Project Report for ECE 351

Lab 9: Fast Fourier Transform

Isaias Munoz

October 30, 2022

University of Idaho

Contents

1 Introduction

The purpose of this lab is to create a user define function to do a Fast Fourier Transfor (FFT).

2 Methodology

2.1 Part 1

The first part of the lab was to use the code provided in Figure 1 to create the fast Fourier transform.

```
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 in your code
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
# ----- End of user defined function ----- #

plt.stem(freq, X_mag) # you will need to use stem to get these plots to be
plt.stem(freq, X_phi) # correct, remember to label all plots appropriately
```

Figure 1: Pre code given to complete the lab

Even thought this code was not complete it served as a basis to creating a user define FFT which was really helpful. There are three main functions that will need to be plotted and they are below.

$$x(1) = cos(2 * \pi * t)$$

$$x(2) = 5sin(2 * \pi * t)$$

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

There is a fourth and it will be labs 8 square function approximated at a specific k value and period.

First I began with creating the inputs to the function $myfft(x, fs)$. As can be seen in the code for figure 2. I then made it return $freq, X_(mag), X_(phi)$. For the first task the first function that needed to be plot was $cos(2 * pi * t)$ using a time range of 2 seconds along with an frequency sample of 100. I then called the function in lines 55-57 and

```
1
31  fs=100
32  T=1/fs
33  t = np.arange(0, 2 , T)
34  x1=np.cos(np.pi*2*t)
35  x2=5*np.sin(np.pi*2*t)
36  x3=2*np.cos(np.pi*4*t-2)+np.sin((12*np.pi*t)+3)**2
37
38  def myfft(x,fs):
39
40
41      N = len(x) # find the length of the signal
42      X_fft = scipy.fftpack.fft(x) # perform the fast Fourier transform (fft)
43      X_fft_shifted = scipy.fftpack.fftshift(X_fft) # shift zero frequency components
44  # to the center of the spectrum
45      freq = np.arange(-N/2, N/2)*fs/N # compute the frequencies for the output
46  # signal , (fs is the sampling frequency and
47  # needs to be defined previously in your code
48
49
50      X_mag = np.abs(X_fft_shifted)/N # compute the magnitudes of the signal
51      X_phi = np.angle(X_fft_shifted) # compute the phases of the signal
52  # ----- End of user defined function ----- #
53      return freq,X_mag,X_phi #Need to assign these to a value
54  ##Starting plots for task 1,2,3
55  freq1,X_mag1,X_phi1=myfft(x1,fs)#labeling the returns so i can acces them when i go to plot them
56  freq2,X_mag2,X_phi2=myfft(x2, fs)
57  freq3,X_mag3,X_phi3=myfft(x3, fs)
58
59
60  #performing the fourier of the three functions task 1 and 2 and 3
61  # fft1=myfft(x1,fs)
62  # fft2=myfft(x2,fs)
63  # fft3=myfft(x3,fs)
```

Figure 2: User defined FFT unclean version

passed it task 1 function and the frequency samples and made all of this equal to the frequency, magnitude and phase of the fft user define function. Task 1 also said to plot here it was a little different as can be seen in Figure 2 when using time to plot a normal *plt.plot* works but when you introduce frequency you need to use *plt.stem*.

Figure 3 shows the plotting of task 1 and how to go about it.

```
#Performing task 1 for the first signal
plt.figure(figsize=(10,7))
plt.subplot(3,1,1)
plt.plot(t,x1)
plt.grid()
plt.ylabel('Y output')
plt.title('Task 1')


plt.subplot(3,2,3)
plt.stem(freq1,X_mag1)
plt.grid()
plt.ylabel('magnitutde ')

plt.subplot(3,2,4)
plt.stem(freq1,X_mag1)
plt.xlim([-2,2])
plt.grid()
plt.ylabel('magnitutde ')


plt.subplot(3,2,5)
plt.stem(freq1,X_phi1)
plt.grid()
plt.ylabel('phase')

plt.subplot(3,2,6)
plt.stem(freq1,X_phi1)
plt.xlim([-2,2])
plt.grid()
plt.ylabel('phase')
```

Figure 3: User defined FFT unclean version

As can be seen the plotting of $x1$ which is just $cos(2*pi*t)$ is straightforward and does not need any special plotting technique. Moving to plotting $frequency$ vs $magnitude$ we need to use $plt.stem$. This is repeated because it is asked to limit the $x-axis$ to an adequate window [-2,2]. Therefore for the $magnitude$ and $phase$ there are doubles with limited windows to focus better. The same process is used to graph the rest of the equations presented earlier which correspond to Task 2 and 3. This leaves task 4 which is described below.

In task 4 we needed to make the FFT cleaner in short. We do this by limiting the phase elements of to a certain value. Figure 4 shows the $myfft2$ which is the same user define FFT used earlier but with an if statement to restrict phases at a certain number and not show them. In this case the value that will not be accepted is $1e-10$.

```
177  def myfft2(x,fs):
178
179
180      N = len(x) # find the length of the signal
181      X_fft = scipy.fftpack.fft(x) # perform the fast Fourier transform (fft)
182      X_fft_shifted = scipy.fftpack.fftshift(X_fft) # shift zero frequency compor
183  # to the center of the spectrum
184      freq = np.arange(-N/2, N/2)*fs/N # compute the frequencies for the output
185  # signal , (fs is the sampling frequency and
186  # needs to be defined previously in your code
187
188      X_mag = np.abs(X_fft_shifted)/N  # compute the magnitudes of the signal
189      X_phi=np.angle(X_fft_shifted)
190      for i in range (len(X_mag)):
191          if X_mag[i]<1e-10:
192
193              X_phi[i]=0
194          # else:
195              # X_phi = np.angle(X_fft_shifted) # compute the phases of the signa
196  # ----- End of user defined function ----- #
197      return freq,X_mag,X_phi #Need to assign these to a value
198  #task 2 doing it
199  freq1new,X_mag1new,X_phi1new=myfft2(x1,fs)#labeling the returns so i can acces
200  freq2new,X_mag2new,X_phi2new=myfft2(x2, fs)
201  freq3new,X_mag3new,X_phi3new=myfft2(x3, fs)
202
203
```

Figure 4: User defined FFT clean version

As seen in Figure 4 if the magnitude is less then the specific value we make the phase equal 0 if not then we continue normally. Doing so the same plotting technique is repeated like figure 3 and the graphs are shown in the result section. Lastly Task 5 is done in a similar way except this time we are going to use a square wave as our input. The square wave is going to be Lab's 8 which we approximated with a certain period and K number. Below is the code used from Lab 8 with a few changes in the inputs.

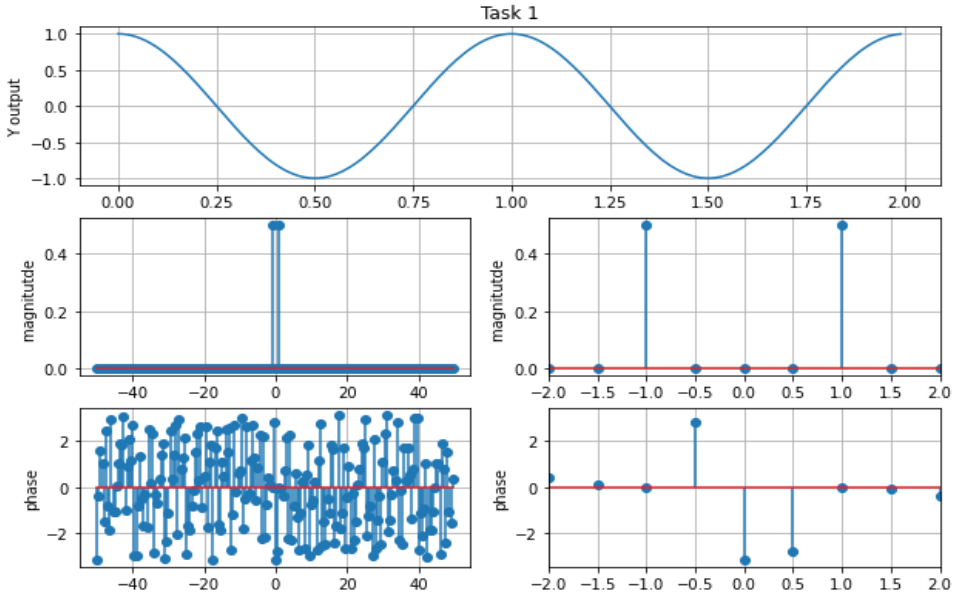
```
T5 = 8 #whatver value
w5 = (2*np.pi)/T5
ak = 0  #this is ze
k=15
##Doing task 5 now
timerange = np.arange(0,16  , T)

def fourierfunc(k,t):
    #k=n or number of iterations
    y=0
    for i in range(1,k+1):
        # y+=bkfunc(i)*np.sin((2*np.pi*i*t)/T)
        # y+=(2/(i*np.pi))*(1-np.cos(i*np.pi))*np.sin((i*w*t)
         y +=  (2/(i * np.pi)) * (1 - np.cos(i * np.pi)) * (np.sin(i * w5 * timerange))

    return y


task5func=fourierfunc(k,T5)
freqtask5,X_magtask5,X_phitask5=myfft2(task5func,fs)#labeling the returns so i can acces
```

Figure 5: Using Lab 8 square wave approximation code

As can be seen in Figure 5 the code approximate the square wave at an period and K value or iteration. In this task k=15 and everything else stay the same with a time span of 16 instead this time. The same approach was used as the different functions used earlier by passing this square wave into the clean FFT which is the $myfft2$ the one with the if statement. This same approach was used in solving the first three input signals. Also the same approach to graphing the frist three inputs talked earlier about was used to graph. A total of 35 graphs were constructed.

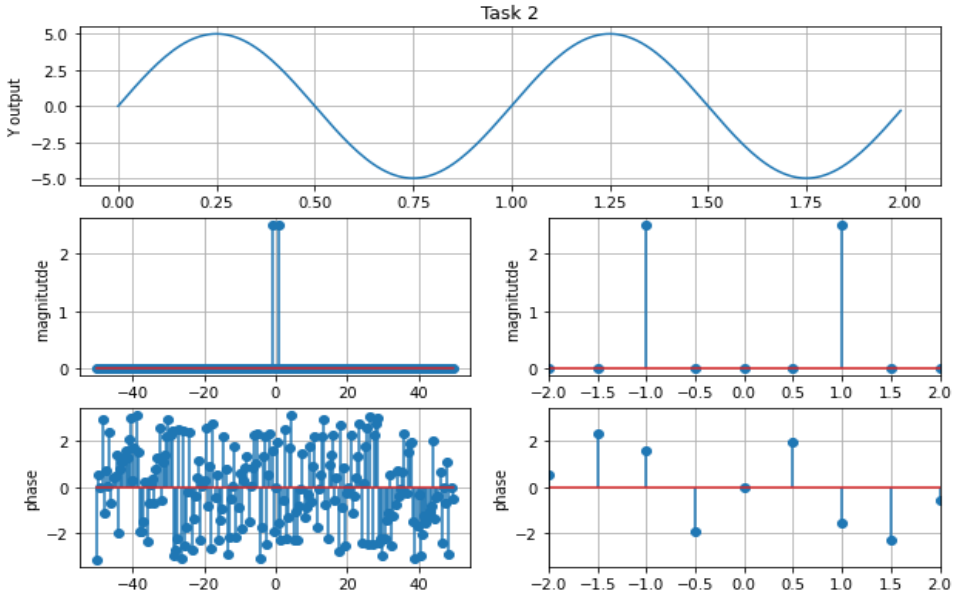
3 Results

3.1 Part 1



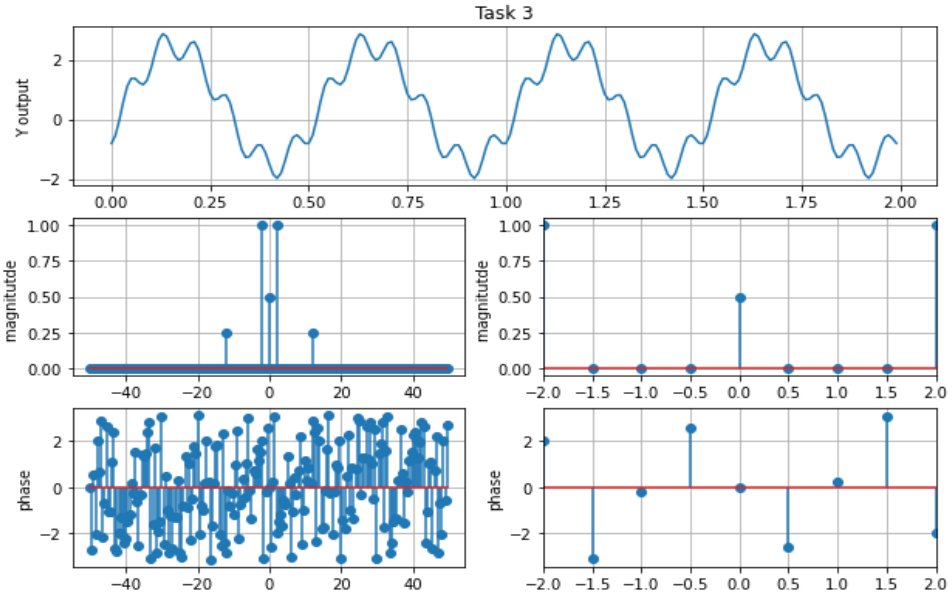
1

Figure 6: Cos(2πt) unreadable



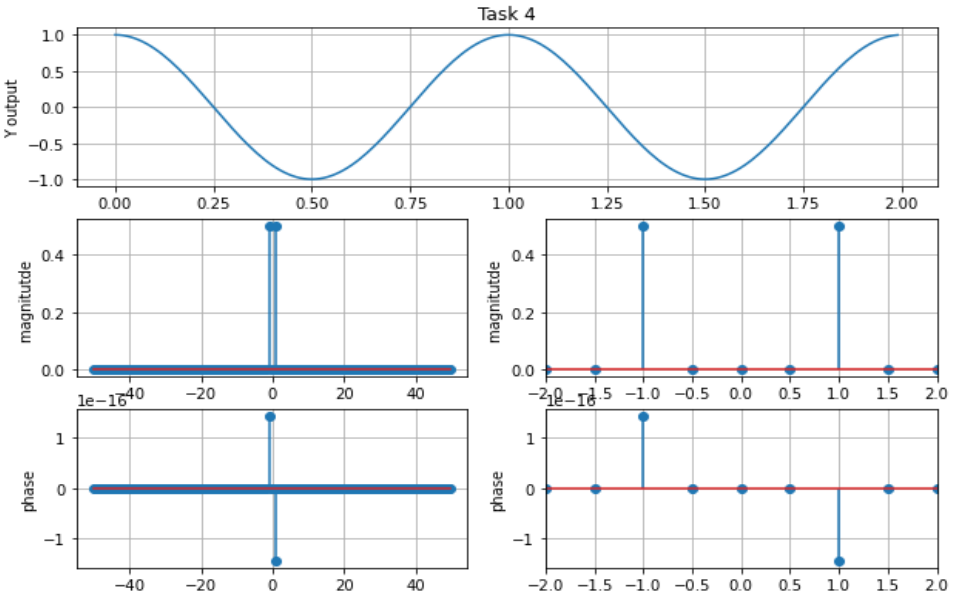
1

Figure 7: 5sin(2πt) unredabale



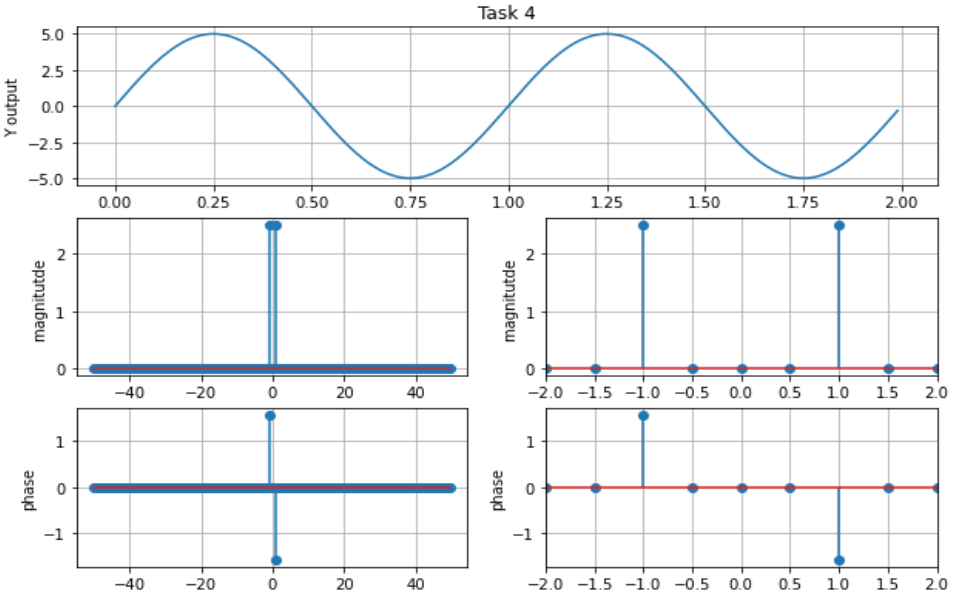
1

Figure 8: 2*cos((2*π*t)-2)+sin^2((2 ∗ π ∗ 6 ∗ t))*unreadable*



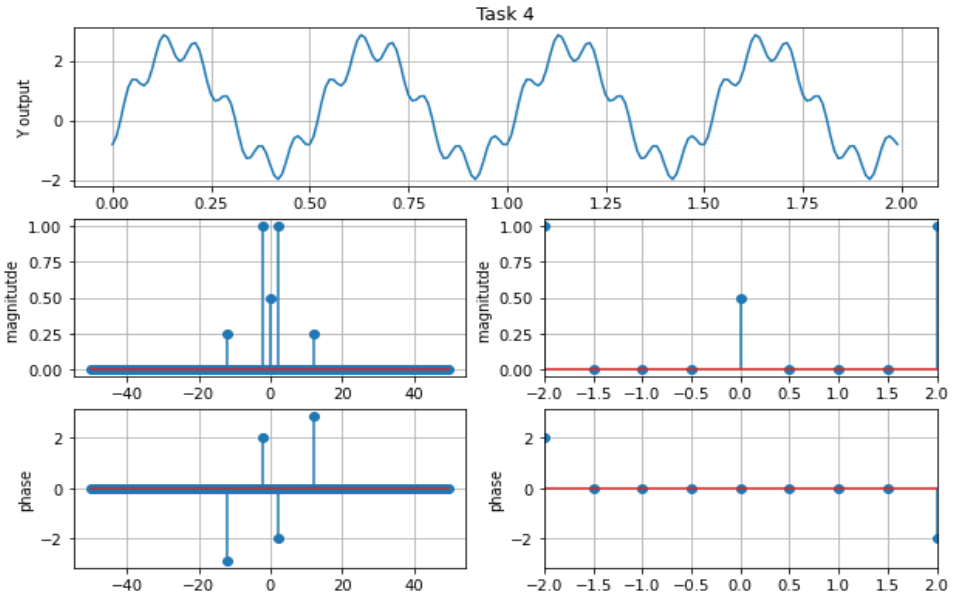
1

Figure 9: Cos(2πt) readable



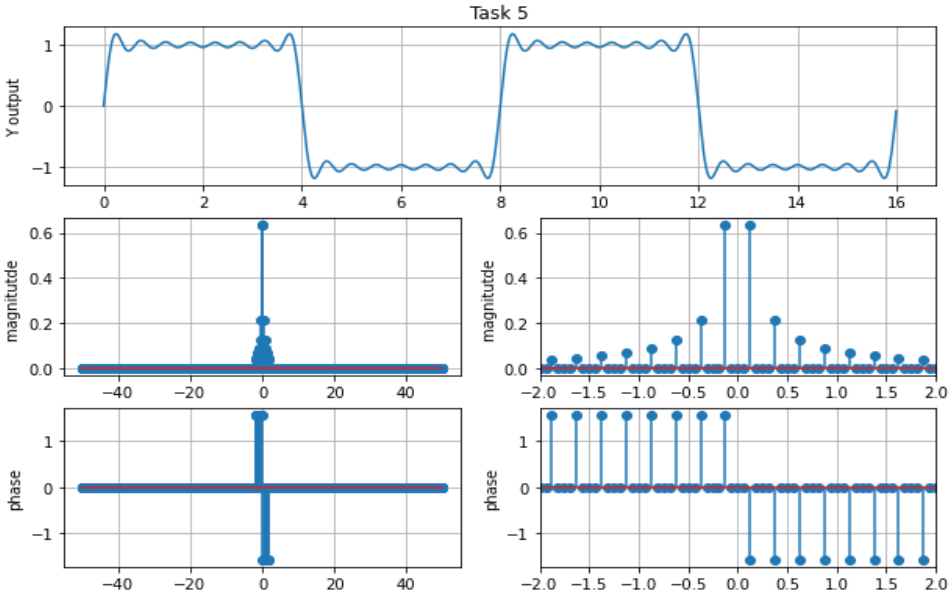
1

Figure 10: 5sin(2πt) readable



1

Figure 11: 2*cos((2*π*t)-2)+$\sin^2((2 * \pi * 6 * t))$*readable*



1

Figure 12: Approximation of Lab's 8 square wave readable

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 then the period is bigger if fs is higher then the period is smaller. A larger period or repetition of the wave means more approximations needed to get a good picture of it.

2. What difference does eliminating the small phase magnitudes make?
We might not care about the small spikes in our graph so eliminating them help us see the important points.

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: Fcos(2πf0t)= 1/2 [δ (f f0) + δ (f + f0)]
For task one it becomes 1/2 [δ (f 1) + δ (f + 1)] which match my graph for task 1 occurring at x=1 and x=2.

4. Leave any feedback on the clarity of lab tasks, expectations, and deliverables.
It was straightforward I think.