

①

NAME - SANTOSH (CB.EN.U4 CCE 20053)

DEPARTMENT - COMPUTER AND COMMUNICATION ENGINEERING (CCE) A

LAB TITLE AND CODE : SIGNAL PROCESSING LAB IACCCE281

EXPERIMENT NUMBER : 4

DATE : 05/10/2021

PROPERTIES OF SYSTEM

* AIM :

To determine whether a given Input sequence is a Linear Time-Invariant (LTI) system or not.

* SOFTWARE REQUIRED :

Spyder IDE (Anaconda 3) - Python 3.9.1 (64-bit)

* THEORY : (LINEARITY)

The principle of linearity is equivalent to the principle of superposition, i.e., a system can be said to be linear if, for any two input signals, their linear combination yields as output the same linear combination of the corresponding output signals.

→ Definition - Given any two inputs, acted on by the system as follows:

$$x_1(t) \rightarrow S \rightarrow y_1(s); \quad x_2 \rightarrow S \rightarrow y_2(s)$$

The system is said to be linear if, for any two constants α and β , and the signal: $x_3(t) = \alpha x_1(t) + \beta x_2(t)$.

The system acts on it as follows:

$$x_3(t) \rightarrow S \rightarrow y_3(s), \text{ where } y_3(s) = \alpha y_1(s) + \beta y_2(s)$$

It is not necessary for the input and output signals to have the same independent variable for linearity to make sense. The definition for systems with input and/or output signal being discrete-time is similar.

②

$$f(t) \xrightarrow{\otimes} \boxed{H} \rightarrow y(t) \equiv f(t) \xrightarrow{\otimes} \boxed{H} \rightarrow \otimes \rightarrow y(t)$$

\uparrow
 K

\uparrow
 K

$H(Kf(t)) = KH(f(t)) \rightarrow 1^{\text{st}}$ step where K is the scaling factor

A block diagram demonstrating the scaling property of linearity.

$$\begin{array}{c} f_1 \xrightarrow{\otimes} \boxed{H} \rightarrow y \\ f_2 \xrightarrow{\otimes} \boxed{H} \rightarrow y \\ \hline f_1 + f_2 \xrightarrow{\otimes} \boxed{H} \rightarrow y \end{array}$$

$$H(f_1(t) + f_2(t)) = H(f_1(t)) + H(f_2(t))$$

A block diagram demonstrating the superposition property of linearity.

$$H(\kappa_1 f_1(t) + \kappa_2 f_2(t)) = \kappa_1 H(f_1(t)) + \kappa_2 H(f_2(t))$$

→ Example -

A capacitor, an inductor, a resistor or any combination of these are all linear systems. If we consider the voltage applied across them as an input signal, and the current through them as an output signal. This is because these simple passive circuit components follow the principle of superposition within the ranges of operation.

* THEORY: (TIME INVARIANCE)

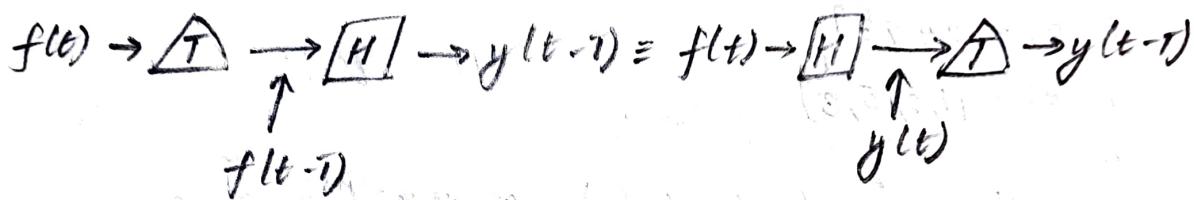
In a time-invariant system, there are no changes in system structure as a function of a time t . A system is time-invariant if -

$$\frac{x(t)}{x(t-t_0)} \rightarrow \boxed{\text{SYSTEM}} \xrightarrow{\quad} \frac{y(t)}{y(t-t_0)}$$

③

From the diagram, we can observe that for any time t_0 and any input $x(t)$ that produces response $y(t)$, the response to the time-shifted input $x(t-t_0)$ is equal to the time-shifted output $y(t-t_0)$ of $y(t)$.

Therefore, in a time-invariant system, the response to a left or right shift of the input $x(t)$ is equal to a corresponding left or right shift in the response $y(t)$.



This block diagram shows the condition for time invariance. The output is the same whether the delay is put on the input or the output.

→ Example -

An electric circuit with fixed values of Resistance R , Inductance L , and capacitance C can be considered as time invariant.

→ GRAPH PLOTTING ALGORITHM :

The following steps were followed -

- ① Define the x -axis and corresponding y -axis as lists.
- ② Plot them on canvas using `plot()` function.
- ③ Give a name to x -axis and y -axis using `xlabel()` and `ylabel()` functions.
- ④ Give a title to your plot using the `title()` function.
- ⑤ Finally, to view your plot, we use the `.show()` function.

PTD

(A)

+ PROGRAM WITH COMMENTS:

```
1 # Import library source files
2 import numpy as np
3 import matplotlib.pyplot as plt
4
5 n = np.arange(0, 4) # Get evenly spaced values within the
6 Interval [0, 4]
7 # Given Input sequences:
8 x1 = [2, 4, 2, 3]
9 x2 = [1, 2, 5, 3]
10
11 def scaling (scale, choice): # Check Homogeneity condition
12     before_scaling = []
13     after_scaling = []
14
15     if choice == str(1): # calculate for x1 = [2, 4, 2, 3]
16
17         for sample in range(len(n)):
18             temp = sample + x1[sample]
19             before_scaling.append (int(scale) * temp) # Scaling the
20             output
21
22             for sample in range(len(n)):
23                 temp = int(scale) + x1[sample]
24                 after_scaling.append (sample * temp) # sending output
25                 into system
26
27     else: # calculate for x2 = [1, 2, 5, 3]
28
29         for sample in range(len(n)):
30             temp = sample + x2[sample]
31             before_scaling.append (int(scale) * temp) # scaling the
32             output
```

⑤

```
30     for sample in range (len(n)):  
31         temp = float (scale) * a2 [sample]  
32         after_scaling.append (sample + temp) # sending  
33         output %to system  
34     plt.subplot (3, 1, 1) # subplot (rows, columns, index)  
describes the figure layout  
35     plt.xlabel ('n')  
36     plt.ylabel ('y' + str (choice) + '[n]')  
37     plt.title ('Before scaling x' + str (choice) + 'z'.format  
(before_scaling)) # Formats the specified values and insert  
the string's placeholder.  
38     plt.stem (n, before_scaling) # stem plot plots vertical  
lines at each x position covered under the graph from the  
baseline to y, and places a marker there.  
39  
40     plt.subplot (3, 1, 3) # subplot (rows, columns, index) describes  
the figure layout  
41     plt.xlabel ('n')  
42     plt.ylabel ('y' + str (choice) + '[n]')  
43     plt.title ('After scaling x' + str (choice) + 'z'.format  
(after_scaling)) # Formats the specified values and insert  
them inside the string's placeholder.  
44     plt.stem (n, after_scaling) # stem plot plots vertical  
lines at each x position covered under the graph from the  
baseline to y, and places a marker there.  
45  
46     plt.show () # To view both the sub-plots  
47  
48     if (before_scaling == after_scaling): # Condition to check if  
both the systems are equal or not after scaling  
49     print ("The given system satisfies Homogeneity condition.")  
50     return (1)
```

⑥

```

51 else:
52     print("The given system does not satisfy Homogeneity")
      Homogeneity condition.')
53     return (0)
54
55 def superposition (): # check additivity condition
56     before_adding = []; after_adding = []; y1 = []; y2 = []; x = []
57
58     for sample in range (len(n)):
59         y1.append (sample + x1[sample])
60         y2.append (sample + x2[sample])
61         before_adding.append (y1[sample] + y2[sample])
# Processing System Individually
62
63     plt.subplot (3,1,1) # subplot (rows, columns, index) describes
      the figure layout
64     plt.xlabel ('n')
65     plt.ylabel ('y[n]')
66     plt.title ('H(x1(t)) + H(x2(t))')
67     plt.stem (n, before_adding)
68
69     for sample in range (len(n)):
70         x.append (x1[sample] + x2[sample])
71         after_adding.append (sample + x[sample])
# Processing system Together
72
73     plt.subplot (3,1,3) # subplot (rows, columns, index)
      describes the figure layout
74     plt.xlabel ('n')
75     plt.ylabel ('y[n]')
76     plt.title ('H(x1(t)) + H(x2(t))')
77     plt.stem (n, after_adding) # stem plot plots vertical
      lines at each a position covered under the graph from the
      baseline to y, and places a marker there.

```

⑦

```

78
79 plt.show() # To view both the sub-plots
80
81 print('In 2. Superposition condition :- ')
82 if (before_adding == after_adding): # Condition to check
83     if both the systems are equal or not after adding
84         print("The given system satisfies Superposition condition.")
85     return (1)
86 else:
87     print("The given system does not satisfy Superposition
88 condition.")
89     return (0)
90
91
92 def time (shift, choice): # check Time Variance
93     Input_delay = []; output_delay = []
94
95     if choice == str(1): # calculate for x1= [2, 4, 2, 3]
96
97         # Output Delay
98         for sample in range (len(x1)):
99             output_delay.append (sample + all_sample) # sending
100            output into system
101
102         for sample in range (int(shift)):
103             output_delay.append (0) # adds extra sample at the end
104            for shifting
105
106         for sample in range (int(shift)):
107             for sample in range (len(output_delay)-1, 0, -1):
108                 output_delay[sample] = output_delay[sample-1]
109             # shift the sample

```

```

105     for sample in range (int (shift)):
106         output-delay [sample] = 0 # Assign zero to the empty
elements for the shifted bit
107
108     # Input Delay
109     for sample in range (int (shift)):
110         x1.append (0) # Adds extra sample at the end for
shifting
111
112     for sample in range (int (shift)):
113         for sample in range (len(x1)-1, 0, -1):
114             x1 [sample] = x1 [sample-1] # shift the sample
115
116     for sample in range (int (shift)):
117         x1 [sample] = 0 # Assign zero to the empty elements
for the shifted bit
118
119     for sample in range (len (x1)):
120         Input-delay.append (sample + x1 [sample]) # sending
input into system
121
122     else: # calculate for x2-[1, 2, 5, 3]
123
124     # Output Delay
125     for sample in range (len (x2)):
126         output-delay.append (sample + x2 [sample]) # sending
output into system
127
128     for sample in range (int (shift)):
129         output-delay.append (0) # Adds extra sample at
the end for shifting
130
131

```

```

131     for sample in range (int (shift)):
132         for sample in range (len (output_delay)-1, 0, -1):
133             output_delay [sample] = output_delay [sample-1]
134
135     # shift the output
136
137     for sample in range (int (shift)):
138         output_delay [sample] = 0 # Assign zero to the empty
139         elements for the shifted bit
140
141     # Input Delay
142     for sample in range (int (shift)):
143         x2.append (0) # Adds extra sample at the end for shifting
144
145     for sample in range (int (shift)):
146         for sample in range (len (x2)-1, 0, -1):
147             x2 [sample] = x2 [sample-1] # shift the sample
148
149     for sample in range (int (shift)):
150         x2 [sample] = 0 # Assign zero to the empty elements for
the shifted bit
151
152     plt.subplot (3,1,1) # subplot (rows, columns, index) describes
the figure layout
153     plt.xlabel ('t')
154     plt.ylabel ('y(t)')
155     plt.title ('shifting input by ' + str (shift) + ' for a ' + str (choice)
+ ' & ' + format (input_delay)) # Formats the specified value(s)
and insert them inside the string's placeholder.

```

10

```

156 plt.stem(np.arange(len(input-delay)), input-delay) # stem
plot plots vertical lines at each n position covered under the
graph from the baseline to y, and places a marker there.
157
158 plt.subplot(3,1,3) # subplot (rows, columns, index) describes
the figure layout
159 plt.xlabel('t')
160 plt.ylabel('y(t)')
161 plt.title('shifting output by ' + str(shift) + ' for x' +
str(choice) + '3'.format(output-delay)) # Formats the
specified values and insert them inside the string's
placeholder.
162 plt.stem(np.arange(len(output-delay)), output-delay) # stem
plot plots vertical lines at each n position covered under the
graph from the baseline to y, and places a marker there.
163
164 plt.show() # To view both the sub-plots
165
166 if (input-delay == output-delay): # Condition to check if
both the system are equal or not after shifting
167     print("Since both the outputs are same, the given
system is time invariant.")
168     second-condition = 1
169 else:
170     print("Since both the outputs are not same, the given
system is time variant.")
171     second-condition = 0
172
173 return(second-condition)
174
175 # Main
176 while True: # This simulates a Do Loop

```

(11)

177 choice
~~choice~~ = Input()
 178 "MENU: Determine whether the system $y[n] = nx[n]$ is
 linear and time invariant. In 1. $a1 = [2, 4, 1, 3]$ In 2. $a2 = [1, 3, 5, 3]$
 In 3. Exit In Enter the number corresponding to the menu to
 Implement the Input sequence : ") # Menu Driven Implementation
 179
 180 If choice == str(1) or choice == str(2):
 181
 182 scale = input ("A linear system is any system that
 obeys the properties of scaling (homogeneity) and superposition
 (additivity). In 1. Homogeneity Condition: - In Enter the
 value for scaling factor: ")
 183
 184 if scaling (scale, choice) == superposition (1): # Condition
 to check if both the systems are linear or not
 185 print ("In Since both the properties are satisfied; the
 given system is linear.")
 186 first_condition = 1
 187 else:
 188 print ("In Since both the properties are not satisfied; the
 given system is not linear.")
 189 first_condition = 0
 190
 191 shift = input ("A system is time-invariant if a time shift
 i.e., advance or delay in the input always results only in an
 identical time shift in the output. In Enter the value for
 shifting factor: ")
 192
 193 if (first_condition + time (shift, choice) == 2): # Condition
 to check for LTI system or not
 194 print ("In The given system is a Linear Time Invariant
 (LTI) system. ")
 195 else:

(ii)

```
196     print("The given system is not a Linear Time Invariant  
197 (LTI) system.")  
198     break  
199  
200 elif choice == str(3):  
201     break # Exit Loop  
202  
203 else:  
204     print("Error! Invalid Input! Please try again. An")
```

+ INFERENCE:

Examine the given time sequence as Linear Time Invariant (LTI) system or not using Python code and results verified.

Check Homogeneity Condition

Spyder (Python 3.8)

```
File Edit Search Source Run Debug Consoles Projects Tools View Help
E:\Plan B\Amrita Vishwa Vidyapeetham\Subject Materials\Semester III\Signal Processing Lab (19CCE281)\Assignments\Experiment 4\Experiment 4.py
Experiment 4.py
1 # import library source files
2 import numpy as np
3 import matplotlib.pyplot as plt
4
5 n=np.arange(0,4) # Get evenly spaced values within the interval [0,4)
6
7 # Given Input Sequences:
8 x1 = [2,4,7,3]
9 x2 = [1,2,5,3]
10
11 def scaling (scale, choice): # Check Homogeneity Condition
12     before_scaling=[]
13     after_scaling=[]
14
15     if choice == str(1): # Calculate for x1 = [2,4,7,3]
16
17         for sample in range(len(n)):
18             temp = sample * x1[sample]
19             before_scaling.append(int(scale)*temp) # Scaling the output
20
21         for sample in range(len(n)):
22             temp = int(scale) * x1[sample]
23             after_scaling.append(sample*temp) # Sending output into system
24
25     else: # Calculate for x2 = [1,2,5,3]
26
27         for sample in range(len(n)):
28             temp = sample * x2[sample]
29             before_scaling.append(int(scale)*temp) # Scaling the output
30
31         for sample in range(len(n)):
32             temp = int(scale) * x2[sample]
33             after_scaling.append(sample*temp) # Sending output into system
34
35     plt.subplot(3,1,1) # subplot(rows, columns, index) describes the figure layout
36     plt.xlabel('n')
37     plt.ylabel('y' + str(choice) + ' [n]')
38     plt.title('Before Scaling x' + str(choice) + ' [0, 8, 28, 18]') # Formats the specified value(s) and insert them inside the string's placeholder.
39     plt.stem(n,before_scaling) # Stem plot plots vertical lines at each x position covered under the graph from the baseline to y, and places a marker there.
40
41     plt.subplot(3,1,3) # subplot(rows, columns, index) describes the figure layout
42     plt.xlabel('n')
43     plt.ylabel('y' + str(choice) + ' [n]')
44     plt.title('After Scaling x' + str(choice) + ' [0, 8, 28, 18]') # Formats the specified value(s) and insert them inside the string's placeholder.
45     plt.stem(n,after_scaling) # Stem plot plots vertical lines at each x position covered under the graph from the baseline to y, and places a marker there.
46
47     plt.show() # To view both the sub-plots
48
49     if (before_scaling == after_scaling): # Condition to check if both the systems are equal or not after scaling
50         print("The given system satisfies Homogeneity Condition.")
51     else:
52         print("The given system does not satisfy Homogeneity Condition.")
53
54
55 def superposition(): # Check Additivity Condition
```

Before Scaling x[0, 8, 28, 18]

After Scaling x[0, 8, 28, 18]

Variable explorer Help File

Console 1/A

```
Python 3.8.0 (default, Apr 13 2021, 15:08:03) [MSC v.1916 64 bit (AMD64)]
Type "copyright", "credits" or "license" for more information.

IPython 7.22.0 -- An enhanced Interactive Python.

In [1]: runfile('E:/Plan B/Amrita Vishwa Vidyapeetham/Subject Materials/Semester III/Signal Processing Lab (19CCE281)/Assignments/Experiment 4/Experiment 4.py', wdir='E:/Plan B/Amrita Vishwa Vidyapeetham/Subject Materials/Semester III/Signal Processing Lab (19CCE281)/Assignments/Experiment 4')

MENU: Determine whether the system  $y[n] = nx[n]$  is linear and time invariant.
1. x1 = [2, 4, 7, 3]
2. x2 = [1, 2, 5, 3]
3. Exit
Enter the number corresponding to the menu to implement the input sequence: 1
A linear system is any system that obeys the properties of scaling (homogeneity) and superposition (additivity).
1. Homogeneity Condition: -
   Enter the value for scaling factor: 2
```

LSP Python: ready Kite ready conda (Python 3.8.0) Line 4, Col 1 ASCII CRLF RW Mem 91%

26°C 01:44 13-10-2021

Spyder (Python 3.8)

```
File Edit Search Source Run Debug Consoles Projects Tools View Help
E:\Plan B\Amrita Vishwa Vidyapeetham\Subject Materials\Semester III\Signal Processing Lab (19CCE281)\Assignments\Experiment 4\Experiment 4.py
Experiment 4.py
19
20     for sample in range(len(n)):
21         temp = int(scale) * x1[sample]
22         after_scaling.append(sample*temp) # Sending output into system
23
24     else: # Calculate for x2 = [1,2,5,3]
25
26         for sample in range(len(n)):
27             temp = sample * x2[sample]
28             before_scaling.append(int(scale)*temp) # Scaling the output
29
30         for sample in range(len(n)):
31             temp = int(scale) * x2[sample]
32             after_scaling.append(sample*temp) # Sending output into system
33
34     plt.subplot(3,1,1) # subplot(rows, columns, index) describes the figure layout
35     plt.xlabel('n')
36     plt.ylabel('y' + str(choice) + ' [n]')
37     plt.title('Before Scaling x' + str(choice) + ' [0, 8, 28, 18]') # Formats the specified value(s) and insert them inside the string's placeholder.
38     plt.stem(n,before_scaling) # Stem plot plots vertical lines at each x position covered under the graph from the baseline to y, and places a marker there.
39
40     plt.subplot(3,1,3) # subplot(rows, columns, index) describes the figure layout
41     plt.xlabel('n')
42     plt.ylabel('y' + str(choice) + ' [n]')
43     plt.title('After Scaling x' + str(choice) + ' [0, 8, 28, 18]') # Formats the specified value(s) and insert them inside the string's placeholder.
44     plt.stem(n,after_scaling) # Stem plot plots vertical lines at each x position covered under the graph from the baseline to y, and places a marker there.
45
46     plt.show() # To view both the sub-plots
47
48     if (before_scaling == after_scaling): # Condition to check if both the systems are equal or not after scaling
49         print("The given system satisfies Homogeneity Condition.")
50     else:
51         print("The given system does not satisfy Homogeneity Condition.")
52
53
54 def superposition(): # Check Additivity Condition
```

Before Scaling x[0, 8, 28, 18]

After Scaling x[0, 8, 28, 18]

Variable explorer Help File

Console 1/A

```
2. x2 = [1, 2, 5, 3]
3. Exit
Enter the number corresponding to the menu to implement the input sequence: 1
A linear system is any system that obeys the properties of scaling (homogeneity) and superposition (additivity).
1. Homogeneity Condition: -
   Enter the value for scaling factor: 2

Figures now render in the Plots pane by default. To make them also appear inline in the Console, uncheck "Mute Inline Plotting" under the Plots pane options menu.

The given system satisfies Homogeneity Condition.
2. Superposition Condition: -
   The given system satisfies Superposition Condition.
```

LSP Python: ready Kite ready conda (Python 3.8.0) Line 4, Col 1 ASCII CRLF RW Mem 92%

01:45 13-10-2021

Step 1: Scale the output.

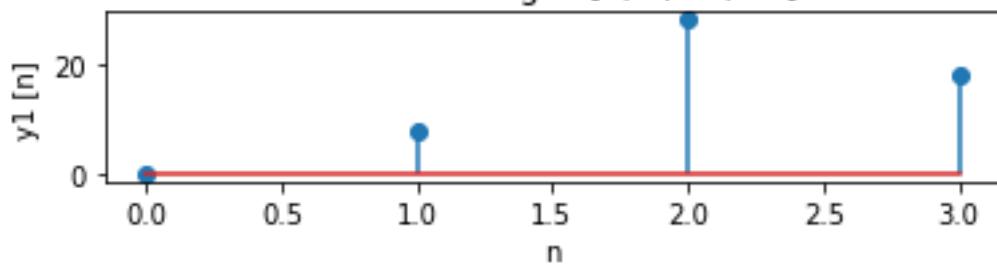
Step 2: Send output into the system.

Step 3: Plot graphs before and after scaling.

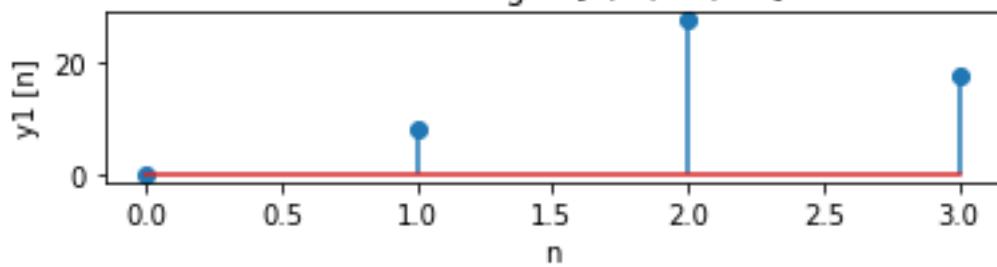
Step 4: Check if both the systems are equal or not after scaling.

$$x1 = [2, 4, 7, 3]$$

Before Scaling $x1[0, 8, 28, 18]$

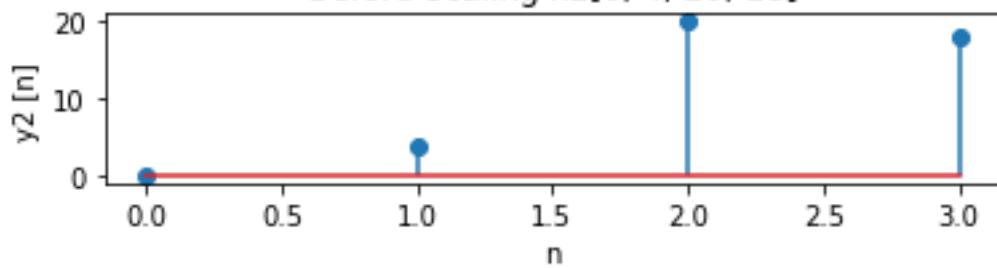


After Scaling $x1[0, 8, 28, 18]$

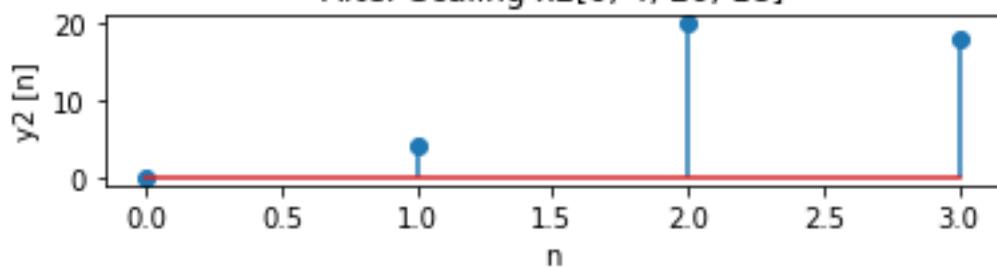


$$x2 = [1, 2, 5, 3]$$

Before Scaling $x2[0, 4, 20, 18]$



After Scaling $x2[0, 4, 20, 18]$



The given system satisfies the Homogeneity Condition.

Check Additivity Condition

Spyder (Python 3.8)

```
File Edit Search Source Run Debug Consoles Projects Tools View Help
E:\Vitan B\Amrita Vishwa Vidyapeetham\Subject Materials\Semester III\Signal Processing Lab (19CCE281)\Assignments\Experiment 4\Experiment 4.py
Experiment 4.py
53
54     return(0)
55
56 def superposition(): # Check Additivity Condition
57     before_adding=[]; after_adding=[]; y1=[]; y2=[]; x=[]
58
59     for sample in range(len(n)):
60         y1.append(sample*x1[sample])
61         y2.append(sample*x2[sample])
62         before_adding.append(y1[sample]*y2[sample]) # Processing System Individually
63
63     plt.subplot(3,1,1) # subplot(rows, columns, index) describes the figure layout
64     plt.xlabel('n')
65     plt.ylabel('y[n]')
66     plt.title('H (x1 (t) + x2 (t))')
67     plt.stem(n,after_adding) # Stem plot plots vertical lines at each x position covered under the graph from the baseline to y, and places a marker there.
68
69     for sample in range(len(n)):
70         x.append(x1[sample]+x2[sample])
71         after_adding.append(sample*x[sample]) # Processing System Together
72
73     plt.subplot(3,1,3) # subplot(rows, columns, index) describes the figure layout
74     plt.xlabel('n')
75     plt.ylabel('y[n]')
76     plt.title('H (x1 (t)) + H (x2 (t))')
77     plt.stem(n,after_adding) # Stem plot plots vertical lines at each x position covered under the graph from the baseline to y, and places a marker there.
78
79     plt.show() # To view both the sub-plots
80
81     print('\n 2. Superposition Condition: -')
82     if (before_adding == after_adding): # Condition to check if both the systems are equal or not after adding
83         print("  The given system satisfies Superposition Condition.")
84         return(1)
85     else:
86         print("  The given system does not satisfy Superposition Condition.")
87         return(0)
88
89 def time(shift, choice): # Check Time Variance
90     input_delay=[]; output_delay=[]
91
92
93
94
95
96
97
98
99
99
```

Type here to search 0 26°C 13-10-2021

Variable explorer Help Files

Console 1/A

1. Homogeneity Condition: -
Enter the value for scaling factor: 2

The given system satisfies Homogeneity Condition.

2. Superposition Condition: -
The given system satisfies Superposition Condition.

Since both the properties are satisfied; the given system is linear.

A system is time-invariant if a time shift (i.e., advance or delay) in the input always results only in an identical time shift in the output.
Enter the value for shifting factor: 1
Since both the outputs are same the given system is time variant

LSP Python: ready Kite: ready coda (Python 3.8.6) Line 54, Col 1 ASCII CRLF RW Mem 92%

Spyder (Python 3.8)

```
File Edit Search Source Run Debug Consoles Projects Tools View Help
E:\Vitan B\Amrita Vishwa Vidyapeetham\Subject Materials\Semester III\Signal Processing Lab (19CCE281)\Assignments\Experiment 4\Experiment 4.py
Experiment 4.py
53
54     return(0)
55
56 def superposition(): # Check Additivity Condition
57     before_adding=[]; after_adding=[]; y1=[]; y2=[]; x=[]
58
59     for sample in range(len(n)):
60         y1.append(sample*x1[sample])
61         y2.append(sample*x2[sample])
62         before_adding.append(y1[sample]*y2[sample]) # Processing System Individually
63
63     plt.subplot(3,1,1) # subplot(rows, columns, index) describes the figure layout
64     plt.xlabel('n')
65     plt.ylabel('y[n]')
66     plt.title('H (x1 (t) + x2 (t))')
67     plt.stem(n,after_adding) # Stem plot plots vertical lines at each x position covered under the graph from the baseline to y, and places a marker there.
68
69     for sample in range(len(n)):
70         x.append(x1[sample]+x2[sample])
71         after_adding.append(sample*x[sample]) # Processing System Together
72
73     plt.subplot(3,1,3) # subplot(rows, columns, index) describes the figure layout
74     plt.xlabel('n')
75     plt.ylabel('y[n]')
76     plt.title('H (x1 (t)) + H (x2 (t))')
77     plt.stem(n,after_adding) # Stem plot plots vertical lines at each x position covered under the graph from the baseline to y, and places a marker there.
78
79     plt.show() # To view both the sub-plots
80
81     print('\n 2. Superposition Condition: -')
82     if (before_adding == after_adding): # Condition to check if both the systems are equal or not after adding
83         print("  The given system satisfies Superposition Condition.")
84         return(1)
85     else:
86         print("  The given system does not satisfy Superposition Condition.")
87         return(0)
88
89 def time(shift, choice): # Check Time Variance
90     input_delay=[]; output_delay=[]
91
92
93
94
95
96
97
98
99
99
```

Type here to search 0 26°C 13-10-2021

Variable explorer Help Files

Console 1/A

4'

MENU: Determine whether the system $y[n] = nx[n]$ is linear and time invariant.
1. $x1 = [2, 4, 7, 3]$
2. $x2 = [1, 2, 5, 3]$
3. Exit

Enter the number corresponding to the menu to implement the input sequence: 2

A linear system is any system that obeys the properties of scaling (homogeneity) and superposition (additivity).

1. Homogeneity Condition: -
Enter the value for scaling factor: 2
The given system satisfies Homogeneity Condition.

2. Superposition Condition: -
The given system satisfies Superposition Condition.

Since both the properties are satisfied; the given system is linear.

A system is time-invariant if a time shift (i.e., advance or delay) in the input always results only in an identical time shift in the output.

LSP Python: ready Kite: ready coda (Python 3.8.6) Line 54, Col 1 ASCII CRLF RW Mem 92%

Step 1: Process System Individually.

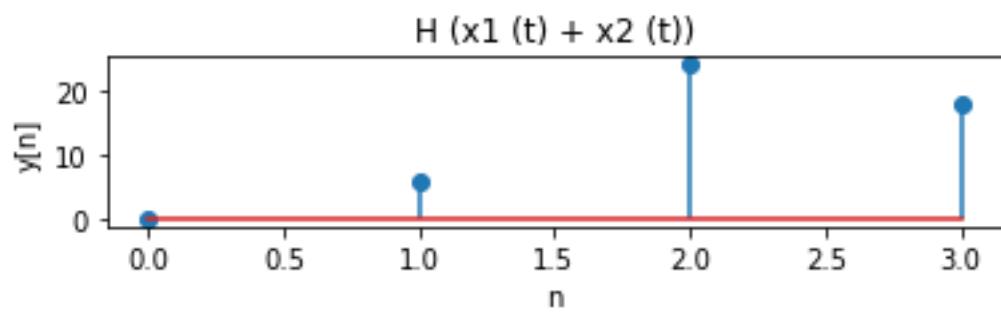
Step 2: Plot graph before adding.

Step 3: Process System Together.

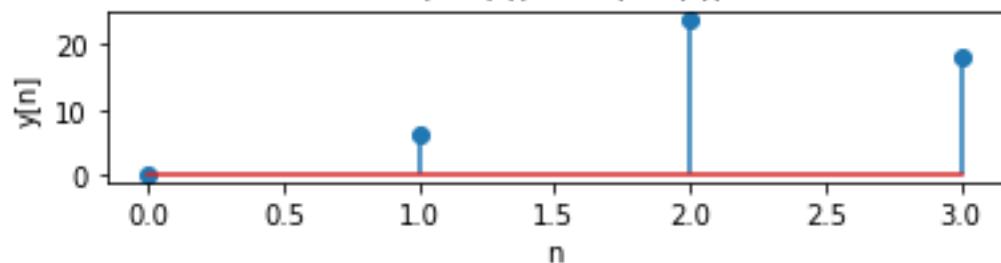
Step 4: Plot graph after adding.

Step 5: Check if both the systems are equal or not after adding.

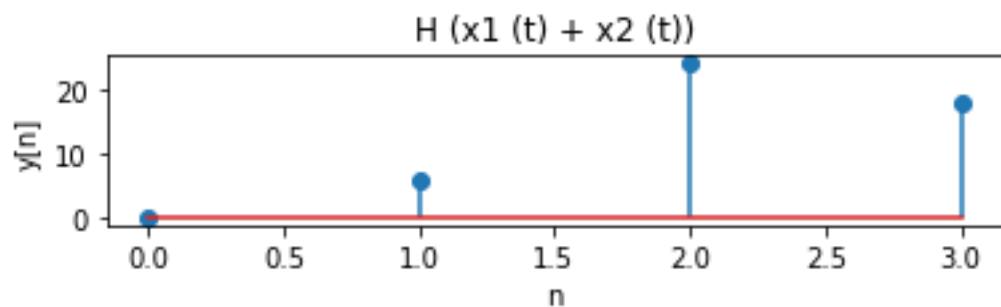
$$x1 = [2, 4, 7, 3]$$



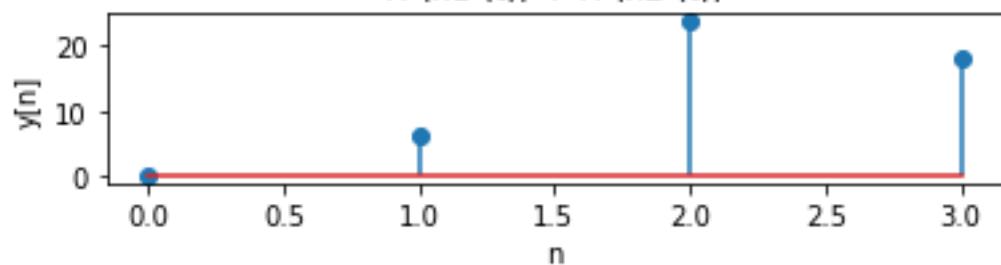
$$H(x_1(t)) + H(x_2(t))$$



$$x2 = [1, 2, 5, 3]$$



$$H(x_1(t)) + H(x_2(t))$$



The given system satisfies Superposition Condition.

Since both the properties are satisfied; the given system is linear.

Check Time Variance

Spyder (Python 3.8)

```
File Edit Search Source Run Debug Consoles Projects Tools View Help
E:\Vlban B\Amrita Vishwa Vidyapeetham\Subject Materials\Semester III\Signal Processing Lab (19CCE281)\Assignments\Experiment 4\Experiment 4.py
Experiment 4.py
89 def time(shift, choice): # Check Time Variance
90     input_delay=[]
91     output_delay=[]
92
93     if choice == str(1): # Calculate for x1 = [2,4,7,3]
94
95         # Output Delay
96         for sample in range(len(x1)):
97             output_delay.append(sample*x1[sample]) # Sending output into system
98
99         for sample in range(int(shift)):
100             for sample in range(len(x1)):
101                 output_delay.append(0) # Adds extra sample at the end for shifting
102
103             for sample in range(int(shift)):
104                 for sample in range(len(x1)-1,0,-1):
105                     output_delay[sample]=output_delay[sample-1] # Shift the sample
106
107             for sample in range(int(shift)):
108                 output_delay[sample]=0 # Assign zero to the empty elements for the shifted bit
109
110             # Input Delay
111             for sample in range(int(shift)):
112                 x1.append(0) # Adds extra sample at the end for shifting
113
114             for sample in range(int(shift)):
115                 for sample in range(len(x1)-1,0,-1):
116                     x1[sample]=x1[sample-1] # Shift the sample
117
118             for sample in range(int(shift)):
119                 x1[sample]=0 # Assign zero to the empty elements for the shifted bit
120
121             for sample in range(len(x1)):
122                 input_delay.append(sample*x1[sample]) # Sending input into system
123
124     else: # Calculate for x2 = [1,2,5,3]
125
126         # Output Delay
127         for sample in range(len(x2)):
128             output_delay.append(sample*x2[sample]) # Sending output into system
129
130         for sample in range(int(shift)):
131             for sample in range(len(x2)):
132                 output_delay.append(0) # Adds extra sample at the end for shifting
133
134             for sample in range(int(shift)):
135                 for sample in range(len(x2)-1,0,-1):
136                     output_delay[sample]=output_delay[sample-1] # Shift the sample
137
138             for sample in range(int(shift)):
139                 output_delay[sample]=0 # Assign zero to the empty elements for the shifted bit
140
141             # Input Delay
142             for sample in range(int(shift)):
143                 x2.append(0) # Adds extra sample at the end for shifting
144
145             for sample in range(int(shift)):
146                 for sample in range(len(x2)-1,0,-1):
147                     x2[sample]=x2[sample-1] # Shift the sample
148
149             for sample in range(len(x2)):
150                 input_delay.append(sample*x2[sample]) # Sending input into system
151
152             plt.subplot(3,1,1) # subplot(rows, columns, index) describes the figure layout
153             plt.xlabel('t')
154             plt.ylabel('y (t)')
155             plt.title('Shifting input by ' + str(shift) + ' for x' + str(choice) +
156             '(I)').format(input_delay) # Formats the specified value(s) and insert them inside the string's placeholder.
157             plt.stem(np.arange(len(input_delay)),input_delay) # Stem plot plots vertical lines at each x position covered under the graph from the baseline to y, and places a marker there.
158
159             plt.subplot(3,1,3) # subplot(rows, columns, index) describes the figure layout
160             plt.xlabel('t')
161             plt.ylabel('y (t)')
162             plt.title('Shifting output by ' + str(shift) + ' for x' + str(choice) +
163             '(O)').format(output_delay) # Formats the specified value(s) and insert them inside the string's placeholder.
164             plt.stem(np.arange(len(output_delay)),output_delay) # Stem plot plots vertical lines at each x position covered under the graph from the baseline to y, and places a marker there.
165
166             plt.show() # To view both the sub-plots
```

Type here to search

Variable explorer Help Note Files

Console 1/A

Figures now render in the Plots pane by default. To make them also appear inline in the Console, uncheck "Mute Inline Plotting" under the Plots pane options menu.

The given system satisfies Homogeneity Condition.

2. Superposition Condition: -
The given system satisfies Superposition Condition.

Since both the properties are satisfied; the given system is linear.

A system is time-invariant if a time shift (i.e., advance or delay) in the input always results only in an identical time shift in the output.
Enter the value for shifting factor: 1
Since both the outputs are not same, the given system is time variant.

The given system is not a Linear Time Invariant (LTI) system.

LSP Python: ready Kite: ready coda (Python 3.8.0) Line 91, Col 1 ASCII CRLF RW Mem 88% 26°C ENG 13-10-2021

Spyder (Python 3.8)

```
File Edit Search Source Run Debug Consoles Projects Tools View Help
E:\Vlban B\Amrita Vishwa Vidyapeetham\Subject Materials\Semester III\Signal Processing Lab (19CCE281)\Assignments\Experiment 4\Experiment 4.py
Experiment 4.py
131     for sample in range(int(shift)):
132         for sample in range(len(output_delay)-1,0,-1):
133             output_delay[sample]=output_delay[sample-1] # Shift the sample
134
135     for sample in range(int(shift)):
136         output_delay[sample]=0 # Assign zero to the empty elements for the shifted bit
137
138     # Input Delay
139     for sample in range(int(shift)):
140         x2.append(0) # Adds extra sample at the end for shifting
141
142     for sample in range(int(shift)):
143         for sample in range(len(x2)-1,0,-1):
144             x2[sample]=x2[sample-1] # Shift the sample
145
146     for sample in range(int(shift)):
147         x2[sample]=0 # Assign zero to the empty elements for the shifted bit
148
149     for sample in range(len(x2)):
150         input_delay.append(sample*x2[sample]) # Sending input into system
151
152     plt.subplot(3,1,1) # subplot(rows, columns, index) describes the figure layout
153     plt.xlabel('t')
154     plt.ylabel('y (t)')
155     plt.title('Shifting input by ' + str(shift) + ' for x' + str(choice) +
156     '(I)').format(input_delay) # Formats the specified value(s) and insert them inside the string's placeholder.
157     plt.stem(np.arange(len(input_delay)),input_delay) # Stem plot plots vertical lines at each x position covered under the graph from the baseline to y, and places a marker there.
158
159     plt.subplot(3,1,3) # subplot(rows, columns, index) describes the figure layout
160     plt.xlabel('t')
161     plt.ylabel('y (t)')
162     plt.title('Shifting output by ' + str(shift) + ' for x' + str(choice) +
163     '(O)').format(output_delay) # Formats the specified value(s) and insert them inside the string's placeholder.
164     plt.stem(np.arange(len(output_delay)),output_delay) # Stem plot plots vertical lines at each x position covered under the graph from the baseline to y, and places a marker there.
165
166     plt.show() # To view both the sub-plots
```

Type here to search

Variable explorer Help Note Files

Console 1/A

Enter the number corresponding to the menu to implement the input sequence: 2

A linear system is any system that obeys the properties of scaling (homogeneity) and superposition (additivity).

1. Homogeneity Condition: -
Enter the value for scaling factor: 2
The given system satisfies Homogeneity Condition.

2. Superposition Condition: -
The given system satisfies Superposition Condition.

Since both the properties are satisfied; the given system is linear.

A system is time-invariant if a time shift (i.e., advance or delay) in the input always results only in an identical time shift in the output.
Enter the value for shifting factor: 1
Since both the outputs are not same, the given system is time variant.

The given system is not a Linear Time Invariant (LTI) system.

LSP Python: ready Kite: ready coda (Python 3.8.0) Line 91, Col 1 ASCII CRLF RW Mem 88% 26°C ENG 13-10-2021

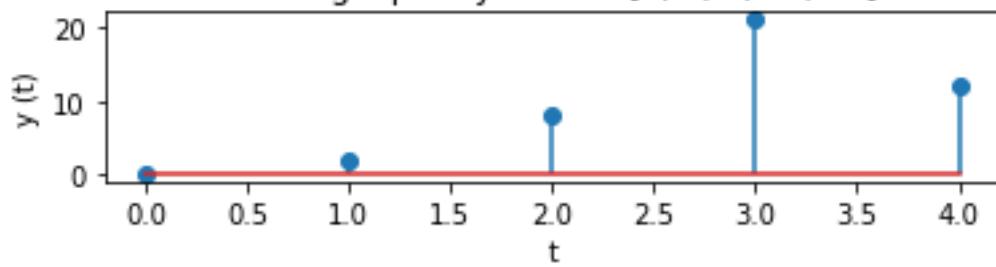
Step 1: Process input and output delay.

Step 2: Plot graphs before and after shifting.

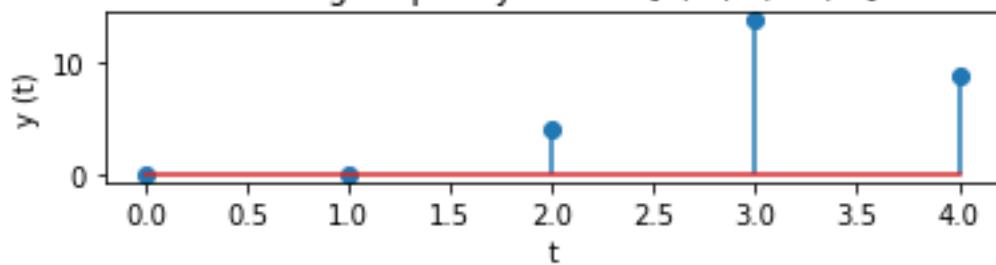
Step 3: Check if both the systems are equal or not after shifting.

$$x1 = [2, 4, 7, 3]$$

Shifting input by 1 for $x1[0, 2, 8, 21, 12]$

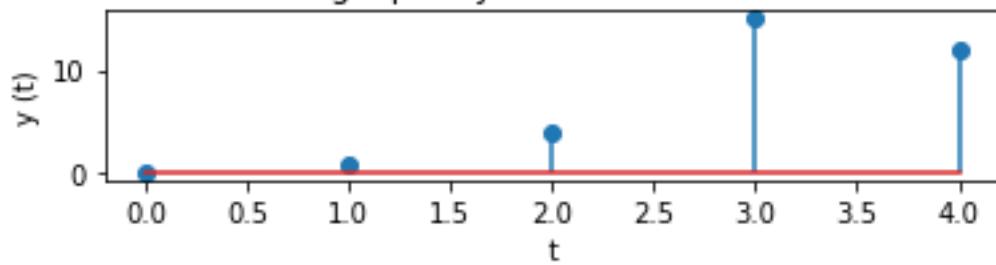


Shifting output by 1 for $x1[0, 0, 4, 14, 9]$

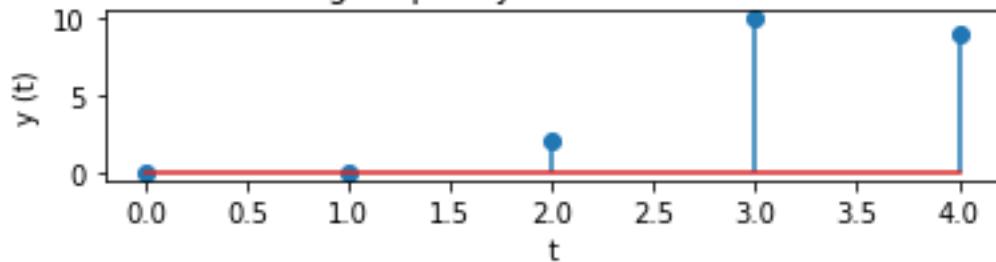


$$x2 = [1, 2, 5, 3]$$

Shifting input by 1 for $x2[0, 1, 4, 15, 12]$



Shifting output by 1 for $x2[0, 0, 2, 10, 9]$



Since both the outputs are not the same, the given system is time-variant.

The given system is not a Linear Time-Invariant (LTI) system.

Main Menu-Driven Implementation

The screenshot shows the Spyder Python IDE interface. On the left, the code editor displays 'Experiment 4.py' with Python code for determining system properties. On the right, there are two plots: 'Shifting input by 1 for x2[0, 1, 4, 15, 12]' and 'Shifting output by 1 for x2[0, 0, 2, 10, 9]'. Below the plots is a 'Console' window showing the execution of the script. The bottom status bar indicates the date and time as 13-10-2021.

```
125 # Main
126 while True: # This simulates a Do Loop
127     choice = input(
128         "MENU: Determine whether the system y[n] = nx[n] is linear and time invariant.\n"
129         "1. x1 = [2, 4, 7, 3]\n"
130         "2. x2 = [1, 2, 5, 3]\n"
131         "3. Exit\n"
132         "Enter the number corresponding to the menu to implement the input sequence: ") # Menu Driven Implementation
133
134     if choice == str(1) or choice == str(2):
135
136         scale = input ("A Linear system is any system that obeys the properties of scaling (Homogeneity) and superposition (additivity).\n"
137                     "\n"
138                     "1. Homogeneity Condition: -\n"
139                     "Enter the value for scaling factor: ")
140
141         if (scaling(scale, choice) == superposition()): # Condition to check if both the systems are linear or not
142             print("\nSince both the properties are satisfied; the given system is linear.")
143         else:
144             print("\nSince both the properties are not satisfied; the given system is not linear.")
145
146         shift = input("A system is time-invariant if a time shift (i.e., advance or delay) in the input always results only in an identical time shift in the output.\n"
147                     "Enter the value for shifting factor: ")
148
149         if (first_condition + time(shift, choice) == 2): # Condition to check for LTI system or not
150             print("\nThe given system is a Linear Time Invariant (LTI) system.")
151         else:
152             print("\nThe given system is not a Linear Time Invariant (LTI) system.")
153
154         break
155
156     elif choice == str(3):
157         break # Exit loop
158
159     else:
160         print("Error: Invalid Input! Please try again.\n")
```

Thank You!

```
Python 3.9.7 (default, Sep 16 2021, 16:59:28) [MSC v.1916 64 bit (AMD64)]  
Type "copyright", "credits" or "license" for more information.
```

```
IPython 7.29.0 -- An enhanced Interactive Python.
```

```
Restarting kernel...
```

```
In [1]:      'E:/Plan B/Amrita Vishwa Vidyapeetham/Subject Materials/Semester III/  
Signal Processing Lab (19CCE281)/Assignments/Experiment 4/Experiment 4.py'      = 'E:/Plan  
B/Amrita Vishwa Vidyapeetham/Subject Materials/Semester III/Signal Processing Lab  
(19CCE281)/Assignments/Experiment 4'
```

```
MENU: Determine whether the system  $y[n] = nx[n]$  is linear and time invariant.
```

1. $x_1 = [2, 4, 7, 3]$
2. $x_2 = [1, 2, 5, 3]$
3. Exit

```
Enter the number corresponding to the menu to implement the input sequence: 1
```

```
A linear system is any system that obeys the properties of scaling (homogeneity) and superposition (additivity).
```

1. Homogeneity Condition: -

```
Enter the value for scaling factor: 2
```

```
The given system satisfies Homogeneity Condition.
```

2. Superposition Condition: -

```
The given system satisfies Superposition Condition.
```

```
Since both the properties are satisfied; the given system is linear.
```

```
A system is time-invariant if a time shift (i.e., advance or delay) in the input always results only in an identical time shift in the output.
```

```
Enter the value for shifting factor: 1
```

```
Since both the outputs are not same, the given system is time variant.
```

```
The given system is not a Linear Time Invariant (LTI) system.
```

```
In [2]:
```