Assignment No 8, DFT

Pranav Phatak, EE19B105

May 11, 2021

Objective

In this assignment, we analyse signals using the Fast Fourier transform. We do this by using the numpy.fft module.

Accuracy of DFT

To check how accurate the package is, we shall take the DFT of a sequence of random numbers, then take it's IDFT and compare the two, as to how close they are.

```
#Calculate error between actual and inversed values of dft of series of random value
x=np.random.rand(100)
X=fft(x)
y=ifft(X)
c_[x,y]
maxError = max(np.abs(y-x))
print('Magnitude of maximum error between actual and computed values of the random s
```

The error is of the order of 10^{-15}, which is usually good enough for our applications.

Defining functions and plotter

We will define all the functions that are needed in the assignment as follows :

```
#Defining all functions
cos3 = lambda x : np.cos(x)**3
sin3 = lambda x : np.sin(x)**3
gauss = lambda x : np.exp(-x**2/2)
coscos = lambda x : np.cos(20*x+5*np.cos(x))
sin5 = lambda x : np.sin(5*x)
modulated = lambda x : (1+0.1*np.cos(x))*np.cos(10*x)
```

```
#Dictionary of functions
func_dict = {'sin5':sin5,
'modul':modulated,
'cos^3' : cos3,
'sin^3' : sin3,
'coscos' : coscos,
'gauss' : gauss }
```

The plotter function is:

```
#Plotting function
def customplot(func,x,y,xlim,savename):
    plt.figure()

    plt.subplot(2,1,1)
    plt.plot(x,np.abs(y),lw=2)
    plt.xlim(-1*xlim,xlim)
    plt.ylabel(r"$|y|$")
    plt.title(f"Spectrum of %s"%func)
    plt.grid(True)

    plt.subplot(2,1,2)
    ii = np.where(np.abs(y)>1e-3)
    plt.plot(x[ii], np.angle(y[ii]),'ro',lw=2)
    plt.xlim(-1*xlim,xlim)
    plt.ylim(-5,5)
    plt.xlabel(r"$k$")
    plt.ylabel(r"Phase of $Y$")
    plt.grid(True)
    plt.savefig(savename)
```

DFT of various signals

We calculate the DFT of various signals by first sampling the function at certain points and then using the fft() function on them. The function to do this is

```
#Function to perform dft and plot spectrum
def perform_dft(func,N=512,steps = 513, r=4*np.pi, phase_limit=1e-3, xlim=40, w_lim=
t = np.linspace(-r,r,steps)[:-1] #Time range
y = func_dict[func](t) #Sampled function values
Y = fftshift(fft(y))/N  #Shifting freq
w = np.linspace(-w_lim,w_lim,steps)[:-1] #Frequency values to be plotted
```

```
customplot(func,w,Y,xlim,func+"DFT_spectrum.jpg")
```

To get the spectrum of any signal, we will now simply call the function using the arguments.

```
#DFT of various functions
perform_dft('sin5', xlim=10)
perform_dft('modul',xlim=40)
perform_dft('cos^3',xlim=15, steps= 129 , w_lim=16, N = 128)
perform_dft('sin^3',xlim=15)
perform_dft('coscos',xlim=40)
```

0.1 Spectrum of sin(5t)

We can write

$$sin(5t) = \frac{1}{2j}(e^{5jt} - e^{-5jt})$$

. Thus we expect peaks at $\omega = \pm 5$

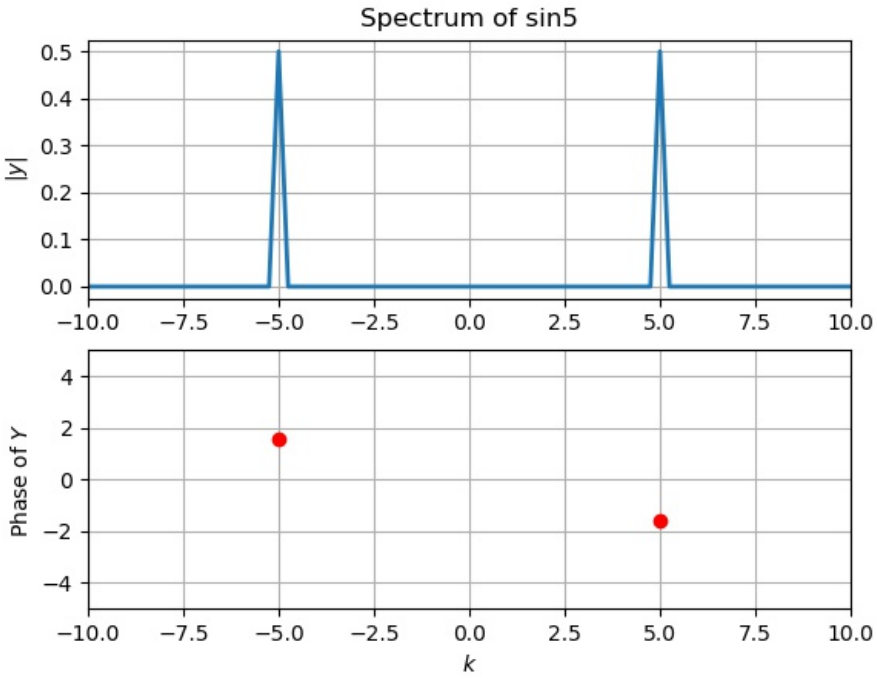

Figure 1: Sin(5t)

0.2 Amplitude modulated signal

Our modulated signal is $(1 + 0.1cos(t))cos(10t)$ which can be rewritten as,

$$(1+0.1cos(t))cos(10t) \;=\; \frac{1}{2}(e^{10jt}+e^{-10jt}) + 0.1\cdot\frac{1}{2}\cdot\frac{1}{2}(e^{11jt}+e^{-11jt}+e^{9jt}+e^{-9jt})$$

We observe that the frequencies present in the signal are $\omega = \pm 10, \omega = \pm 11 and \omega = \pm 9$. Thus we expect the DFT also to have non-zero magnitudes only at these frequencies. The magnitude has two large peaks at frequency

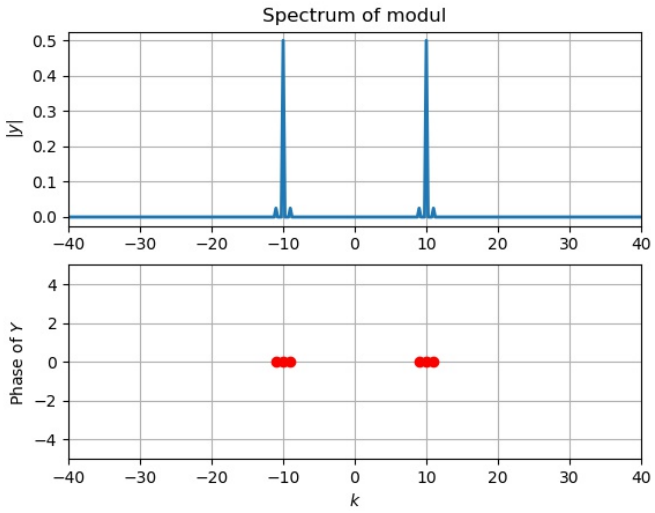

Figure 2: Amplituded modulated signal

of 10 corresponding to the carrier cosine wave.

0.3 Spectrum of $Sin^3(t)$ and $Cos^3(t)$

Breaking down both the functions we get,

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

Thus, here we would expect peaks at $\omega = \pm 1 \; and \; \pm 3$ with bigger peaks at $\omega = \pm 1$ Similarly,

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

Here too, we would expect peaks at $\omega = \pm 1 \; and \; \pm 3$ with bigger peaks at $\omega = \pm 1$

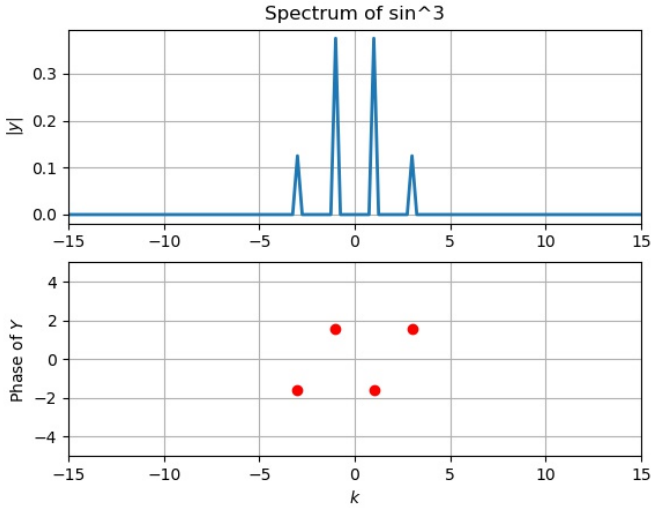

Figure 3: $sin^3(x)$

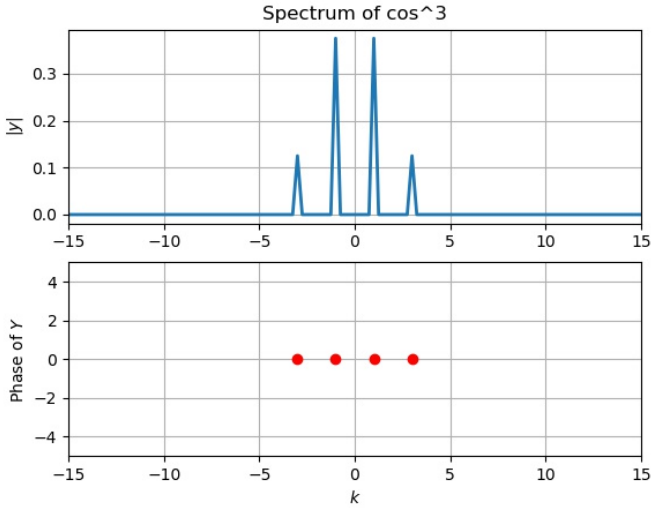

Figure 4: $cos^3(x)$

0.4 Frequency modulated Signal

The DFT of cos(20t + 5cos(t)) can be seen below:

There are many more sidebands compared to AM in the case of FM. Most of the energy of the signal is also present in these sidebands rather than the carrier band in the case of AM.

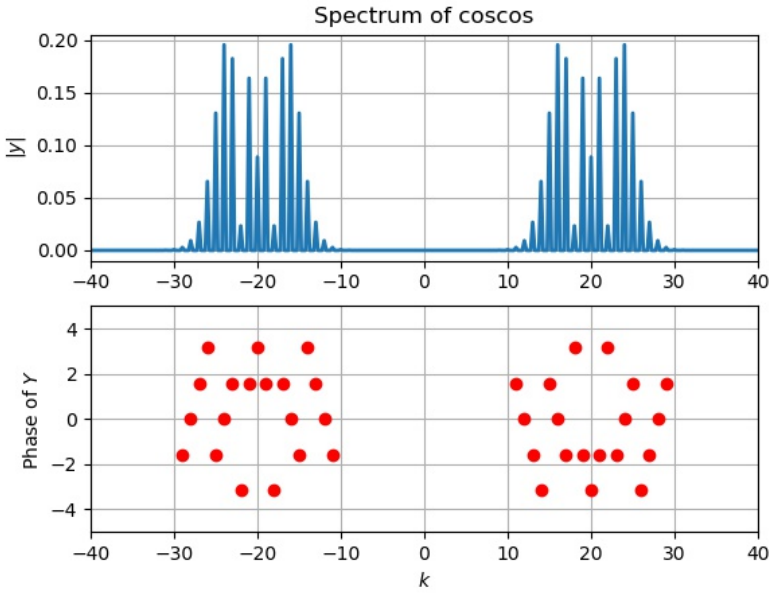

Figure 5: Frequency Modulated Signal

Gaussian Signal

A function to estimate the DFT of the gaussian is shown below. The function iteratively increases window size and sample number until the consecutive total absolute error between estimates reduces below a given threshold.

```
def perform_dft_gaussian(func,tolerance=1e-6,N=128):
T = 8*np.pi
Y_old = 0

while 1:

#Time resolution
dt = T/N
#Frequency resolution
dw = 2*np.pi/T

#Freq window size
W = N*dw

#Time samples
t = np.linspace(-T/2,T/2,N+1)[:-1]
#Freq samples
```

```
w = np.linspace(-W/2,W/2,N+1)[:-1]

y = gauss(t)

Y_new = dt/(2*np.pi) * fftshift(fft(ifftshift(y)))

error = sum(abs(Y_new[::2]) - Y_old)
Y_old = Y_new

if error < tolerance:
customplot(func,w,Y_new,5,"DFT_Gaussian.jpg")
print("Error in Gaussian case is {}".format(error))
return

T*=2
N*=2

perform_dft_gaussian('gauss')
```

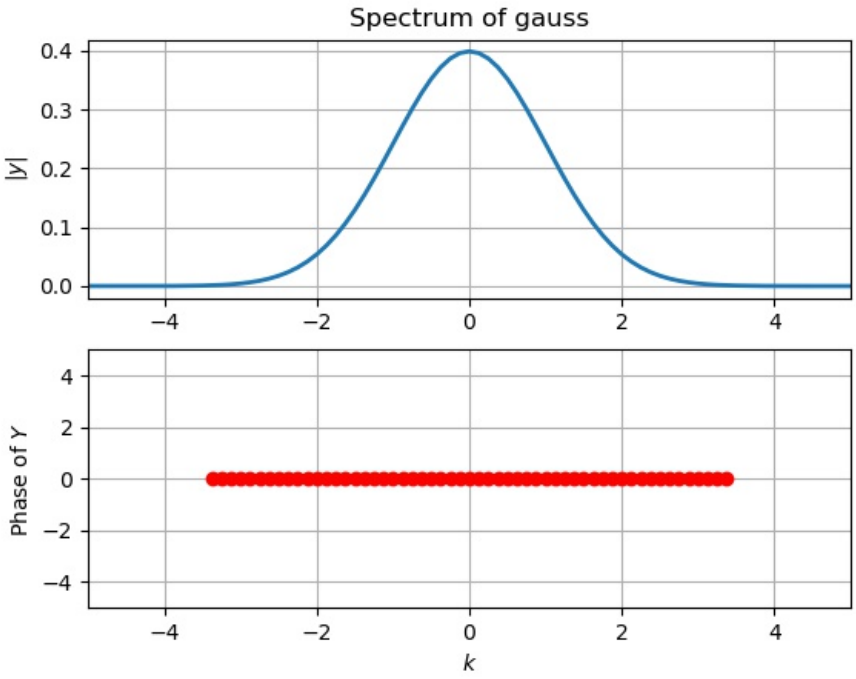

Figure 6: Frequency Modulated Signal

Observations

On the basis of the results obtained, we can make the following observations:

- In the first plot of sin(5x), there are 2 peaks at k = ± 5. This is natural since sin(5x) is a superposition of 2 complex exponentials of frequencies 5 and -5
- In the AM signal, we expect 3 peaks on each side of the x-axis, at $|k| =$ 9, 10, 11, which is same as what we can see in the plot. However, the peaks at k = 9 and k = 11 will be less dominant due to the Amplitude reduction factor of 0.1
- Since the cubes of sinusoids can be written as the superposition of a same frequency sinusoid and thrice the frequency sinusoid (and same for cosinusoids), that is the reason we can see peaks at k = ± 1 and k = ± 3 for f(x) = $sin^3(x)$ and f(x) = $cos^3(x)$
- For the frequency-modulated signal : Here, since the argument of the outer cosine is also oscillating we can find various frequency components, and as a result there are many peaks. Some of the side-bands have greater energy as compared to the peaks at k = ± 20.
- We ran an iteration till the error between between the actual Gaussian function and it's DFT was less than a threshold of 10^{-6}. Hence, we can verify the fact that the transform of a Gaussian distribution in timedomain is a Gaussian distribution in frequency domain since the DFT plot we find is similar to the original function.

Conclusion

Hence, we used the numpy.fft package to simulate the Discrete-Fourier transforms of some standard functions and carry out frequency component analysis.