Bu-Ali Sina University

Project

Singal and System

Mehran Shahidi

supervised by
Dr. Mahdi Abasi

February 10, 2020

Summary

This is Summary of my Project.

Contents

1 introduction **1**

2 Plotting Signals **1**

2.1 Q1.1 1

2.1.1 Unit Step 1

2.1.2 Ramp 2

2.2 Q1.2 3

2.3 Q1.3 5

2.4 Q1.4 6

3 Signal Analysis **8**

4 Systems **8**

5 Signal Energy **8**

6 Convolution **8**

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 **ramp** 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).

```python
def unit(x):
    return np.array([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**.

```python
# create a figure with two axes in a row
fig, (ax1,ax2) = plt.subplots(1,2,figsize=(11,5))
#sampling with rate of 10
t1= np.linspace(-10,10,200)
u = unit(t1)
ax1.plot(t1,u)
ax1.set_xlabel('t')
ax1.set_ylabel(r'$x(t)$')
ax1.set_title(r'$u(t)$ with sampling rate of $F_s=100$')

#sampling with rate of 100
t2 = np.linspace(-10,10,2000)
u2 = unit(t2)
ax2.plot(t2,u2)
ax2.set_xlabel('t')
ax2.set_ylabel(r'$x(t)$')
ax2.set_title(r'$u(t)$ with sampling rate of $F_s=1000$')
plt.savefig('doc/images/Q1-1-unit.pgf')
fig.show()
```

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

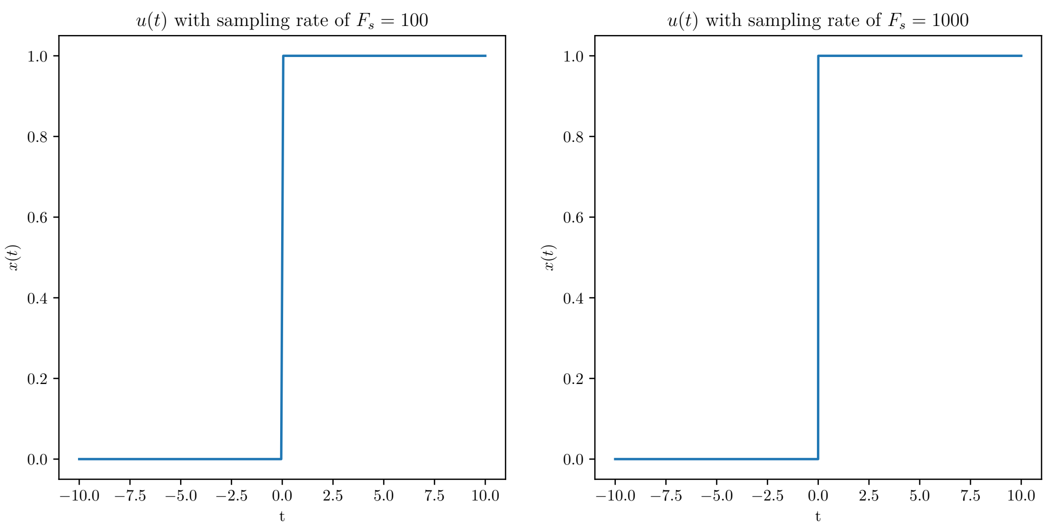

Figure 1: Graph result of the **Listing 2**

2.1.2 Ramp

The ramp signal definition is shown in expression 2. For the negative values, it returns zero and otherwise returns the value of the input. The implementation is also straightforward as the definition. You can see the implementation in python in **Listing 3**. The function takes one input as a simple list or **Numpy** array and returns the output list in the same structure that has explained for the unit signal implementation.

$$x(t) = \begin{cases} x & : x \geq 0 \\ 0 & : x < 0 \end{cases} \tag{2}$$

```python
def ramp(x):
    return np.array([0 if i<0 else i for i in x])
```

Listing 3: **Ramp** Signal implementation

For the next part, we are using the function that was defined to plot graphs 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
# create a figure with two axes in a row
fig, (ax1,ax2) = plt.subplots(1,2,figsize=(11,5))

#sampling with rate of 10
t1= np.linspace(-10,10,200)
u = ramp(t1)
ax1.plot(t1,u)
ax1.set_xlabel('t')
ax1.set_ylabel(r'$x(t)$')
ax1.set_title(r'$r(t)$ with sampling rate of $F_s=100$')

#samping with rate of 100
t2 = np.linspace(-10,10,2000)
u2 = ramp(t2)
ax2.plot(t2,u2)
ax2.set_xlabel('t')
ax2.set_ylabel(r'$x(t)$')
```

```python
ax2.set_title(r'$r(t)$ with sampling rate of $F_s=1000$')
plt.savefig('doc/images/Q1-1-ramp.pgf')
fig.show()
```

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

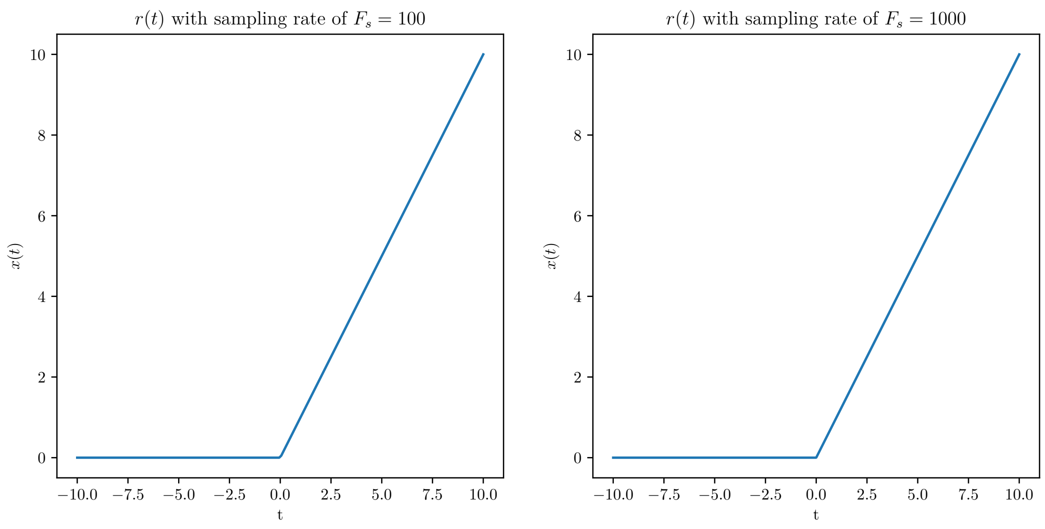

Figure 2: Graph result of the **Listing 4**

2.2 Q1.2

In this part, we are going to plot an example graph using what we have defined before. The example has shown in the **Figure 3**.

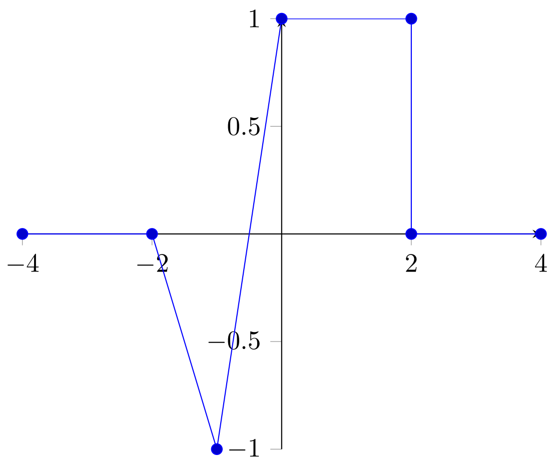

Figure 3: example graph of the siganl $x(t)$

So now, we define our graph using ramp and unit signal and plot them using that definition in Python. We split our graph into three parts. our first part is from $-\infty$ to -1 that is equal to $-r(t+2)*u(-t-1)$. The second part is from -1 to 0 that is equal to $(r(2*t+2)-1)*(u(t+1)-u(-t))$. The third part is from 0 to ∞ that is equal to $u(t)-u(t-2)$. Our final definition for our signal is :

$$x(t) = -r(t+2)*u(-t-1) + (r(2*t+2)-1)*(u(t+1)-u(-t)) + u(t) - u(t-2) \tag{3}$$

The implementation of the signal in Python has shown in the **Listing 5**. First, each part calculated separately, and then the main signal is plotted as the sum of each part. The result of code also has shown in the **Figure 4**.

```python
# create grid for the figure
gridsize = (3, 3)
fig = plt.figure(figsize=(16,11))
# The main siganl is on the right with two column to make it bigger
ax = plt.subplot2grid(gridsize, (0,1), rowspan=3,colspan=2)
ax1 = plt.subplot2grid(gridsize, (0,0))
ax2 = plt.subplot2grid(gridsize,(1,0))
ax3 = plt.subplot2grid(gridsize, (2,0))

# Sampling with the rate 8/1000
t= np.linspace(-4,4,1000)
# Parts of function
x1 = -ramp(t+2) *unit(-t-1)
x2 = (ramp(2*t+2)-1) * (unit(t+1)-unit(t))
x3 = unit(t) - unit(t-2)
x = x1+x2+x3

# Plotting first part
ax1.plot(t,x1)
ax1.set_title(r'$x_1(t)$')
ax1.set_xticks(np.arange(-4,4,step=1))

# Plotting second part
ax2.plot(t,x2)
ax2.set_title(r'$x_2(t)$')
ax2.set_xticks(np.arange(-4,4,step=1))

# Plotting third part
ax3.plot(t,x3)
ax3.set_title(r'$x_3(t)$')
ax3.set_xticks(np.arange(-4,4,step=1))

# Plotting the full signal
ax.plot(t,x)
ax.set_xlabel('t')
ax.set_yticks(np.arange(-1,2,step=1))
ax.set_title(r'$x(t) = x_1(t) + x_2(t) + x_3(t)$')
ax.set_xticks(np.arange(-4,4,step=1))

plt.savefig('doc/images/Q1-2-ex1.pgf')
fig.show()
```

Listing 5: Plotting signal $x(t)$ and its parts

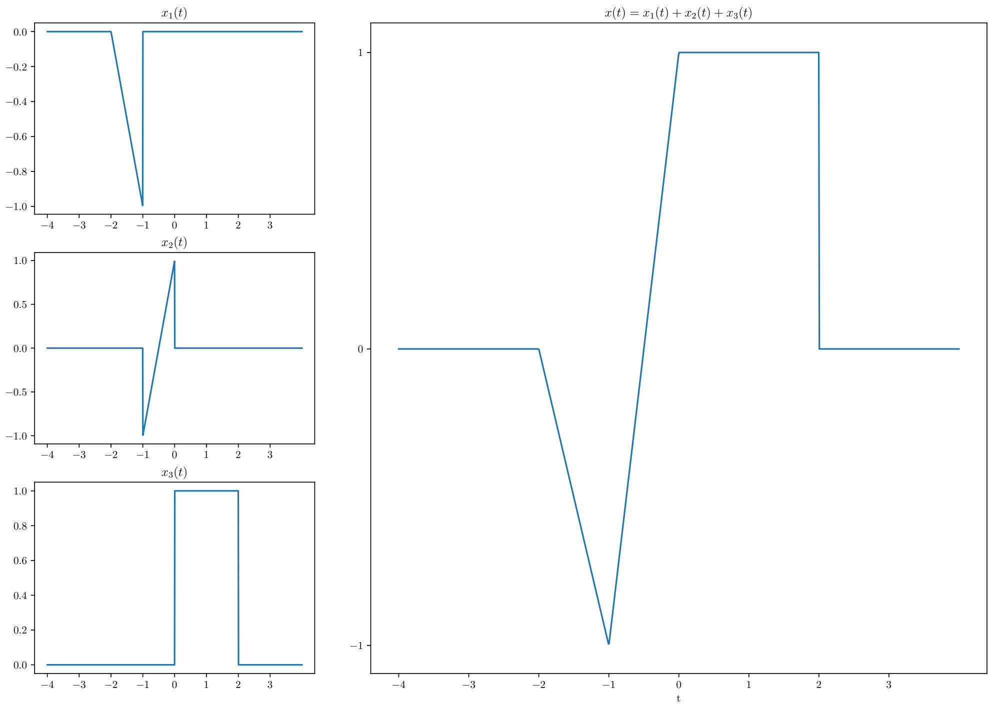

Figure 4: Graph result of the **Listing 5**

2.3 Q1.3

In this part, we are going to plot the signal $y(t)$ that has shown in the **Figure 5**. we can find the signal $y(t)$ concerning $x(t)$ that has defined in the previous section. We can find $y(t)$ by doing some transformation on $x(t)$. One way to see it is to first shift $x(t)$ 2 unit to the left and then compress it with the scale number of 2.

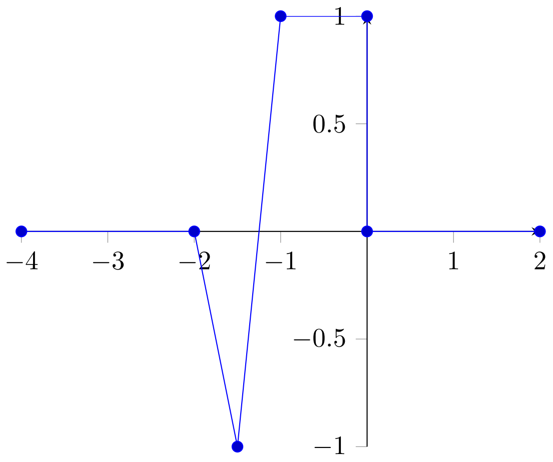

Figure 5: example graph of the siganl $y(t)$

So we can easily find $y(t)$ in respect to $x(t)$ just that is $y(t) = x(2t + 2)$. For implementation in Python we do the same. after defining function $x(t)$ we find $y(t)$ in a same manner. The code has shown in **Listing 6**. And the corospnding graph result has shown in **Figure 6**.

```python
fig, ax = plt.subplots(1,1)

t = np.linspace(-4,2,1000)
# Definition of the x(t) signal as a lambda function
x = lambda t : -ramp(t+2) *unit(-t-1) +\
               (ramp(2*t+2)-1) * (unit(t+1)-unit(t)) +\
               unit(t) - unit(t-2)
# Shifting x(t) to the left and compress it
y = x(2*t+2)
ax.plot(t,y)
ax.set_xlabel('t')
ax.set_ylabel(r'$y(t)$')
ax.set_yticks(np.arange(-1,2,step=1))

plt.savefig('doc/images/Q1-3-ex2.pgf')
fig.show()
```

Listing 6: Plotting signal $y(t)$

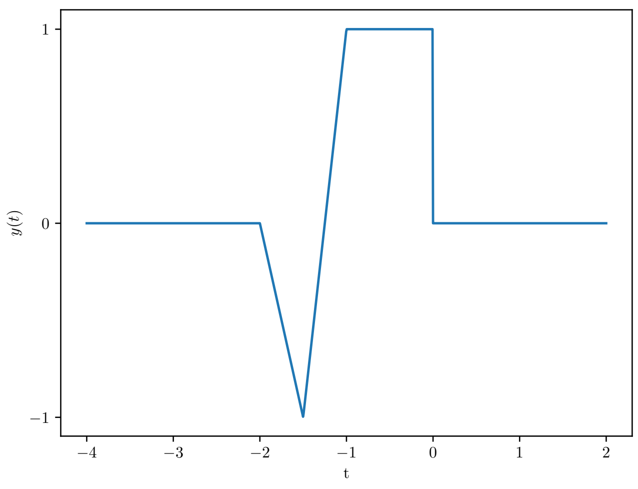

Figure 6: Graph result of the **Listing 6**

2.4 Q1.4

In this part, we are going to plot another signal, $y_2(t)$ that has shown in the **Figure 7**. The process is like the previous section. We can find $y_2(t)$ by first shifting the $x(t)$ to the left and then mirror it with respect to y axis.

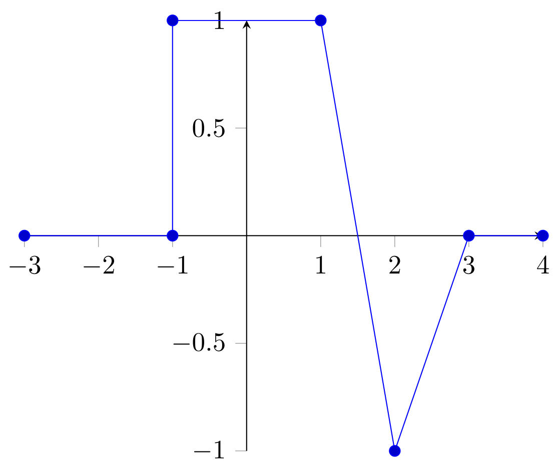

Figure 7: example graph of the siganl $y_2(t)$

So we can easily find $y_2(t)$ with respect to $x(t)$ that is $y_2(t) = x(-t+1)$. For implementation in Python we do the same. after defining function $x(t)$ we find $y_2(t)$ in a same manner. The code has shown in **Listing 7**. And the corospnding graph result has shown in **Figure 8**.

```python
fig, ax = plt.subplots(1,1)

t = np.linspace(-3,4,1000)
# Definition of the x(t) signal as a lambda function
x = lambda t : -ramp(t+2) *unit(-t-1) +\
               (ramp(2*t+2)-1) * (unit(t+1)-unit(t)) +\
               unit(t) - unit(t-2)
# Shifting x(t) to the left and mirror it
y2 = x(-t+1)
ax.plot(t,y2)
ax.set_xlabel('t')
ax.set_ylabel(r'$y_2(t)$')
ax.set_yticks(np.arange(-1,2,step=1))

plt.savefig('doc/images/Q1-4-ex3.pgf')
fig.show()
```

Listing 7: Plotting signal $y_2(t)$

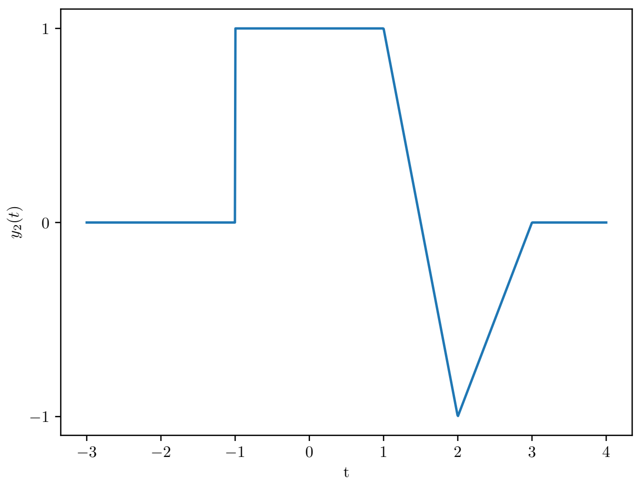

Figure 8: Graph result of the **Listing 7**

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.