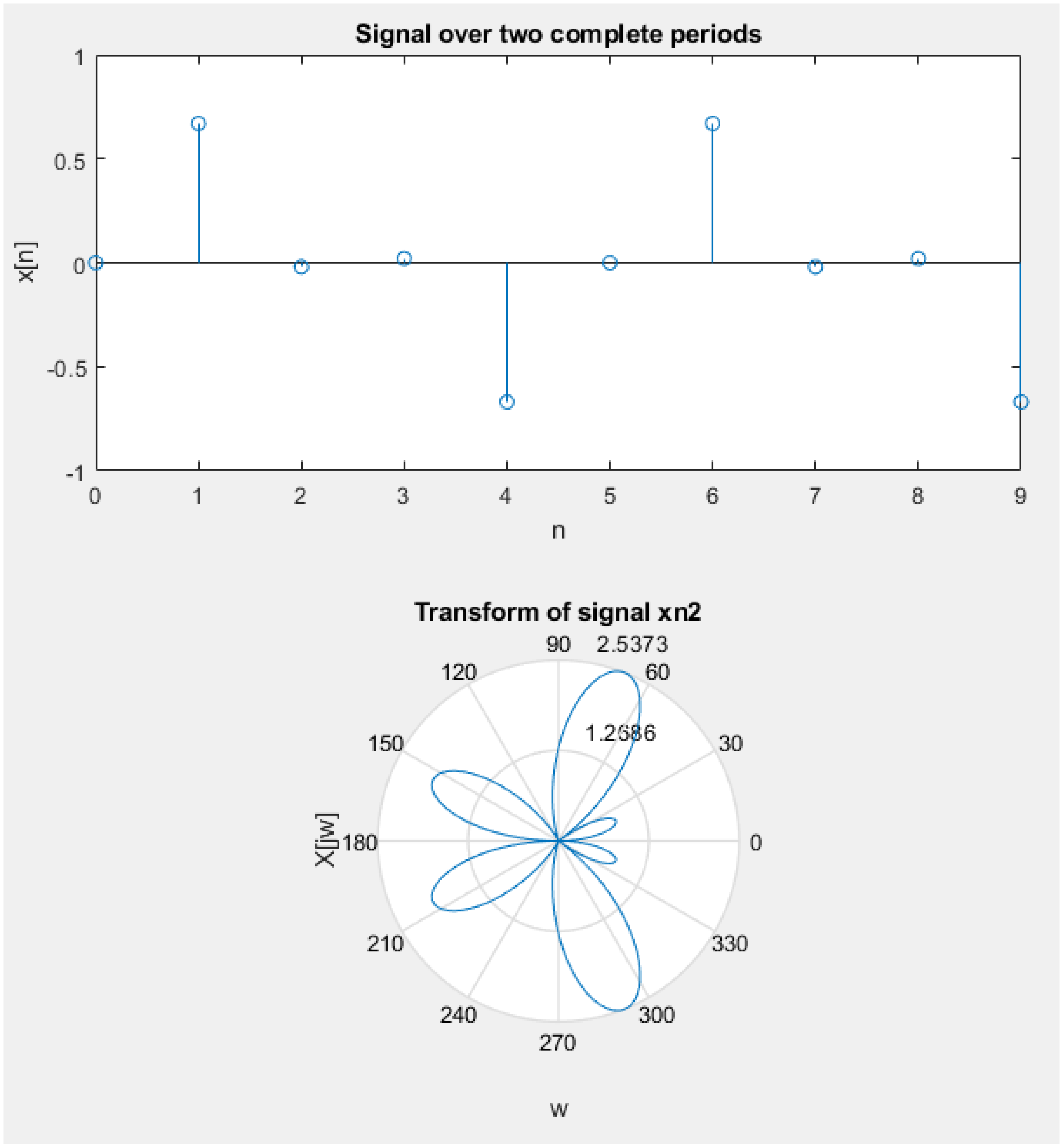
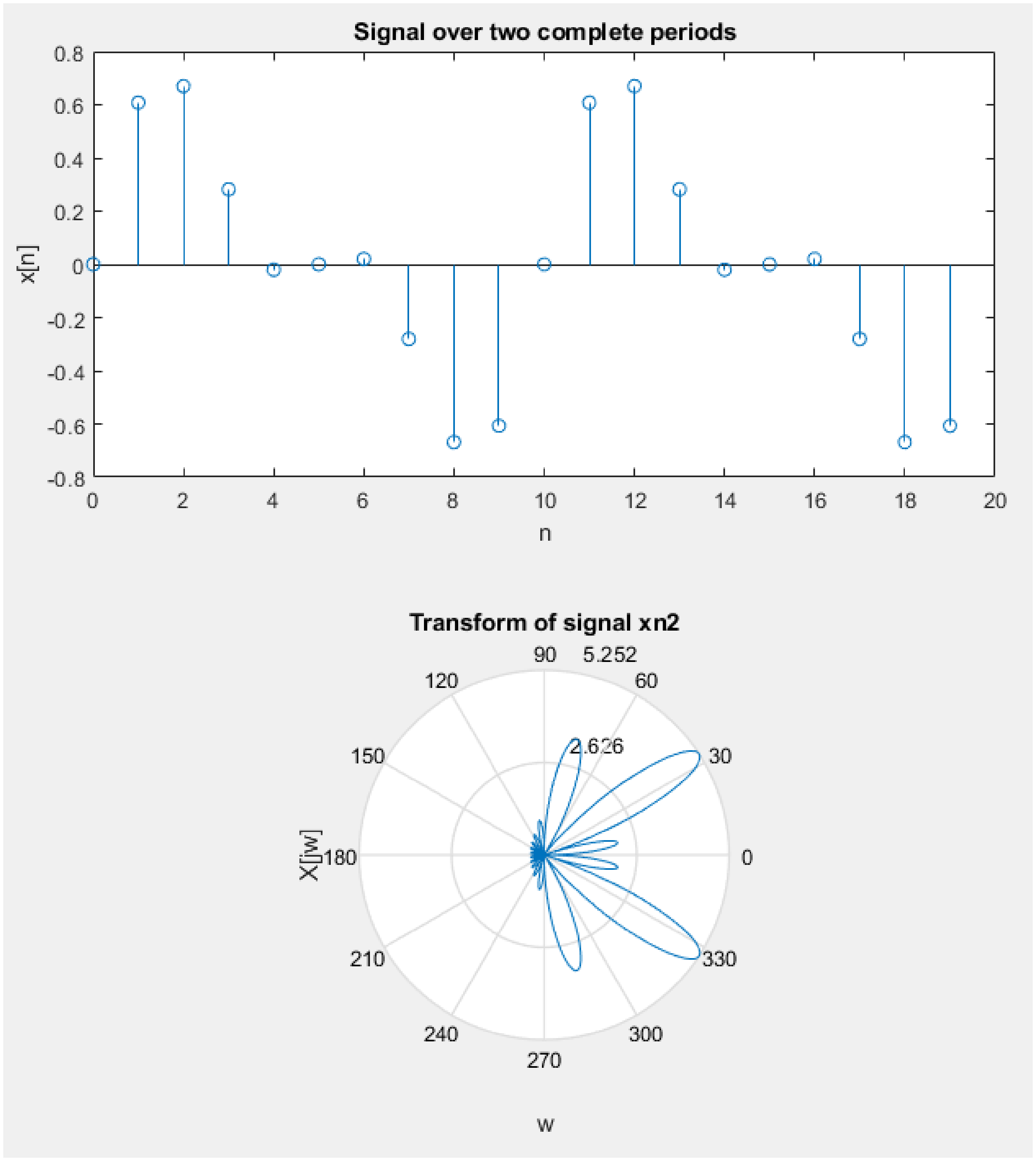
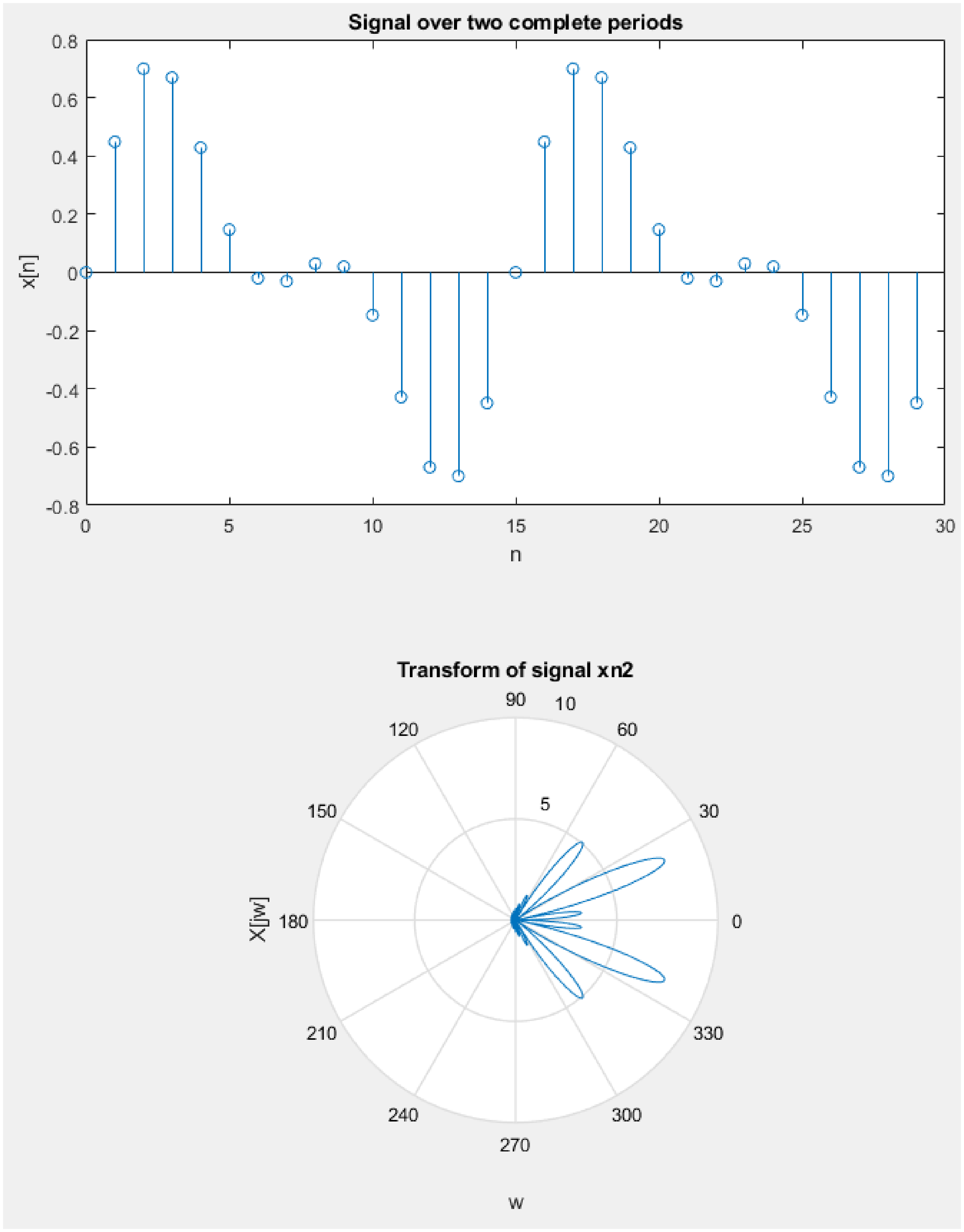

Signal over two complete periods
x[n]
n
1
0.5
0
-0.5
-1
0
1
2
3
4
5
6
7
8
9
Transform of signal xn2
90
2.5373
60
120
1.2686
30
150
180
0
210
330
240
300
270
X[jw]
w

Figure 15


Signal over two complete periods
x[n]
n
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
0
2
4
6
8
10
12
14
16
18
20
Transform of signal xn2
90
5.252
60
120
2.626
150
30
X[jw]
180
0
210
330
240
300
270
w
Figure 16



Signal over two complete periods
x[n]
n
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
0
5
10
15
20
25
30
Transform of signal xn2
X[jw]
w
90
10
120
60
150
5
30
180
0
210
330
240
300
270

Figure 17

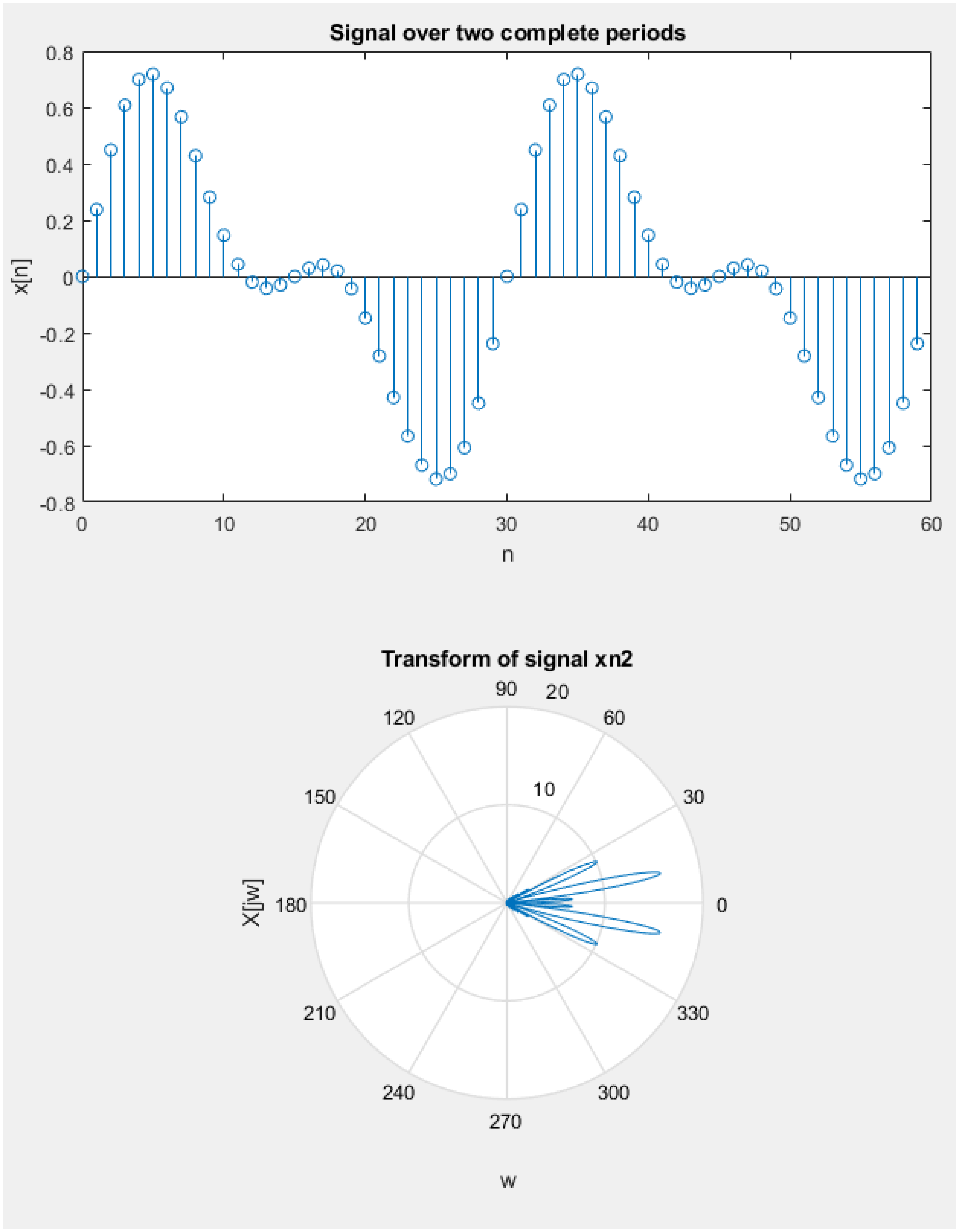
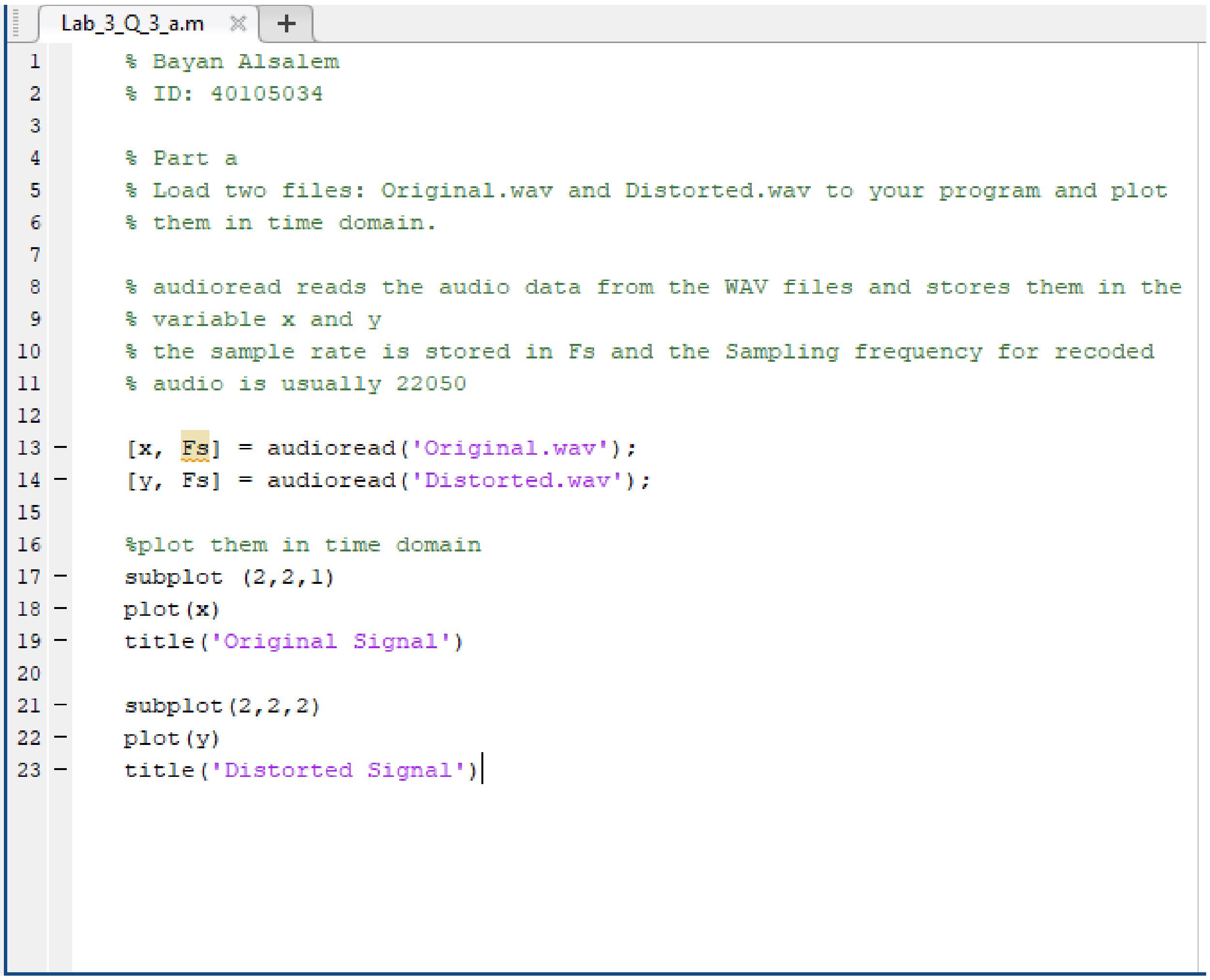

Signal over two complete periods
x[n]
n
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
0
10
20
30
40
50
60
Transform of signal xn2
X[jw]
w
90
120
60
150
30
180
0
210
330
240
300
270
10
20

Figure 18



```
% Bayan Alsalem
% ID: 40105034

% Part a
% Load two files: Original.wav and Distorted.wav to your program and plot
% them in time domain.

% audioread reads the audio data from the WAV files and stores them in the
% variable x and y
% the sample rate is stored in Fs and the Sampling frequency for recoded
% audio is usually 22050

[x, Fs] = audioread('Original.wav');
[y, Fs] = audioread('Distorted.wav');

%plot them in time domain
subplot (2,2,1)
plot(x)
title('Original Signal')

subplot(2,2,2)
plot(y)
title('Distorted Signal')
```

Figure 19 MATLAB code for Question 3 part a

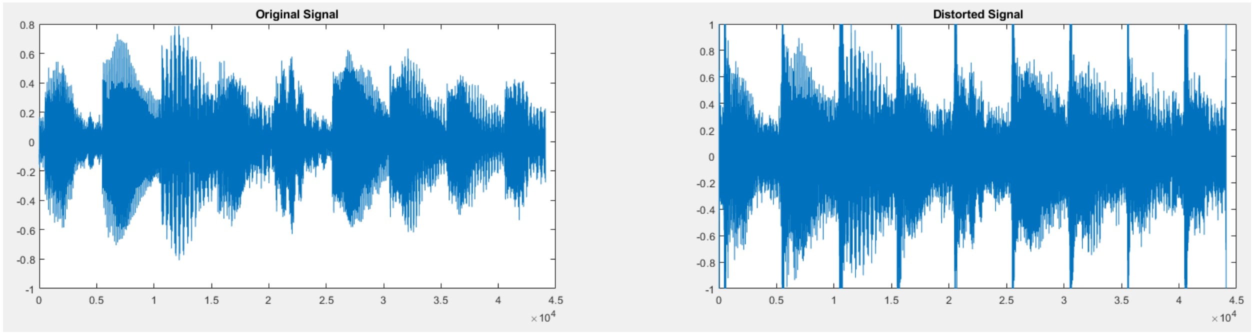
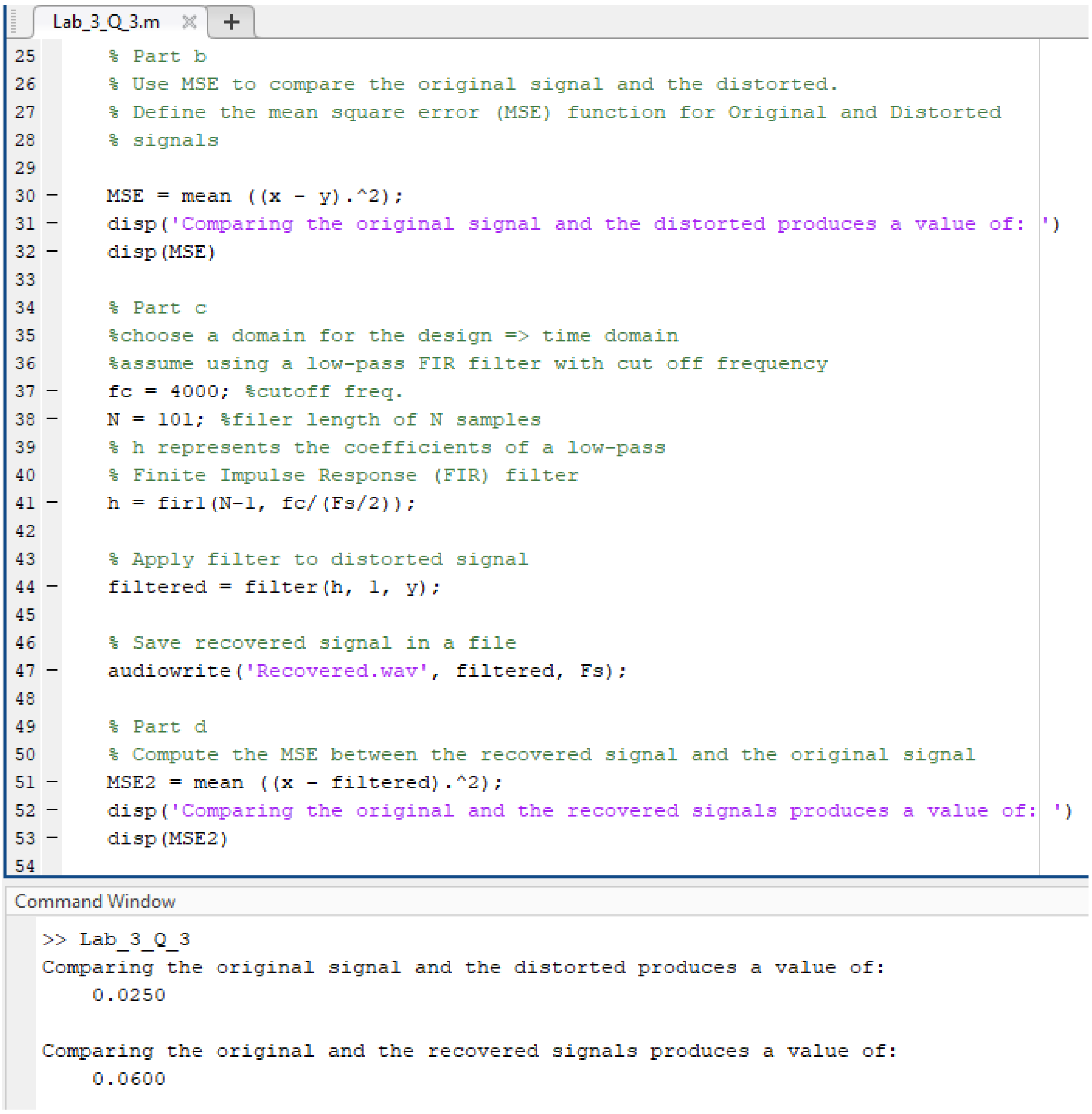


Figure 20 Plots of Question 3 part a



```
% Part b
% Use MSE to compare the original signal and the distorted.
% Define the mean square error (MSE) function for Original and Distorted
% signals

MSE = mean ((x - y).^2);
disp('Comparing the original signal and the distorted produces a value of: ')
disp(MSE)

% Part c
%choose a domain for the design => time domain
%assume using a low-pass FIR filter with cut off frequency
fc = 4000; %cutoff freq.
N = 101; %filer length of N samples
% h represents the coefficients of a low-pass
% Finite Impulse Response (FIR) filter
h = fir1(N-1, fc/(Fs/2));

% Apply filter to distorted signal
filtered = filter(h, 1, y);

% Save recovered signal in a file
audiowrite('Recovered.wav', filtered, Fs);

% Part d
% Compute the MSE between the recovered signal and the original signal
MSE2 = mean ((x - filtered).^2);
disp('Comparing the original and the recovered signals produces a value of: ')
disp(MSE2)
```

```
>> Lab_3_Q_3
Comparing the original signal and the distorted produces a value of:
    0.0250

Comparing the original and the recovered signals produces a value of:
    0.0600
```

Figure 21 Matlab code and command window for Question 3 part b, c, and d