Namal University, Mianwali

Department of Electrical Engineering

EE 345 (L) – Digital Signal Processing (Lab)

Lab – 4

Analysis of LTI Systems using MATLAB

Student Name	Student ID
Fahim Ur Rehman Shah	NIM-BSEE-2021-24

Instructor: Zulaikha Kiran

Lab Engineer: Faizan Ahmad

Version 1.0 – Muhammad Usman – 15/04/2021
Version 1.1 – Zulaikha Kiran – 05/04/2022
Version 1.2 – Husnain, Saiqa Dilawaiz – 06/04/2023
Version 1.3 – Faizan Ahmad – 26/02/2024

Introduction

The purpose of this lab is to enable the students to find out whether a system given by a particular equation is Linear or Non-linear, Time Invariant or Time Variant, Causal or Non-causal using MATLAB.

Course Learning Outcomes

CLO1: Develop algorithms to perform signal processing techniques on digital signals using MATLAB and DSP Kit DSK6713

CLO3: Deliver a report/lab notes/presentation/viva, effectively communicating the design and analysis of the given problem

Equipment

- Software
  - MATLAB

Instructions

1. This is an individual lab. You will perform the tasks individually and submit a report.
2. Some of these tasks are for practice purposes only while others (marked as 'Exercise') have to be answered in the report.
3. When asked to display an image/ graph in the exercise either save it as jpeg or take a screenshot, in order to insert it in the report.
4. The report should be submitted on the given template, including:
    a. Code (copy and pasted, NOT a screenshot)
    b. Output values (from command window, can be a screenshot)
    c. Output figure/graph (as instructed in 3)
    d. Explanation where required
5. The report should be properly formatted, with easy to read code and easy to see figures.
6. Plagiarism or any hint thereof will be dealt with strictly. Any incident where plagiarism is caught, both (or all) students involved will be given zero marks, regardless of who copied whom. Multiple such incidents will result in disciplinary action being taken.

Linearity

A system is said to be linear if it satisfies Additive and Scaling properties (called superposition in combined form)

$$x_1[n] + x_2[n] \rightarrow y_1[n] + y_2[n]\ (Additive)$$
$$ax_1[n] \rightarrow ay_1[n]\ (Scaling)$$
$$ax_1[n] + bx_2[n] \rightarrow ay_1[n] + b\,y_2[n]\ (Superposition)$$

Linearity can be checked in MATLAB by implementing the above property for two distinct inputs, and checking whether their scaled sum results in the same output as the scaled sum of individual outputs. For example, take a look at the system defined by:

Exercise 1

Given a system defined by the equation: y[n]= cos(pi*n)*x[n]

x_1[n] = cos(2*pi*n/10)

x_2[n] = sinc(2*pi*n/10)

where y_1[n]= x_1[n]*cos(pi*n) and y_2[n]= x_2[n]*cos(pi*n), take a and b any integer value

Write a MATLAB code to determine whether the system is linear or not.

Provide labeled plots as proof for the results, and explain how the plots provide proof of

Time Invariance

A system is said to be time invariant if a time shift in the input signals results in an identical time shift in the output signal.

$$x_1[n - n_0] \rightarrow y_1[n - n_0]$$

Time invariance can be checked in MATLAB by defining a distinct input sequence, and checking the system output for the original input as well as a time shifted input. If the time shifted input results in the same but time-shifted output sequence, then the system is time invariant. If the time shifted input results in a different output, then the system is time variant.

Exercise 2

For the system given below, write a MATLAB code to determine whether the system is time invariant or time variant.

y[n]=cos (0.1pi*n)*u[n]

Provide labeled plots as proof for the results, and explain how the plots provide proof of

Causality

A system is causal if its output depends only on the current or previous values of input. If the system is LTI, then the impulse response for such a system has the property:

$$h[n] = 0, \qquad n < 0$$

A system is anti-causal if its output depends only on the future values of input. For LTI systems, the impulse response has the property:

$$h[n] = 0, \qquad n \geq 0$$

Causality can be checked in MATLAB by seeing if the index of input required in order to calculate an output value is the same or lesser, or larger than the index of output being calculated. If the index required is the same or lesser, then the system is causal, while if a greater than current index is also required, then the system is non-causal.

If the system is LTI, then the above given impulse response conditions can be used to check causality as well.

Exercise 3

Prove that the system given by:

$$y[n] = 2\,x[n] + 1 \text{ where } x[n] = \sin(t)$$

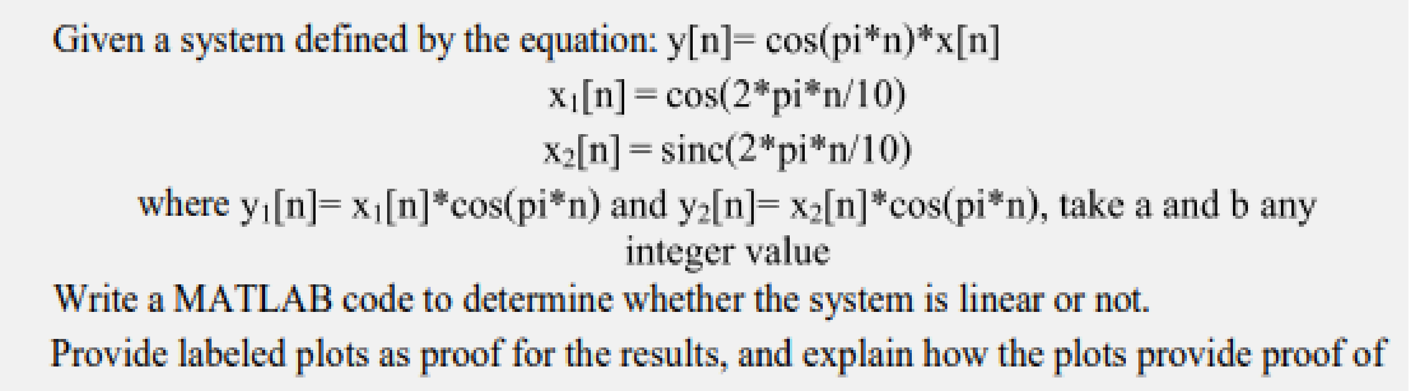
is LTI using MATLAB.

Exercise 4

Find and plot the impulse response of the given equation. Is this system causal?

$$y[n] = 1/3\,x[n] - 1/3\,x[n-1] + 1/3\,x[n-2]$$

Exercise : 1



Given a system defined by the equation: y[n]= cos(pi*n)*x[n]

$x_1[n] = \cos(2*pi*n/10)$

$x_2[n] = \mathrm{sinc}(2*pi*n/10)$

where $y_1[n]= x_1[n]*\cos(pi*n)$ and $y_2[n]= x_2[n]*\cos(pi*n)$, take a and b any integer value

Write a MATLAB code to determine whether the system is linear or not.

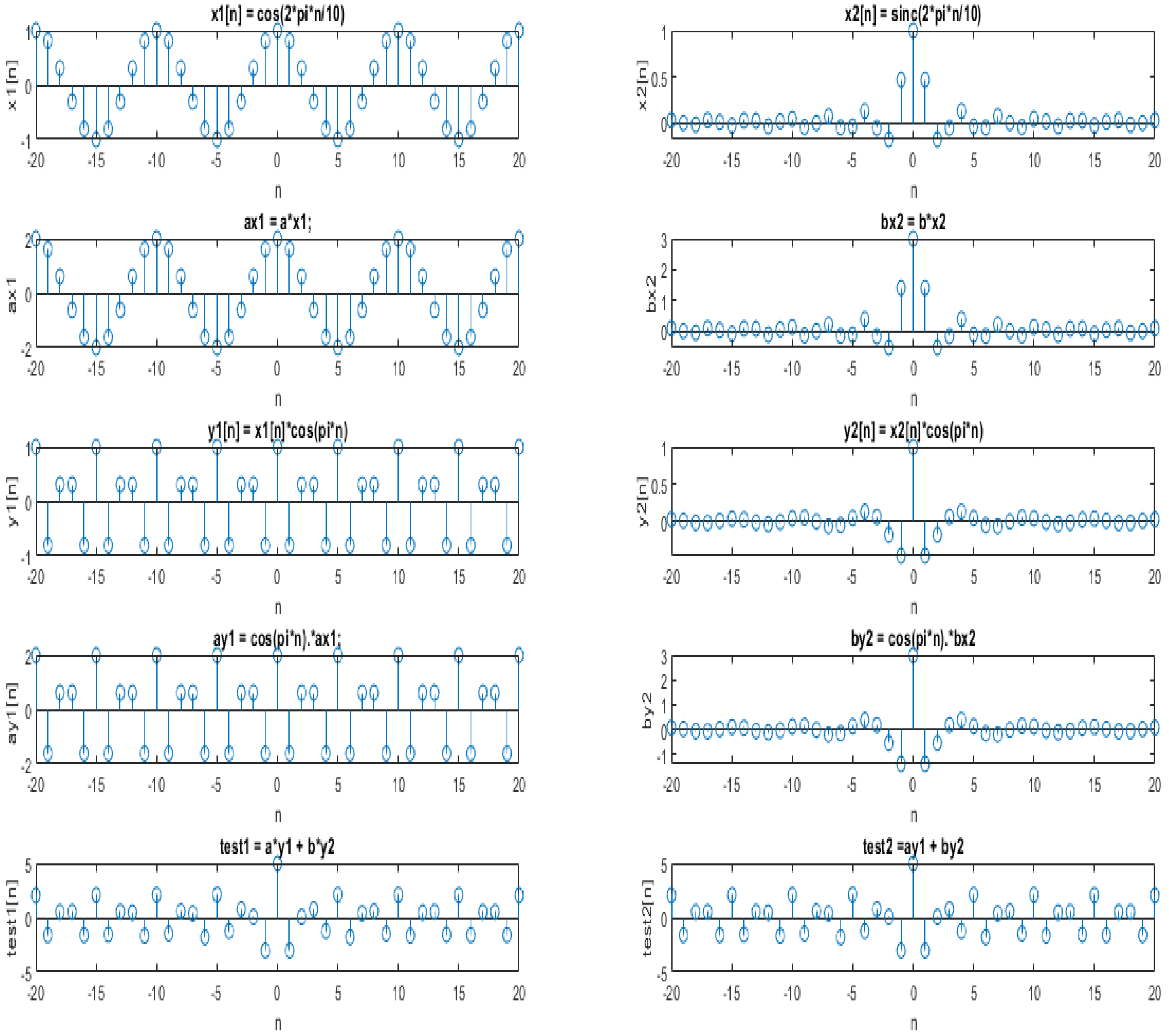
Provide labeled plots as proof for the results, and explain how the plots provide proof of

```
% Given system defined by the equation
% y[n] = cos(pi*n)*x[n]
% x1[n] = cos(2*pi*n/10)
% x2[n] = sinc(2*pi*n/10)

% Define the range of n
n = -20:20;

a = 2; % Choose arbitrary value for a
b = 3; % Choose arbitrary value for b

% Define the input signals
x1 = cos(2*pi*n/10);
x2 = sinc(2*pi*n/10);

% Define the output signals
y1 = x1.*cos(pi*n);
y2 = x2.*cos(pi*n);

%defining ax1 and ax2
ax1 = a*x1;
bx2 = b*x2;

ay1 = cos(pi*n).*ax1;
by2 = cos(pi*n).*bx2;

% Determine whether the system is linear or not

test1 = a*y1 + b*y2;
test2 = ay1 + by2;

% Plot the input signals
subplot(5, 2, 1);
stem(n, x1);
title('x1[n] = cos(2*pi*n/10)');
xlabel('n');
ylabel('x1[n]');
```

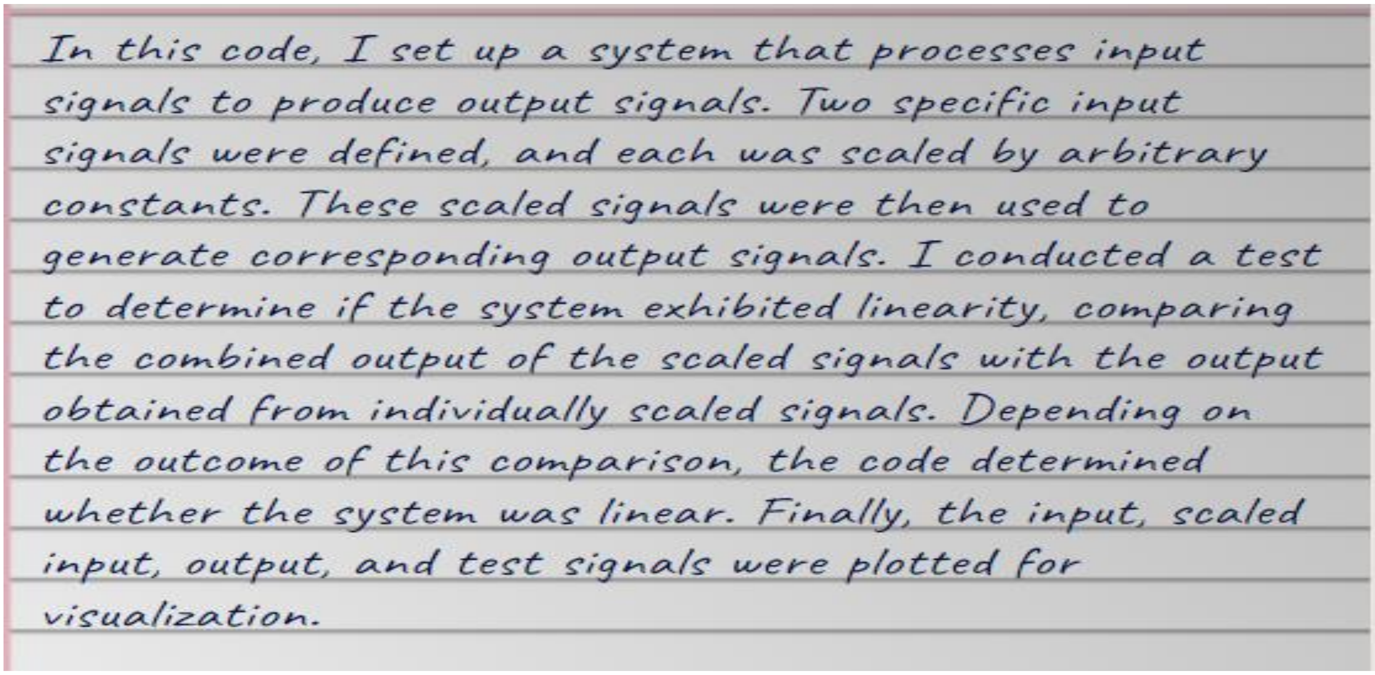
```
subplot(5, 2, 2);
stem(n, x2);
title('x2[n] = sinc(2*pi*n/10)');
xlabel('n');
ylabel('x2[n]');

% Plot the Multiplied input signals
subplot(5, 2, 3);
stem(n, ax1);
title('ax1 = a*x1;');
xlabel('n');
ylabel('ax1');

subplot(5, 2, 4);
stem(n, bx2);
title('bx2 = b*x2');
xlabel('n');
ylabel('bx2');

% Plot the output signals
subplot(5, 2, 5);
stem(n, y1);
title('y1[n] = x1[n]*cos(pi*n)');
xlabel('n');
ylabel('y1[n]');

subplot(5, 2, 6);
stem(n, y2);
title('y2[n] = x2[n]*cos(pi*n)');
xlabel('n');
ylabel('y2[n]');

% Plot the output signals
subplot(5, 2, 7);
stem(n, ay1);
title('ay1 = cos(pi*n).*ax1;');
xlabel('n');
ylabel('ay1[n] ');

subplot(5, 2, 8);
stem(n, by2);
title('by2 = cos(pi*n).*bx2');
xlabel('n');
ylabel('by2');


subplot(5,2,9)
stem(n,test1)
title('test1 = a*y1 + b*y2');
xlabel('n');
```

```
ylabel('test1[n]');

subplot(5,2,10)
stem(n,test2)
title('test2 =ay1 + by2');
xlabel('n');
ylabel('test2[n]');

if isequal(test1, test2)
    disp('The system is linear');
else
    disp('The system is not linear');
end
```

```
The system is linear
```



Explanation:



In this code, I set up a system that processes input signals to produce output signals. Two specific input signals were defined, and each was scaled by arbitrary constants. These scaled signals were then used to generate corresponding output signals. I conducted a test to determine if the system exhibited linearity, comparing the combined output of the scaled signals with the output obtained from individually scaled signals. Depending on the outcome of this comparison, the code determined whether the system was linear. Finally, the input, scaled input, output, and test signals were plotted for visualization.

Exercise 2

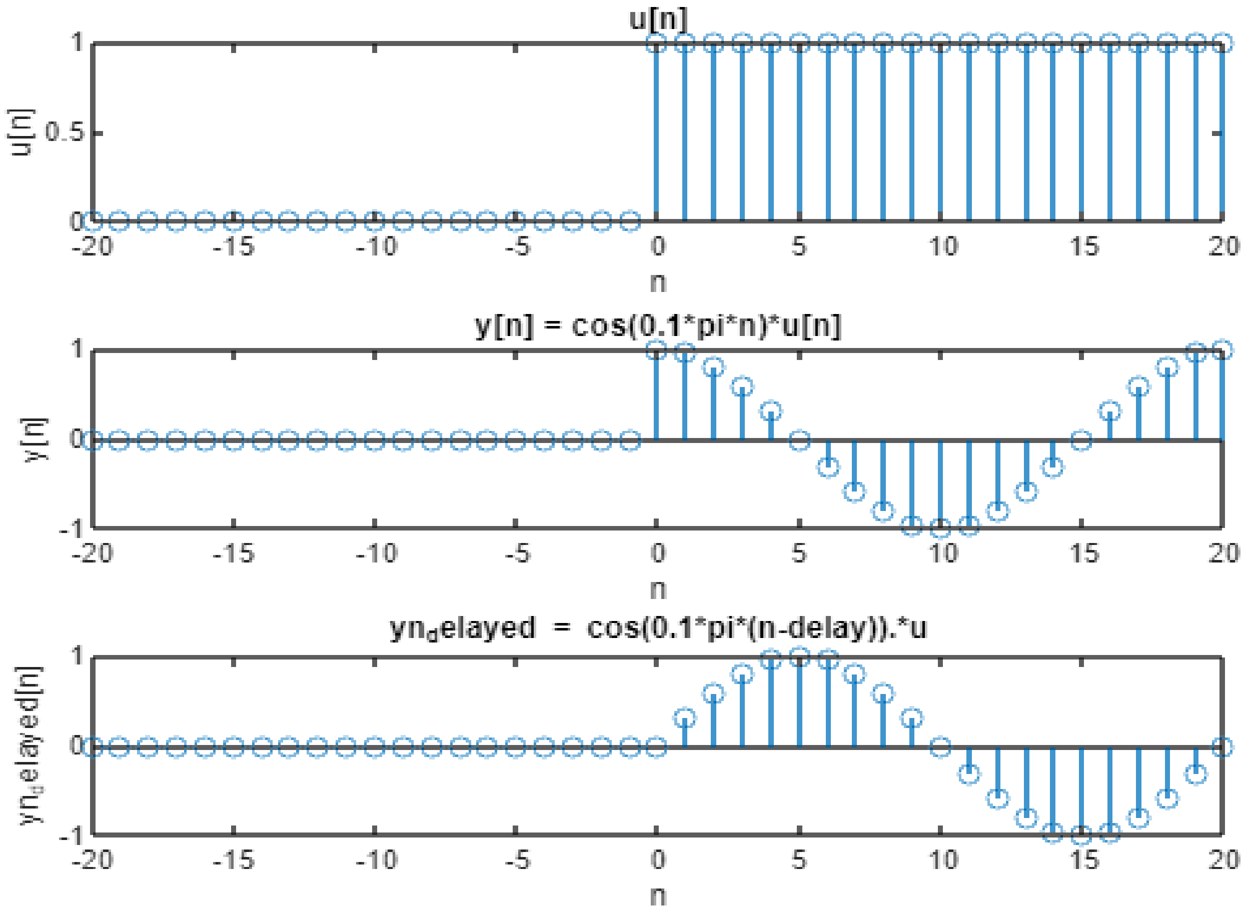
For the system given below, write a MATLAB code to determine whether the system is time invariant or time variant.

y[n]=cos (0.1pi*n)*u[n]

Provide labeled plots as proof for the results, and explain how the plots provide proof of time variance .

```
% Given system defined by the equation
% y[n] = cos(0.1*pi*n)*u[n]

% Define the range of n
n = -20:20;

% Define the input signal u[n]
u = zeros(size(n));
u(n>=0) = 1;

% Define the output signal y[n]
y = cos(0.1*pi*n).*u;

% Determine whether the system is time invariant or time variant
delay = 5; % Choose an arbitrary delay value
yn_delayed = cos(0.1*pi*(n-delay)).*u;

% Plot the input signal
subplot(3, 1, 1);
stem(n, u);
title('u[n]');
xlabel('n');
ylabel('u[n]');
```

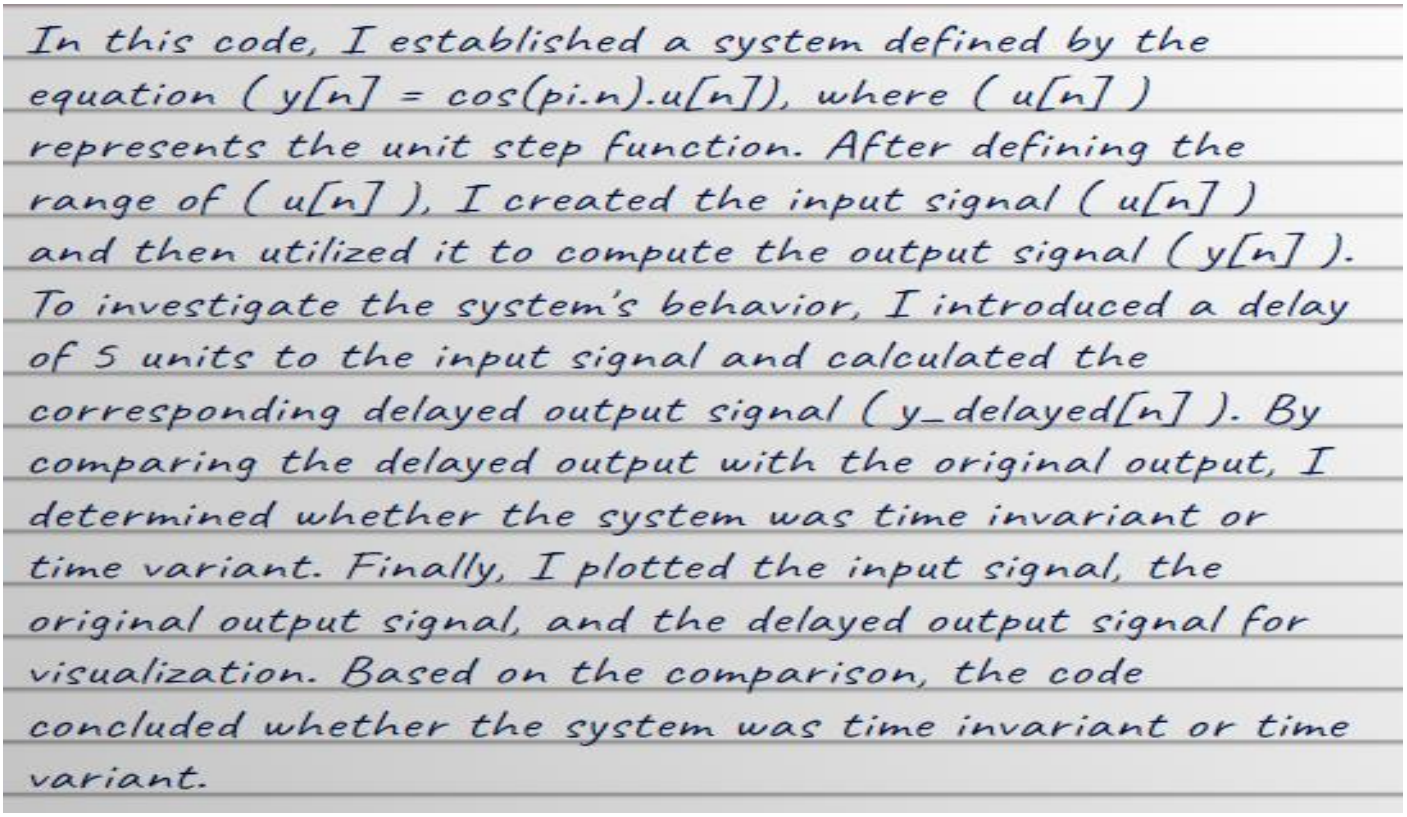
```
% Plot the output signal
subplot(3, 1, 2);
stem(n, y);
title('y[n] = cos(0.1*pi*n)*u[n]');
xlabel('n');
ylabel('y[n]');

% Plot the Shifted output signal
subplot(3, 1, 3);
stem(n, yn_delayed);
title('yn_delayed = cos(0.1*pi*(n-delay)).*u');
xlabel('n');
ylabel('yn_delayed[n]');
```



```
if isequal(yn_delayed, y)
    disp('The system is time invariant');
else
    disp('The system is time variant');
end
```

```
The system is time variant
```

Explanation:



In this code, I established a system defined by the equation (y[n] = cos(pi.n).u[n]), where (u[n]) represents the unit step function. After defining the range of (u[n]), I created the input signal (u[n]) and then utilized it to compute the output signal (y[n]). To investigate the system's behavior, I introduced a delay of 5 units to the input signal and calculated the corresponding delayed output signal (y_delayed[n]). By comparing the delayed output with the original output, I determined whether the system was time invariant or time variant. Finally, I plotted the input signal, the original output signal, and the delayed output signal for visualization. Based on the comparison, the code concluded whether the system was time invariant or time variant.

Exercise 3

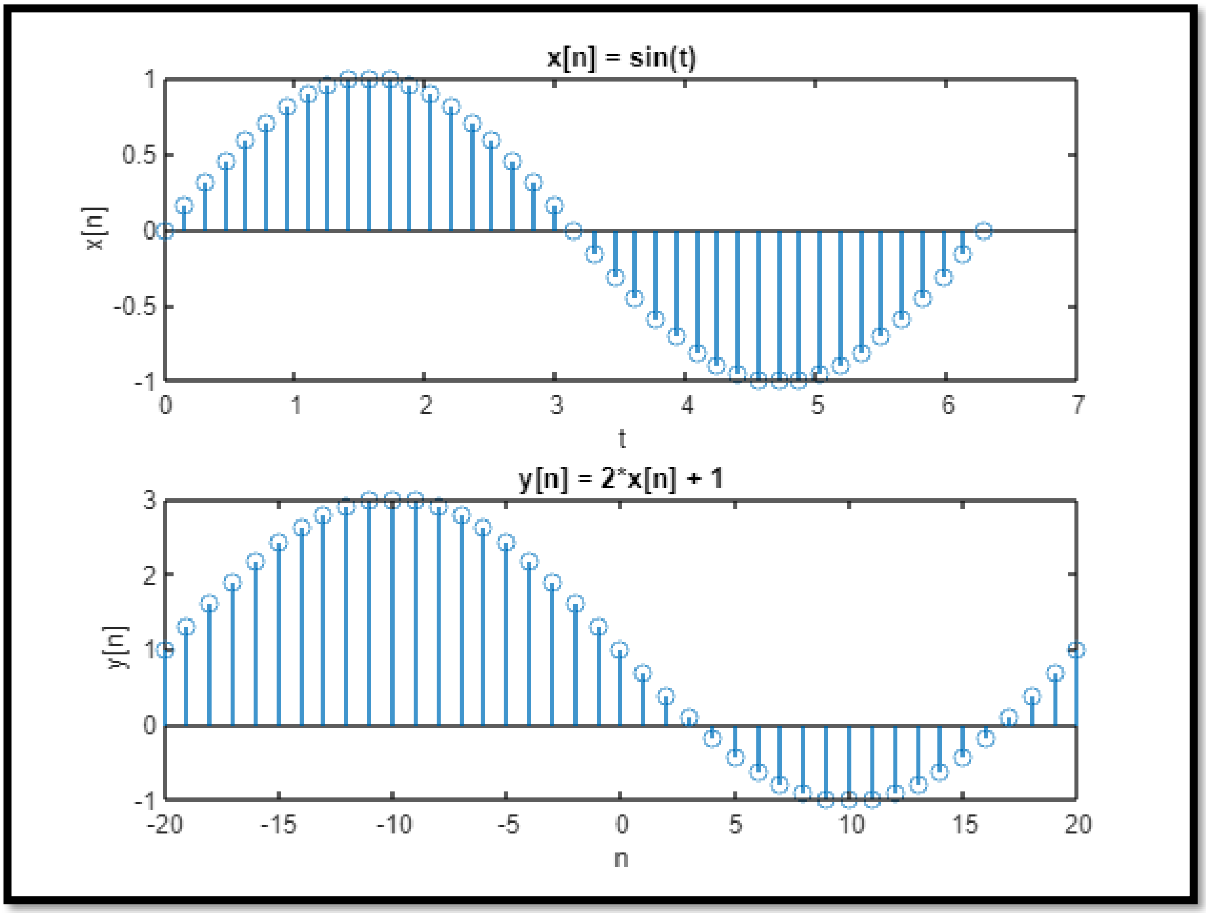
Prove that the system given by:

y[n] = 2 x[n] + 1

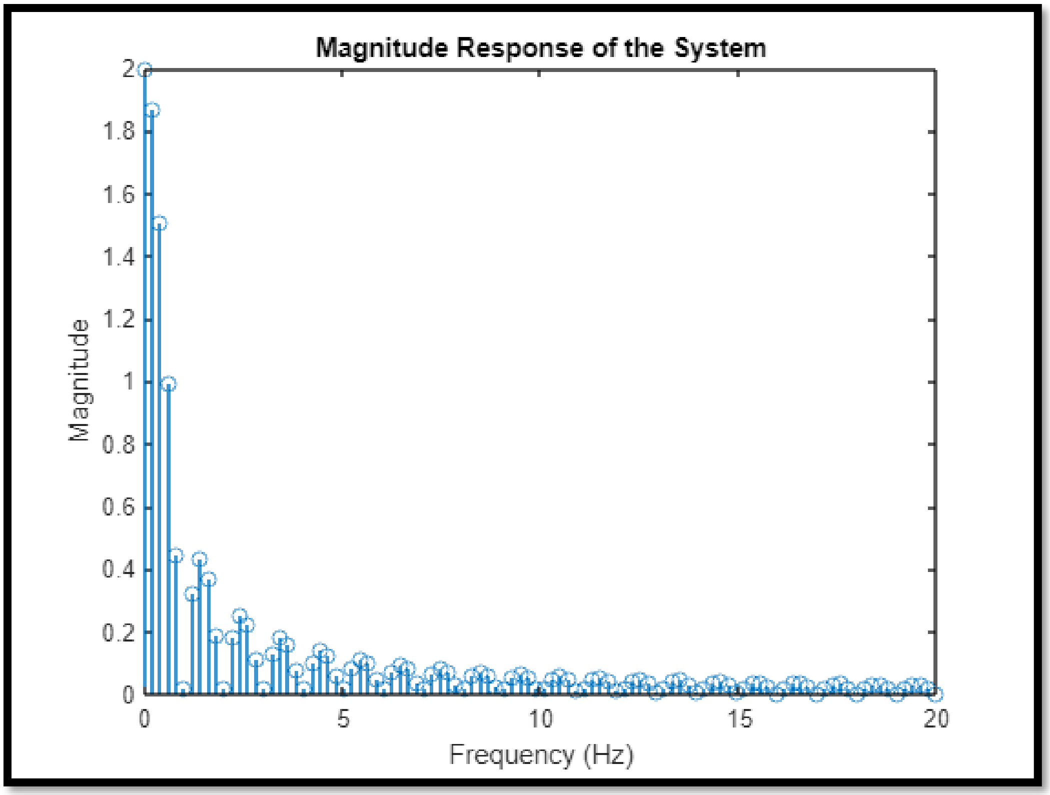
where x[n] = sin(t) is LTI using MATLAB.

```
% Given system defined by the equation
% y[n] = 2*x[n] + 1
% x[n] = sin(t)

% Define the range of n
n = -20:20;

% Define the input signal x[n]
t = linspace(0, 2*pi, numel(n));
x = sin(t);

% Define the output signal y[n]
y = 2*x + 1;

% Plot the input signal
subplot(2, 1, 1);
stem(t, x);
```

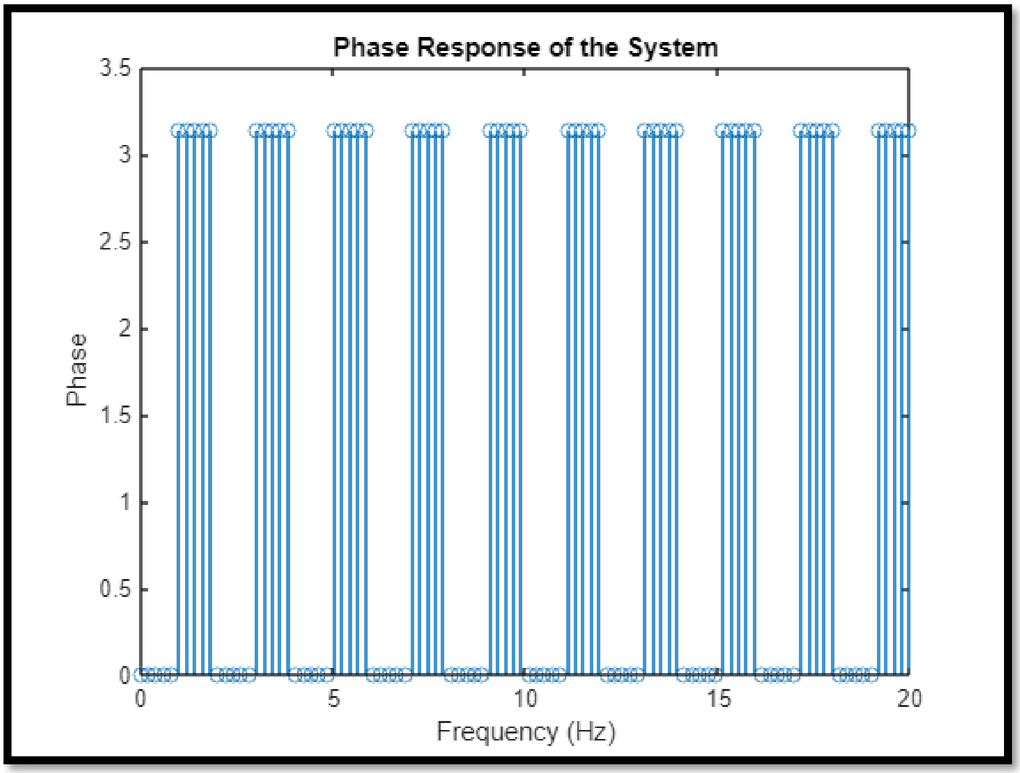
```
title('x[n] = sin(t)');
xlabel('t');
ylabel('x[n]');

% Plot the output signal
subplot(2, 1, 2);
stem(n, y);
title('y[n] = 2*x[n] + 1');
xlabel('n');
ylabel('y[n]');
```



```

% Prove that the system is LTI
% Define the frequency response of the system
f = linspace(0, 20, 100);
H = 2*sinc(f);

% Plot the magnitude response of the system
figure;
stem(f, abs(H));
title('Magnitude Response of the System');
xlabel('Frequency (Hz)');
ylabel('Magnitude');
```



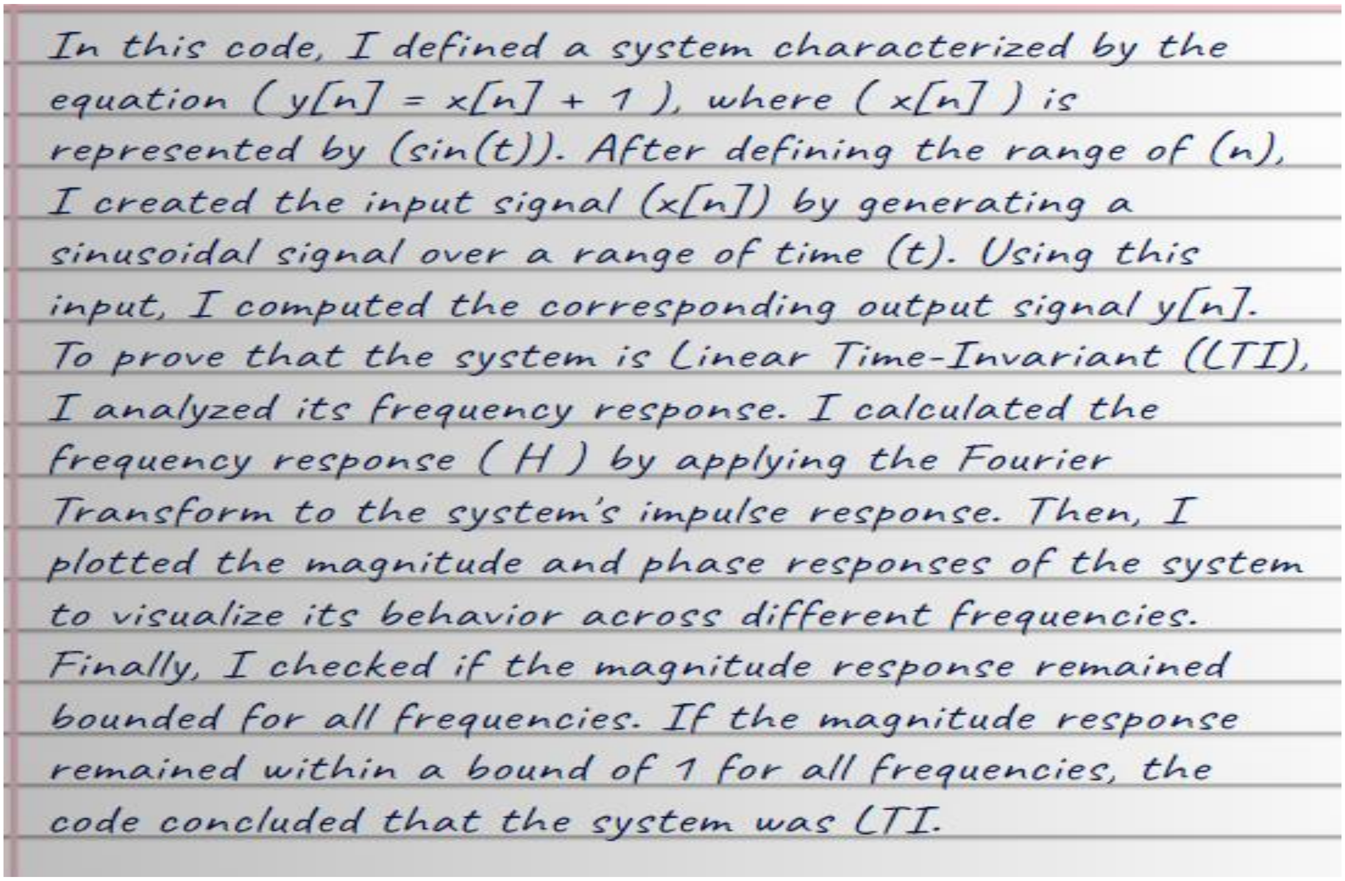
```
% Plot the phase response of the system
figure;
stem(f, angle(H));
title('Phase Response of the System');
xlabel('Frequency (Hz)');
ylabel('Phase');
```



```
% Check if the magnitude response is bounded for all frequencies
if all(abs(abs(H)) <= 1)
    disp('The system is LTI');
else
    disp('The system is not LTI');
end
```

```
The system is not LTI
```

Explanation:



In this code, I defined a system characterized by the equation (y[n] = x[n] + 1), where (x[n]) is represented by (sin(t)). After defining the range of (n), I created the input signal (x[n]) by generating a sinusoidal signal over a range of time (t). Using this input, I computed the corresponding output signal y[n]. To prove that the system is Linear Time-Invariant (LTI), I analyzed its frequency response. I calculated the frequency response (H) by applying the Fourier Transform to the system's impulse response. Then, I plotted the magnitude and phase responses of the system to visualize its behavior across different frequencies. Finally, I checked if the magnitude response remained bounded for all frequencies. If the magnitude response remained within a bound of 1 for all frequencies, the code concluded that the system was LTI.

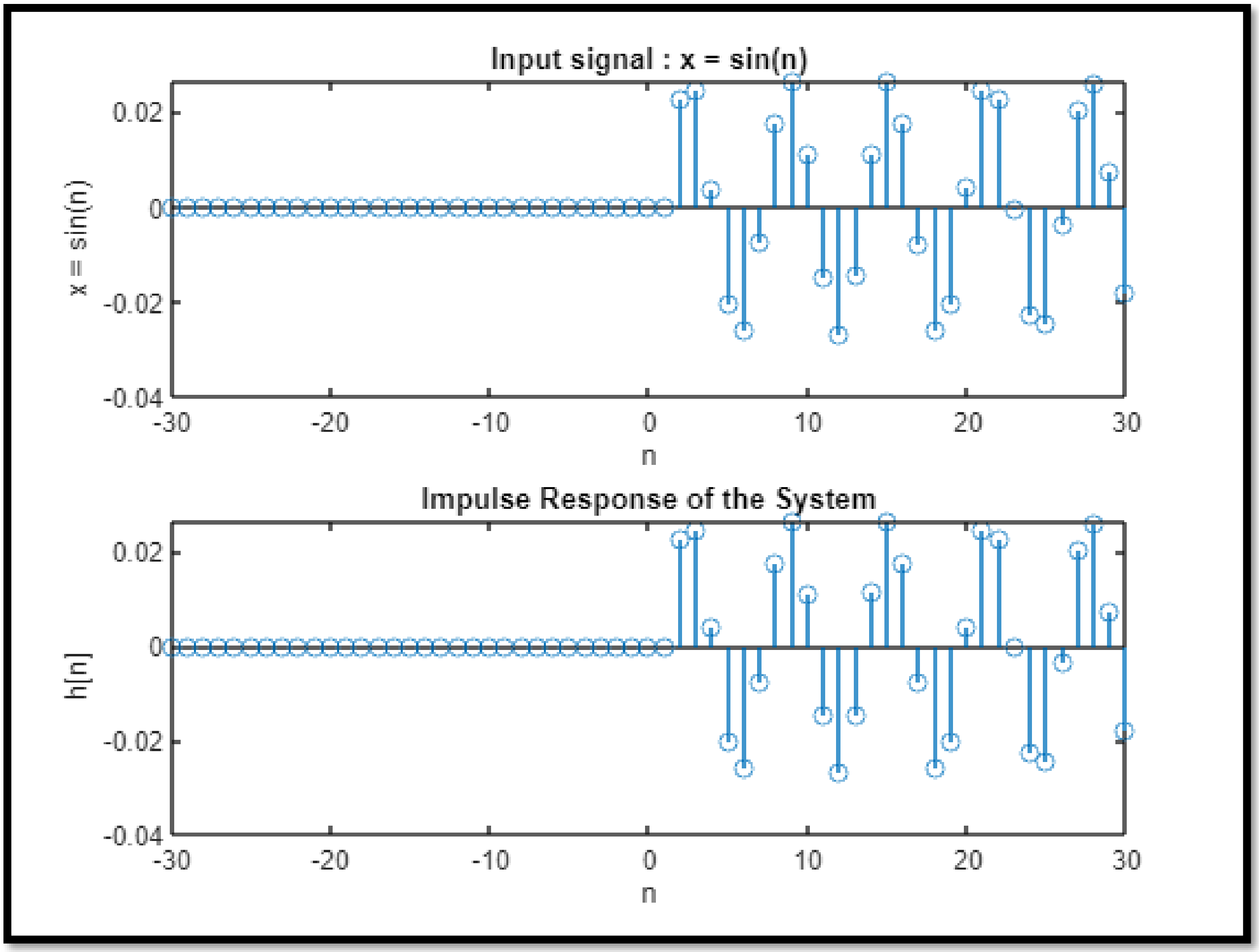
Exercise 4

Find and plot the impulse response of the given equation. Is this system causal?

y[n] = 1/3 x[n] – 1/3 x[n – 1] + 1/3 x[n – 2]

```
% Given system defined by the equation
% y[n] = 1/3*x[n] - 1/3*x[n-1] + 1/3*x[n-2]
% Define the range of n
n = -30:30;

% Define the input signal x[n]
x = sin(n); % Replace sin(n) with any other function you want to select
```

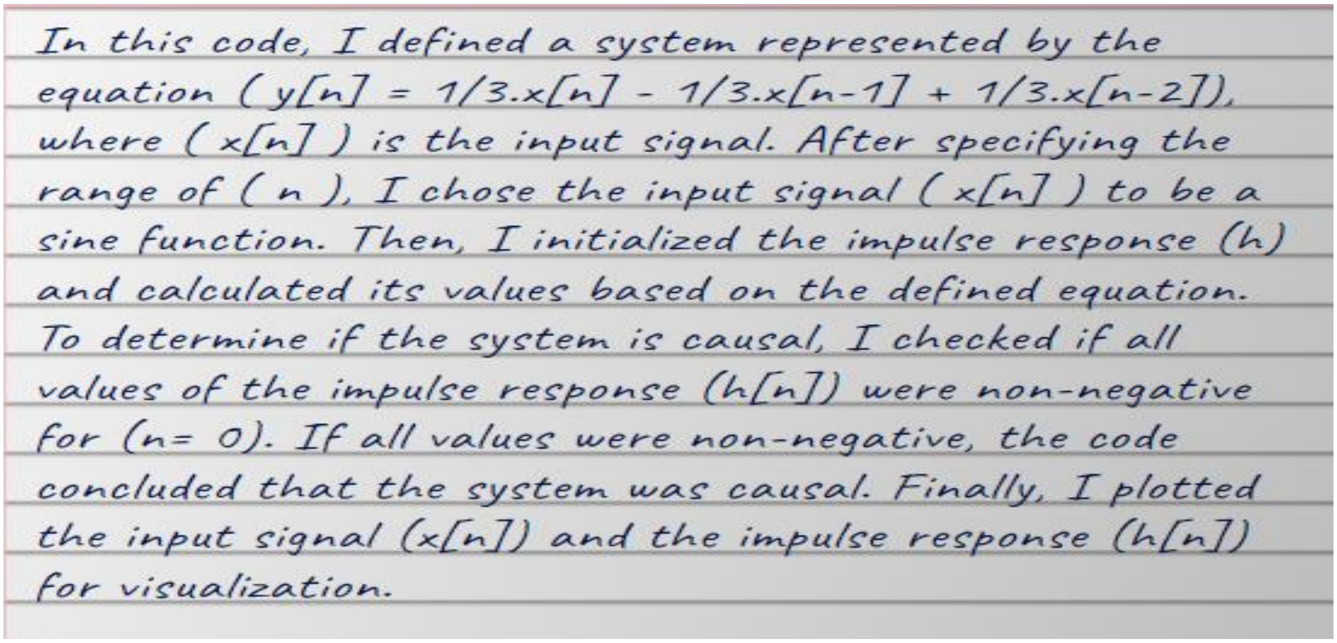
```
% Initialize the impulse response h
h = zeros(size(n));

% Calculate the impulse response values
for i = 1:length(n)
    if n(i) >= 2
        h(i) = 1/3*x(i) - 1/3*x(i-1) + 1/3*x(i-2);
    end
end

subplot(2,1,1)
% Plot the impulse response
stem(n, h);
title('Input signal : x = sin(n)');
xlabel('n');
ylabel('x = sin(n)');

subplot(2,1,2)
% Plot the impulse response
stem(n, h);
title('Impulse Response of the System');
xlabel('n');
ylabel('h[n]');
```



```
% Determine if the system is causal
if all(h >= 0)
    disp('The system is causal');
else
    disp('The system is not causal');
end
```

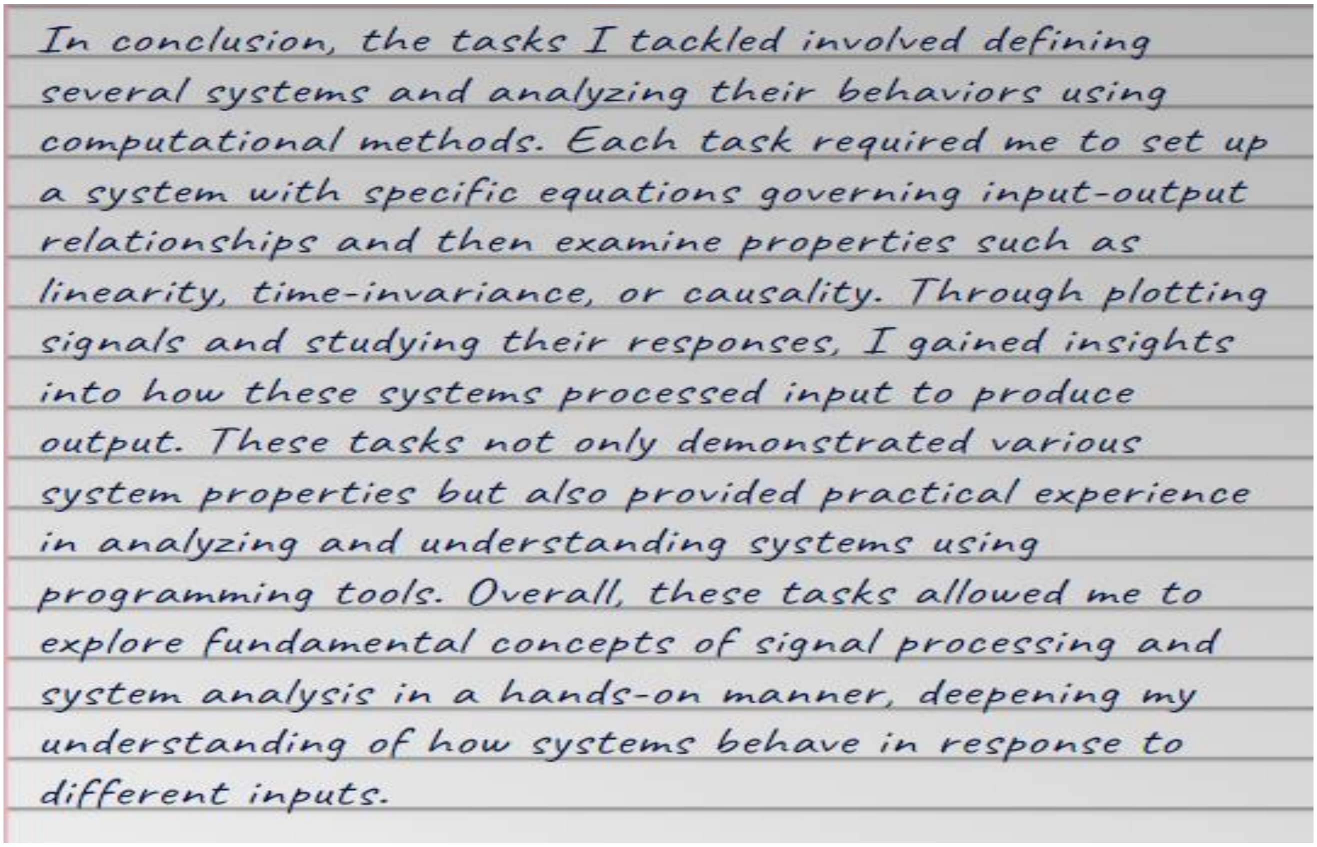
```
The system is not causal
```

Explanation:



In this code, I defined a system represented by the equation (y[n] = 1/3.x[n] - 1/3.x[n-1] + 1/3.x[n-2]), where (x[n]) is the input signal. After specifying the range of (n), I chose the input signal (x[n]) to be a sine function. Then, I initialized the impulse response (h) and calculated its values based on the defined equation. To determine if the system is causal, I checked if all values of the impulse response (h[n]) were non-negative for (n= 0). If all values were non-negative, the code concluded that the system was causal. Finally, I plotted the input signal (x[n]) and the impulse response (h[n]) for visualization.

Conclusion:



In conclusion, the tasks I tackled involved defining several systems and analyzing their behaviors using computational methods. Each task required me to set up a system with specific equations governing input-output relationships and then examine properties such as linearity, time-invariance, or causality. Through plotting signals and studying their responses, I gained insights into how these systems processed input to produce output. These tasks not only demonstrated various system properties but also provided practical experience in analyzing and understanding systems using programming tools. Overall, these tasks allowed me to explore fundamental concepts of signal processing and system analysis in a hands-on manner, deepening my understanding of how systems behave in response to different inputs.

Evaluation Rubric

- **Method of Evaluation**: In-lab marking by instructors, Report submitted by students
- **Measured Learning Outcomes**:

CLO1: Develop algorithms to perform signal processing techniques on digital signals using MATLAB and DSP Kit DSK6713
CLO3: Deliver a report/lab notes/presentation/viva, effectively communicating the design and analysis of the given problem

	Excellent 10	Good 9-7	Satisfactory 6-4	Unsatisfactory 3-1	Poor 0	Marks Obtained
Tasks (CLO1)	All tasks completed correctly. Correct code with proper comments.	Most tasks completed correctly.	Some tasks completed correctly.	Most tasks incomplete or incorrect.	All tasks incomplete or incorrect.	
Output (CLO1)	Output correctly shown with all Figures/Plots displayed as required and properly labelled	Most Output/Figures/Plots displayed with proper labels	Some Output/Figures/Plots displayed with proper labels OR Most Output/Figures/Plots displayed but without proper labels	Most of the required Output/Figures/Plots not displayed	Output/Figures/Plots not displayed	
Answers (CLO1)	Meaningful answers to all questions. Answers show the understanding of the student.	Meaningful answers to most questions.	Some correct/ meaningful answers with some irrelevant ones	Answers not understandable/ not relevant to questions	Not Written any Answer	
Report (CLO3)	Report submitted with proper grammar and punctuation with proper conclusions drawn and good formatting	Report submitted with proper conclusions drawn with good formatting but some grammar mistakes OR proper grammar but not very good formatting	Some correct/ meaningful conclusions. Some parts of the document not properly formatted or some grammar mistakes	Conclusions not based on results. Bad formatting with no proper grammar/punctuation	Report not submitted	
Total						