School of Sciences and Engineering

Digital Signal Processing

PROJECT

Omar Elsayed - 900183884

Kareem A. Mohammed Talaat - 900192903

Sherif Sakran - 900193609

CSCE361101

Dr. Hossam Hassan

Spring 2022

Contents

I - Problem Definition and Importance

It is required to implement the continuous time convolution, discrete time convolution, and the filter effect using the discrete time fourier transform. Convolution is used heavily in systems. It has a special significance in the context of LTI systems. To simplify it, convolution is some sort of a mathematical method of getting a response signal using a system and an input to that system. The system is described by the impulse response. Convolution relates the input signal, the impulse response, and the output signal. The input can be demuxed into an array of impulses (which are shifted and scaled delta functions). Moreover, the output from each impulse is also a scaled and shifted version of the impulse response, so we can get the output signal by adding those shifted and scaled impulse responses. Based on that idea, we can calculate what the output signal will be, if we have the impulse response of the system, for any input signal using convolution. This allows us to do a lot of important life calculations, for example, Image processing, as we can do a lot of operations on a certain image using a certain Kernel (system's impulse response) some of which are sharpening the picture, blurring, enhancing, etc.. Convolution implementation also allows us to do signal filtering. It also allows us to do Audio processing to enhance the quality of the sound greatly and

remove background noise, etc. Convolution also is used in artificial intelligence, optics, probability theory, computed tomography, and the list goes on!

II- Methods and Algorithms

Part 1 and 2:

Convolution was implemented to convolute an input signal with an impulse response and produce an output signal. The user selects those signals from a pre-defined set of signals or inputs the values in case of the discrete time signals. After selecting/inserting the input signal, we faced a number of problems to preprocess the signal before they underwent the convolution process.

The first challenge that faced us was how to simulate the signal reflection on y-axis to simulate this operation h(-n) easily without much complication. The solution that we used is that the impulse response signal was reversed so that the array that stores the signal became of this shape: h[n-1 … 0] instead of h[0 …. N-1]. After that, the second challenge that we faced is that the input signal and the impulse response cannot be convoluted right away, however, the input signal should be padded with zeros from left and right with length equal to the length of the impulse respons array - 1. This problem was solved using this approach:

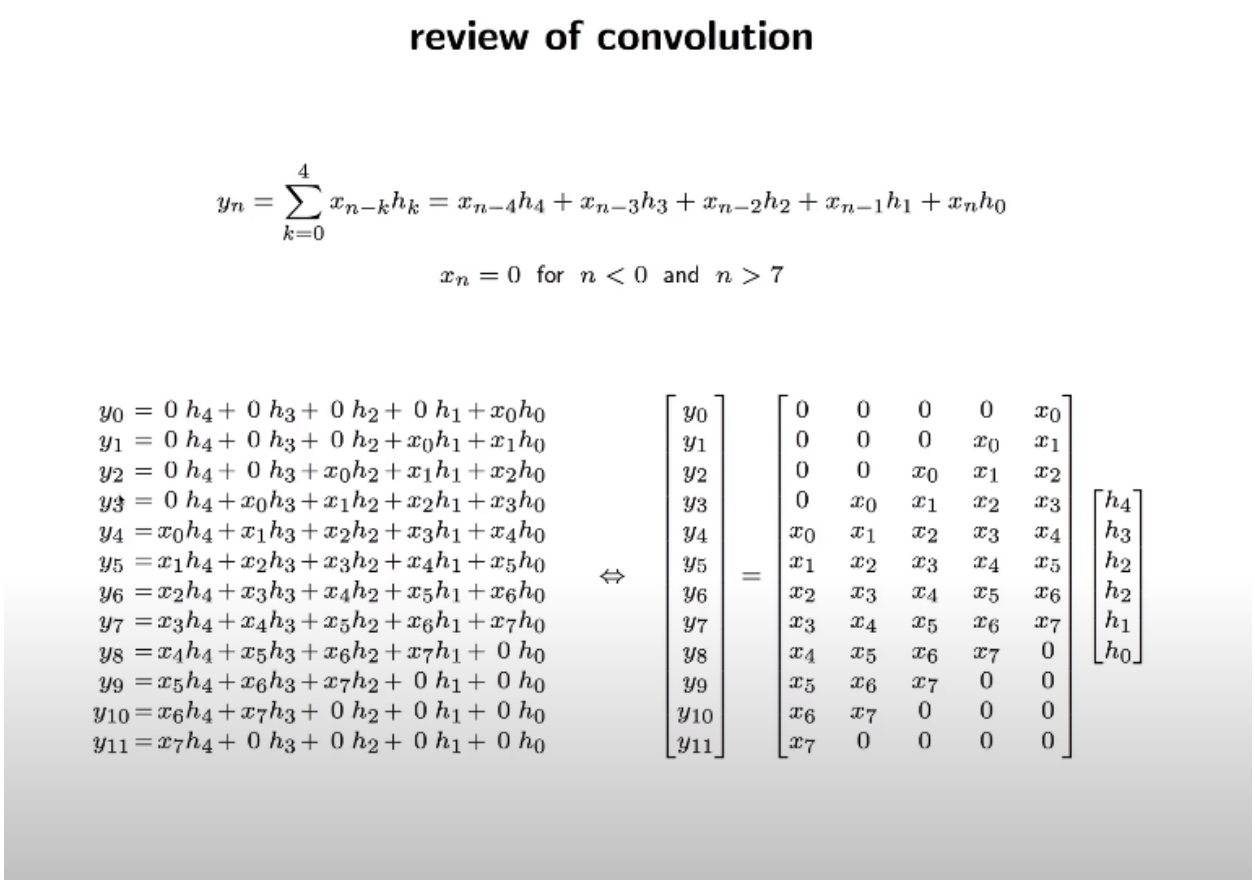
[0 .. 0 | x[0] … x[Lx - 1] | 0 .. 0]

[Lh - 1 | Lx | Lh - 1]

After this point, the two signals were ready to to be convoluted together. The convolution itself was done by applying matrix multiplication using a simple nested loop of complexity O(n^2) that implements the following equation.

```
for i in range(Ly):
    for j in range(Lh):
        y[i] += h_impulse[j] * x_signal[i+j]
```

The outer loop iterates over every element in the output array y[n] while the inner loop iterates over each term in the equation of y[n] which contains x[n] and h[n] that are multiplied together and then summed up to get the final result.



Design and Functionality

The flow of the program is very user-friendly. It asks the user to choose whether to select a ready-made signal or to insert a signal values himself. If the user chooses to select one of the pre-defined functions in the program, the program asks the user questions based on the type of the function to get the needed parameters to implement this function. For example, if the user chooses to use the unit step function, then the program prompts a question to get the shift value of the t0 and then an array is populated with the value of the function, namely 1 in this case, from the point after the t0 while all the values before the t0 will have the value of zero. However, if the user chooses the rectangle function, then a prompt asks him/her to input the starting limit, the ending limit, and the amplitude of the rectangle signal and then an array is populated with this amplitude value within the indices values that lie in the difference between the starting limit and the ending limit. If the user chooses the unit impulse/sample function, then a prompt asks to enter the shift value of the signal and the input signal is shifted with the same amount of this shift value. After the whole process, all the the three functions are visualized: y[n], h[n], and x[n] using matplotlib library in python using a graph plot.

Filters

As shown in figure 1, we can see the different equations to obtain the Impulse response hd[n] for the different types of filters. We implemented these frequency response functions which are then used in a convolution step with the input signal x[n] to obtain the filtered output y[n]. Filtering signals is done according to the type of the filter which is dependent on its impulse response.

Type of filter	Frequency response $h_d[n]$
low-pass filter	$h_d[n] = \begin{cases} \frac{\sin[\omega_c(n-M)]}{\pi(n-M)}; & n \neq M \\ \frac{\omega_c}{\pi}; & n = M \end{cases}$
high-pass filter	$h_d[n] = \begin{cases} 1-\frac{\omega_c}{\pi}; & n \neq M \\ -\frac{\sin(\omega_c(n-M))}{\pi(n-M)}; & n = M \end{cases}$
band-pass filter	$h_d[n] = \begin{cases} \frac{\sin(\omega_{c2}(n-M))}{\pi(n-M)} - \frac{\sin(\omega_{c1}(n-M))}{\pi(n-M)}; & n \neq M \\ \frac{\omega_{c2}-\omega_{c1}}{\pi}; & n = M \end{cases}$
band-stop filter	$h_d[n] = \begin{cases} \frac{\sin(\omega_{c1}(n-M))}{\pi(n-M)} - \frac{\sin(\omega_{c2}(n-M))}{\pi(n-M)}; & n \neq M \\ 1-\frac{\omega_{c2}-\omega_{c1}}{\pi}; & n = M \end{cases}$

Figure 1

To implement the filters, we would do the following:

- User chooses the type of filter to be used and enters its parameters.
- Import the math library to calculate the complex equations as shown in figure 1.
- We would ask the user the number of samples N.
- A for loop is then used to calculate h[n] for the chosen number of samples.

- After obtaining the h[n], we convolute it with the specified input signal x[n] to obtain the output signal y[n]

III- Experimental Results

Part 1: Continuous (unit step signal, unit impulse)

Test 1:

The user selected a rectangle for the input signal and an impulse response of unit impulse without a shift.

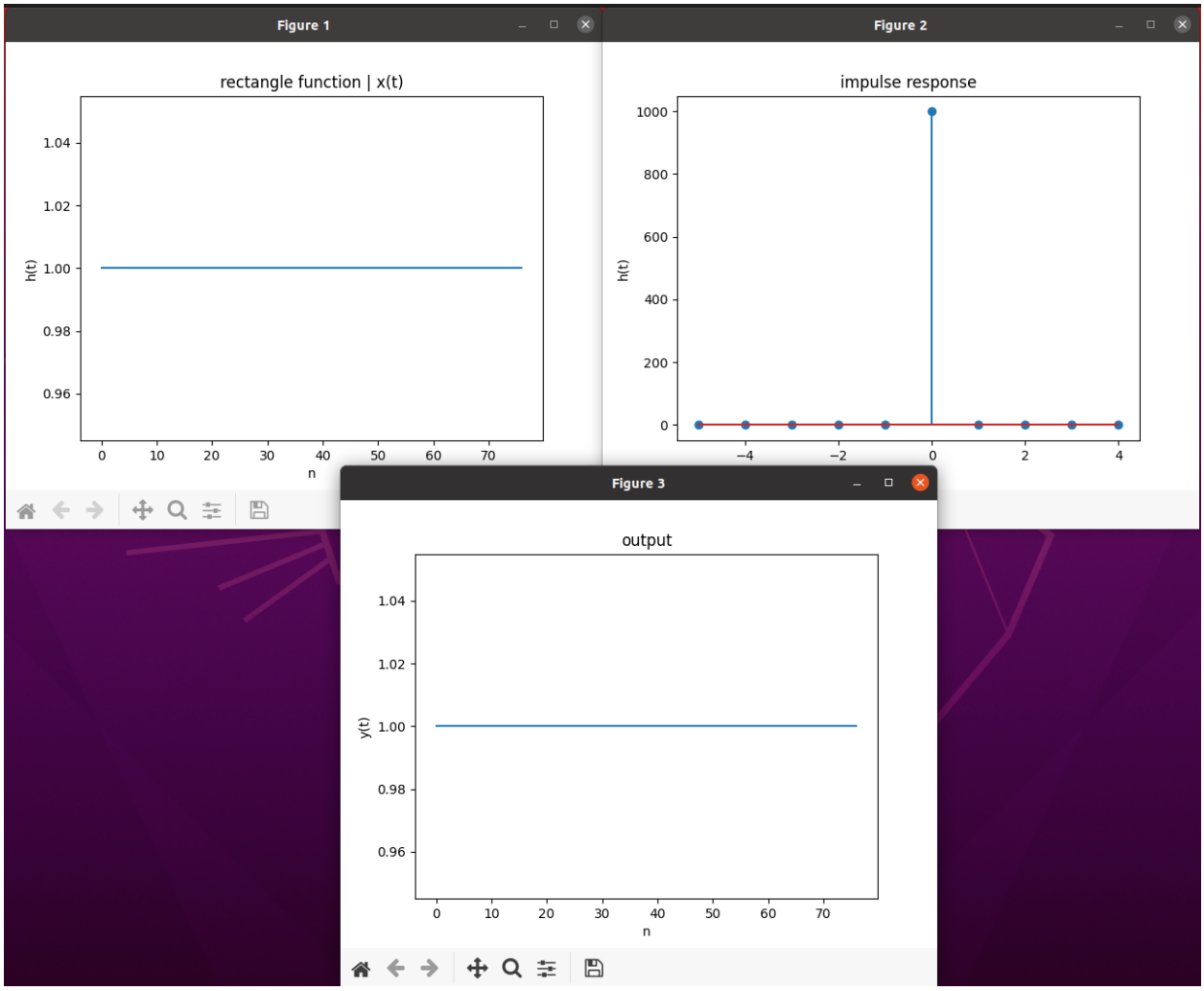
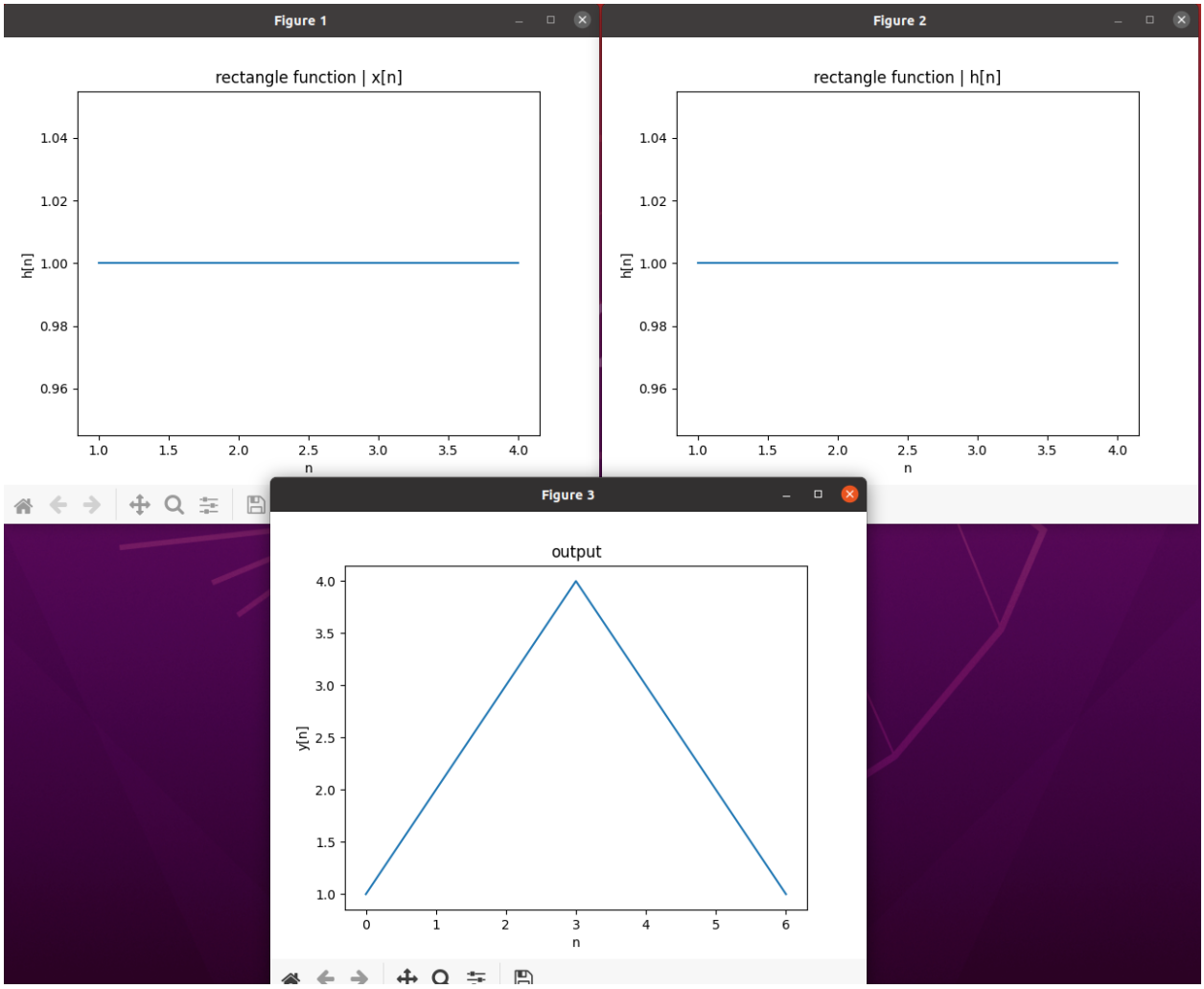


Figure 1
rectangle function | x(t)
h(t)
n
Figure 2
impulse response
h(t)
Figure 3
output
y(t)
n

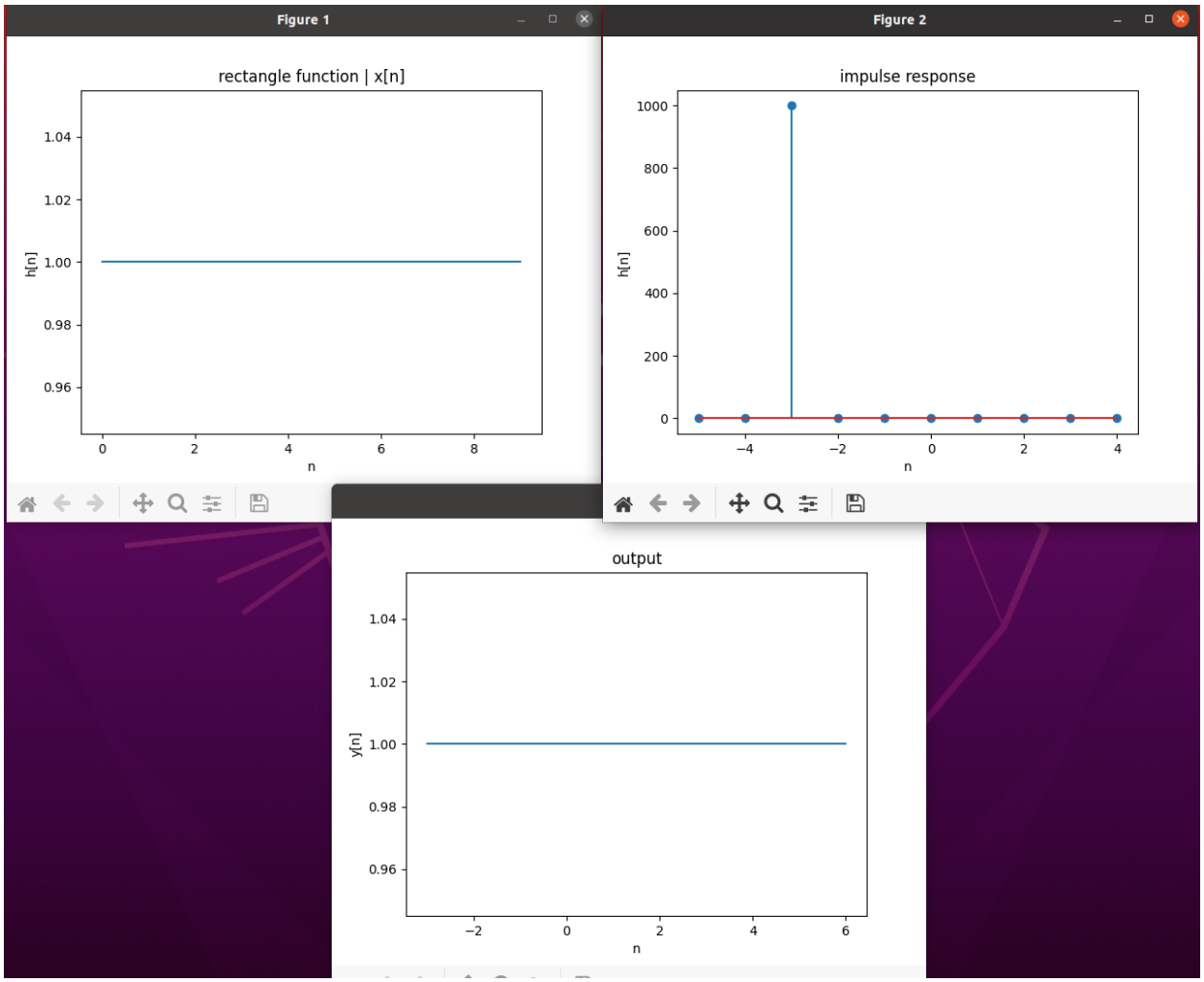
Test 2:

The user selected two identical rectangles for the input and impulse response of amplitude 1.



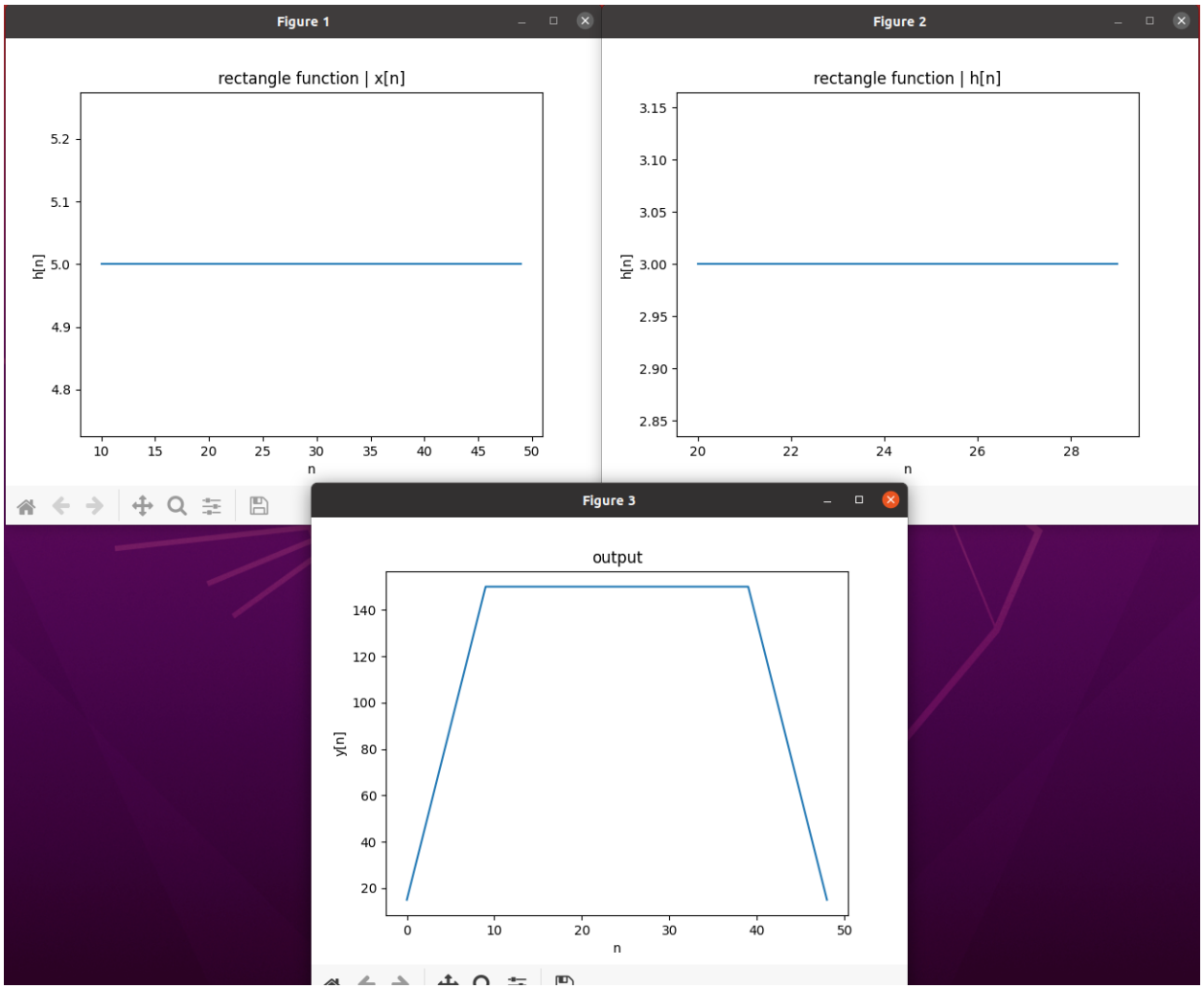
Test 3:

The user selected a rectangle for the input signal and an impulse response of unit impulse with a shift of 3.



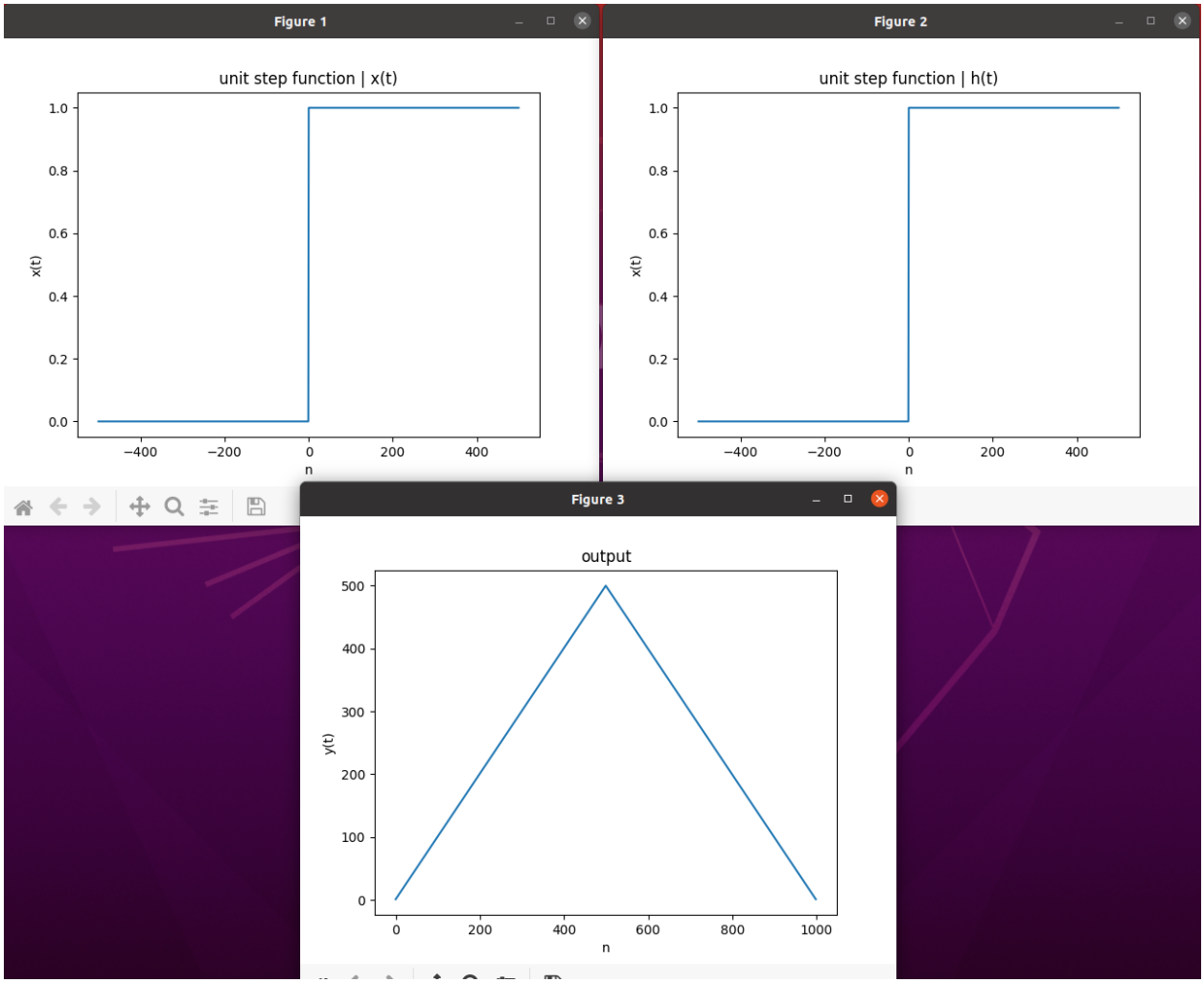
Test 4:

The user selected a rectangle for the input signal that starts from 10 to 50 with amplitude = 5 and another that starts from 20 to 30 with amplitude = 3.



Test 5:

The user selects both the input signal and impulse response to be the unit step function.

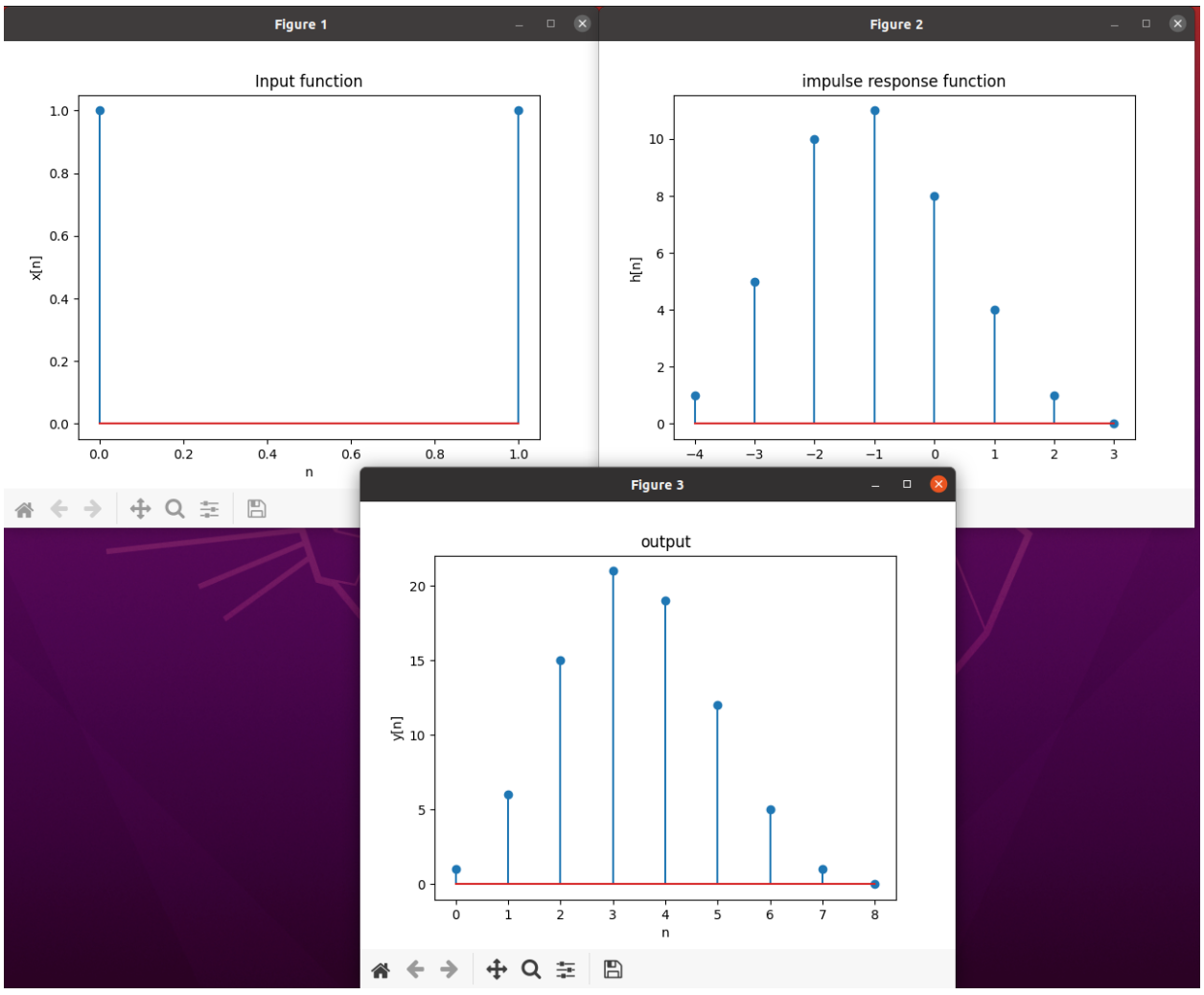


This was supposed to be generating an increasing function without any further decrease, due to the nature of unit step sequence of being extended to infinity. However, due to a limitation of the length of the list that we assumed of 1000, the output decreased after reaching the maximum length of the signal. That is, the output signal should be.

Part 2: Discrete (Unit step sequence, unit sample signal)

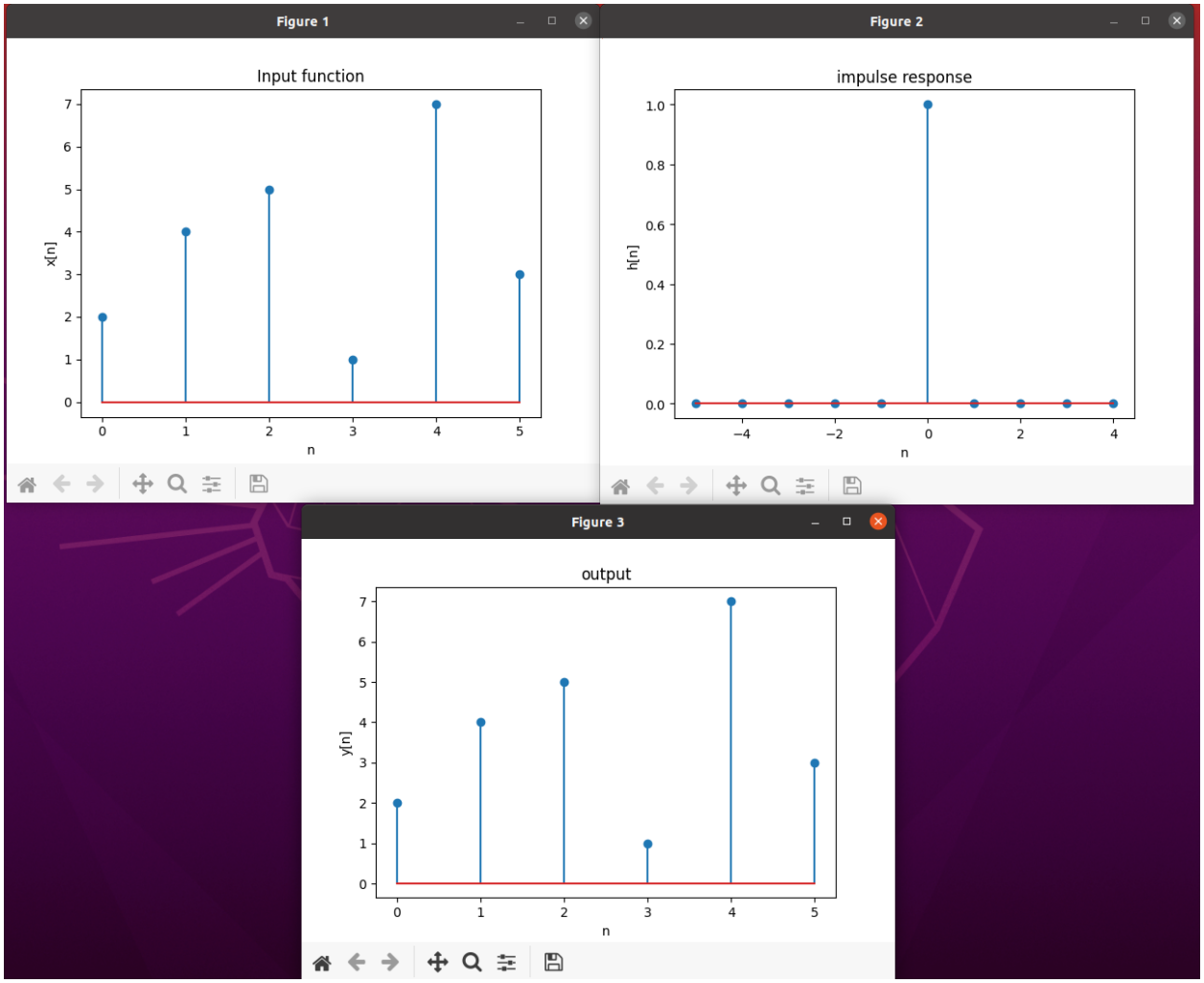
Test 1:

Data was entered by the user for both the signal and impulse response. (the midterm problem)



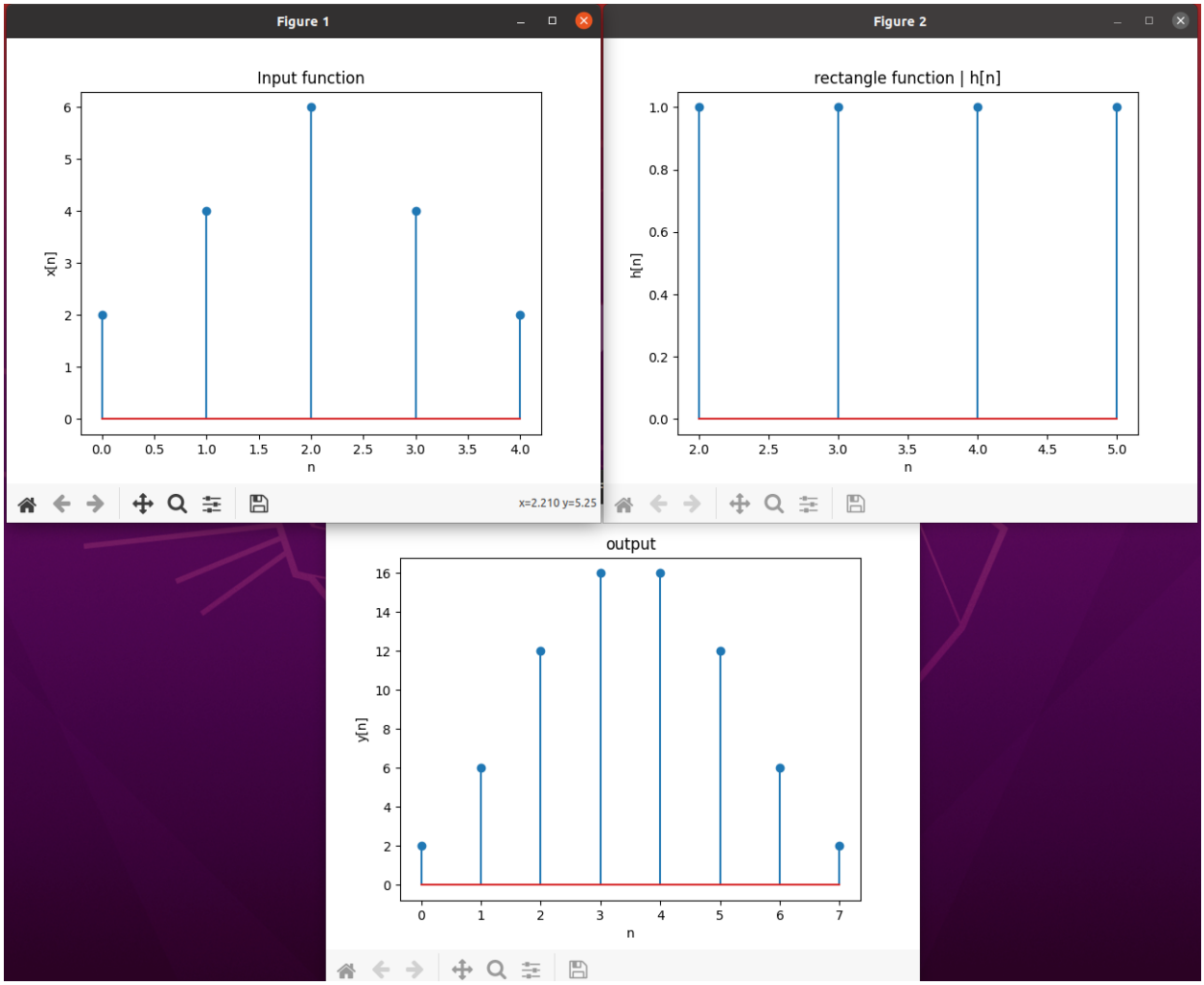
Test 2:

Input signal was entered by the user, and they selected the impulse response as the unit impulse signal. x [n] = [2, 4, 5, 1, 7, 3]



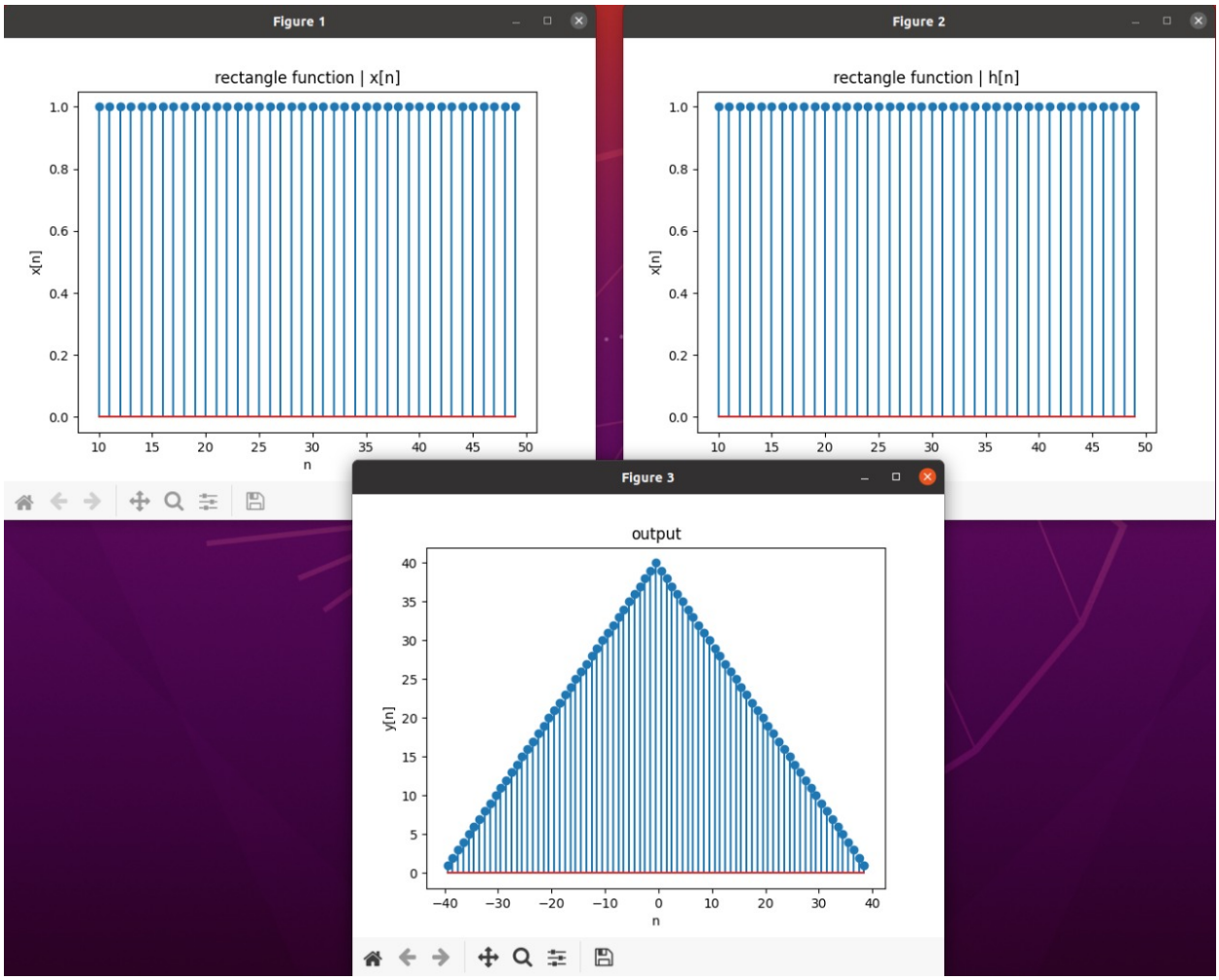
Test 3:

Input signal was entered by the user, and they selected the impulse response as a rectangle. x [n] = [2, 4, 6, 4, 2]



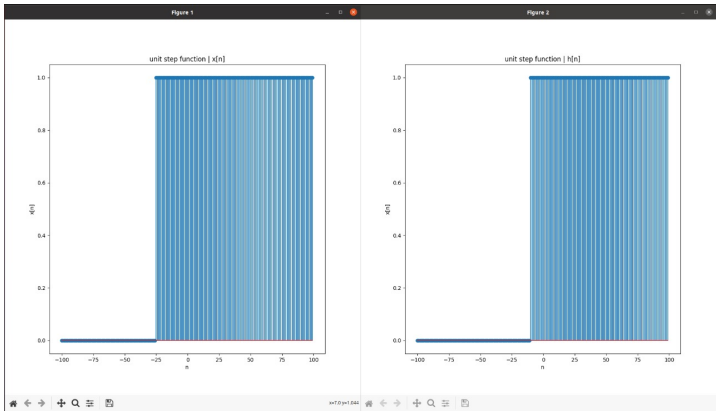
Test 4:

The user selected both the input signal and impulse response to be identical rectangles ranging from 10 to 50 with an amplitude of 1. The output is a triangle as expected.

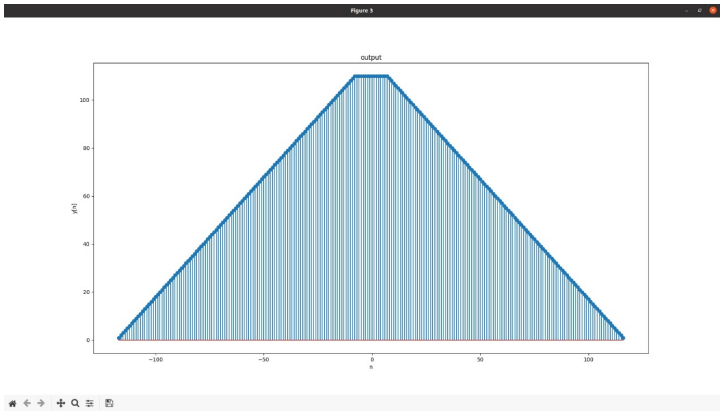


Test 5:

The user selects both the input and impulse response to be unit step sequences.

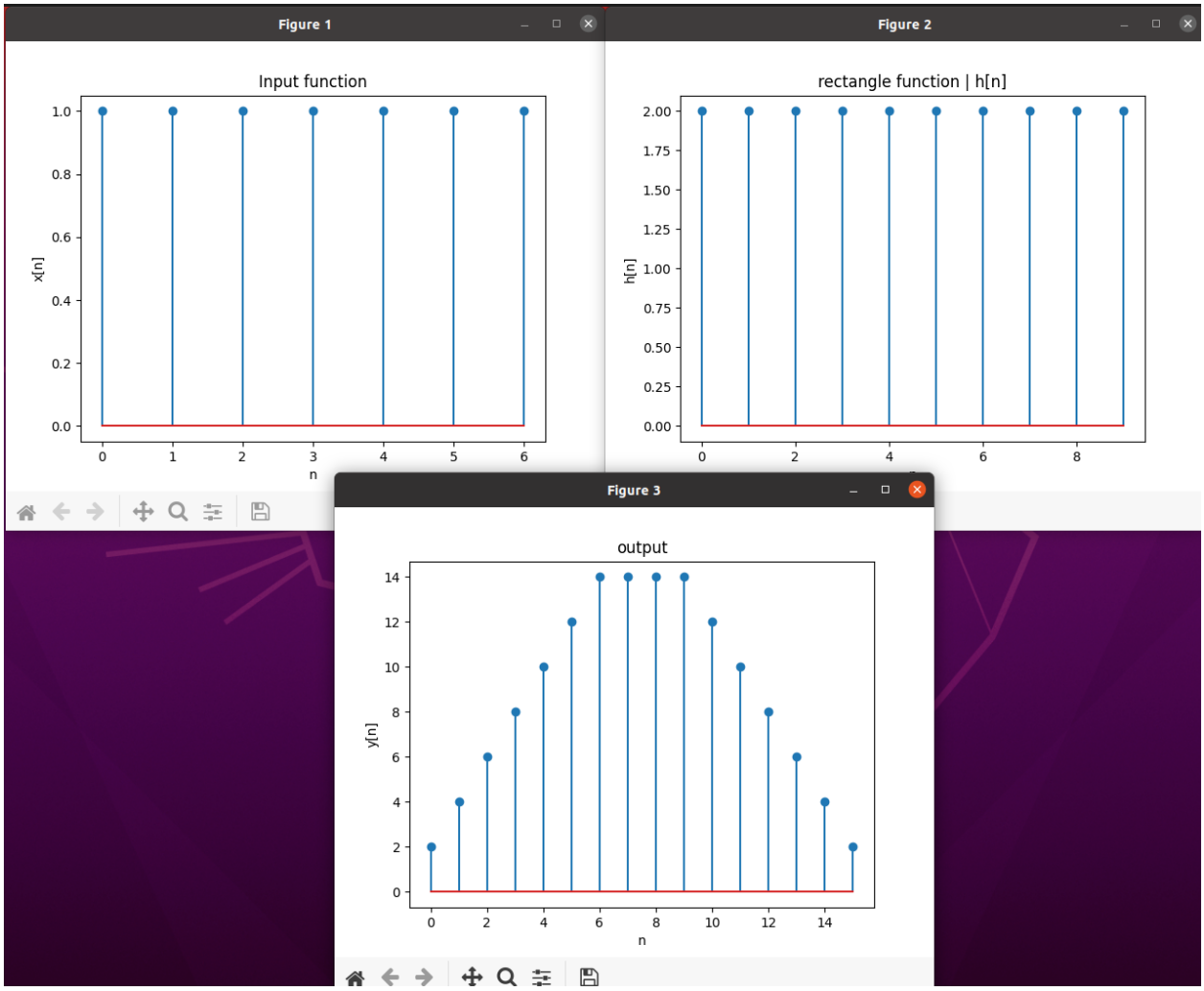


This was supposed to be generating an increasing function without any further decrease, due to the nature of unit step sequence of being extended to infinity. However, due to a limitation of the length of the list that we assumed of 1000, here was the output.



Test 6:

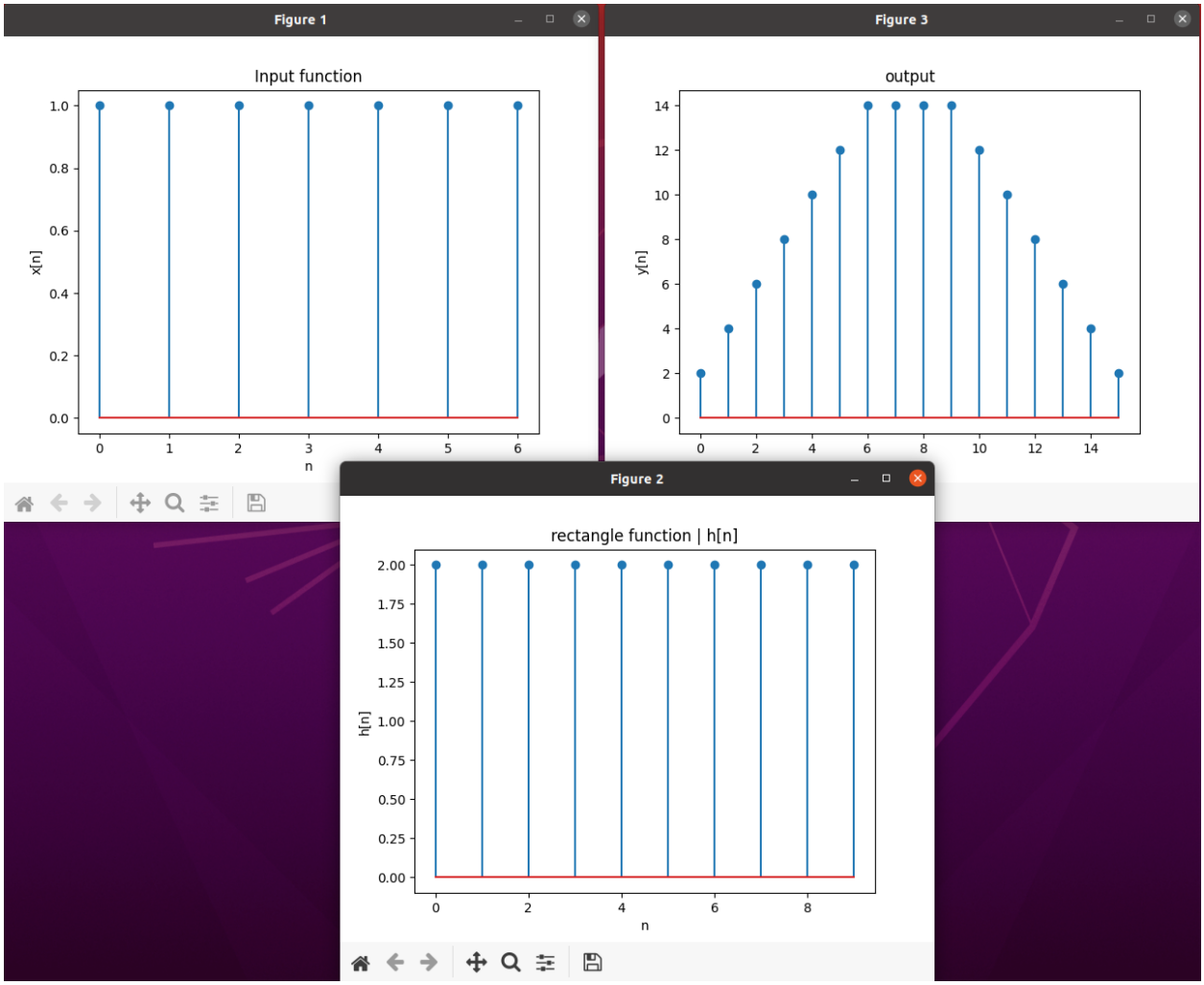
The user entered the signal as a train of pulses (1, 1, 1, 1, 1, 1, 1) and selected the impulse response to be a rectangle that starts from 0 to 9 of amplitude 2.



Test 7:

Data was entered by the user for both the signal and impulse response.

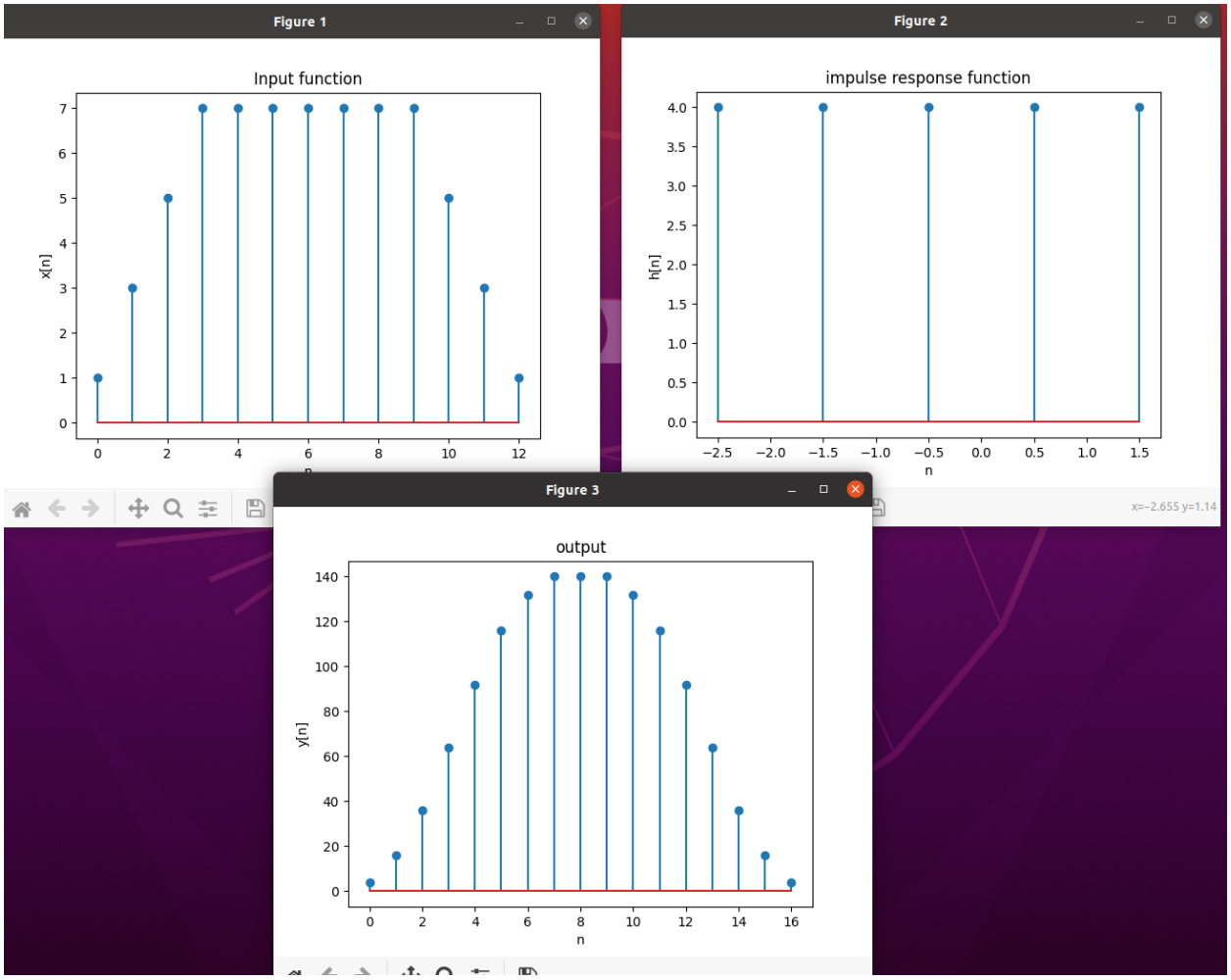
X[n] = 2 4 6 8 6 4 2, h[n] = 4, 4, 4, 4, 4



Test 8:

Data was entered by the user for both the signal and impulse response.

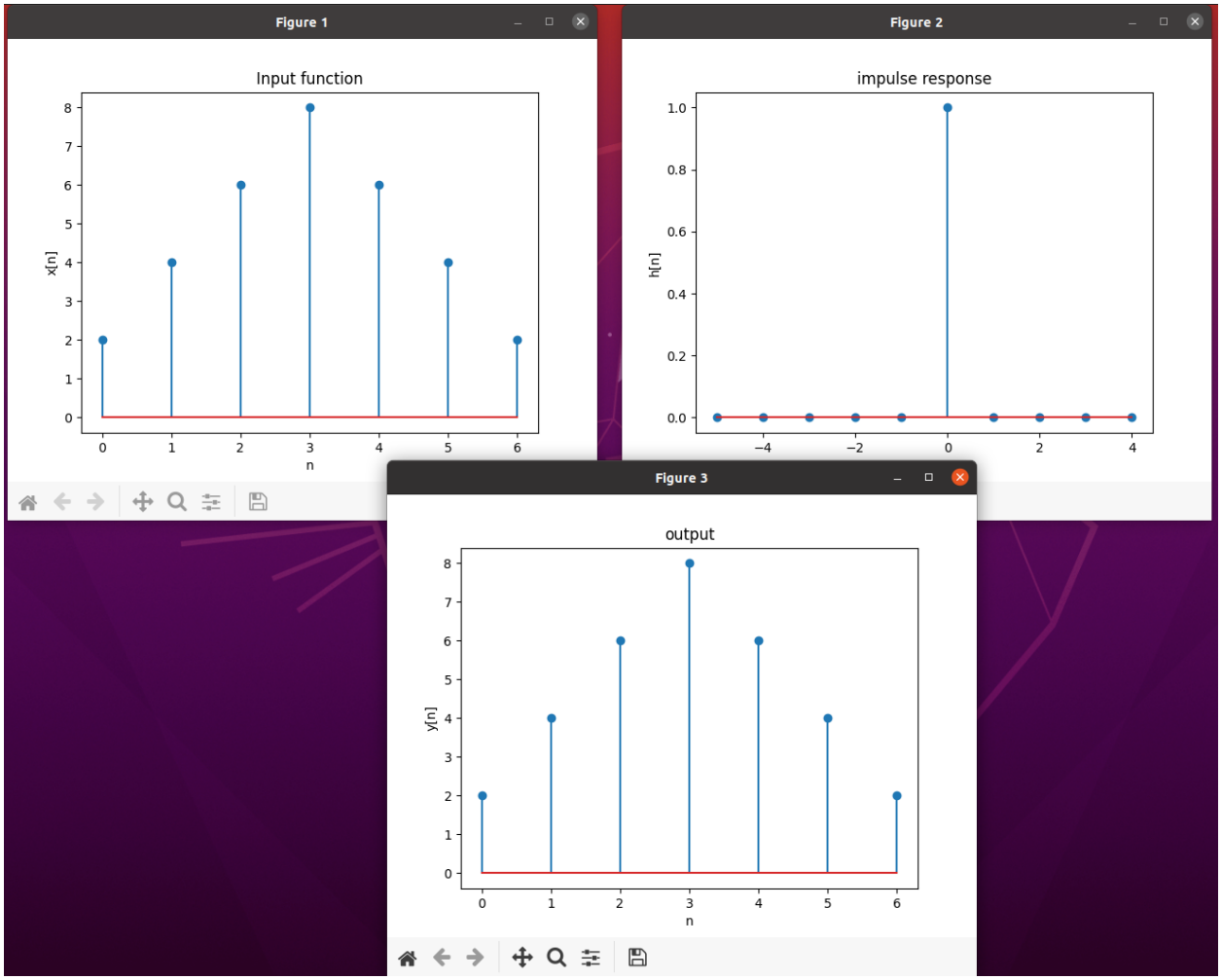
X[n] = 1 3 5 7 7 7 7 7 7 7 5 3 1, h[n] = 4, 4, 4, 4, 4



Test 9:

Data was entered by the user for the input signal while the impulse response was chosen to be the unit sample signal.

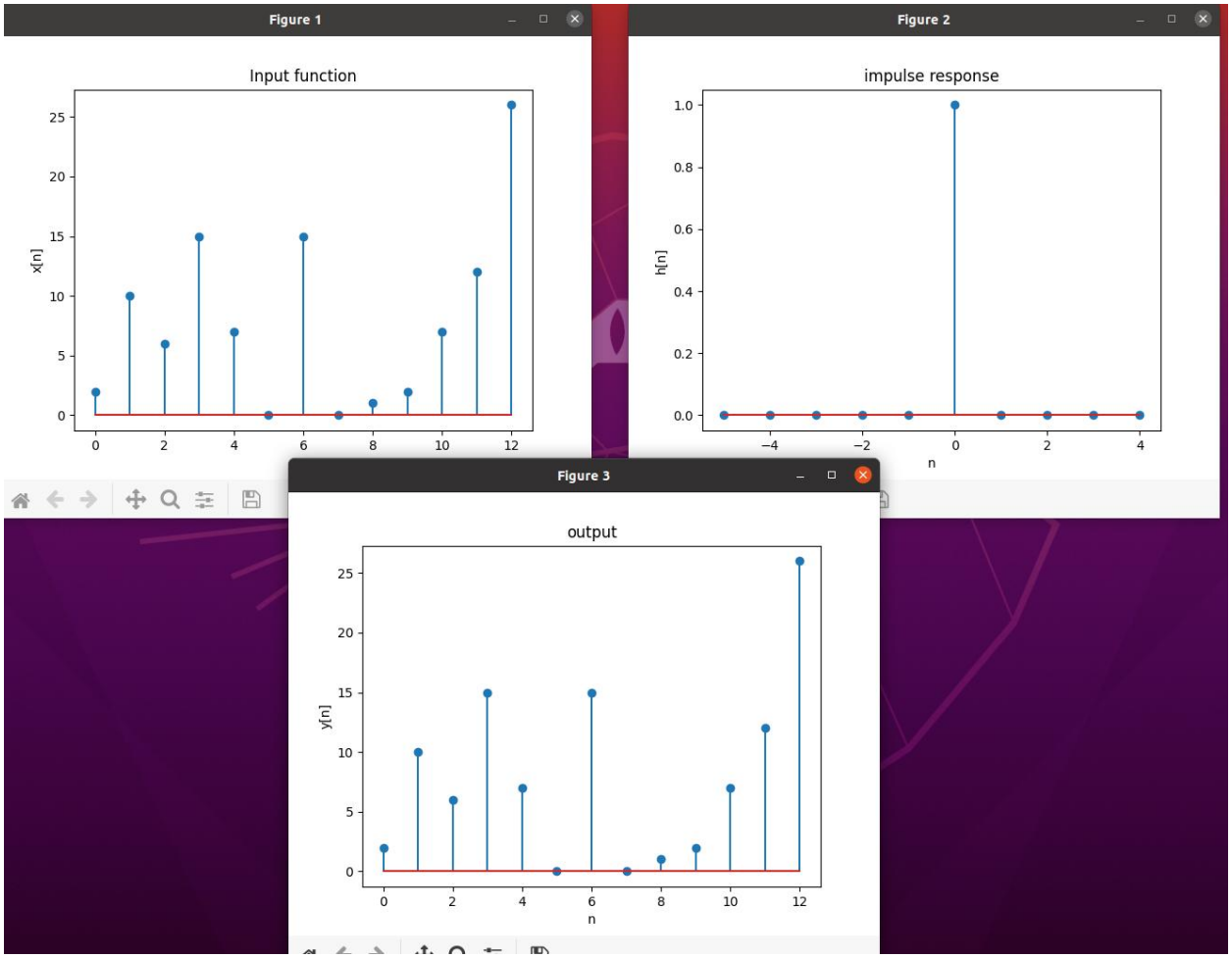
X[n] = 2 4 6 8 6 4 2, h[n] = delta[t]



Test 10:

Data was entered by the user for the input signal while the impulse response was chosen to be the unit sample signal.

X[n] = 2 10 6 15 7 0 15 0 1 2 7 12 26 , h[n] = delta[t]

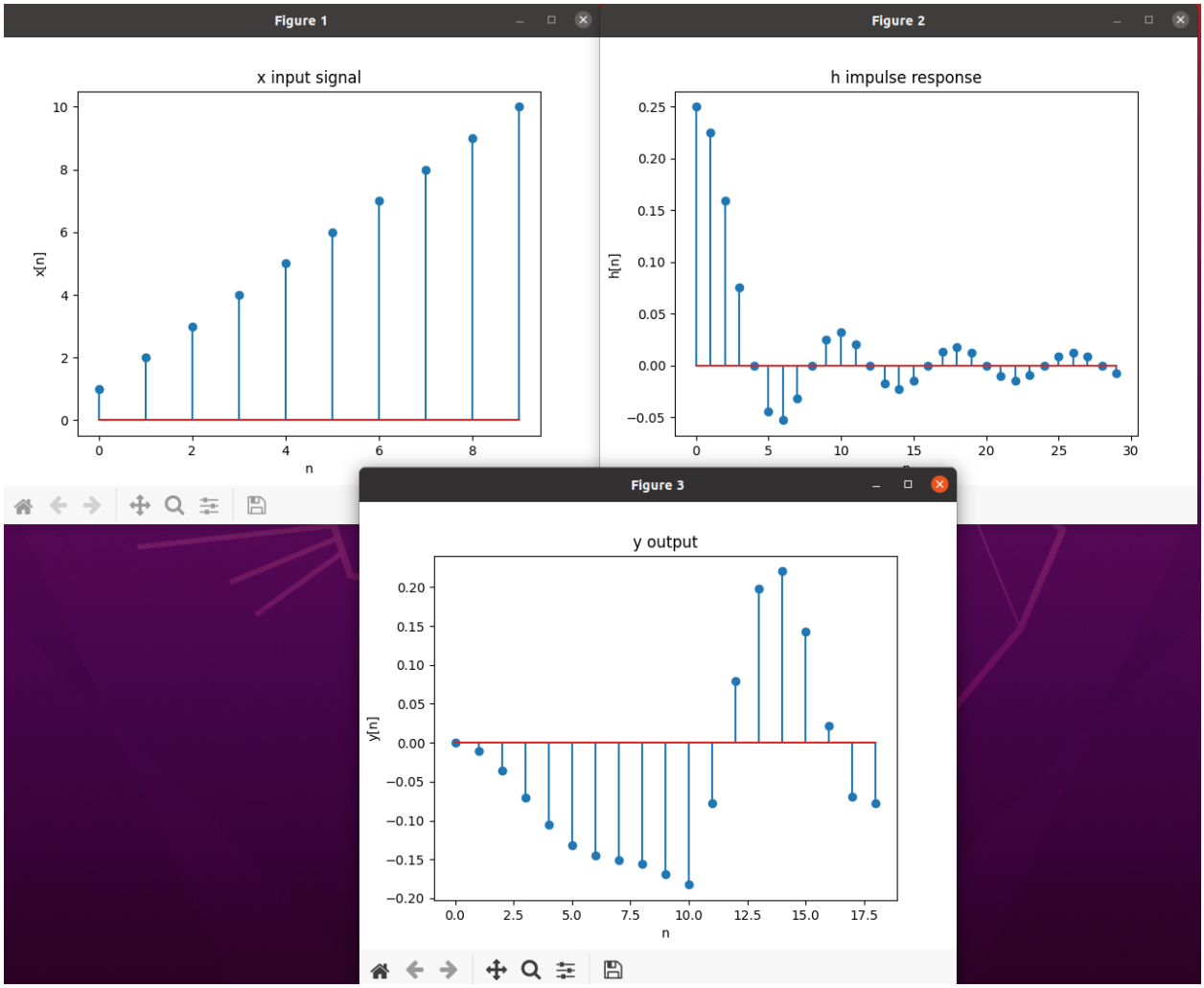


Part 3

Test 1:

Low Pass Filter

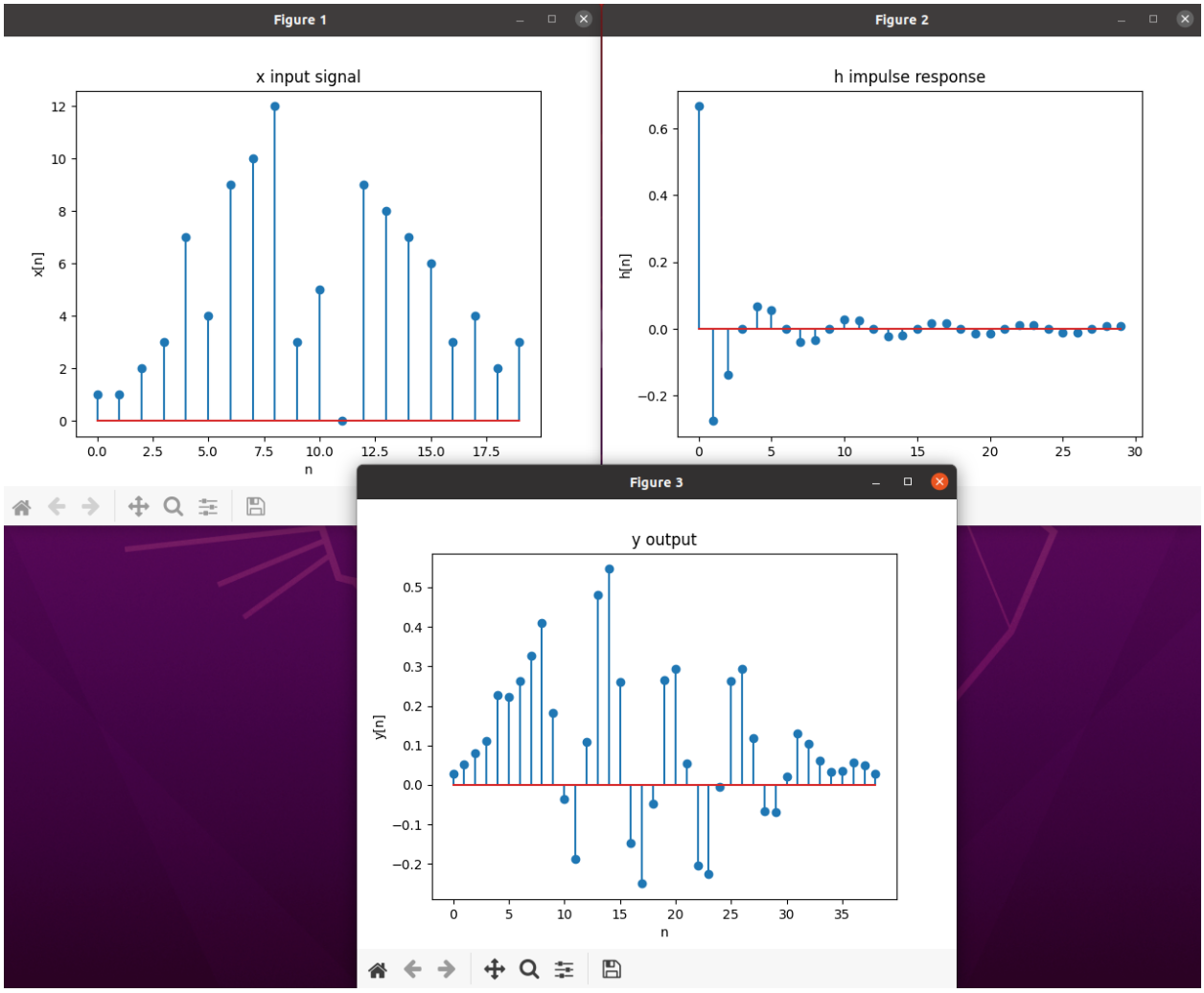
N = 8



Test 2:

High Pass Filter

N= 6

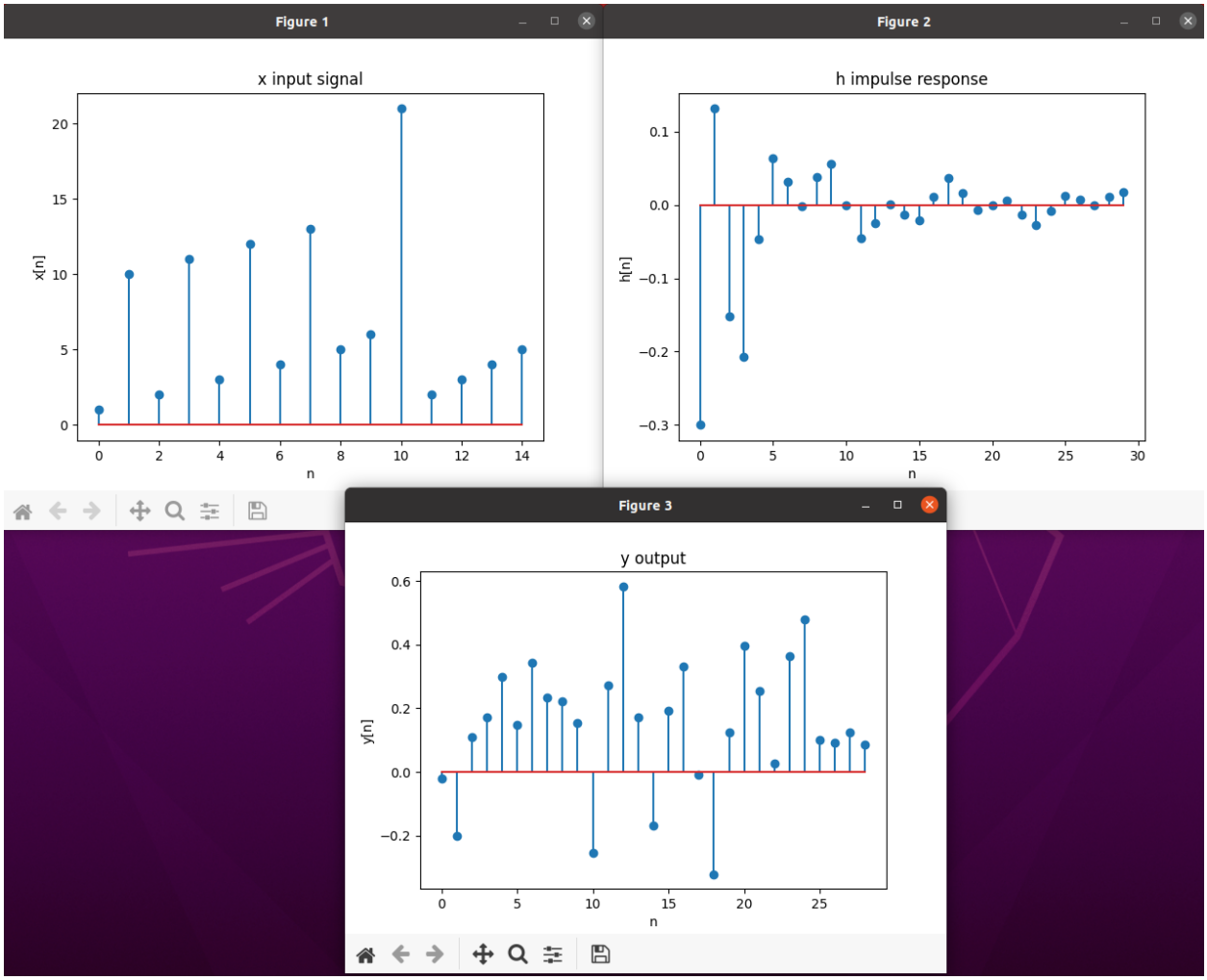


Test 3:

Band-pass filter

N1 = 2

N2 = 8

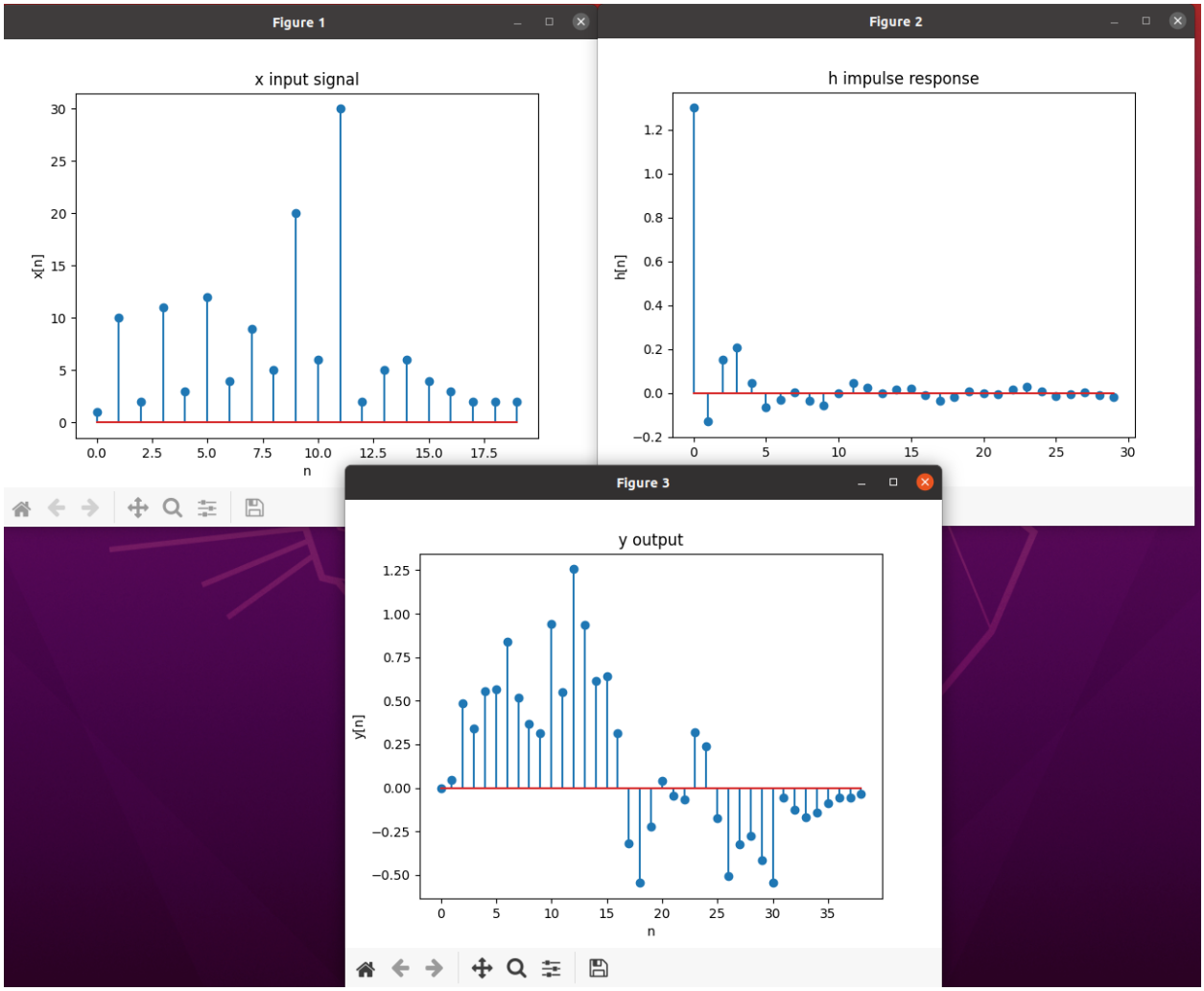


Test 4:

Band-stop filter

N1 = 4

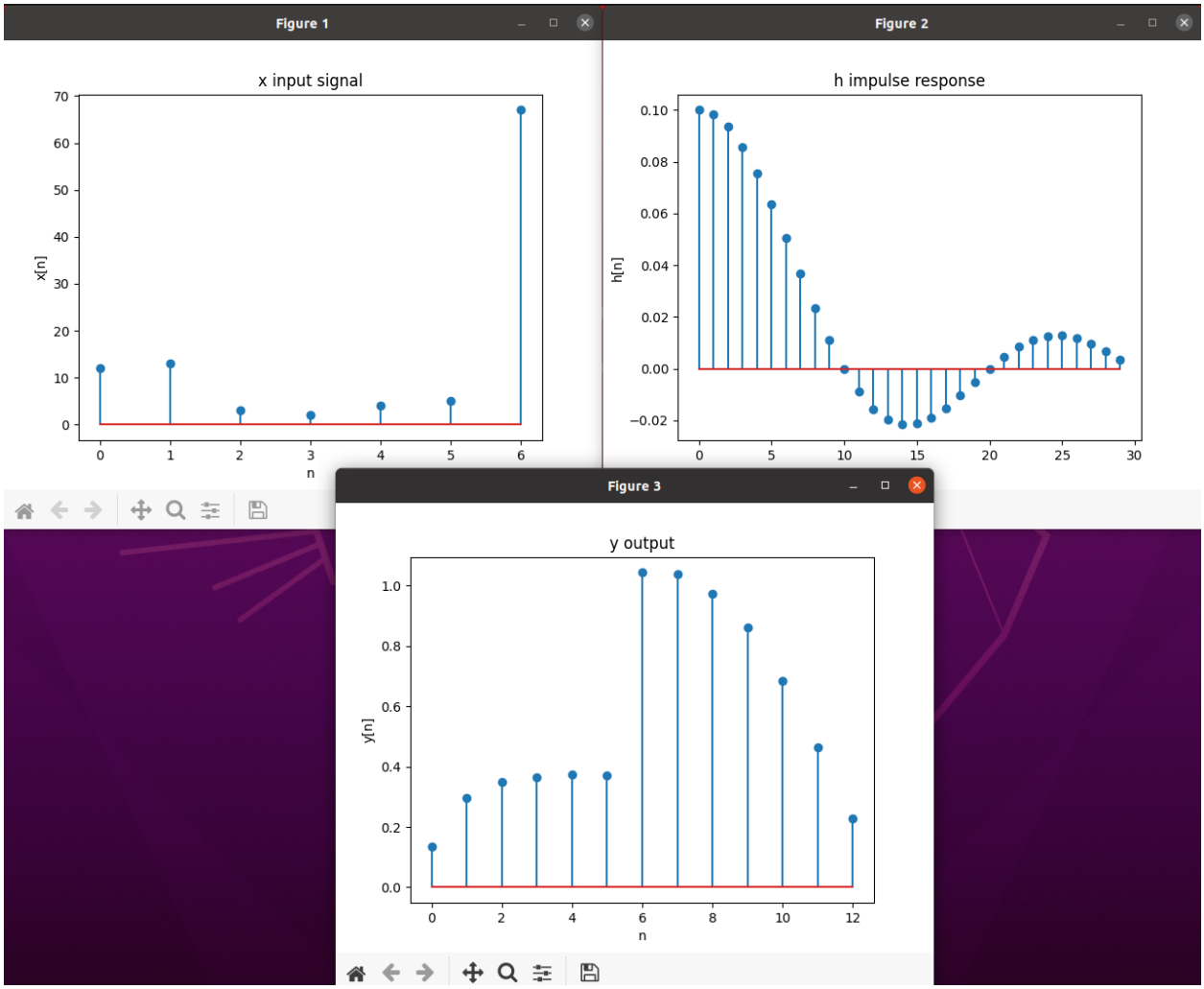
N2 = 10



Test 5:

Low-pass filter

N = 20

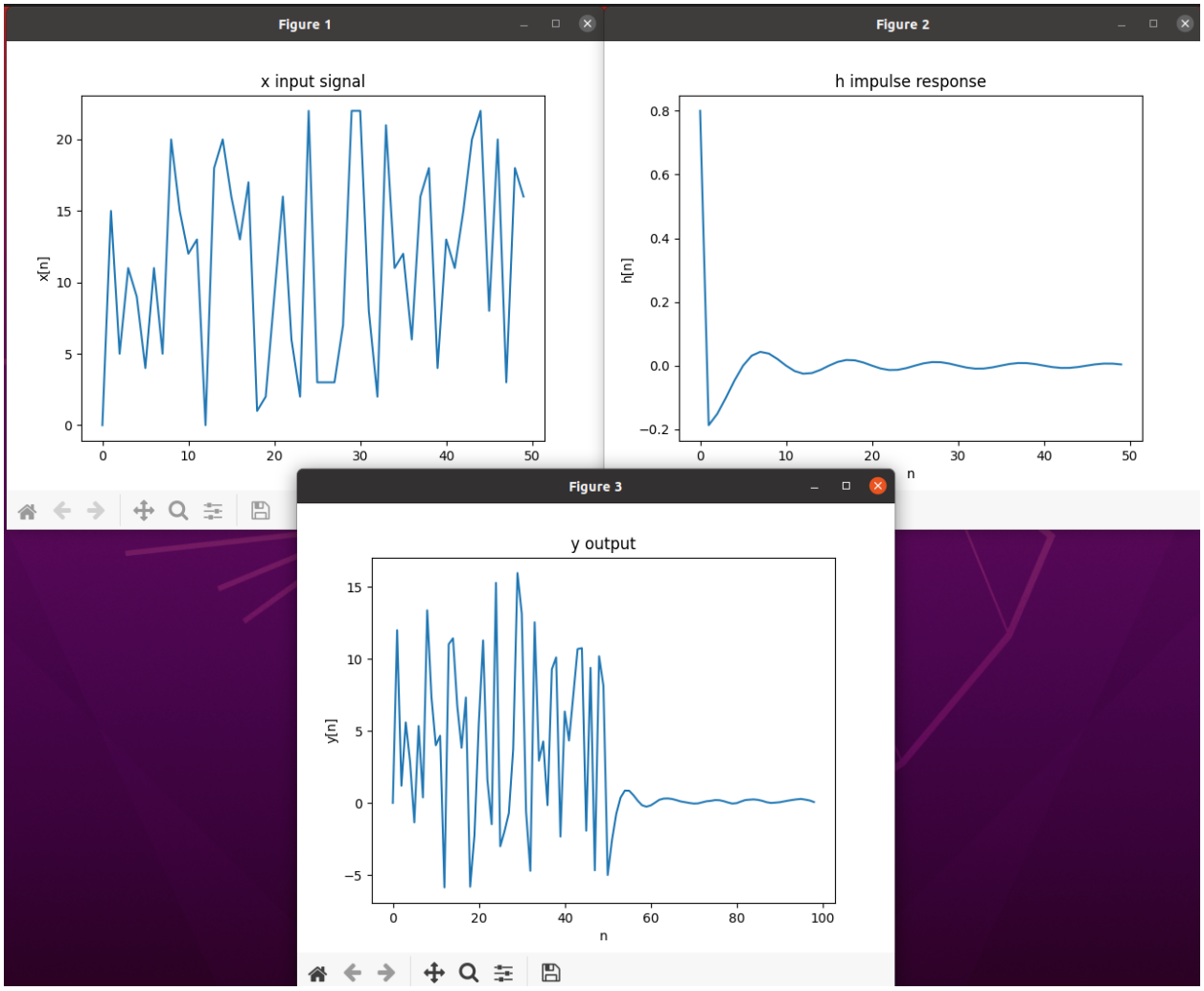


Test 6:

High-pass filter

The parameter N = 10

Number of points = 50

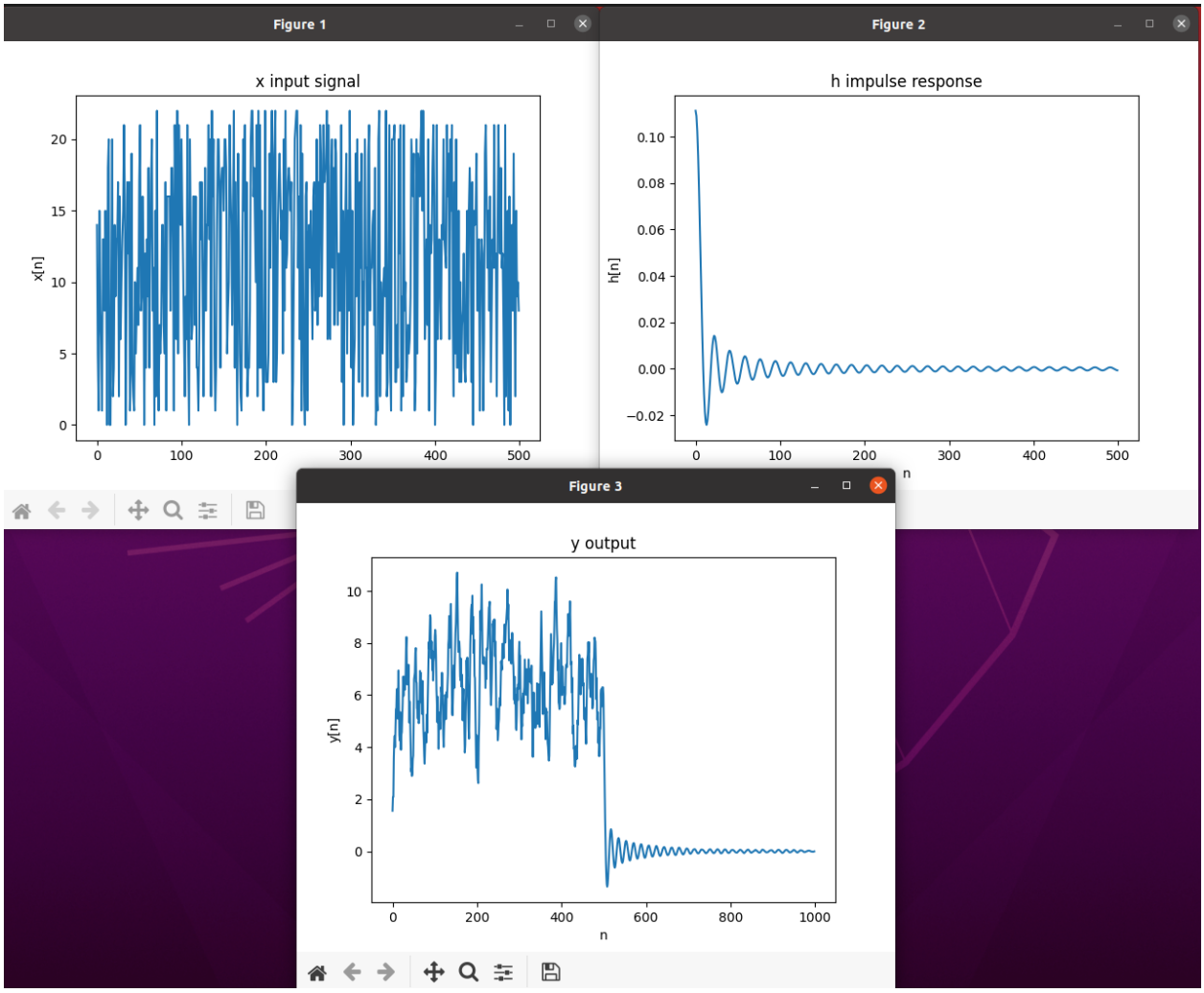


Test 7:

Low-filter pass

The parameter N = 18

Number of points = 500

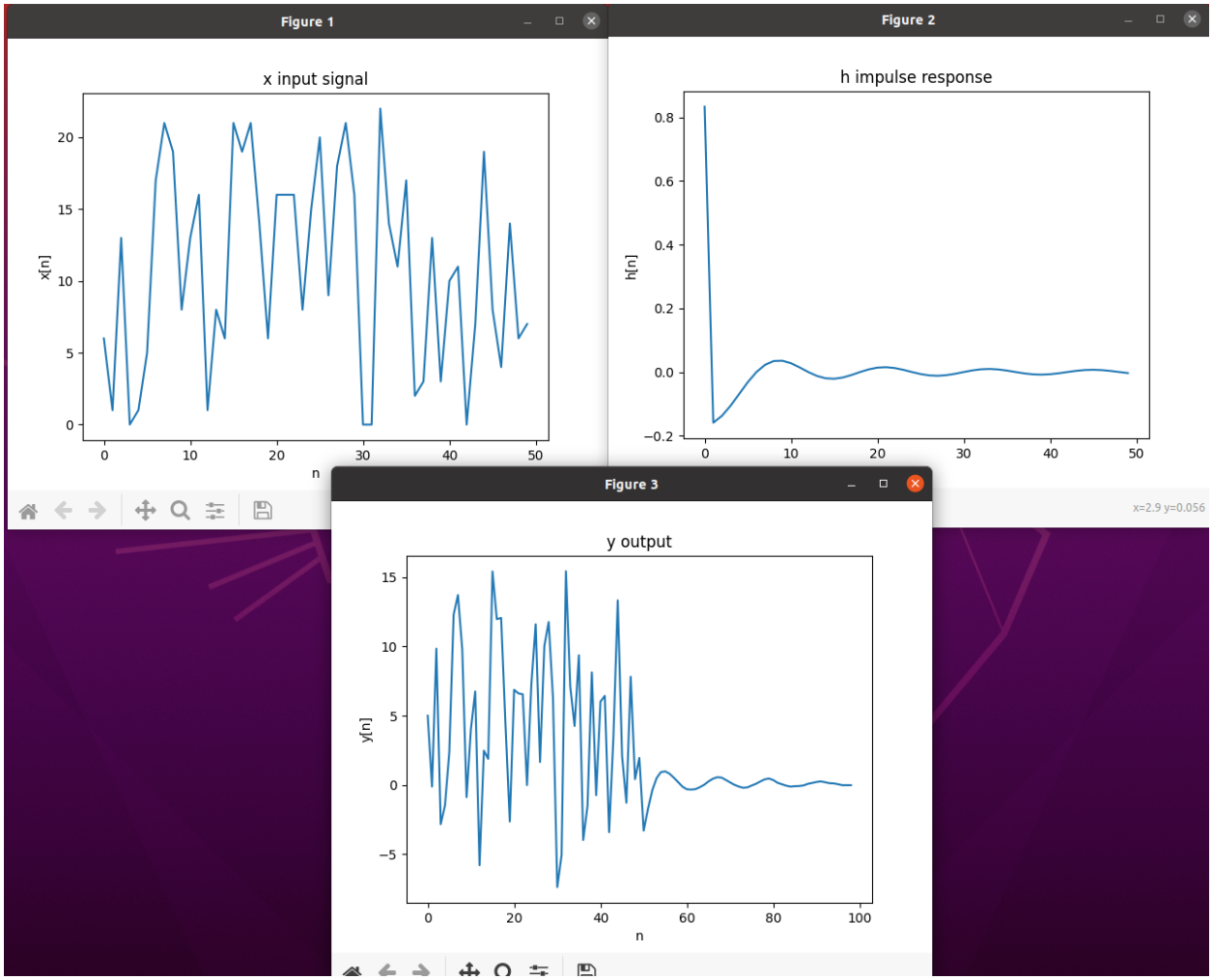


Test 8:

High-pass filter

The parameter N = 12

Number of points = 50



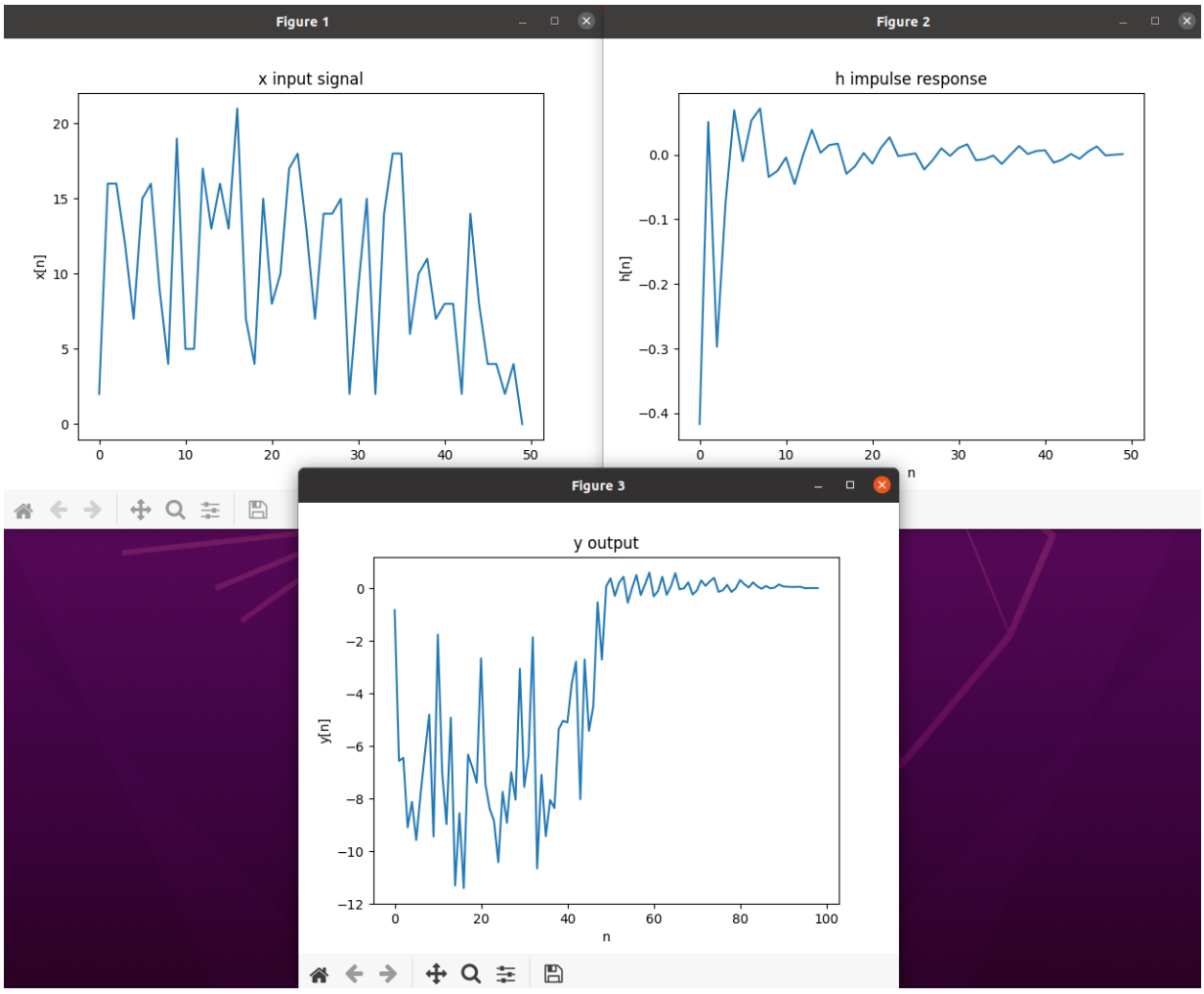
Test 9:

Band-pass filter

N1 = 3

N2 = 8

Number of points = 50



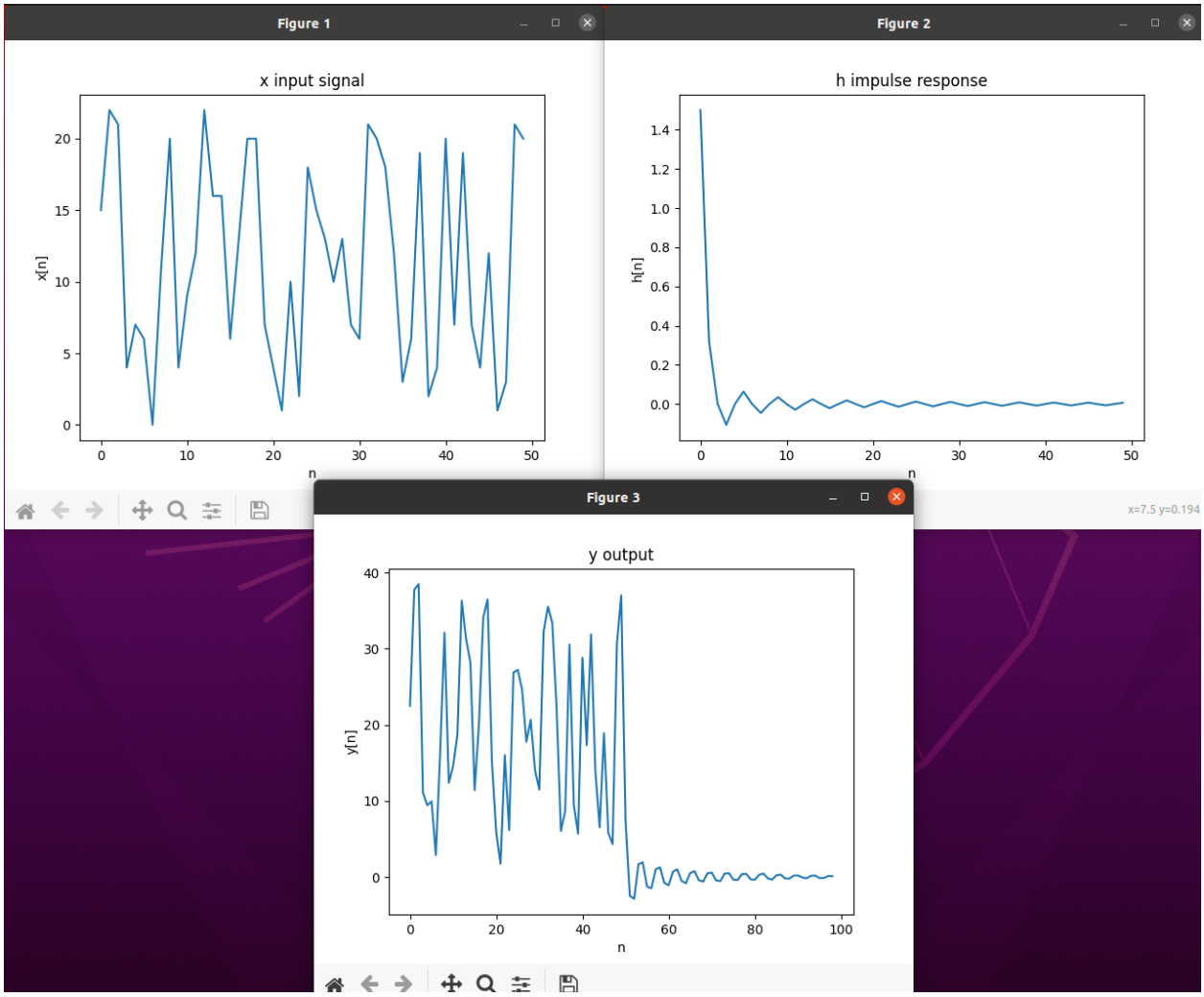
Test 10:

High-pass filter

N1 = 2

N2 = 4

Number of points = 50



IV - Appendix

V - References

Inverse z transform calculator. ascsepositive. (n.d.). Retrieved May 18, 2022, from https://ascsepositive.weebly.com/blog/inverse-z-transform-calculator

Lowpass filter. Lowpass Filter - an overview | ScienceDirect Topics. (n.d.). Retrieved May 18, 2022, from https://www.sciencedirect.com/topics/engineering/lowpass-filter