

NAME - SANTOSH (CB.EN.VHCE20053)

① DEPARTMENT - COMPUTER AND COMMUNICATION ENGINEERING

LAB TITLE AND CODE: SIGNAL PROCESSING LAB MCLU 281

EXPERIMENT NO: 3

DATE: 28/09/2021

## OPERATION ON SIGNALS

### \* AIM:

Generate and visualize the basic set of signal operations using Python code.

### \* SOFTWARE REQUIRED :

Spyder IDE (Anaconda 3) - Python 3.9.7 (64-bit)

### \* THEORY:

The basic set of signal operations can be broadly classified as -

#### → BASIC SIGNAL OPERATIONS PERFORMED ON DEPENDENT VARIABLES :-

In this transformation, only the quadrature axis values are modified i.e. magnitude of the signal changes, with no effects on the horizontal axis values or periodicity of signals. Let us look into these types in detail.

#### ① Amplitude Scaling of Signals -

Amplitude scaling is a very basic operation performed on signals to vary its strength. It can be mathematically represented as -

$$y(t) = c x(t)$$

where  $x(t)$  denotes continuous time signal,

$y(t)$  denotes resulting signal,

and  $c$  denotes the scaling factor.

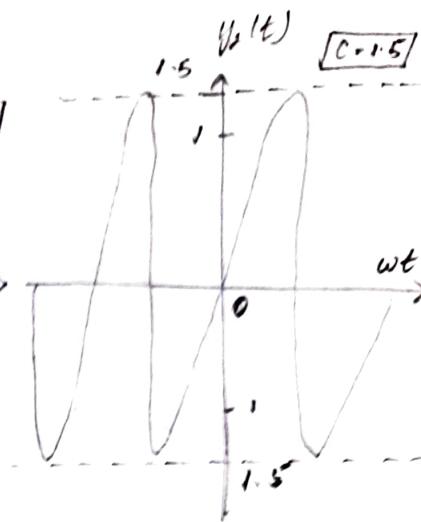
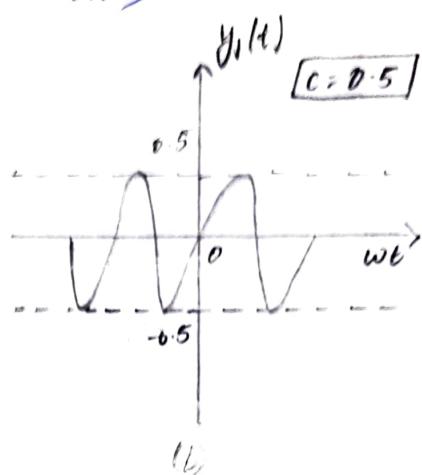
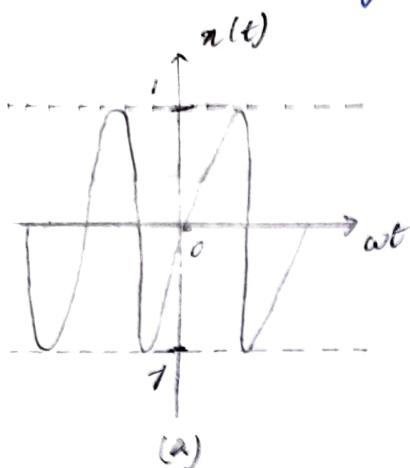
If  $c < 1 \rightarrow$  signal is attenuated.

$c > 1 \rightarrow$  signal is amplified.

⑨

For discrete time signals,

$$y[n] = cx[n]$$



This is illustrated in the diagram, where the signal is attenuated when  $c = 0.5$  in figure (b) and amplified when  $c = 1.5$  in figure (c).

Examples of amplitude scaling include amplifier, attenuator and resistor.

⑩ Addition of signals-

This particular operation involves the addition of amplitude of two or more signals at each instance of time or any other independent variables which are common between the signals. Addition of signals is illustrated in the diagram below, where  $x_1(t)$  and  $x_2(t)$  are two time dependent signals, performing the additional operation on them, we get-

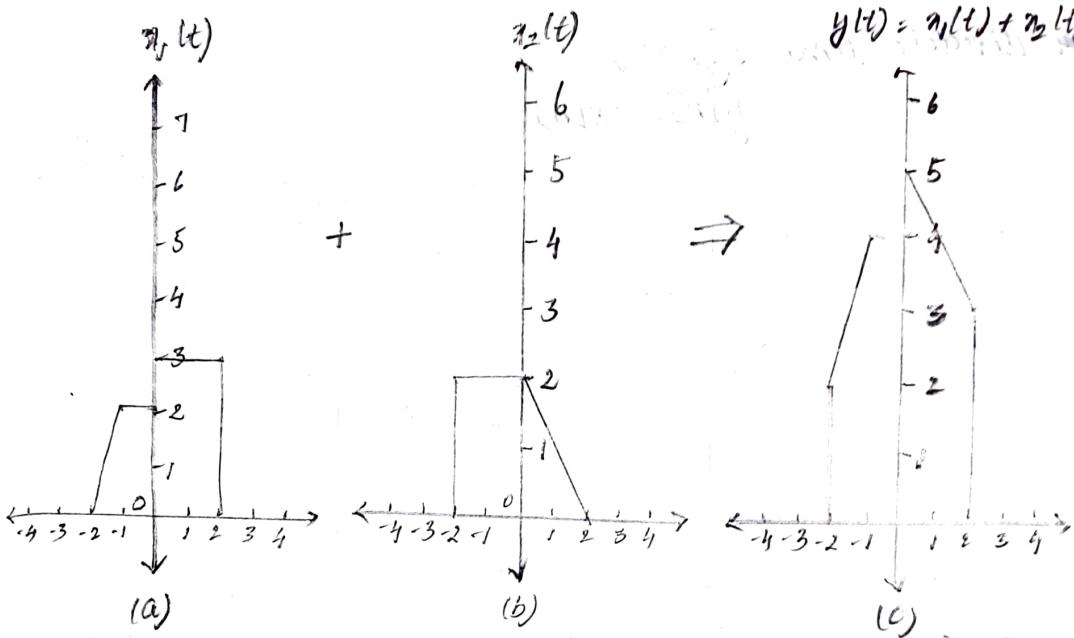
$$y(t) = x_1(t) + x_2(t)$$

For discrete time signals,

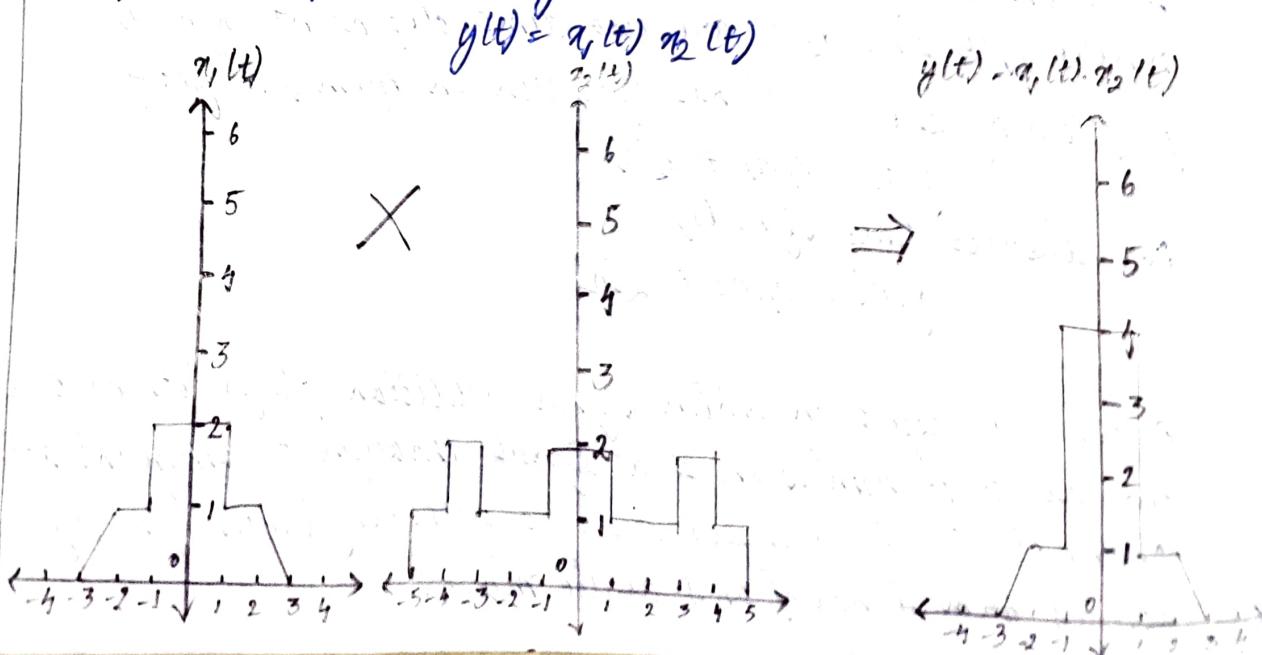
$$y[n] = x_1[n] + x_2[n]$$

A practical aspect in which signal addition plays its role is in the case of transmission of a signal through a communication channel. Other examples include dithering (Intentionally applied form of noise) and audio mixers.

(3)



③ Multiplication of Signals - Like addition, multiplication of signals also falls under the category of basic signal operations. Here multiplication of amplitude of two or more signals at each instance of time or any other independent variables is done which are common between the signals. The resultant signal we get has values equal to the product of amplitude of the parent signals for each instance of time. Multiplication of signals is illustrated in the diagram below, where  $a_1(t)$  and  $a_2(t)$  are two time dependent signals, on whom after performing the multiplication operation we get -

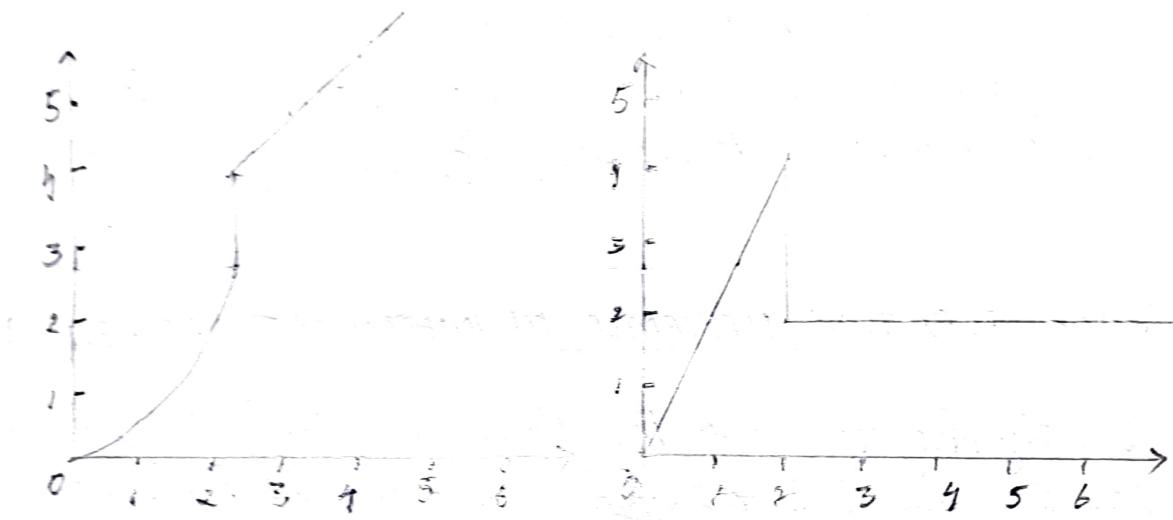


④

### Differentiation of Signals -

For differentiation of signals, it must be denoted that this operation is only applicable for continuous signals, as a discrete signal cannot be differentiated. The modified signal we get on differentiation has tangential values of the parent signal at all instance of time. Mathematically, it can be expressed as -

$$y(t) = \frac{d}{dt} a(t)$$



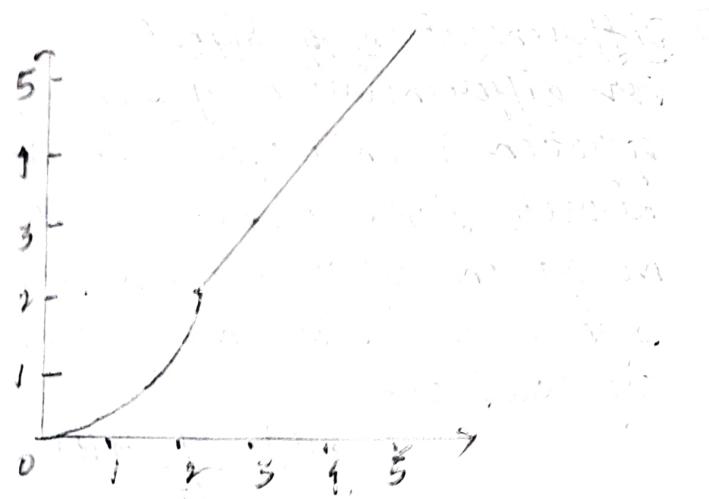
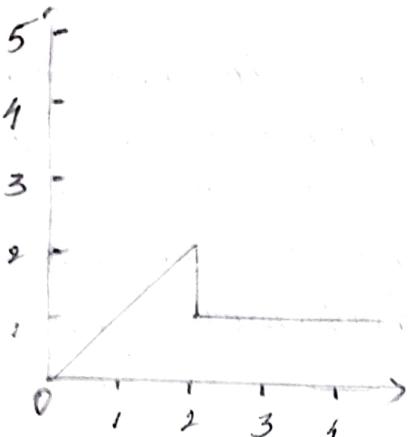
Differentiation of a signal takes the form of the gradient operator in the field of image or video processing. In the case of image processing, the gradient technique is a popular method which is used to detect the edges in the given image. With video processing, this operator is used for motion detection. This kind of processing is important in the field of robotics.

### Integration of Signals -

Like differentiation, integration of signals is also applicable to only continuous time signals. The limits of integration will be from  $-\infty$  to present instance of time  $t$ . It is mathematically expressed as -

$$y(t) = \int_{-\infty}^t a(\tau) d\tau$$

5



Integration is fundamental in signal-processing operations such as the Fourier transform, correlation, and convolution. These are, in turn, used to analyze different properties of a signal.

#### → BASIC SIGNAL OPERATIONS PERFORMED ON INDEPENDENT VARIABLES :-

This is exactly the opposite of the above mentioned cases, here the periodicity of a signal is ~~modified~~ modifying the horizontal axis values, while the amplitude or the strength remains constant. Let us look into these operations in detail.

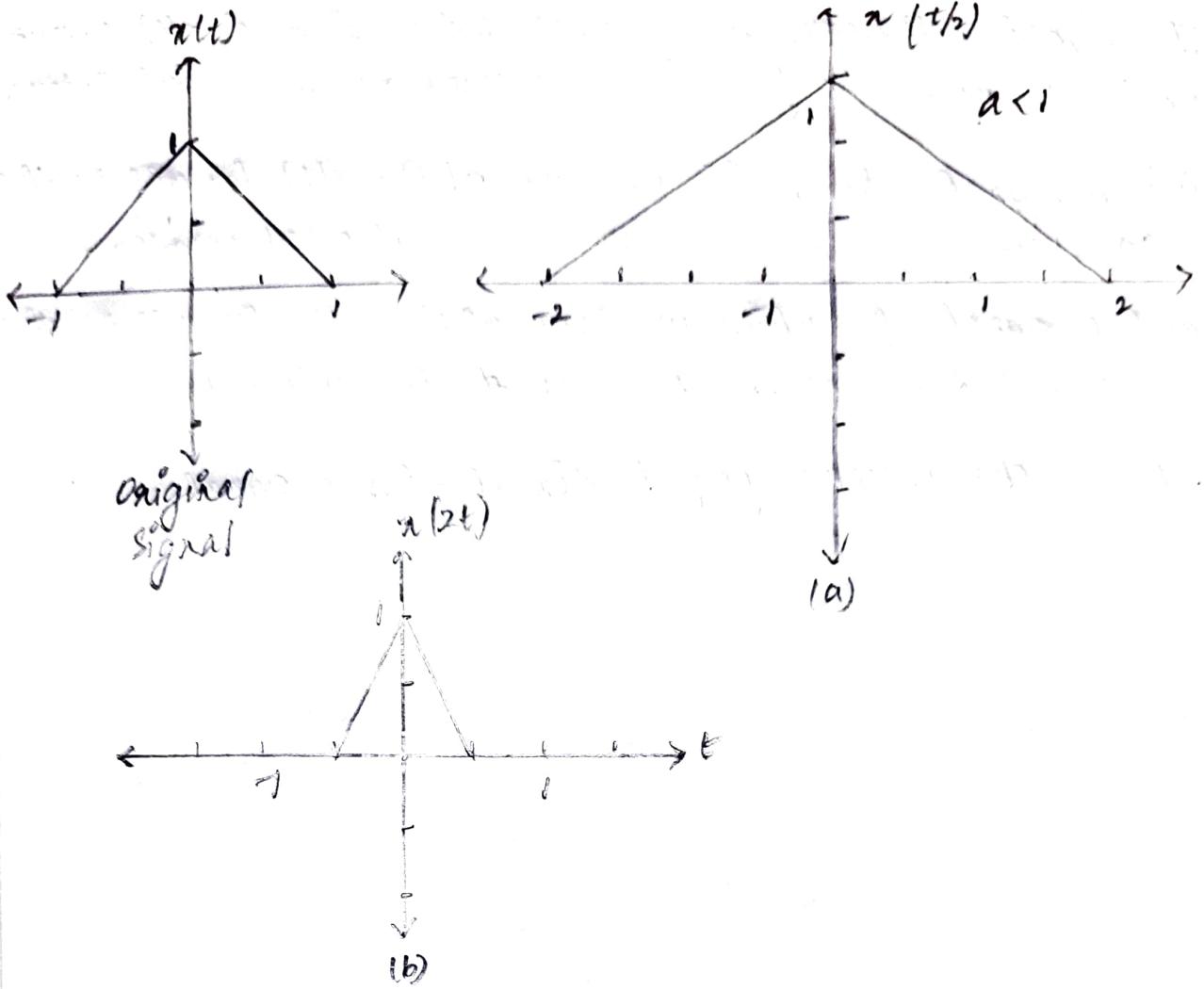
##### ① Time Scaling of Signals -

Time scaling of signals involves the modification of a signal's periodicity by stretching or compressing it along the time axis. It is mathematically expressed as,

$$y(at)$$

If  $a > 1$  implies the signal is compressed and  $a < 1$  implies the signal is expanded. This is illustrated diagrammatically for better understanding.

$\xrightarrow{\text{PTD}}$



Basically, when we perform time scaling, we change the rate at which the signal is sampled. changing the sampling rate of a signal is employed in the field of speech processing. A particular example of this would be a time-scaling-algorithm-based system developed to read text to the visually impaired.

② Reflection of signals -

Reflection of signal is a very interesting operation applicable on both continuous and discrete signals. Here in this case, the vertical axis acts as the mirror, and the transformed image obtained is exactly the mirror image of the parent signal.

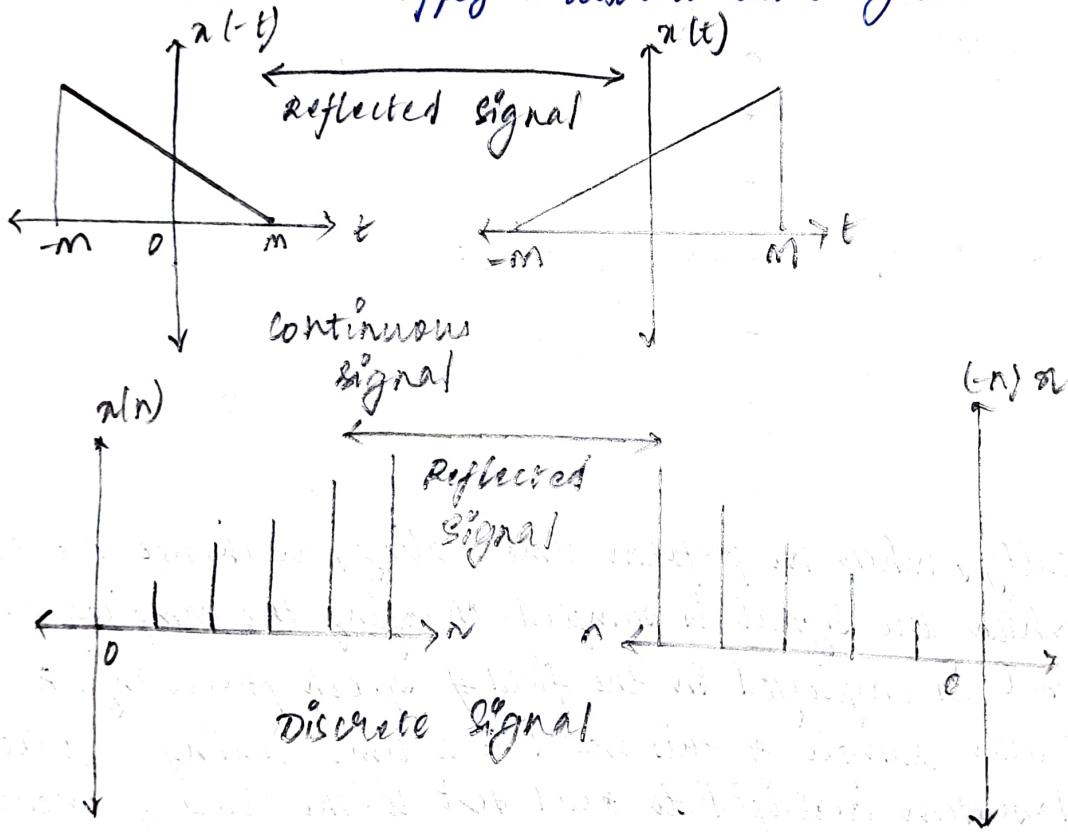
Let  $n(t)$  denote continuous time signal. Let  $y(t)$  denote the signal obtained by replacing time  $t$  with  $-t$  as shown by -

$$y(t) = n(-t)$$

The signal  $y(t)$  represents a reflected version of  $x(t)$  about the amplitude axis. The following two cases are of special interest -

- Even signals, for which we have  $x(-t) = x(t)$  for all  $t$ ; that is, an even signal is the same as its reflected version.
- Odd signals, for which we have  $x(-t) = -x(t)$  for all  $t$ ; that is, an odd signal is the negative of its reflected version.

Similar observations apply to discrete-time signals.



Time reversal is an important preliminary step when computing the convolution of signals; one signal is kept in its original state while the other is mirror-imaged and slid along the former signal to obtain the result. Time-reversal operations, therefore, are useful in various image-processing procedures, such as edge detection.

(8)

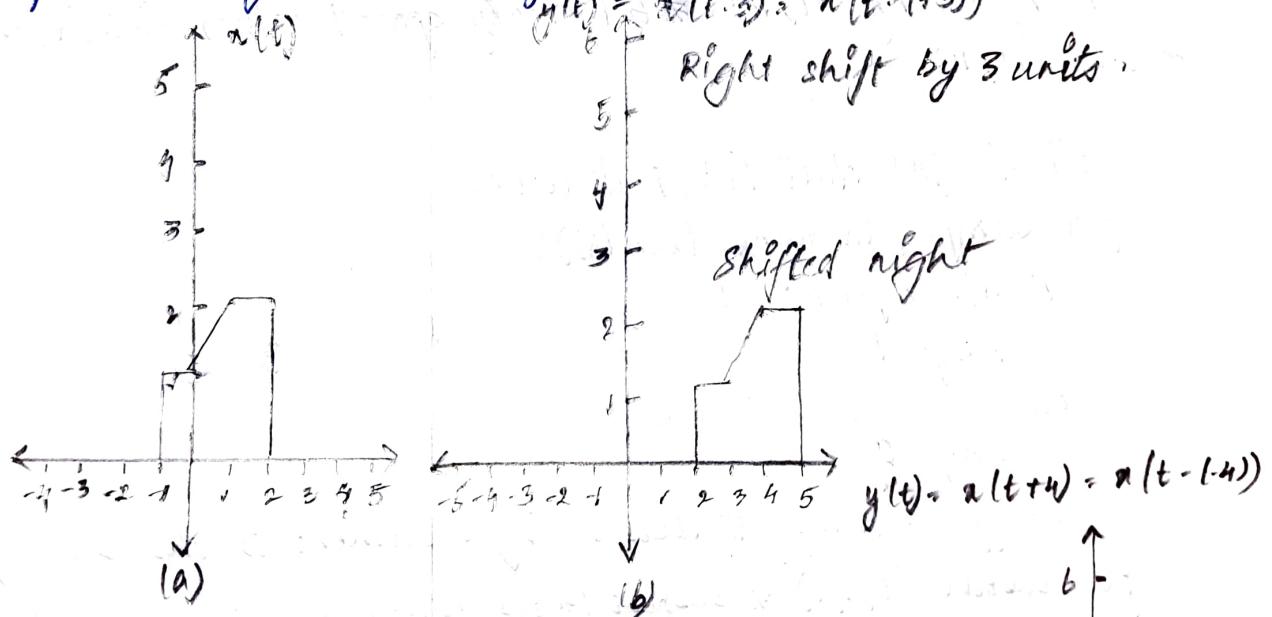
## ⑨ Time Shifting of Signals -

Time shifting of signals is probably the most important one, and most widely used amongst all basic signal operations. It's generally used to fast-forward or delay a signal, as is necessary in most practical circumstances. Time shifting is mathematically expressed as,

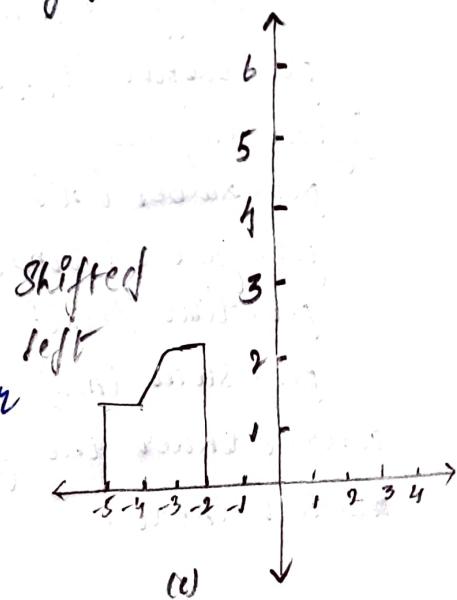
$$y(t) = a(t-t_0)$$

where,  $a(t)$  is the original signal,  
and  $t_0$  represents the shift in time.

For a signal  $a(t)$ , if the position shift to  $>0$ , then the signal is said to be right shifted or delayed. In the same manner, if  $t_0 < 0$ , implies a signal is left shifted or delayed. This has been explained diagrammatically below.



Time-shifting is an important operation that is used in many signal-processing applications. For example, a time-delayed version of the signal is used when ~~are~~ performed autocorrelation. Other examples include artificial intelligence, such as in systems that use Time Delay Neural Networks.



(9)

## \* GRAPH PLOTTING ALGORITHM :

The following steps were followed -

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

## \* PROGRAM WITH COMMENTS :

```
import numpy as np
```

```
import matplotlib.pyplot as plt
```

```
def step_sub # Step Signal Subtraction
```

```
    n = np.arange(0, 10) # Get evenly spaced values within the
    intervals [0, 10)
```

```
    a[] # u[n] Null List Declaration
```

```
    for sample in range(len(n)):
```

```
        if n[sample] >= 0
```

```
            temp = 1
```

```
        else :
```

```
            temp = 0
```

```
        a.append(temp) # Adds a single element to the existing list
```

```
plt. subplot(5, 1, 1) # subplot(rows, columns, index) describes the
figure layout
```

```
plt.xlabel('n')
```

```
plt.ylabel('u[n]')
```

```
plt.title('Unit Step Discrete-Time Function for u[n]')
```

```
plt.stem(n, a) # Stem plot plots vertical lines at each n position
covered under the graph from the baseline to y, and places a
marker there.
```

10

$b = [] \# u[n-2]$  Null list Declaration  
for sample in range (len(n)):

If  $n[\text{sample}] >= 2$ :

temp = 1

else

temp = 0

b.append (temp) # Adds a single item to the existing list

plt. subplot (5, 1, 3) # subplot (rows, columns, index) describes the figure layout

plt. xlabel ('n')

plt. ylabel ('u[n-2]')

plt. title ("Unit Step Discrete-Time Function for u[n-2]")

plt. stem (n, b) # Stem plot plots vertical lines at each x position covered under the graph from the baseline to y, and places a marker there.

result = [] # u[n] - u[n-2] Null list Declaration

for sample in range (len(n)):

result.append (a[sample] - b[sample]) # adds a single item to the existing list by subtracting u[n-2] from u[n] list.

plt. subplot (5, 1, 5) # subplot (rows, columns, index) describes the figure layout

plt. xlabel ('n')

plt. ylabel ('u[n] - u[n-2]')

plt. title ("Unit Step Discrete-Time Function for u[n] - u[n-2]")

plt. stem (n, result) # Stem plot plots vertical lines at each x position covered under the graph from the baseline to y, and places a marker there.

plt. show () # To view all the three sub-plots

11

```
def Impulse_plot () # Plot Impulse signal
    n = np.arange (0, 5) # get evenly spaced values within the
    interval [0, 5)
    arr = [2, 3, 4, 5, 0] # Initialize array with values as given in
    the question.
    plt.xlabel ('n')
    plt.ylabel ('x[n]')
    plt.stem (n, arr) # stem plot plots vertical lines at each n
    position covered under the graph from the baseline to y, and
    places a marker there.
    plt.show () # To view the plot.
```

```
def x_t_n (n): # Generate x[n] as given in the question
    list = [] # Null Array Declaration
    for sample in n: # If sample <= 3
        if sample >= -3:
            list.append (abs (sample)) # abs () returns the
            absolute value of a number
        else:
            list.append (0) # Adds a single item to the existing list
    else:
        list.append (0) # Adds a single item to the existing list
    return (list) # sends the value of list back and exits the
    function.
```

```
def y_t_n (n): # Generate y[n]
    list = [] # Null Array Declaration
    for sample in n:
        if sample == 0:
            list.append (0) # Adds a single item to the existing list.
        elif sample <= 4:
            if sample >= 4:
                list.append (sample / abs (sample)) # abs returns the
                absolute value of a number.
```

(12)

else :

list.append(0) # Adds a single item to the existing list.

else :

list.append(0) # Adds a single item to the existing list.

return(list) # send the value of list back and exits the function

def sketch\_i(): # sketch  $a[3n-1]$ 

n = np.arange(-6, 7) # get evenly spaced values within the interval [-6, 7]

plt.subplot(3, 1, 1) # subplot (rows, columns, index) describes the figure layout

plt.xlabel('n')

plt.ylabel('a[n]')

plt.title('Function of a[n]')

plt.stem(n, a\_of\_n(n)) # calls the function a-of-n() and displays a[n] plot

list = [] # Null list Declaration

for sample in n:

list.append(3 + sample - i) # Adds a single item to the existing list, a[3n-1].

plt.subplot(3, 1, 3) # subplot (rows, columns, index) describes the figure layout

plt.xlabel('n')

plt.ylabel('a[3n-1]')

plt.title('Function for a[3n-1]')

plt.stem(n, a\_of\_n(list)) # calls the function a-of-n(list) and displays a[3n-1] plot.

plt.show() # To view both the sub-plots.

PIV

(13)

```
def sketch_2() # sketch  $x[n-2] + y[n+2]$ 
```

$n = np.arange(-6, 7) #$  get evenly spaced values within the interval  $[-6, 7]$

$list\_x[] #$  Null List Declaration  
for sample in  $n:$

$list\_x.append(sample\_r) #$  adds a single item to the existing list,  $x[n-2]$

$plt.subplot(5, 1, 1) #$  subplot (rows, columns, index) describes the figure layout

$plt.xlabel('n')$

$plt.ylabel('x[n-2]')$

$plt.stem(n, x\_of\_n) #$  calls the function  $x\_of\_n(list\_x)$  and displays  $x[n-2]$  plot.

$list\_y[] #$  Null List Declaration

for sample in  $n:$

$list\_y.append(sample\_r2) #$  adds a single item to the existing list,  $y[n+2]$

$plt.subplot(5, 1, 3) #$  subplot (rows, columns, index) describes the figure layout

$plt.xlabel('n')$

$plt.ylabel('y[n+2]')$

$plt.title('Function for y[n+2]')$

$plt.stem(n, y\_of\_n(list\_y)) #$  calls the function  $y\_of\_n(list\_y)$  and displays  $y[n+2]$  plot.

$result = np.add(x\_of\_n(list\_x), y\_of\_n(list\_y)) #$  display the resultant graph,  $x[n-2] + y[n+2]$

$plt.subplot(5, 1, 5) #$  subplot (rows, columns, index)  
describes the figure layout

(14)

```

plt.xlabel('n')
plt.ylabel('x[n-2] + y[n+3]')
plt.title('Function for x[n-2] + y[n+3]')
plt.stem(n, result) # stem plot plots vertical lines at each n
position covered under the graph from the baseline to y, and
places a marker there.
plt.show() # To view all the three sub-plots.

```

# Main

while True: # This simulates a do loop

choice = input()

"MENU": In 1. Step Signal Subtraction \n 2. Plot Impulse  
 signal. \n 3. Sketch  $x[3n-1]$ . \n 4. Sketch  $x[n-2] + y[n+3]$ . \n  
 5. Exit \n Enter the number corresponding to the menu to  
 implement the choice: ") # Menu Driven Implementation

If choice == str(1): # str() returns the string version of the  
 variable "choice"

step\_sub() # Step Signal subtraction  
 break

elif choice == str(2):

Impulse - plot() # Plot Impulse signal  
 break

elif choice == str(3):

sketch\_1() # sketch  $x[3n-1]$   
 break

elif choice == str(4):

sketch\_2() # sketch  $x[n-2] + y[n+3]$   
 break

elif choice == str(5):

break # Exit loop

else:

print ("Error & Invalid Input! Please try again.\n")

\* INFERENCE:

performed the basic set of signal processing operations using Python code and results verified.

## Step Signal Subtraction

Spyder (Python 3.8)

```
# -*- coding: utf-8 -*-
...
Created on Tue Sep 28 13:56:58 2021
@author: Santosh (dell)
...
import numpy as np
import matplotlib.pyplot as plt
...
def step_sub(): # Step Signal Subtraction
    n = np.arange(0,10) # Get evenly spaced values within the interval [0,10)
    ...
    a=[ ] # u[n] Null List Declaration
    for sample in range(len(n)):
        if n[sample]>=0:
            temp1
        else:
            temp0
        a.append(temp0) # Adds a single item to the existing list
    plt.subplot(3,1,1) # subplot(rows, columns, index) describes the figure layout
    plt.xlabel('n')
    plt.ylabel('u [n]')
    plt.title('Unit Step Discrete-Time Function for u[n]')
    plt.stem(n,a) # Stem plot plots vertical lines at each x position covered under the graph from the baseline to y, and places a marker there.
    ...
    b=[ ] # u[n-2] Null List Declaration
    for sample in range(len(n)):
        if n[sample]>=2:
            temp1
        else:
            temp0
        b.append(temp0) # Adds a single item to the existing list
    plt.subplot(3,1,2) # subplot(rows, columns, index) describes the figure layout
    plt.xlabel('n')
    plt.ylabel('u [n-2]')
    plt.title('Unit Step Discrete-Time Function for u[n-2]')
    plt.stem(n,b) # Stem plot plots vertical lines at each x position covered under the graph from the baseline to y, and places a marker there.
    ...
    result=[ ] # u[n] - u[n-2] Null List Declaration
    for sample in range(len(n)):
        result.append(a[sample]-b[sample])
    plt.subplot(3,1,3) # subplot(rows, columns, index) describes the figure layout
    plt.xlabel('n')
    plt.ylabel('u [n] - u [n-2]')
    plt.title('Unit Step Discrete-Time Function for u[n] - u[n-2]')
    plt.stem(n,result) # Stem plot plots vertical lines at each x position covered under the graph from the baseline to y, and places a marker there.
    ...
    return result # u[n] - u[n-2] Null List Declaration
```

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 3\Experiment 3.py

Experiment 3.py

Unit Step Discrete-Time Function for  $u[n]$

Unit Step Discrete-Time Function for  $u[n-2]$

Unit Step Discrete-Time Function for  $u[n] - u[n-2]$

Variable explorer Help File

Console 1/A

Python 3.8.8 (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 3/Experiment 3.py', wdir='E:/Plan B/Amrita Vishwa Vidyapeetham/Subject Materials/Semester III/Signal Processing Lab (19CCE281)/Assignments/Experiment 3')

MENU:

1. Step Signal Subtraction.
2. Plot Impulse Signal.
3. Sketch  $x[3n - 1]$ .
4. Sketch  $x[n - 2] + y[n + 2]$ .
5. Exit

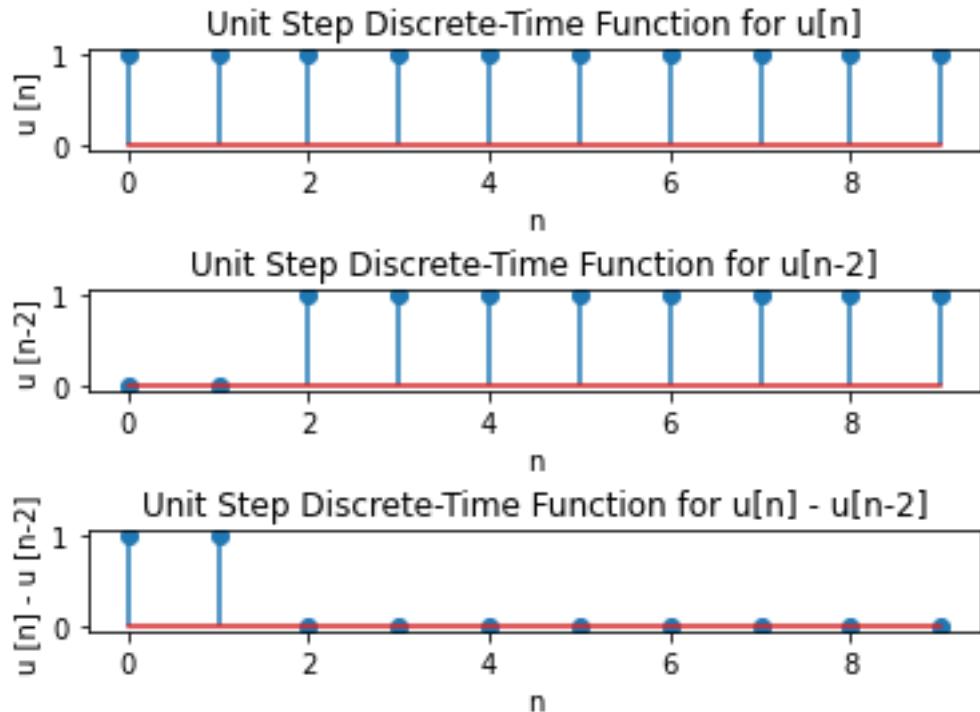
Enter the number corresponding to the menu to implement the choice: 1

Pictures now render in the Plots name by default. To make them also appear inline in the console

Type here to search

LSP Python: ready Kite: ready code (Python 3.8.8) Line 156, Col 1 UTF-8 CRLF RW Mem 86%

27°C Mostly cloudy 08:01 ENG 07-10-2021



## Plot Impulse Signal

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 3\Experiment 3.py
Experiment 3.py
22 plt.xlabel('n')
23 plt.ylabel('x[n]')
24 plt.title('Unit Step Discrete-Time Function for u[n]')
25 plt.stem(n,a) # Stem plot plots vertical lines at each x position covered under the graph from the baseline to y, and places a marker there.
26 b=[ ] # u[n-2] Null List Declaration
27 for sample in range(len(n)):
28     if n[sample]>=2:
29         temp1
30     else:
31         temp0
32     b.append(temp) # Adds a single item to the existing list
33 plt.subplot(5,1,3) # subplot(rows, columns, index) describes the figure layout
34 plt.xlabel('n')
35 plt.ylabel('u[n-2]')
36 plt.title('Unit Step Discrete-Time Function for u[n-2]')
37 plt.stem(n,b) # Stem plot plots vertical lines at each x position covered under the graph from the baseline to y, and places a marker there.
38
39 results[ ] = u[n] - u[n-2] Null List Declaration
40 for sample in range(len(n)):
41     result.append([sample]-b[sample]) # Adds a single item to the existing list by subtracting u[n-2] from u[n] list.
42     plt.subplot(5,1,5) # subplot(rows, columns, index) describes the figure layout
43     plt.xlabel('n')
44     plt.ylabel('u[n] - u[n-2]')
45     plt.title('Unit Step Discrete-Time Function for u[n] - u[n-2]')
46     plt.stem(n,result) # Stem plot plots vertical lines at each x position covered under the graph from the baseline to y, and places a marker there.
47     plt.show() # To view all the three sub-plots.
48
49 def impulse_plot(): # Plot Impulse Signal
50     n=np.arange(0,5) # Get evenly spaced values within the interval [0,5]
51     arr=[2,3,4,5,0] # Initialize array with values as given in question.
52     plt.xlabel('n')
53     plt.ylabel('x[n]')
54     plt.title('Unit Impulse Discrete-Time Function for x[n] = [2 3 4 5 0]')
55     plt.stem(n,arr) # Stem plot plots vertical lines at each x position covered under the graph from the baseline to y, and places a marker there.
56     plt.show() # To view the plot.
57
58
```

Unit Impulse Discrete-Time Function for  $x[n] = [2 3 4 5 0]$

In [2]: runfile('E:/Plan B/Amrita Vishwa Vidyapeetham/Subject Materials/Semester III/Signal Processing Lab (19CCE281)/Assignments/Experiment 3/Experiment 3.py', wdir='E:/Plan B/Amrita Vishwa Vidyapeetham/Subject Materials/Semester III/Signal Processing Lab (19CCE281)/Assignments/Experiment 3')

MENU:

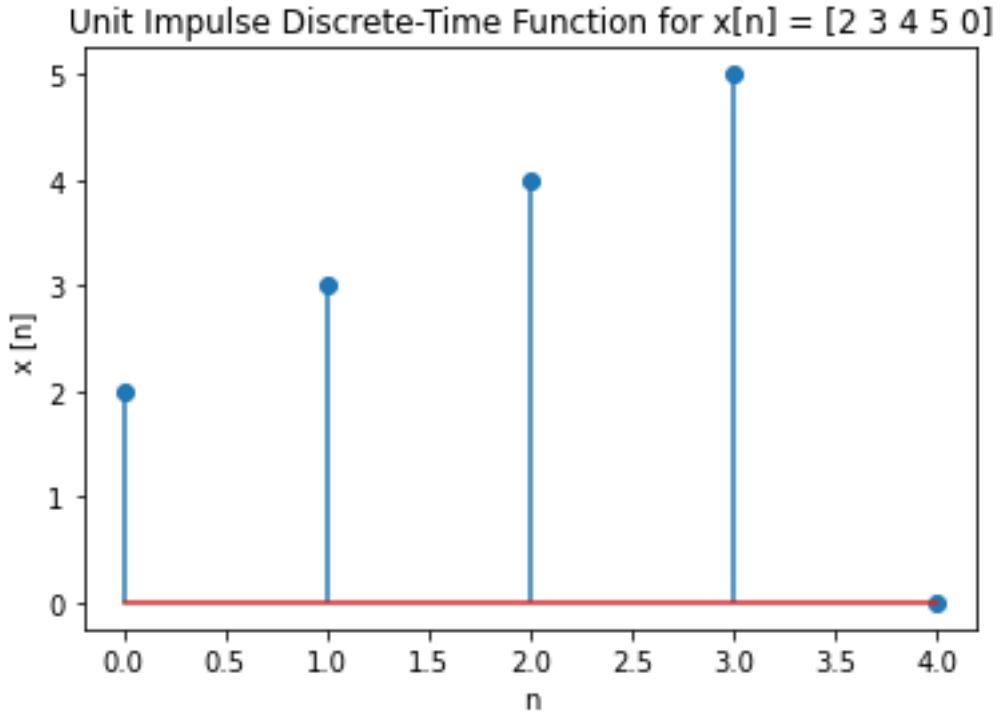
1. Step Signal Subtraction.
2. Plot Impulse Signal.
3. Sketch  $x[3n - 1]$ .
4. Sketch  $x[n - 2] + y[n + 2]$ .
5. Exit

Enter the number corresponding to the menu to implement the choice: 2

Type here to search

LSP Python: ready Kite: ready conda (Python 3.8.0) Line 156, Col 1 UTF-8 CRLF RW Mem 82%

27°C Mostly cloudy 08:03 07-10-2021



### Generate $x[n]$ as given in the question

The screenshot shows the Spyder Python 3.8 IDE interface. The code in the editor is as follows:

```
48 plt.show() # To view all the three sub-plots.
49
50 def impulse_plot(): # Plot Impulse Signal
51     n=np.arange(0,5) # Get evenly spaced values within the interval [0,5)
52     arr=[2,3,4,5,0] # Initialize array with values as given in question.
53     plt.stem(n,arr)
54     plt.xlabel('n')
55     plt.title('Unit Impulse Discrete-Time Function for x[n] = {2 3 4 5 0}')
56     plt.stem(n,arr) # Stem plot plots vertical lines at each x position covered under the
graph from the baseline to y, and places a marker there.
57     plt.show()
58
59 def x_of_n(n): # Generate x[n] as given in the question
60     list=[] # Null List Declaration
61     for sample in n:
62         if sample<=3:
63             if sample>-3:
64                 list.append(abs(sample)) # abs() returns the absolute value of a number
65             else:
66                 list.append(0) # Adds a single item to the existing list.
67         else:
68             list.append(0) # Adds a single item to the existing list.
69     return (list) # Send the value of list back and exits the function.
70
71 def y_of_n(n): # Generate y[n] as given in the question
72     list=[] # Null List Declaration
73     for sample in n:
74         if sample==0:
75             list.append(0) # Adds a single item to the existing list.
76         elif sample<=4:
77             if sample>-4:
78                 list.append(sample/abs(sample)) # abs() returns the absolute value of a
number.
79             else:
80                 list.append(0) # Adds a single item to the existing list.
81         else:
82             list.append(0) # Adds a single item to the existing list.
83     return(list) # Send the value of list back and exits the function.
84
85 def sketch_1(): # Sketch x[3n - 1]
86     n = np.arange(-6,7) # Get evenly spaced values within the interval [-6,7)
87     plt.subplot(3,1,1) # subplot(rows, columns, index) describes the figure layout
88     plt.xlabel('n')
89     plt.ylabel('x[n]')
90     plt.title('Function for x[n]')
91     plt.stem(n,x_of_n(n)) # Calls the function x_of_n(n) and displays x[n] plot.
92
93     list=[] # Null List Declaration
94     for sample in n:
95         list.append(3*sample - 1) # Adds a single item to the existing list, x[3n-1]
96
97     plt.subplot(3,1,2) # subplot(rows, columns, index) describes the figure layout
98     plt.xlabel('n')
99     plt.ylabel('x[3n-1]')
```

The console output shows the following plots:

- Function for  $x[n]$ : A stem plot with vertical lines at  $n = 0, 1, 2, 3$  with heights 2, 3, 4, 5 respectively.
- Function for  $x[3n-1]$ : A stem plot with vertical lines at  $n = -5, -2, 0, 1, 2, 4$  with heights 2, 1, 0, 1, 2, 0 respectively.

The status bar at the bottom right indicates: LSP Python: ready, Kite: ready, conda (Python 3.8.0), Line 156, Col 1, UTF-8, CRLF, RW, Mem 82%, 27°C Mostly cloudy, ENG, 07-10-2021.

### Generate $y[n]$ as given in the question

The screenshot shows the Spyder Python 3.8 IDE interface. The code in the editor is as follows:

```
59 def x_of_n(n): # Generate x[n] as given in the question
60     list=[] # Null List Declaration
61     for sample in n:
62         if sample<=3:
63             if sample>-3:
64                 list.append(abs(sample)) # abs() returns the absolute value of a number
65             else:
66                 list.append(0) # Adds a single item to the existing list.
67         else:
68             list.append(0) # Adds a single item to the existing list.
69     return (list) # Send the value of list back and exits the function.
70
71 def y_of_n(n): # Generate y[n] as given in the question
72     list=[] # Null List Declaration
73     for sample in n:
74         if sample==0:
75             list.append(0) # Adds a single item to the existing list.
76         elif sample<=4:
77             if sample>-4:
78                 list.append(sample/abs(sample)) # abs() returns the absolute value of a
number.
79             else:
80                 list.append(0) # Adds a single item to the existing list.
81         else:
82             list.append(0) # Adds a single item to the existing list.
83     return(list) # Send the value of list back and exits the function.
84
85 def sketch_1(): # Sketch x[3n - 1]
86     n = np.arange(-6,7) # Get evenly spaced values within the interval [-6,7)
87     plt.subplot(3,1,1) # subplot(rows, columns, index) describes the figure layout
88     plt.xlabel('n')
89     plt.ylabel('x[n]')
90     plt.title('Function for x[n]')
91     plt.stem(n,x_of_n(n)) # Calls the function x_of_n(n) and displays x[n] plot.
92
93     list=[] # Null List Declaration
94     for sample in n:
95         list.append(3*sample - 1) # Adds a single item to the existing list, x[3n-1]
96
97     plt.subplot(3,1,2) # subplot(rows, columns, index) describes the figure layout
98     plt.xlabel('n')
99     plt.ylabel('x[3n-1]')
```

The console output shows the following plots:

- Function for  $x[n]$ : A stem plot with vertical lines at  $n = 0, 1, 2, 3$  with heights 2, 3, 4, 5 respectively.
- Function for  $y[n+2]$ : A stem plot with vertical lines at  $n = -5, -2, 0, 1, 2, 4$  with heights 0, 1, 0, 1, 2, 0 respectively.
- Function for  $x[n-2] + y[n+3]$ : A stem plot with vertical lines at  $n = -5, -2, 0, 1, 2, 4$  with heights 2, 1, 0, 1, 2, 0 respectively.

The status bar at the bottom right indicates: LSP Python: ready, Kite: ready, conda (Python 3.8.0), Line 156, Col 1, UTF-8, CRLF, RW, Mem 83%, 27°C Mostly cloudy, ENG, 07-10-2021.

### Sketch $x[3n - 1]$

Spyder (Python 3.8)

```
File Edit Search Source Run Debug Consoles Projects Tools View Help
Experiment 3.py
68         list.append(0) # Adds a single item to the existing list.
69     return (list) # Send the value of list back and exits the function.
70
71 def x_of_n(n): # Generate y[n] as given in the question
72     list=[] # Null List Declaration
73     for sample in n:
74         if sample==0:
75             list.append(0) # Adds a single item to the existing list.
76         elif sample<=4:
77             if sample<=4:
78                 list.append(sample/abs(sample)) # abs() returns the absolute value of a
number.
79         else:
80             list.append(0) # Adds a single item to the existing list.
81     else:
82         list.append(0) # Adds a single item to the existing list.
83     return(list) # Send the value of list back and exits the function.
84
85 def sketch_1(): # Sketch x[3n - 1]
86     n = np.arange(-6,7) # Get evenly spaced values within the interval [-6,7)
87     plt.subplot(3,1,1) # subplot(rows, columns, index) describes the figure layout
88     plt.xlabel('n')
89     plt.ylabel('x [n]')
90     plt.title('Function for x[n]')
91     plt.stem(n, x_of_n(n)) # Calls the function x_of_n and displays x[n] plot.
92
93     list=[] # Null List Declaration
94     for sample in n:
95         list.append(3*sample - 1) # Adds a single item to the existing list, x[3n-1]
96
97     plt.subplot(3,1,3) # subplot(rows, columns, index) describes the figure layout
98     plt.xlabel('n')
99     plt.ylabel('x [3n-1]')
100    plt.title('Function for x[3n-1]')
101    plt.stem(n, list) # Calls the function x_of_n(list) and displays x[3n-1] plot.
102    plt.show() # To view both the sub-plots.
103
104 def sketch_2(): # Sketch x[n - 2] + y[n + 2]
105     n = np.arange(-6,7) # Get evenly spaced values within the interval [-6,7)
106
107     list_x1=[] # Null List Declaration
108     for sample in n:
```

Function for  $x[n]$

Function for  $x[3n-1]$

Variable explorer Help File

Console 1/A

5. Exit  
Enter the number corresponding to the menu to implement the choice: 2

In [3]: runfile('E:/Plan B/Amrita Vishwa Vidyapeetham/Subject Materials/Semester III/Signal Processing Lab (19CCE281)/Assignments/Experiment 3/Experiment 3.py', wdir='E:/Plan B/Amrita Vishwa Vidyapeetham/Subject Materials/Semester III/Signal Processing Lab (19CCE281)/Assignments/Experiment 3')

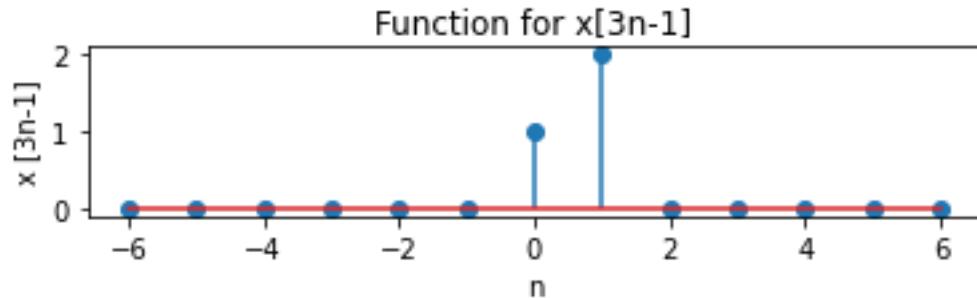
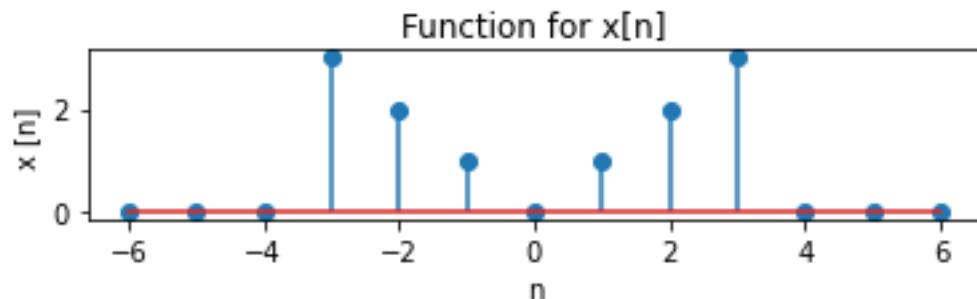
MENU:  
1. Step Signal Subtraction.  
2. Plot Impulse Signal.  
3. Sketch  $x[3n - 1]$ .  
4. Sketch  $x[n - 2] + y[n + 2]$ .  
5. Exit  
Enter the number corresponding to the menu to implement the choice: 3

In [4]: runfile('E:/Plan B/Amrita Vishwa Vidyapeetham/Subject Materials/Semester III/Signal Processing Lab (19CCE281)/Assignments/Experiment 3/Experiment 3.py', wdir='E:/Plan B/Amrita Vishwa Vidyapeetham/Subject Materials/Semester III/Signal Processing Lab (19CCE281)/Assignments/Experiment 3')

Python console History

LSP Python: ready Kite: ready conda (Python 3.8.0) Line 156, Col 1 UTF-8 CRLF RW Mem 83%

27°C Mostly cloudy 08:06 ENG 07-10-2021



### Sketch $x[n - 2] + y[n + 2]$

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 3\Experiment 3.py
Experiment 3.py
100     plt.title('Function for x[3n-1]')
101     plt.stem(n, x_of_n(list)) # Calls the function x_of_n(list) and displays x[3n-1] plot.
102     plt.show() # To view both the sub-plots.
103
104 def sketch_2(): # Sketch x[n - 2] + y[n + 2]
105     n = np.arange(-6,7) # Get evenly spaced values within the interval [-6,7)
106
107     list_x=[] # Null List Declaration
108     for sample in n:
109         list_x.append(sample - 2) # Adds a single item to the existing list, x[n-2]
110
111     plt.subplot(3,1,1) # subplot(rows, columns, index) describes the figure layout
112     plt.xlabel('n')
113     plt.ylabel('x [n-2]')
114     plt.title('Function for x[n-2]')
115     plt.stem(n, x_of_n(list_x)) # Calls the function x_of_n(list_x) and displays x[n-2]
116
117     list_y=[] # Null List Declaration
118     for sample in n:
119         list_y.append(sample + 2) # Adds a single item to the existing list, y[n+2]
120
121     plt.subplot(3,1,2) # subplot(rows, columns, index) describes the figure layout
122     plt.xlabel('n')
123     plt.ylabel('y [n+2]')
124     plt.title('Function for y[n+2]')
125     plt.stem(n, y_of_n(list_y)) # Calls the function y_of_n(list_y) and displays y[n+2]
126
127     result = np.add(x_of_n(list_x),y_of_n(list_y)) # Display the resultant graph, x[n - 2]
128     + y[n + 2]
129     plt.subplot(3,1,3) # subplot(rows, columns, index) describes the figure layout
130     plt.xlabel('n')
131     plt.ylabel('x[n-2]+y[n+3]')
132     plt.title('Function for x[n-2] + y[n+3]')
133     plt.stem(n, result) # Stem plot plots vertical lines at each x position covered under
134     the graph from the baseline to y, and places a marker there.
135
136     plt.show() # To view all the three sub-plots.
137
138 # Main
139 while True: # This simulates a Do Loop
140     choice = input(

```

Variable explorer Help Files

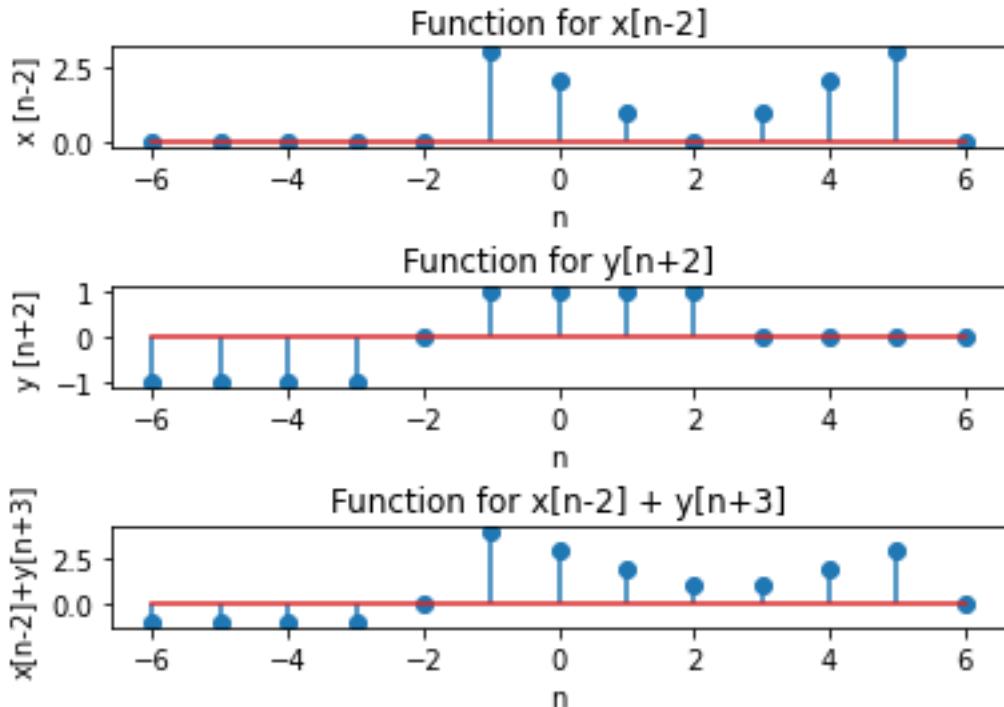
Console 1/A

4. Sketch  $x[n - 2] + y[n + 2]$ .  
5. Exit  
Enter the number corresponding to the menu to implement the choice: 3

In [4]: runfile('E:/Plan B/Amrita Vishwa Vidyapeetham/Subject Materials/Semester III/Signal Processing Lab (19CCE281)/Assignments/Experiment 3/Experiment 3.py', wdir='E:/Plan B/Amrita Vishwa Vidyapeetham/Subject Materials/Semester III/Signal Processing Lab (19CCE281)/Assignments/Experiment 3')  
4. Sketch  $x[n - 2]$ .  
5. Exit  
Enter the number corresponding to the menu to implement the choice: 4

In [5]: runfile('E:/Plan B/Amrita Vishwa Vidyapeetham/Subject Materials/Semester III/Signal Processing Lab (19CCE281)/Assignments/Experiment 3/Experiment 3.py', wdir='E:/Plan B/Amrita Vishwa Vidyapeetham/Subject Materials/Semester III/Signal Processing Lab (19CCE281)/Assignments/Experiment 3')  
4. Sketch  $x[n - 2] + y[n + 2]$ .  
5. Exit  
Enter the number corresponding to the menu to implement the choice: 5

LSP Python: ready Kite: ready conda (Python 3.8.0) Line 156, Col 1 UTF-8 CRLF RW Mem 83% 27°C Mostly cloudy 08:07 ENG 07-10-2021



## Menu Driven Implementation

The screenshot shows the Spyder Python 3.8 IDE interface. The code editor displays a script named 'Experiment 3.py' containing a menu-driven implementation for signal processing tasks. The script uses a while loop to prompt the user for a choice from a menu. The menu options include Step Signal Subtraction, Plot Impulse Signal, Sketch x[3n - 1], Sketch x[n - 2] + y[n + 2], and Exit. The script then performs the selected task, such as plotting signals or sketching them. Three subplots are shown in the figure window, labeled 'Function for x[n-2]', 'Function for y[n+2]', and 'Function for x[n-2] + y[n+3]'. The plots show discrete-time signals with vertical stems at specific n values.

```
121 plt.subplot(3,1,3) # subplot(rows, columns, index) describes the figure layout
122 plt.xlabel('n')
123 plt.ylabel('y[n+2]')
124 plt.title('Function for y[n+2]')
125 plt.stem(n, y_of_n(list_y)) # Calls the function y_of_n(list_y) and displays y[n+2]
126 plot.
127
128 result = np.add(x_of_n(list_x),y_of_n(list_y)) # Display the resultant graph, x[n - 2]
129 + y[n + 2]
130 plt.subplot(3,1,5) # subplot(rows, columns, index) describes the figure layout
131 plt.xlabel('n')
132 plt.ylabel('x[n-2]+y[n+3]')
133 plt.title('Function for x[n-2] + y[n+3]')
134 plt.stem(n, result) # Stem plot plots vertical lines at each x position covered under
135 the graph from the baseline to y, and places a marker there.
136 plt.show() # To view all the three sub-plots.
137
138 # Main:
139 while True: # This simulates a Do Loop
140     choice = input()
141     if choice == "1": # Step Signal Subtraction.
142         step_sub()
143     elif choice == str(2):
144         impulse_plot() # Plot Impulse Signal
145     elif choice == str(3):
146         sketch_1() # Sketch x[3n - 1]
147     elif choice == str(4):
148         sketch_2() # Sketch x[n - 2] + y[n + 2]
149     elif choice == str(5):
150         break # Exit loop
151     else:
152         print("Error: Invalid Input! Please try again.\n")
153
154
155
156
```

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 3/Experiment 3.py'      = 'E:/Plan  
B/Amrita Vishwa Vidyapeetham/Subject Materials/Semester III/Signal Processing Lab  
(19CCE281)/Assignments/Experiment 3'
```

MENU:

1. Step Signal Subtraction.
2. Plot Impulse Signal.
3. Sketch  $x[3n - 1]$ .
4. Sketch  $x[n - 2] + y[n + 2]$ .
5. Exit

```
Enter the number corresponding to the menu to implement the choice: 1
```

MENU:

1. Step Signal Subtraction.
2. Plot Impulse Signal.
3. Sketch  $x[3n - 1]$ .
4. Sketch  $x[n - 2] + y[n + 2]$ .
5. Exit

```
Enter the number corresponding to the menu to implement the choice: 2
```

MENU:

1. Step Signal Subtraction.
2. Plot Impulse Signal.
3. Sketch  $x[3n - 1]$ .
4. Sketch  $x[n - 2] + y[n + 2]$ .
5. Exit

```
Enter the number corresponding to the menu to implement the choice: 3
```

MENU:

1. Step Signal Subtraction.
2. Plot Impulse Signal.
3. Sketch  $x[3n - 1]$ .
4. Sketch  $x[n - 2] + y[n + 2]$ .
5. Exit

```
Enter the number corresponding to the menu to implement the choice: 4
```

MENU:

1. Step Signal Subtraction.
2. Plot Impulse Signal.
3. Sketch  $x[3n - 1]$ .
4. Sketch  $x[n - 2] + y[n + 2]$ .
5. Exit

```
Enter the number corresponding to the menu to implement the choice: 6
```

```
Error: Invalid Input! Please try again.
```

MENU:

1. Step Signal Subtraction.
2. Plot Impulse Signal.
3. Sketch  $x[3n - 1]$ .
4. Sketch  $x[n - 2] + y[n + 2]$ .
5. Exit

Enter the number corresponding to the menu to implement the choice: 5

In [2]: