1 Introduction

The goal of this lab is to become familiar with fast Fourier transforms using Python.

2 Equations

For this lab, we used the follwing functions:

$$cos(2\pi t) \tag{1}$$

$$5sin(2\pi t) \tag{2}$$

$$2cos((2\pi \cdot 2t) - 2) + sin^2((2\pi \cdot 6t) + 3) \tag{3}$$

3 Methodology and Results

This lab consists of five tasks:

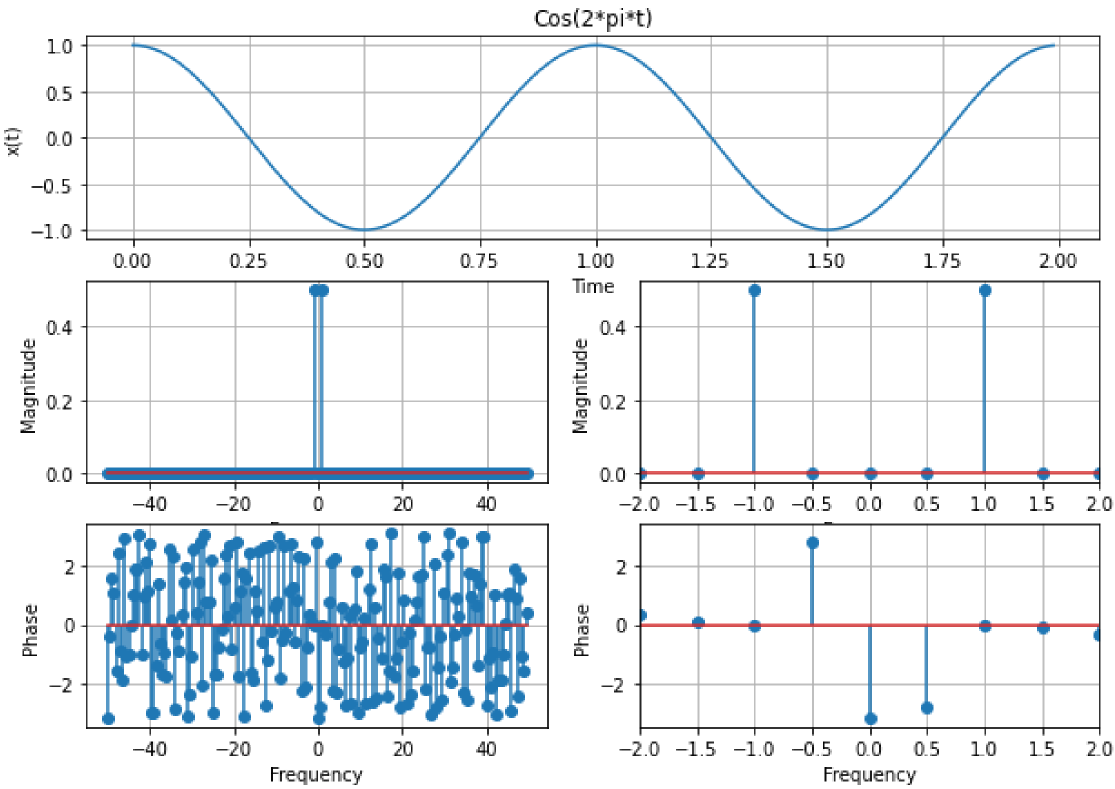
- Task 1: plot cos(2*pi*t) from 0 to 2s. In the same

  gure (separate subplot), use the above code to plot the magnitude of the Fourier transform of this same signal, setting the sampling frequency fs=100. Additionally, plot the phase of the Fourier transform:

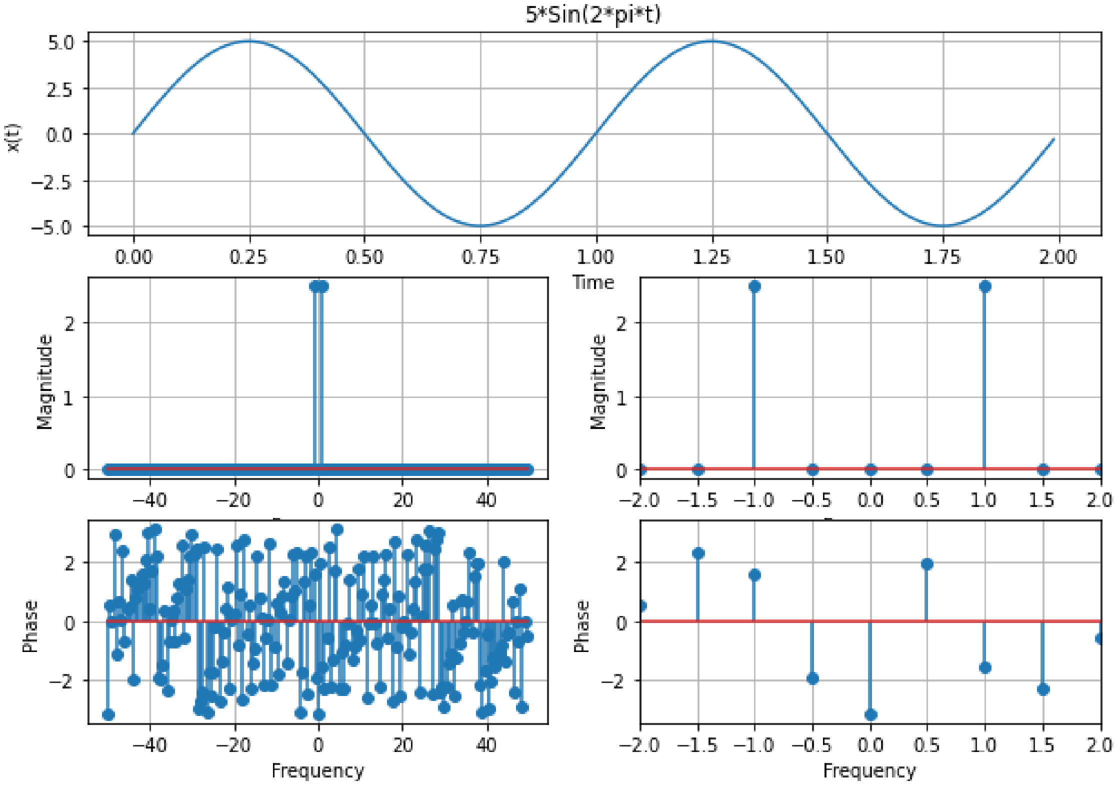
```

    #%% task 1

    fs = 100

    def FFT(x, fs):
    N = len(x) # find the length of the signal
    X_fft = scipy.fftpack.fft(x) # perform the fast Fourier transform (fft)
    X_fft_shifted = scipy.fftpack.fftshift(X_fft) # shift zero frequency
 components
    # to the center of the spectrum
    freq = np.arange(-N/2, N/2)*fs/N # compute the frequencies for the
 output
    # signal , (fs is the sampling frequency and
    # needs to be defined previously
    X_mag = np.abs(X_fft_shifted)/N # compute the magnitudes of the signal
    X_phi = np.angle(X_fft_shifted) # compute the phases of the signal

    return X_mag, X_phi, freq
    # ----- End of user defined function ----- #

    t = np.arange(0, 2, 1/fs)
    y = np.cos(2*np.pi*t)
```

```
    X_mag, X_phi, freq = FFT(y, fs)
    plt.figure(figsize=(10,7))
    plt.subplot("311")
    plt.title("Cos(2*pi*t)")
    plt.plot(t, y)
    plt.xlabel("Time")
    plt.ylabel("x(t)")
    plt.grid()
    plt.subplot("323")
    plt.stem(freq, X_mag, use_line_collection = True)
    plt.xlabel("Frequency")
    plt.ylabel("Magnitude")
    plt.grid()
    plt.subplot("324")
    plt.stem(freq, X_mag, use_line_collection = True)
    plt.xlim([-2, 2])
    plt.xlabel("Frequency")
    plt.ylabel("Magnitude")
    plt.grid()
    plt.subplot("325")
    plt.stem(freq, X_phi, use_line_collection = True)
    plt.xlabel("Frequency")
    plt.ylabel("Phase")
    plt.grid()
    plt.subplot("326")
    plt.stem(freq, X_phi, use_line_collection = True)
    plt.xlim([-2, 2])
    plt.xlabel("Frequency")
    plt.ylabel("Phase")
    plt.grid()
```

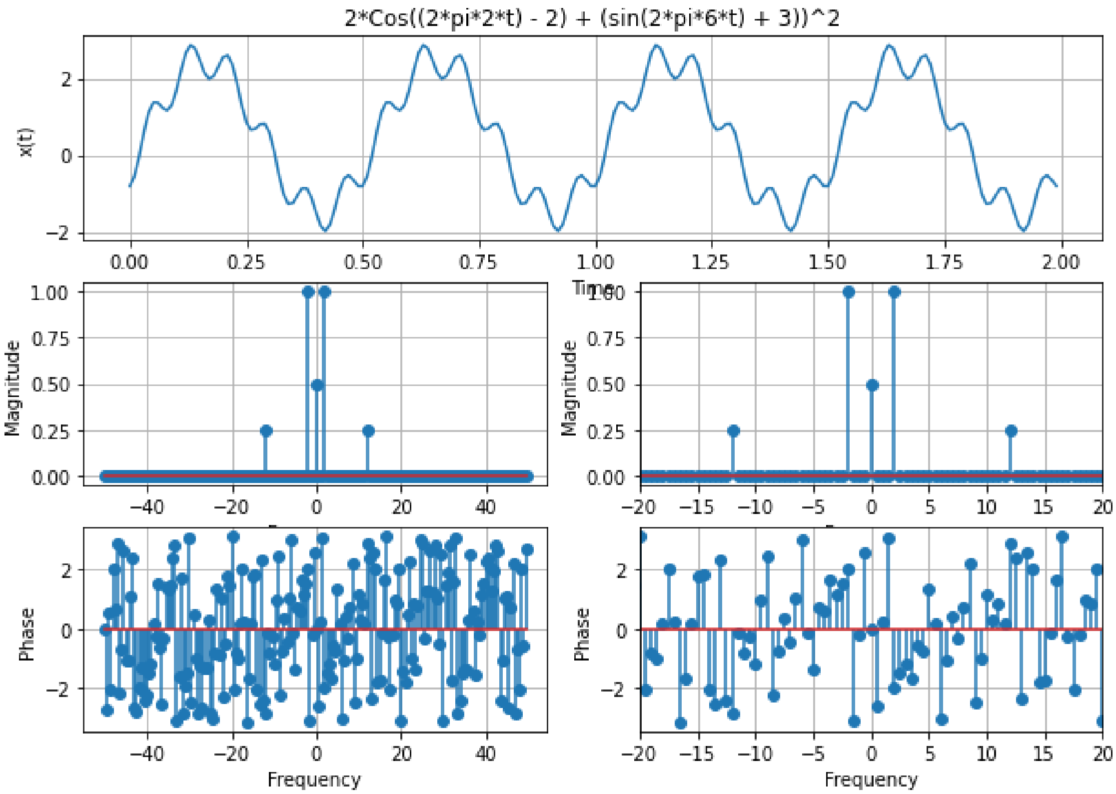
- The output of this code:



- Task 2: repeat Task 1 for the signal 5sin(2*pi*t):



- Task 3: repeat Task 1 for the signal 2cos((2*pi*2t) - 2) + sin2((2*pi*6t) + 3):

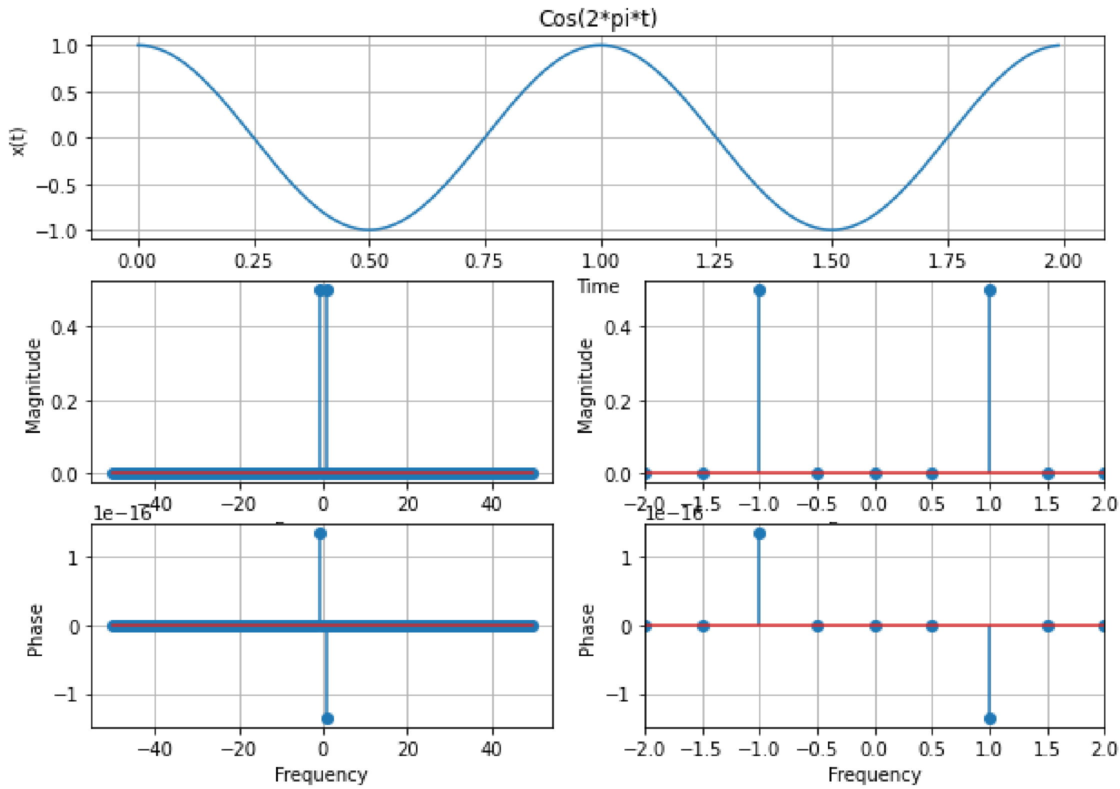


- Task 4: In Task 1, 2, and 3, the plots of the phase should be unreadable. Resolve this by editing your FFT function so, for all elements of X mag ¡ 1e-10, set the corresponding element of X phi = 0. Then, re-run the code for each figure in 1, 2, and 3:

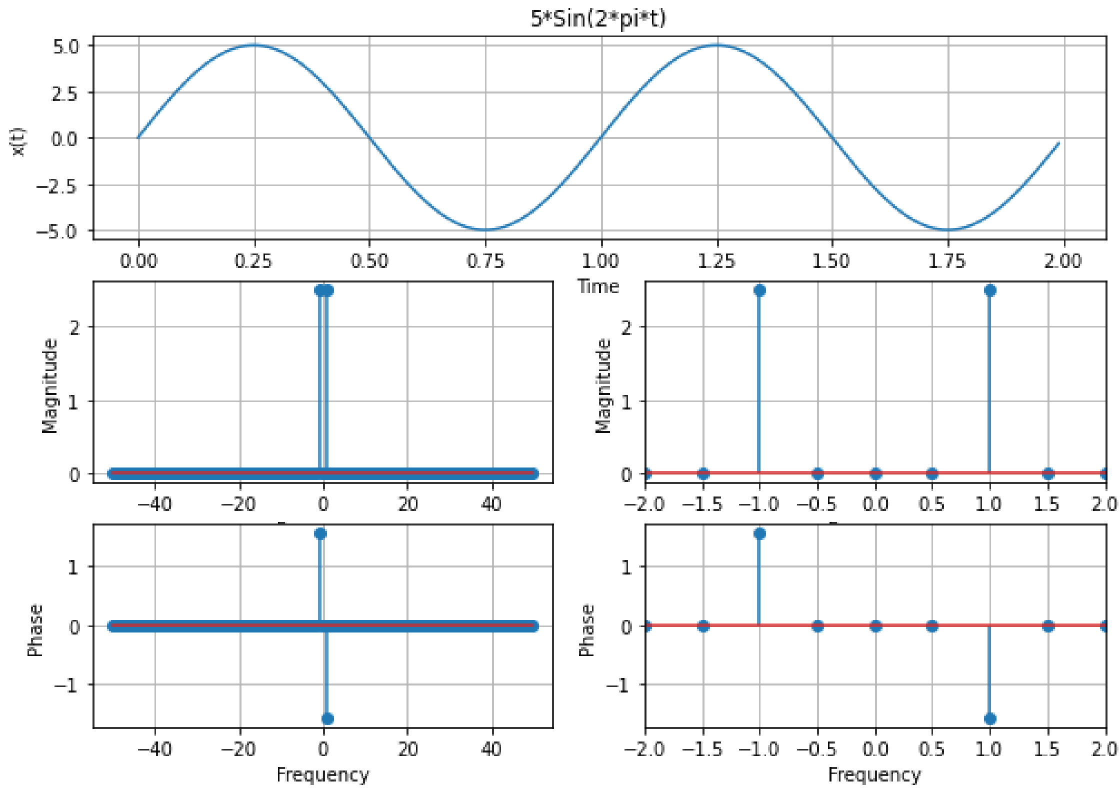
```

    fs = 100

    def FFT(x, fs):
    N = len(x) # find the length of the signal
    X_fft = scipy.fftpack.fft(x) # perform the fast Fourier transform (fft)
    X_fft_shifted = scipy.fftpack.fftshift(X_fft) # shift zero frequency components
    # to the center of the spectrum
    freq = np.arange(-N/2, N/2)*fs/N # compute the frequencies for the output
    # signal , (fs is the sampling frequency and
    # needs to be defined previously
    X_mag = np.abs(X_fft_shifted)/N # compute the magnitudes of the signal
    X_phi = np.angle(X_fft_shifted) # compute the phases of the signal

    num = len(X_mag)
    X_phi_c = deepcopy(X_phi)

    for i in range(0, num):
    if X_mag[i]<1e-10:
```

```
X_phi_c[i] = 0

return X_mag, X_phi, X_phi_c, freq
# ----- End of user defined function ----- #

t = np.arange(0, 2, 1/fs)
y = np.cos(2*np.pi*t)
X_mag, X_phi, X_phi_c, freq = FFT(y, fs)
plt.figure(figsize=(10,7))
plt.subplot("311")
plt.title("Cos(2*pi*t)")
plt.plot(t, y)
plt.xlabel("Time")
plt.ylabel("x(t)")
plt.grid()
plt.subplot("323")
plt.stem(freq, X_mag, use_line_collection = True)
plt.xlabel("Frequency")
plt.ylabel("Magnitude")
plt.grid()
plt.subplot("324")
plt.stem(freq, X_mag, use_line_collection = True)
plt.xlim([-2, 2])
plt.xlabel("Frequency")
plt.ylabel("Magnitude")
plt.grid()
plt.subplot("325")
plt.stem(freq, X_phi_c, use_line_collection = True)
plt.xlabel("Frequency")
plt.ylabel("Phase")
plt.grid()
plt.subplot("326")
plt.stem(freq, X_phi_c, use_line_collection = True)
plt.xlim([-2, 2])
plt.xlabel("Frequency")
plt.ylabel("Phase")
plt.grid()


```

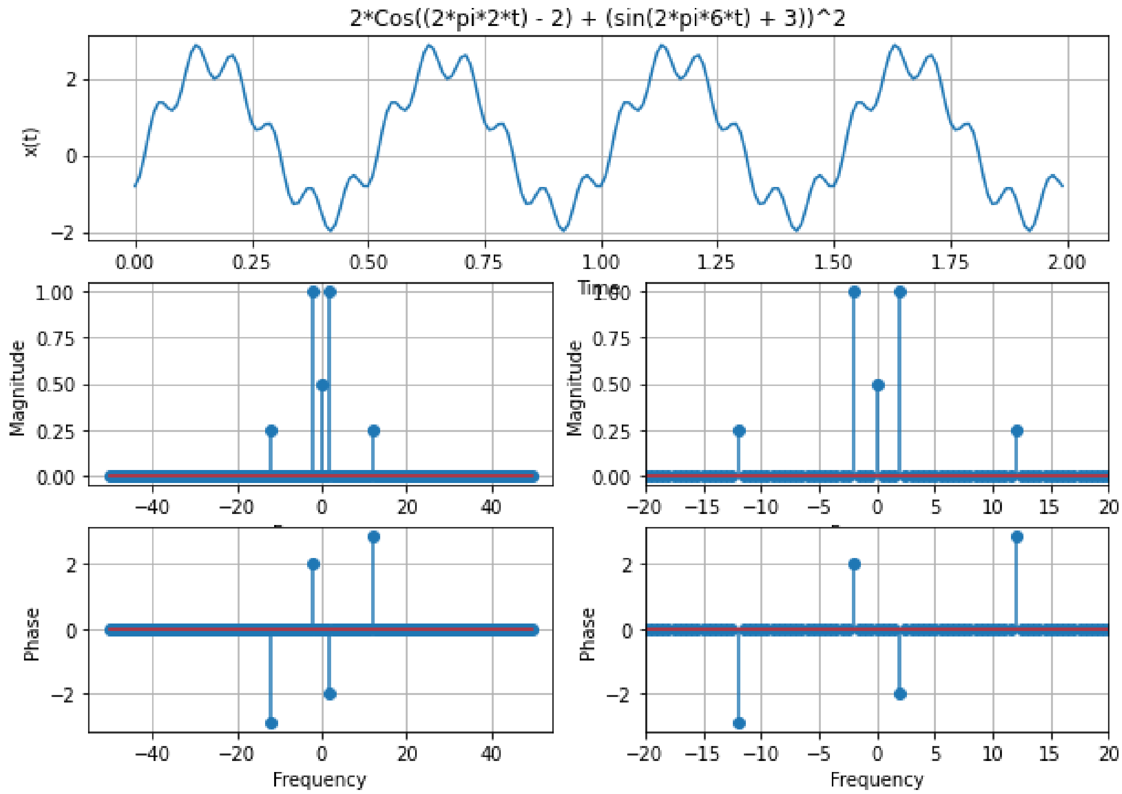
- The result of editing FFT for task 1:



- The result of editing FFT for task 2:



- The result of editing FFT for task 3:

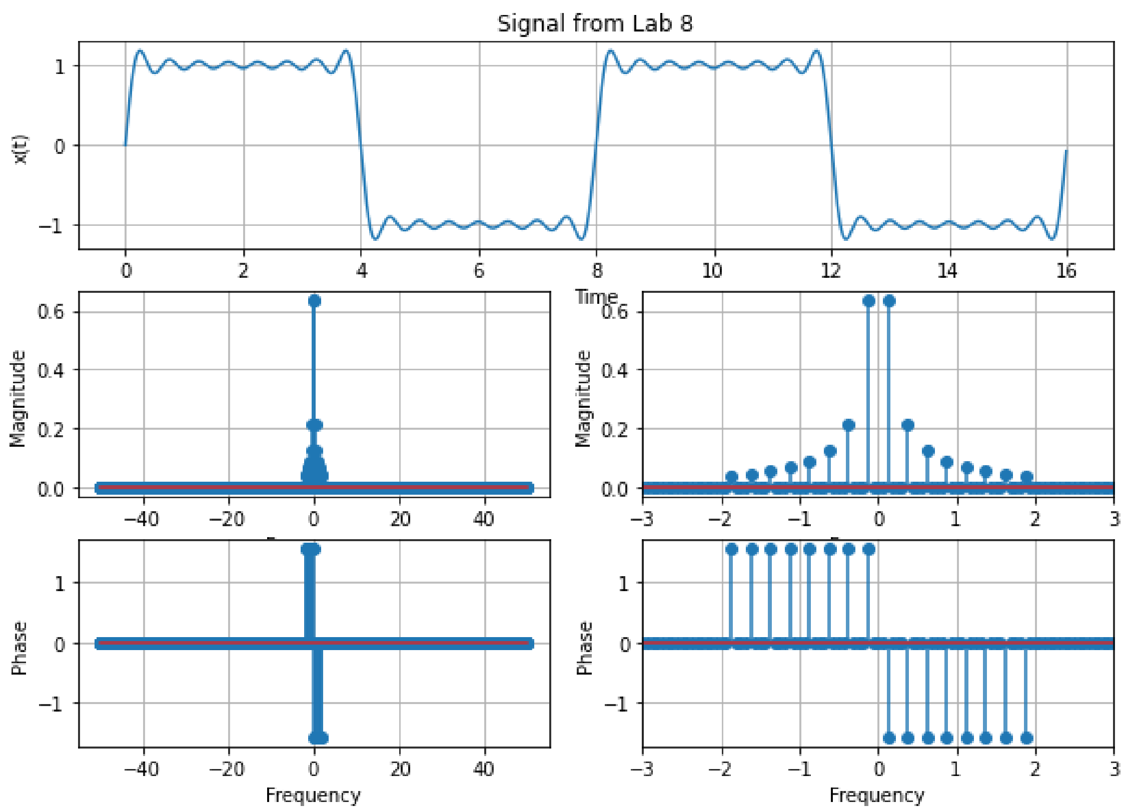


- Task 5: run for the Fourier series approximation of the square wave plotted in Lab 8, using only the N = 15 case through your clean version of the fft developed in Task 4:

```

#%% task 5

T = 8
w = 2*np.pi/T
t = np.arange(0, 16, 1/fs)
N = 15

f = np.zeros(len(t))
for k in range(1, N+1):
a = 2/(k*w*T)*(2*np.sin(k*w*T/2) - np.sin(k*w*T))
b = 2/(k*w*T)*(-2*np.cos(k*w*T/2) + np.cos(k*w*T) + 1)
f = f+b*np.sin(w*k*t)+a*np.cos(w*k*t)

X_mag, X_phi, X_phi_c, freq = FFT(f, fs)
plt.figure(figsize=(10, 7))
plt.subplot("311")
plt.title("Signal from Lab 8")
plt.plot(t, f)
plt.xlabel("Time")
plt.ylabel("x(t)")
plt.grid()
```

```
        plt.subplot("323")
        plt.stem(freq, X_mag, use_line_collection = True)
        plt.xlabel("Frequency")
        plt.ylabel("Magnitude")
        plt.grid()
        plt.subplot("324")
        plt.stem(freq, X_mag, use_line_collection = True)
        plt.xlim([-3, 3])
        plt.xlabel("Frequency")
        plt.ylabel("Magnitude")
        plt.grid()
        plt.subplot("325")
        plt.stem(freq, X_phi_c, use_line_collection = True)
        plt.xlabel("Frequency")
        plt.ylabel("Phase")
        plt.grid()
        plt.subplot("326")
        plt.stem(freq, X_phi_c, use_line_collection = True)
        plt.xlim([-3, 3])
        plt.xlabel("Frequency")
        plt.ylabel("Phase")
        plt.grid()
        plt.show()


```

- The output is :



4 Questions

1. What happens if fs is lower? If it is higher? fs in your report must span a few orders of magnitude.
   If FS is lower the signal will be sampled at very low frequency. Therefore the signal will not be very accurate as it will only have a few points that will be used for FFT analysis. If FS is high the signal will be very accurate and more component frequencies will be detected.

2. What difference does eliminating the small phase magnitudes make?

   By eliminating the small phase magnitudes, the phase will be clear and readable and few samples will remain.

3. Verify your results from Tasks 1 and 2 using the Fourier transforms of cosine and sine. Explain why your results are correct. You will need the transforms in terms of Hz, not rad/s. For example, the Fourier transform of cosine (in Hz) is:

$$F[cos(2\pi f_0 t)] = \frac{1}{2}[\delta(f - f_0) + \delta(f + f_0)]$$

   The frequency for both signals is 1 therefore we will
   For task 1:

$$y = cos(2\pi t)$$
$$F[cos(2\pi t)] = \frac{1}{2}[\delta(f - 1) + \delta(f + 1)]$$

   For task 2:

$$y = 5sin(2\pi t)$$
$$F[5cos(2\pi t)] = \frac{5}{2}(\delta(f - 1) + \delta(f + 1))$$

5 Conclusion

In this lab, we became familiar with fast Fourier transforms using Python.