Digital Signal Processing Laboratory
Department of Electrical & Computer Engineering

Digital Signal Processing Laboratory

Laboratory 2

Discrete Time Systems

Heraklion 2025

Dr. Konstantinos Karampidis

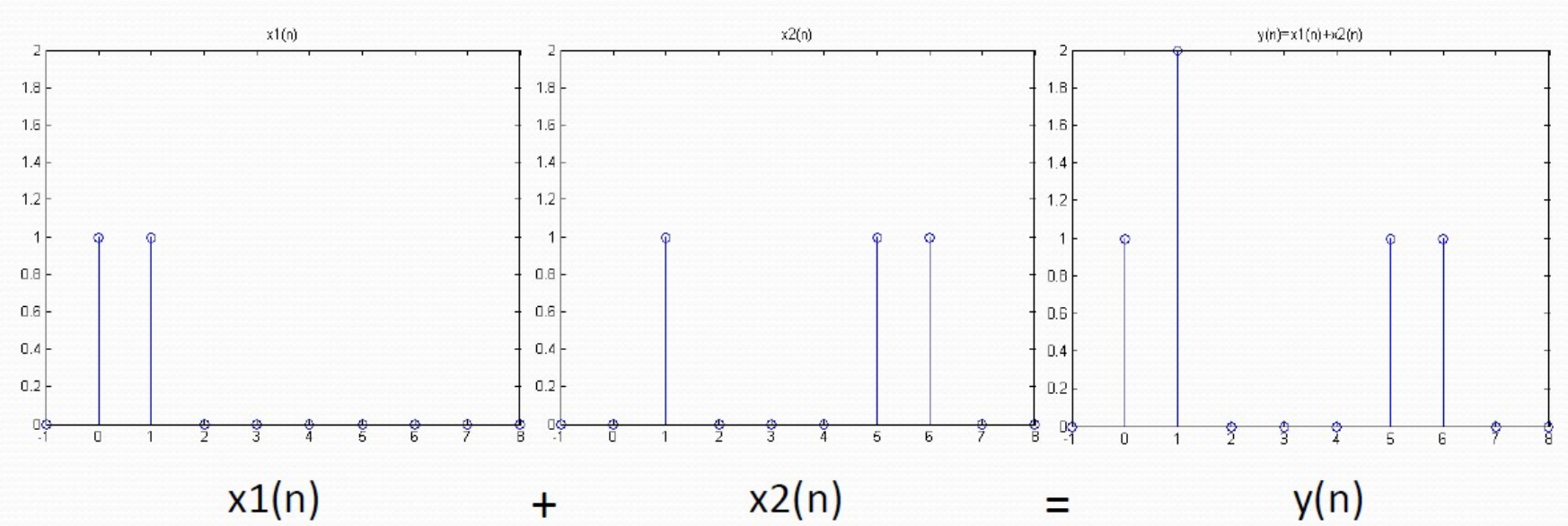
Discrete signal operations

Addition	x(n) + y(n)
Subtraction	x(n) – y(n)
Multiplication	x(n) * y(n)
Division	x(n) / y(n) with y(n)≠0

The operations are performed **per element** and for the same value of the independent variable.

Example

Suppose we want to add the following two signals x1(n) and x2(n):



Signal Analysis

Using the discrete delta sequence δ, we can analyze a random signal. This is done by summing the appropriately shifted δ(n) which have been multiplied by a weight factor.

The weight factor corresponds to the signal's value x(n), as shown in the example below:

$$x(n) = \cdots + x(-1)\delta(n+1) + x(0)\delta(n) + x(1)\delta(n-1) + x(2)\delta(n-2) + \cdots$$

Signal Analysis

$$x(n) = \cdots + x(-1)\delta(n+1) + x(0)\delta(n) + x(1)\delta(n-1) + x(2)\delta(n-2) + \cdots$$

This sum can be written as:

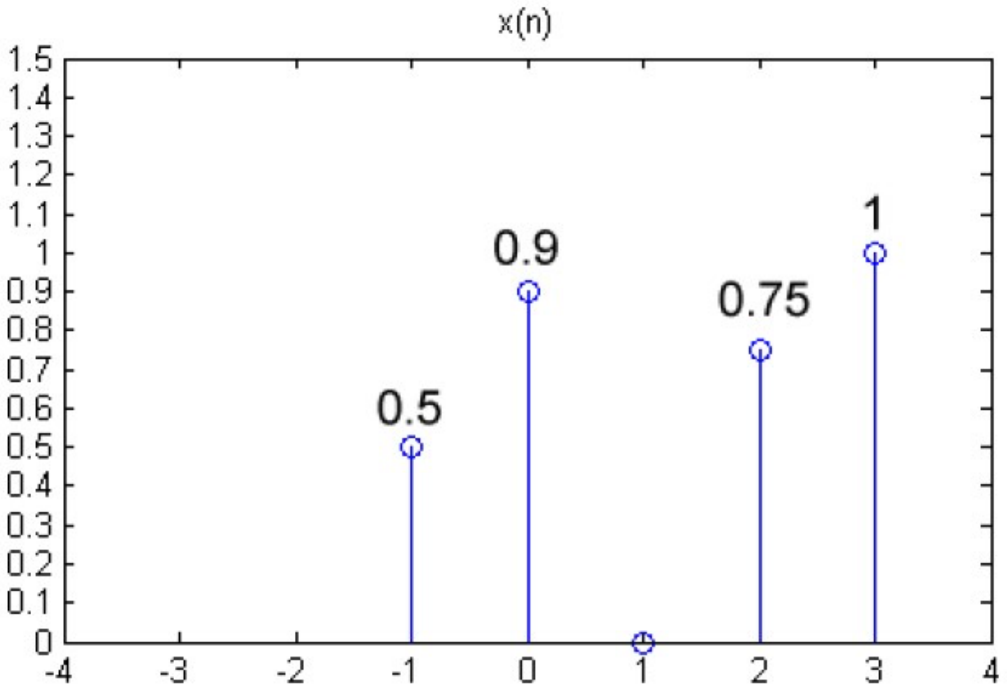
$$x(n) = \sum_{k=-\infty}^{\infty} x(k)\delta(n-k)$$

where each term x(k) δ(n-k) is a signal with amplitude x(k) at time n=k, and it is zero for any other value.

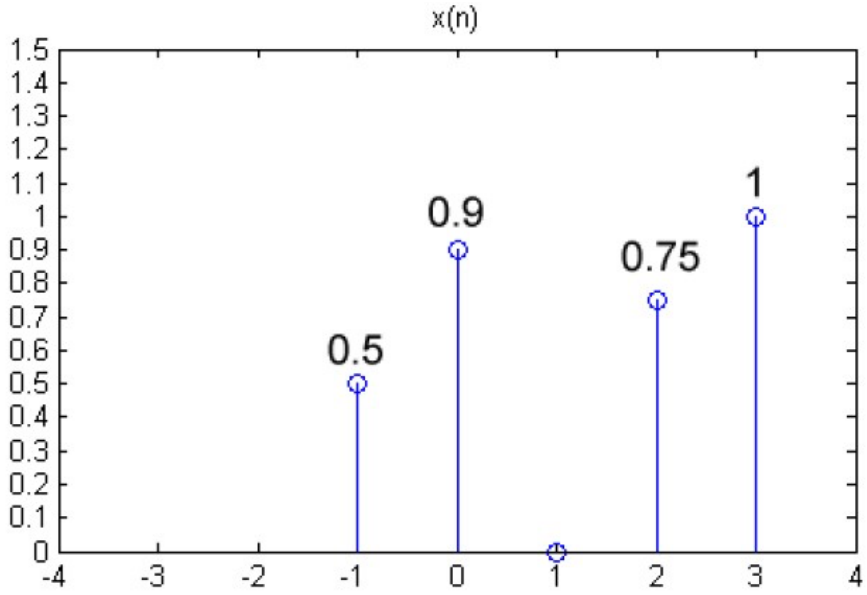
Signal Analysis

Using the signal analysis expression we have seen, write the following signal on paper in the form of sums of discrete delta functions:

$$x(n) = \sum_{k=-\infty}^{\infty} x(k)\delta(n-k)$$



Signal Analysis

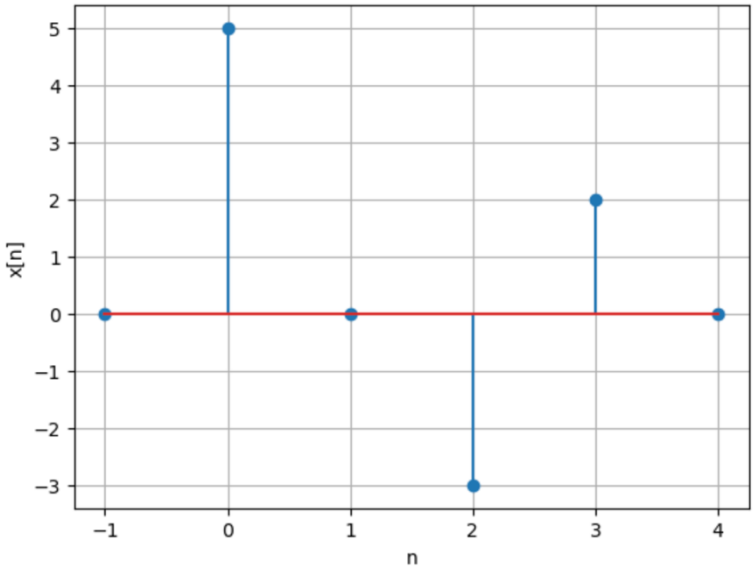


The above signal can be written in the form of sums of discrete delta functions as:

$$x(n) = 0.5*\delta(n+1) + 0.9*\delta(n) + 0.75*\delta(n-2) + \delta(n-3)$$

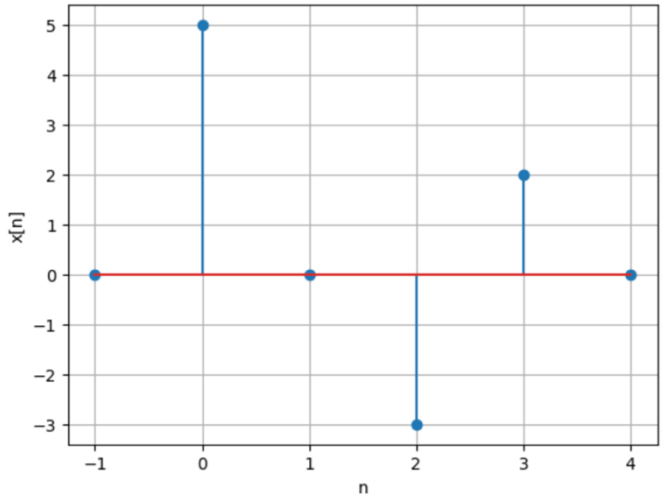
Signal Analysis

Using the signal analysis expression, write the following signal on paper in the form of sums of discrete delta functions:



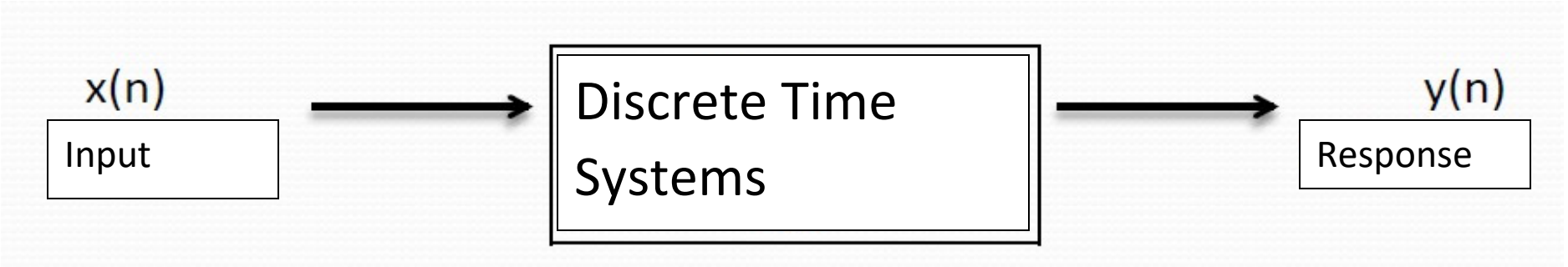
Signal Analysis

The signal will be $x(n) = 5\delta(n) - 3\delta(n-2) + 2\delta(n-3)$



Discrete Time Systems

A discrete-time system accepts a discrete-time input x(n) and produces a discrete-time output y(n) by transforming it.
The input signal can be called an excitation while the output signal can be called a response.



This process is symbolized as follows:

$$y(n) = T[x(n)]$$

Properties of Discrete signals

Causal Systems

A causal system is a system in which the output for any time n_0 depends on the input at time n_0 and past times (not future times). For example, the following system is a causal system:

$$y(n) = \alpha x(n) - \beta x(n-1)$$

Non-causal is a system in which the output at time n depends on future times at the input.

Causal Systems - Example

Let y[n]=0.5y[n-1]+x[n], where:

- y[n] is the output of the system at time n,
- y[n-1] is the output of the system at the previous time step [n-1], and
- x[n] is the input to the system at time step n.

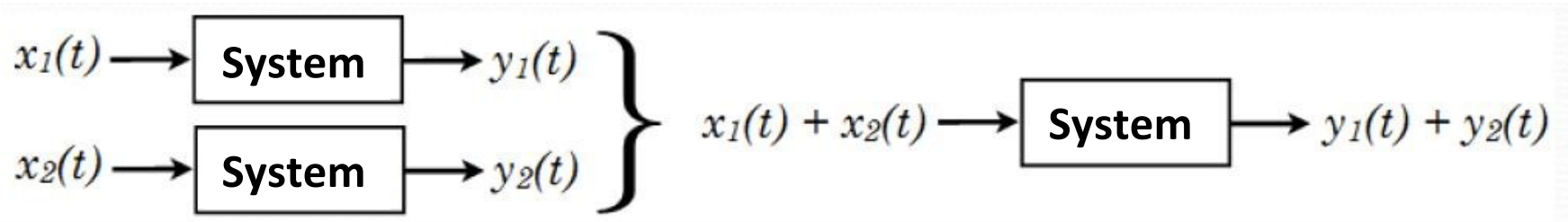
The system is causal because:

- At any time n, the output y[n] depends on the current input x[n] and the previous output y[n-1]
- It does not depend on future values of the input

Properties of Discrete signals

Principle of Superposition

The principle of Superposition: is defined if the sum of the outputs of multiple systems is equal to the output of the system where the input is the sum of the inputs of those systems.



The principle of superposition is defined in a system when:

$$T[x_1(n) + x_2(n)] = T[x_1(n)] + T[x_2(n)]$$

Properties of Discrete signals

A system is called Homogeneous if multiplying the input by a constant, results in multiplying the output by exactly the same constant.

A system is called Homogeneous when:

$$T[\alpha x(n)] = \alpha T[x(n)]$$

where α is a constant.

Properties of Discrete signals

Linear Systems

A system is linear if it is homogeneous and for which the principle of superposition holds.
Therefore, for a system to be linear, the following relationship should apply:

$$T[\alpha_1 x_1(n) + \alpha_2 x_2(n)] = \alpha_1 T[x_1(n)] + \alpha_2 T[x_2(n)]$$

In the theory notes you can find several examples of linear and non-linear systems.

Properties of Discrete signals

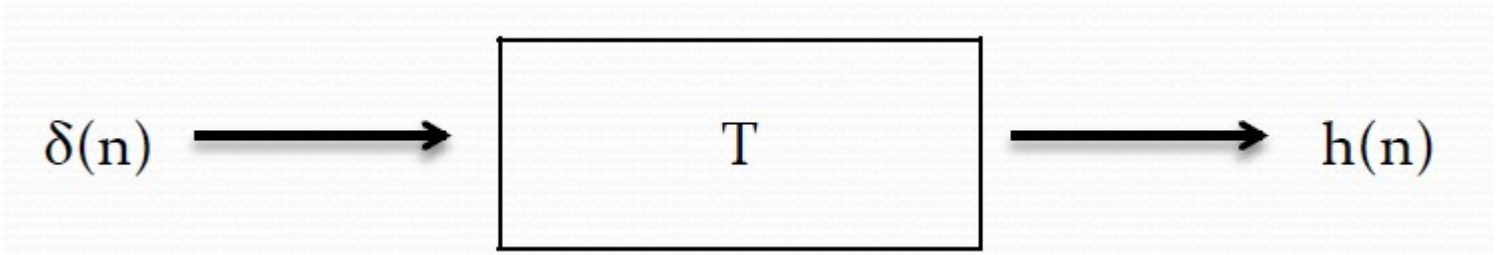
Time-invariant systems

If a system has a shift (delay) in the input by n_0 and produces an output with the same shift n_0, then the system is called **Time-Invariant or Shift-Invariant.**

If a system is both Linear and Time invariant, we will use the abbreviation **LSI** (Linear Shift Invariant).

Impulse response

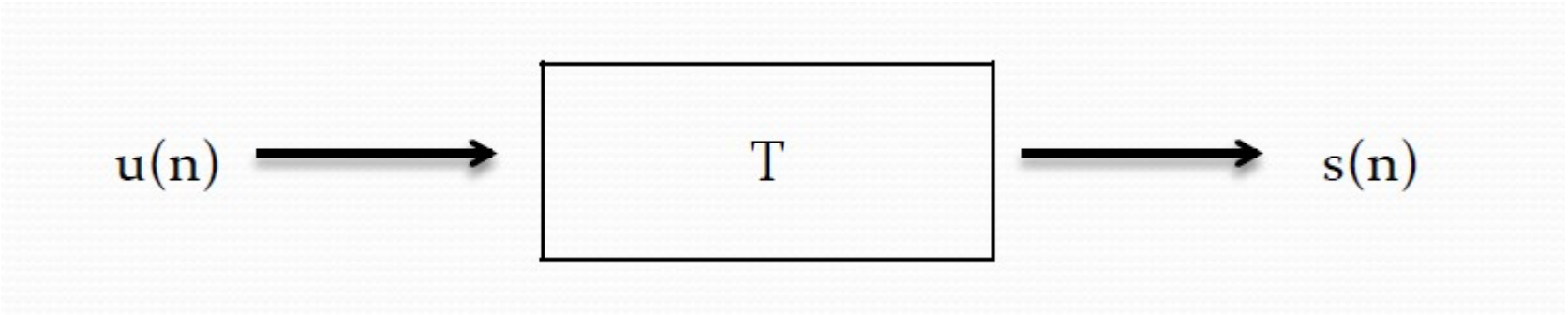
If in a linear and time invariant (LSI) system, the input is the impulse sequence δ(n), then the output signal (response) is called the **impulse response** h(n)**.**



The impulse response h(n) will be causal **if and only if** it is equal to zero for every n<0.

Step Response

If in a linear and time invariant (LSI) system the input is the step sequence u(n), then the output signal (response) is called the **step response s(n).**



Difference Equation

The general form of a linear difference equation with constant coefficients is:

$$y(n) = \sum_{k=0}^{q} b(k)x(n-k) - \sum_{k=1}^{p} a(k)y(n-k)$$

where a(k) and b(k) are constants which define the system.

The difference equation provides a method of calculating the response of a system for a random input. To solve such equations, it is often necessary to compute a set of Initial Conditions.

For a LSI system described by a difference equation, its impulse response h(n) is calculated by solving the corresponding difference equation when x(n)=d(n) and y(n)=h(n).

Difference Equation

The lfilter function in scipy helps us to solve difference equations to find the response of the system:

Syntax

scipy.signal.lfilter(b, a, x)

where: b=[b1,…, bm] and α=[α0, α1,…, αm], are the coefficients of the difference equation, while vector x is the matrix with the values of the input signal of the system.

The “lambda” function

With the "lambda" function we can define a function dynamically, with the following syntax: function_name =lambda(expression)

For example, to calculate the square of a number:

```
my_square = lambda x: x*x
```

Then the result of my_square(5) will be 25.

This function can be used temporarily.

We should mention that instead of x we could have written any valid variable name.

Exercise

Write on “paper” in the form of sums of discrete delta functions δ, and then graph the signal:

$$x(n) = \left\{ -2, 1, -\frac{1}{3}, \underset{\uparrow}{\frac{1}{2}}, 2, 1, 0, 1, 0, 0, 3 \right\}$$

The arrow indicates the value for time n=0.

Solution

$$x(n) = \left\{ -2, 1, -\frac{1}{3}, \underset{\uparrow}{\frac{1}{2}}, 2, 1, 0, 1, 0, 0, 3 \right\}$$

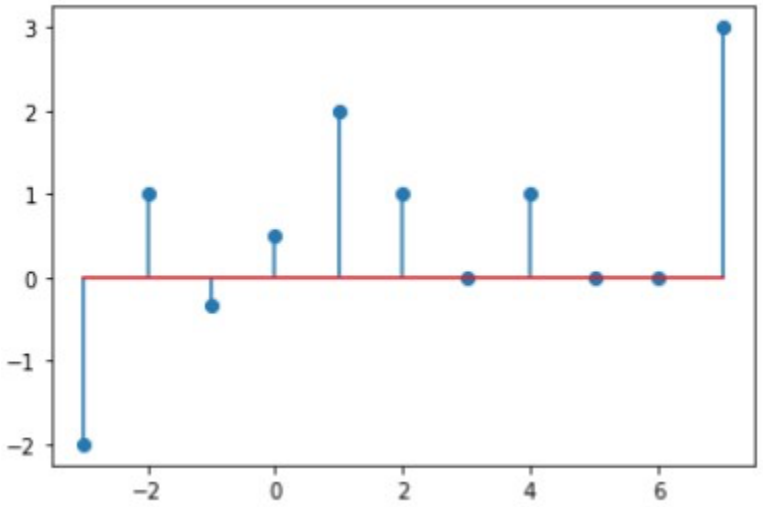
$$x(n) = -2\delta(n+3) + \delta(n+2) - 1/3\delta(n+1) + 1/2\delta(n)$$
$$+2\delta(n-1) + \delta(n-2) + \delta(n-4) + 3\delta(n-7)$$

Graphical representation

```
import numpy as np
import matplotlib.pyplot as plt
delta = lambda n: n==0
n=range(-3,8,1)
x=np.zeros(len(n));
h=np.zeros(len(n));
 h[1]=1;
out=[]
for i in range(0,len(n),1):
   x[i]=-2*delta(n[i]+3) + delta(n[i]+2) -
            1/3*delta(n[i]+1)+ 1/2*delta(n[i])
         +2*delta(n[i]-1) + delta(n[i]-2)+ delta(n[i]-4) + 3*delta(n[i]-7);

out.append(x[i])

plt.stem(n,out)
```



Impulse response calculation

Calculate and graph the impulse response of the following causal signal:

$$y(n) = -0.9y(n-1) + x(n)$$

1st way to solve:

We solve the difference equation when $x(n)=\delta(n)$ and $y(n)=h(n)$

Therefore $h(n) = -0.9h(n-1) + \delta(n)$

We calculate the initial conditions:

$h(0) = -0.9 * 0 + 1 = 1$

$h(1) = -0.9 * h(0) + \delta(1) = -0.9 * 1 + 0 = -0.9$

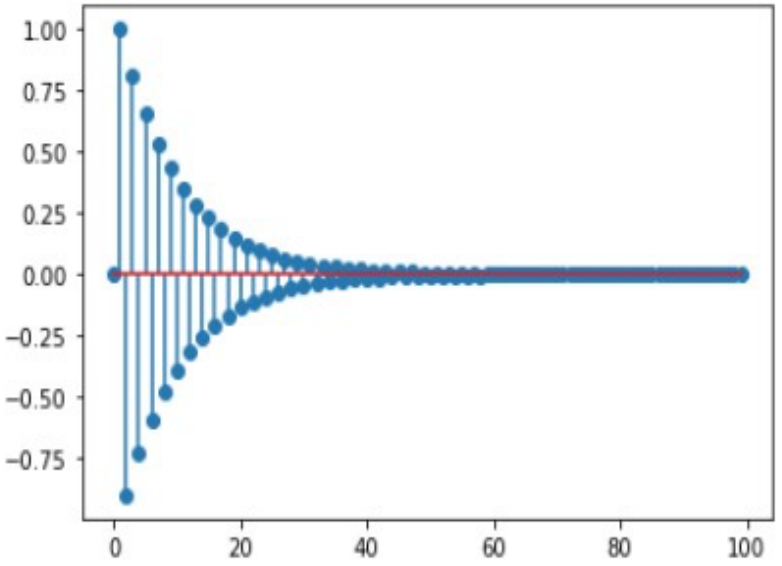
$h(2) = -0.9 * h(1) + \delta(2) = -0,9 * -0,9 + 0 = 0,81$

$h(3) = -0.9 * h(2) + \delta(3) = -0,9 * 0,81 + 0 = -0,729$

Impulse response calculation

1st way to solve

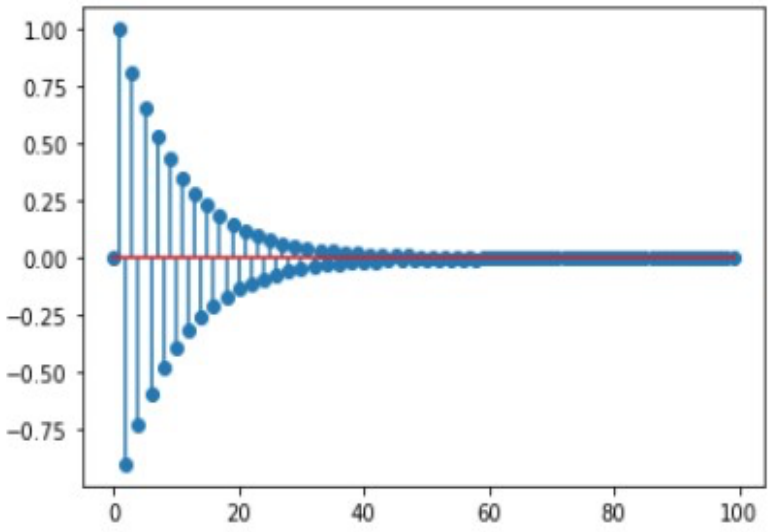
```
import numpy as np
import matplotlib.pyplot as plt
n=range(0,100,1)
h=np.zeros(len(n));
delta = lambda n: n==0
h[1]=1
for i in range(2,len(n),1):
  h[i]=-0.9*h[i-1]+int(delta(n[i]))
plt.stem(n,h)
```



Impulse response calculation

1st way to solve – different approach

```
import numpy as np
import matplotlib.pyplot as plt
h[1]=1
for n in range(2,100):
    h[n]=-0.9*h[n]+int(delta(n))
plt.stem(range(0,100,1),h)
```



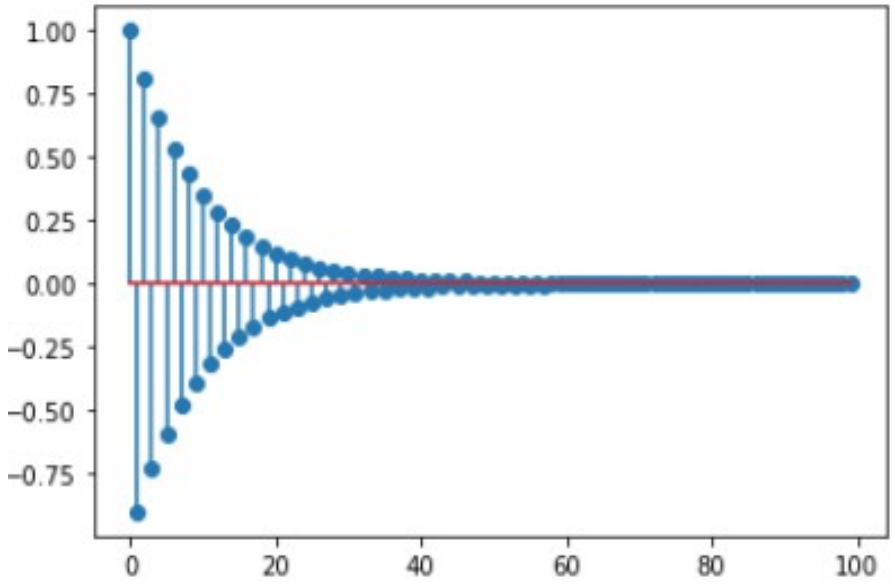
Impulse response calculation

The 2nd way to solve this is to use the lfilter function.

- Having in mind the general form of a linear difference equation with fixed coefficients, we separate y from x.
- We then find the coefficients a and b (coefficients of the difference equation), where a corresponds to the weight coefficients of y and b corresponds to the weight coefficients of x.

Impulse response calculation

```
## y(n) = -0.9y(n-1) + x(n) => y(n) + 0.9y(n-1) = x(n)
import scipy
from scipy import signal
a=[1 , 0.9]
b=[1, 0]
x=np.zeros(100)
for n in range(0,100,1):
   temp=int(delta(n))
   x[n]=temp
n=range(0,100,1)
y= scipy.signal.lfilter(b,a,x)
plt.stem(n,y)
```



Step response calculation

Calculate and graph the step response of the following causal signal: y(n)= y(n-1)+ x(n)-x(n-8)

1st way to solve

We solve the difference equation for x(n)=u(n) and y(n)=s(n)

s(n) = s(n-1) + u(n)-u(n-8)

Calculation of initial conditions:

$$u(n) = \begin{cases} 1 & n \geq 0 \\ 0 & n < 0 \end{cases}$$

s(0) = s(-1) + u(0) - u(-8) = 0 + 1 - 0 = 1

s(1) = s(0) + u(1) - u(-7) = 1 + 1 - 0 = 2

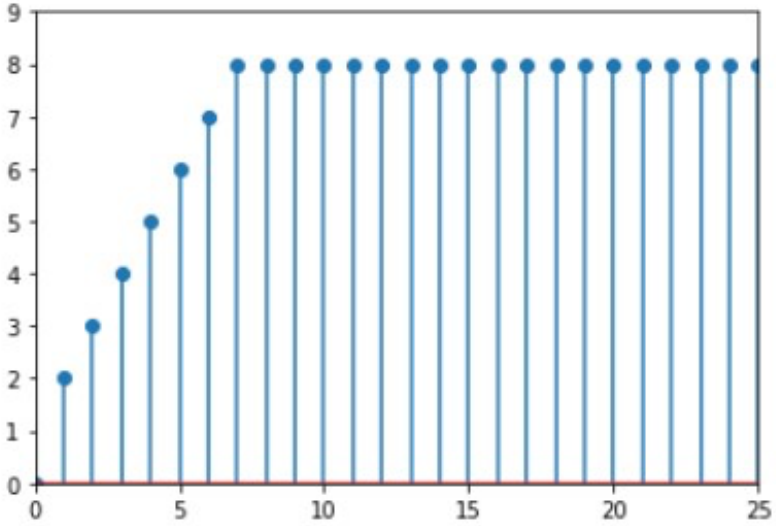
s(2) = s(1) + u(2) - u(-6) = 2 + 1 - 0 = 3

….

s(7) = s(6) + u(7) -u(-1) = 7 + 1 -0 = 8

s(8) = s(7) + u(8) -u(0) = 8 + 1 -1 = 8 s(9) = s(8) + u(9) -u(1) = 8 + 1 -1 = 8

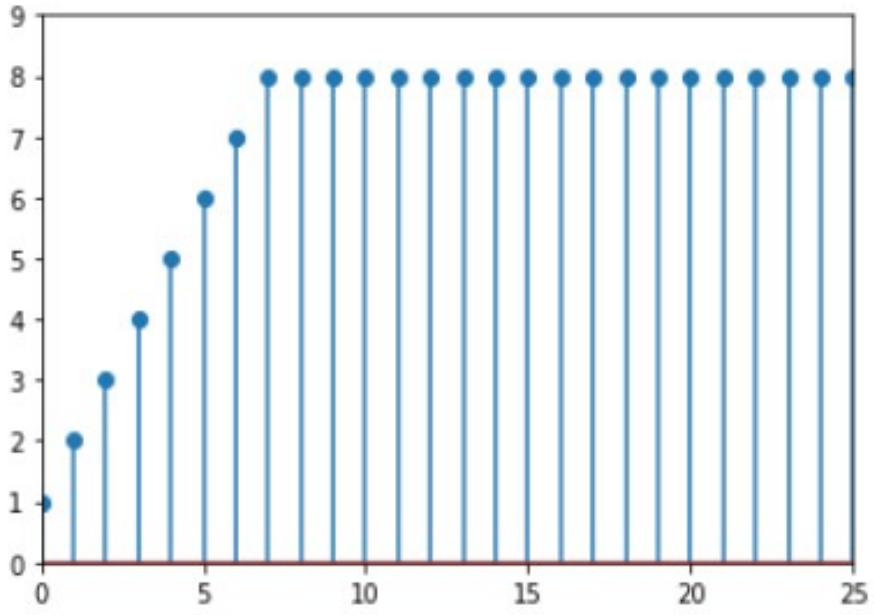
```
n=range(0,100,1)
s=np.zeros(len(n));
 u = lambda n: n>=0
 s[1]=2
for i in range(2,len(n),1):
    s[i]=s[i-1]+int(u(n[i]))-int(u(n[i]-8))
plt.stem(s) plt.xlim([0, 25])
plt.ylim([0, 9])
```



2nd way to solve using the lfilter function

y(n) = y(n-1) + x(n) – x(n-8) => y(n) – y(n-1) = x(n) – x(n-8)

```
n=range(0,100,1)
unit = lambda n: n>=0
a=[1,-1,0,0,0,0,0,0,0]
 b=[1,0,0,0,0,0,0,0,-1]
u=np.zeros(100)
 for n in range(0,100,1):
    temp=int(unit(n))
    u[n]=temp
y=scipy.signal.lfilter(b,a,u)
plt.stem(y)
plt.xlim([0, 25])
plt.ylim([0, 9])
```

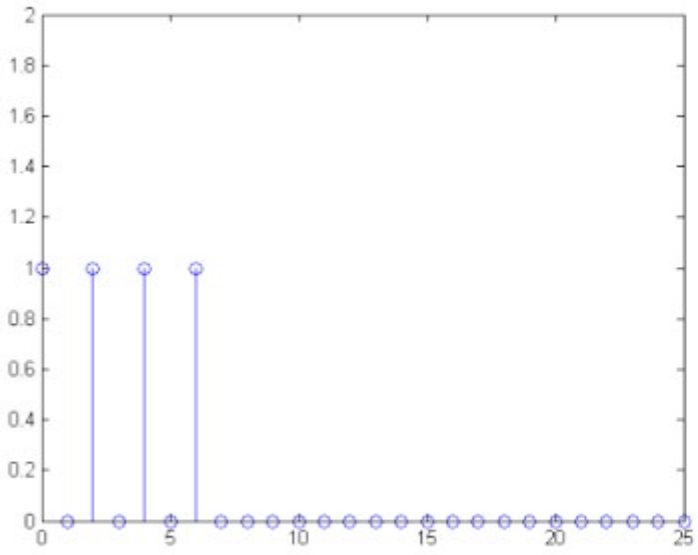


Exercise

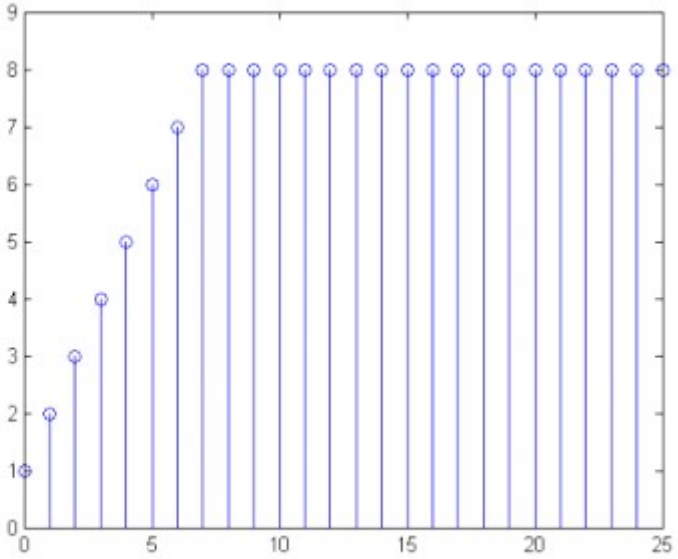
Plot the response of the previous system with input **$x(n) = a^n u(n)$**, when a=1 and a=-1.

$y(n)= y(n-1) + x(n) - x(n-8) \Rightarrow y(n) = y(n-1) + a^n u(n) - a^{n-8} u(n-8)$

```
a=1
s[1]=0
n=range(0,100,1)
unit = lambda n: n>=0
for i in range(2,100,1):
    s[i]=s[i-1]+a**n[i]*unit((n[i]))-a**(n[i]-8)*unit((n[i]-8))
plt.stem(s)
plt.xlim([0, 25])
plt.ylim([0, 9])
```



a=-1



a=1