ECE 351

Signals and Systems

Lab 9

Fast Fourier Transform

Submitted By :
Zachary Pfaff
https://github.com/ZachPfaff

Contents

Fast Fourier Transform

Zachary Pfaff

March 29th 2022

1 Introduction

Previously in ECE 351, we have created our own Fourier transform functions of a square wave. The following lab will expand on this by introducing the Fast Fourier Transform (FFT) which is a computer generated algorithm that finds the Fourier Transform of a series of signals. The goal of lab 9 is to create our own FFT routine to be used for various signals.

2 Methodology

To start creating our own Fast Fourier Transform routine, we were given a block of code to work from. This block of code is shown in listing 1. This block is responsible for calculating the frequency, magnitude, and phase of a given signal when placed in a user defined function. The block of code is not complete, however it will serve as a place to start. Note that the frequency variable, fs, is already declared previously in code to be 100.

```
def fft1(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 the zero
   frequency components
                                                  #to the center of
    the spectrum
    freq = np.arange(-N/2, N/2)*fs/N #fs is sampling frequency and
   needs to be defined
                                     #previously in code
    X_mag = np.abs(X_fft_shifted)/N #compute magnitudes of signal
    X_phi = np.angle(X_fft_shifted) #compute phases of signal
    return freq, X_mag, X_phi
```

Listing 1: First FFT Function Code

The first task of lab 9 is to plot the magnitude and phase of the Fourier transform of the following signals:

$$cos(2\pi t)$$

$$5sin(s\pi t)$$

$$2cos((2\pi * 2t) - 2) + sin^2((2\pi * 6t) + 3)$$

To approach this, each signal is implemented into python code as a function that will be used as the "x" parameter in the fft1 function. The previously defined frequency variable, fs, is also placed in the second fft1 parameter. Code for defining the signals

and calculating the magnitude and phase of the Fourier transform are shown below in listing 2.

```
x1_1 = np.cos(2*np.pi*t)
x1_2 = 5*np.sin(2*np.pi*t)
x1_3 = 2*np.cos((2*np.pi*2*t)-2) + np.sin((2*np.pi*6*t)+3)**2

f1_1 = fft1(x1_1, fs)
f1_2 = fft1(x1_2, fs)
f1_3 = fft1(x1_3, fs)
```

Listing 2: First Signal Code

To observe the magnitude and phase of the fourier transform of each signal, a series of plots are made to showcase the signal, magnitude, and phase of the fourier transforms being plotted. Additionally, the x-axis of the magnitude and phase are limited on separate plots to show values of the stem plot in a smaller range. Listing 3 below shows how the plots for the first cosine signal was implemented into python. Note that the code for the other signal plots are almost exactly the same, so they will not be included into the report.

```
#Task 1 Plots
plt.rcParams.update({'font.size': 14})
plt.figure(figsize = (15 ,15))
plt.subplot(3, 1, 1)
plt.plot(t, x1_1)
plt.title("Sig 1")
plt.ylabel("y")
plt.grid()

plt.subplot(3, 2, 3)
plt.stem(f1_1[0], f1_1[1])
plt.ylabel("Magnitude")
plt.grid()

plt.subplot(3, 2, 4)
plt.xlim([-5, 5])
plt.stem(f1_1[0], f1_1[1])
plt.ylabel("Magnitude")
plt.grid()

plt.subplot(3, 2, 5)
plt.stem(f1_1[0], f1_1[2])
plt.ylabel("Phase")
plt.grid()

plt.subplot(3, 2, 6)
plt.xlim([-5, 5])
plt.stem(f1_1[0], f1_1[2])
```

```
plt.ylabel("Phase")
plt.grid()
```

Listing 3: Plot code to be referenced for each signal

Naturally, when each signal is plotted, the phase stem plots will show a lot of noise. To reduce this, we are tasked to modify the original fft user defined function to make the element $Xphi = 0$ when the element $Xmag < 1e - 10$. To do this, I added a 'for' loop to the end of the user defined function that sets each Xphi value into an array. The magnitude Xmag is then checked to see if it is less than 1e-10 for each corresponding Xphi value. If it is, the value in the Xphi array is set to zero, limiting many of the values shown on the phase graphs. The code for the new fft user defined function is shown below in listing 4.

```
def fft2(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 the zero
   frequency components
                                                  #to the center of
    the spectrum
    freq = np.arange(-N/2, N/2)*fs/N #fs is sampling frequency and
   needs to be defined
                                        #previously in code
    X_mag = np.abs(X_fft_shifted)/N #compute magnitudes of signal
    X_phi = np.angle(X_fft_shifted) #compute phases of signal
    for i in range(len(X_phi)):
        if X_mag[i] < 1e-10:
            X_phi[i] = 0
    return freq, X_mag, X_phi
```

Listing 4: New FFT user defined function code

The last task of this lab is to bring in the Fourier series approximation of the square wave produced in lab 8. The Fourier series is approximated at the N=15 case and is plotted in the time domain $0 < t < 16s$. This code was straightforward to implement since most of it was a direct copy of my previous lab. The code below in listing 5 shows the function definition of the previously created Fourier series, as well as the functions magnitude and phase being calculated from the new FFT user defined function.

```
T = 8
w = (2*np.pi)/T
t2 = np.arange(0, 16, Ts)
N = 15

```

```
def fourier(t2, n):
    y = 0
    for i in np.arange(1, n + 1):
        y = y + (2/(i * np.pi)) * (1 - np.cos(i * np.pi)) * (np.sin
   (i * w * t2))
    return y

x2 = fourier(t2, N)

f3 = fft2(x2, fs)
```

Listing 5: Task 5 code

3 Data

The first sets of graphs below show the magnitudes and phases of the Fourier transforms of the three specified signals. These first three sets of graphs demonstrates that the phase has a lot of noise which is due to the original fft user defined function being used. Figures 1 through 3 below shows the first three sets of plots. The next set of figures demonstrates the same signals, except this time they are put into the new FFT function which reduces the noise in the phase. These figures are shown as figures 4 through 6 below. The final figure, figure 7, shows the Fourier series evaluated from lab 8. This signal is placed into the new FFT function to determine its magnitude and phase of the Fourier transform.

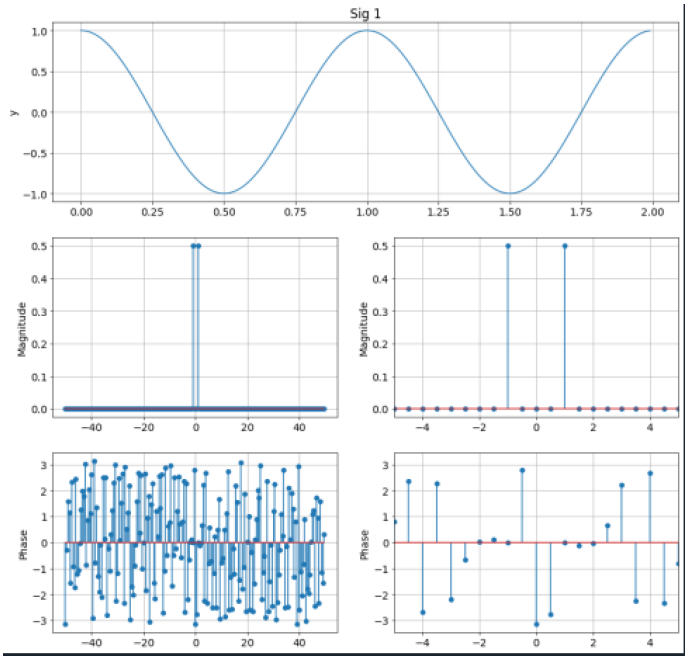

Figure 1: Old FFT, First Signal

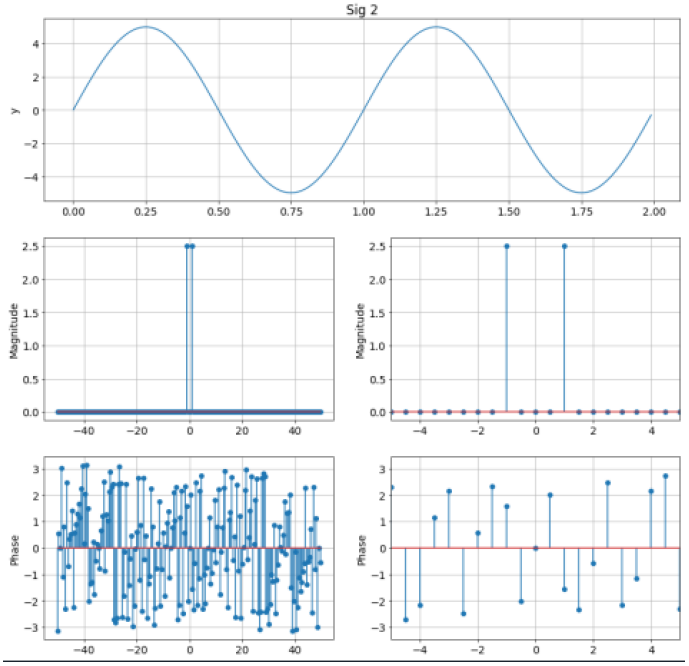

Figure 2: Old FFT, Second Signal

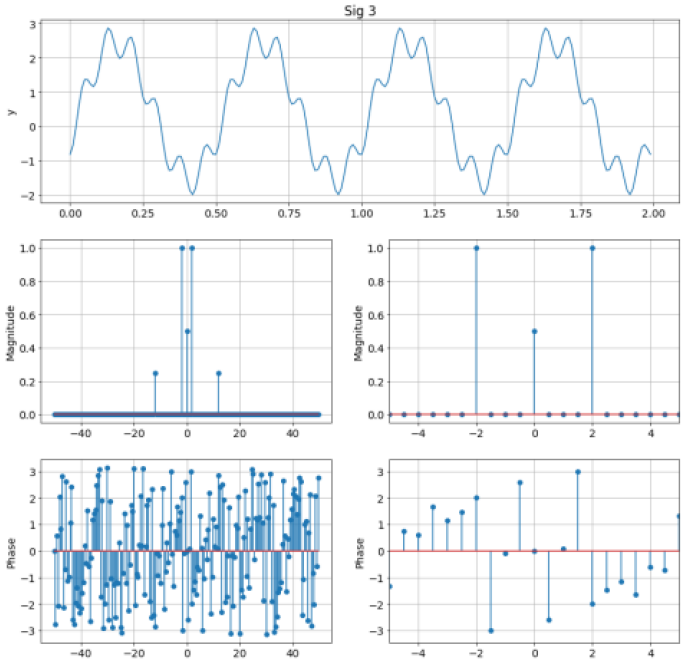

Figure 3: Old FFT, Third Signal

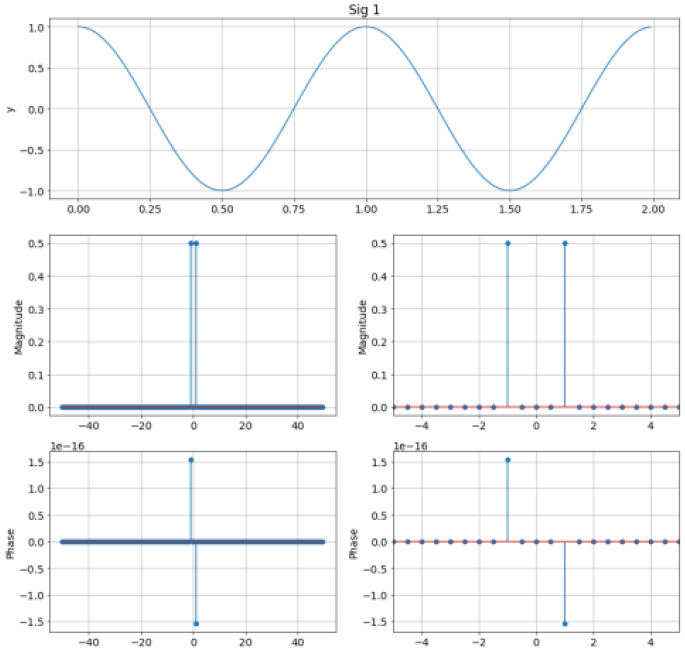

Figure 4: New FFT, First Signal

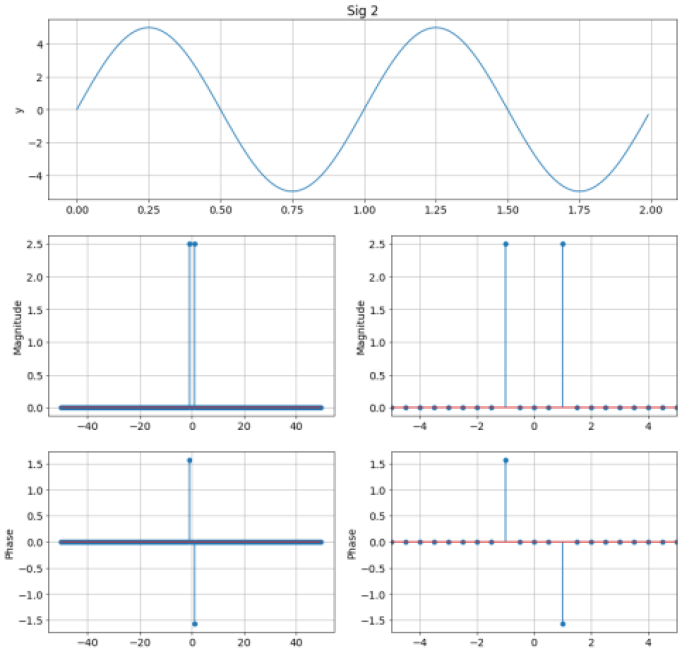

Figure 5: New FFT, Second Signal

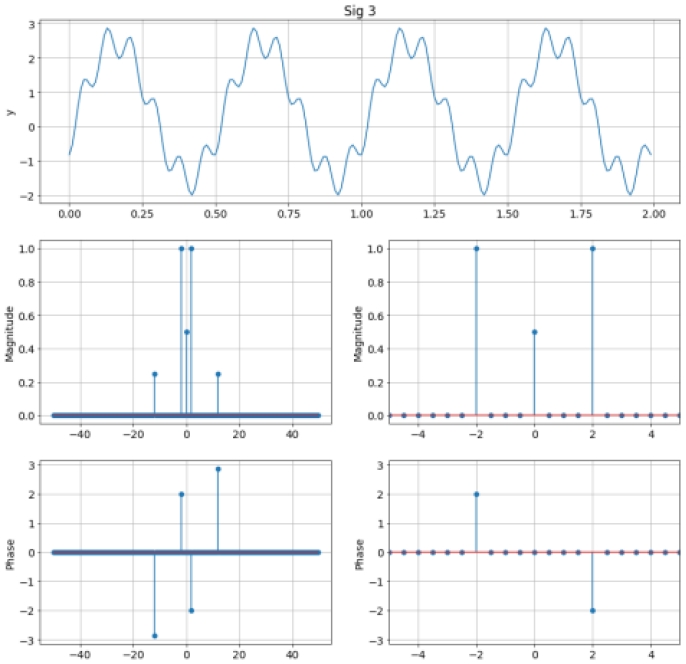

Figure 6: New FFT, Third Signal

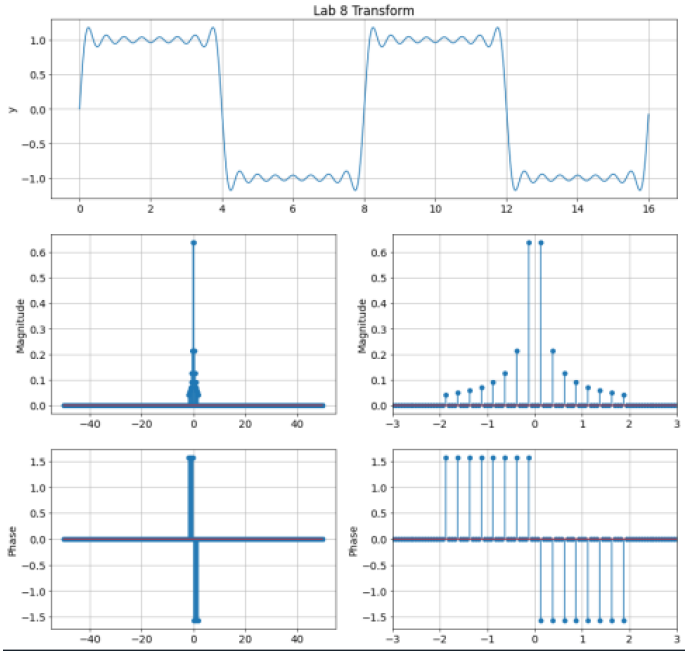

Figure 7: New FFT, Third Signal

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 period Ts will increase which should decrease the resolution since the sampling period is larger. Likewise, if fs is higher, then the period Ts will decrease which should increase the resolution since the sampling period is smaller.

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

The elimination of the small phase magnitudes lets us distinguish what points in the phase are important to us and which are not. Without eliminating the smaller phase magnitudes, all of our values would mean nothing since they are lost in noise.

3. Verify your results from Tasks 1 and 2 using the Fourier transforms of cosine and sine. Explain why your results are correct.

When looking at the first signal, $cos(2\pi t)$, the fourier transform for this becomes $\frac{1}{2}[(\delta(f-1)+\delta(f+1)]$. The amplitude for this is 0.5, and the delta functions occur at x=1 and x=-1. When looking at the magnitude plot for signal 1, we see that the maximum y-value, the amplitude, is 0.5 and the only two vertical lines in the stem plot occur at x=1 and x=-1 which indicates that my result for task 1 was correct. In the case of task 2, the fourier transform of the signal becomes $j\frac{5}{2}[(\delta(f-1)-\delta(f+1)]$. When looking at the plot we can see that the amplitude is 2.5 and the two vertical lines in the stem plot occur at x=1 and x=-1, which indicates that the plot matches the expected results as depicted in the fourier transform of signal 2.

4. Leave any feedback on the clarity of lab tasks, expectations, and deliverables.

I did not have any problems understanding what the lab report was asking of me.