Analysis of Fourier series using Python Code

Dr. Shyamal Bhar
Department of Physics
Vidyasagar College for Women
Kolkata – 700 006

We know that there are many ways by which any complicated function may be expressed as power series. This is not the only way in which a function may be expressed as a series but there is a method of expressing a periodic function as an infinite sum of sine and cosine functions. This representation is known as Fourier series. The computation and study of Fourier series is known as harmonic analysis and is useful as a way to break up an arbitrary periodic function into a set of simple harmonic terms that can be plugged in, solved individually, and then recombined to obtain the solution to the original problem or an approximation to it to whatever accuracy is desired. Unlike Taylor series, a Fourier series can describe functions that are not everywhere continuous and/or differentiable. There are other advantages of using trigonometric terms. They are easy to differentiate and integrate and each term contain only one characteristic frequency. Analysis of Fourier series becomes important because this method is used to represent the response of a system to a periodic input and the response depends on the frequency content of the input.

Dirichlet Conditions: The conditions that a function $f(x)$ may be expressed as Fourier series are known as the Dirichlet conditions. The conditions are

i) The function must be periodic

ii) It must be single valued and continuous. There may a finite number of finite discontinuities

iii) It must have only a finite number of maxima and minima within one period

iv) The integral over one period of $|f(x)|$ must converge.

Fourier series makes of the orthogonality relationships of the sine and cosine functions. The integral over one period of the product of any two terms have the following properties:

$$\int_{x_0}^{x_0+L} \sin\left(\frac{2\pi nx}{L}\right)\cos\left(\frac{2\pi mx}{L}\right)dx = 0 \text{ for all } m \text{ and } n$$

$$\int_{x_0}^{x_0+L} \cos\left(\frac{2\pi nx}{L}\right)\cos\left(\frac{2\pi mx}{L}\right)dx = \begin{cases} L \text{ for } m = n = 0 \\ \frac{L}{2} \text{for } m = n > 0 \\ 0 \text{ for } m \neq n \end{cases}$$

$$\int_{x_0}^{x_0+L} \sin\left(\frac{2\pi nx}{L}\right)\sin\left(\frac{2\pi mx}{L}\right)dx = \begin{cases} L \text{ for } m = n = 0 \\ \frac{L}{2} \text{for } m = n > 0 \\ 0 \text{ for } m \neq n \end{cases}$$

So the Fourier series of the function $f(x)$ over the periodic interval $[0, L]$ is written as

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

where a_n and b_n are constants called the Fourier coefficients and

$$a_0 = \frac{2}{L}\int_0^L f(x')dx'$$

$$a_n = \frac{2}{L}\int_0^L f(x')\cos\left(\frac{2\pi nx'}{L}\right)dx'$$

$$b_n = \frac{2}{L}\int_0^L f(x')\sin\left(\frac{2\pi nx'}{L}\right)dx'$$

The Fourier series of the function $f(x)$ over the periodic interval $[-L, L]$ is written as

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

where,

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

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

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

The Fourier series of the function $f(x)$ over the periodic interval $[-\pi,\pi]$ is written as

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

where,

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

$$a_n = \frac{1}{\pi}\int_{-\pi}^{\pi} f(x')\cos(nx')\,dx'$$

$$b_n = \frac{1}{\pi}\int_{-\pi}^{\pi} f(x')\sin(nx')\,dx'$$

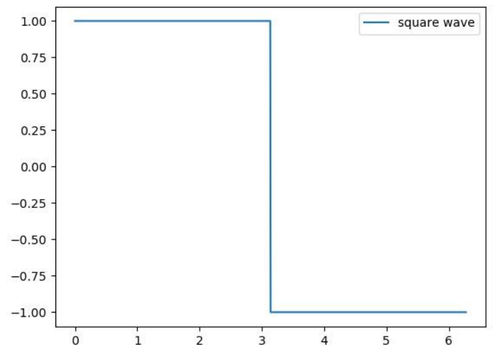
- **built-in piecewise continuous functions such as square wave, sawtooth wave and triangular wave**

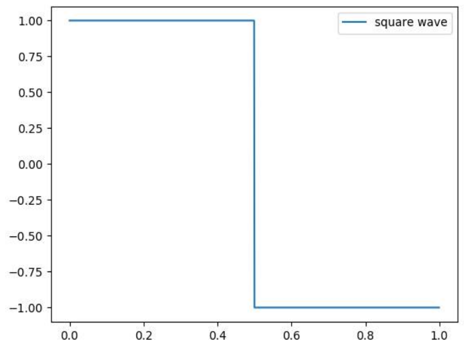
1. scipy.signal.square module

scipy.signal.square (x, duty=0.5)

Return a periodic square-wave waveform.

The square wave has a period 2*pi, has value +1 from 0 to 2*pi*duty and -1 from 2*pi*duty to 2*pi. *duty* must be in the interval [0,1].



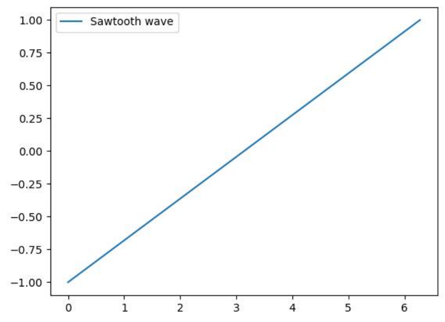


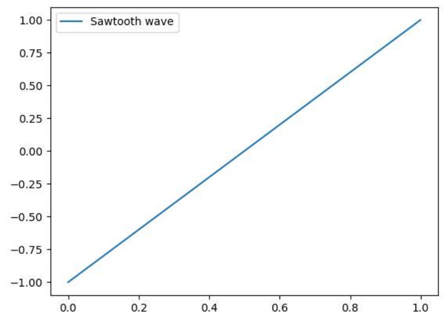
2. scipy.signal.sawtooth module

scipy.signal.sawtooth(x, width=1)

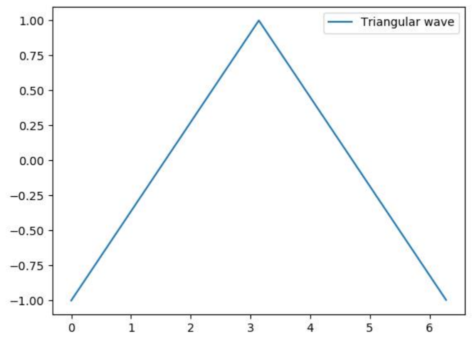
Return a periodic sawtooth or triangle waveform.

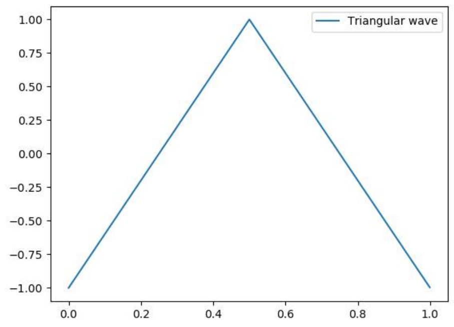
The sawtooth waveform has a period 2*pi, rises from -1 to 1 on the interval 0 to width*2*pi, then drops from 1 to -1 on the interval width*2*pi to 2*pi. *width* must be in the interval [0, 1].





❖ Triangular wave from sawtooth signal





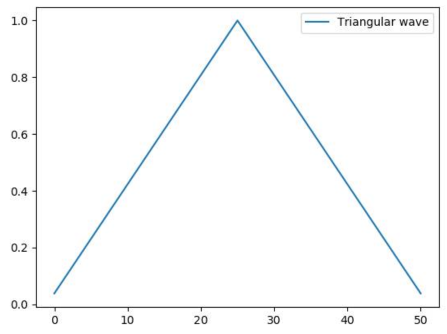
3. Scipy.signal.triang () module

scipy.signal.triang(M, sym=True)

Return a triangular window.

w : ndarray

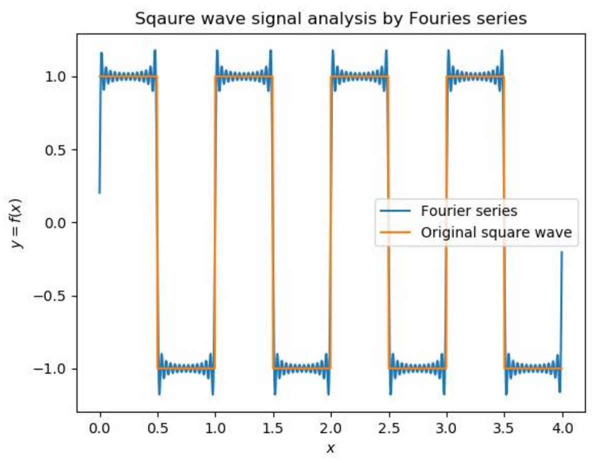
The window, with the maximum value normalized to 1 (though the value 1 does not appear if *M* is even and *sym* is True).



Example -1

```
# Fourier series analysis for a sqaure wave function
# user defined function

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

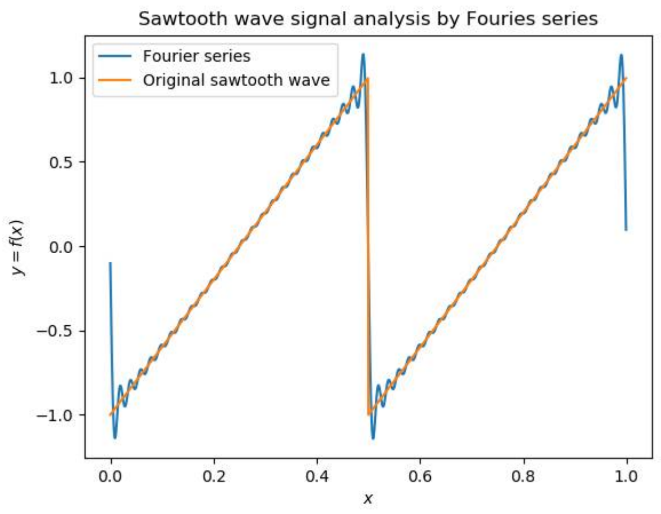
```
L=4                 # Periodicity of the periodic function f(x)
freq=4              # No of waves in time period L
dutycycle=0.5
samples=1000
terms=100

# Generation of square wave

x=np.linspace(0,L,samples,endpoint=False)
y=square(2.0*np.pi*x*freq/L,duty=dutycycle)

# Calculation of Fourier coefficients
a0=2./L*simps(y,x)
an=lambda n:2.0/L*simps(y*np.cos(2.*np.pi*n*x/L),x)
bn=lambda n:2.0/L*simps(y*np.sin(2.*np.pi*n*x/L),x)

# sum of the series
s=a0/2.+sum([an(k)*np.cos(2.*np.pi*k*x/L)+bn(k)*np.sin(2.*np.pi*
k*x/L) for k in range(1,terms+1)])

# Plotting

plt.plot(x,s,label="Fourier series")
plt.plot(x,y,label="Original square wave")
plt.xlabel("$x$")
plt.ylabel("$y=f(x)$")
plt.legend(loc='best',prop={'size':10})
plt.title("Sqaure wave signal analysis by Fouries series")
plt.savefig("fs_square.png")
plt.show()
```



Example-2 :

# Fourier series analysis for a sawtooth wave function

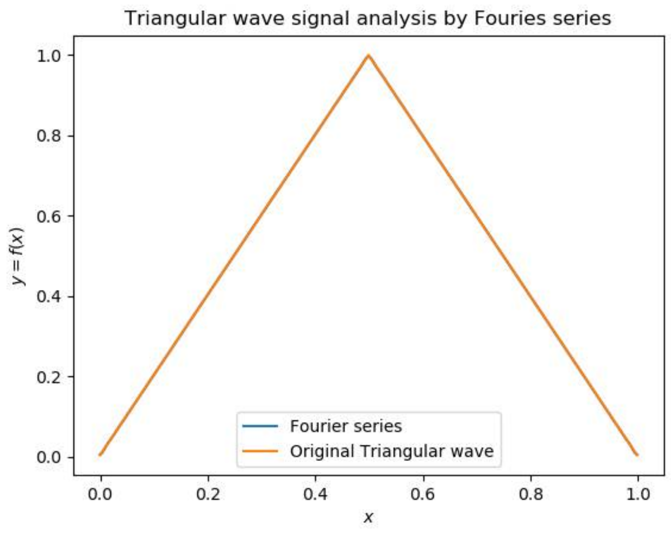
```
import numpy as np
from scipy.signal import square,sawtooth
import matplotlib.pyplot as plt
from scipy.integrate import simps


L=1                  # Periodicity of the periodic function f(x)
freq=2               # No of waves in time period L
width_range=1
samples=1000
terms=50

# Generation of Sawtooth function

x=np.linspace(0,L,samples,endpoint=False)
y=sawtooth(2.0*np.pi*x*freq/L,width=width_range)

# Calculation of Co-efficients
a0=2./L*simps(y,x)
an=lambda n:2.0/L*simps(y*np.cos(2.*np.pi*n*x/L),x)
bn=lambda n:2.0/L*simps(y*np.sin(2.*np.pi*n*x/L),x)

# Sum of the series
s=a0/2.+sum([an(k)*np.cos(2.*np.pi*k*x/L)+bn(k)*np.sin(2.*np.pi*
k*x/L) for k in range(1,terms+1)])

# Plotting

plt.plot(x,s,label="Fourier series")
plt.plot(x,y,label="Original sawtooth wave")
plt.xlabel("$x$")
plt.ylabel("$y=f(x)$")
plt.legend(loc='best',prop={'size':10})
plt.title("Sawtooth wave signal analysis by Fouries series")
plt.savefig("fs_sawtooth.png")
plt.show()
```



Example -3:

# Fourier series analysis for a Triangular wave function

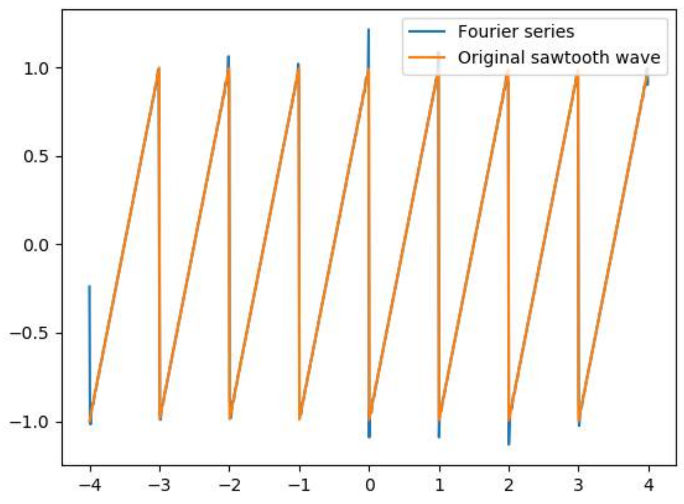
```
import numpy as np
from scipy.signal import square,sawtooth,triang
import matplotlib.pyplot as plt
from scipy.integrate import simps


L=1                  # Periodicity of the periodic function f(x)
samples=501
terms=50

# Generation of Triangular wave
x=np.linspace(0,L,samples,endpoint=False)
y=triang(samples)

# Fourier Coefficients
a0=2./L*simps(y,x)
an=lambda n:2.0/L*simps(y*np.cos(2.*np.pi*n*x/L),x)
bn=lambda n:2.0/L*simps(y*np.sin(2.*np.pi*n*x/L),x)

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

# Plotting
plt.plot(x,s,label="Fourier series")
plt.plot(x,y,label="Original Triangular wave")
plt.xlabel("$x$")
plt.ylabel("$y=f(x)$")
plt.legend(loc='best',prop={'size':10})
plt.title("Triangular wave signal analysis by Fouries series")
plt.savefig("fs_triangular.png")
plt.show()
```



Example-4

```
# Fourier series analysis for a sawtooth wave function
# User defined function

import numpy as np
import matplotlib.pyplot as plt
from scipy.integrate import simps


L=1.0 # half wavelength, Wavelength=2L
freq=2  #  frequency
samples=1001
terms=300

# Defining sawtooth function
x=np.linspace(-L,L,samples,endpoint=False)
f=lambda x: (freq*x%(2*L)-L)/L

# Fouriers coefficients

a0=1./L*simps(f(x),x)
an=lambda n:1.0/L*simps(f(x)*np.cos(1.*np.pi*n*x/L),x)
bn=lambda n:1.0/L*simps(f(x)*np.sin(1.*np.pi*n*x/L),x)
```

```
# Series sum
xp=4*x
s=a0/2.+sum([an(k)*np.cos(1.*np.pi*k*xp/L)+bn(k)*np.sin(1.*np.pi
*k*xp/L) for k in range(1,terms+1)])

# Plotting
plt.plot(xp,s,label="Fourier series")
plt.plot(xp,f(xp),label="Original sawtooth wave")
plt.legend(loc='best',prop={'size':10})
plt.savefig("saw_ud.png")
plt.show()
```



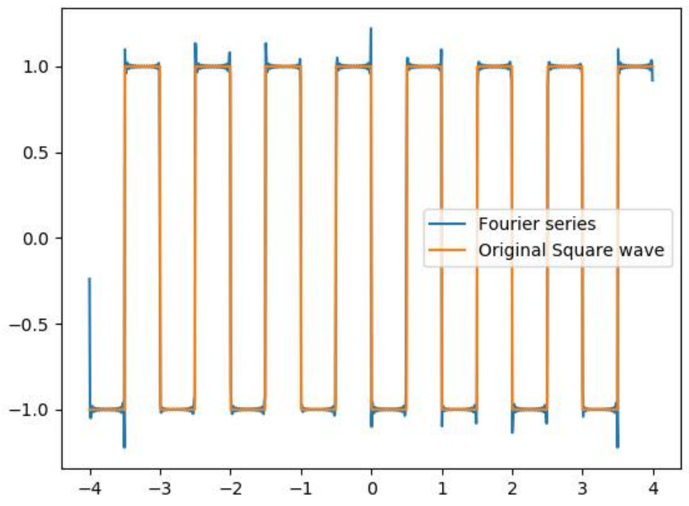
Example-5:

# Fourier series analysis for a square wave function

# User defined function

```
import numpy as np
import matplotlib.pyplot as plt
from scipy.integrate import simps

L=1.0 # half wavelength, Wavelength=2L
freq=2  #  frequency
samples=1001
terms=300
```

```
# Generating Square wave
x=np.linspace(-L,L,samples,endpoint=False)
F=lambda x: np.array([-1 if -L<=u<0 else 1 for u in x])

f=lambda x: F(freq*x%(2*L)-L)

# Fourier Coefficients
a0=1./L*simps(f(x),x)
an=lambda n:1.0/L*simps(f(x)*np.cos(1.*np.pi*n*x/L),x)
bn=lambda n:1.0/L*simps(f(x)*np.sin(1.*np.pi*n*x/L),x)

# Series sum
xp=4*x
s=a0/2.+sum([an(k)*np.cos(1.*np.pi*k*xp/L)+bn(k)*np.sin(1.*np.pi
*k*xp/L) for k in range(1,terms+1)])

#Plotting
plt.plot(xp,s,label="Fourier series")
plt.plot(xp,f(xp),label="Original Square wave")
plt.legend(loc='best',prop={'size':10})
plt.savefig("square_ud.png")
plt.show()
```

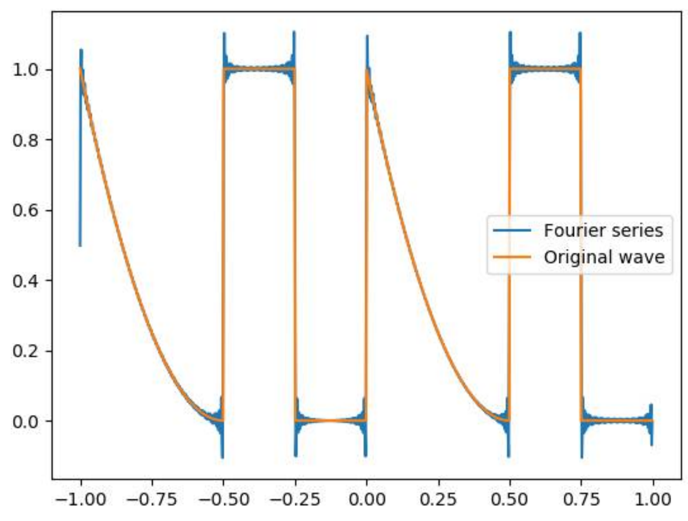


Example -6

# Fourier series analysis for a Arbitrary waves function

# User defined function

```
import numpy as np
import matplotlib.pyplot as plt
from scipy.integrate import simps

L=1.0 # half wavelength, Wavelength=2L
freq=2  #  frequency
samples=1001
terms=300

# Generating  wave
x=np.linspace(-L,L,samples,endpoint=False)
F=lambda x: np.array([u**2 if -L<=u<0 else 1 if 0<u<0.5 else 0
for u in x])
#F=lambda x: abs(np.sin(2*np.pi*x))

f=lambda x: F(freq*x%(2*L)-L)

# Fourier Coefficients
a0=1./L*simps(f(x),x)
an=lambda n:1.0/L*simps(f(x)*np.cos(1.*np.pi*n*x/L),x)
bn=lambda n:1.0/L*simps(f(x)*np.sin(1.*np.pi*n*x/L),x)

# Series sum
xp=x
s=a0/2.+sum([an(k)*np.cos(1.*np.pi*k*xp/L)+bn(k)*np.sin(1.*np.pi
*k*xp/L) for k in range(1,terms+1)])

#Plotting
plt.plot(xp,s,label="Fourier series")
plt.plot(xp,f(xp),label="Original wave")
plt.legend(loc='best',prop={'size':10})
plt.savefig("arb_ud.png")
plt.show()
```


1.0
0.8
0.6
0.4
0.2
0.0
−1.00
−0.75
−0.50
−0.25
0.00
0.25
0.50
0.75
1.00
Fourier series
Original wave