ACOUSTICS ASSIGNMENT 1

Submitted by:
KESAVRAM VS
AI DS - A
23110286

ENGINEERING PHYSICS

08-09-2023

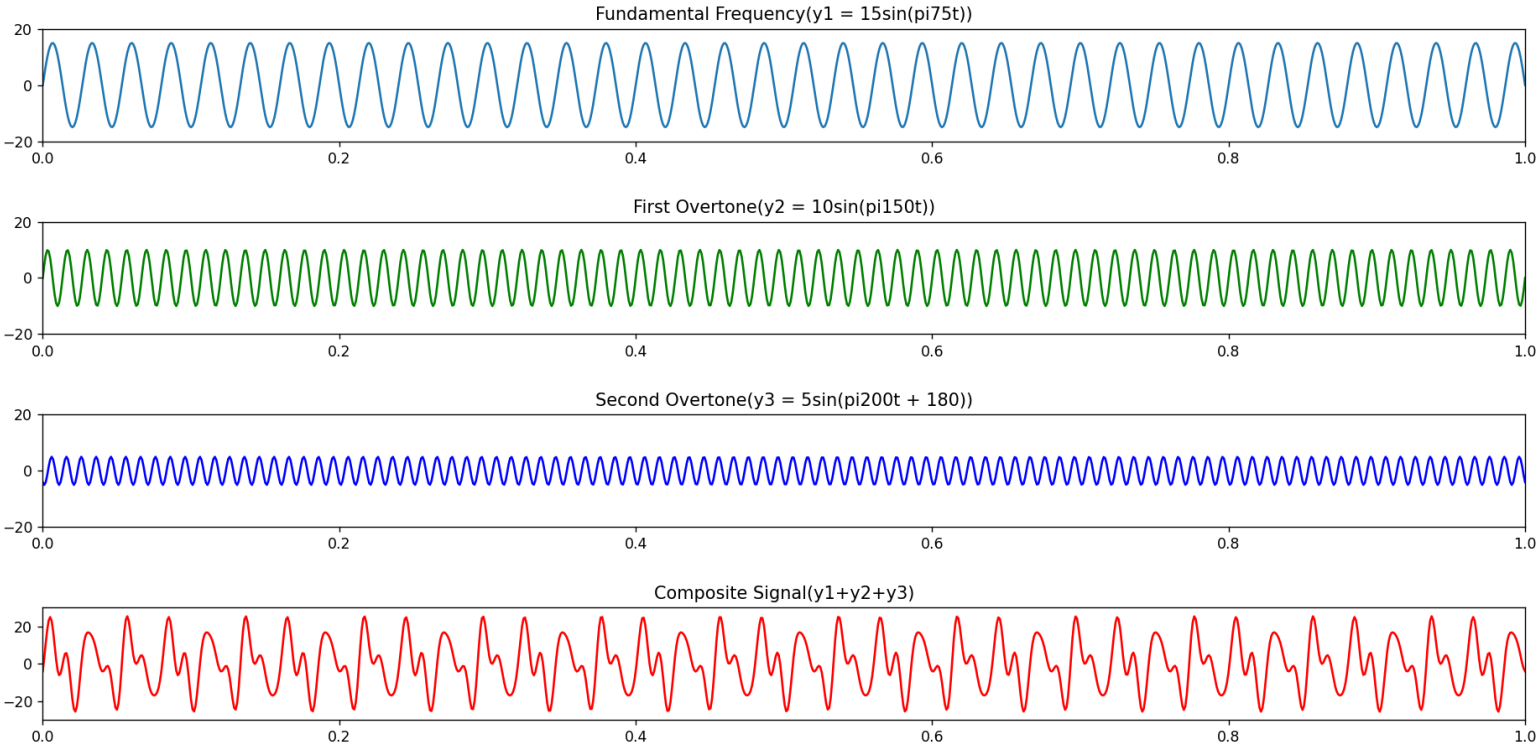
1) Use an excel sheet/python programming and generate numerically 3 sine waves ,ie a fundamental note and at least 2 overtones (see eg in slide 5). Vary the phases and amplitudes and generate a composite time domain signal. Generate plots of the 3 input waves and the generated composite temporal signal. The report should contain the input phase and frequencies of the signals added.

SOURCE CODE:

```
import numpy as np
import matplotlib.pyplot as plot

time = np.linspace(0, 1,1000)

def fundamental_frequency():
    frequency = 75/2
    amplitude = 15
    fundamental_wave = amplitude * np.sin(2 * np.pi * frequency * time)
    return fundamental_wave

def first_overtone():
    frequency = 75
    amplitude = 10
    overtone1_wave = amplitude * np.sin(2 * np.pi * frequency * time)
    return overtone1_wave

def second_overtone():
    frequency = 100
    amplitude = 5
    overtone2_wave = amplitude * np.sin(2 * np.pi * frequency * time + 180)
    return overtone2_wave

composite_signal = fundamental_frequency() + first_overtone() + second_overtone()


plot.figure(figsize=(10, 6))

#FUNDAMENTAL FREQUENCY
plot.subplot(4, 1, 1)
plot.xlim(0,1)#To set min and max value in x axis
plot.ylim(-20,20)#To set min and max value in y axis
plot.plot(time, fundamental_frequency())
plot.title("Fundamental Frequency(y1 = 15sin(pi75t))")

#FIRST OVERTONE
plot.subplot(4, 1, 2)
plot.xlim(0,1)#TO set min and max value in x axis
plot.ylim(-20,20)#To set min and max value in y axis
plot.plot(time, first_overtone() , color = 'g')
plot.title("First Overtone(y2 = 10sin(pi150t))")

#SECOND OVERTONE
plot.subplot(4, 1, 3)
```

```
plot.xlim(0,1)#TO set min and max value in x axis
plot.ylim(-20,20)#To set min and max value in y axis
plot.plot(time, second_overtone(), color = 'b')
plot.title("Second Overtone(y3 = 5sin(pi200t + 180))")

#COMPOSITE SIGNAL
plot.subplot(4, 1, 4)
plot.xlim(0,1)#TO set min and max value in x axis
plot.ylim(-30,30)#To set min and max value in y axis
plot.plot(time, composite_signal, color='r')
plot.title("Composite Signal(y1+y2+y3)")

plot.tight_layout()
plot.show()
```

OUTPUT:



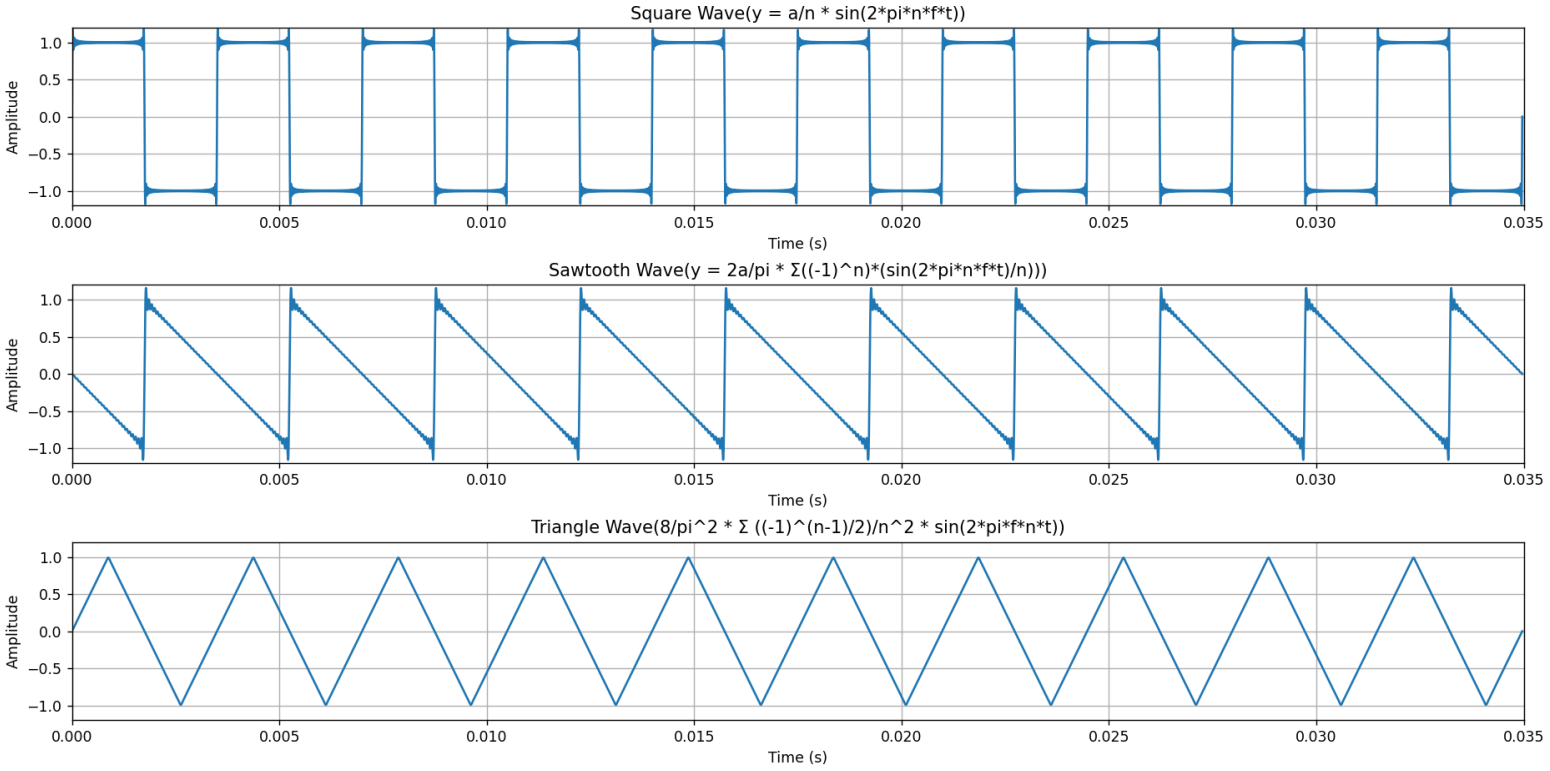
2) Generate a saw tooth wave, a square wave and triangular wave at a frequency equal to the last 4 digits of your roll number.

SOURCE CODE:

```
import numpy as np
import matplotlib.pyplot as plt

f = 286.0  #Setting roll number as frequency
num_cycles = 10  #Number of waves
t = np.linspace(0, num_cycles * (1.0 / f), 10000) #Time array

#Empty array for square
square_wave = np.zeros_like(t)

#Empty array for saw tooth
sawtooth_wave = np.zeros_like(t)

#Empty array for triangle
triangle_wave = np.zeros_like(t, dtype=np.float64)  # Ensure real dtype

#Number of overtones
n_max_square = 100
n_max_sawtooth = 50
n_max_triangle = 50

#SQUARE WAVE
for n in range(1, n_max_square, 2):
    square_wave += (1/n) * np.sin(2 * np.pi * n * f * t)

square_wave *= (4/np.pi)

#SAW TOOTH WAVE
for i in range(1,n_max_sawtooth):
    frequency = i * f
    harmonic = ((-1) ** i) * (np.sin(2 * np.pi * frequency * t) / i)
    sawtooth_wave += harmonic

sawtooth_wave *= (2 / np.pi)

#TRIANGLE WAVE
for n in range(1, n_max_triangle + 1):
    harmonic = (((-1) ** ((n - 1)/2)) / (n ** 2)) * np.sin(2 * np.pi * f * n * t)
    triangle_wave += np.real(harmonic)

triangle_wave *= 8 / (np.pi ** 2)
triangle_wave = np.real(triangle_wave)


plt.figure(figsize=(12, 8))

#PLOTTING SQUARE WAVE
plt.subplot(3, 1, 1)
```

```
plt.plot(t, square_wave)
plt.title("Square Wave(y = a/n * sin(2*pi*n*f*t))")
plt.xlabel("Time (s)")
plt.ylabel("Amplitude")
plt.grid(True)
plt.ylim(-1.2, 1.2)
plt.xlim(0.000,0.035)

#PLOTTING SAW TOOTH WAVE
plt.subplot(3, 1, 2)
plt.plot(t, sawtooth_wave)
plt.title("Sawtooth Wave(y = 2a/pi * Σ((-1)^n)*(sin(2*pi*n*f*t)/n)))")
plt.xlabel("Time (s)")
plt.ylabel("Amplitude")
plt.grid(True)
plt.ylim(-1.2, 1.2)
plt.xlim(0.000,0.035)

#PLOTTING TRIANGLE WAVE
plt.subplot(3, 1, 3)
plt.plot(t, triangle_wave)
plt.title("Triangle Wave(8/pi^2 * Σ ((-1)^(n-1)/2)/n^2 * sin(2*pi*f*n*t))")
plt.xlabel("Time (s)")
plt.ylabel("Amplitude")
plt.grid(True)
plt.ylim(-1.2, 1.2)
plt.xlim(0.000,0.035)

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

OUTPUT:



3) Find the function used to analyze Fourier transform of a time domain signal in MATLAB and Python.

The `fft` function in MATLAB uses a fast Fourier transform algorithm to compute the Fourier transform of a time domain signal.

A fast Fourier transform (FFT) is a highly optimized implementation of the discrete Fourier transform (DFT), which convert discrete signals from the time domain to the frequency domain. FFT computations provide information about the frequency content, phase, and other properties of the signal.

SYNTAX:

```
y = fft(x) #where x is the function.
```

w