① NAME - SANTOSH (CB.EN.U4CCE20053)
DEPARTMENT - COMPUTER AND COMMUNICATION ENGINEERING (CCE) A

LAB TITLE AND CODE : SIGNAL PROCESSING LAB 19CCE281
EXPERIMENT NUMBER : 5
DATE : 12/10/2021

DISCRETE - TIME CONVOLUTION SUM

* **AIM :**
Given the input $x[n]$ and impulse response $h[n]$, determine the system response $y[n]$ in discrete-time LTI system.

* **SOFTWARE REQUIRED :**
Spyder IDE (Anaconda3) - Python 3.9.7 (64-bit)

* **THEORY :**
Convolution can be used to determine the output of a system produces for a given input signal. It can be shown that a linear time-invariant system is completely characterized by its impulse response. The shifting property of the discrete-time impulse function tells us that the input signal to a system can be represented as a sum of scaled and shifted unit impulses.

Thus, by linearity, it would seem reasonable to compute the output signal as the sum of scaled and shifted unit impulse responses. That is exactly what the operation of convolution accomplishes. Hence, convolution can be used to determine a linear time-invariant system's output from knowledge of the input and the impulse response.

Discrete time convolution is an operation on two discrete time signals defined by the integral -

$$(f*g)[n] = \sum_{k=-\infty}^{\infty} f[k]g[n-k]$$

for all signals f, g defined on $\mathbb{Z}$. It is important to note that the operation of convolution is commutative, meaning that -

$$f * g = g * f$$

for all signals f, g defined on $\mathbb{Z}$. Thus, the convolution operation could have been just as easily stated using the equivalent definition -

$$(f*g)[n] = \sum_{k=-\infty}^{\infty} f[n-k]\, g[k]$$

for all signals f, g defined on $\mathbb{Z}$.

The convolution sum is realized as follows -

1. Invert $h[k]$ about $k=0$ to obtain $h[-k]$.
2. The function $h[n-k]$ is given by $h[-k]$ shifted to the right by n (if n is positive) and to the left (if n is negative) (note the sign of the independent variable).
3. Multiply $x[k]$ and $h[n-k]$ for the same coordinates on the k axis. The value obtained is the response at n i.e., the value of $y[n]$ at a particular n, the value chosen in step 2.

* 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 () functions.
4. Give a title to your plot using the ~~plot~~ .title () function.
5. Finally, to view your plot, we use the .show () function.

* THEORETICAL CALCULATION :

Given, $x[n] = [2, 3, 4]$ To find, $y[n]$

$h[n] = [1, 2, 3]$

SANTOSH - [CB.EN.U4CLE20053]

(3)

We know that, $y[n] = x[n] * h[n]$. Therefore,

	1	2	3			$\leftarrow h[n]$
$\times$		2	3	4		$\leftarrow x[n]$
	2	4	6	0	0	$\leftarrow x[0] . h[n] , i=0$
	0	3	6	9	0	$\leftarrow x[1] . h[n-1] , i=1$
+	0	0	4	8	12	$\leftarrow x[2] . h[n-2] , i=2$
	2	7	16	17	12	$\leftarrow y[n]$

$$\boxed{y[n] = [2, 7, 16, 17, 12]}$$

* PROGRAM WITH COMMENTS :

```
# Import library source files
Import numpy as np
Import matplotlib.pyplot as plt

size_x = int(input("Enter the size of input x[n]: "))
x = [0] * (size_x) # Declare an array with the number of elements equal to size
print ("Enter the elements of the input x[n] one-by-one as follows:-")
for i in range (0, size_x, 1):
    x[i] = input ("Element "+ str(i+1) + ": ")
print ("The entered input x[n] is :- " + str(x) + "\n")

plt. subplot (3,1,1)
plt. xlabel ('n')
plt. ylabel ('x[n]')
plt. title ('Input x{}'.format(x)) # Formats the specified value(s) and insert them inside the string's placeholder.
plt. stem (np. arange (0, size_x), x)

```

PTO

```
size_h = int(input("Enter the size of impulse response h[n]: "))
h=[0] * (size_h) # Declare an array with the number of elements equals to value of size.
print("Enter the elements of the impulse response h[n]: ")
for i in range (0, size_h, 1):
    h[i] = input("Element " + str(i+1) + ": ")
print ("The entered impulse response h[n] is :- " + str(h) + "\n")

plt.subplot (3,1,3)
plt.xlabel ('n')
plt.ylabel ('h[n]')
plt.title ('Impulse Response h{}'.format(h)) # Formats the specified value(s) and insert them inside the string's placeholder.
plt.stem (np.arange (0, size_h), h)

plt.show()


temp =[] # Duplicate for impulse response
for i in range (size_x):
    temp = h.copy () # Creates a copy of an existing list
    for j in range (len (temp)):
        temp[j] = int (temp[j])* int(x[i])
    for k in range (i):
        temp.append (0) # Adds required zeros at the end of list
        for l in range (len (temp)-1, 0,-1):
            temp[l] = temp[l-1] # Shifts the zeros at the beginning of list
        temp[k]=0 # Prepares for next iteration
    plt.title ('Intermediate plot x[{}] * h[n-{}] = {}'.format (i,i, temp)) # Formats the specified value(s) and insert them inside the string's placeholder.
    plt.xlabel ('n')
    plt.ylabel ('x[{}] * h[n-{}]'.format (i,i))
```

```
    plt.stem(np.arange(0, len(temp)), temp)
    plt.show()

# string padding refers to adding, usually, non-informative
characters to a string to one or both ends of it. This is most often
done for output formatting and alignment purposes, but it can
have useful practical applications. numpy.pad() function is used
to pad the numpy arrays.

size = (size_x + size_h) - 1  # Compute the size of system response
x = np.pad(x, (0, size - size_x), 'constant')
h = np.pad(h, (0, size - size_h), 'constant')
y = np.zeros(size, dtype = int) # Returns a new array of given
shape and type, with zeros.
iteration = 1 # Variable used for diplaying the iteration sequence

for i in range(size):
    for j in range(size):
        if i >= j:
            y[i] = int(y[i] + (int(x[i-j]) * int(h[j])))
            print("Iteration " + str(iteration) + ": " + str(y))
# Display result of each iteration
            iteration += 1
print("\n The system response is : - " + str(y) + "\n")

plt.xlabel('n')
plt.ylabel('y[n]')
plt.title('System Response y{}'.format(y)) # Formats the
specified value(s) and insert them inside the string's placeholder.
plt.stem(np.arange(0, size), y)
plt.show()
```

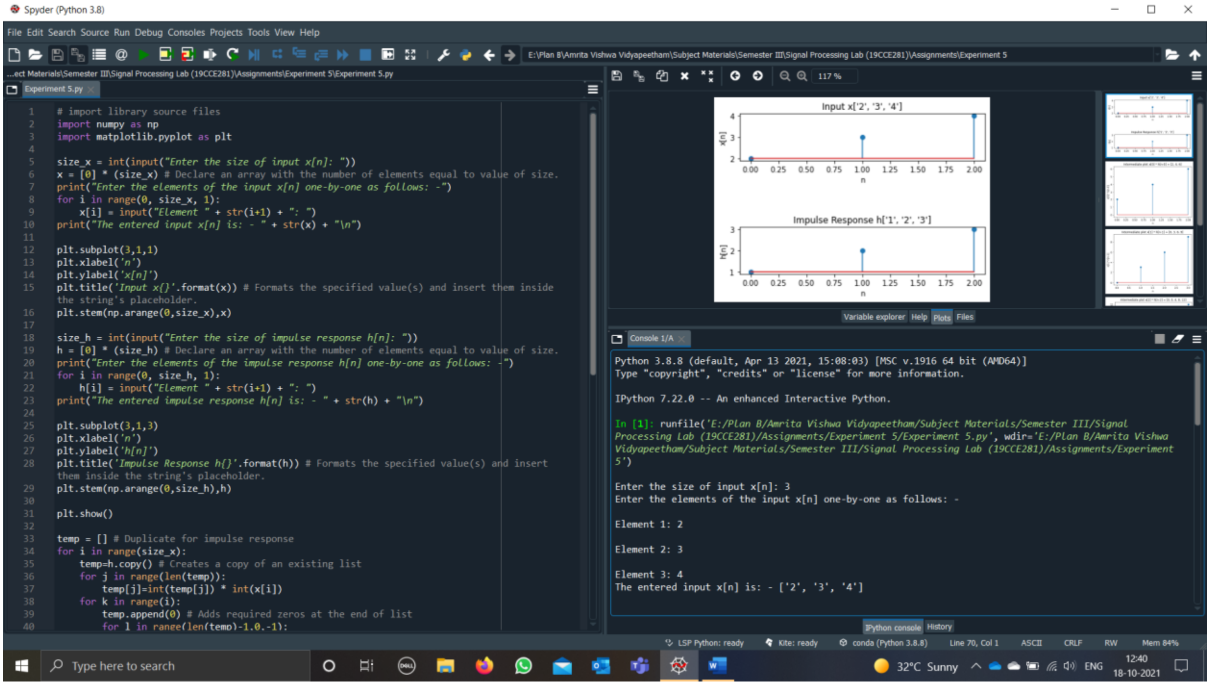
SANTOSH - [CB.EN.U4 CCE 20053]

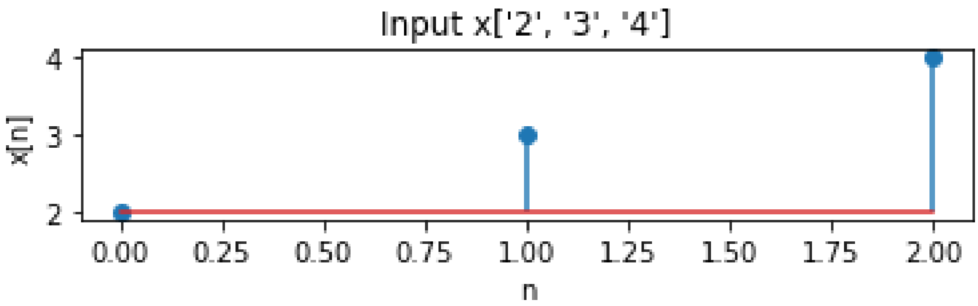
6

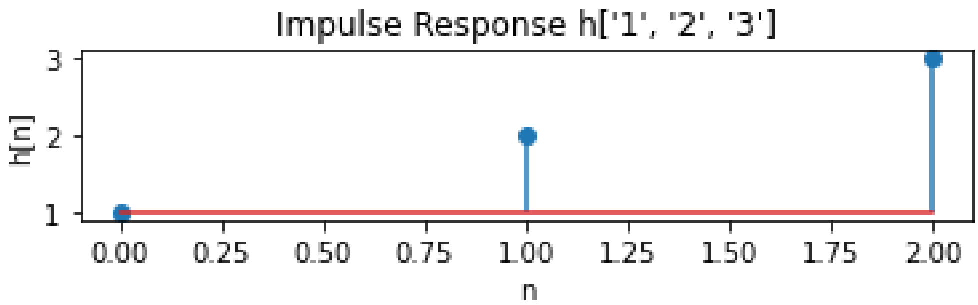
* INFERENCE:

system response y[n] calculated from input x[n] and impulse response h[n], responses sketched and results verified.

Input and Impulse Response







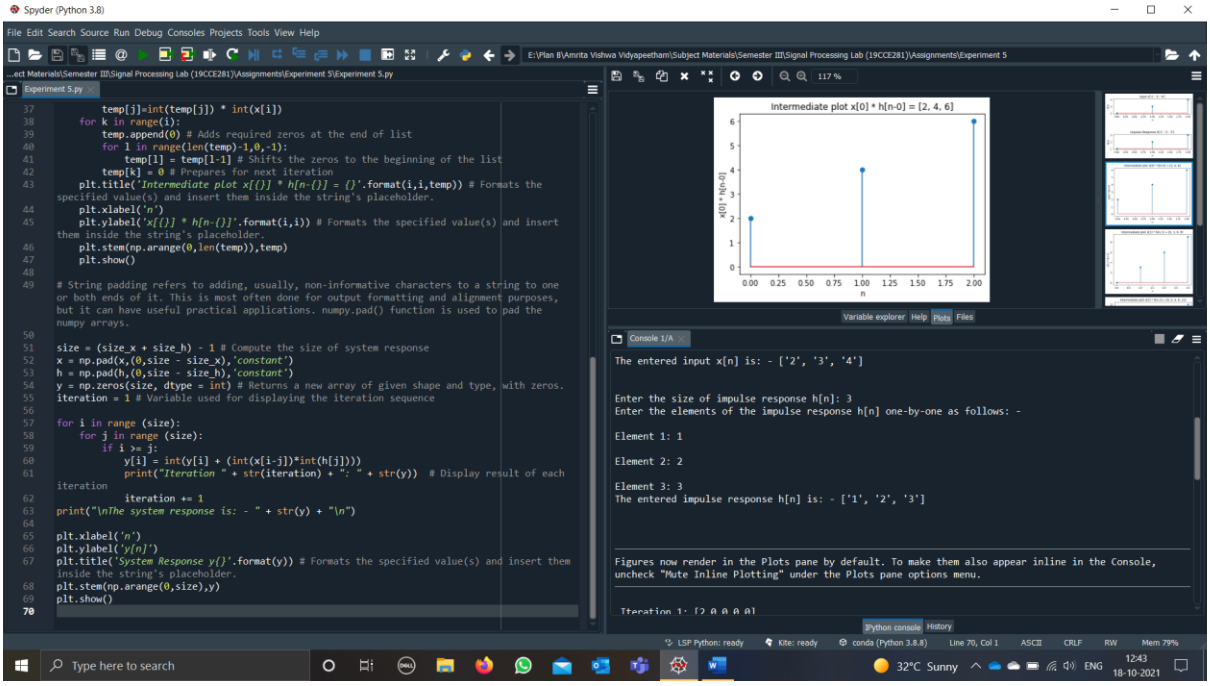
Step 1: Enter the size of input x[n] and declare an array with the number of elements equal to the value of size.

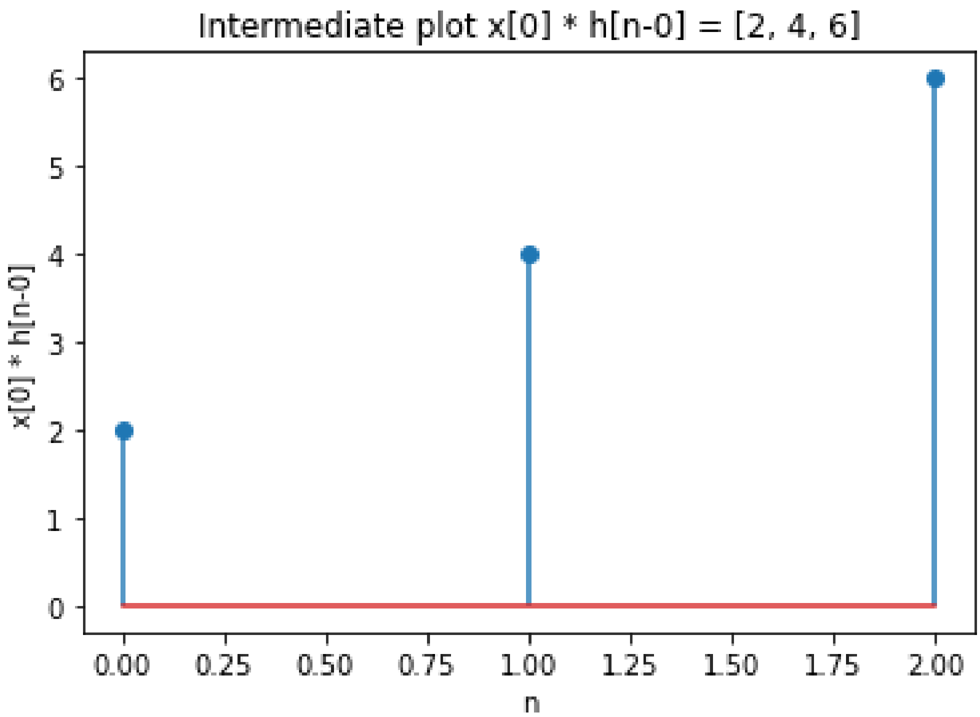
Step 2: Enter the elements of the input x[n].

Step 3: Show the labelled plot x[n].

Step 4: Repeat steps 1, 2 and 3 for impulse response h[n].

Display Intermediate Plot





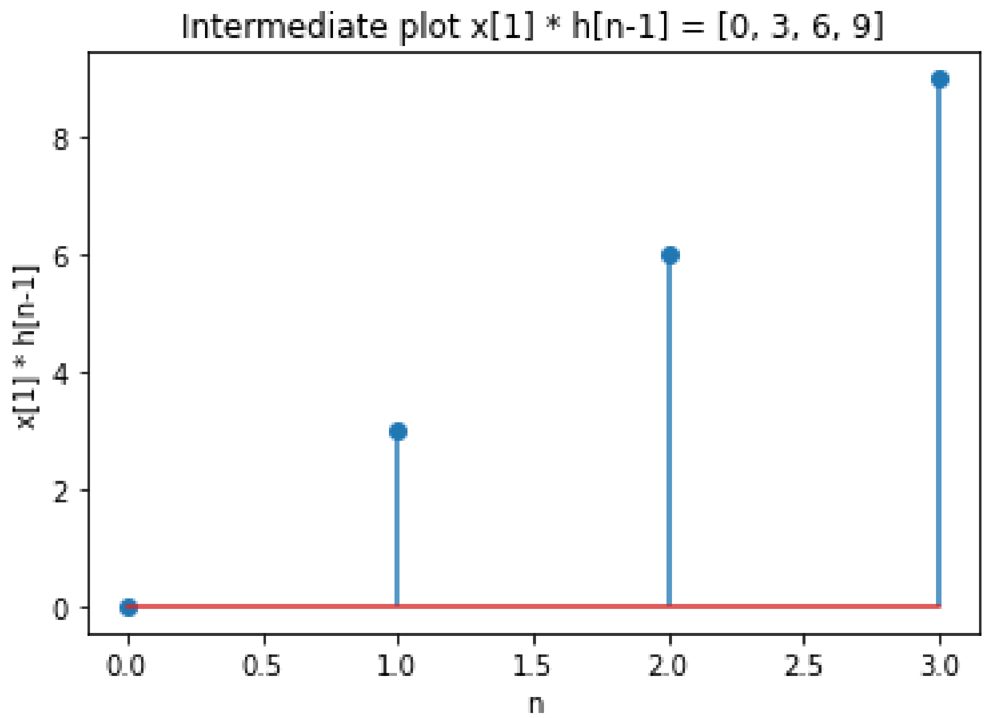
Step 1: Create a duplicate list temp[] and copy the elements impulse response to the duplicate list.

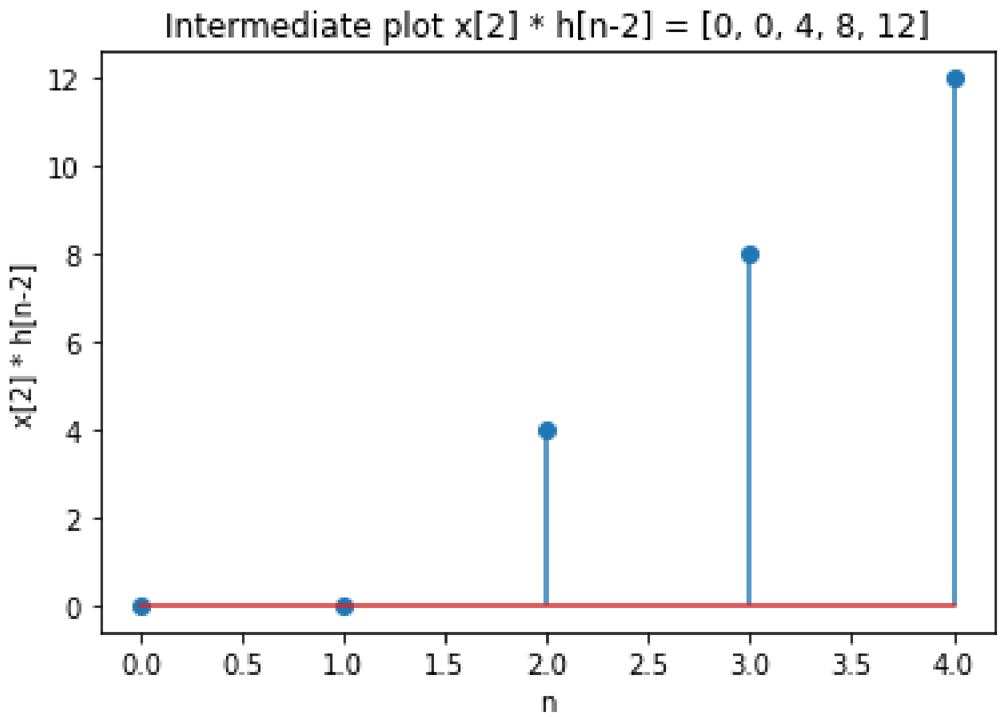
Step 2: Perform convolution sum integral (x[n] * h[n]).

Step 3: Adds required zeros at the end of the list.

Step 4: Shifts the elements to bring the above-added zeros to the beginning of the list.

Step 5: Show the labelled intermediate plot.





For the given input x[n] = [2, 3, 4] and impulse response = h[n] = [1, 2, 3], we have three intermediates:

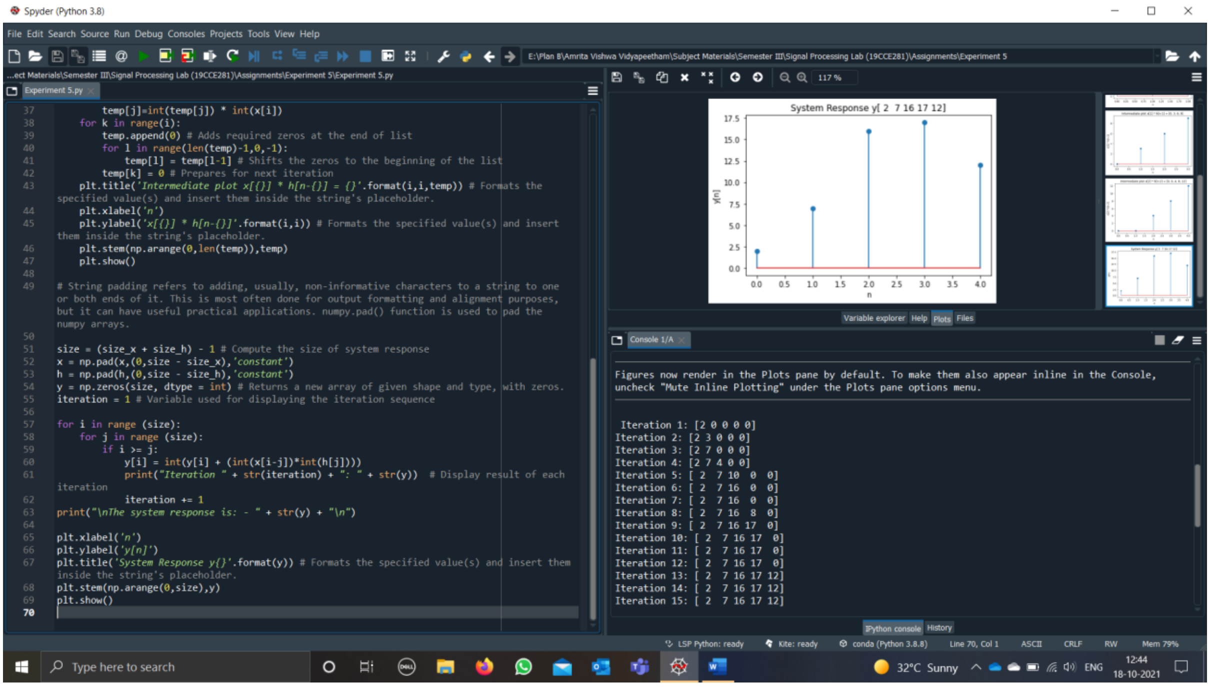
x[0] * h[n] = [2, 4, 6] for i = 0

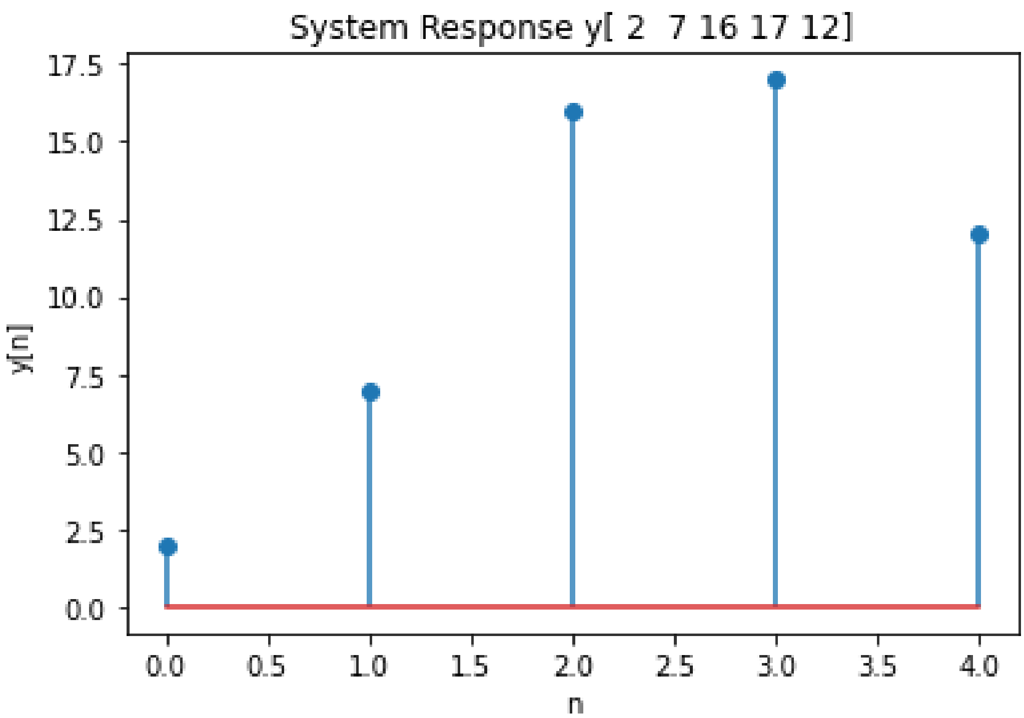
x[1] * h[n-1] = [0, 3, 6, 9] for i = 1

x[2] * h[n-2] = [0, 0, 4, 8, 12] for i = 2

Upon adding these, we compute that y[n] = [2, 7, 16, 17, 12]

Iterations and System Response





Step 1: Compute the size of the system response and pad for input x[] and impulse response h[].

Step 2: Initialize all the elements of system response y[n] as zeros.

Step 3: Using two nested for loops, perform convolution sum integral (x[n] * h[n]).

Step 4: Print the result of each iteration.

Step 5: Show the labelled plot for system response.

```
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 5/Experiment 5.py'       ='E:/Plan B/Amrita Vishwa Vidyapeetham/Subject Materials/Semester III/Signal Processing Lab (19CCE281)/Assignments/Experiment 5'

Enter the size of input x[n]: 3
Enter the elements of the input x[n] one-by-one as follows: -

Element 1: 2

Element 2: 3

Element 3: 4
The entered input x[n] is: - ['2', '3', '4']


Enter the size of impulse response h[n]: 3
Enter the elements of the impulse response h[n] one-by-one as follows: -

Element 1: 1

Element 2: 2

Element 3: 3
The entered impulse response h[n] is: - ['1', '2', '3']

Iteration 1: [2 0 0 0 0]
Iteration 2: [2 3 0 0 0]
Iteration 3: [2 7 0 0 0]
Iteration 4: [2 7 4 0 0]
Iteration 5: [ 2  7 10  0  0]
Iteration 6: [ 2  7 16  0  0]
Iteration 7: [ 2  7 16  0  0]
Iteration 8: [ 2  7 16  8  0]
Iteration 9: [ 2  7 16 17  0]
Iteration 10: [ 2  7 16 17  0]
Iteration 11: [ 2  7 16 17  0]
Iteration 12: [ 2  7 16 17  0]
Iteration 13: [ 2  7 16 17 12]
Iteration 14: [ 2  7 16 17 12]
Iteration 15: [ 2  7 16 17 12]

The system response is: - [ 2  7 16 17 12]


In [2]:
```