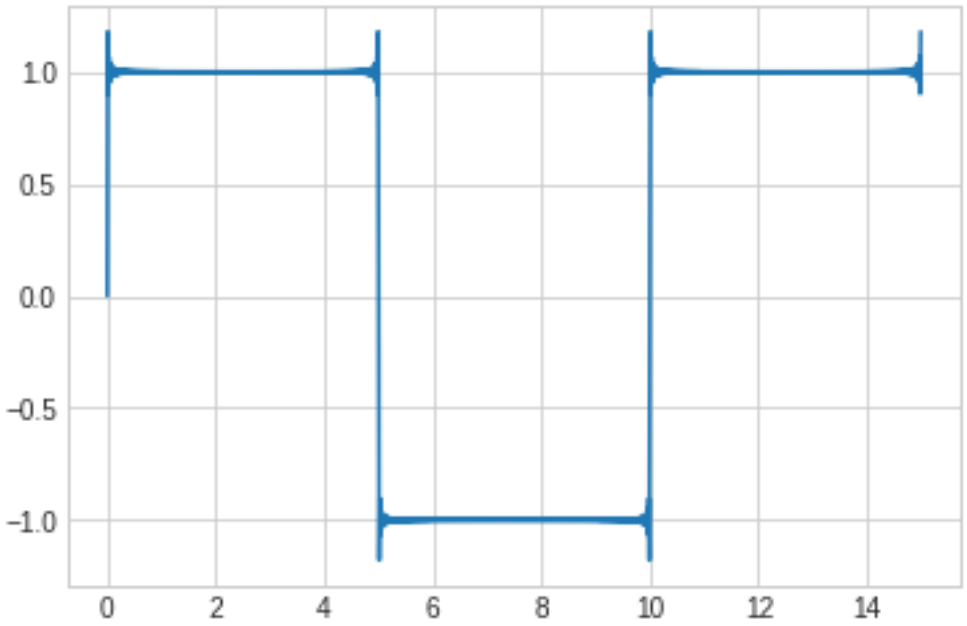
st121413-lab2

August 28, 2020

[1]:
```
import numpy as np
import matplotlib as mpl
import matplotlib.pyplot as plt
plt.style.use('seaborn-whitegrid')

from scipy import signal
import scipy.integrate as integrate

def createSubplot(n):
    fig,ax = plt.subplots(n,figsize=(10,10))
    fig.tight_layout(pad=3.0)
    return fig,ax
```

[2]:
```
# # T = 1/f
# a0 = integrate.quad(funcAn,0,T)[0] / T
# a0
# b0 = integrate.quad(funcBn,0,T)[0] *2  / T
# b0
t = np.arange(0,15,0.01)
s = np.zeros(len(t))
for i in range(1,500,2):
    # print(i)
    coff = (4/(np.pi*i))
    v = coff * np.sin(i * np.pi / 5 * t)
    s = s + v
    # sum = sum + (4/(pi*i))
# print(sum,b0)

plt.plot(t,s)
```

[2]:
```
[<matplotlib.lines.Line2D at 0x7f865140d6d0>]
```



1. Implement Fourier Series (Sinusoial form) as a function
2. Generate 3 functions using symbolic constant

- square wave
- sawtooth
- your choice

3. show the output in N = 10,50,100

1 Lab 2

1. Implement Fourier Series (Sinusoial form) as a function

[3]:
```
def genFourierSeries(wave, time, freq, N=10):
    """
    wave: the function of the waveform with argument t,
    time: array of time in numpy
    freq: frequncy of the wave
       N: number of sine + cos wave
    """
    w = 2 * np.pi * f
    T = 1/f
    def f_a0(t,f):
        s1 = Vn(t,f)
        return s1
```

```
    def f_an(t,f,n):
        s1 = Vn(t,f)
        s2 = np.cos(n * 2 * np.pi * f * t)
        return s1 * s2

    def f_bn(t,f,n):
        s1 = Vn(t,f)
        s2 = np.sin(n * 2 * np.pi * f * t)
        return s1 * s2

    a0 = (1/T) * integrate.quad(f_a0,0,T,args=(f))[0]
    s = np.ones(len(t)) + a0
    for n in np.arange(1,N+1,1):
        an = (2/T) * np.cos(n * 2 * np.pi * f * t) * (integrate.
↪quad(f_an,0,T,args=(f,n))[0])
        bn = (2/T) * np.sin(n * 2 * np.pi * f * t) * (integrate.
↪quad(f_bn,0,T,args=(f,n))[0])
        s = s + an + bn

    return s - 1
```

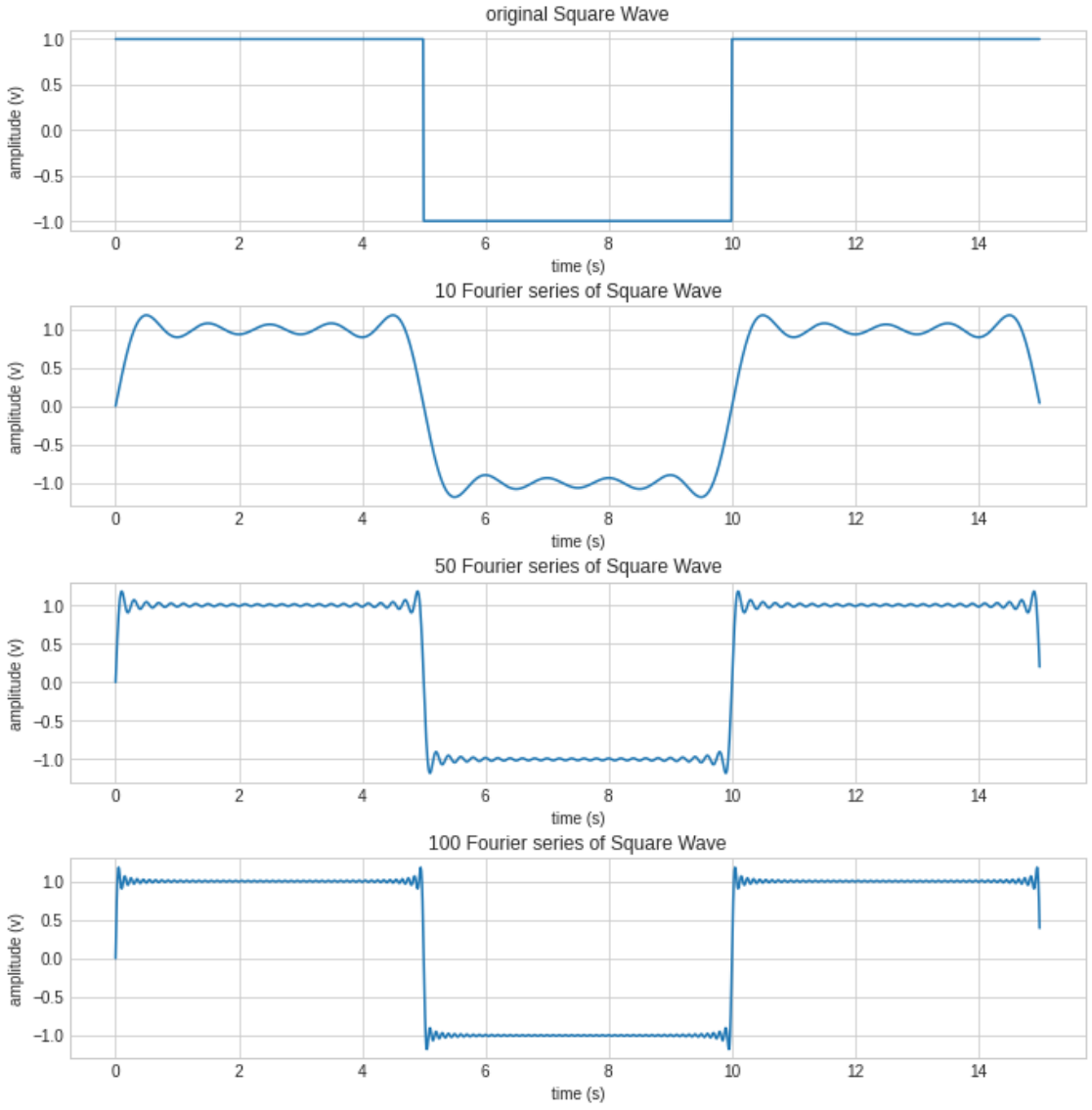
2. Generate 3 functions using symbolic constant

2.1 square wave

[4]:
```
def Vn(t,f):
    return signal.square(2 * np.pi * f * t)

t = np.arange(0,15,0.01)
f = 0.1
s = Vn(t,f)
fig, ax = createSubplot(4)
ax[0].plot(t,s)
ax[0].set_xlabel('time (s)')
ax[0].set_ylabel('amplitude (v)')
ax[0].set_title('original Square Wave')

N = 10
s = genFourierSeries(wave = Vn, freq=f, time = t, N = N)
ax[1].plot(t,s)
ax[1].set_xlabel('time (s)')
ax[1].set_ylabel('amplitude (v)')
ax[1].set_title(f"{N} Fourier series of Square Wave")

N = 50
s = genFourierSeries(wave = Vn, freq=f, time = t, N = N)
ax[2].plot(t,s)
ax[2].set_xlabel('time (s)')
```

```
ax[2].set_ylabel('amplitude (v)')
ax[2].set_title(f"{N} Fourier series of Square Wave")

N = 100
s = genFourierSeries(wave = Vn, freq=f, time = t, N = N)
ax[3].plot(t,s)
ax[3].set_xlabel('time (s)')
ax[3].set_ylabel('amplitude (v)')
ax[3].set_title(f"{N} Fourier series of Square Wave")
```

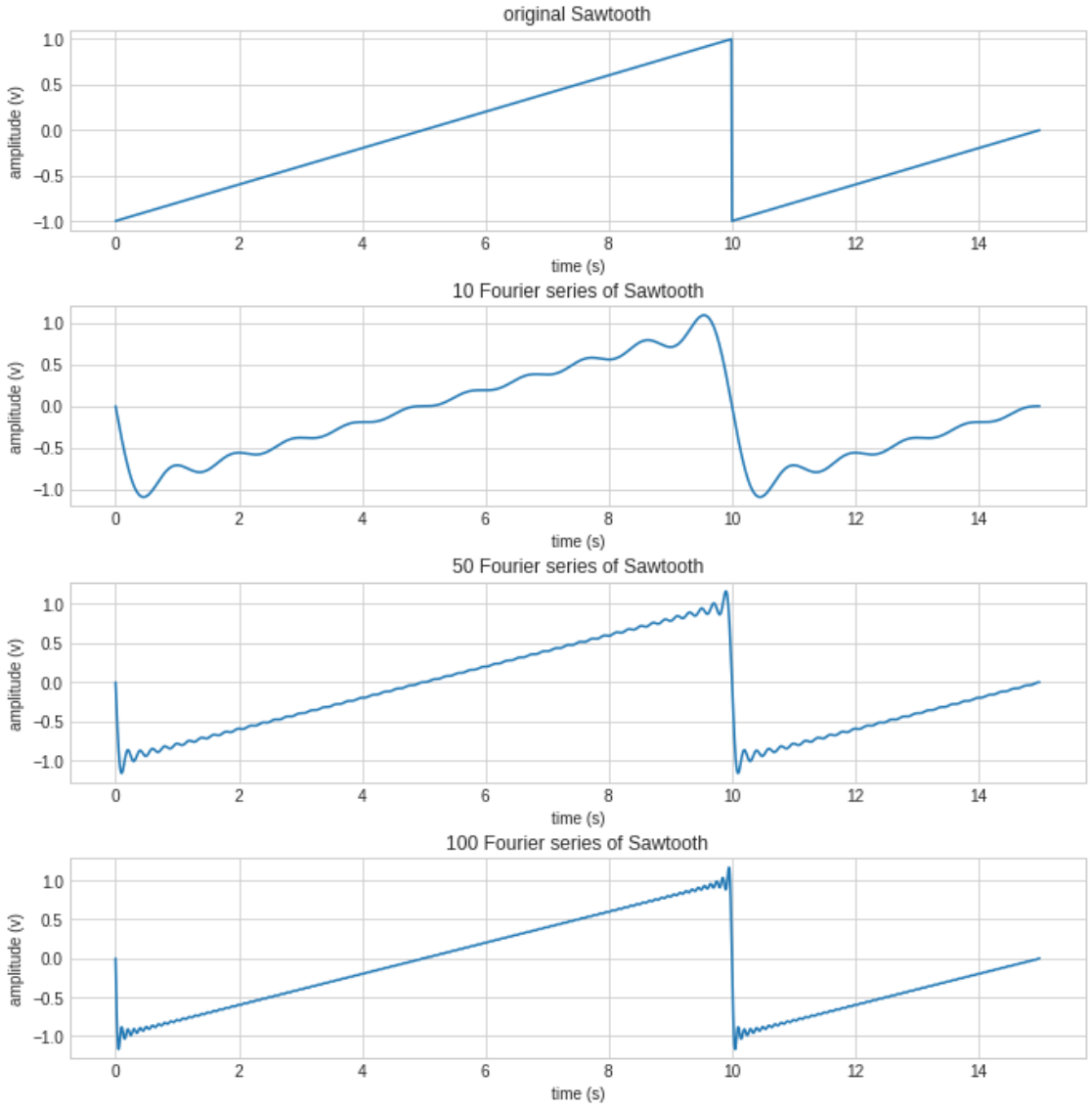
[4]: Text(0.5, 1.0, '100 Fourier series of Square Wave')



2.2 sawtooth

[5]:
```
def Vn(t,f):
    return signal.sawtooth(2 * np.pi * f * t)

t = np.arange(0,15,0.01)
f = 0.1
s = Vn(t,f)
fig, ax = createSubplot(4)
ax[0].plot(t,s)
ax[0].set_xlabel('time (s)')
ax[0].set_ylabel('amplitude (v)')
ax[0].set_title('original Sawtooth')

N = 10
s = genFourierSeries(wave = Vn, freq=f, time = t, N = N)
ax[1].plot(t,s)
ax[1].set_xlabel('time (s)')
ax[1].set_ylabel('amplitude (v)')
ax[1].set_title(f"{N} Fourier series of Sawtooth")

N = 50
s = genFourierSeries(wave = Vn, freq=f, time = t, N = N)
ax[2].plot(t,s)
ax[2].set_xlabel('time (s)')
ax[2].set_ylabel('amplitude (v)')
ax[2].set_title(f"{N} Fourier series of Sawtooth")

N = 100
s = genFourierSeries(wave = Vn, freq=f, time = t, N = N)
ax[3].plot(t,s)
ax[3].set_xlabel('time (s)')
ax[3].set_ylabel('amplitude (v)')
ax[3].set_title(f"{N} Fourier series of Sawtooth")
```

[5]:
```
Text(0.5, 1.0, '100 Fourier series of Sawtooth')
```

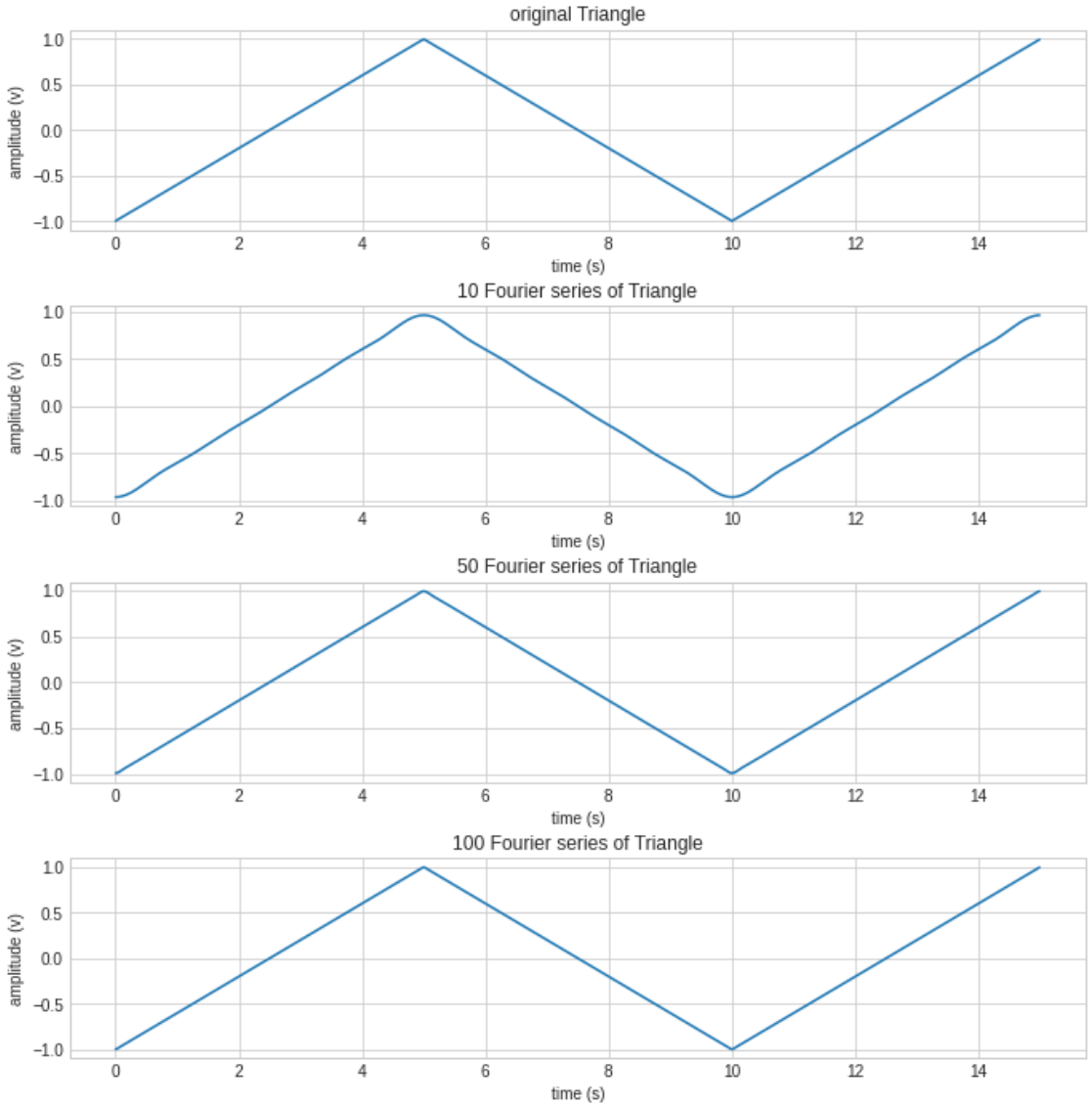


2.3 my choice - triangle

[6]:
```
def Vn(t,f):
    # Triangle in python use sawtooth with peak at 0.5 -- the default peak at 1
    return signal.sawtooth(2 * np.pi * f * t,0.5)

t = np.arange(0,15,0.01)
f = 0.1
s = Vn(t,f)
fig, ax = createSubplot(4)
ax[0].plot(t,s)
ax[0].set_xlabel('time (s)')
ax[0].set_ylabel('amplitude (v)')
```

```
ax[0].set_title('original Triangle')

N = 10
s = genFourierSeries(wave = Vn, freq=f, time = t, N = N)
ax[1].plot(t,s)
ax[1].set_xlabel('time (s)')
ax[1].set_ylabel('amplitude (v)')
ax[1].set_title(f"{N} Fourier series of Triangle")

N = 50
s = genFourierSeries(wave = Vn, freq=f, time = t, N = N)
ax[2].plot(t,s)
ax[2].set_xlabel('time (s)')
ax[2].set_ylabel('amplitude (v)')
ax[2].set_title(f"{N} Fourier series of Triangle")

N = 100
s = genFourierSeries(wave = Vn, freq=f, time = t, N = N)
ax[3].plot(t,s)
ax[3].set_xlabel('time (s)')
ax[3].set_ylabel('amplitude (v)')
ax[3].set_title(f"{N} Fourier series of Triangle")
```

[6]: Text(0.5, 1.0, '100 Fourier series of Triangle')


original Triangle
amplitude (v)
time (s)
10 Fourier series of Triangle
amplitude (v)
time (s)
50 Fourier series of Triangle
amplitude (v)
time (s)
100 Fourier series of Triangle
amplitude (v)
time (s)