

# 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 following functions:

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

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

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

## 3 Methodology and Results

This lab consists of five tasks:

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

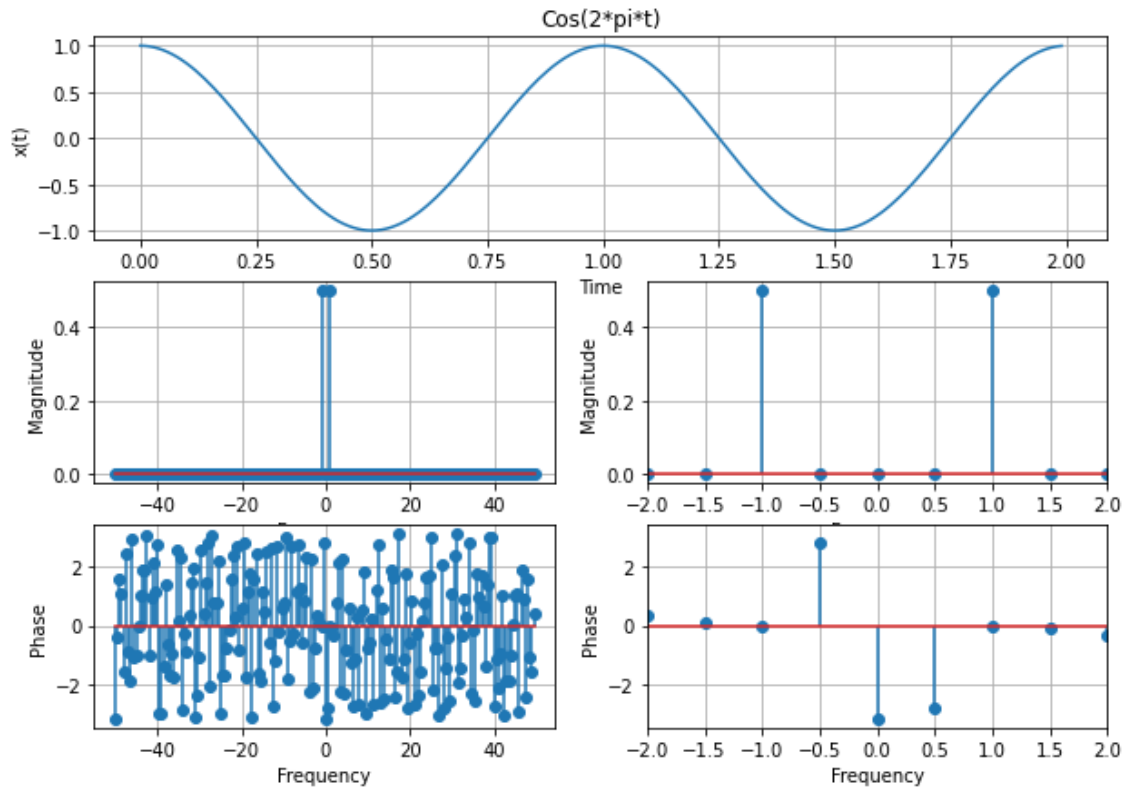
```
1      %% task 1
2
3
4      fs = 100
5
6      def FFT(x, fs):
7          N = len(x) # find the length of the signal
8          X_fft = scipy.fftpack.fft(x) # perform the fast Fourier transform (fft)
9          X_fft_shifted = scipy.fftpack.fftshift(X_fft) # shift zero frequency
10         components
11         # to the center of the spectrum
12         freq = np.arange(-N/2, N/2)*fs/N # compute the frequencies for the
13         output
14         # signal , (fs is the sampling frequency and
15         # needs to be defined previously
16         X_mag = np.abs(X_fft_shifted)/N # compute the magnitudes of the signal
17         X_phi = np.angle(X_fft_shifted) # compute the phases of the signal
18
19         return X_mag, X_phi, freq
20         # ----- End of user defined function ----- #
21
22         t = np.arange(0, 2, 1/fs)
23         y = np.cos(2*np.pi*t)
```

```

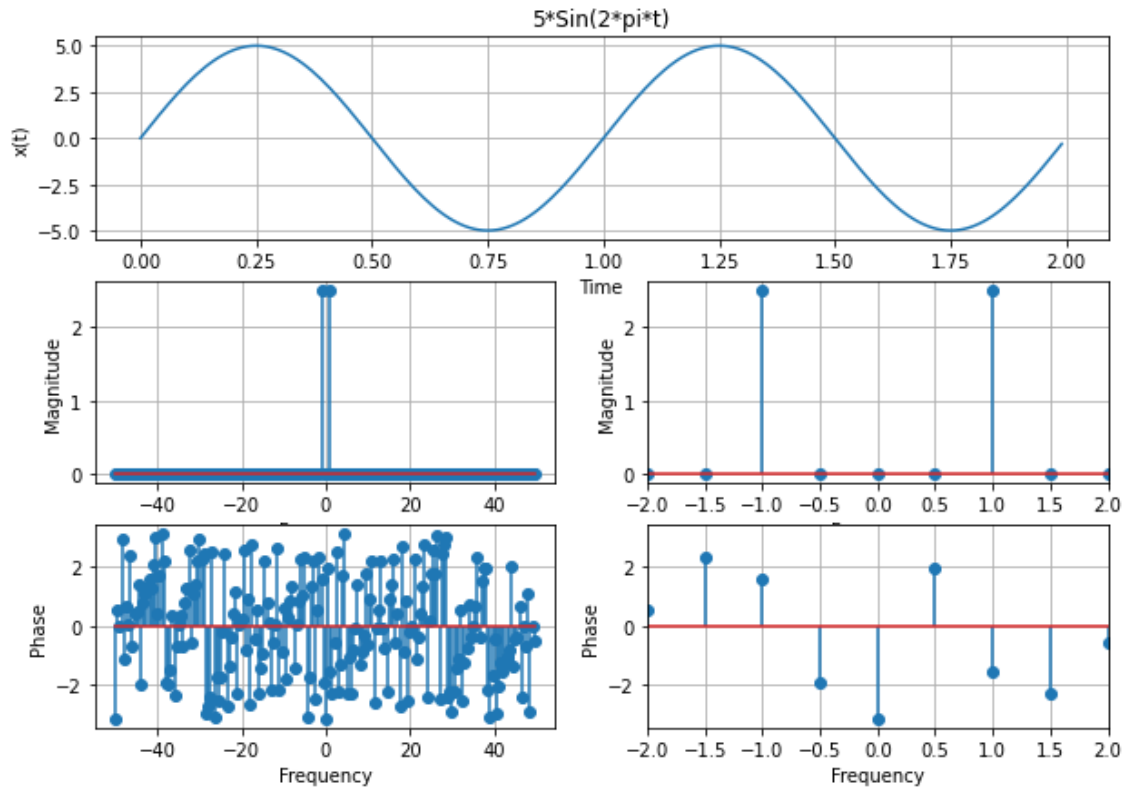
22     X_mag, X_phi, freq = FFT(y, fs)
23     plt.figure(figsize=(10,7))
24     plt.subplot("311")
25     plt.title("Cos(2*pi*t)")
26     plt.plot(t, y)
27     plt.xlabel("Time")
28     plt.ylabel("x(t)")
29     plt.grid()
30     plt.subplot("323")
31     plt.stem(freq, X_mag, use_line_collection = True)
32     plt.xlabel("Frequency")
33     plt.ylabel("Magnititude")
34     plt.grid()
35     plt.subplot("324")
36     plt.stem(freq, X_mag, use_line_collection = True)
37     plt.xlim([-2, 2])
38     plt.xlabel("Frequency")
39     plt.ylabel("Magnititude")
40     plt.grid()
41     plt.subplot("325")
42     plt.stem(freq, X_phi, use_line_collection = True)
43     plt.xlabel("Frequency")
44     plt.ylabel("Phase")
45     plt.grid()
46     plt.subplot("326")
47     plt.stem(freq, X_phi, use_line_collection = True)
48     plt.xlim([-2, 2])
49     plt.xlabel("Frequency")
50     plt.ylabel("Phase")
51     plt.grid()
52
53

```

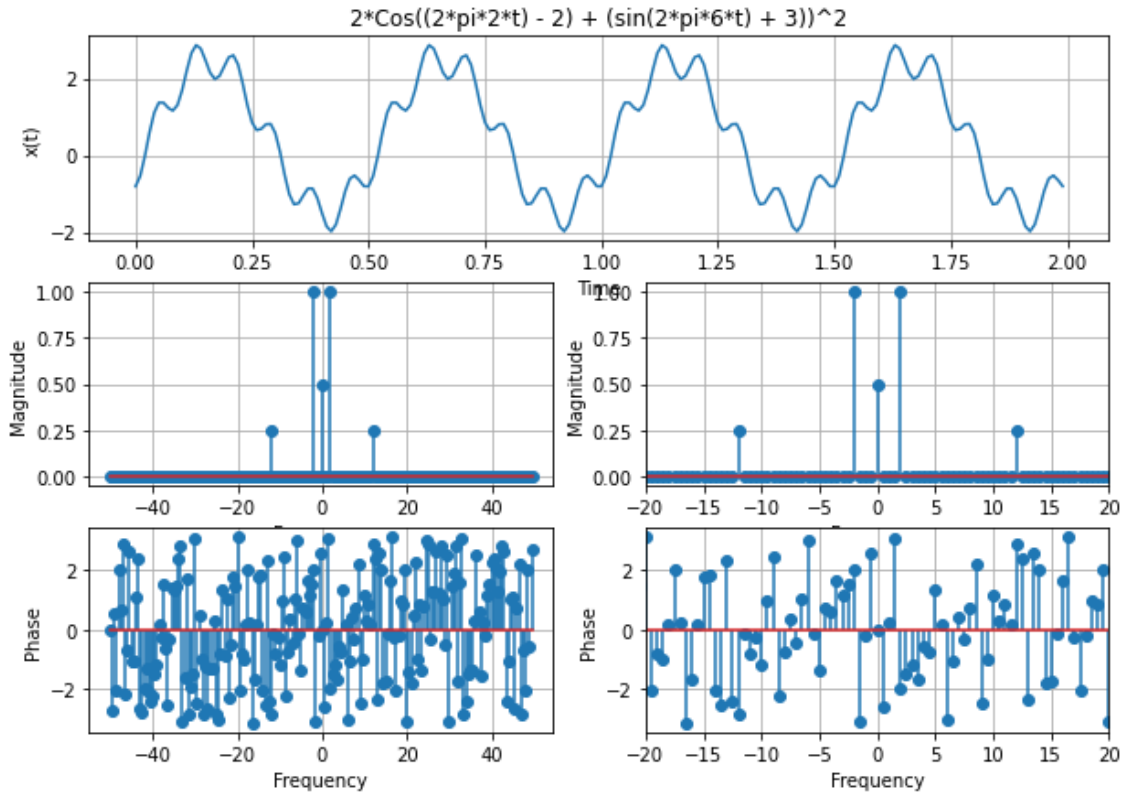
- The output of this code:



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



- Task 3: repeat Task 1 for the signal  $2\cos((2\pi \cdot 2t) - 2) + \sin(2\pi \cdot 6t + 3)$ :



- 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:

```

1
2     fs = 100
3
4     def FFT(x, fs):
5         N = len(x) # find the length of the signal
6         X_fft = scipy.fftpack.fft(x) # perform the fast Fourier transform (fft)
7         X_fft_shifted = scipy.fftpack.fftshift(X_fft) # shift zero frequency
8         # to the center of the spectrum
9         freq = np.arange(-N/2, N/2)*fs/N # compute the frequencies for the
10        output
11        # signal , (fs is the sampling frequency and
12        # needs to be defined previously
13        X_mag = np.abs(X_fft_shifted)/N # compute the magnitudes of the signal
14        X_phi = np.angle(X_fft_shifted) # compute the phases of the signal
15
16        num = len(X_mag)
17        X_phi_c = deepcopy(X_phi)
18
19        for i in range(0, num):
20            if X_mag[i]<1e-10:

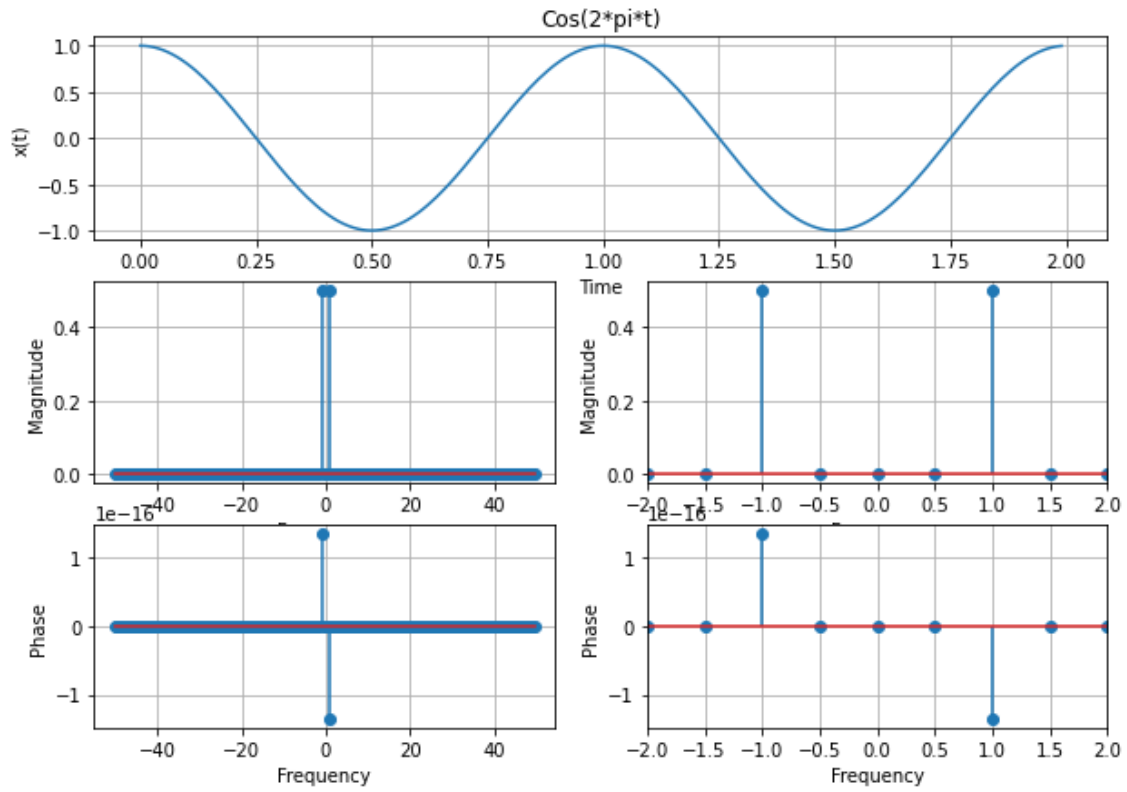
```

```

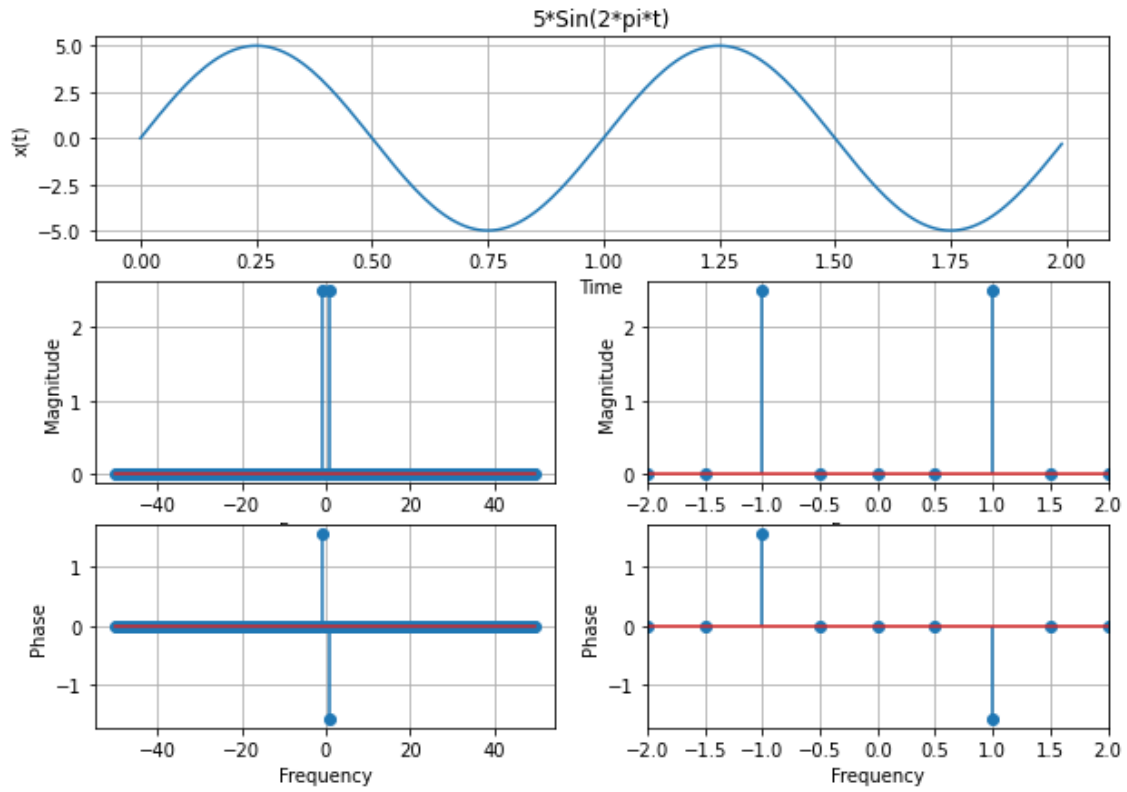
20     X_phi_c[i] = 0
21
22     return X_mag, X_phi, X_phi_c, freq
23     # ----- End of user defined function ----- #
24
25     t = np.arange(0, 2, 1/fs)
26     y = np.cos(2*np.pi*t)
27     X_mag, X_phi, X_phi_c, freq = FFT(y, fs)
28     plt.figure(figsize=(10,7))
29     plt.subplot("311")
30     plt.title("Cos(2*pi*t)")
31     plt.plot(t, y)
32     plt.xlabel("Time")
33     plt.ylabel("x(t)")
34     plt.grid()
35     plt.subplot("323")
36     plt.stem(freq, X_mag, use_line_collection = True)
37     plt.xlabel("Frequency")
38     plt.ylabel("Magnitude")
39     plt.grid()
40     plt.subplot("324")
41     plt.stem(freq, X_mag, use_line_collection = True)
42     plt.xlim([-2, 2])
43     plt.xlabel("Frequency")
44     plt.ylabel("Magnitude")
45     plt.grid()
46     plt.subplot("325")
47     plt.stem(freq, X_phi_c, use_line_collection = True)
48     plt.xlabel("Frequency")
49     plt.ylabel("Phase")
50     plt.grid()
51     plt.subplot("326")
52     plt.stem(freq, X_phi_c, use_line_collection = True)
53     plt.xlim([-2, 2])
54     plt.xlabel("Frequency")
55     plt.ylabel("Phase")
56     plt.grid()
57
58

```

- The result of editing FFT for task 1:

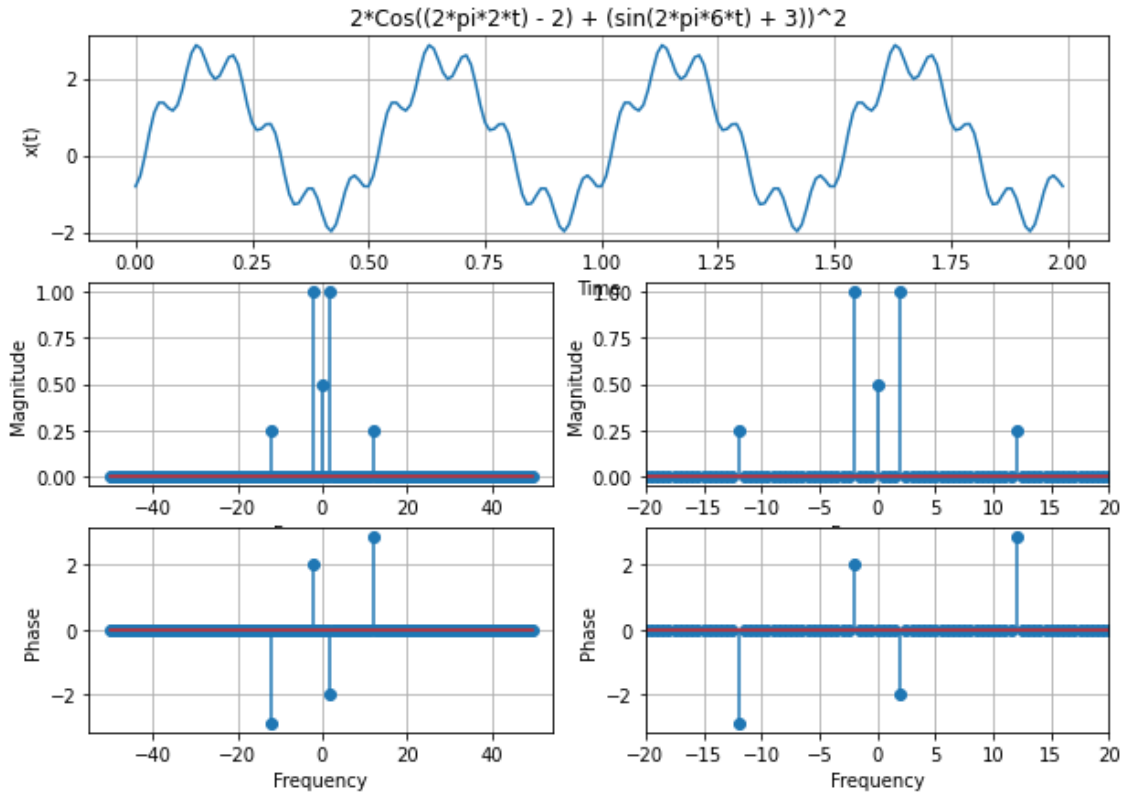


- The result of editing FFT for task 2:



- 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:

```

1
2     #%% task 5
3
4     T = 8
5     w = 2*np.pi/T
6     t = np.arange(0, 16, 1/fs)
7     N = 15
8
9     f = np.zeros(len(t))
10    for k in range(1, N+1):
11        a = 2/(k*w*T)*(2*np.sin(k*w*T/2) - np.sin(k*w*T))
12        b = 2/(k*w*T)*(-2*np.cos(k*w*T/2) + np.cos(k*w*T) + 1)
13        f = f+b*np.sin(w*k*t)+a*np.cos(w*k*t)
14
15    X_mag, X_phi, X_phi_c, freq = FFT(f, fs)
16    plt.figure(figsize=(10, 7))
17    plt.subplot("311")
18    plt.title("Signal from Lab 8")
19    plt.plot(t, f)
20    plt.xlabel("Time")
21    plt.ylabel("x(t)")
22    plt.grid()

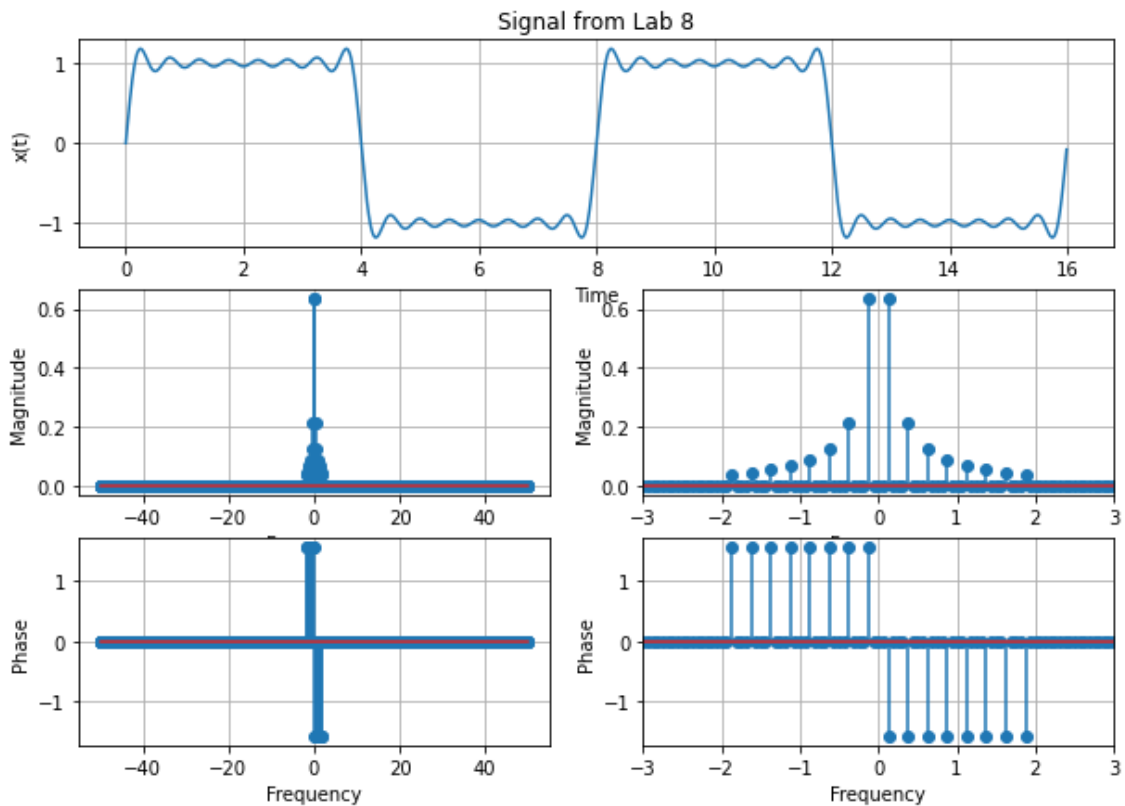
```

```

23 plt.subplot("323")
24 plt.stem(freq, X_mag, use_line_collection = True)
25 plt.xlabel("Frequency")
26 plt.ylabel("Magnitude")
27 plt.grid()
28 plt.subplot("324")
29 plt.stem(freq, X_mag, use_line_collection = True)
30 plt.xlim([-3, 3])
31 plt.xlabel("Frequency")
32 plt.ylabel("Magnitude")
33 plt.grid()
34 plt.subplot("325")
35 plt.stem(freq, X_phi_c, use_line_collection = True)
36 plt.xlabel("Frequency")
37 plt.ylabel("Phase")
38 plt.grid()
39 plt.subplot("326")
40 plt.stem(freq, X_phi_c, use_line_collection = True)
41 plt.xlim([-3, 3])
42 plt.xlabel("Frequency")
43 plt.ylabel("Phase")
44 plt.grid()
45 plt.show()
46
47

```

- 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:

$$y = 5\sin(2\pi t)$$

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

## 5 Conclusion

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