Assignment - 8

Ramanan S (EE18B145)

April 23, 2020

1 Introduction

In this assignemnt, we are going to find the discrete fourier transform of various signals using the fft function of numpy. We also try to approximate the CTFT of a gaussian by taking samples and finding it DFT. We attain the required accuracy by increasing the number of samples and the window size iteratively.

2 Procedure

Importing required libraries.

[1]:
```
from pylab import *
```

2.1 Part - 0

Writing general function to output the transform given any function. The function *transform* takes the function, the time interval and the number of samples as the input and plots the magnitude and phase plot of the DFT.

[2]:
```
def transform(func = sin, T = 8*pi, N = 512, lim = 5, ret = False, c = 1):
    t=linspace(-T/2,T/2,N+1);t=t[:-1]
    y=func(t)
    Y=fftshift(fft(ifftshift(y)))/N*c
    w=linspace(-N*(pi/T),N*(pi/T),N+1);w=w[:-1]
    figure()
    subplot(2,1,1)
    plot(w,abs(Y),lw=2)
    xlim([-lim,lim])
    ylabel(r"$|Y|$",size=16)
    grid(True)
    subplot(2,1,2)
    ii = where(abs(Y) > 1e-3)
    plot(w,angle(Y),'go',markersize= 3)
    plot(w[ii],angle(Y[ii]),'ro',markersize = 5)
    xlim([-lim,lim])
    ylabel(r"Phase of $Y$",size=16)
    xlabel(r"$\omega$",size=16)
```

```
    grid(True)
    show()
    if(ret):
        return Y
```

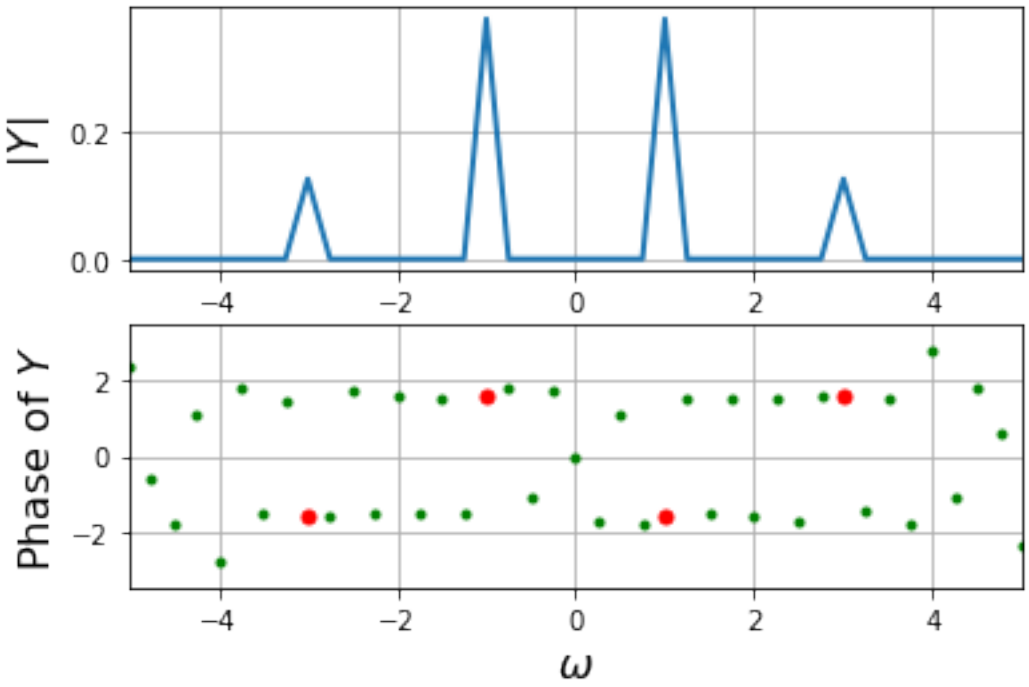
2.2 Part 2 - sin^3x and cos^3x

We can write :

$$sin^3(t) = \frac{3}{4}sin(t) - \frac{1}{4}sin(3t)$$

So, we expect peaks at $\omega = 3$ and $\omega = 1$ with amplitudes -0.25 and 0.75 respectively.

[3]:
```
transform(func = lambda x : (sin(x))**3, lim = 5)
```



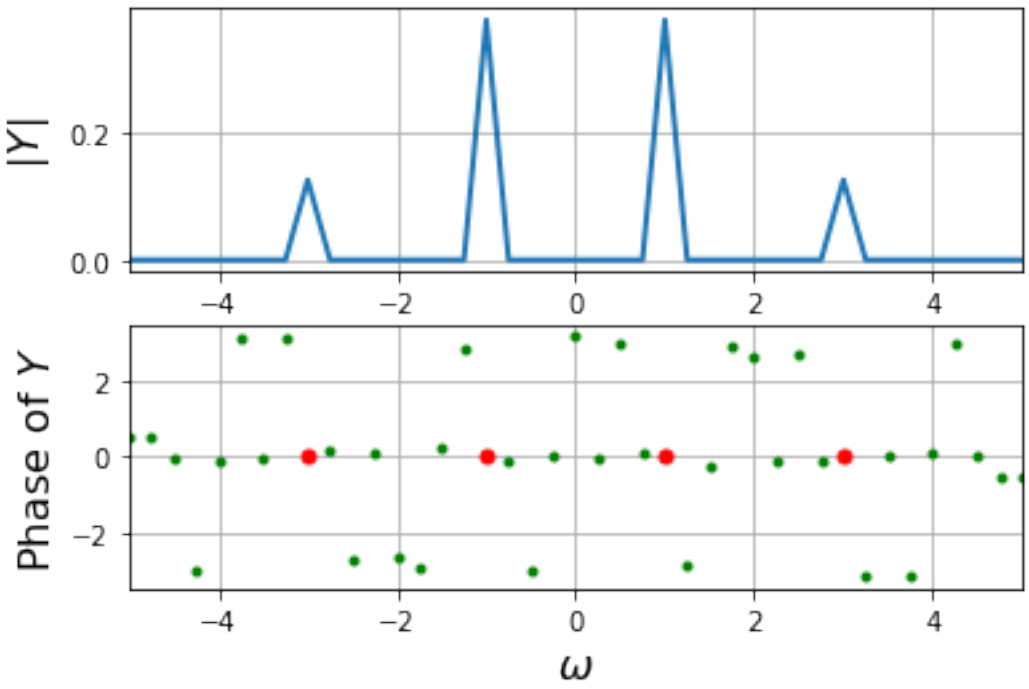
It can be observed that the output is as expected, the magnitude and phase both satisfy the above equation.

Now :

$$cos^3(t) = \frac{3}{4}cos(t) + \frac{1}{4}cos(3t)$$

So, we expect peaks at $\omega = 3$ and $\omega = 1$ with amplitudes 0.25 and 0.75 respectively.

[4]:
```
transform(func = lambda x : (cos(x))**3, lim = 5)
```



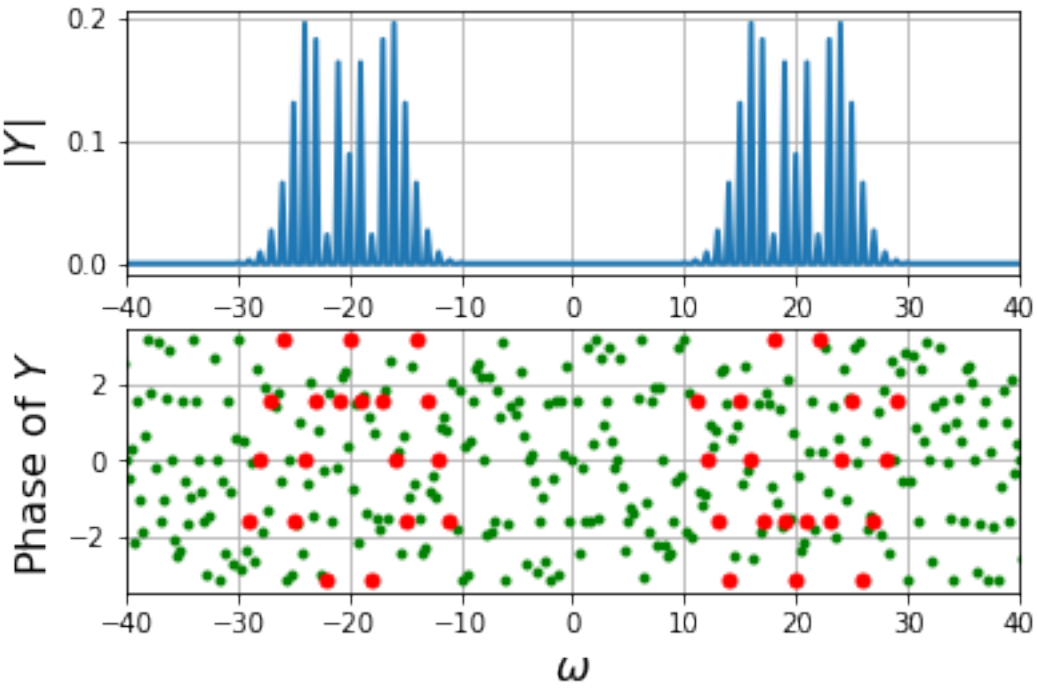
It can be observed that the output is as expected, the magnitude and phase both satisfy the above equation.

2.3 Part 3 - Frequency Modulation

We have,

$$x(t) = cos(20t + 5cos(t))$$

[5]:
```
transform(func = lambda x : cos(20*x + 5*cos(x)), lim = 40)
```



It is observed that the frequency band contains more number of peaks than the ones considered before and their amplitudes are also comparable. So, the energy is distributed among and the frequencies in the band almost equally.

2.4 Part 4 - Approximating CTFT of Gaussian

We have

$$x(t) = e^{-t^2/2}$$

The analytical expression of the CTFT can be found as,

$$X(e^{j\omega}) = \frac{1}{\sqrt{2\pi}} e^{-\omega^2/2}$$

Approximating the Fourier integral using Reimman sum, we can write:

$$X(k\Delta\omega) \approx \frac{\delta t}{2\pi} DFT\{x(n\delta t)\}$$

We start with $N = 128$ and $T = 8\pi$ and keep on increasing N and T by doubling them until we reach the required accuracy (which is given to be 10^{-6}).

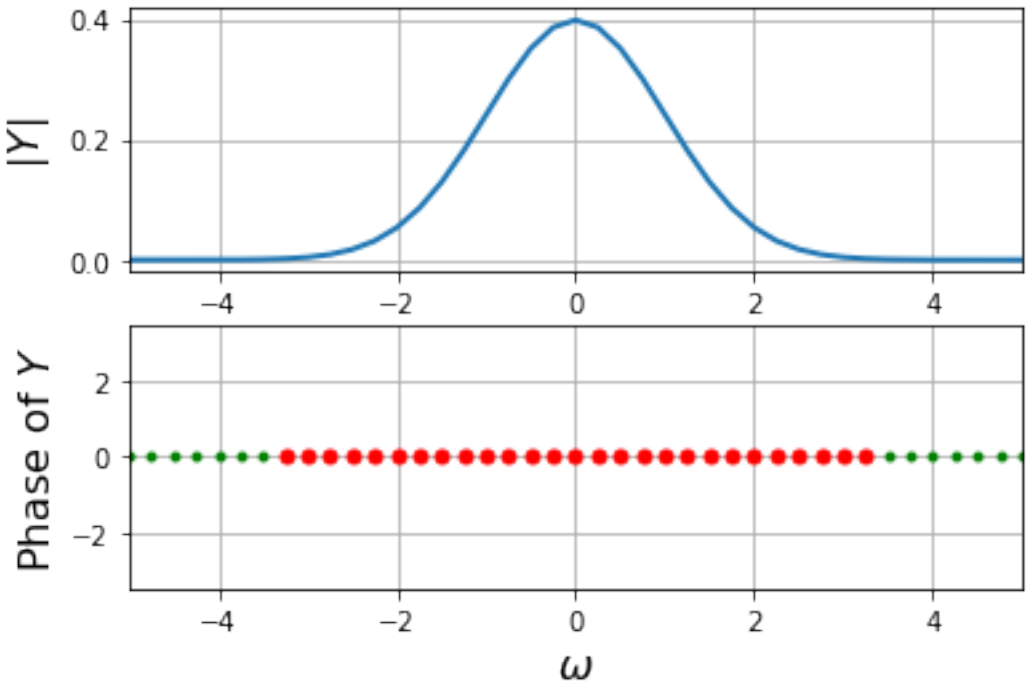
[6]:
```
def gaussian_transform(x):
    return (1/sqrt(2*pi))*exp((-x**2)/2)
```

[7]:
```
N = 128
T = 8 * pi
```

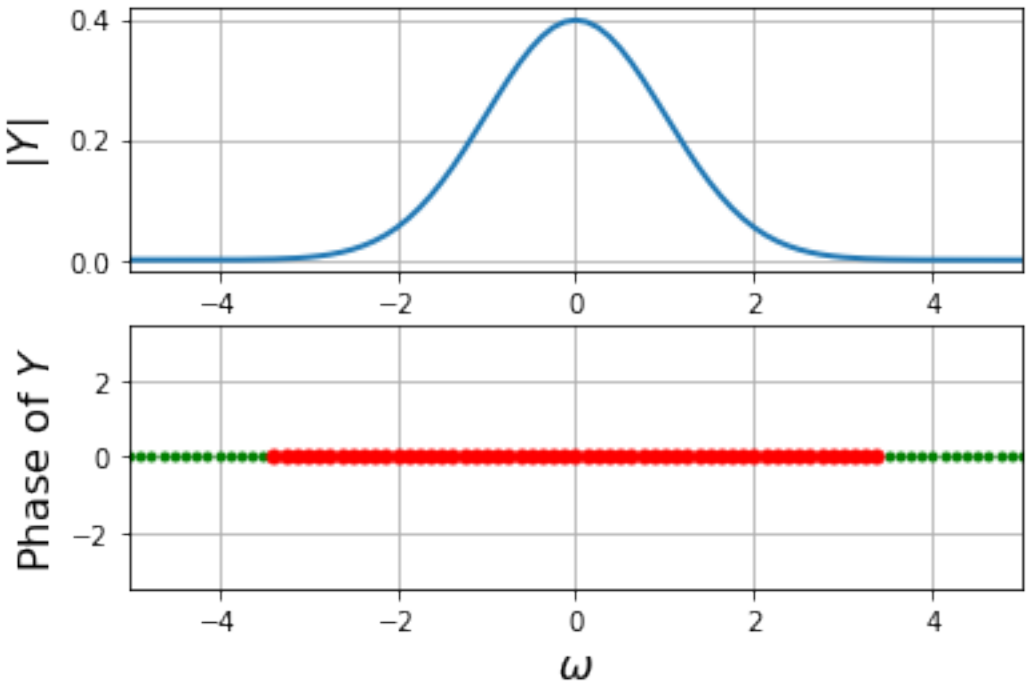
```
err = 1
tol = 1e-6
yprevious = 0
while(err > tol):
    print('For N = '+str(N)+' and T = '+str(T))
    y = transform(func = lambda x : exp(-x**2/2), T = T, N = N, ret = True, c =␣
 ↪T/(2*pi))
    err = sum(abs(y[::2]-yprevious))
    yprevious = y
    N = 2*N
    T = 2*T

print('Preferred Values are :')
print('N = '+str(N))
print('T = '+str(T))
print('True Error = '+str(err))
print('The original transform is:')
w = linspace(-N*(pi/T),N*(pi/T),N+1);w=w[:-1]
y = gaussian_transform(w)
subplot(2,1,1)
plot(w,abs(y),lw=2)
xlim([-5,5])
ylabel(r"$|Y|$",size=16)
grid(True)
subplot(2,1,2)
grid()
ylabel("Phase of Y",size=16)
plot(w,angle(y),'go',markersize= 3)
show()
```

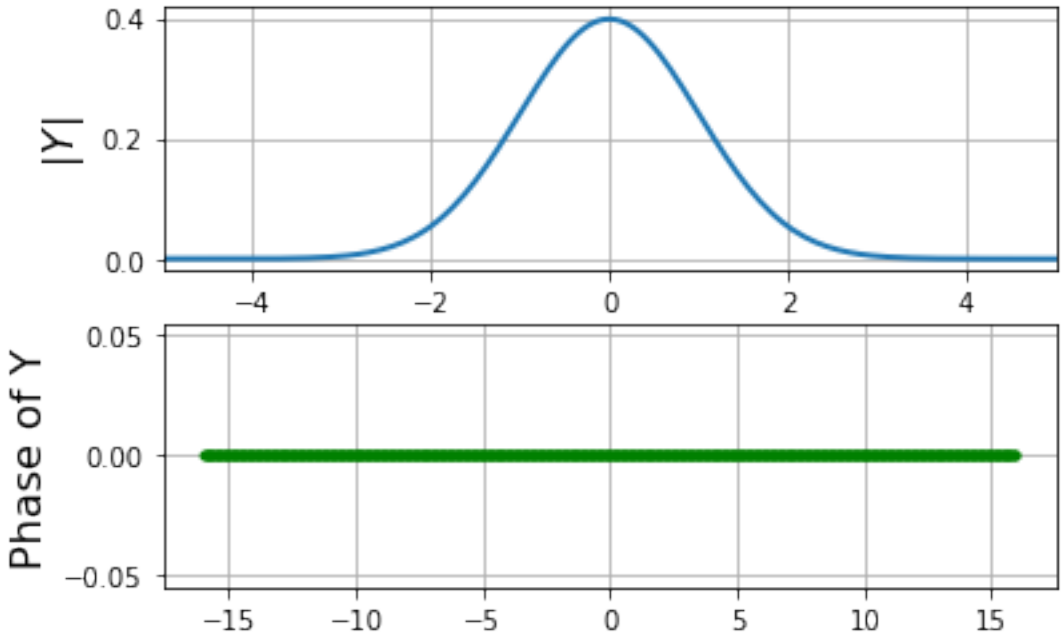
```
For N = 128 and T = 25.132741228718345
```



For N = 256 and T = 50.26548245743669



```
Preferred Values are :
N = 512
T = 100.53096491487338
True Error = 8.960039519784248e-15
The original transform is:
```



3 Conclusion

- From the above pairs of plots, it is clear that with a sufficiently large window size and sampling rate, the DFT approximates the CTFT of the gaussian.
- Sampling after windowing is done so that the DFT can be calculated using the Fast Fourier Transform. This is then a sampled version of the DTFT of the sampled time domain signal. With sufficiently large sampling rates, this approximates the CTFT of the original time domain signal.
- This process is done on the gaussian and the results are in agreement with what is expected