EE2703 : Applied Programming Lab

G.Rohith Kumar
EE18B008

May 6, 2020

Assignment 9

Abstract

In this Assignment we implement DFT Spectra for non-periodic signals.We also try to understand what is windowing and what is unique about chirped signal.

Requirments:

Modules to be imported,

```
from pylab import *
import mpl_toolkits.mplot3d.axes3d as p3
```

DFT of non-periodic signals

for example, even though $sin(\sqrt{2}t)$ is a periodic function, the portion between -π and π is not the part that can be replicated to generate the function.There is discontinuity after each period.so, what we do is windowing .We use this to reduce the discontinuity to a large extent so that we can replicate that part in the periodic interval to generate the function.

Windowing

we multiply our function sequence f[n] by a "window" sequence w[n]. The window we will use is called the Hamming window.

$$w(n) = \begin{cases} 0.54 + 0.46cos(\pi n/(N-1)) & |n| \leq (N-1)/2 \\ 0 & else \end{cases}$$

$$g(n) = f(n)w(n)$$

The new spectrum is got by convolving the two fourier transforms:

$$G_k = \sum_{n=0}^{N-1} F_n W_{k-n}$$

Suppose f_n is a sinusoid. Then F_k has two spikes. But the two spikes are now smeared out by W_k. So we expect to get broader peaks. But what this also does is to suppress the jump at the edge of the window.

Other Requierments

Functions created for given tasks.

- *fft_wnd* - returns DFT of the given signal and the windowing option is decided by one of the arguments called window_flag.
- *get_data* - Plots both magnitude and phase plots of DFT of the signal.
- *estimator* - estimates where the peaks are from the DFT spectrum and also error between actual and calculated values

Code:

```
def get_data(w,Y,titlee,xlimit):
    figure()
    subplot(211)
    plot(w,abs(Y),lw=2)
    title(titlee)
    ylabel(r"$|Y|$",size=16)
    xlim([-xlimit,xlimit])
    grid(True)
    subplot(212)
    ii=where(abs(Y)>1e-3)
    '''  if(plotgo==1):
        plot(w[ii],angle(Y[ii]),'go',lw=2)
    else:
        plot(w,angle(Y),'ro',lw=2) '''
    plot(w,angle(Y),'ro',lw=2)
    xlabel(r"$omega$",size=16)
    ylabel(r"Phase of $Y$",size=16)
    xlim([-xlimit,xlimit])
    grid(True)
    show()

def fft_wnd(t,y,window_flag):
    n=arange(len(t))
    wnd=fftshift(0.54+0.46*cos(2*pi*n/(len(t)-1)))
    if(window_flag==1):
        y=y*wnd
        y[0]=0
        y=fftshift(y)
```

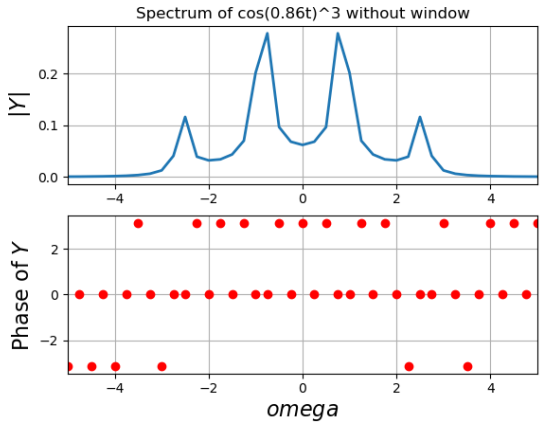
```
        Y=fftshift(fft(y))/len(y)
        return Y
    else:
        y[0]=0
        y=fftshift(y)
        Y=fftshift(fft(y))/len(y)
        return Y

def estimator(t,w,omega,delta,window_flag,noise_flag):
    if(noise_flag==1):
        y=cos(omega*t+delta)+0.1*randn(len(t3))
    else:
        y=cos(omega*t+delta)
    Y=fft_wnd(t,y,window_flag)
    ii = np.where(w>0)[0]
    ii = np.where((abs(Y) == max(abs(Y[ii]))))
    est_delta = abs(np.angle(Y[ii])[0])
    ii = np.where((abs(Y) > 3.5e-2) & (w >= 0))[0]
    est_omega = abs(Y[ii]*w[ii])
    est_omega = sum(est_omega)/(sum(abs(Y[ii])))
    # As peak is spread out, omega is estimated as the
    weighted average(centre of mass) of the broad area near the peak.
    get_data(w,Y,'Spectrum of $cos($'+str(omega)+'$t$'+'+'+str(delta)+'$)$'
    ,omega+4)
    print ('the Calculated delta is %.6f and the error in the
    calculated delta is %.6f' %(est_delta, abs(est_delta - delta)))
    print ('the Calculated omega is %.6f and the error in the
    calculated omega is %.6f' %(est_omega, abs(est_omega - omega)))
```

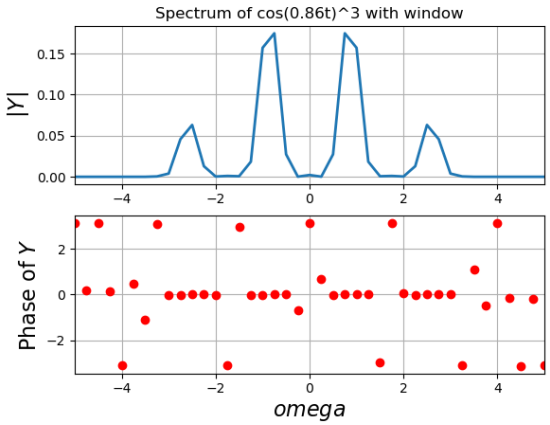
Q2:Spectrum of $cos^3(0.86t)$ with and without windowing

Code:

```
t1=linspace(-pi,pi,65)[:-1]
fmax1=1/(t1[1]-t1[0])
t2=linspace(-4*pi,4*pi,257)[:-1]
fmax2=1/(t2[1]-t2[0])
w1=linspace(-pi*fmax1,pi*fmax1,65)[:-1]
w2=linspace(-pi*fmax2,pi*fmax2,257)[:-1]
```

```
Y=fft_wnd(t2,cos(0.86*t2)**3,0)          # DFT of cos(0.86t)^3 without window
get_data(w2,Y,r"Spectrum of cos(0.86t)^3 without window",5)
Y=fft_wnd(t2,cos(0.86*t2)**3,1)          # DFT of cos(0.86t)^3 with window
get_data(w2,Y,r"Spectrum of cos(0.86t)^3 with window",5)
```





Q3,Q4:

Spectrum of $cos(\omega t + \delta)$:

Let us consider $\omega = 1.2$ and $\delta = 2.4$

Code:

```
t3=linspace(-pi,pi,129)[:-1]
fmax3=1/(t3[1]-t3[0])
w3=linspace(-pi*fmax3,pi*fmax3,len(t3)+1)[:-1]

omega=float(input("Enter your value of omega between 0.5 and 1.5"))
# Takes input for omega
delta=float(input("Enter your value of delta"))
# Takes input for delta
noise_flag=float(input("Enter 1 to add noise or else 0"))
# Takes input for noise addition

estimator(t3,w3,omega,delta,window_flag=1,noise_flag)
```

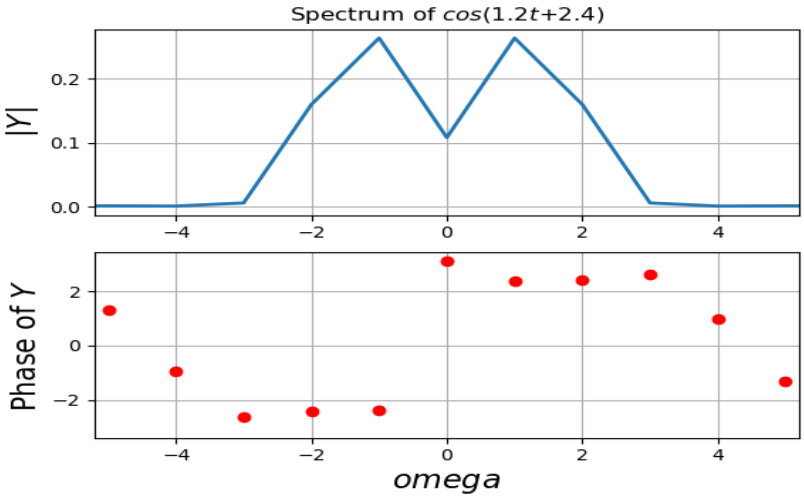


Figure 1: Without Noise Addition

The Calculated delta is 2.386148;the Error in the calculated delta is 0.013852
The Calculated omega is 1.096972;the Error in the calculated omega is 0.103028

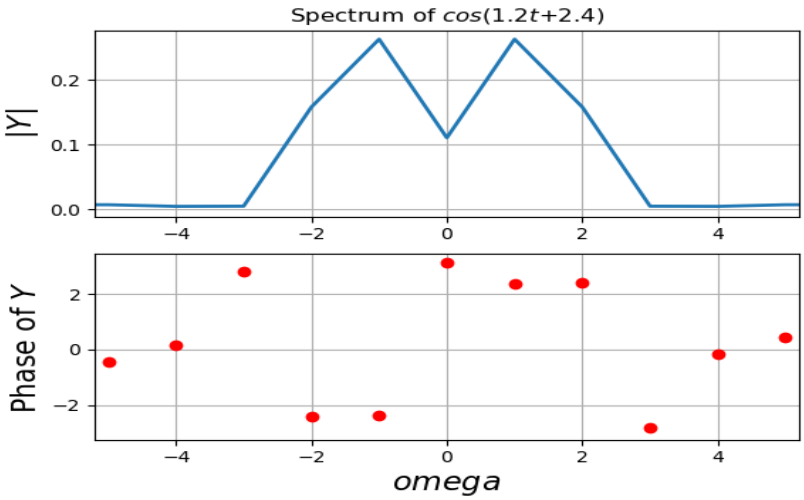


Figure 2: With Noise Addition

The Calculated delta is 2.397947
The Error in the calculated delta is 0.002053
The Calculated omega is 1.095837;the Error in the calculated omega is 0.104163

Q5:Spectrum of chirped signal $cos(16(1.5+t/2\pi)t)$

Code:

```
t4=linspace(-pi,pi,1025)[:-1]
fmax4=1/(t4[1]-t4[0])
w4=linspace(-pi*fmax4,pi*fmax4,len(t4)+1)[:-1]
y=cos(16*(1.5+(t4/(2*pi)))*t4)

Y=fft_wnd(t4,y,0)                    # without window
get_data(w4,Y,'spectrum of $cos(16(1.5+t/2*pi))t$',100)

Y=fft_wnd(t4,y,1)                    # with window
get_data(w4,Y,'spectrum of $cos(16(1.5+t/2*pi))t$',100)
```

The DFT of this signal is

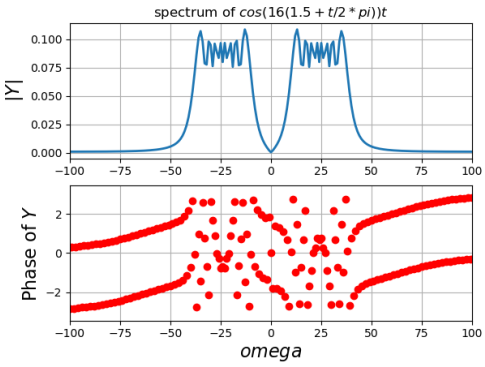


Figure 3: Without Window

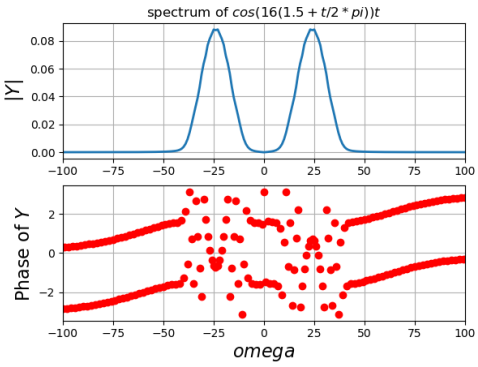


Figure 4: With Window

Q6:

Code:

```
w5=linspace(-pi*fmax4,pi*fmax4,65)[:-1]
t5=linspace(-pi,pi,17)[:-1]
tt,ww=meshgrid(t5,w5)
ns=arange(64)
wnd=fftshift(0.54+0.46*cos(2*pi*ns/63))
```

```
X=[fftshift(fft(fftshift(y[0:64]*wnd)))/64]
for i in range(1,16):
    X.append(fftshift(fft(fftshift(y[i*64:(i+1)*64]*wnd)))/64)
    # finding DFT's for each 64 samples of signal

Y=X[0]
for i in range(1,16):
    Y=c_[Y,X[i]]
    # Storing those 16 DFT sets as columns in 2D-array

figure(6)
ax=p3.Axes3D(figure(6))
title("chirped signal 3D plot")
ax.set_xlim3d(-80,80)
ax.set_xlabel(r"frequency")
ax.set_ylabel(r"time")
ax.set_zlabel(r"DFT Spectrum")
# Without this and the next line, the surface plot overflows
due to the setting of xlim.
surf = ax.plot_surface(ww,tt,abs(Y),rstride=1,cstride=1,cmap=cm.jet,
                linewidth=0,antialiased=False)
ww[ww>80]= nan
ww[ww<-80]= nan

# 3D surface plot of splitted DFTs vs time and frequency.
show()
```

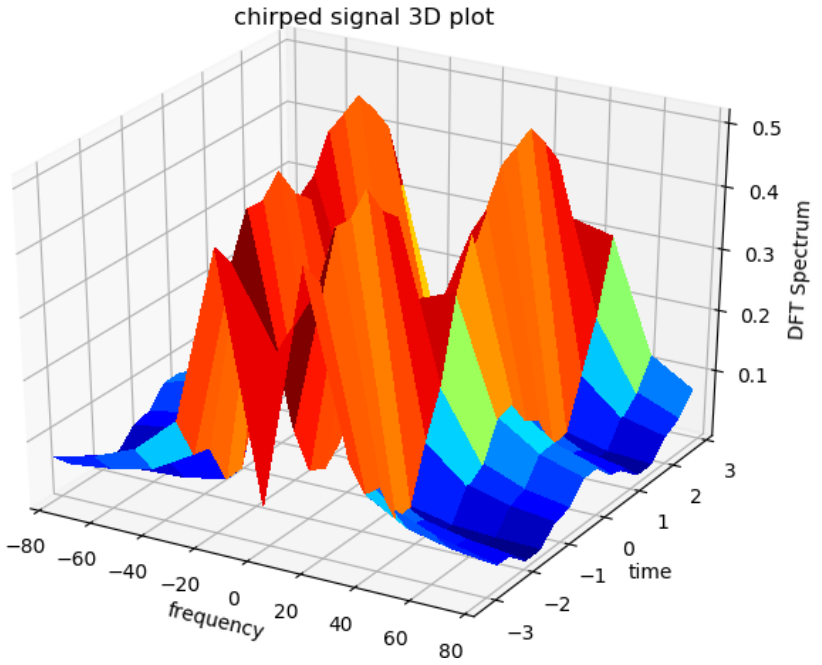


Figure 5: 3D Surface plot Without window

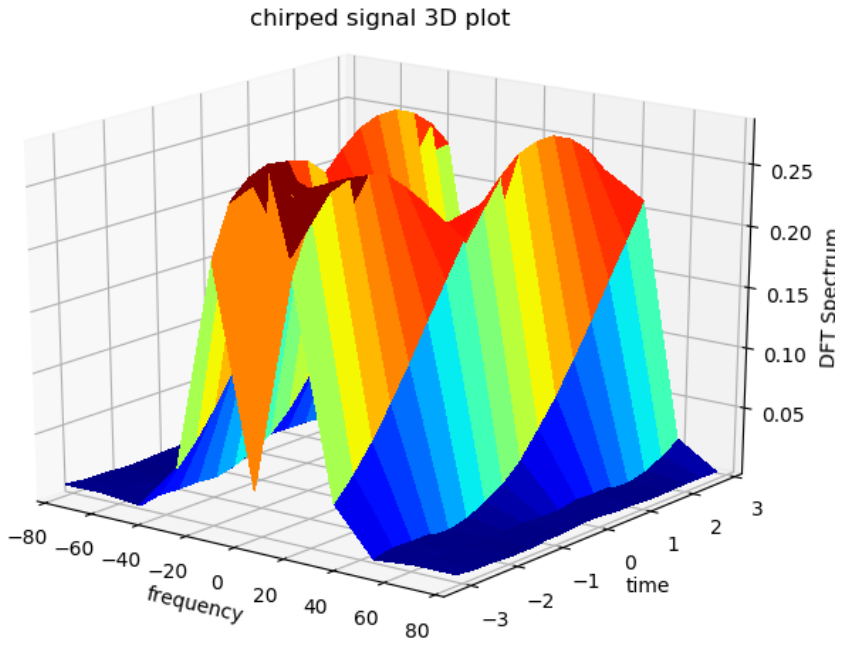


Figure 6: 3D Surface plot Without Noise Addition

Results

- For Q1, It is just an example and results are already mentioined in the PDF.
- For Q2,We can see broader peaks because of discontinuity.if you observe clearly the magnitude spectrum of function which is not windowed decays slowly and falls to zero at large frequencies and the spectrum of function which is windowed falls to zero quickly due to less discontinuity.
- For Q3 and Q4,as we gave input ω_o and δ, the program gives us the DFT spectrum and estimate the peaks from the spectrum.We are restricted to finite no of samples so, there is finite amount of resolution which results an error in estimated ω_o and δ.
- For Q5 and Q6,We plotted Magnitude and phase plots of DFT spectrum of the chirped signal and the 3D-surface plot of DFT spectrum when the vector is divided in to 64 samples wide and observed that the frequency changes with time.