Mathematical Physics - III (Practical)
Paper: PHS-A-CC-4-8-P

Fourier series:
Evaluate the fourier co-efficients of a given periodic function (square wave).

Periodic functions:
A function f(x) is said to be periodic if there exists a number T > 0 such that f(x + T) = f(x) for every x. The smallest such T is called the period of f(x).

Intiutively, periodic functions have repetitive behavior. A periodic function can be defined on a finite interval, then copied and pasted so that it repeats itself.

Fourier Series:

Let L> 0 be a fixed number and f(x) be a periodic function with period 2L, defined on (−L,L). The Fourier series of f(x) is a way of expanding the function f(x) into an infinite series involving sines and cosines:

$$f(x) = \frac{a_0}{2} + \sum_{n=1}^{\infty} \left(a_n \cos\frac{n\pi x}{L} + b_n \sin\frac{n\pi x}{L} \right)$$

where a_0, a_n, and b_n are called the Fourier coefficients of f(x), and are given by the formulas

$$a_0 = \frac{1}{L}\int_{-L}^{L} f(x)dx$$

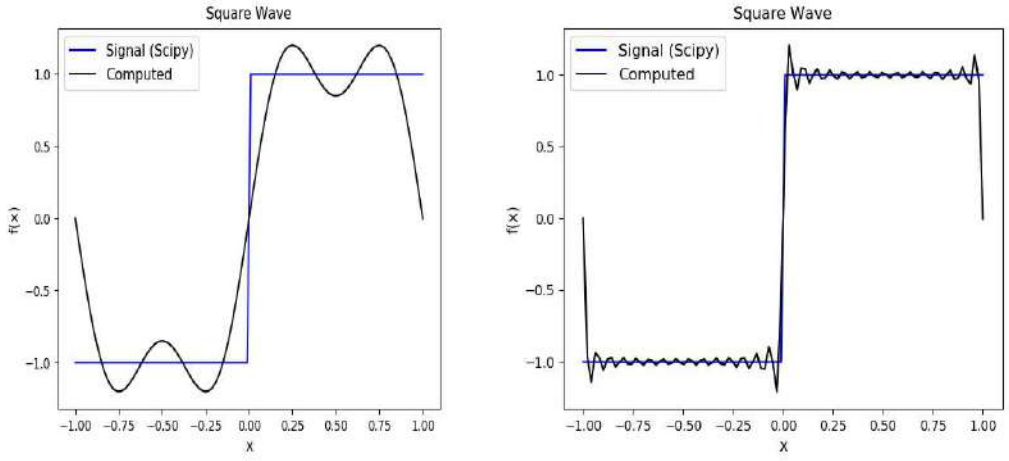
$$a_n = \frac{1}{L}\int_{-L}^{L} f(x)\cos\left(\frac{n\pi x}{L}\right)dx$$

$$b_n = \frac{1}{L}\int_{-L}^{L} f(x)\ \sin\left(\frac{n\pi x}{L}\right)dx$$

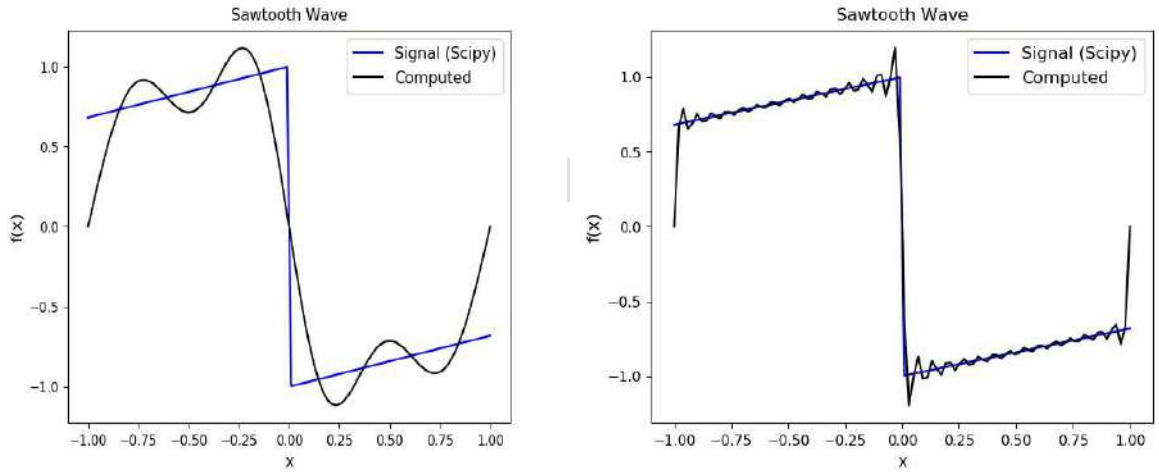
If we plot the first N non-zero terms, we get approximations of the function. In the following programme if increase the number of terms i.e. n=30 we would go to the very close to the original function.

1. The following programme is for the computation of Fourier coefficients and Fourier series of Square Wave, Saw-tooth and Triangular wave.

```
import numpy as np
from scipy.signal import square, sawtooth
from scipy.integrate import simps
import matplotlib.pyplot as plt

L        = 1   # periodicity
harmonics = 5; # total number of term in series
x = np.linspace(-L,L,100) # x-grid

# Square Wave Fourier coefficientsand fouries series
a0 = 1.0/L*simps(square(x),x)
an = lambda n: 1.0/L*simps(square(x)*np.cos(n*np.pi*x/L),x)
bn = lambda n: 1.0/L*simps(square(x)*np.sin(n*np.pi*x/L),x)

summsq = a0/2 + sum([an(k)*np.cos(k*np.pi*x/L) + bn(k)*np.sin(k*np.pi*x/L) for k
            in range(1,harmonics)])

plt.figure()
plt.plot(x, square(x), 'b-', label = 'Signal (Scipy)');
plt.plot(x, summsq,    'k-',  label = 'Computed');
plt.legend(loc='best', prop={'size':12})
plt.title("Square Wave", size=12)
plt.xlabel('x', size=12)
plt.xticks(size=10)
plt.ylabel('f(x)', size=12)
plt.yticks(size=10)
plt.show()
```

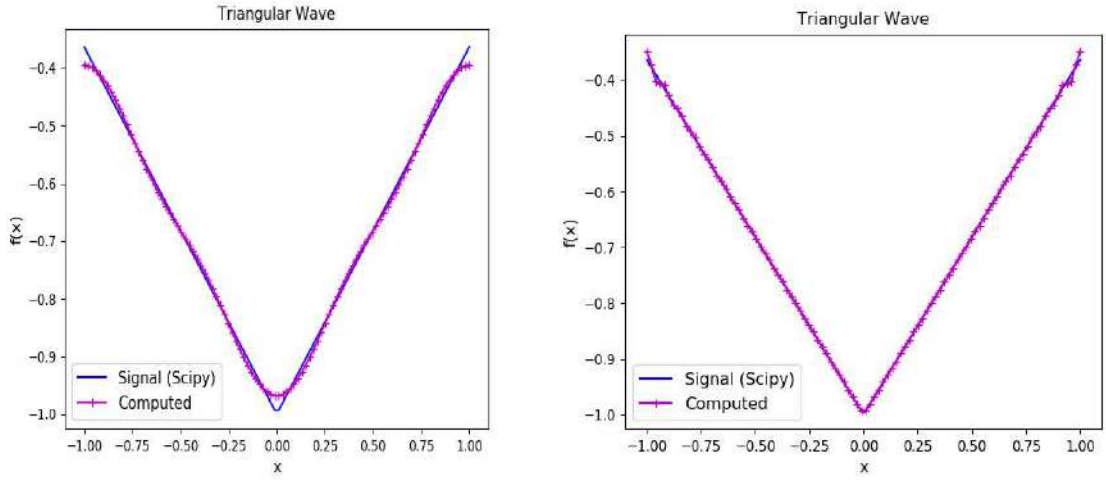



```
# Sawtooth Wave: Fourier coefficients and fourier series
a0 = 1.0/L*simps(sawtooth(x),x)
an = lambda n: 1.0/L*simps(sawtooth(x)*np.cos(n*np.pi*x/L),x)
bn = lambda n: 1.0/L*simps(sawtooth(x)*np.sin(n*np.pi*x/L),x)
summst = a0/2 + sum([an(k)*np.cos(k*np.pi*x/L) + bn(k)*np.sin(k*np.pi*x/L) for k
            in range(1,harmonics)])

plt.plot(x, sawtooth(x), 'b-', label = 'Signal (Scipy)');
plt.plot(x, summst,      'k-',  label = 'Computed');
plt.legend(loc='best', prop={'size':12})
plt.title("Sawtooth Wave", size=12)
plt.xlabel('x', size=12)
plt.xticks(size=10)
plt.ylabel('f(x)', size=12)
plt.yticks(size=10)
plt.show()
```




```
# Triangular Wave: Fourier coefficients and fourier series
a0 = 1.0/L*simps(sawtooth(x,width=0.5),x)
an = lambda n: 1.0/L*simps(sawtooth(x,width=0.5)*np.cos(n*np.pi*x/L),x)
bn = lambda n: 1.0/L*simps(sawtooth(x,width=0.5)*np.sin(n*np.pi*x/L),x)

summtr = a0/2 + sum([an(k)*np.cos(k*np.pi*x/L) + bn(k)*np.sin(k*np.pi*x/L) for k
            in range(1,harmonics)])

plt.plot(x, sawtooth(x,width=0.5), 'b-', label = 'Signal (Scipy)');
plt.plot(x, summtr,      'm+-',  label = 'Computed');
plt.legend(loc='best', prop={'size':12})
plt.title("Triangular Wave", size=12)
plt.xlabel('x', size=12)
plt.xticks(size=10)
```

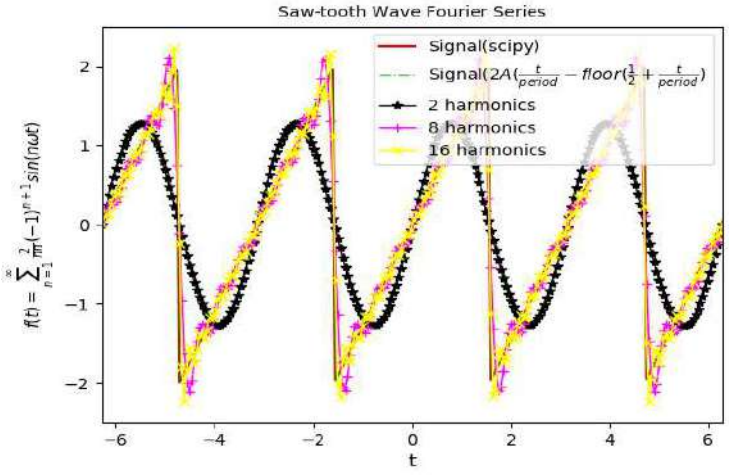
```
plt.ylabel('f(x)', size=12)
plt.yticks(size=10)
plt.show()
```



2. Generation of saw tooth wave form and then compute Fourier Coefficients and Fourier series:


```
import numpy as np
from scipy import signal
import matplotlib.pyplot as plt

A      = 2;      # Amplitude
period = np.pi;  # periodicity
t      = np.linspace(-2*period, 2*period, 256)
harmonics = 2;   # Harmonics

# generate saw-tooth waveform
def stwave(t, period):
   return A*2*(t/period - np.floor(.5+t/period))

# Fourier coefficients; an=0
def bn(n):
   return pow(-1,n+1)*2/(np.pi*n)

# generate angular frequency
def wn(n, period):
   return (2*np.pi*n)/period

# Fourier series
def fourierSt(harmonics,t,period):
   summ = 0
   for i in range(1, harmonics):
```

```
        summ += A*bn(i)*np.sin(wn(i,period)*t)
    return summ

# Main
y = []; f1 = []; f2 = []; f3 = []
for i in t:
    y.append(stwave(i,period))
    f1.append(fourierSt(  harmonics,i,period))
    f2.append(fourierSt(4*harmonics,i,period))
    f3.append(fourierSt(8*harmonics,i,period))

sg = A*signal.sawtooth(2*(t-period/2)) # Constructed signal
plt.plot(t, sg, '-',  lw='1.5', color="red",    label="Signal(scipy)")
plt.plot(t,  y, '-.', lw='1', color="green",   label=r'Signal$(2A(\frac{t}{period} -
floor({\frac{1}{2}+\frac{t}{period}})}$')
plt.plot(t, f1, '*-', lw='1', color="black",  label=str(harmonics)+" harmonics")
plt.plot(t, f2, '-+', lw='1', color="magenta",label=str(4*harmonics)+" harmonics")
plt.plot(t, f3, '-x', lw='1', color="yellow",  label=str(8*harmonics)+" harmonics")
plt.title("Saw-tooth Wave Fourier Series", size=10)
plt.legend(loc='best', prop={'size':10})
plt.xlabel('t', size=12)
plt.xticks(size=10)
plt.ylabel(r'$f(t)=\sum_{n=1}^\infty\;\frac{2}{n\pi}(-1)^{n+1} sin(n\omega t)$', size=10)
plt.yticks(size=10)
plt.xlim([-2*period, 2*period])
plt.ylim([-A-.5, A+.5])
plt.show()
```

Assignments: Repeat problem (2) for Triangular wave and Square Wave.