División de Ingenierías
Dept. de Ingenierías Eléctrica y Electrónica
Señales y sistemas IEN 4100. Docente: Carlos cárdenas

Lab 2: Convolution of signals on the discrete and continuos domain

Group : Samuel Barrios, Michelle Suárez
{barriossd, michellejs}@uninorte.edu.co
16 de octubre de 2022

Abstract
This report presents the final results of the simulation made for differents forms of signals just like senusoidal, pulse, triangular, quadratic, and different types of ramps. The project needed to show how the results of the convolution between any of them would look like. The fundamental elements for the development of the practice were the programming language python and the graphic interface streamlit following the parameters stablished on the laboratory guide.
Keywords: Signals, convolution, interface, python.

1. Introduction

This laboratory practice was done in order to have a better understanding of the generation of signals and the process of convolution between them both on the discrete and continous domain. This project will be plasmaded on the software Streamlit. The development of this project was divided in two parts, the first one consisted in the generation of different signals like senousioal, triangular, pulse, exponential and differents types of ramps, those signals must follow the parameters that the user requires, for example, the user must be able to choose wich signal is going to be the first or second signal to convolve. This first part requires an understanding of the Python language, programmig, and how to generate other signals from it.

On the second part of this laboratory, the code design of the convolution of each pair of signal wad mande, this part of the laboratory requires the bases of the theoretical part of the topic 'convolutions' to achieve the principal objective of this laboratory[[1]]. This part of the laboratory was developed following two ways of convolution one was using the command numpy.convolve on python and the other one was implemented using a code with the command .simps.

The two parts mentioned before, were an excelent method of pratice to develop the habilities of programming and designing. This laboratory was made with the purpose of studying the generation and convolution of any kind of signal.

2. Procedure

This section will present the development of this laboratory by describing each activity that was required on it.

2.1. App Design

The software used for the design of the interface was Streamlit. Some of the elements used were Axeses, Dropdowns, edit labels and buttons, the function of each element will be described on the next list.

1. Axes: In total, four axes were used to achieve the purpose of this laboratory, two of them were used to portray the generation of the signas requested for the user, and the others to show the final result of convolution and their process, these last two

axes were implemented with the subplot command.

2. Dropdown: In total, Three dropdowns were used, Those dropdown were used to show the different options that the user has in each part of the lab. this dropdown represents the diffents signal that the app provides to the user, and the domain in which the signals are presented.

3. Button: In total, there was one button on the app with the purpose of run the options chosen by the user in each part of the laboratory.

4. Edit field: The were a few of this component in the journey of the project but basically, they were used to provide the user and space to describe the conditions that the signal has to accopmlish.

The final result of the interface will be represented on the next figure:

Figura 1: Interface design

2.2. Code Design for generation of signals

On this section will be described the develpoment of the code made to achieve the generation and convolution of each signal. Before describing the code used on this practice. It is important to call out the software used for this code develpoment were the Anaconda Navigator to create the environment for the practice and Studio Visual Code in which Python was included.

On the software Studio Visual Code and Anaconda different libraries were installed that helped the action of coding a lot easier and determinated the pronouns that this apps will have to be called on the rest of the code. On the next figure will be represented the libraries used.

```
from typing import final
import streamlit as st
import math
from re import X
from turtle import end_fill
import numpy as np
import matplotlib.pyplot as plt
from scipy import signal
from scipy import integrate
from scipy.integrate import simps
#import sk_dsp_comm.sigsys as ss
import time
```

Figura 2: Library used

On the first part of the signals generation it was created the different dropdowns that will contain the options that the user have to choose which type of signal wants to generate and another where can choose in which domain will that signal be, those options were made using the command st.sidebar.select box:

```
option = st.sidebar.selectbox(
    'Select your first function, X(t)',
    ('-', 'Exponential', 'Sinusoidal', 'Triangular', 'Rectangular',

st.write('You selected', option, 'as your x(t)')
```

Figura 3: Sidebar code

Then, after the user selects the option through an edit label will described how that signal wanted to be. Then different conditionals were created to achieve what was requested, one of them was selecting between which domain will be develop the signal and the other describing which signal will be generated. An example of it will be represented on the next figure:

```
if option == 'Sinusoidal':
    st.sidebar.latex("x(t) = A*sin(2*pi*f*t)")
    a = st.sidebar.number_input('Please type the value for amplittude',
    f = st.sidebar.number_input('Please type the value for the frequenc
    xinicio = st.sidebar.number_input('Type the point of start for the

    if f == 0:
        t = 0
        xfinal = 0
    else:
        xfinal=(1/f)+xinicio
        t=np.arange(xinicio,xfinal+s,s)
        if dom== 1:
```

Figura 4: Code first part

To construct the signal in a discret form the dom option mark one at the line of the code and then it start running the code for discrete signals, the solution and the code plasmed was making a loop using while in which will determinate the magnitude of it depending of the frequency or the form of the signal and their start and ending point.And example of it will be represented on the next figure:

```
if dom==1 :
    s=1/(10*f)
    t = np.arange (xinicio, 3/f+xinicio,s)
    vx=np.arange((xfinal-xinicio)/s)
    i=0
    pos=0
    while i<abs(xfinal) and pos<abs((xfinal-xinicio)/s):
        vx[pos]=a*signal.sawtooth(2*np.pi*f*i,0.5)
        i=i+s
        pos=pos+1
    with fila_graficas[0]:
        plt.xlabel("t(s)")
        plt.ylabel("x(t)")
        graph1.stem(t,vx)
```

Figura 5: Code for discrete signals

On the other hand, The continous domain in each signal, it was implement the equation that describes it to construct the signal in this domain as an example of it

```
        graph1.stem(t,vx)
else:
    t = np.arange (xinicio, xfinal+s,s)
    y = a*signal.sawtooth(2*np.pi*f*t,0.5)
    graph1.plot(t,y)
    with fila_graficas[0]:
        plt.xlabel("t(s)")
        plt.ylabel("x(t)")
```

Figura 6: Code for continous signals

A different script was design when it was time to construct ramp signals, on this practice it was required to construct three types of ramps. The solution or the way to make it was trying to do a function by parts describing the range of each part and the function that it will presented using the command "Lambda", dependending on the parts and what type of ramp the user wants to generate. An example of the code of it will be representes on the next figure:

```
=np.arange(xinicio,xfinal,0.01)
=np.piecewise(x1,[(xinicio<=x1) & (x1<b), (b<=x1) & (x1<=c), (c<x1) & (x1<=xfinal)],[lambda x:tramo1(x1), lamb
amo1=np.vectorize(tramo1)
amo3=np.vectorize(tramo3)
t.xlabel("t(s)")
t.ylabel("x(t)")
aph1.plot(x1,y1)
```

Figura 7: Code for ramp Signals

It is important to mention that each signal was graphicaded with the command "plot.ªnd made it in some vectors that contain the numbers between the start and the end of the signal.

2.3. Code for convolution of signals

On this part of programming, it was required to take two of any of the signals generated by the program and represent the convolution of them in any axes, a bonus was to demostrate the process of convolution on a dinamic way. it were made two solutions for the situation situation requested as a first solution was implemented a couple of code line using the command simps to simulated the convolution of two signals as it was represented in the theoric classes as an integration of a function a and b.As it will represented on the next figure:

```
x=np.linspace(0,xfinal2,100)
conb=[]
for xx in x:
    xp=np.linspace(0,xx,100)
    h=g(xp,y)*g(xx-xp,y2)
    I=simps(h,xp)
    conb.append(I)
plt.plot(x,conb,label="Convolución")
xr=np.linspace(0,xfinal2,1000)
plt.plot(xr,g(xr,y),label="Función 1")
plt.plot(xr,g(xr,y2),label="Función 2")
```

Figura 8: Convolution code using simps

The other solution for the sitution was using the command convolve which make this convolution a lot of more easier, on this solution also it was demostrated in a dinamic way the convolution of two signals by making frames of them in any of the domain.This code line will be represented on the next figure:

```
y_conv = np.convolve(x1,h1, mode='full')*r

y_conv=np.resize(y_conv, np.shape(ty))

hs = h1[::-1]
tm = np.arange(xi1-(xf2-xi2),xi1+r,r)
aniy = np.zeros(len(y_conv))
frames = 30
factor = len(ty)/frames
factor = math.floor(factor)
factor_t = (max(ty)-min(ty))/frames

x1 =np.resize(x1, np.shape(t1))
hs =np.resize(hs,np.shape(tm))
for i in range(frames+2):
    aniy[:i*factor] = y_conv[:i*factor]
    grafica3.clear()
    grafica3.plot(tm,hs,t1,x1)
```

Figura 9: Convolution code using Convolve

3. Results and analysis

The proper testings were done in order to ensure the functionality of the code. the code on python started tu run it was verify the different signals in a discrete or continous domain.As it is represented in the nexts figures.

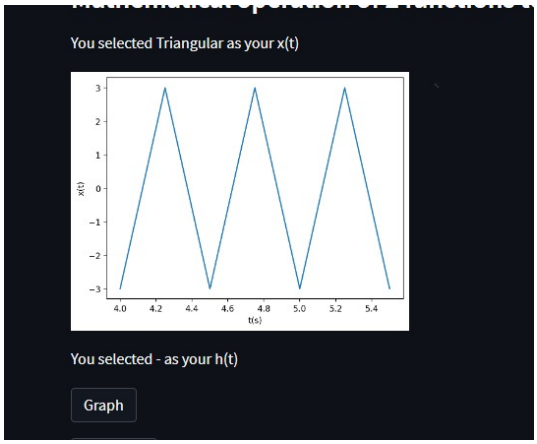

Figura 10: Triangular signal on continous domain

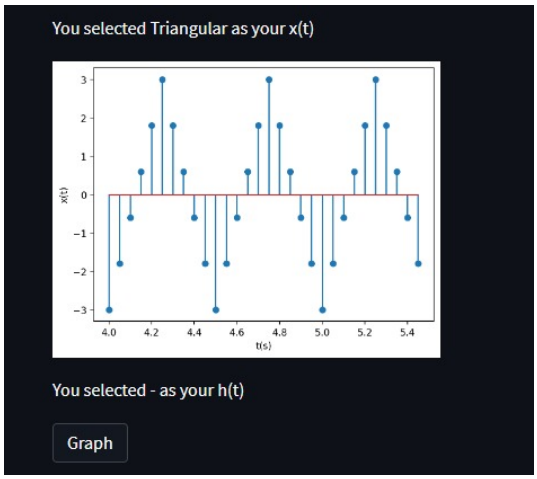

Figura 11: Triangular signal on discrete domain

In this image, it can be seen the different parameters that the user provide to the app, and how this it successfully translated into the graph following the condiction that the user said. On the other hand, on this section can be analyzed that the design of it, was done correctly cause the axes show a good view of the graphics with a great amplitude in both axes. Later on,the part of convolution using the command simps was tested and registered on the next figure.

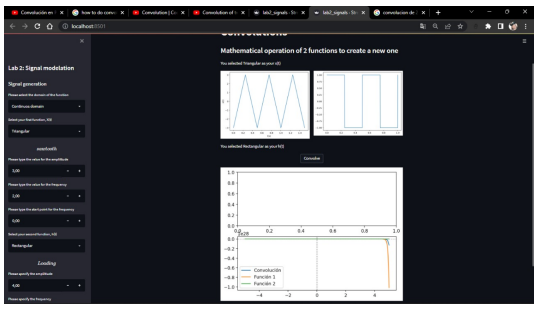

Figura 12: GUI visual adjust

as it can be seen this part of the code does not present the expected results it is estimated that the code has an error in te interpretation of the signals before past under the code line of the integration that define the convolution.

Then, the code using convolve on python was analyzed and the results will be represented on the following images:

Figura 13: Signals to convolve

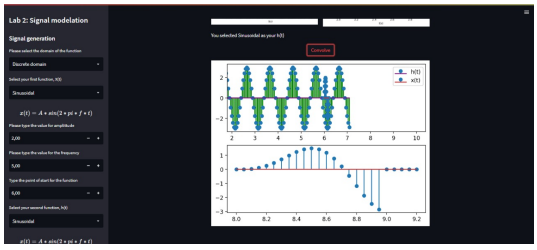

Figura 14: Results of convolution

As it can be seen, the practice was completed succesfully using th convolve code and the goals to present an excelent result when two signal on the discrete and continous domain are on the process of convolution were fulfilled.

4. Conclusions

It can be said that the main purpose of the laboratory was fulfilled, since the requirements were met. There was a process of learning since the beginning, because the there was a lack of code programming knowledge when trying to work with Python programming language. On the journey, it was a challenge to achieve the process of convolution in a dinamic way but it can be seen it was completed.

The laboratory helped in the understanding of the signals and their process of convolution in time, this was a good way to let the theory previously learned sink.

Referencias

[1] Tello Portillo, J. P. Introducción a las señales y sistemas. Universidad del Norte [2016].