**LAB SHEET 1**

**Representation of Basic signals**

1. Try the following Python code to plot sine wave using Matplotlib.

#Importing required library

import numpy as np

import matplotlib.pyplot as plt

# Creating x axis with range and y axis with Sine

# Function for Plotting Sine Graph

x = np.arange(0, 6\*np.pi, 0.1)

y = np.sin(x)

# Plotting Sine Graph

plt.plot(x, y, color='green')

plt.show()

1. Try plotting the following signals in the same way as in Q1 [Hint: You may need SciPy]
2. Cosine wave
3. Unit impulse
4. Unit step wave
5. Square wave
6. Exponential waveform
7. Sawtooth waveform

**Fourier transformation**

1. Let’s generate an audio signal. Explore the following code (Ex. Try varying the sample rate and see what happens).

import numpy as np

from matplotlib import pyplot as plt

SAMPLE\_RATE =

DURATION =

def generate\_sine\_wave(freq, sample\_rate, duration):

x = np.linspace(0, duration, sample\_rate \* duration, endpoint=False)

frequencies = x \* freq

# 2pi because np.sin takes radians

y = np.sin((2 \* np.pi) \* frequencies)

return x, y

# Generate a 2 hertz sine wave that lasts for 5 seconds

x, y = generate\_sine\_wave(2, SAMPLE\_RATE, DURATION)

plt.plot(x, y)

plt.show()

1. Now use generate\_sine\_wave() to generate two signals. How would you change pitch of the signal?

\_, nice\_tone = generate\_sine\_wave(#You may fill this)

\_, noise\_tone = generate\_sine\_wave(#You may fill this) #This will be an unwanted noise signal

noise\_tone = noise\_tone \* 0.3 #Try changing the power of the noise signal

mixed\_tone = nice\_tone + noise\_tone #Mix noise with the original signal

plt.plot(mixed\_tone)

plt.show()

1. If you want, you can normalize and save it using the following code snippet [Optional].

from scipy.io.wavfile import write

#Normalization

normalized\_tone = np.int16((mixed\_tone / mixed\_tone.max()) \* 32767)

write("mysinewave.wav", SAMPLE\_RATE, normalized\_tone)

1. Now, let’s implement the fourier transform. [Explore the functions fft() and fftfreq() in detail]

from scipy.fft import fft, fftfreq

# Number of samples in normalized\_tone

N = SAMPLE\_RATE \* DURATION

# Calculate the Fourier transform

yf = fft(mixed\_tone)

xf = fftfreq(N, 1 / SAMPLE\_RATE)

plt.plot(xf, np.abs(yf))

plt.show()

1. Use the function ifft() to reverse the operation. See how it reproduces the original signal.