Bu-Ali Sina University

Project

Singal and System

Mehran Shahidi

supervised by
Dr. Mahdi Abasi

February 9, 2020

Summary

This is Summary of my Project.

Contents

1 introduction

This the introduction of my project and the tool that I am using.

2 Plotting Signals

2.1 Q1.1

In this section, I show you how to implement **unit step** and **impulse** function and how to plot them with the different sampling rates with the use of **Numpy** and **Matplotlib** in python.

2.1.1 Unit Step

Unit step is a signal with magnitude one for time greater than zero . We can assume it as a dc signal which got switched on at time equal to zero. this is the mathematical definition of unit step:

$$x(t) = \begin{cases} 0 & : t < 0 \\ 1 & : t \geq 0 \end{cases} \tag{1}$$

The unit step function as it is shown in the **Listing 1**, takes one **Numpy** array or simple list as an input(samples of the time) and then return a list of output. this is list comprehension in Python that iterates through time samples and for each time samples if the sample is less than 0 it returns 0 and otherwise returns 1(this is the exact definition of the unit step function).

```
1 def unit_step(x):
2     return [0 if i<0 else 1 for i in x ]
```

Listing 1: **unit step** Signal

for the next part, we are using the function that was defined to plot the diagrams for sampling rates of $F_s = 100$ and $F_s = 1000$(**Listing 2**). the code is straightforward. first, it makes two grid in a row, and gets two axes and drawing the unit function with $F_s = 100$ in the first axis and $F_s = 1000$ in the second one. plots is shown in **Figure 1**.

```
1  t1= np.linspace(-10,10,200)
2  u = unit_step(t1)
3  fig, (ax1,ax2) = plt.subplots(1,2,figsize=(11,5))
4  ax1.plot(t1,u)
5  ax1.set_xlabel('t')
6  ax1.set_ylabel(r'$x(t)$')
7  ax1.set_title(r'$u(t)$ with sampling rate of $F_s=100$')
8  t2 = np.linspace(-10,10,2000)
9  u2 = unit_step(t2)
10 ax2.plot(t2,u2)
11 ax2.set_xlabel('t')
12 ax2.set_ylabel(r'$x(t)$')
13 ax2.set_title(r'$u(t)$ with sampling rate of $F_s=1000$')
14 plt.savefig('doc/images/Q1-1-unit.pgf')
15 fig.show()
```

Listing 2: Plotting **unit step** with two sampling rates

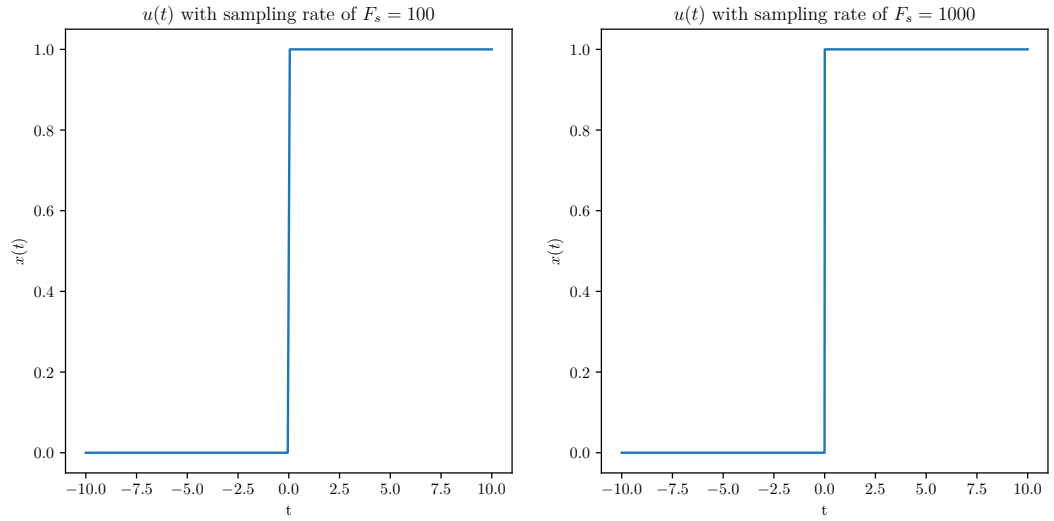

Figure 1: Graph result of the **Listing 2**

2.1.2 Impulse

The pulse function, also called the impulse function, in DT is easy: everywhere zero except at n=0 where the values is 1. The DT pulse is written as $\delta[n]$.

In CT it is more difficult. The pulse in CT is written as $\delta(t)$. It is mathematically defined with the sifting property:

$$x(0) = \int_{-\infty}^{\infty} x(t)\delta(t)dt \tag{2}$$

A usefull way to think about a pulse signal is to consider a signal in which you want to concentrate a finite amount of energy in a short a time interval as possible. Energy in a signal is measured by integrating a signal over a period of time.

Consider for instance the signal $x_a(t)$ defined as:

$$x_a(t) = \begin{cases} \frac{1}{2a} & : -a \leq t \leq a \\ 0 & : \text{elsewhere} \end{cases} \tag{3}$$

The total energy in this signal is 1 (the integral of $x(t)$ over the entire domain). This is true irrespective of the value of a. In the limit when $a \to 0$ the value $x_a(0) \to \infty$ but still the total area under the function remains 1. In a sloppy way we may define:

$$\delta(t) = \lim_{a \to 0} x_a(t)$$

For the implementation of the Impulse signal in python, we are using this sloppy definition. Because even for plotting continuous-time signals in python, we use an array of time samples. So we are going to implement the Impulse approximately by considering our sampling rate.

The implementation has shown in **Listing 3**. The function takes four inputs. The first one is the array or list of time samples, the second and third ones are low-bound and upper-bound of the time samples and the fourth one is the number of sampling. The function uses these arguments in addition to the first one(array of time samples) to find the closest samples to zero in lesser time that is needed. after finding those time samples create the smallest rectangular signal in a way that its integral becomes 1.

```python
def impulse(t,start,end,step):
    result= np.zeros_like(t)
    dt= (end-start)/step
    n =int((0 - start)//dt)
    if [n]==0:
        v = 1/(t[n+1]-t[n-1])
        result[n]=v
        result[n-1] = v
        result[n+1] = v
    else:
        v = 1/(t[n+1]-t[n])
        result[n] = v
        result[n+1] = v

    return result
```

Listing 3: **Impulse** Signal implementation

For the next part, we are using the function that was defined to plot the diagrams for sampling rates of $F_s = 100$ and $F_s = 1000$(**Listing 4**). the code is straightforward. first, it makes two grid in a row, and gets two axes and drawing the unit function with $F_s = 100$ in the first axis and $F_s = 1000$ in the second one. plots is shown in **Figure 2**.

```python
t1 = np.linspace(-10,10,200)
u = impulse(t1,-10,10,200)
fig, (ax1,ax2) = plt.subplots(1,2,figsize=(11,5))
ax1.plot(t1,u)
ax1.set_xlabel('t')
ax1.set_ylabel(r'$x(t)$')
ax1.set_title(r'$\delta (t)$ with sampling rate of $F_s=100$')
t2 = np.linspace(-10,10,2000)
u = impulse(t2,-10,10,2000)
ax2.plot(t2,u)
ax2.set_xlabel('t')
ax2.set_ylabel(r'$x(t)$')
ax2.set_title(r'$\delta (t)$ with sampling rate of $F_s=1000$')
plt.savefig('doc/images/Q1-1-delta.pgf')
fig.show()
```

Listing 4: Plotting **impulse** with two sampling rates

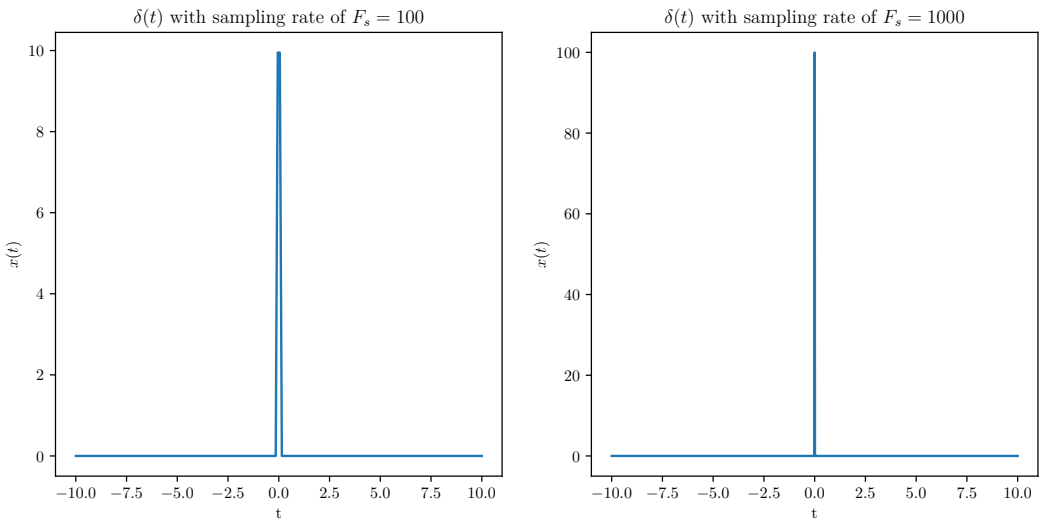

Figure 2: Graph result of the **Listing 4**

2.2 Q1.2

3 Signal Analysis

This is answer to Question2

4 Systems

This is the answer to Question3

5 Signal Energy

This is the answer to Question 4.

6 Convolution

This is the answer to Question5.