2020

Data Communication

Name-Ayush Patel
Sch No-191112253
Department-C.S.E.
Section-2

Assignment-1

SIN WAVE

Generate a sine wave of specific frequency for specified time duration. Sine waves represent periodic oscillations.
Sine waves have the shape of sine curve.
The X-axis of the sine curve represents the time.
The Y-axis of the sine curve represents the amplitude of the sine wave.
The amplitude of the sine wave at any point in Y is proportional to the sine of a variable.
The sine wave is

given by the equation

A sin(ft)

A -

Amplitude t -

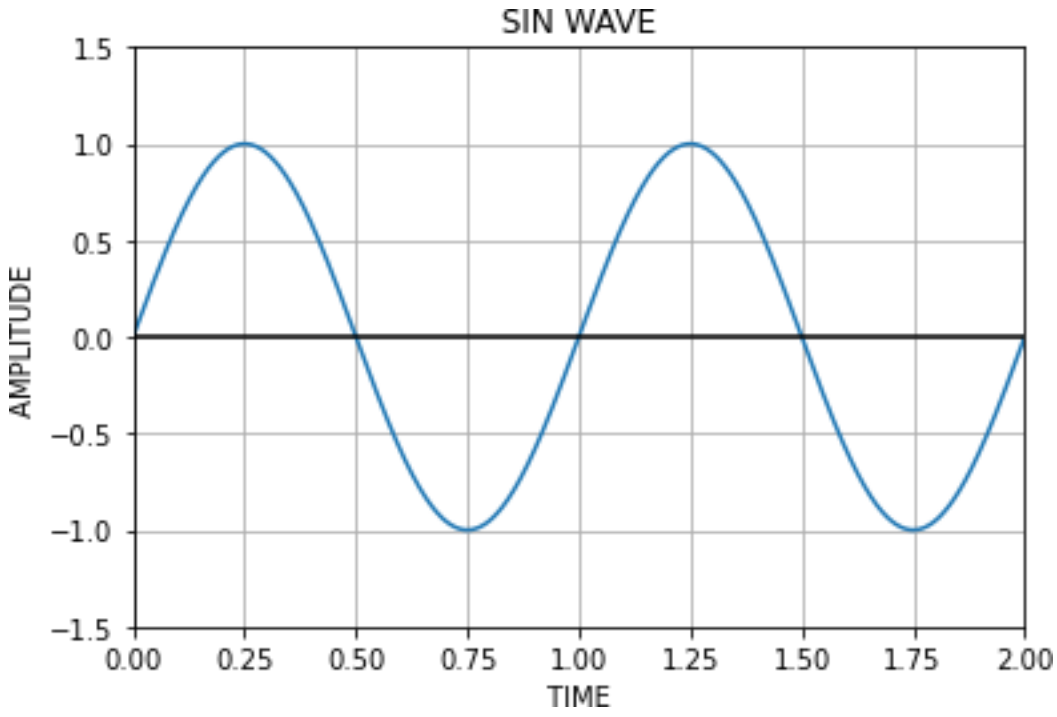
Time f -

Frequency

The sine curve goes through origin.
A cycle of sine wave is complete when the position of the sine wave starts from a position and comes to the same position after attaining its maximum and minimum amplitude during its course.
The time taken to complete one cycle is called the period of the sine wave.
The frequency of the sine wave is given by number of cycles per second.
'A' denotes amplitude of a sine wave.
The distance covered by a cycle measures the wavelength of the sine wave.
The wavelength of the sine wave is denoted by ʎ.
Examples of sine waves include the oscillations produced by the suspended weight on spring and the alternating current.
NumPy has the sin() function, which takes an array of values and provides the sine value for them.
Using the numpy sin() function and the matplotlib plot()a sine wave can be drawn.

```
import matplotlib.pyplot as plot
import numpy as np
time = float (input("Enter Total Time ( in Seconds ) : "))
freq = float (input("Enter Frequency ( in Hz ) : "))

# Sampling rate 1000 hz / second
t = np.linspace(0, time , 1000 , endpoint=True)

# Plot the sin wave signal
plot.plot(t, np.sin(2 * np.pi * freq * t))

# Give a title for the sin wave plot
plot.title(' SIN WAVE ')

# Give x axis label for the sin wave plot
plot.xlabel(' TIME ')

# Give y axis label for the sin wave plot
plot.ylabel(' AMPLITUDE ')

plot.grid(True, which='both')

# Provide x axis and line color
plot.axhline(y=0, color='k')

# Set the max and min values for y axis
plot.ylim(-1.5, 1.5)
plot.xlim( 0, time )

# Display the square wave drawn
plot.show()
```

```
Enter Total Time ( in Seconds ) : 2
Enter Frequency ( in Hz ) : 1
```

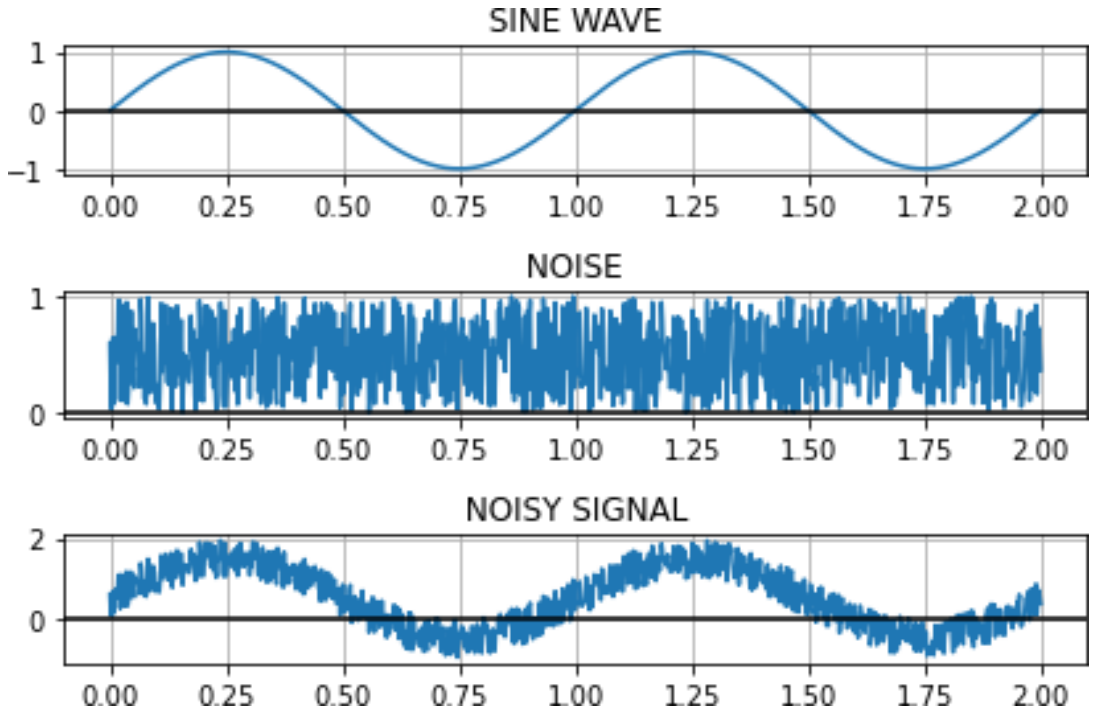


SINE WAVE NOISE

[2]:
```
import matplotlib.pyplot as plt
import numpy as np

time = float (input("Enter Total Time ( in Seconds ) : "))
freq = float (input("Enter Frequency ( in Hz ) : "))

# Sampling rate 1000 hz / second
t = np.linspace(0, time , 1000 , endpoint = True)
y = np.sin(2 * np.pi * freq * t)
noise = np.random.rand(len(y))
corrupt = y + noise


plt.subplot (3,1,1)
plt.title (" SINE WAVE ")
plt.plot (t,y)
plt.grid (which= "Both")
plt.axhline(y=0, color='k')
```

```
plt.subplot (3,1,2)
plt.title (" NOISE ")
plt.plot(t,noise)
plt.grid (which="Both")
plt.axhline(y=0, color='k')

plt.subplot (3,1,3)
plt.title (" NOISY SIGNAL ")
plt.plot (t,corrupt)
plt.grid (which= "Both")
plt.axhline(y=0, color='k')

plt.tight_layout()
plt.show()
```

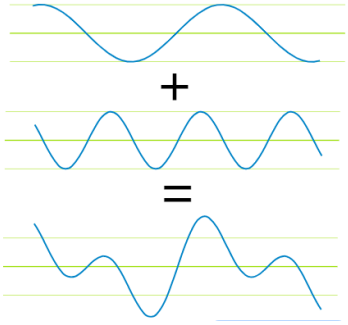
```
Enter Total Time ( in Seconds ) : 2
Enter Frequency ( in Hz ) : 1
```



Construct a square wave using multiple sine waves.

Sine and cosine waves can make other functions!

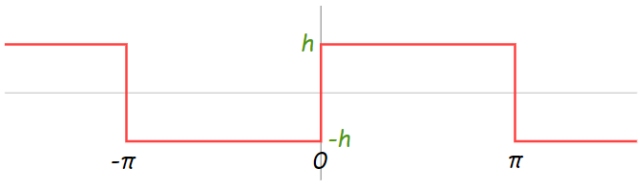
Here two different sine waves add together to make a new wave:



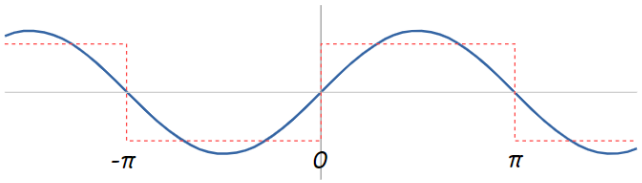
Square Wave

Can we use sine waves to make a **square wave**?

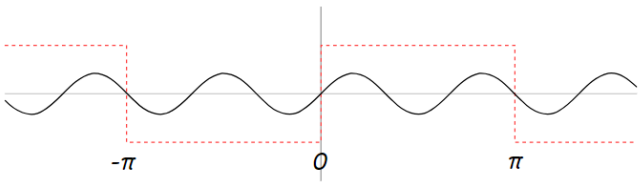
Our target is this square wave:



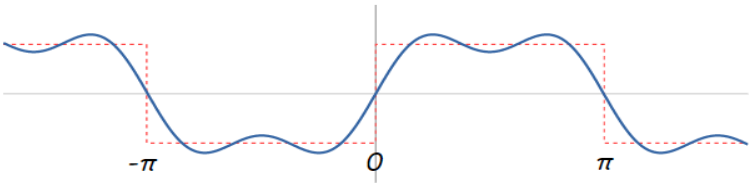
Start with **sin(x)**:



Then take **sin(3x)/3**:

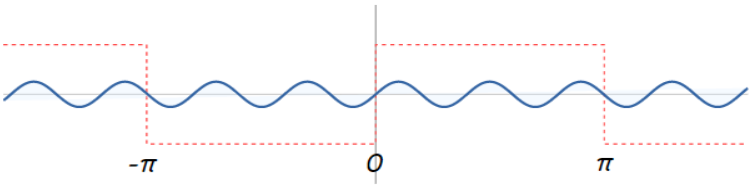


And add it to make **sin(x)+sin(3x)/3**:

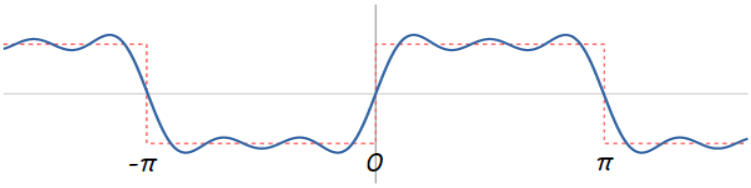


Can you see how it starts to look a little like a square wave?

Now take **sin(5x)/5**:

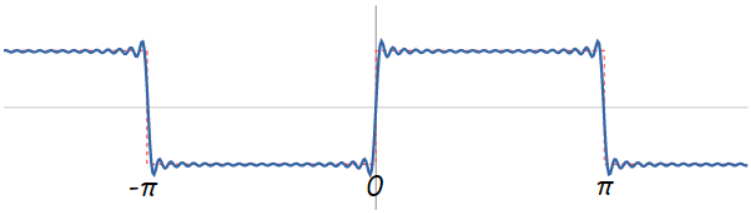


Add it also, to make **sin(x)+sin(3x)/3+sin(5x)/5**:

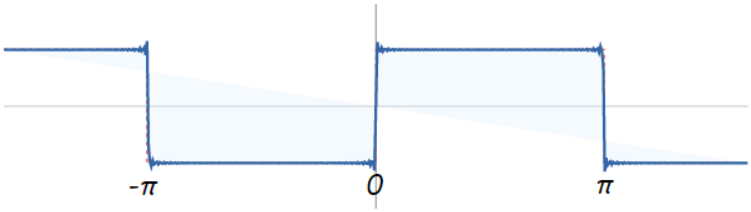


Getting better! Let's add a lot more sine waves.

Using 20 sine waves we get **sin(x)+sin(3x)/3+sin(5x)/5 + ... + sin(39x)/39**:



Using 100 sine waves we get **sin(x)+sin(3x)/3+sin(5x)/5 + ... + sin(199x)/199**:



And if we could add infinite sine waves in that pattern we would **have** a square wave!

So we can say that:

$$\text{a square wave} = \sin(x) + \sin(3x)/3 + \sin(5x)/5 + \ldots \text{ (infinitely)}$$

That is the idea of a Fourier series.

By adding infinite sine (and or cosine) waves we can make other functions, even if they are a bit weird.

Finding the Coefficients

How did we know to use sin(3x)/3, sin(5x)/5, etc?

There are formulas!

First let us write down a full series of sines and cosines, with a name for all coefficients:

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

Where:

- f(x) is the function we want (such as a square wave)
- L is **half of the period** of the function
- a_0, a_n and b_n are **coefficients** that we need to calculate!

To find the coefficients a_0, a_n and b_n we use these formulas:

$$a_0 = \frac{1}{2L} \int_{-L}^{L} f(x)\, dx$$

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

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

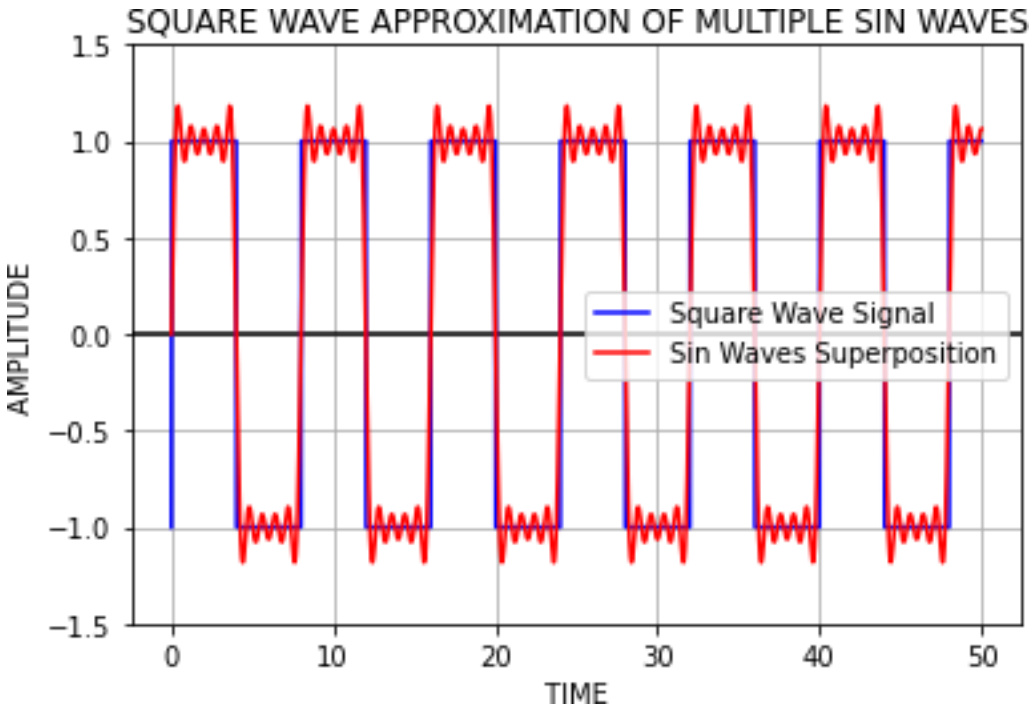
SQUARE WAVE

Construct a square wave using multiple sine waves.

[6]:
```
import numpy as np
import matplotlib.pyplot as plt

time = float (input("Enter Time period ( in Sec ) : "))
# Setup
x_ = np.linspace(0,time,10000)

T = 8
harmonics = 10
# func to form the square wave
def squareWave(x):
    global T
    lowerBoundLeft =(-T/2)
    lowerBoundRight =0
    upperBoundLeft = 0
    upperBoundRight =(T/2)
    one = 1
    negativeOne = -1

    while True:
        if (x >= lowerBoundLeft) and (x <= lowerBoundRight):
            return negativeOne
        elif (x >= upperBoundLeft) and (x <= upperBoundRight):
            return one
        else:
            lowerBoundLeft -= T/2
            lowerBoundRight -=T/2
            upperBoundLeft +=T/2
            upperBoundRight +=T/2
            if one == 1:
                one =-1
                negativeOne = 1
            else:
                one = 1
                negativeOne = -1

# fourier coeff --> bn
def bn(n):
    n = int(n)
    if (n% 2 != 0):
        return 4/(np.pi*n)
    else:
        return 0
```

```
# fourier coeff -->an
def an (n):
    global T
    an = (2*np.pi*n)/T
    return an

# Fourier Series function
def fourierSeries(n_max,x):
    a0 = 0
    partialSums = a0
    for n in range(1,n_max):
        try:
            partialSums = partialSums +bn(n)*np.sin(an(n)*x)
        except:
            print("pass")
            pass
    return partialSums

y =[]
f =[]
for i in x_:
    y.append(squareWave(i))
    f.append(fourierSeries(harmonics,i))

plt.grid( True , which='both')
plt.axhline (y=0 , color='k')
plt.plot(x_,y,color="blue", label="Square Wave Signal" ,)
plt.plot(x_,f,color="red" , label="Sin Waves Superposition")
plt.legend(loc = "upper right")
plt.ylim(-1.5,1.5)
plt.xlabel(" TIME ")
plt.ylabel(" AMPLITUDE ")
plt.title(" SQUARE WAVE APPROXIMATION OF MULTIPLE SIN WAVES ")
plt.legend()
plt.show()
```

```
Enter Time period ( in Sec ) : 50
```


SQUARE WAVE APPROXIMATION OF MULTIPLE SIN WAVES
AMPLITUDE
TIME
Square Wave Signal
Sin Waves Superposition
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
0
10
20
30
40
50

[ ]: