Assignment on Quality (based on Lecture 4)

Vishal M Kalathil

December 2, 2021

1 Composite Time Domain Signal

1.1 Code

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

# Initialising sine waves
wave1 = np.arange(0, 10, 0.1)
wave2 = np.arange(0, 20, 0.2)
wave3 = np.arange(0, 30, 0.3)
sin_wave1 = np.sin(wave1)
sin_wave2 = np.sin(wave2)
sin_wave3 = np.sin(wave3)
wave2 /= 2

# Creating composite time domain signal
sin_wave4 = sin_wave1+sin_wave2+sin_wave3

# Plotting waves
plt.plot(sin_wave1)
plt.plot(sin_wave2)
plt.plot(sin_wave3)
plt.plot(sin_wave4)

# Setting title, labels, grid and limits to x axis
plt.title('Composite Time Domain Signal')
plt.ylabel('Amplitude')
plt.xlabel('Time')
plt.xlim(0, 50)
plt.ylim(-2, 3)
plt.grid(True, which='both')

# Draws graph
plt.show()
```

1.2 Graph

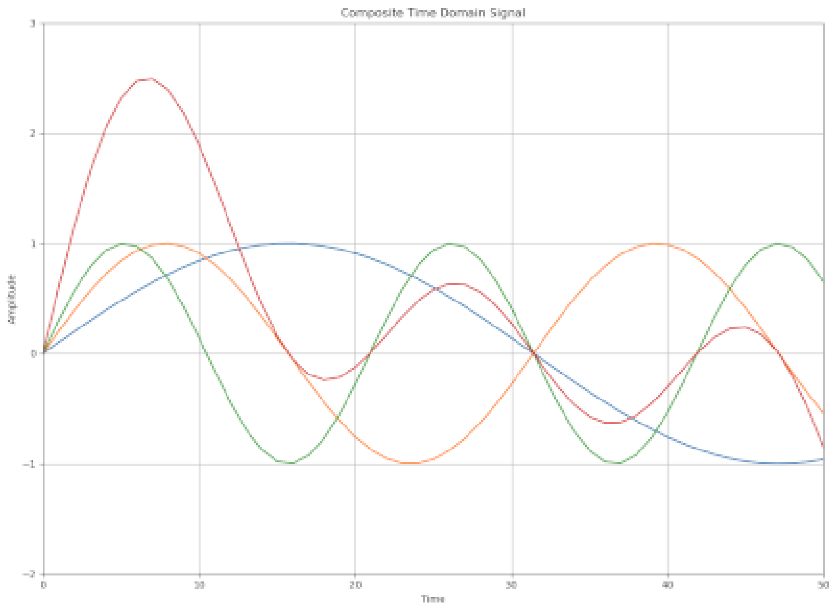


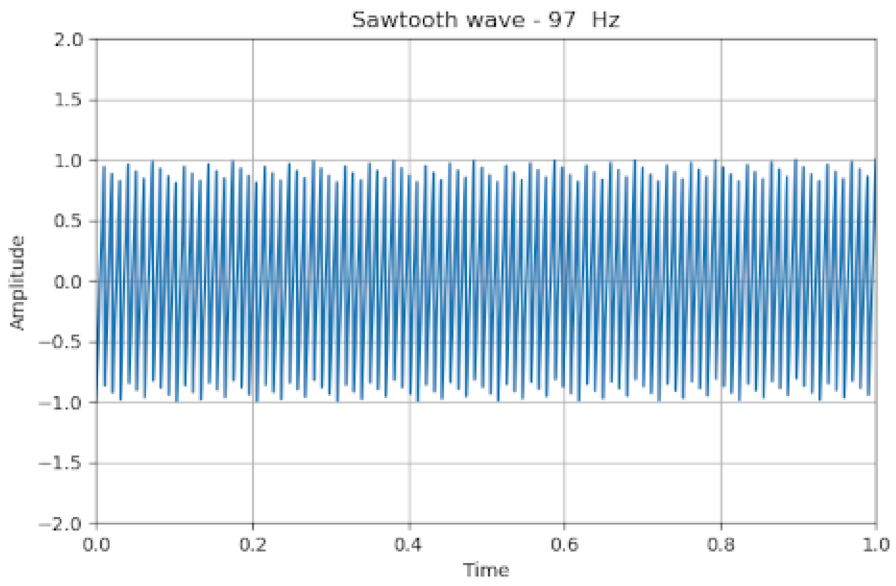
Figure 1: The red line is the composite time domain signal

2 SawTooth

2.1 Code

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

# Initializing wave
timePoints = np.linspace(0, 1, 1000)

# Plots a sawtooth wave of frequency 97
plt.plot(timePoints, signal.sawtooth(2 * np.pi * 97 * timePoints))

# Setting title, labels, grid and limit to axis
plt.title('Sawtooth wave - 97 Hz')
plt.xlabel('Time')
plt.ylabel('Amplitude')
plt.grid(True, which='both')
plt.ylim(-2, 2)
plt.xlim(0, 1.0)

# Draws graph
plt.show()
```

2.2 Graph

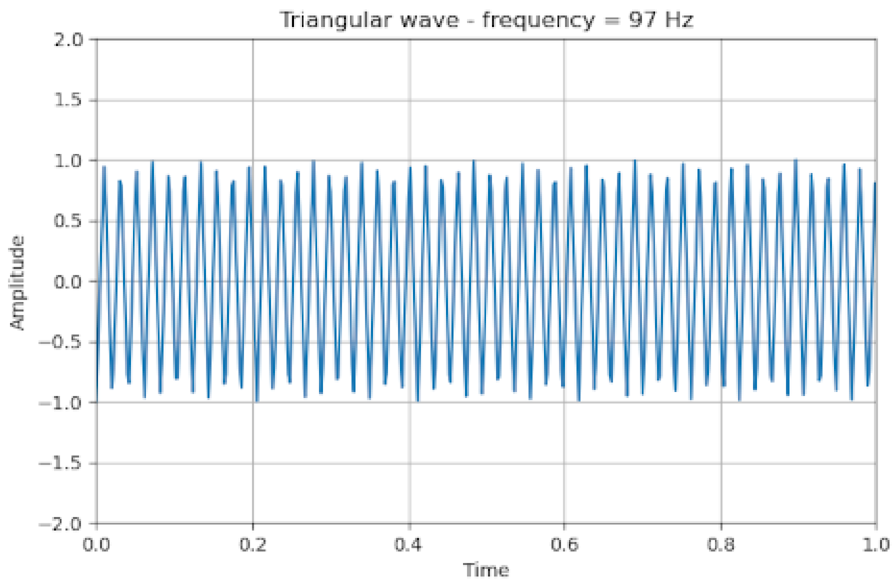


3 Triangle

3.1 Code

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

# Initialise wave
t = np.linspace(0, 1, 1000)

# Plots triangle wave of frequency 97
plt.plot(2 * t, signal.sawtooth(2 * np.pi * 97 * t, 0.5))

# Create title, lables, grid and limit to axes
plt.xlabel("Time")
plt.ylabel("Amplitude")
plt.title("Triangular wave - frequency = 97 Hz")
plt.grid(True, which='both')
plt.ylim(-2, 2)
plt.xlim(0, 1.0)

# Draws graph
plt.show()
```

3.2 Graph

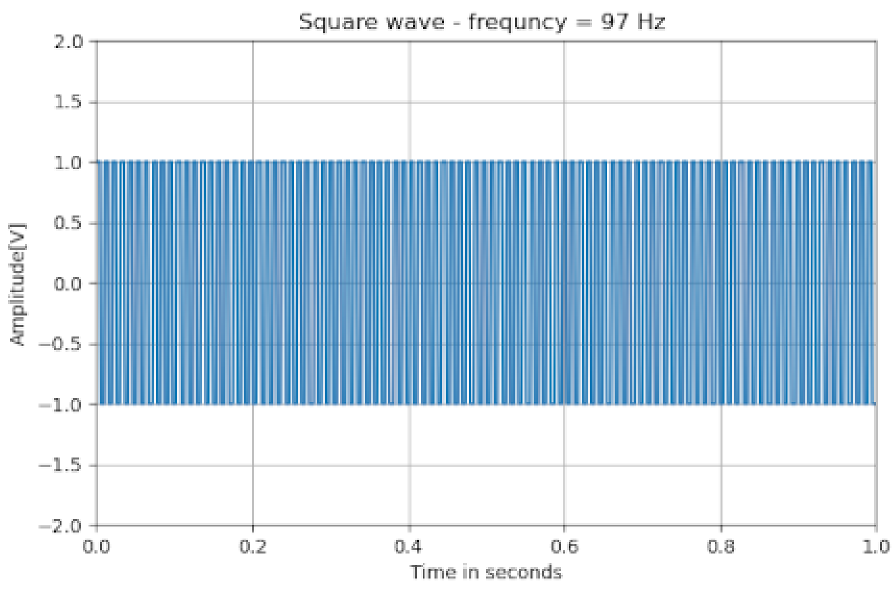


4 Square

4.1 Code

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

# Initializing wave
wave = np.linspace(0, 1, 1000, endpoint=True)

# Plots a wave of frequncy 97
plt.plot(wave, signal.square(2 * np.pi * 97 * wave))

# Creats title,labels, grid and limits to axes
plt.title('Square wave - frequncy = 97 Hz ')
plt.xlabel('Time in seconds')
plt.ylabel('Amplitude[V]')
plt.grid(True, which='both')
plt.xlim(0, 1.0)
plt.ylim(-2, 2)

# Draws graph
plt.show()
```

4.2 Graph



5 Fourier's Transformation

Fourier analysis is a method for expressing a function as a sum of periodic components, and for recovering the signal from those components. When both the function and its Fourier transform are replaced with discretized counterparts, it is called the discrete Fourier transform (DFT).

1. In python we can use scipy fft or numpy fft to analyze the Fourier Transformation
2. In MATLAB we use fft to do the same