Assgnment 8 - EE17B114

April 3, 2019

In this week's assignment, the digital fourier transform of some given functions were found out using python functions numpy.f f t.f f t() . Then, the values found by this function were normalized to their actual numpy.f f t.f f tshif t() , after which the phase and magnitude plots of the resulting transforms for obtaining the same like numerical values using were plotted.

```
In [2]: import numpy as np
        from numpy import fft as f
        import matplotlib.pyplot as plt
        import math
        from pylab import *
```

PREREQUISITE FUNCTIONS ARE DEFINED:

```
In [2]: def f1(x):
            return np.sin(5*x)

In [3]: def f2(x):
            return (1 + 0.1*np.cos(x))*np.cos(10*x)

In [4]: def f3(x):
            return np.sin(x)**3

In [5]: def f4(x):
            return np.cos(x)**3

In [6]: def f5(x):
            return np.cos(20*x + 5*np.cos(x))

In [7]: def f6(x):
            return np.exp(-(x*x)/2)

In [8]: def f7(x):
            index = np.argwhere(x==0.0)[0][0]
            x = np.delete(x, index)
            z=np.sin(x)/x
            y = []
            for i in range(0,len(z)):
                if i != index :
```

```
                y.append(z[i])
            else:
                y.append(1)
                y.append(z[i])
        return y
```

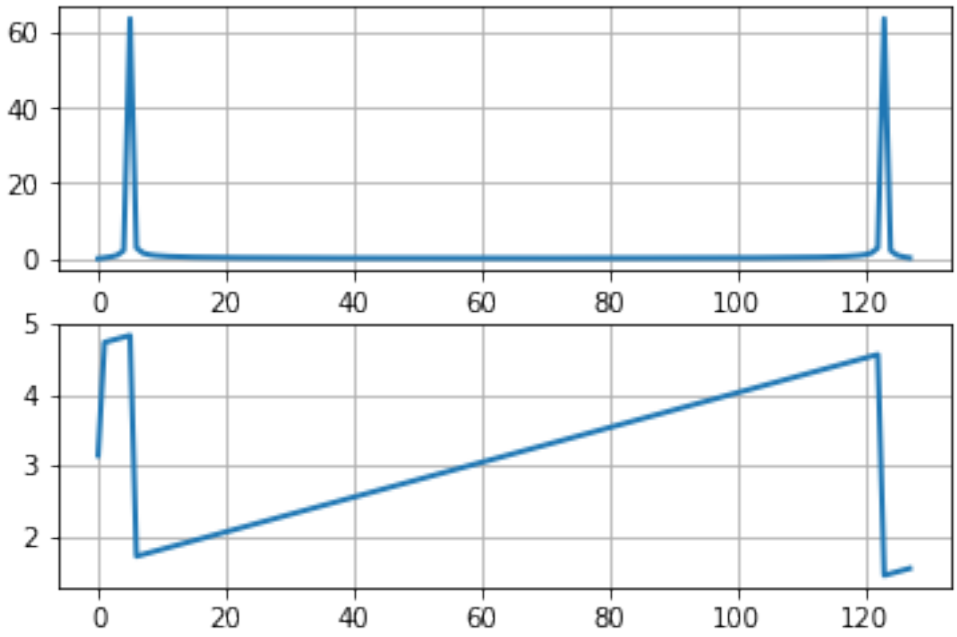
```
In [9]: def plot_fft(Y,x_lim,y_lim,function,steps,offset,show_data_points):
            w=np.linspace(-64,64,steps+1)
            w = w[:-1]
            ctr = 0
            fig, axes = plt.subplots(2, 1, figsize=(15, 7), sharex = True)

            plt.suptitle("The DFT plots for " + function, fontsize=18)

            axes[0].plot(w,abs(Y),lw=2)
            if show_data_points:
                for xy in zip(w, abs(Y)):
                    if xy[1] > 1e-3:
                        axes[0].annotate('(%s, %s)' % xy, xy=xy, textcoords='data')
            axes[0].set_xlim([-x_lim,x_lim])
            axes[0].set_ylabel(r"$|Y|$",size=16)
            axes[0].set_title("Spectrum of " + function, fontsize=14)
            axes[0].grid(True)

            ii=np.where(abs(Y)>1e-3)
            axes[1].plot(w[ii],np.angle(Y[ii]),'go',lw=2)
            if show_data_points:
                for xy in zip(w[ii], np.angle(Y[ii])):
                    axes[1].annotate('(%0.2f, %0.2f)' % xy, xy=(xy[0],xy[1]+((-1)**ctr)*offset),
                    ctr = ctr + 1
            axes[1].set_xlim([-x_lim,x_lim])
            axes[1].set_ylim([-y_lim,y_lim])
            axes[1].set_ylabel(r"Phase of $Y$",size=16)
            axes[1].set_title("Phase Plot of " + function, fontsize=14)
            axes[1].set_xlabel(r"$k$",size=16)
            axes[1].grid(True)
            plt.show()
```

```
In [28]: def calc_fft(func, steps):
             x=np.linspace(-8*math.pi,8*math.pi,steps+1);
             x=x[:-1]
             y=func(x)
             Y=f.fftshift(f.fft(y))/steps
             return Y
```

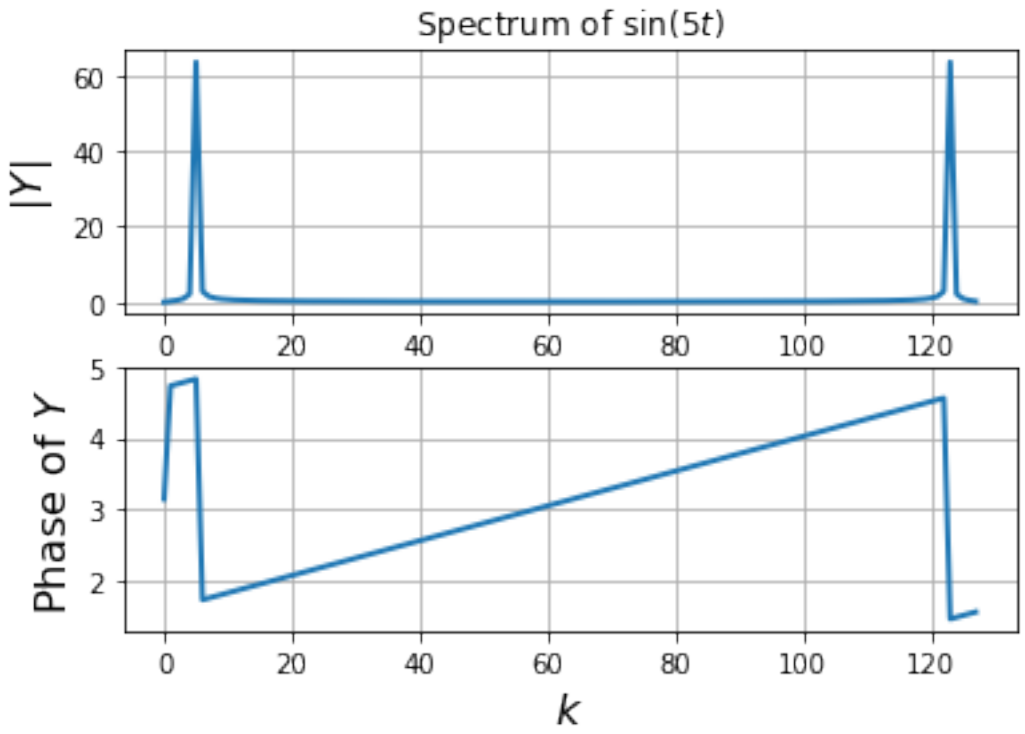
CODES GIVEN TO BE TRIED OUT:

```
In [3]: x=linspace(0,2*pi,128)
        y=sin(5*x)
```

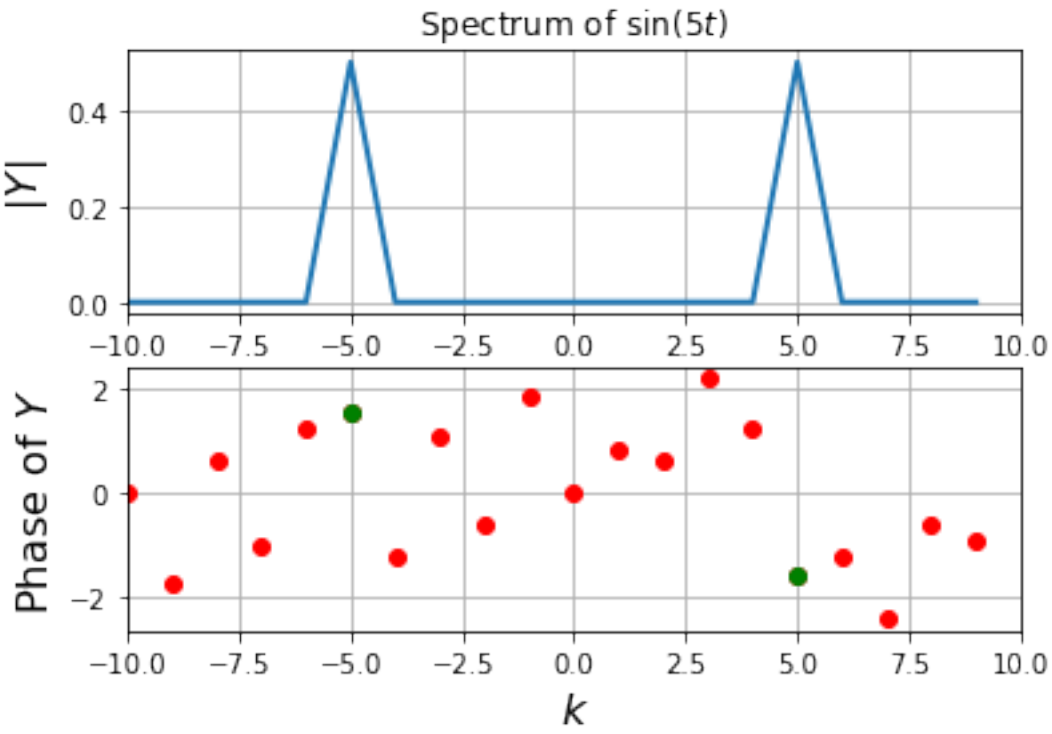
```
Y=fft(y)
figure()
subplot(2,1,1)
plot(abs(Y),lw=2)
grid(True)
subplot(2,1,2)
plot(unwrap(angle(Y)),lw=2)
grid(True)
show()
```



In [4]:
```
x=linspace(0,2*pi,128)
y=sin(5*x)
Y=fft(y)
figure()
subplot(2,1,1)
plot(abs(Y),lw=2)
ylabel(r"$|Y|$",size=16)
title(r"Spectrum of $\sin(5t)$")
grid(True)
subplot(2,1,2)
plot(unwrap(angle(Y)),lw=2)
ylabel(r"Phase of $Y$",size=16)
xlabel(r"$k$",size=16)
grid(True)
savefig("fig9-1.png")
show()
```




```
In [5]: x=linspace(0,2*pi,21);x=x[:-1]
        y=sin(5*x)
        #Y = fft(y)
        Y=fftshift(fft(y))/20.0
        w=linspace(-10,9,20)
        figure()
        subplot(2,1,1)
        plot(w,abs(Y),lw=2)
        xlim([-10,10])
        ylabel(r"$|Y|$",size=16)
        title(r"Spectrum of $\sin(5t)$")
        grid(True)
        subplot(2,1,2)
        plot(w,angle(Y),'ro',lw=2)
        ii=where(abs(Y)>1e-3)
        plot(w[ii],angle(Y[ii]),'go',lw=2)
        xlim([-10,10])
        ylabel(r"Phase of $Y$",size=16)
        xlabel(r"$k$",size=16)
        grid(True)
        savefig("fig9-2.png")
        show()
```



2.1 $sin(5t) $ spectrum One of the simplest functions available, $sin(5t)$ is used, and its digital fourier transform plots (magnitude and phase) are plotted.

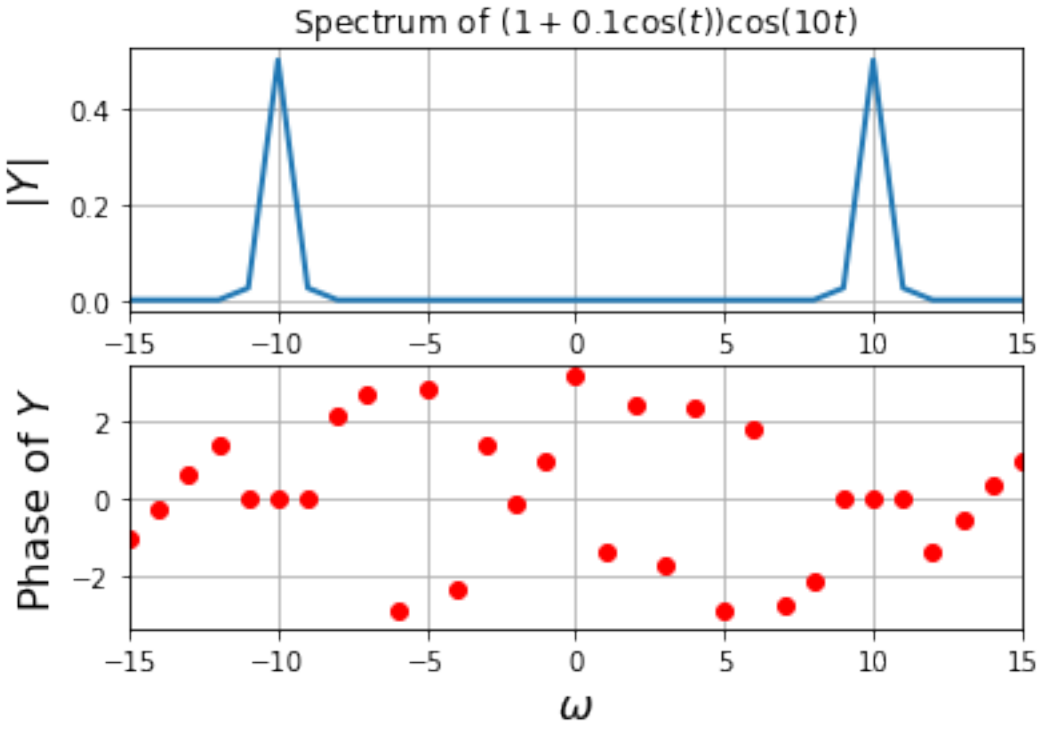
Thus, from the plots, it is seen that the impulse occurs at frequencies of +5 and -5, as expected as the input sine wave has a frequency of 5. Also, the phase at these points are + $\pi/2$ and $-\pi/2$, which is also expected, as the complex representation of sin(5t) is

$$sin(5t) = \frac{e^{-}j5t}{2j} - \frac{e^{-}j5t}{2j}$$

where, both elements are purely imaginary, with opposite signs and a magnitude of 0.5. Thus, the calculated graph is verified

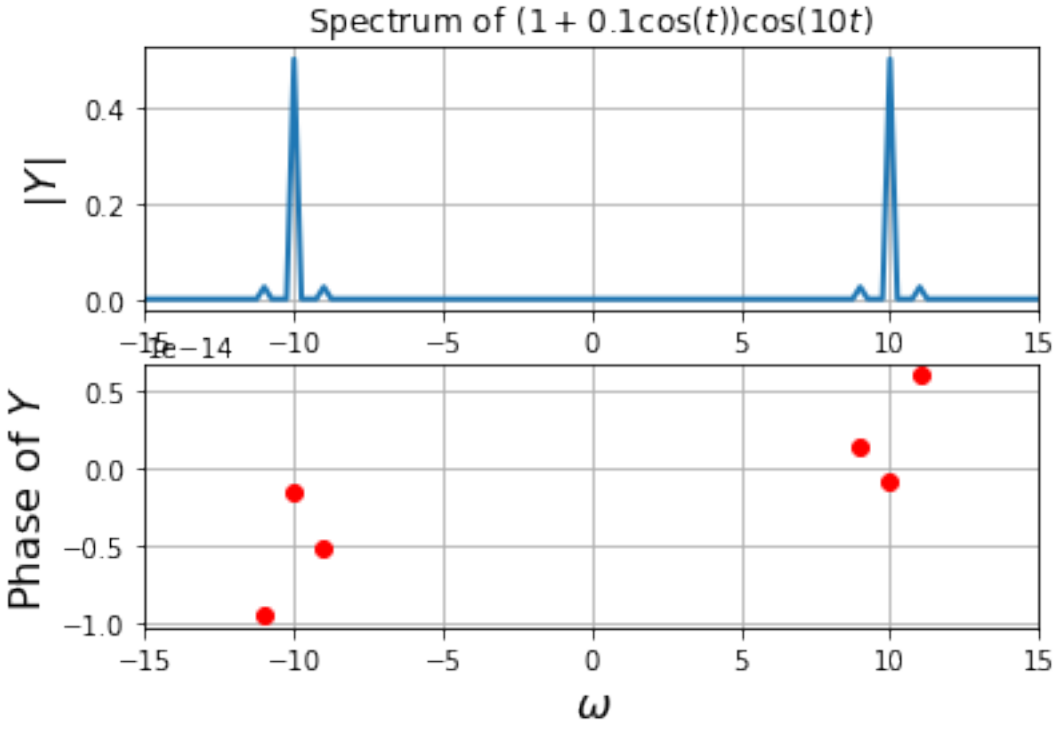
```
In [6]: from pylab import *
        t=linspace(0,2*pi,129);t=t[:-1]
        y=(1+0.1*cos(t))*cos(10*t)
        Y=fftshift(fft(y))/128.0
        w=linspace(-64,63,128)
        figure()
        subplot(2,1,1)
        plot(w,abs(Y),lw=2)
        xlim([-15,15])
        ylabel(r"$|Y|$",size=16)
        title(r"Spectrum of $\left(1+0.1\cos\left(t\right)\right)\cos\left(10t\right)$")
        grid(True)
```

```
subplot(2,1,2)
plot(w,angle(Y),'ro',lw=2)
xlim([-15,15])
ylabel(r"Phase of $Y$",size=16)
xlabel(r"$\omega$",size=16)
grid(True)
savefig("fig9-3.png")
show()
```



In [7]:
```
t=linspace(-4*pi,4*pi,513);t=t[:-1]
y=(1+0.1*cos(t))*cos(10*t)
Y=fftshift(fft(y))/512.0
w=linspace(-64,64,513);w=w[:-1]
figure()
subplot(2,1,1)
plot(w,abs(Y),lw=2)
xlim([-15,15])
ylabel(r"$|Y|$",size=16)
title(r"Spectrum of $\left(1+0.1\cos\left(t\right)\right)\cos\left(10t\right)$")
ii=where(abs(Y)>1e-3)
grid(True)
subplot(2,1,2)
plot(w[ii],angle(Y[ii]),'ro',lw=2)
```

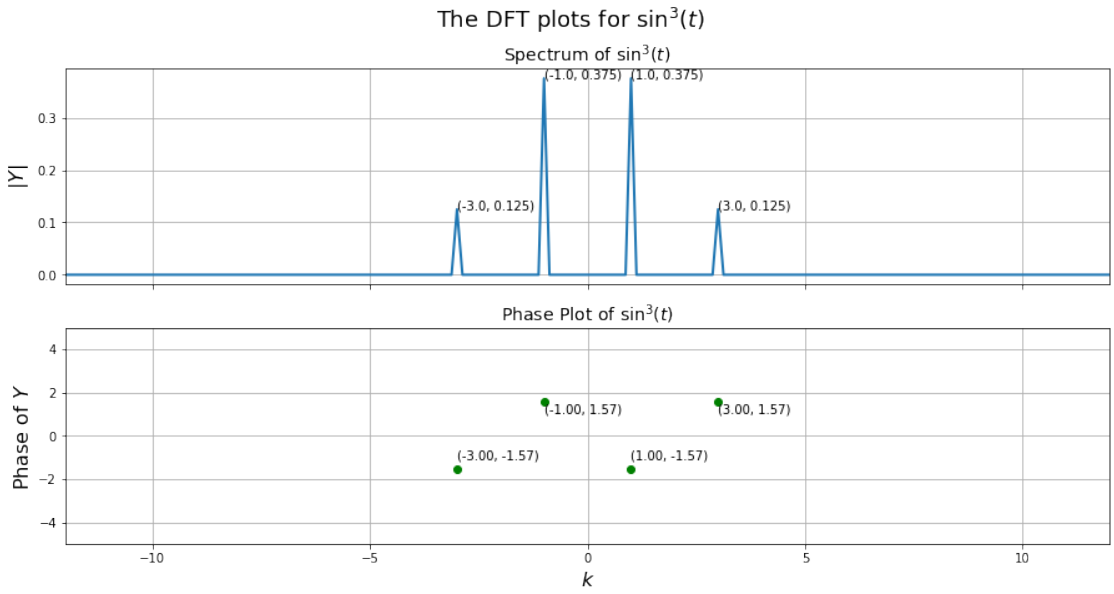
```
xlim([-15,15])
ylabel(r"Phase of $Y$",size=16)
xlabel(r"$\omega$",size=16)
grid(True)
savefig("fig9-4.png")
show()
```



2.2 $(1+0.1cos(t))cos(10t)$ spectrum In this example, spectrum of the the amplitude modulated wave $(1+0.1cos(t))cos(10t)$ is plotted, which is a slightly more complicated function than the one plotted above, and its obtained graphs are verified.

```
In [11]: no_of_steps=1024
```

```
In [14]: Y = calc_fft(f3, no_of_steps)
         plot_fft(Y,12,5,r"$\sin^3(t)$",no_of_steps,0.5,True)
```

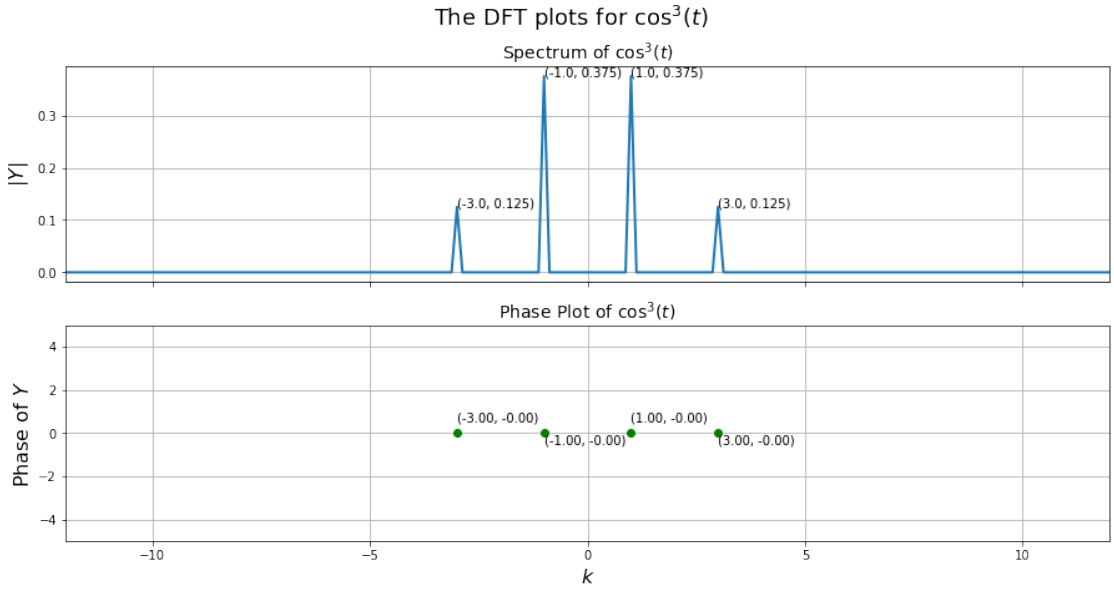


The complex representation of $f(t) = sin^3(t)$

$$f(t) = -j(0.375(e^jt - e^-jt) - 0.125(e^3jt - e^-3jt))$$

formula $sin(3t) = 3sin(t) - 4sin^3(t)$ can be used. Thus, the observation from the graph is verified by the formula, in that, the magnitude of the peak at 3 and -3 are 0.125, and those at 1 and -1 are 0.375. Also, the phase at frequencies of -1 and 3 are + $\pi/2$, while those at -3 and 1 are $-\pi/2$, which also can be observed in the formula, as the coecients of these terms are +j and $-$j , respectively, when the equation is expanded.

```
In [15]: Y = calc_fft(f4, no_of_steps)
         plot_fft(Y,12,5,r"$\cos^3(t)$",no_of_steps,0.5,True)
```

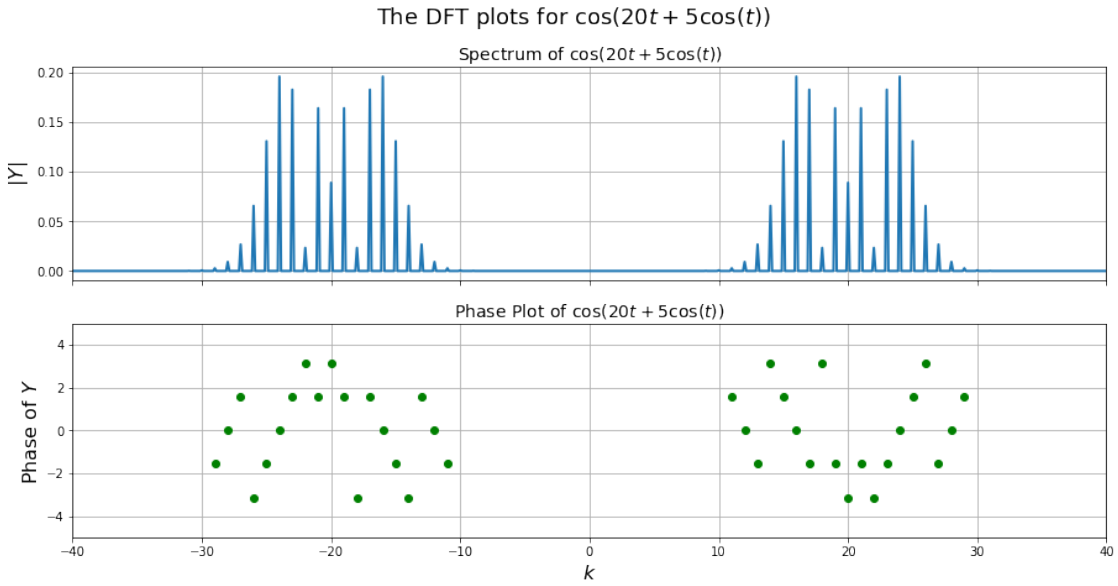


The complex representation of $f(t) = cos^3(t)$

$$f(t) = (0.375(e^jt + e^-jt) = 0.125(e^3jt + -e^-3jt))$$

formula $cos(3t) = 4cos^3(t) - 3cos(t)$ can be used. formula cos(3t) = 4cos^3(t) − 3cos(t) . Thus, the observation from the graph is verified by the formula, in that, the magnitude of the peak at 3 and -3 are 0.125, and those at 1 and -1 are 0.375. Also, the phase at all frequencies (-1, 1, -3 and 3) are 0, which can also be observed from the formula, as the coeficients of all these terms are positive real numbers.

```
In [16]: no_of_steps = 1024
         Y = calc_fft(f5, no_of_steps)
         plot_fft(Y,40,5,r"$\cos(20t + 5\cos(t))$",no_of_steps,0,False)
```




```
In [33]: def calc_fft_2(func, steps):
             x=np.linspace(-4*math.pi,4*math.pi,steps+1);
             x=x[:-1]
             y=func(x)
             Y=f.fftshift(f.fft(f.ifftshift(y)))/(steps/4)
             return Y
```

```
In [18]: def gauss(w):
             return np.sqrt(1/(2*math.pi))*np.exp(-(w**2)*2)
```

In this, the transform of the gaussian $e^{t^2/2}$ is found out. This is also expected to be a gaussian itself, with different scaling. For the input gaussian as:

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

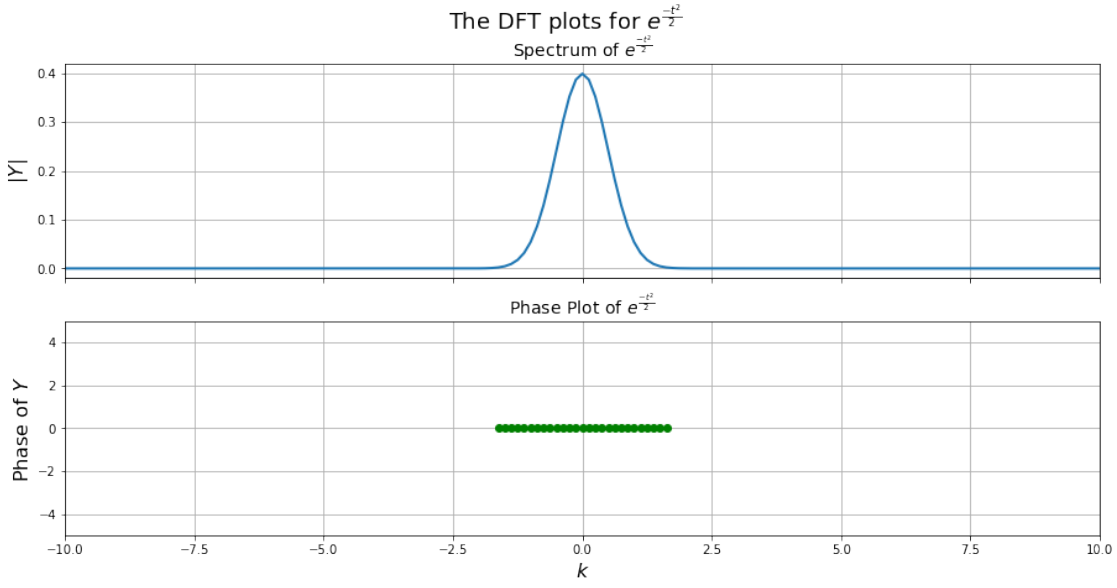
the fourier transform is also a gaussian, represented by

$$F(w) = \frac{e^{-2w^2}}{\sqrt{(2)\pi}}$$

So, the values obtained via numpy.f f t.f f t compared with one obtained with this formula, and the maximum error is computed. For this, another calc_fft() function is declared. This is because, for this case, the scaling factor, to be multiplied with Y is different, as this is a non-periodic function. So, only the no. of steps in a single period of 0 to 2π is to be divided. So, in this particular case, that is the 4 1 th of the total number of steps. Also, here the interval is also different ($-4\pi, 4\pi$), rather than ($-8\pi, 8\pi$), as was done for the other functions. This is so as to scale the output of the python computed fit correctly, so as to match with the actual transform of the function.

```
In [34]: no_of_steps=1024
         Y = calc_fft_2(f6, no_of_steps)
         w=np.linspace(-64,64,no_of_steps+1)
         w = w[:-1]
         print ("The maximum error between the expected and calculated gaussians are ", max(np.a
         plot_fft(Y,10,5,r"$e^{\frac{-t^2}{2}}$",no_of_steps,0,False)
```

```
The maximum error between the expected and calculated gaussians are  1.80490527317e-15
```



Thus, it is proved from the graph, that the resulting transform is also a gaussian, with all phases as 0, as it is also only a real valued gaussian, with no imaginary components. It also matches very closely with the expected gaussian, as the maximum error that is returned is of the order of 10^{-15} .

Conclusion Thus, from the initial plots, we verified the fourier transform that was calculated by python using numpy.f f t.f f t , as matching the expected ones, for those functions. Then,

the python function was used to calculate the fourier transforms of more complex functions like gaussian and a frequency modulated wave, for which, calculating their actual expressions are difficult. So, this obtained fourier transform was analysed and verified with the known properties of the signal.