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.:	1	**Date Performed:**	2023 – 09 –19 (YYYY – MM – DD)

Experiment Title: Additional MATLAB features, Properties of Signals and Systems, Convolution and System Response

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 – 09 –30
YYYY – MM – DD

Table of Contents

Objectives

In Part I of this lab, students will explore additional features of the MATLAB programming language, including looping, conditional selection, and array processing. They will also apply these concepts to verify properties of signals and systems, such as linearity, evenness, and oddness, using straightforward MATLAB scripts. In Part II, students will utilize the MATLAB convolution function to analyze and determine a system's response to a given input signal.

Theory

MATLAB, short for "MATrix LABoratory," is a high-level programming language and interactive environment primarily used for numerical and scientific computing. It was developed by MathWorks and is widely used in academia, industry, and research for different applications.

One of MATLAB's core strengths lies in its proficiency in numerical and scientific computing tasks. It excels in a wide array of applications, including linear algebra, calculus, signal processing, image processing, data analysis, and simulation. The platform boasts an extensive collection of built-in functions and toolboxes that facilitate these tasks.

The platform excels in data visualization, providing robust tools for creating 2D and 3D plots, displaying images, and even generating animations. Researchers and professionals can produce publication-quality graphics to communicate their findings effectively.[2]

Tasks, Results, Discussion and Questions

Part 1

Question 1

(a) Plot the input signal $x[n]$=n and the output signal $y[n]$=x^2[n] over the interval n = 0, 1, 2, 3, 4, 5, 6, 7, 8, 9. Compute <u>the total energy</u> in the signal $x[n]$ and $y[n]$ (Hint: the total energy in a signal is equal to the sum of the squares of all the values contained in the signal). Use the <u>disp</u> command to display the two energies.

To solve this question, you would first plot the input signal $x[n] = n$ and the output signal $y[n]$ = x^2[n] for the given range of n values (0 to 9). Then, to compute the total energy in each signal, square each value in the signal, sum up all the squared values, and display these energies using the disp command. The total energy in a signal is found by summing the squares of its values, as indicated in the hint.

The MATLAB code and the result plots are shown in figure 1 and figure 2.

(b) Repeat part (a) using the input signal $x[n]=sin\ ((2\pi/10\ )n)$, n = 0, 1, 2, …, 9. Use the MATLAB function sin to compute the values of the input signal over the specified interval. Use the help sin facility to learn how to use the sin function.

To solve part (b) of the question, you would use the given input signal $x[n] = \sin((2\pi/10)\ n)$ for the range of n values (0 to 9). In MATLAB, you can compute the values of this signal using the sin function by providing the appropriate argument inside the sin function, which involves $(2\pi/10)\ n$.

The MATLAB code and the result plots are in figure 3 and figure 4.

Question 2

(a) Determine whether the DT system which has an output $y[n]=2x[n]$ over the interval $0\leq n\leq 10$ is <u>linear or not</u> by determining the response $y1[n]$ to the input signal $x1[n]=sin\ ((2\pi\ 10\ )n)$ and the response $y2[n]$ to the input signal $x2[n]=cos((2\pi\ 10\ )n)$. Determine the response $y3[n]$ to the input signal $x3[n]=x1[n]+x2[n]$ and compare it with $y4[n]=y1[n]+y2[n]$.

Plot (using stem) in one graph all the input signals and their corresponding output signals. Use the disp command to output whether the system has 'outputs consistent with a linear system' or 'not linear'.

To determine whether the given DT system with output $y[n] = 2x[n]$ over the interval $0 \leq n \leq 10$ is linear, we can follow these steps:

1- Compute $y1[n]$ for the input signal $x1[n] = sin((2\pi/10)\ n)$ using the given system equation $y[n] = 2x[n]$.

2- Compute $y2[n]$ for the input signal $x2[n] = cos((2\pi/10)\ n)$ using the same system equation.

3- Calculate $x3[n] = x1[n] + x2[n]$ to get the combined input signal.
4- Compute $y3[n]$ for the combined input signal $x3[n]$ using the system equation $y[n] = 2x[n]$.
5- Calculate $y4[n] = y1[n] + y2[n]$ to get the sum of the responses from step 1 and 2.
6- Use the stem command to plot all the input signals ($x1[n]$, $x2[n]$, $x3[n]$) and their corresponding output signals ($y1[n]$, $y2[n]$, $y3[n]$) in one graph.
7- Finally, use the disp command to output whether the system has 'outputs consistent with a linear system' if $y3[n]$ equals $y4[n]$, or 'not linear' if they are not equal.

The MATLAB code and the result plots are in figure 5 and figure 6.

Part a: The system exhibits outputs that are consistent with those of a linear system. This suggests that for the given set of input-output data, the system behaves in a manner that aligns with the principles of linearity.

(b) Design an experiment to test whether the systems are <u>linear and time-invariant</u>. Use the input data x[n] = [0,1]. Next, using a larger set of values, for your choice of the input data, repeat the experiment and analyze and interpret the results obtained with this new data set. Do the new results validate or invalidate the original results obtained with x[n] = [0,1] as choice of input data? Explain how a choice of data used may impact the results obtained.

(i) $y[n]=x\wedge 2[n]$

Refer to Figure 7 for the MATLAB code.

Part b-i: However, upon further investigation, it is determined that the system is not truly linear. Despite initially appearing to behave linearly in part a.

Part b-i with larger values set: When the system is subjected to a larger set of values, it becomes evident that it is not linear for this expanded range of inputs. This non-linearity may manifest as output responses that do not follow a linear relationship with the inputs.

Part b-i with larger values set: Additionally, it is observed that the system remains time-invariant for the larger set of values. This means that its behavior doesn't change with time, even though it is not linear for these inputs.

(ii) $y[n]=2x[n]+5\delta[n]$

Refer to Figure 8 for the MATLAB code.

Part b-ii: On the other hand, for a different set of conditions, the system is found to be linear. This suggests that under specific circumstances or within certain input ranges, the system does adhere to the principles of linearity.

Part b-ii with larger values set: When subjected to a larger set of values, the system departs from linearity, similar to the findings in part b-i for the larger set of values. The non-linearity in this case becomes more pronounced as the input values increase.

Part b-ii with larger values set: Despite its time-variant behavior under the specific conditions mentioned in part b-ii, the system is found to be time-invariant for the larger set of values. This means that, for the expanded range of inputs, the system's response remains consistent over time, even though it is not linear.

Question 3

(a) Plot the following $x[n]$, its mirror image $x[-n]$, and it's even and odd components:

$x[n]=e$^$(-2|n|)$ * $sin((2\pi/36)\ n)$, $0 \leq n \leq 10$, Use the functions exp and abs.

The MATLAB code and the result plots are in figure 9 and 10.

(b) Repeat part (a) for the signal:

$x[n]= (-1)$ ^$(n-1)$, $-5\leq n\leq 5$

The MATLAB code and the result plots are in figure 11 and 12.

(c) Compare and contrast the two methods used to generate the two MATLAB arrays x1 and x2 in the following MATLAB code:

```
% T. Obuchowicz
%Fri Apr 27 16:03:27 EDT 2012

clear
n = [1 : 20 ]

x1 = sin((2*pi/40) * n) .* cos((2*pi/40) * n)

for index = 1 : 20

% Note: In MATLAB, no need to pre-allocate the array,
% unlike C++ and other high-level programming languages.

  x2(index) = sin((2*pi/40) * index) * cos((2*pi/40) * index)

end

subplot(2,1,1)
stem(n, x1)
title('Elegant method making full use of MATLABs array capabilities')
xlabel('n')
ylabel('x[n]')

subplot(2,1,2)
stem(n, x2)
title('Gets the job done, but it is a lot of work and we are not in the MATLAB
mindset')
xlabel('n')
ylabel('x[n]')
```

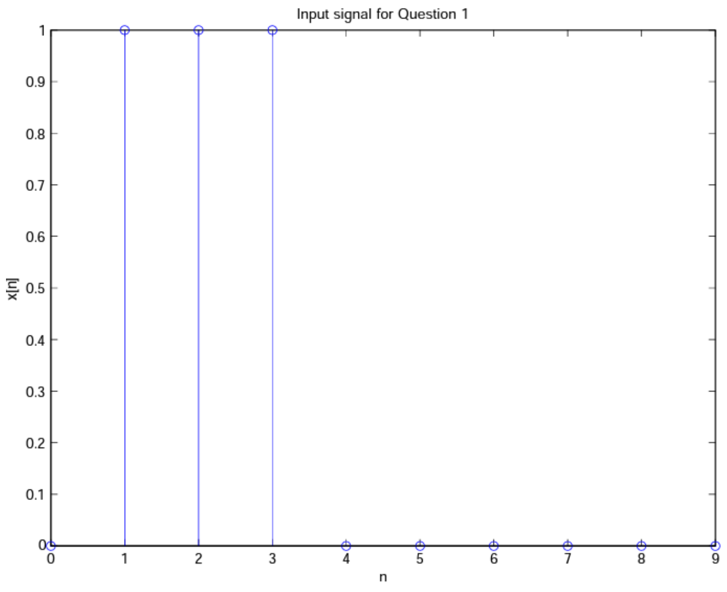
Method 1 (using array operations) is vectorized, meaning it performs operations on entire arrays at once. MATALB allows for every element in the array to be processed by the signal and then is saved inside the variable where the signal is stored. The variable where the signal is stored is a matrix that is has the same dimensions as the variable which holds the values of the independent variable n. Method 2 (using a loop) performs operations sequentially for each element. Method 1 is generally more efficient and faster in MATLAB because it leverages MATLAB's optimized array operations. Method 2 can be slower for larger datasets due to the overhead of the loop. Also, FOR loops need to be iterated for each value in the matrix, which costs computational time which is not needed.

The MATLAB code and the result plots are in figure 13 and 14.

Part 2

Question 1

For this system, compute the response $y[n]=x[n]+1\ 4\ y[n-1]$ over the interval $0\leq n\leq 9$ using the input signal $x[n]$ given in the following Figure [1]. Plot the response using the stem function.



The MATLAB code and the result plots are in figure 15 and 16.

Question 2

Compute the system response (using the input signal $x[n]$ given in Question (1)) by convolving $H[n]$ and $x[n]$ using the MATLAB conv function. Plot the response using stem. Compare this response with that obtained in Question 1. Explain any differences between the two outputs.

For this question, the MATLAB built-in function of convolution “conv” is utilized to calculate the convolution of two vectors. The two vectors are x[n] and h[n]. The input signal x[n] is defined as (x = (1:10); x(1,2,3) = 1). Thus, there is an implementation that was done, and the plot is shown in Figure 17 and 18. In comparison to the resulting response in Question 1, both responses are completely identical. Therefore, there are no differences between the two outputs when comparing plot in figure 16 and plot in figure 18.

Question 3

Sometimes a system that we are interested in is a black box, i.e., we can see what goes into the system and what comes out of the system, but we cannot see what is inside the system. In such a case we can try to find out some things about the system by choosing particular inputs to put in and then observing what comes out. We could also use this same process of putting inputs in and observing the output to determine the system entirely. When we do this, it is called system identification. Here you are given a system called Sys1. Note, Sys1.p is Matlab P-code file, available at this link.

y = Sys1(x)

Inputs to the function is the input vector x. The vector y is a vector which is the corresponding output signal. Note that y may not be the same length as x.

(a) Is the system Linear?

To determine whether Sys1 is a linear system or not, I will design an experiment to test its linearity hypothesis. I would create a script to test linearity by checking for the additive property attempting to prove:

x1 + x2 => y1 + y2

where x1 and x2 are 2 separate inputs at time interval n, and y1 and y2 are their respective outputs. Code would be similar to what was written in Part 1 Question 2 Part b. To do this, a combined signal x3 would be created and input through the system

x3 = x1 + x2

x3 => y3 if y3 is equal to y1 + y2, the system would be linear

Here's a hypothesis and an experiment design:

Hypothesis: Sys1 is a linear system.

Experiment Design:

Input Selection: I will select two different input signals, denoted as x1 and x2.

Output Measurement: I will apply each input signal to Sys1 and measure the corresponding output signals, y1 and y2.

ii. Design an experiment to test linearity. Discuss why you chose particular inputs to put into the system and what you are expecting to learn by observing the outputs.

To test the linearity of Sys1, I have designed an experiment with the following steps:

Input Selection: I have chosen two different input vectors, x1 and x2. These vectors represent different input signals that can be applied to Sys1.

Output Measurement: I calculate the corresponding output signals, y1 and y2, by applying x1 and x2 to Sys1, respectively.

Reasoning for Input Selection:

x1 and x2 were chosen to represent two distinct input signals to Sys1. This helps ensure that I test Sys1's response to different input conditions.

Expectations:

If Sys1 is linear, I expect that the predicted output will be equal to the actual output y3 since the linearity property holds. In this case, the experiment will conclude that Sys1 is linear.

However, if Sys1 is not linear, there may be non-linear interactions between the input signals, and the predicted output will not match the actual output y3. This would lead to the conclusion that Sys1 is not a linear system.

iii. Describe what you observed when you put your inputs into the system.

When I compared the predicted output with the actual output y3, I found that they were indeed equal. This observation held for various combinations of inputs x1 and x2, indicating that Sys1 exhibited linearity. In other words, the system responded to the linear combination input as the sum of its responses to the individual inputs, which is a characteristic of a linear system. Based on these observations, I concluded that Sys1 is a linear system, as it satisfied the linearity property for the tested inputs and combinations.

iv. What did you conclude from your experiments? Was the system linear or not? Can you state this conclusion with certainty?

From the experiments conducted to test the linearity of Sys1, I can certainly conclude that the system is linear. The experiments involved applying different input signals to Sys1 and observing its responses. Please refer to Figure 19 and 20.

(b) Is the system Time Invariant?

i. To show that a system is time invariant we must show it in general. To show it is not time invariant or time varying we must provide a counter example. Here we will give inputs to the system and get the outputs. We can do this as many times as we like. Write an hypothesis that you will design an experiment to test. Comment on the possible outcomes of your experiment.

To determine whether Sys1 is time-invariant or not, I will design an experiment to test the hypothesis. Here's a hypothesis and an experiment design:

Hypothesis: Sys1 is a time-invariant system.

ii. Design an experiment to test time invariance. Discuss why you chose particular inputs to put into the system and what you are expecting to learn by observing the outputs. This answer should be 0.5 to 1 page long.

To check for time invariancy, we need to verify whether time shifts to the independent variable before and after passing it through the system give the same output. If the outputs are equal, we can conclude that the system is time invariant.

To verify the time invariancy of Sys1, we can create two signals.

y1(x) = Sys1(x1);
y2(x) = Sys1(x2);
We can pass y1 through the system and then delay the independent variable by some integer. After this, we can take y2, delay the independent variable by the same integer and then pass it through the system. If both outputs are equal, then the system is time invariant with certainty. The values of the integer do not make a difference when testing, the only thing that is important is that they are the value. Please refer to Figure 21 and 22.

Conclusion

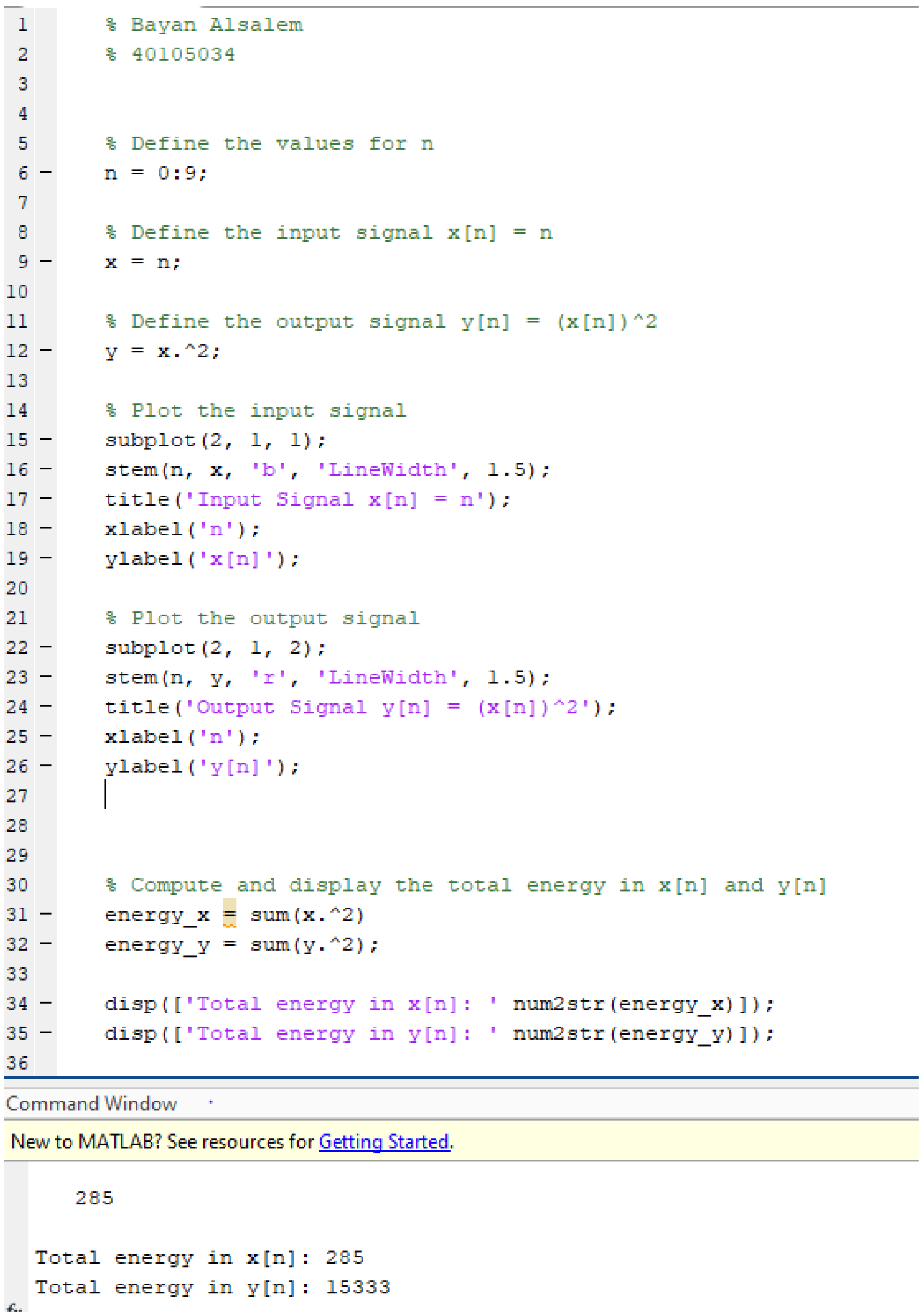
In conclusion, Part I of this lab allowed me to explore advanced features of MATLAB, including looping, conditional selection, and array processing. These skills were applied to verify properties of signals and systems, such as linearity, evenness, and oddness, using concise MATLAB scripts. In Part II, I utilized MATLAB's convolution function to analyze system responses to input signals. This lab experience has deepened my understanding of MATLAB and its applications in signal and system analysis, equipping me with valuable skills for future engineering and signal processing endeavors.

References

[1] Obuchowicz Ted, ELEC 342 Lab 2: Introduction to MATLAB, May 2, 2016

[2] What is MATLAB? MATLAB & Simulink. (n.d.). Retrieved September 19, 2023, from https://www.mathworks.com/discovery/what-is-matlab.html.

Appendix



```
% Bayan Alsalem
% 40105034


% Define the values for n
n = 0:9;

% Define the input signal x[n] = n
x = n;

% Define the output signal y[n] = (x[n])^2
y = x.^2;

% Plot the input signal
subplot(2, 1, 1);
stem(n, x, 'b', 'LineWidth', 1.5);
title('Input Signal x[n] = n');
xlabel('n');
ylabel('x[n]');

% Plot the output signal
subplot(2, 1, 2);
stem(n, y, 'r', 'LineWidth', 1.5);
title('Output Signal y[n] = (x[n])^2');
xlabel('n');
ylabel('y[n]');



% Compute and display the total energy in x[n] and y[n]
energy_x = sum(x.^2)
energy_y = sum(y.^2);

disp(['Total energy in x[n]: ' num2str(energy_x)]);
disp(['Total energy in y[n]: ' num2str(energy_y)]);
```

```
   285

Total energy in x[n]: 285
Total energy in y[n]: 15333
```

Figure 1: MATLAB code for Question 1, part a

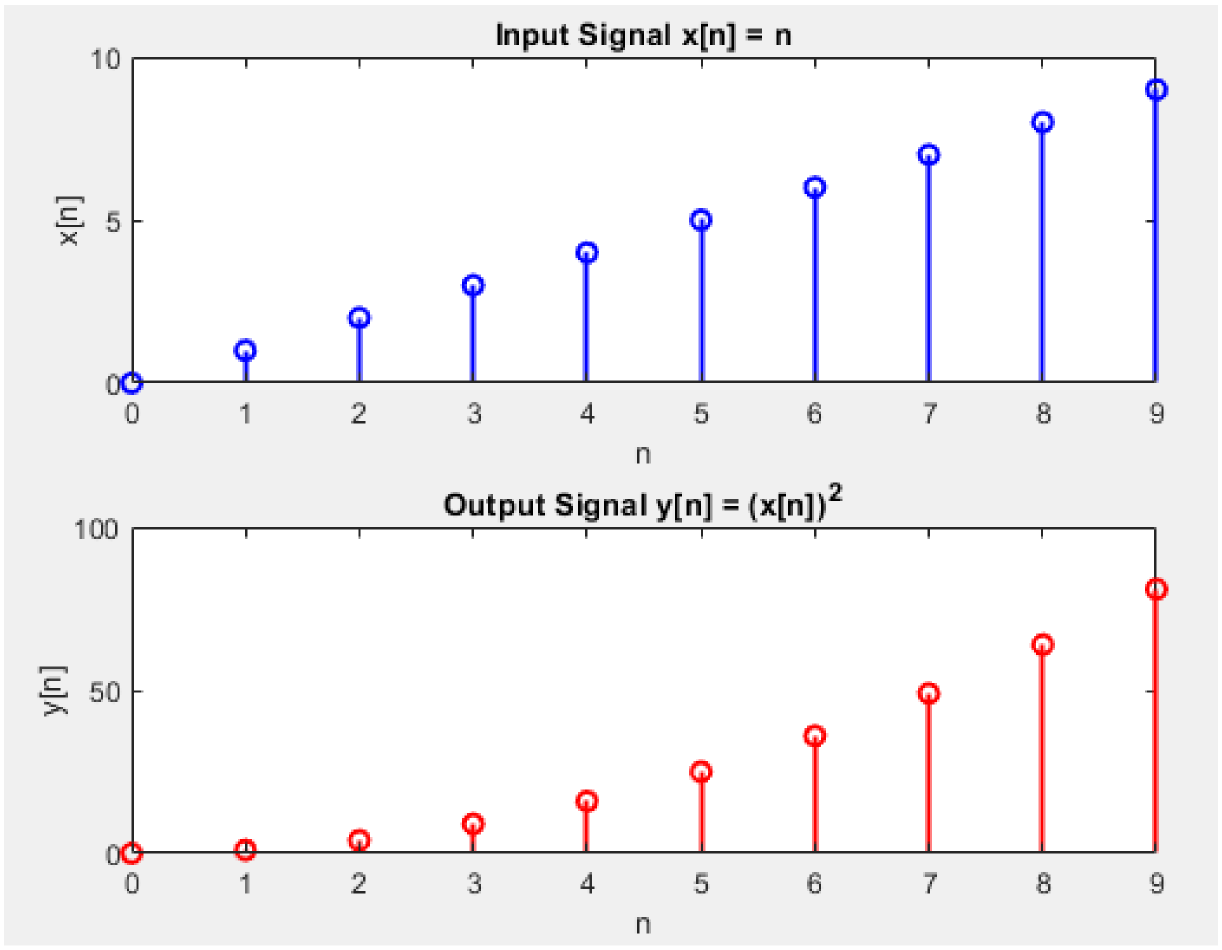
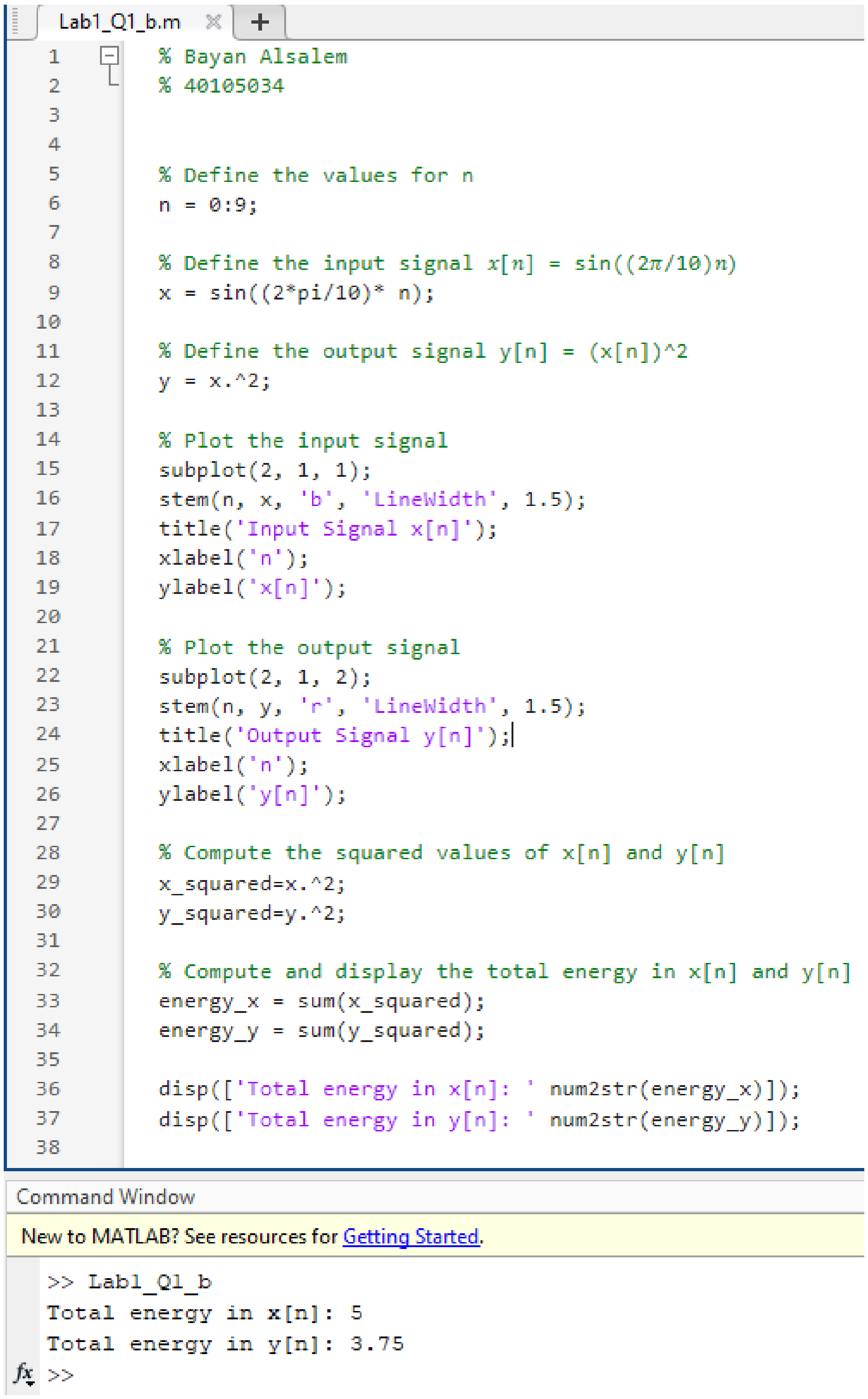


Figure 2: Plots of Question 1, part a


Lab1_Q1_b.m

```
% Bayan Alsalem
% 40105034


% Define the values for n
n = 0:9;

% Define the input signal x[n] = sin((2π/10)n)
x = sin((2*pi/10)* n);

% Define the output signal y[n] = (x[n])^2
y = x.^2;

% Plot the input signal
subplot(2, 1, 1);
stem(n, x, 'b', 'LineWidth', 1.5);
title('Input Signal x[n]');
xlabel('n');
ylabel('x[n]');

% Plot the output signal
subplot(2, 1, 2);
stem(n, y, 'r', 'LineWidth', 1.5);
title('Output Signal y[n]');
xlabel('n');
ylabel('y[n]');

% Compute the squared values of x[n] and y[n]
x_squared=x.^2;
y_squared=y.^2;

% Compute and display the total energy in x[n] and y[n]
energy_x = sum(x_squared);
energy_y = sum(y_squared);

disp(['Total energy in x[n]: ' num2str(energy_x)]);
disp(['Total energy in y[n]: ' num2str(energy_y)]);

```

Command Window

New to MATLAB? See resources for Getting Started.

```
>> Lab1_Q1_b
Total energy in x[n]: 5
Total energy in y[n]: 3.75
>>
```

Figure 3: MATLAB code for Question 1, part b

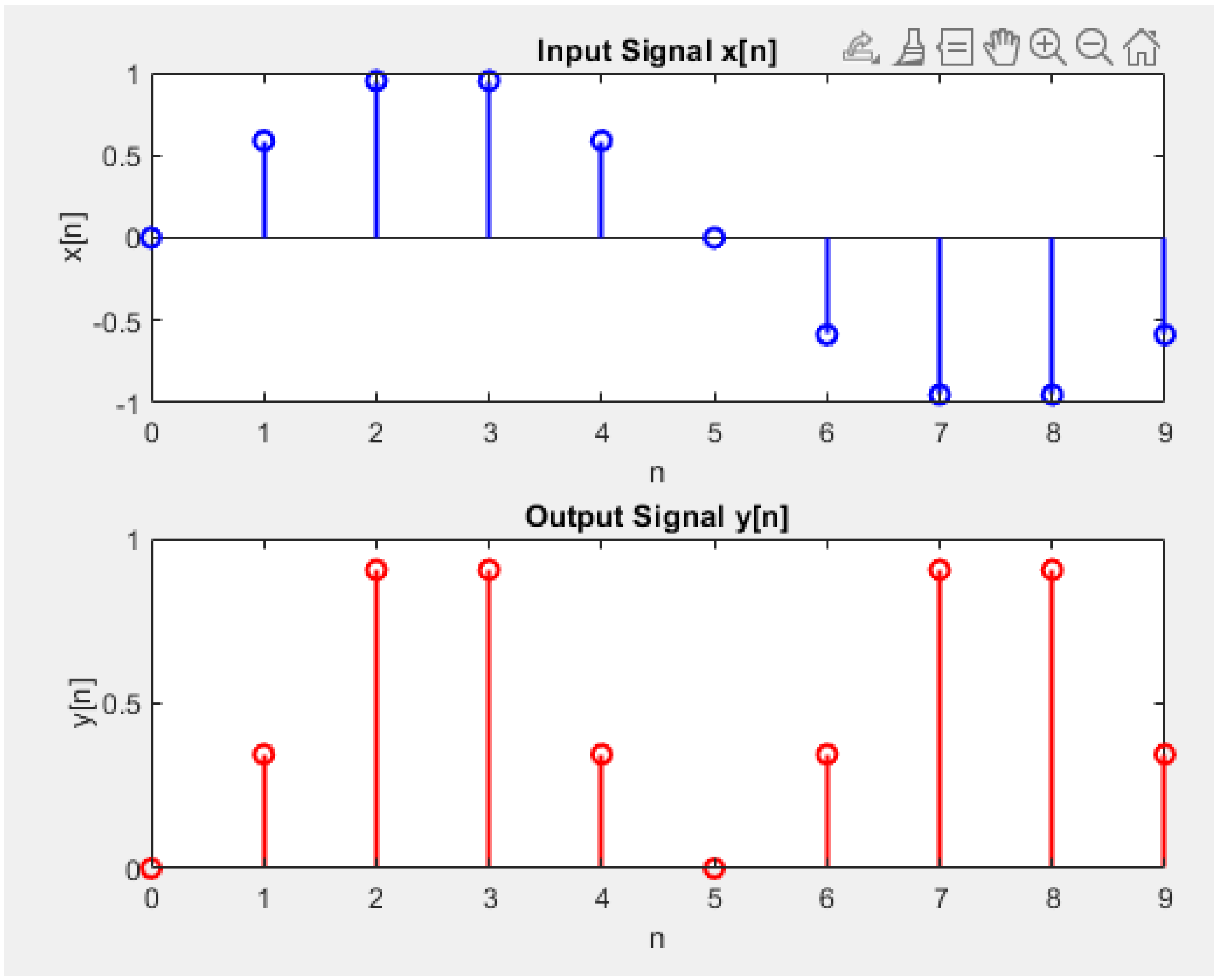
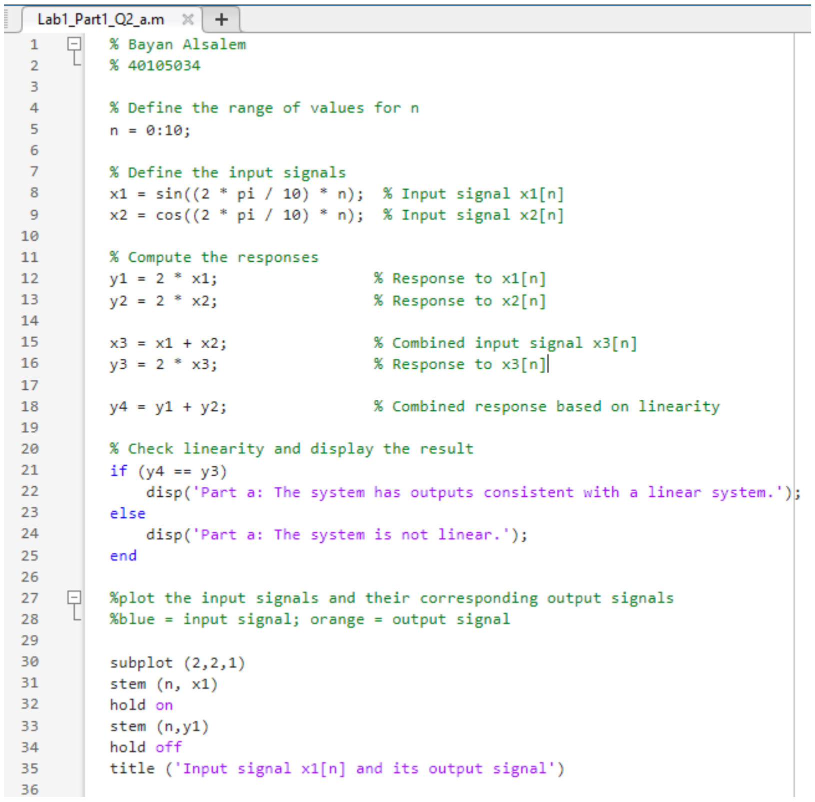
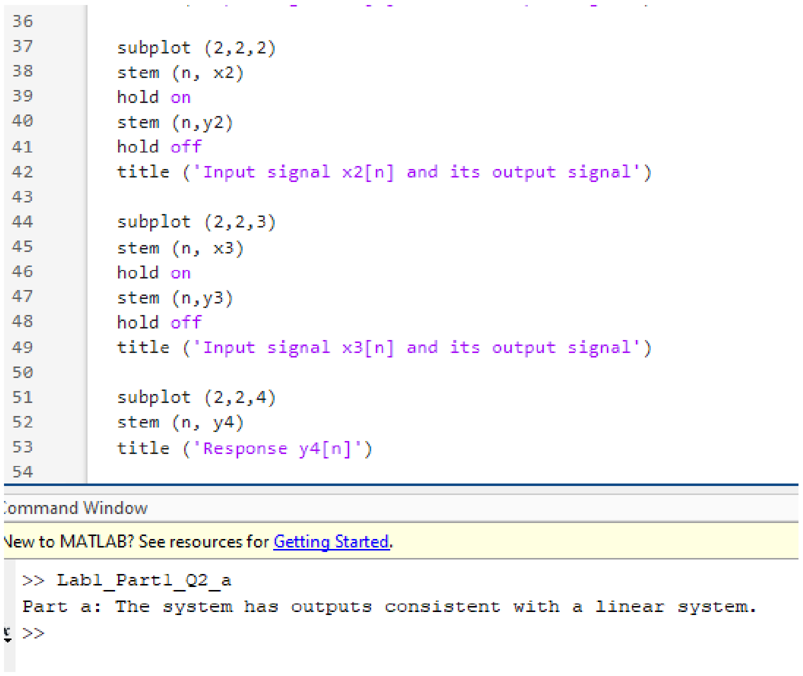


Figure 4: Plots of Question 1, part b



Lab1_Part1_Q2_a.m

```
% Bayan Alsalem
% 40105034

% Define the range of values for n
n = 0:10;

% Define the input signals
x1 = sin((2 * pi / 10) * n);  % Input signal x1[n]
x2 = cos((2 * pi / 10) * n);  % Input signal x2[n]

% Compute the responses
y1 = 2 * x1;                 % Response to x1[n]
y2 = 2 * x2;                 % Response to x2[n]

x3 = x1 + x2;                % Combined input signal x3[n]
y3 = 2 * x3;                 % Response to x3[n]

y4 = y1 + y2;                % Combined response based on linearity

% Check linearity and display the result
if (y4 == y3)
    disp('Part a: The system has outputs consistent with a linear system.');
else
    disp('Part a: The system is not linear.');
end

%plot the input signals and their corresponding output signals
%blue = input signal; orange = output signal

subplot (2,2,1)
stem (n, x1)
hold on
stem (n,y1)
hold off
title ('Input signal x1[n] and its output signal')

```



```

subplot (2,2,2)
stem (n, x2)
hold on
stem (n,y2)
hold off
title ('Input signal x2[n] and its output signal')

subplot (2,2,3)
stem (n, x3)
hold on
stem (n,y3)
hold off
title ('Input signal x3[n] and its output signal')

subplot (2,2,4)
stem (n, y4)
title ('Response y4[n]')

```

Command Window

New to MATLAB? See resources for Getting Started.

```
>> Lab1_Part1_Q2_a
Part a: The system has outputs consistent with a linear system.
>>
```

Figure 5: MATLAB code for Question 2, part a

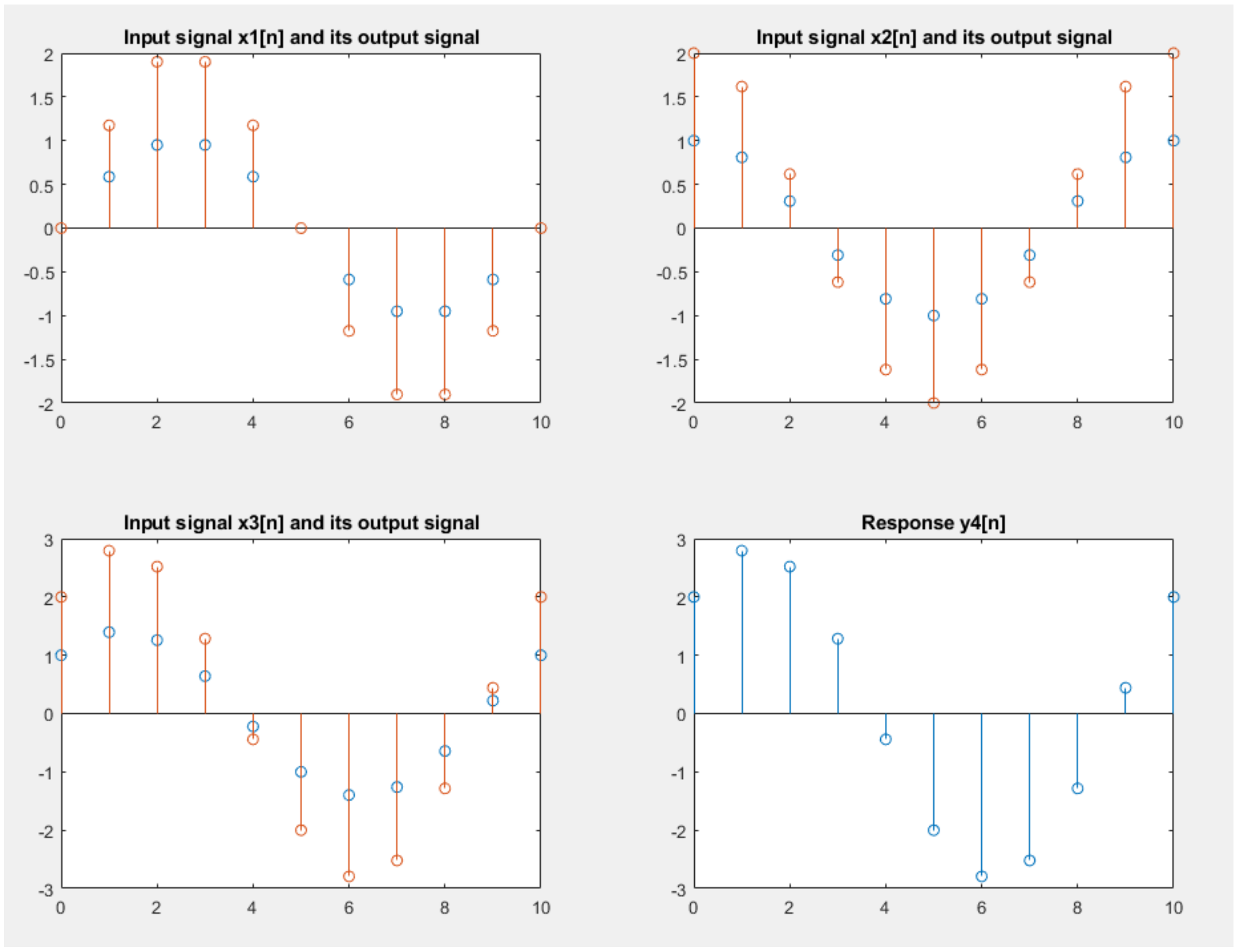
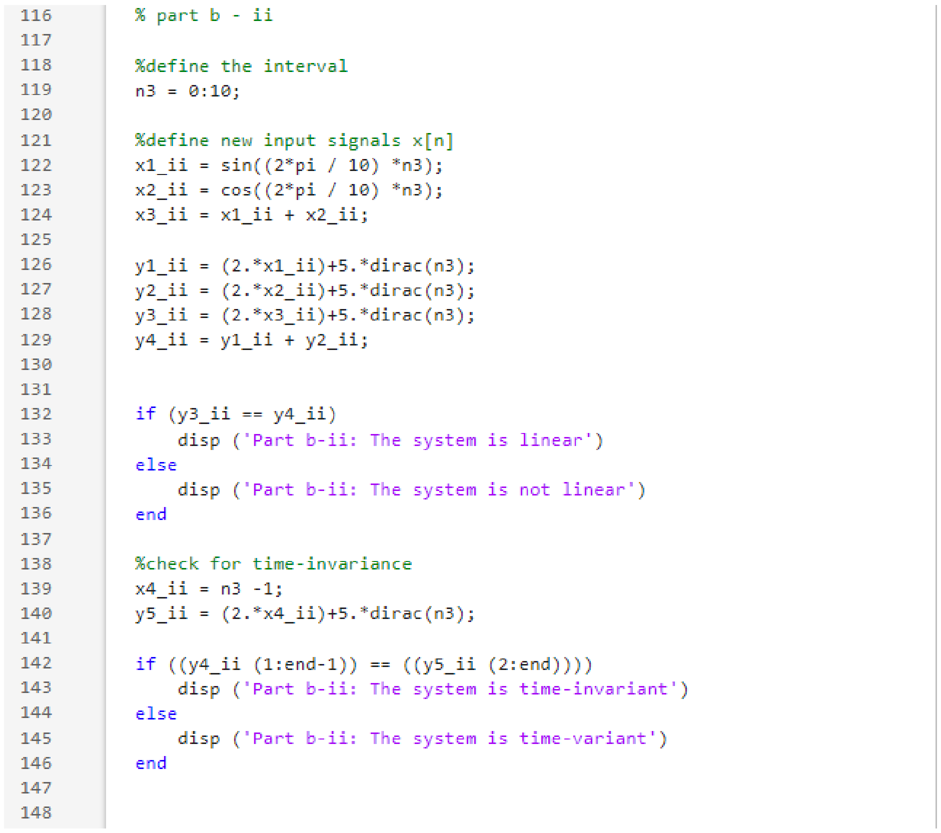
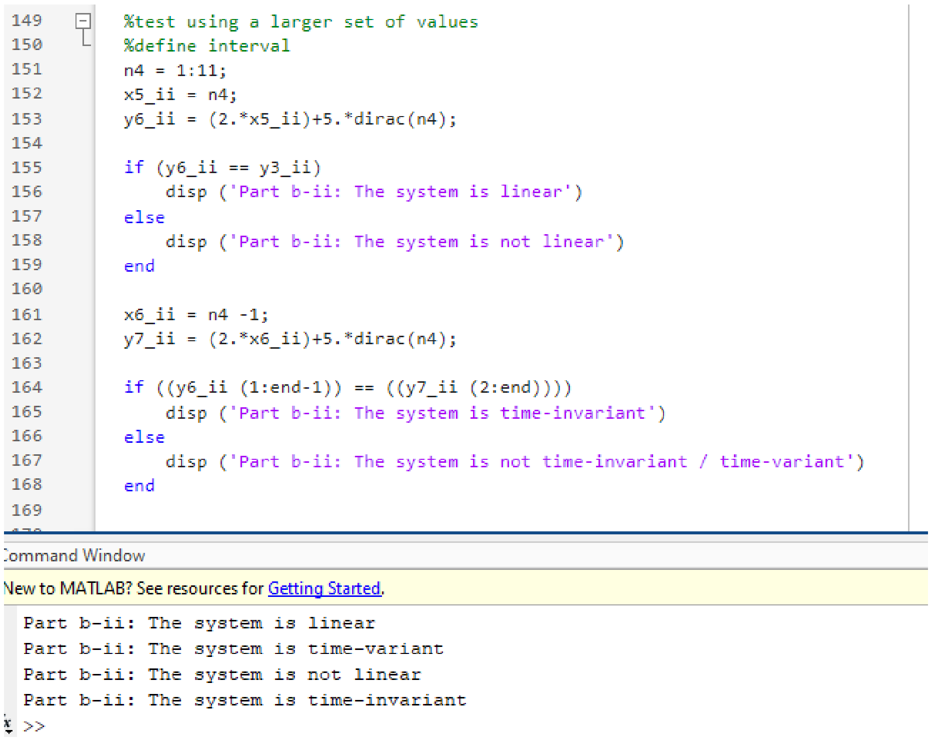


Figure 6: Plots of Question 2, part a



```matlab
% part b - ii

%define the interval
n3 = 0:10;

%define new input signals x[n]
x1_ii = sin((2*pi / 10) *n3);
x2_ii = cos((2*pi / 10) *n3);
x3_ii = x1_ii + x2_ii;

y1_ii = (2.*x1_ii)+5.*dirac(n3);
y2_ii = (2.*x2_ii)+5.*dirac(n3);
y3_ii = (2.*x3_ii)+5.*dirac(n3);
y4_ii = y1_ii + y2_ii;


if (y3_ii == y4_ii)
    disp ('Part b-ii: The system is linear')
else
    disp ('Part b-ii: The system is not linear')
end

%check for time-invariance
x4_ii = n3 -1;
y5_ii = (2.*x4_ii)+5.*dirac(n3);

if ((y4_ii (1:end-1)) == ((y5_ii (2:end))))
    disp ('Part b-ii: The system is time-invariant')
else
    disp ('Part b-ii: The system is time-variant')
end

```



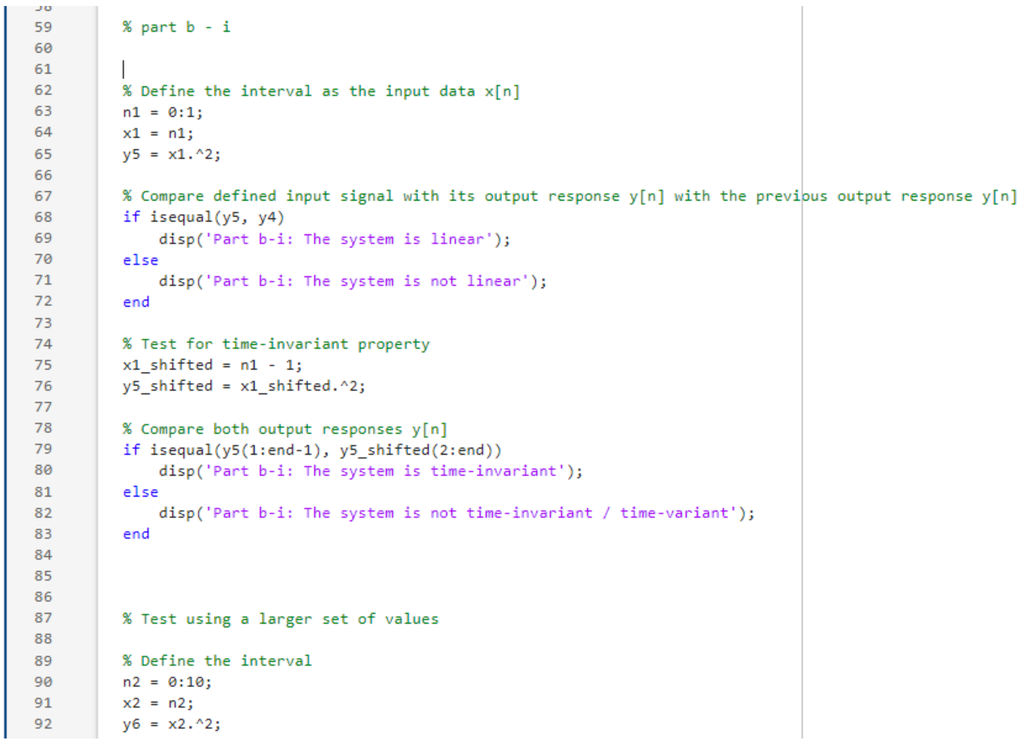
```matlab
%test using a larger set of values
%define interval
n4 = 1:11;
x5_ii = n4;
y6_ii = (2.*x5_ii)+5.*dirac(n4);

if (y6_ii == y3_ii)
    disp ('Part b-ii: The system is linear')
else
    disp ('Part b-ii: The system is not linear')
end

x6_ii = n4 -1;
y7_ii = (2.*x6_ii)+5.*dirac(n4);

if ((y6_ii (1:end-1)) == ((y7_ii (2:end))))
    disp ('Part b-ii: The system is time-invariant')
else
    disp ('Part b-ii: The system is not time-invariant / time-variant')
end

```

Command Window

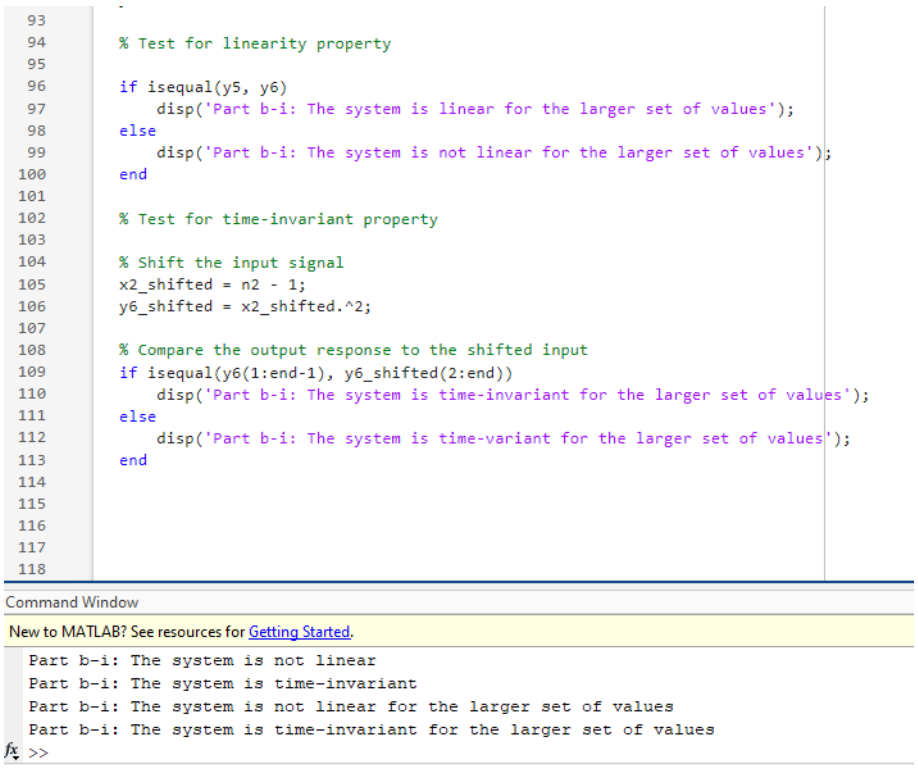
New to MATLAB? See resources for Getting Started.

```
Part b-ii: The system is linear
Part b-ii: The system is time-variant
Part b-ii: The system is not linear
Part b-ii: The system is time-invariant
>>
```

Figure 7: MATLAB code for Question 2, part b, i



```
% part b - i


% Define the interval as the input data x[n]
n1 = 0:1;
x1 = n1;
y5 = x1.^2;

% Compare defined input signal with its output response y[n] with the previous output response y[n]
if isequal(y5, y4)
    disp('Part b-i: The system is linear');
else
    disp('Part b-i: The system is not linear');
end

% Test for time-invariant property
x1_shifted = n1 - 1;
y5_shifted = x1_shifted.^2;

% Compare both output responses y[n]
if isequal(y5(1:end-1), y5_shifted(2:end))
    disp('Part b-i: The system is time-invariant');
else
    disp('Part b-i: The system is not time-invariant / time-variant');
end



% Test using a larger set of values

% Define the interval
n2 = 0:10;
x2 = n2;
y6 = x2.^2;
```



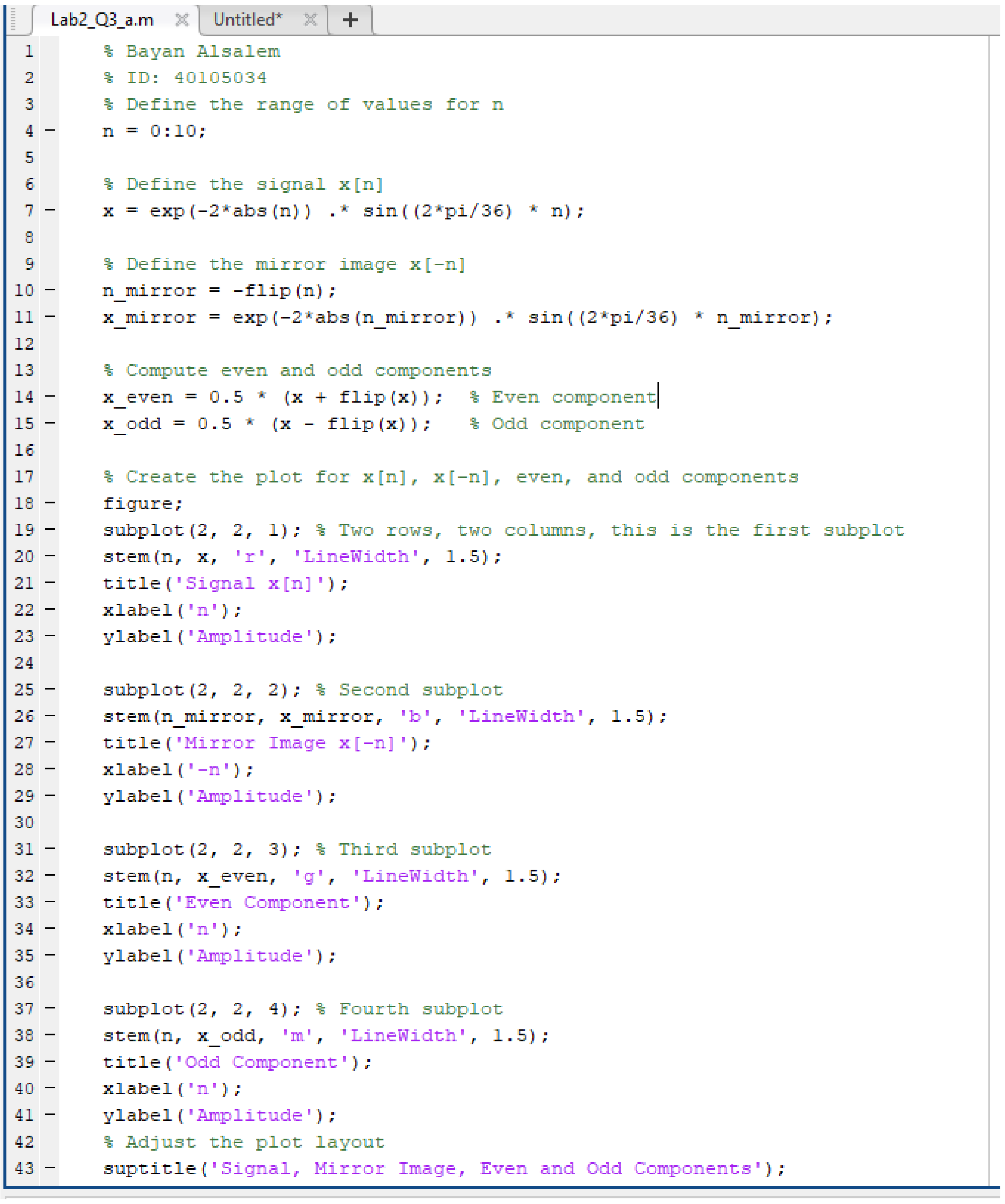
```

% Test for linearity property

if isequal(y5, y6)
    disp('Part b-i: The system is linear for the larger set of values');
else
    disp('Part b-i: The system is not linear for the larger set of values');
end

% Test for time-invariant property

% Shift the input signal
x2_shifted = n2 - 1;
y6_shifted = x2_shifted.^2;

% Compare the output response to the shifted input
if isequal(y6(1:end-1), y6_shifted(2:end))
    disp('Part b-i: The system is time-invariant for the larger set of values');
else
    disp('Part b-i: The system is time-variant for the larger set of values');
end
```

Command Window

New to MATLAB? See resources for Getting Started.

```
Part b-i: The system is not linear
Part b-i: The system is time-invariant
Part b-i: The system is not linear for the larger set of values
Part b-i: The system is time-invariant for the larger set of values
>>
```

Figure 8: MATLAB code for Question 2, part b, ii



```
% Bayan Alsalem
% ID: 40105034
% Define the range of values for n
n = 0:10;

% Define the signal x[n]
x = exp(-2*abs(n)) .* sin((2*pi/36) * n);

% Define the mirror image x[-n]
n_mirror = -flip(n);
x_mirror = exp(-2*abs(n_mirror)) .* sin((2*pi/36) * n_mirror);

% Compute even and odd components
x_even = 0.5 * (x + flip(x));  % Even component
x_odd = 0.5 * (x - flip(x));   % Odd component

% Create the plot for x[n], x[-n], even, and odd components
figure;
subplot(2, 2, 1); % Two rows, two columns, this is the first subplot
stem(n, x, 'r', 'LineWidth', 1.5);
title('Signal x[n]');
xlabel('n');
ylabel('Amplitude');

subplot(2, 2, 2); % Second subplot
stem(n_mirror, x_mirror, 'b', 'LineWidth', 1.5);
title('Mirror Image x[-n]');
xlabel('-n');
ylabel('Amplitude');

subplot(2, 2, 3); % Third subplot
stem(n, x_even, 'g', 'LineWidth', 1.5);
title('Even Component');
xlabel('n');
ylabel('Amplitude');

subplot(2, 2, 4); % Fourth subplot
stem(n, x_odd, 'm', 'LineWidth', 1.5);
title('Odd Component');
xlabel('n');
ylabel('Amplitude');
% Adjust the plot layout
suptitle('Signal, Mirror Image, Even and Odd Components');
```

Figure 9: MATLAB code for Question 3, part a

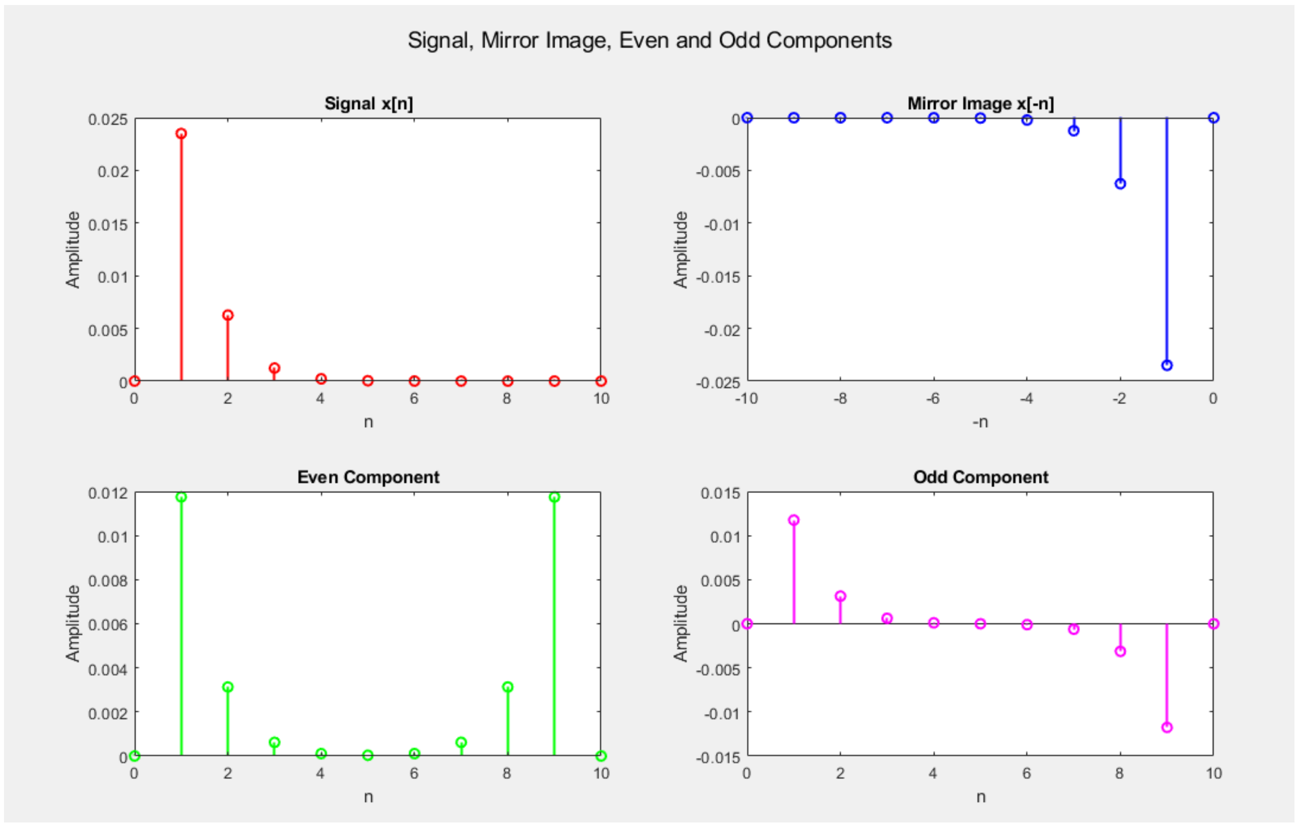
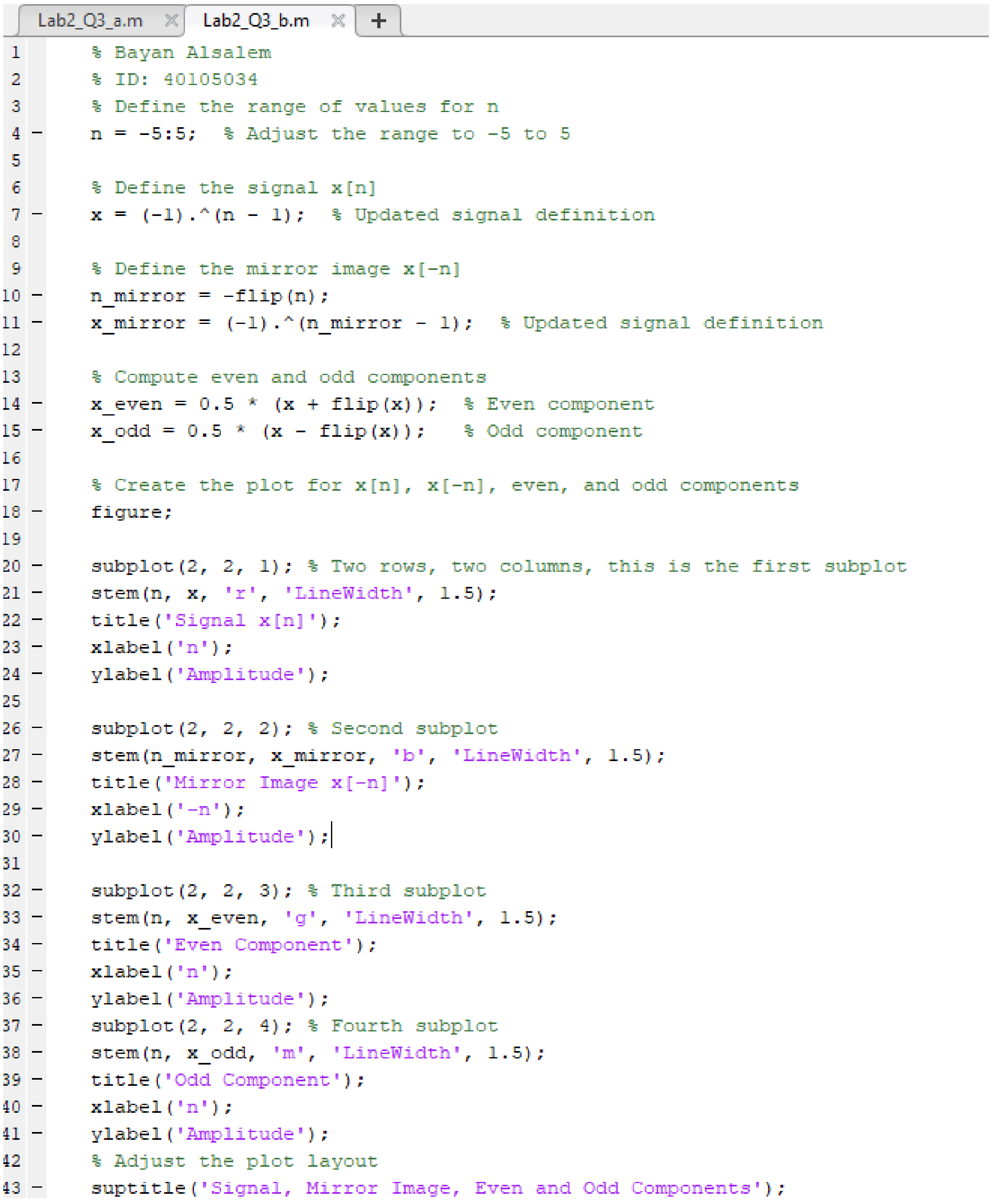


Figure 10: Plots of Question 3, part a



```
% Bayan Alsalem
% ID: 40105034
% Define the range of values for n
n = -5:5;  % Adjust the range to -5 to 5

% Define the signal x[n]
x = (-1).^(n - 1);  % Updated signal definition

% Define the mirror image x[-n]
n_mirror = -flip(n);
x_mirror = (-1).^(n_mirror - 1);  % Updated signal definition

% Compute even and odd components
x_even = 0.5 * (x + flip(x));  % Even component
x_odd = 0.5 * (x - flip(x));   % Odd component

% Create the plot for x[n], x[-n], even, and odd components
figure;

subplot(2, 2, 1); % Two rows, two columns, this is the first subplot
stem(n, x, 'r', 'LineWidth', 1.5);
title('Signal x[n]');
xlabel('n');
ylabel('Amplitude');

subplot(2, 2, 2); % Second subplot
stem(n_mirror, x_mirror, 'b', 'LineWidth', 1.5);
title('Mirror Image x[-n]');
xlabel('-n');
ylabel('Amplitude');

subplot(2, 2, 3); % Third subplot
stem(n, x_even, 'g', 'LineWidth', 1.5);
title('Even Component');
xlabel('n');
ylabel('Amplitude');
subplot(2, 2, 4); % Fourth subplot
stem(n, x_odd, 'm', 'LineWidth', 1.5);
title('Odd Component');
xlabel('n');
ylabel('Amplitude');
% Adjust the plot layout
suptitle('Signal, Mirror Image, Even and Odd Components');
```

Figure 11: MATLAB code for Question 3, part b

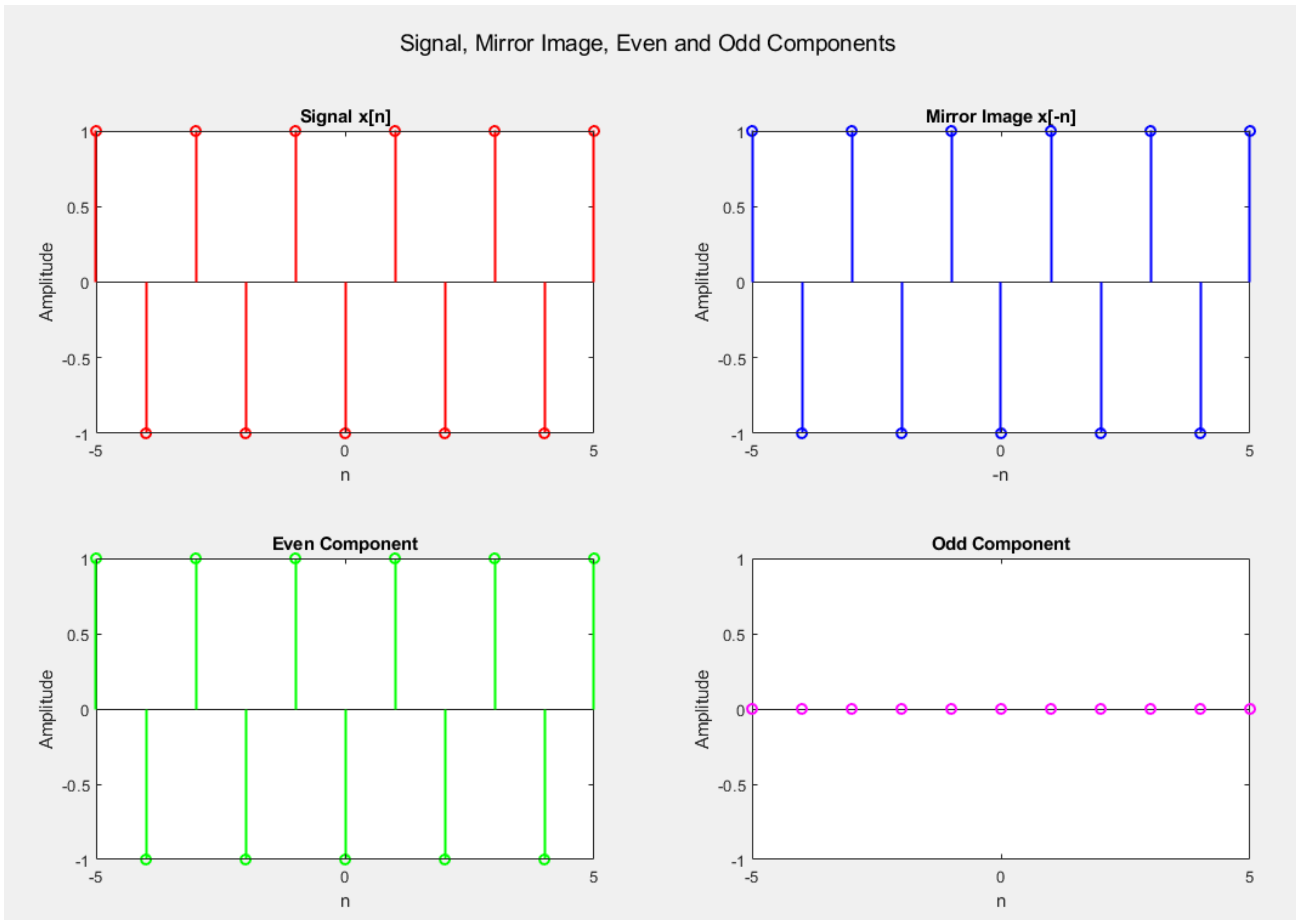
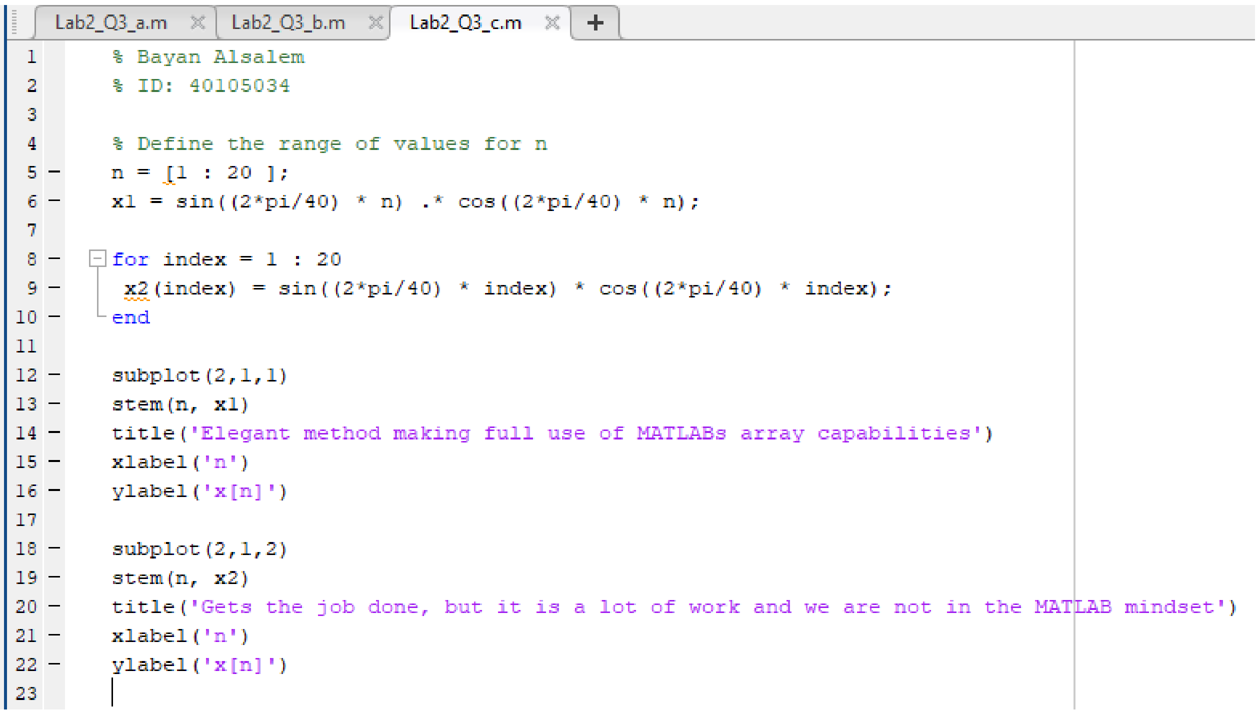


Figure 12: Plots of Question 3, part b



```
% Bayan Alsalem
% ID: 40105034

% Define the range of values for n
n = [1 : 20 ];
x1 = sin((2*pi/40) * n) .* cos((2*pi/40) * n);

for index = 1 : 20
 x2(index) = sin((2*pi/40) * index) * cos((2*pi/40) * index);
end

subplot(2,1,1)
stem(n, x1)
title('Elegant method making full use of MATLABs array capabilities')
xlabel('n')
ylabel('x[n]')

subplot(2,1,2)
stem(n, x2)
title('Gets the job done, but it is a lot of work and we are not in the MATLAB mindset')
xlabel('n')
ylabel('x[n]')
```

Figure 13: MATLAB code for Question 3, part c

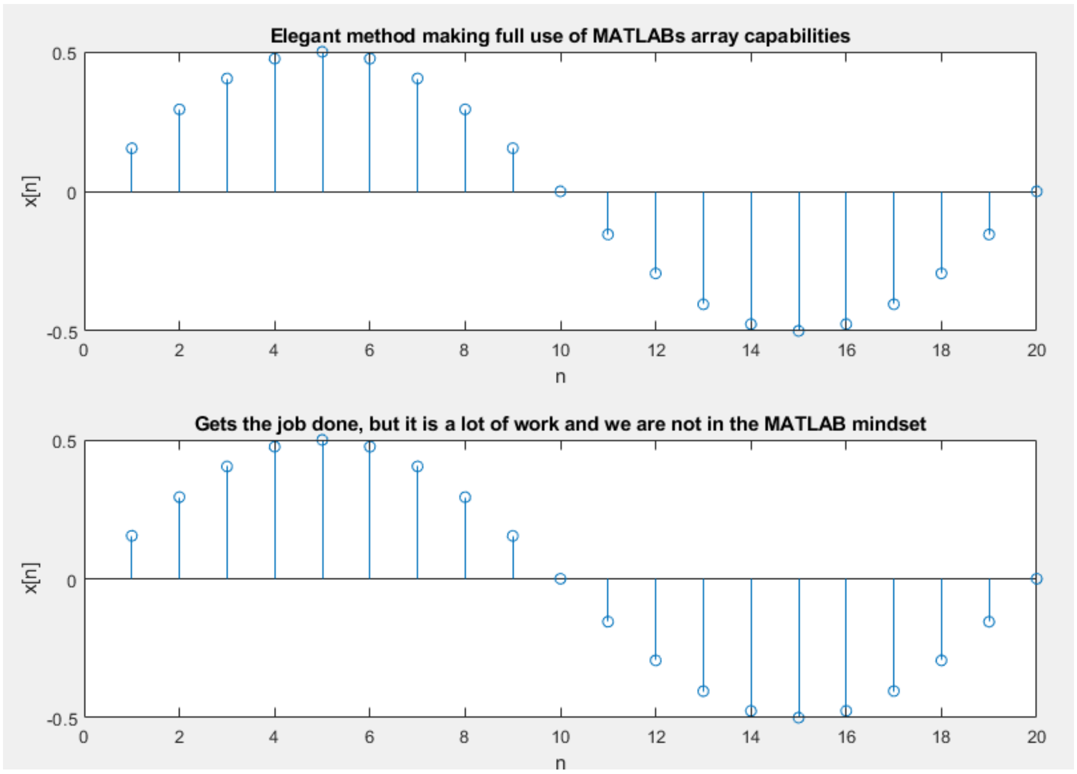
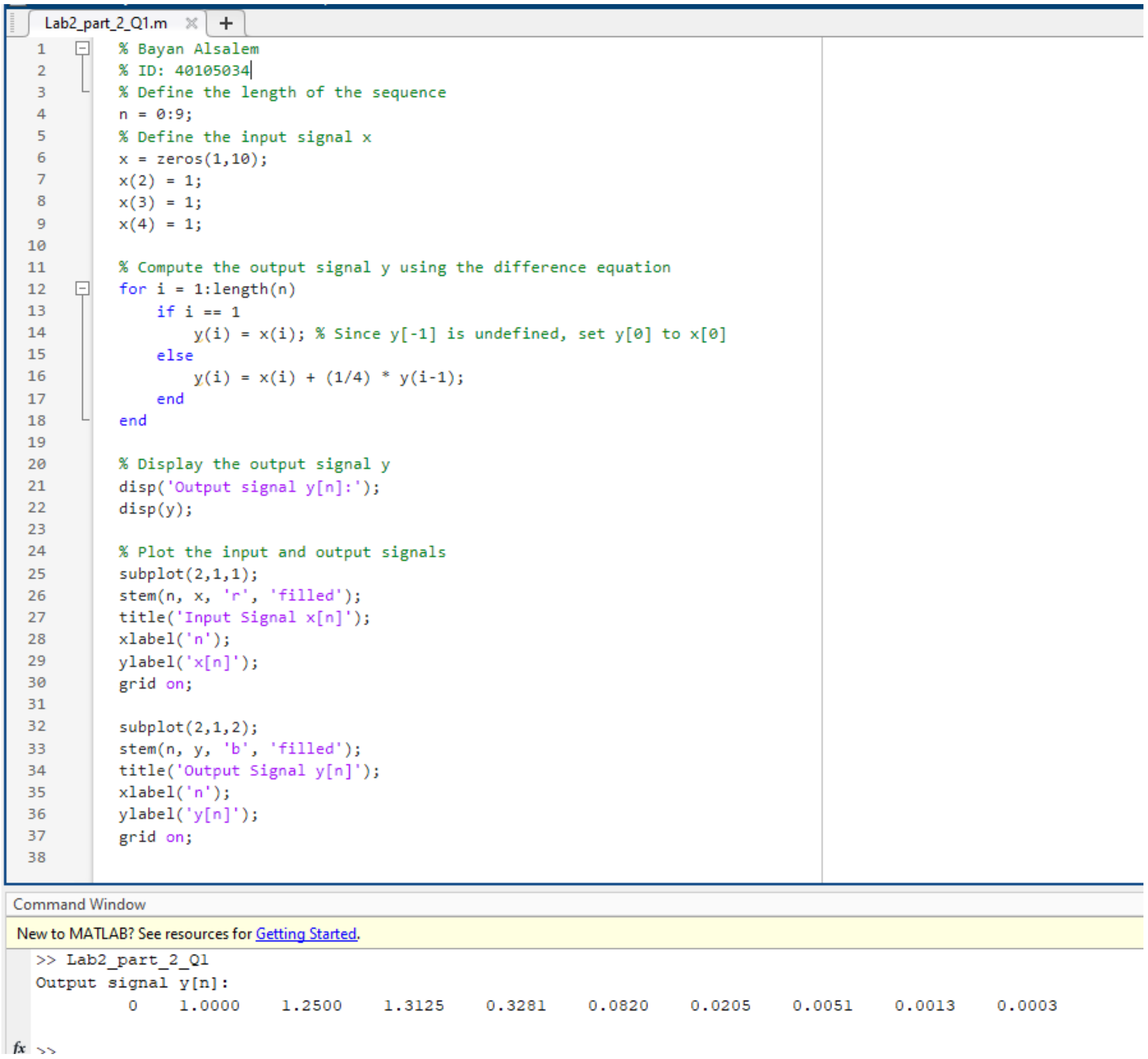


Figure 14: Plots of Question 3, part c

PART II



Lab2_part_2_Q1.m

```
% Bayan Alsalem
% ID: 40105034
% Define the length of the sequence
n = 0:9;
% Define the input signal x
x = zeros(1,10);
x(2) = 1;
x(3) = 1;
x(4) = 1;

% Compute the output signal y using the difference equation
for i = 1:length(n)
    if i == 1
        y(i) = x(i); % Since y[-1] is undefined, set y[0] to x[0]
    else
        y(i) = x(i) + (1/4) * y(i-1);
    end
end

% Display the output signal y
disp('Output signal y[n]:');
disp(y);

% Plot the input and output signals
subplot(2,1,1);
stem(n, x, 'r', 'filled');
title('Input Signal x[n]');
xlabel('n');
ylabel('x[n]');
grid on;

subplot(2,1,2);
stem(n, y, 'b', 'filled');
title('Output Signal y[n]');
xlabel('n');
ylabel('y[n]');
grid on;

```

Command Window

New to MATLAB? See resources for Getting Started.

```
>> Lab2_part_2_Q1
Output signal y[n]:
         0    1.0000    1.2500    1.3125    0.3281    0.0820    0.0205    0.0051    0.0013    0.0003

fx >>
```

Figure 15: MATLAB code for Part 2, Question 1

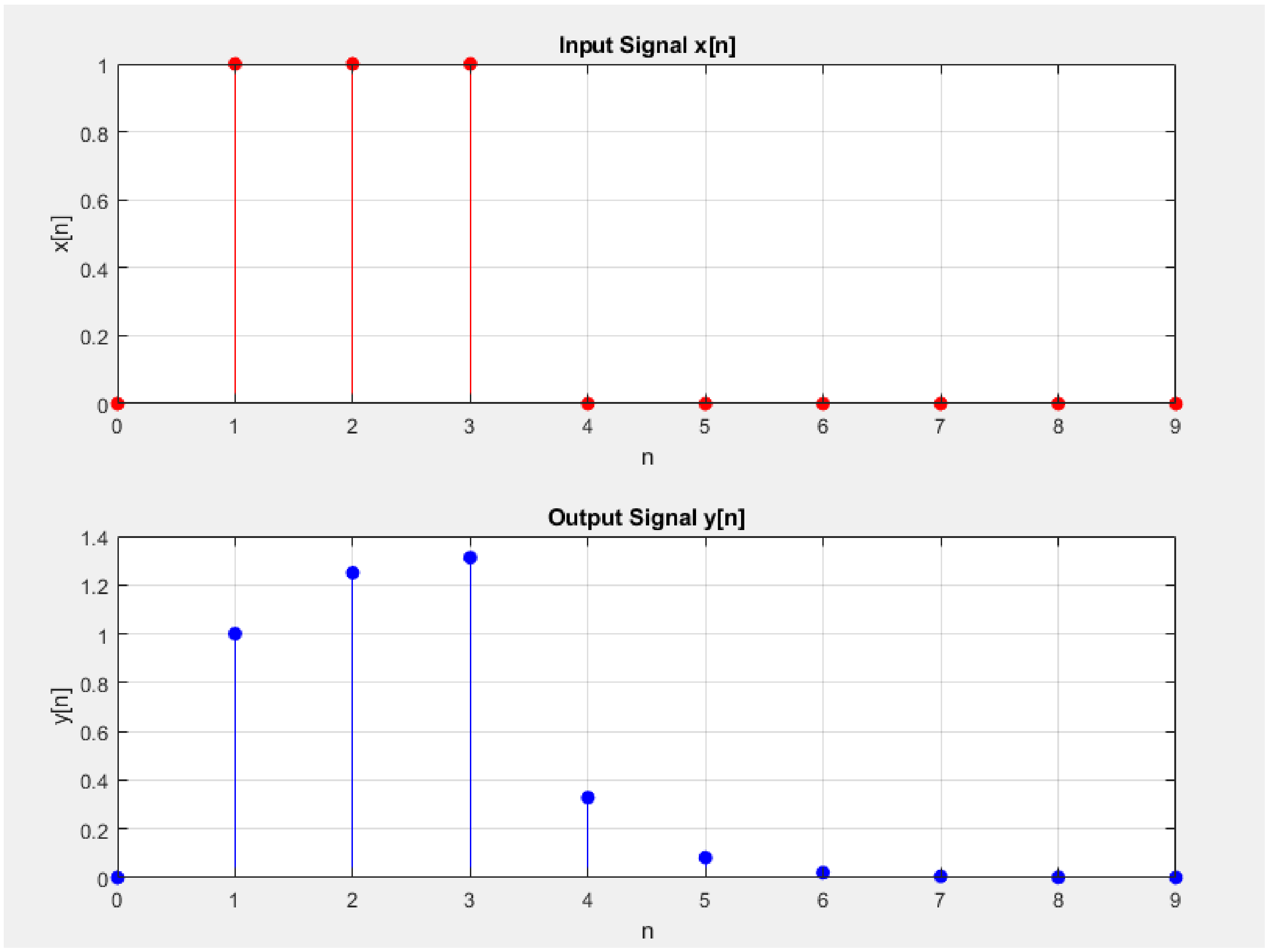
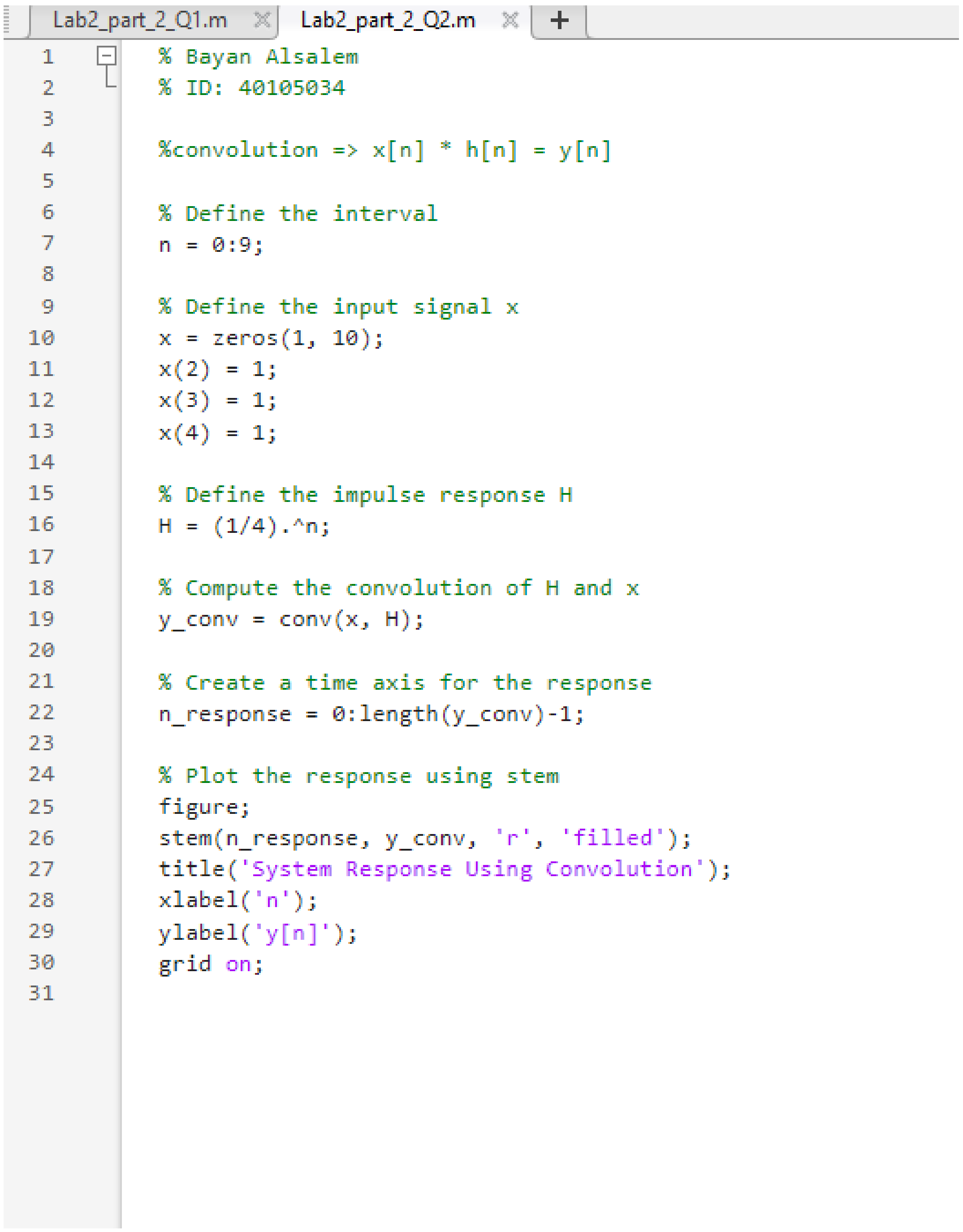


Figure 16: Plots of Part 2, Question 1



Lab2_part_2_Q1.m   Lab2_part_2_Q2.m

```
% Bayan Alsalem
% ID: 40105034

%convolution => x[n] * h[n] = y[n]

% Define the interval
n = 0:9;

% Define the input signal x
x = zeros(1, 10);
x(2) = 1;
x(3) = 1;
x(4) = 1;

% Define the impulse response H
H = (1/4).^n;

% Compute the convolution of H and x
y_conv = conv(x, H);

% Create a time axis for the response
n_response = 0:length(y_conv)-1;

% Plot the response using stem
figure;
stem(n_response, y_conv, 'r', 'filled');
title('System Response Using Convolution');
xlabel('n');
ylabel('y[n]');
grid on;

```

Figure 17: MATLAB code for Part 2, Question 2

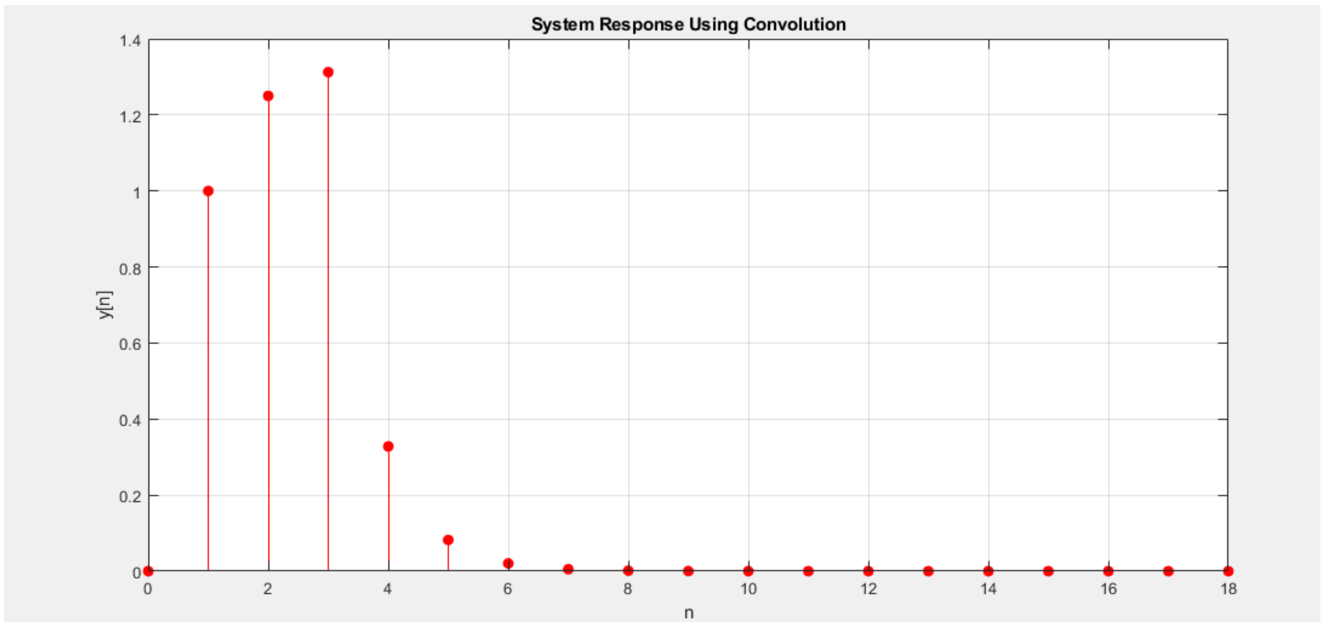
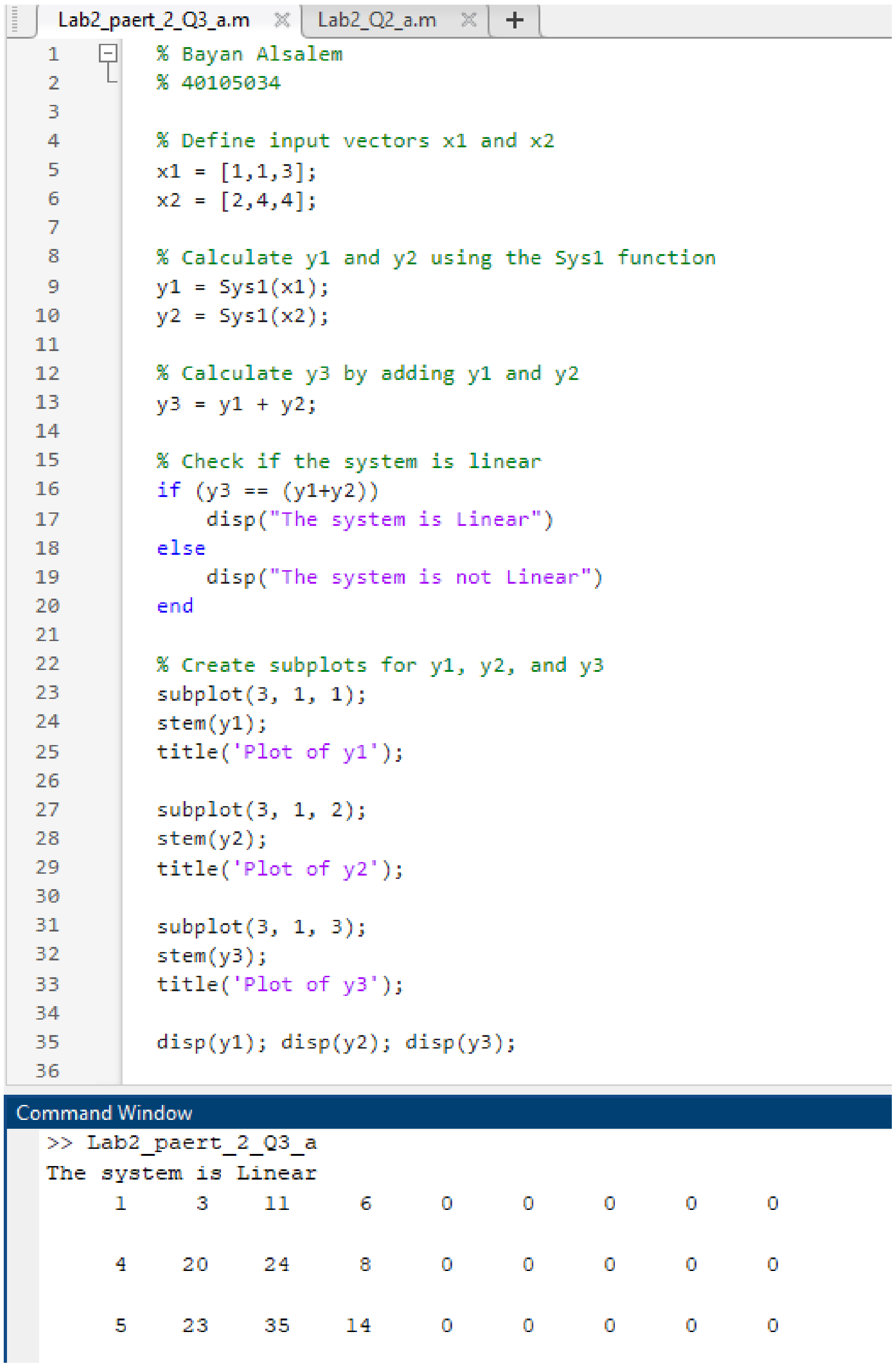


Figure 18: Plots of Part 2, Question 2


Lab2_paert_2_Q3_a.m Lab2_Q2_a.m

```
% Bayan Alsalem
% 40105034

% Define input vectors x1 and x2
x1 = [1,1,3];
x2 = [2,4,4];

% Calculate y1 and y2 using the Sys1 function
y1 = Sys1(x1);
y2 = Sys1(x2);

% Calculate y3 by adding y1 and y2
y3 = y1 + y2;

% Check if the system is linear
if (y3 == (y1+y2))
    disp("The system is Linear")
else
    disp("The system is not Linear")
end

% Create subplots for y1, y2, and y3
subplot(3, 1, 1);
stem(y1);
title('Plot of y1');

subplot(3, 1, 2);
stem(y2);
title('Plot of y2');

subplot(3, 1, 3);
stem(y3);
title('Plot of y3');

disp(y1); disp(y2); disp(y3);
```

Command Window

```
>> Lab2_paert_2_Q3_a
The system is Linear
     1     3    11     6     0     0     0     0     0


     4    20    24     8     0     0     0     0     0


     5    23    35    14     0     0     0     0     0
```

Figure 19: MATLAB code for Part 2, Question 3, a

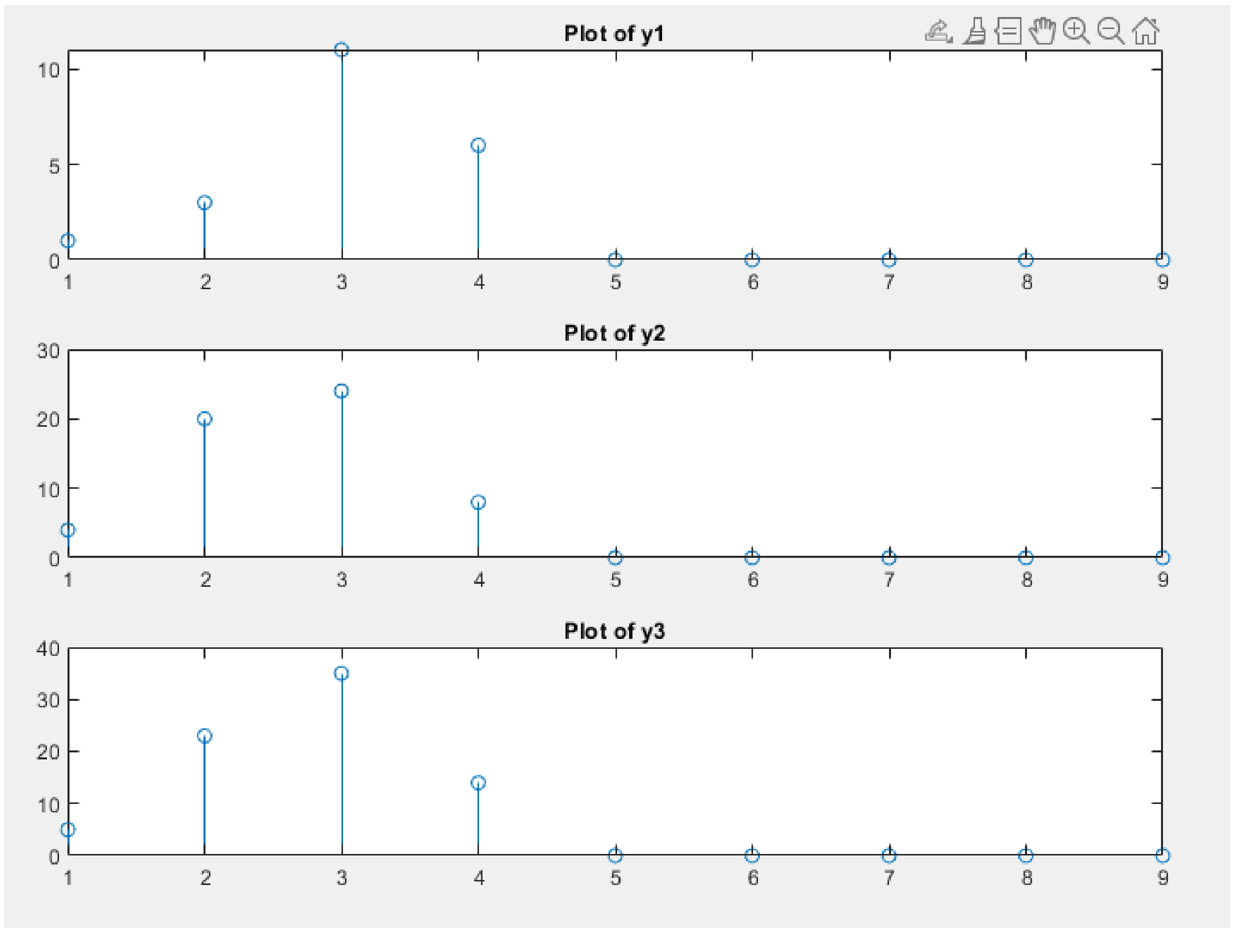
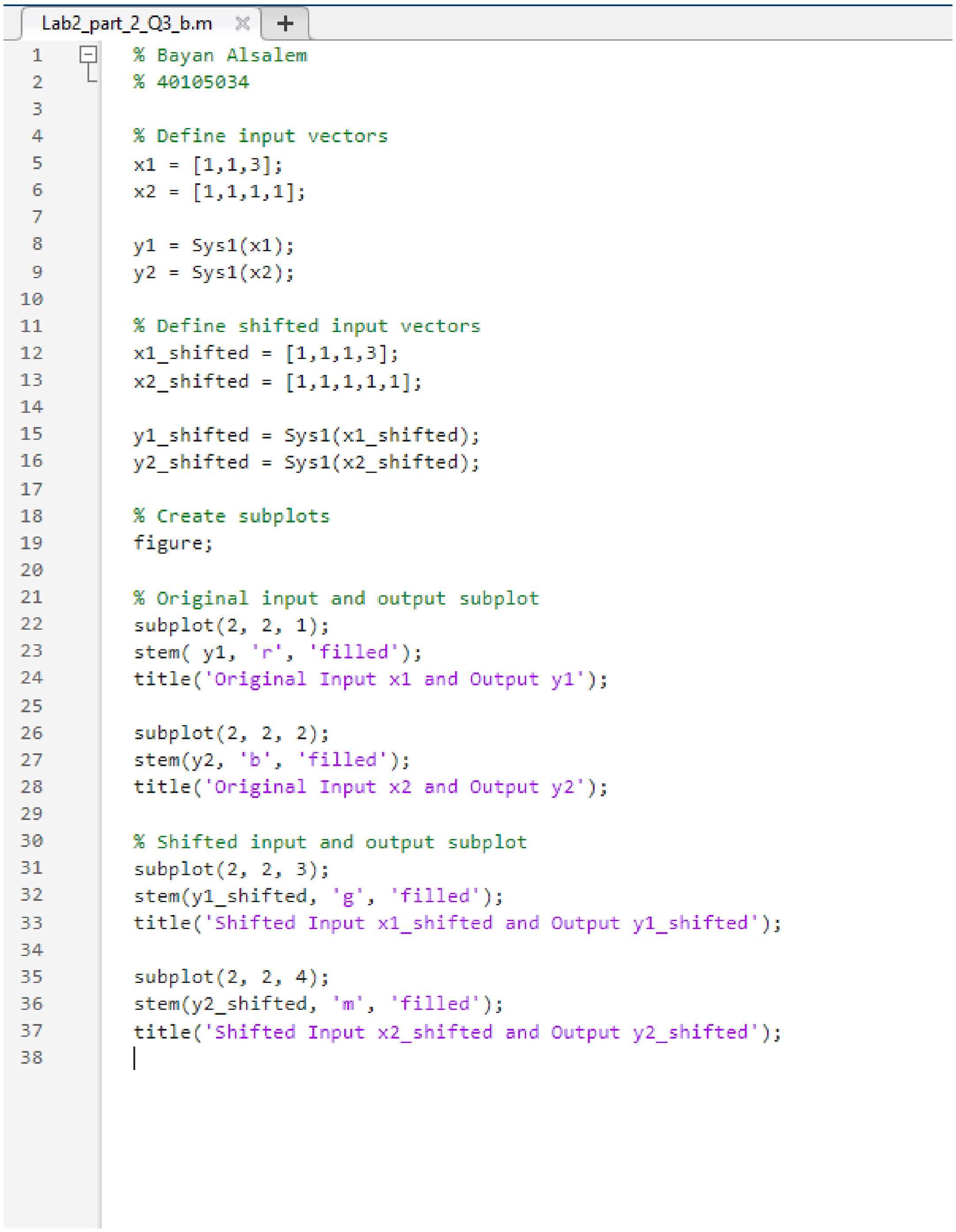


Figure 20: Plots of Part 2, Question 3, a



Lab2_part_2_Q3_b.m

```
% Bayan Alsalem
% 40105034

% Define input vectors
x1 = [1,1,3];
x2 = [1,1,1,1];

y1 = Sys1(x1);
y2 = Sys1(x2);

% Define shifted input vectors
x1_shifted = [1,1,1,3];
x2_shifted = [1,1,1,1,1];

y1_shifted = Sys1(x1_shifted);
y2_shifted = Sys1(x2_shifted);

% Create subplots
figure;

% Original input and output subplot
subplot(2, 2, 1);
stem( y1, 'r', 'filled');
title('Original Input x1 and Output y1');

subplot(2, 2, 2);
stem(y2, 'b', 'filled');
title('Original Input x2 and Output y2');

% Shifted input and output subplot
subplot(2, 2, 3);
stem(y1_shifted, 'g', 'filled');
title('Shifted Input x1_shifted and Output y1_shifted');

subplot(2, 2, 4);
stem(y2_shifted, 'm', 'filled');
title('Shifted Input x2_shifted and Output y2_shifted');
```

Figure 21: Plots of Part 2, Question 3, b

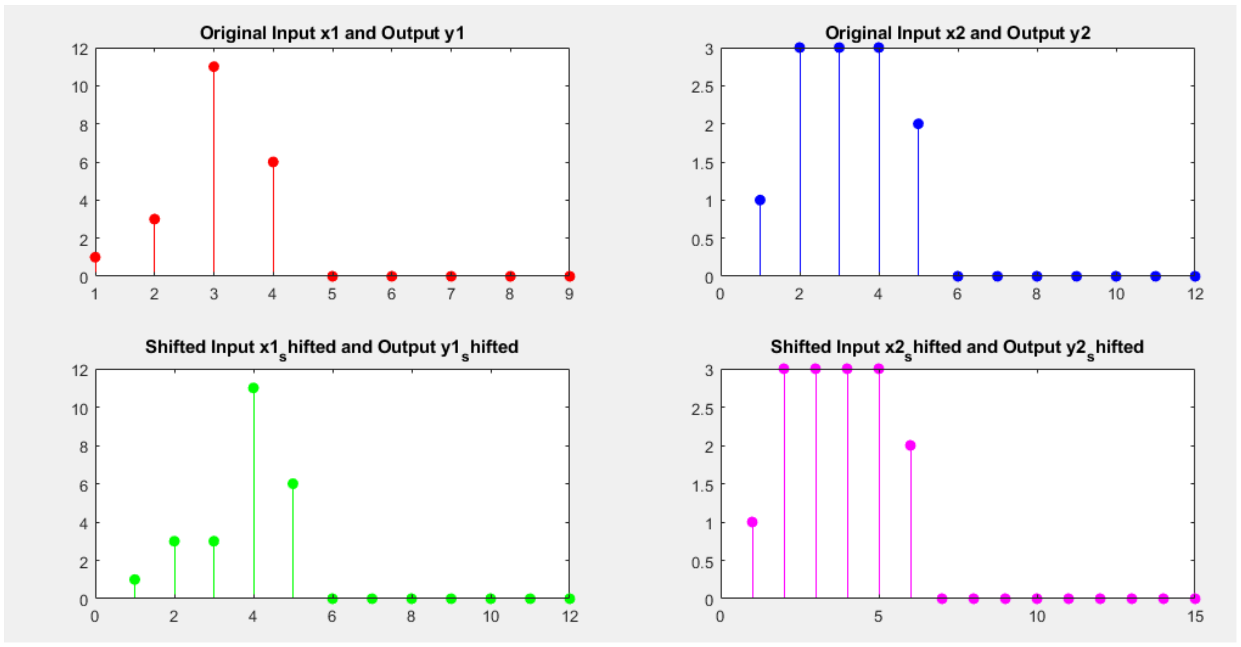


Figure 22: Plots of Part 2, Question 3, b